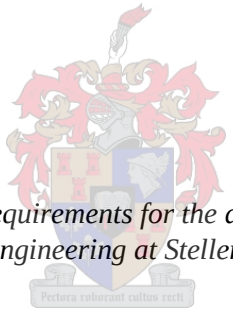
Investigation into a Smartphone Application for Household Water Data Management as a Tool to Improve Conservation Efforts

by

Bradley Vaughan Warren



Thesis presented in fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Civil Engineering at Stellenbosch University

Supervisor: Prof. HE Jacobs

March 2021

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2021

Abstract

Water demand management strategies are receiving increasing focus in literature for advancing water conservation. Various smart metering products are available on the market to help household users monitor consumption, however they are not widely implemented. By providing water users with access to smart meter data via a smartphone application, users would be empowered to monitor their household water usage and take steps to reduce consumption. Such an application has the potential to benefit water utilities by reducing overall demand on their resources and would provide a useful platform for communicating directly with users when implementing water restrictions. For this study, a literature review regarding water demand management and smart metering was undertaken, and an Android application with IOT capabilities was developed as a proof-of-concept. The application was able to receive data from a compatible water meter and vibration device via a web-based server and provide the user with important information regarding their water usage. Recommendations were given for further research and developments required in the field.

Acknowledgements

I would like to thank my parents who provided me with the opportunity to educate myself and undertake this research project. I would also like to thank my supervisor, Professor Jacobs, for his invaluable input and guidance. Lastly, I need to thank all my family and friends for supporting me at every step along the way, helping me stay on track.

“ When the well is dry, we know the worth of water ”

– Benjamin Franklin

Contents

Figures

Tables

Symbols

cm	centimeters
kL	kiloliters
kL/d	kiloliters per day
kL/m	kiloliters per month
L	liters
L/c/d	liters per capita per day
L/d	liters per day
L/s	liters per second
m	cubic meters
m^3/s	cubic meters per second
mAh	milliampere hour
mm	millimeters

Abbreviations and Acronyms

API	Application Programming Interface
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBT	Increasing Block Tariffs
IDE	Integrated Development Environment
iOS	iPhone Operating System
IOT	Internet of Things
JDK	Java Development Kit
JSON	JavaScript Object Notation
NFC	Near Field Communication
SDK	Software Development Kit
ToU	Time of Use
URL	Uniform Resource Locator
XML	Extensible Markup Language

1. Introduction

1.1. Background

Water is one of society's fundamental resources, critical for maintaining quality of life and sustaining a growing economy. Without water, life cannot be sustained beyond a few days, and inadequate access to water is known to contribute to the spread of disease. The World Health Organization (WHO) recommends a minimum of 20 litres per person per day in order to meet basic consumption and hygiene requirements, and asserts that upwards of 100 litres per person per day is necessary in order to completely satisfy all needs (WHO, 2003). The recent drought crisis in Cape Town thrust the city into national and international focus, forcing authorities to confront the possibility of the severe social and economic impacts that would be experienced if the taps were to literally run dry – a scenario which authorities dubbed "Day Zero". During the peak of the restrictions the municipality mandated its citizens to limit their usage to less than 87 litres per person daily, with a maximum monthly ceiling of 10 kL per household.

In the wake of the crisis there appears to be a greater public appreciation of the value of water as a scarce resource, not only in Cape Town, but throughout South Africa. Many of the country's metropolitans, including Gauteng, eThekwini, and Nelson Mandela Bay, as well as numerous smaller municipalities were, and still are, at stress points with regards to their water resources. Over the summer period of 2016/2017, seven out of South Africa's eight metropolitans implemented water restrictions due to low dam levels (Ziervogel, 2019). The external factors driving water stress – drought, climate change, population growth, and increased population density – are not unique to South Africa, these are global issues and expected to worsen in the coming decades. As a result, there is an increased focus on various demand management strategies such as water accounting, loss control, pricing structures and education. The success of these strategies is dependent on the availability of reliable feedback data that can be interpreted to help utilities assess outcomes and guide further decision making.

Meaningful data provided in real time would also be a powerful tool for consumers, especially if tied to billing outcomes, to help make informed decisions regarding their water usage and potential savings. Whilst improvements in smartphone technology have provided an opportunity to transform household water management, the implementation thereof has so far been constrained by the availability and transmission of high-resolution data. Numerous commercial devices are available for recording flow rate and capable of communicating with smartphones via a mobile network or Wi-Fi network, however such devices typically require costly plumbing installations and the overall product may be expensive - or perceived as expensive - given the relatively low cost of water. This is expected to change in future, as technology becomes increasingly affordable and the era of cheap water begins to fade.

1.2. Terminology

In order to remain consistent throughout this thesis, the following terms are to be interpreted as follows:

1.2.1. Back-End

Back-end refers to the hardware and software of a computer application or a program's code which allows the application to operate but cannot be accessed by a user. The layer above, which the user interacts with directly, is referred to as the front-end.

1.2.2. Elasticity

Jacobs (2008) defined "elasticity" as a standardized measure of one (dependent) variable to changes in another (independent) variable, which the author noted pertains to the field of economics, but is commonly used in the water field as well. The definition of "elasticity of demand" used in this thesis is as follows: the percentage change in water demand associated with a percentage increase in another variable value.

1.2.3. End Point

The end point or end use is defined as the points around the home at which water is extracted from the distribution system. Examples of end points may include a tap, washing machine, shower, or toilet.

1.2.4. Frequency

Frequency in the context of this paper refers to the specified time intervals at which water use is recorded by the water meter for the purposes of feedback communication, e.g. 15 seconds between recorded water meter pulses.

1.2.5. Front-End

The front-end a computer application or program code refers to the hardware and software which is part of the user interface. The user interacts with the front-end directly, as opposed to the back-end, which allows the application to function but is inaccessible to the user.

1.2.6. Household Size

The term "household size" in this thesis describes the number of people living on a stand, or occupancy rate, and is measured by number of people per household (PPH).

1.2.7. Mobile Application

A mobile application, also referred to as an 'app' in common parlance, is a type of software designed specifically for use on mobile phones, typically a smartphone with access to the internet.

1.2.8. Mobile Operating System

A mobile operating system in this thesis refers to operating systems for mobile phones and tablets. Some computers such as laptops may be mobile, however these devices typically have distinct hardware and software features.

1.2.9. Smart Metering

A water meter is a device which has the ability to measure and record the volume or velocity of water flowing through the meter itself. A smart water meter has the important distinction of being able to communicate with a computer in order to facilitate data logging in real time or near real-time. Beal and Flynn (2014) defined smart metering as "the integration of meter data into business systems and the sharing of information with the customer", and this definition was adopted for this thesis.

1.2.10. Smartphone

A smartphone is defined as a mobile phone with enhanced features beyond making and receiving calls and can perform many of the functions of a computer. Typically, a smartphone may have a touchscreen interface, internet access, and an operating system capable of running downloaded applications.

1.2.11. Software Development Kit

A software development kit (SDK) is a collection of tools made available in one installable package, easing the creation of applications by providing a compiler, debugger, and a software framework. For example, Android Studio is the official SDK for developing Android applications, providing developers with a platform containing all the tools necessary for building and testing their application.

1.2.12. WaterWise

WaterWise is the name of the smartphone application developed for this project and is publicly available for download on Google Play Store.

1.3. Problem Statement

Various existing products are available on the market for users who wish to monitor their water usage and costs. The problem is that such products are relatively expensive, especially when compared to the general cost of water – and are not widely implemented in South Africa. Retrofitted flow monitoring devices and smartphone-based applications may prove to be a valuable tool for helping water users to better manage their water use and to develop a realistic perception of associated costs in real-time or near real-time.

1.4. Aim and Objectives

The aim of this research project was to investigate available smartphone tools for household water management, and to develop a smartphone-based application as a proof-of-concept, for providing a user with high-resolution water use data in real time.

The objectives of this research study were as follows:

- Review existing knowledge of smart metering systems
- Review existing literature regarding demand side management
- Develop a smartphone application as a proof-of-concept. The application should be capable of receiving data, in appropriate format, from connected flow monitoring devices, and present information via a user-friendly interface.
- Compile a baseline water use record based on actual flow recording
- Analyse simulated data to demonstrate the application's capabilities for providing the user with meaningful data and estimating potential savings.
- Compile a set of typical water tariffs
- Investigate water pricing structures which water utilities could employ as a means of managing water demand.

1.5. Motivation

It has been suggested in various studies that household owner's perceptions of their water usage are generally not well-matched with their actual usage. One of the key issues in South Africa, and throughout most of the world, is that meter reading is a resource and time intensive procedure. Smartphones are now ubiquitous accessories in modern society and, when combined with smart meters, could provide society with a tool to conserve water, which is increasingly recognized as a scarce resource. From the perspective of the water user, a meter-linked smartphone application would provide access to meaningful and detailed feedback regarding water usage in real-time or near real-time. By implementing local tariff structures, the application would also be able to track the provisional water bill throughout the billing cycle, and could notify the user of important events such as a potential leak or when the user has progressed into a higher tariff tier (in cases where increasing block prices are implemented). From the perspective of water suppliers, a smartphone application has the potential to automate the water meter reading and billing process and detect post-meter leaks. In the long term, adapting smart metering would provide insightful data for designing water networks. It is common practice in industry to design water networks to meet peak hourly demands (PHD), which are generally obtained by applying peak factors to the average annual daily demand (AADD) and are therefore potentially overdesigned. High resolution data collected from smart meters may provide a more reliable means of estimating the maximum demand when designing infrastructure.

1.6. Methodology

Applied research methods were used in this project to develop a product which household consumers could use as a consumption management tool. Evaluation research was undertaken to establish an overview of existing products on the market, and methods which could be implemented to reduce consumer water demand. To this end, a literature review was performed covering the following pertinent topics: factors which influence water demand, demand side management strategies, consumer behavioural interventions, the internet of things concept, and existing mobile phone applications implemented in the water supply sector. This was followed by research and development of smartphone application which would receive water usage data in real-time. Various hardware and software tools were considered for implementation prior to development. An application with a user-friendly interface was then developed. The application was capable of receiving various forms of data which is presented to the user in tabular and graphical formats. Primary and secondary data was analysed using Microsoft Excel. The analysed data was then used to test the application and demonstrate its capabilities. The findings of this research project are then discussed in the conclusion.

1.7. Scope and Limitations

This research focused on strategies which to achieve household water conservation, with particular focus on the role of smart metering and smartphone applications. To this end, an Android smartphone application was developed as a proof-of-concept to demonstrate the potential for information technologies to alert users of their water usage and drive water saving behaviour.

A simulated smart meter dataset was developed to represent the water usage pattern of a typical middle-income household spanning a 12-month period. The simulated data was then used in the developed application in lieu of live data in order to demonstrate the application's capabilities and potential for future implementation.

2. Literature Review

2.1. Overview

This literature review is structured to address the following topics: observations from the most recent Cape Town drought, factors influencing water use, demand management strategies, analysis of water end-use in households, methods of recording household water data, internet of things, smart metering, and the use of mobile phone technology to provide access to data in the field of water service provision.

2.2. The Cape Town Drought and Future Outlook

In early 2018, Cape Town was in the midst an extreme prolonged drought and at risk of becoming one of the first major metropolitan areas in the world to run out of water – or "Day Zero", as it were. As of May 2018 the city's total dam storage levels in the supply system were at a critical 21%, and the city narrowly avoided the crisis, owing to good early winter rains proceeded by near-average overall rains over the duration of the wet season, raising the levels to approximately 70%. There is ongoing debate regarding the contributing role that poor management and planning may have played in the crisis, however rainfall records analysed by Burls et al. (2019) indicate that the dam levels in the city dropped in reaction to an extreme 3-year drought.

The southwestern Cape, including Cape Town, receives the majority of its rainfall in the winter months from June to August as a result of cold fronts associated with mid-latitude cyclones, which are frequent in this period. The supply dams for the city are located in nearby mountains, where orographic effects enhance precipitation, up to 2000mm per annum at some stations. It is understood that the continental anticyclone - forming part of the Hadley cell - in the interior prevents frontal rainfall from extending towards the north and west, resulting in the Namib and Karoo deserts, respectively (Reason et al., 2002). Furthermore, a study by Sousa et al. (2018) found that in years when the Hadley cell is anomalously strong, or located further south, mid-latitude cyclones tend to track poleward. The associated cold fronts are generally weakened and steered away from Cape Town, resulting in reduced rainfall intensity. Year on year rainfall variability in the region results from differences in the number of cold fronts making landfall and the intensity thereof. Therefore, there exists a relationship between the strength and relative positioning of the Hadley cell, and the annual rainfall in southwestern Cape.

In order to assess the severity of the drought Burls et al. (2019) analysed two sets of rainfall data. The first was a regional set from the mountainous areas east of the city, which was considered to be representative of the rainfall contributing to the city's dams. The second set consisted of centennial records generally laying outside the eastern mountainous region but extended considerably further back in time. A long-term decline in rainfall days has been observed in the region, where a 'rainfall day' is defined as a day when any of the stations in the dataset recorded non-zero precipitation. However, this decline has been somewhat masked by fluctuations in rainfall intensity. The most recent drought was found to be

unprecedented in the centennial records and driven by the long-term decline in rainfall days as well as a more recent decline in rainfall intensity. Analysis of the regional data did not indicate a significant decline in the frequency of cold fronts over the last 40 years, but rather that there had been a general decline in the duration of the rainfall events associated with cold fronts. Additionally, it was found that long-term decline in rainfall days has occurred in tandem with a poleward trend in Southern Hemisphere storms and Hadley cell expansion. As a result, continued drying is predicted for Cape Town and surrounding regions in future, and droughts of similar severity are likely to reoccur.

Given the future outlook, it is clear that improved water resource management strategies must be considered in order to mitigate the potential consequences of a severe water shortage in the city. A smartphone application providing a user with real-time water-use information could prove to be a powerful tool for empowering users to manage their water consumption, especially in times of crises.

2.3. Household Water End-Use

Determining the water use patterns of specific household end-uses (fixtures and appliances) is useful for demand monitoring and for deriving effective demand management strategies. The characterization of household level end-usage has therefore been investigated in numerous studies worldwide, as summarized in Table 1. The results vary greatly across differing nations, local climate and household demographics. Generally showering and outdoor use have been shown to be the two greatest contributors to household usage with washing machines, taps, and toilets also contributing significantly. Dishwashers, bathtubs, and leakage losses have been found to have minimal impact on overall usage (Sterne, 2019).

Table 1: Household water usage characteristics from various studies (Sterne, 2019)

Source	Shower (%)	Tap (%)	Toilet (%)	Bathtub (%)	Dish-washer (%)	Washing Machine (%)	Outdoor (%)	Leak (%)	Total (L/c/d)
(Willis et al., 2013)	33.0	17.0	13.0	4.0	1.0	19.0	12.0	1.0	157.2
(Loh & Coghlan, 2003)	15.0	7.0	10.0	-	-	13.0	54.0	1.0	335.0
(Roberts, 2005)	22.0	12.0	13.0	2.0	1.0	19.0	25.0	6.0	226.2
(Heinrich, 2007)	27.0	14.0	19.0	3.0	1.0	24.0	8.0	4.0	168.1
(Mayer et al., 1999)	6.8	6.3	10.8	0.7	0.6	8.7	58.7	5.5	650.3
(Willis et al., 2011)	31.0	17.0	14.0	4.0	1.0	20.0	12.0	1.0	152.3

(Beal et al., 2013)	29.5	19.0	16.5	1.0	2.0	21.0	5.0	6.0	145.3
(Gurung et al., 2015)	35.4	16.6	18.3	1.5	1.7	22.1	-	4.4	160.0
(Carragher et al., 2012)	29.5	19.3	17.2	1.1	1.6	21.6	4.2	5.6	132.6
Average (%)	**25.1**	**13.8**	**14.3**	**1.8**	**1.1**	**18.3**	**22.1**	**3.5**	**-**
Average (L/c/d)	**59.3**	**32.6**	**33.8**	**4.3**	**2.6**	**43.2**	**52.2**	**8.3**	**236.3**

The results summarized in Table 1 show that shower usage, generally, is the largest contributor to overall household water usage, with taps and flushing of toilets also contributing significantly. These three end-uses combined contribute to 53.2% of overall household water usage on average in the studies reviewed and suggests that utilities could achieve considerable water savings simply by encouraging users to decrease the frequency and duration of these events where applicable, or by incentivizing the use of water efficient devices such as retrofitted shower nozzles or dual cisterns with a decreased flush volume.

2.4. Factors Influencing Water Usage

Residential water consumption has been shown to be influenced by numerous factors such as seasonal changes, water consumption feedback (e.g. via water meters), restriction levels, and education. However, the degree to which these factors influence consumption is country and location specific, owing to differences in: community attitudes and behaviours; efficiency of appliances e.g. water efficient shower heads; environmental conditions; water pricing structures; government restriction regimes; and intensity of conservation messages. Willis et al. (2013) argue that all such factors must be considered when designing demand management interventions.

Factors which influence water consumption are said to have an elasticity of demand, e, which is defined as elastic when $|e| > 1.0$, indicating that demand will change in a greater proportion than the independent variable. When $0 < |e| < 1.0$ the demand is said to be inelastic and will change in a smaller proportion than the independent variable (Jacobs, 2008). The following variables are known to significantly influence water demand and will be briefly discussed in sections to follow:

- Price
- Household size.
- Weather
- Water supply pressure
- Post-meter leakage.

Thereafter, the effects of consumer behaviour are discussed in this chapter as a precursor to Section 2.5, in which behavioural interventions are investigated as a strategy for managing water demand.

2.4.1. Price

Price of water may be regarded as the most common elasticity and was one of the first strategies to be recognized for managing water demand. Price and demand are inversely related, yet price elasticity is relatively inelastic, meaning that increasing the cost of water will result in decreased demand. However, at lower pricing there is a limit on the amount of water which anyone would use, and similarly there is a minimum quantity which anyone would require regardless of the price (Jacobs, 2008). Water pricing does not generally have a major influence on water consumption at a domestic level, which may be due of its relatively low cost in comparison to other household expenses.

Jansen and Schultz (2006) conducted a study on 275 households with varying demographics in the City of Cape Town and found water demand to be price inelastic, with varying results between low-income and high-income households. The results from the study found that a 10% increase in water price was associated with a 9.7% decrease in water consumption in high-income households, giving an elasticity of 0.97. Low-income households were found to be generally unresponsive to pricing.

Veck and Bill (2000) conducted a study across different income groups in Alberton and Thokoza areas in Gauteng which aimed to assess price elasticity by means of a contingent evaluation approach. Results from the study yielded a price elasticity of -0.17 for the two areas combined. The same authors used a macro-econometric model to determine the price elasticity for the Alberton area, and a value of -0.73 was determined.

2.4.2. Household Size

The size of the household, measured by the number of people per household (PPH), is one of the most significant variables influencing total water consumption at residences and one of the most common parameters included in forecasting models. A study by Makki et al. (2013) suggests that higher occupancy rates will result in higher water demand.

Furthermore, household size is also known to influence per capita consumption. Smith (2010) undertook a study in Pietermaritzburg, South Africa investigating the influence of household size on overall consumption and individual consumption, with results for various household sizes summarized in Table 2. The table shows that an increase in household size results in increased overall demand, but the individual demand will decrease as household size increases.

Table 2: Household size and water demand (Smith, 2010)

Household size	Number of households	Consumption per household		Consumption per capita	
		Monthly (kL/m)	Daily (L/d)	Monthly (kL/m)	Daily (L/c/d)
1	8	4.4	147	4.4	147
2	17	8.8	283	4.4	148
3	63	12.5	417	4.2	139
4	59	15.3	512	3.8	128
5	35	17.9	596	3.6	119
6	12	18.5	617	3.1	103
Total	194	14.1	467	3.9	131

2.4.3. Weather

A large number of weather variables can be found in water demand models in literature, however only temperature and rainfall are commonly used. Higher rainfall is associated with a decrease in demand, while higher temperature increases demand, mainly as a result of increased outdoor usage (Jacobs, 2008). Similarly, a study by Praskievicz & Chang (2013) in Seoul, South Korea, found that higher maximum temperatures would result in higher household water demand.

Other weather-related variables reported to influence consumption include the number of daylight hours per day, wind speed, number of rainy days per month, and relative humidity.

2.4.4. Pressure

Pressure in a distribution network has a positive correlation to system losses as a result of leaks, however only post-meter water use is of interest to this study. Certain leaks and water end-use components are influenced by pressure and are therefore deemed pressure dependent, other leaks and end-uses are not influenced by pressure and are thus pressure independent (Mckenzie and Lambert, 2002). For example, a perforation in an underground pipe would result in pressure dependent leakage, whilst the leakage rate on a toilet cistern flush valve would not be linked to the supply pressure. The following are examples of pressure dependent end-use components in households: garden irrigation, showering, toilet/flushing mechanisms without cisterns, and water used directly from taps such as brushing teeth or washing hands. The water demand from such components would be reduced if supply pressure were reduced, but the relationship between water pressure and overall household consumption would vary depending on the presence of leaks and/or water efficient fittings, which has been shown by Willis et al. (2013) to have a significant impact on water use efficiency.

Bamezai and Lessick (2003) investigated the relationship between supply water pressure and residential demand, and their findings indicated that pressure reduction can induce a decrease in demand in

residences, especially where garden irrigation is common. In one of the areas tested, the average water supply pressure was reduced by 17.6% and resulted in a 1.9% decrease in domestic demand, which is equivalent to an elasticity of demand to pressure of 0.11. A decrease in consumption of 4.1% was reported in properties with larger gardens. However, in another test area a 6% reduction in supply pressure had a negligible effect on the domestic demand.

2.4.5. Post-Meter Leakage

Post-meter leakage has generally been found to be a function of supply pressure and will increase as pressure increases. Couvelis and van Zyl (2012) conducted a study to measure the incidence and flow rates of on-site leakage in suburban properties. The study found on-site leakage to occur on 17%, 28%, and 67% of middle- and high-income properties in Cape Town, Mangaung and Johannesburg respectively, suggesting significant variance of on-site leakage in South Africa. Various international studies, summarized in Table 1, have estimated on-site leakage to constitute between 1.0% and 5.6% of overall household use. Studies in the USA and Canada by DeOreo et al. (1996) and Mayer et al. (1999) have reported leakage contributing up to 10.3% of household demand.

Meyer (2018) conducted a trial on three separate district metered areas (DMA) in Pretoria, South Africa, implementing a series of decremental pressure adjustments in order to analyse the pressure-demand relationship. A DMA refers to a discrete portion of the WDS with a permanent, defined boundary, for which all inlet and outlet water pipes are metered. All three sites reported a positive, nearly linear relationship between supply pressure and consumer demand. The trail ran for 25 weeks, and the pressure decreased every 14 days, giving 11 decrements in total. Two significant factors generally influencing the elasticity of demand to pressure were identified, namely: the presence of on-site leakage and the presence of household pressure reducing valves (PRVs). Investigation into the trial sites indicated that on-site leakage was one of the main factors behind the observed pressure-demand relationship. When on-site leakage was excluded, the impact of pressure on demand was found to generally decrease. The power regression model used indicated that the elasticity of demand to pressure was in the range of 0.15 to 0.30 where on-site leakage was included, and in the range of 0.05 to 0.25 when on-site leakage was excluded.

Other local studies suggest that post-meter leakage is typically higher in South Africa than in developed countries. A study by Lugoma et al. (2012), on a sample of 182 properties in Johannesburg found that leakage accounted for approximately 25% of the monthly demand, with an average leakage rate of 12 - 29 kL/month per property. A similar study by Ncube and Taigbenu (2016) analysed a sample of 408 properties and examined the water demand profiles and prevalence of on-site leakage in Johannesburg. Residential, multi residential, business, and public use properties were randomly selected, and flows were recorded over 1-week periods. 63% of the sampled properties were found to have leakage, and it was found that night-time losses accounted for around 9 – 17% of the total water consumption. The researchers estimated monthly leakages for residential, multi residential, and business connections in Johannesburg

to be in the range of 11 – 41 kL/month per property. The reviewed literature suggests there is significant scope for water savings through simple leak management.

2.4.6. Consumer Behaviour

Booysen et al. (2019) describe the sequence of events leading up to Cape Town's purported "Day Zero" crisis in 2017 and 2018 connecting key events to the changes in behaviour of a small sample of users with smart meters measuring both hot and cold water usage. By implementing Talkwalker and Google Trends – a social media analysis tool and search engine term analysis tool respectively – the researchers were able to analyse trends in public responses to government announcements, and found that the biggest responses were not observed when restrictions and tariff increases were imposed, but after a three-phased disaster plan was announced along with a warning of anticipated disastrous scenarios. The smart meter data and billing data from the City of Cape Town indicate that a dramatically revised water consumption pattern was achieved in a relatively short period of time. The results suggested that the threat of waterless taps had a greater effect on consumption patterns than implementing level restrictions, and that inciting a general sense of fear amongst the public appears to have been the most successful intervention in altering water usage behaviour.

As a result of increased frequency and severity of droughts in Australia, there has been a focus over the last decade to investigating various Water Demand Management (WDM) strategies. One such study by Willis et al. (2011) found that households which were environmentally conscious and held positive attitudes towards water conservation had a significantly lower water consumption in comparison to those who were only moderately concerned with water conservation and the environment. This suggests that awareness campaigns, especially during periods of water stress, should generally be expected to lower water consumption. However, water used for drinking, cooking, and maintaining basic hygiene, for example, can be considered to be a bare necessity. Therefore, 'fear mongering' strategies will only be effective if users are able to reduce non-critical usage (e.g. outdoor usage), and it follows that repeated campaigns are unlikely to induce additional demand reductions if users perceive the usage to be at the bare minimum level.

Numerous studies have indicated that there often tends to be a disparity between householders' perceptions of their water usage and their actual usage. Beal et al. (2013) tested this hypothesis in a sample of 222 homes in South-East Queensland fitted with Actaris CTS-5 smart water meters. All members of these household were tasked with recording all their water use activities in a diary for a 7-day period. The respondents were clustered as either 'high', 'medium', or 'low' in terms of their self-reported water usage, and the researchers then analysed the high-resolution end-use data against the matched self-reported responses of the household water-use survey. The results indicated that self-nominated high-water users use less water than both the self-reported medium and low water users. The researchers observed socio-economic and psycho-social differences between the self-reported high, medium, and low water users and

noted that this type of profiling could be used to formulate targeted water policies geared towards water reduction in future. Overall, the study suggests that detailed water usage feedback would be an effective tool for consumers in effecting demand side management.

In a similar study by Stewart et al. (2013) involving 151 households, showers were fitted with a high-resolution smart meter, an Actaris CTS-5, as well as a visual display monitor which would provide feedback by indicating how long the user had been in the shower. The authors of the study hypothesised that by providing users with this feedback, users would become more conscious of their water usage and thereby encourage a reduction in use. The study showed an average initial reduction of 27% in water-use volume shortly after implementation of the monitor, confirming that making consumers aware of their usage has the potential to induce significant behavioural impacts. However, 4 months after the trial concluded and the monitors had been removed, the researchers found that the consumption reductions had diminished, and the mean showering volumes had reverted back to pre-intervention levels. Interestingly, based on a post hoc survey also undertaken 4 months after the initial trial, the majority of the participants of the study (88.2%) believed that the interventions had had a lasting impact on their shower use behaviour - highlighting the disparity between perceived and actual usage by consumers.

In a survey conducted by Datta et al. (2015) in Belén, Costa Rica, households were presented with questionnaires in order to better understand consumers perception of their water usage behaviour to help identify interventions which could be implemented to reduce overall consumption. The researchers noted that although there appeared to be a general consensus regarding the importance of water conservation, few residents believed that they *themselves* needed to use less water. This was found to be partially due to the fact that consumers did not know how much water they used as individuals, but also because residents lacked a benchmark against which their own consumption could be compared. The researchers in this study also found that very few users could identify concrete steps which would help them reduce their household consumption. These findings led the researchers to postulate that an intervention which allowed consumers to benchmark against their peers would be useful. This hypothesis was then tested in a randomized control trial which is reviewed in Section 2.5.1.

2.5. Water Demand Management

Fresh water supply is widely viewed as one of the most critical issues confronting policy makers in the twenty first century, especially in the urban jurisdictions of developing countries where there are limits on the ability to increase supply. Available literature suggests that demand management will be crucial for managing water resources in future. More specifically, the demand management of household users is of particular importance in the urban context where households constitute the bulk of water consumption (Ferraro and Price, 2013). Datta et al. (2015) identified two prominent demand management strategies, namely: pecuniary approaches and communication approaches. The former involves increasing

prices or taxes, and the latter aims to foster awareness of water scarcity in order to encourage water conservation behaviour.

2.5.1. Communication Approaches

A number of researchers have argued that applied behavioural economics provides a useful framework for designing alternative water-saving interventions, without necessarily having to resort to price-based strategies (Datta et al. 2015; Ferraro and Price, 2013; Brick et al., 2017). Such behavioural interventions have already proven to be effective in the energy sector, despite energy demand being considered to be extremely inelastic. Typically, these interventions involve providing the customer with feedback comparing their usage to their peers or the average user in their area. Allcott (2011) undertook a series of experiments with a power utility company whereby 'social norm' interventions were put in place and found that these measures were able to reduce consumption by up to 2.0%.

In order to evaluate the effectiveness of various behavioural interventions, or 'nudges', on water consumption, Datta et al. (2015) performed a randomized control trial in Belén, Costa Rica, Costa Rica. The researchers undertook a survey amongst focus groups in the area regarding their perception of their water use behaviour, as discussed in Section 2.4.6, and concluded that water consumption in households might be reduced if interventions were implemented that would allow users to compare their usage to that of their peers. A trial was then set up in order to consider the effects of three different 'nudges' designed to be technologically undemanding and easily replicable. A total of 5626 households were studied for the experiment and were randomly assorted into one of four groups - three which were subjected to interventions and then the fourth being a control group in which no interventions were put in place. In the first intervention ('neighbourhood comparison' or 'social norm'), the treatment households were given monthly feedback comparing their consumption to that of the average household in their neighbourhood. In the second comparison group, 'city comparison' was implemented, whereby users were given monthly feedback comparing their consumption to that of the average household in their city. The final intervention involved goal-setting and planning ('plan-making'), whereby clear, specific goals were set regarding *how much* water the users intended to save as well as *how*, followed by monthly feedback regarding their own relative consumption and reduction. Households in the randomized control group received their monthly water bill and no interventions were applied. The results of the trial, which lasted two months, indicate that two of the three interventions, 'neighbourhood comparison' and 'plan-making', led to statistically significant water use reduction, while the third intervention, 'city comparison', had very little impact on water consumption. The 'neighbourhood comparison' and 'plan-making' intervention led to estimated reductions of between 3.7% to 5.6%, and 3.4% to 5.5%, respectively, when compared to consumption data from the same period from the previous year. Interestingly, the researchers observed that the 'neighbourhood comparison' intervention was found to be most effective among high-consumption households, and the benefits of the 'plan-making' intervention were found to be greatest among

households which had a relatively low baseline consumption. Understanding these heterogenous effects across the income spectrum is important from a water utility's point of view, since the various feedback strategies can then be implemented towards more responsive groups. Furthermore, the observation also implies that broad-scale implementation of the 'neighbourhood comparison' (social norm) interventions could be a powerful tool for reducing overall consumption, since the highest water consumers are likely to have the biggest effect on overall water usage, especially in affluent areas.

Brick et al. (2017) undertook a similar study involving around 400 000 residential homes Cape Town, evaluating the ability of behavioural nudges to effect conservation during periods of water austerity. The researchers identified 'informational failures' as one of the key issues driving inefficient water use. Household water consumption is often unobservable (e.g. toilet flushes, washing machine, leaks, etc.) and, in cases where consumption is observable, it is difficult to quantify. Quantifying water end-usages can be both complex and costly, and so water usage is therefore *de facto* unobservable. Anecdotal evidence, gathered by the researchers from focus groups in Cape Town prior to the trial, indicates that water consumers are generally not aware of their monthly consumption, nor of the tariff structures and rates in place. The researchers argue that behavioural 'nudges' are especially useful in the South African context, a nation exhibiting extreme levels of income inequality. Pecuniary DSM measures can be punitive towards poorer households, who may experience hardship when faced with higher tariffs or physical restrictions. Furthermore, water consumption is subsidised for many low-income households, and thus may not be responsive to pecuniary interventions. Therefore, non-monetary incentives are more likely to be overall more effective across the income spectrum. In order to test the effects of behavioural nudges, various informative messages were attached to householders' monthly water bill over a 6-month period. Different types of message inserts were given to treatment households, with information such as tips for saving water, potential financial savings which could be achieved, social recognition incentives, 'social norm' messages or appeal for 'public good'. Their results indicated that all of the various treatment messages induced a reduction in household consumptions of between 0.6% and 1.3%, when compared with pre-intervention consumption from the same period a year prior. The most effective motivators were found to be the 'public good' messages (appealing to households to act in the public interest) and the 'social recognition' messages (public appraisal for usage reduction), which both resulted in consumption reductions of approximately 1.3%. The researchers then divided the households into five quintiles according to property value in order to assess the variance in the effectiveness of informative messaging across the income spectrum. The 'social recognition' and 'public good' interventions were found to be particularly effective in the upper quintile (wealthiest homes) where monthly consumption reductions of 852 litres (2.3%) and 702 litres (1.9%), respectively, were recorded.

2.5.2. Water Pricing Strategies

There are various approaches available for managing water resources in order to meet society's needs. Utilities can increase supply by investing in projects to construct new dams and other water infrastructure, or by upgrading existing infrastructure to increase its yield. However, as Jansen and Schultz (2006) noted, such projects are faced with hydrological limits and rising costs associated with pumping and transferring water over long distances as well increased environmental costs, to which society is becoming increasingly sensitive to in modern times. Therefore, in order to maintain economic efficiency, such costs must be traded off against society's willingness and ability to pay for water consumption.

Studies cited previously in this chapter have noted that the price elasticity to water demand is relatively low and can be considered to be inelastic. Authors such as Gaudin (2006) and Ramos et al. (2015) have found that consumers are generally unresponsive to price signals when information is unclear or the pricing system is complex, especially where consumers do not pay for water at the instance of usage or when complex, non-linear tariff structures are implemented.

A study conducted in Zaragoza, Spain, by Arbúes et al. (2004) sought to investigate changes in domestic consumption when tariffs increases were applied. The study found that although the increased tariffs lead to reduced consumption, the response was relatively insignificant with price elasticity values close to 0. The authors of the study noted that the water pricing structures implemented in the study area were designed with general cost recovery in mind, and found that the average household water bill in the study area is a very small percentage, roughly 0.52%, of average family income in Zaragoza. The authors therefore argued that the financial impacts of increased or decreased water consumption may be perceived as insignificant to household users and, as a result, authorities have little scope for manage domestic water demand via tariff adjustments. In conclusion, the authors of the study recommended that increased price levels be imposed, and that focus be placed on nonprice measures for managing water demand.

From an economic standpoint, the challenge that water utilities face is to implement pricing mechanisms that achieve cost recovery, yet are also equitable across the income spectrum, especially in the face of socio-economic inequalities. Pecuniary approaches to demand management therefore need to be designed in such a way that is efficacious without being punitive towards lower income households. To this end, many developing countries, including South Africa, implement increasing block tariff (IBT) structures which addresses the welfare concerns of the poor while curtailing excessive water usage in higher income brackets. Furthermore, in 2001 the South African Government initiated a 'free basic water' policy which sought to provide all citizens with a basic supply of free water. Households registered as indigent receive a free allocation of 6 kL per month but are charged for any additional water used if consumption is in exceedance of this block.

In South Africa, water service is typically provided by the city or local district municipalities, which each have slightly varying pricing structures. Furthermore, many municipalities have a separate set of tariffs

which are applied during periods of water restrictions. In some cases, a tiered system is applied whereby the block rates depend on the drought severity. An analysis of IBT structures applied to domestic consumers was undertaken, the details of which are discussed in Chapter 4. The results of this analysis are summarized in Table 3 to provide the context to the topics discussed in this section. Note that many water service providers in South Africa impose fixed fees and levies in addition to consumption tariffs, however, these were not considered in this analysis.

Table 3: Domestic water tariffs for municipalities in South Africa for the year 2019/2020

Municipality	Block	Normal Tariff - per kL Consumed (15% VAT incl.)	Restriction Tariff - per kL Consumed (15% VAT incl.)	Source
eThekweni Municipality (Durban)	0 - 6 kL	R0.00	N/A	(eThekweni Municipality: Water Tariffs, 2020)
	6 - 25 kL	R24.60	N/A	
	25 - 30 kL	R29.10	N/A	
	30 - 45 kL	R38.76	N/A	
	45 kL +	R59.78	N/A	
Stellenbosch Municipality	0 - 6 kL	R6.54 [1]	R6.93 [1]	(Stellenbosch Municipality Tariffs, 2019)
	6 - 12 kL	R9.90	R12.80	
	12 - 18 kL	R16.74	R48.63	
	18 - 25 kL	R28.69	R85.33	
	25 - 40 kL	R39.00	R117.94	
	40 - 70 kL	R60.95	R274.28	
	70 kL +	R91.43	R383.99	
City of Cape Town	0.0 – 6.0 kL	R17.15 [1]	R26.57 [1]	(City of Cape Town: Residential Water Tariffs, 2019)
	6.0 – 10.5 kL	R24.39 [1]	R43.16 [1]	
	10.5 - 24.5 kL	R34.63	R65.68	
	24.5 - 35.0 kL	R76.04	R376.00	
City of Johannesburg	0 - 6 kL	R9.10 [1]	22.29 [1]	(City of Johannesburg: Tariffs, 2019)
	6 - 10 kL	R9.66 [1]	25.12 [1]	
	10 - 15 kL	R16.49 [1]	57.70 [1]	
	15 - 20 kL	R23.99	95.96	
	20 - 30 kL	R32.95	135.19	
	30 - 40 kL	R36.51	182.76	
	40 - 50 kL	R46.62	270.64	
	50 kL +	R49.66	298.24	
Buffalo City Metropolitan Municipality (East London)	0 - 6 kL	R17.89 [1]	N/A	(Buffalo City Metropolitan Municipality Tariff Book Index, 2020)
	6 - 10 kL	R18.25	N/A	
	10 - 20 kL	R25.34	N/A	
	20 - 30 kL	R32.85	N/A	
	30 kL +	R41.22	N/A	

Nelson Mandela Bay Municipality (Port Elizabeth)	0 - 0.3 kL/d 0.3 - 0.5 kL/d 0.5 - 0.8 kL/d 0.8 - 1.0 kL/d 1.0 - 1.6 kL/d 1.6 kL/d +	R17.00 [1] R17.00 R17.00 R21.30 R21.30 R43.08	21.30 21.30 43.08 86.15 86.15 287.16	(Nelson Mandela Bay Municipality Tariff Book Index, 2020)
City of Tshwane (Pretoria)	0 – 6 kL 6 – 12 kL 12 – 18 kL 18 – 24 kL 24 – 30 kL 30 – 42 kL 42 – 72 kL 72 kL +	R11.11 [1] R15.86 [1] R20.84 R24.10 R27.55 R29.78 R31.86 R34.12	N/A N/A N/A N/A N/A N/A N/A N/A	(City of Tshwane: Supply of Water Tariff, 2019)
Mangaung Metropolitan Municipality (Bloemfontein)	0 – 6 kL 6 – 15 kL 15 – 30 kL 30 – 60 kL 60 kL +	R8.75 [1] R20.65 R22.94 R26.99 R31.30	N/A N/A N/A N/A N/A	(Mangaung Metropolitan Municipality: General Tariffs, 2019)
Note: The superscript [1] indicates that the block is free for indigent households				

Interestingly, although monthly billing cycles are applied in Nelson Mandela Bay Municipality (Port Elizabeth), the IBT blocks are defined in terms of *daily* quantities consumed, i.e. users are given the cost per kL depending on the quantity of water which they consumer *per day*, as shown in Table 3. This may make it more practical for users to be mindful of their usage and take daily steps to remain within a block threshold.

The municipalities reviewed in this study also have slightly varying policies regarding free basic water provision. All the municipalities provide the minimum mandated quantity of 6 kL free water to indigent households, while Nelson Mandela Bay, City of Cape Town and City of Johannesburg provide 9.0 kL, 10.5 kL, and 15 kL, respectively, as a poverty relief measure to indigent households. eThekweni Municipality is the only metropolitan municipality in South Africa to offer the free basic water to all consumers regardless of their income bracket, and also offers a discounted rate for all domestic consumers who install an approved break-pressure tank above their meters, as a means of reducing post-meter leakage and overall network demand.

Demand side management may also be achieved through the application of ‘Time of Use’ (ToU) tariffs, whereby differing rates are applied during peak and off-peak periods, as is commonly practiced in the energy and communication sectors. ToU tariffs have a twofold benefit to utilities since, in addition to demand management, it can be used to reduce peak demand, and thereby delay costly infrastructure upgrades. Implementation of ToU tariffs is less common in the water sector, and available literature

suggests that adoption thereof has so far been hindered by a relatively slow uptake in smart metering technology in the field.

Hui-ting and Yi-jie (2011) presented an overview of considerations to be taken when designing ToU tariffs for managing electricity demand. The authors noted that the first step is to define peak and off-peak periods, and that accurate partitioning of these two periods is critical for achieving effective outcomes. Daily demand curves vary depending on location, the size of the population served, time of year, and the social habits of users. As a result, portioning may be a complex task, with no "one size fits all" type solution. The authors presented a statistical approach for determining 'peak', 'valley', and 'flat' periods as well as the duration of each block, based on data collected from trial sites in China. Daily load data at hourly resolution was acquired over a one-year period and used to derive a continuous load curve, which was then divided into peak, valley, and flat segments. Thereafter, frequency statistics were applied to determine the frequency of each load point in each load segment, which in turn was used to calculate a probability index and plot a distribution curve. Peak and off-peak periods were then defined in terms of exceedance probabilities extracted from the distribution curve.

Venizelou et al. (2018) presented a similar methodology for developing an optimal ToU tariff structure for promoting effective demand side management. In the study, load profiles were recorded over a one-year period (reference year) at a pilot network comprised of three hundred (n = 300) consumer households and then used to generate the seasonal load duration curves of the participants. Statistical methods were used to analyse the seasonal duration curves and define the ToU block periods. Thereafter, the ToU rates were established by implementing an optimization function which maintained a neutral electricity bill (i.e. the electricity bill for the average consumer would remain unchanged if the load profile is kept constant). The developed ToU tariff structure was tested on the pilot network and it was found that the peak consumption during peak periods was reduced by 3.19%, 1.03%, and 1.40% for the summer, transition and winter seasons respectively, when compared to the reference year. The results obtained from the study indicated that implementing ToU tariffs was able to redistribute the system loads to off-peak periods.

2.6. Internet of Things

The term Internet of Things (IOT) is a concept referring to the interconnectivity of computing devices via the internet, enabling embedded objects and devices to send and receive data to connected servers and databases (Xia et al., 2012). The IOT has applications in many different domains, such as industrial automation, mobile healthcare, intelligent energy management, traffic management, and home automation, the latter of which is of particular focus in this research project. The following components are common to all IOT systems:

- Devices
- Network connectivity
- Data processors

- User interfaces.

Surveillance cameras, wearable health and fitness monitors, and smart meters are examples of common IOT devices which have become ubiquitous in everyday use, however virtually any device may be integrated into an IOT system. The device connection may be wired through a local area network (LAN) or more commonly a wireless connection via Wi-Fi, Low Power Wide Area Network (LPWAN), Global System for Mobile Communication (GSM), or Radio Connection (Sterne, 2019). This connectivity enables users to access data without having to be physically in contact with the devices, allowing for remote data analysis in real-time, and is one of the primary advantages of an IOT system.

Smartphones are an increasingly important aspect of the IOT, especially when considering home automation, since they are easily connected to a network via Bluetooth, Wi-Fi, or GSM and are capable of providing both the user interface and data computing components of a system (Zanella et al., 2014) Furthermore, it is possible to develop useful mobile applications which allow users to easily interact with the devices and databases. Examples of such applications are discussed further in Section 2.8.

Software which binds the various components of an IOT system does not conform to a single architectural style. Rather, the tools used to facilitate data transfer will be specific to each application. Software tools which were relevant to this research project are discussed in hereafter in this section.

2.6.1. Application Programming Interface

Application programming interface (API) is a computing interface which facilitates interactions between multiple software intermediaries, defining the kinds of calls or requests that can be made, the format of data that should be transferred, and any protocols and conventions that should be followed. APIs play a critical role in communication between applications and databases and authorizing the transfer of data. An API key can be used as a secret authentication token or unique identifier for locating resources (API, 2020).

2.6.2. Hypertext Transfer Protocol

Hypertext Transfer Protocol (HTTP) is a request-response application protocol which facilitates client-server data transactions (Hypertext Transfer Protocol, 2020). For example, a web browser may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server URL The server, which manages resources such as text files, images, and data, or performs other functions on behalf of the client, returns a response message.

2.6.3. Uniform Resource Locator

A uniform resource locator (URL), also known as a web address, is a reference to a web resource that specifies its location on a computer network, thereby providing access to the resource. For example, a

web URL may typically have the form ‘http://www.facebook.com/home.php’, which indicates a protocol to be followed (‘http’), a server host name (‘www.facebook.com’), and a file identifier (‘home.php’).

2.6.4. Markup Languages

Mark up languages are human-readable computer languages which use tags for identifying elements within a document and are useful for creating graphical user interfaces in applications. Two of the most commonly used markup languages are hypertext markup language (HTML) and extensible markup language (XML, 2020).

HTML is implemented by most webpages and functions as a data renderer, dictating the appearance and layout of data on a webpage. In HTML script an element is marked by predefined opening tags (e.g. <heading>) and a closing tags (e.g. </heading>) indicating that everything between the two tags is part of the heading and should be rendered accordingly.

XML is another markup language which is typically used for describing data and is very similar to HTML, which is concerned with displaying data, particularly on webpages. However, XML is more flexible in that developers can create custom tags which are compatible with the unique API framework of an intended application. It is often used in mobile applications for the layout of user interfaces.

```
<TextView
            android:id="@+id/heading"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#3F51B5"
            android:text="Main Menu"
            android:textAlignment="center"
            android:textColor="#FFFFFF"
            android:textSize="18sp"
            android:textStyle="bold"
/>
```

Figure 1: Example of XML markup

Figure 1 shows an element in XML format taken from the application developed in fulfilment of this thesis project. In this case the element represents a TextView object which is used for displaying character text in Android applications. The markup defines values for the object’s declared attributes such as its layout height, width, and text size.

2.6.5. JavaScript Object Notation

JavaScript Object Notation (JSON) is text data format which is easy for humans to read and write but is also easy for machines to process. JSON is built on two structures: A collection of key/value pairs and ordered lists of values, as shown in Figure 9. These two structures are universal, in that they are supported by virtually all modern programming languages in some form. This makes JSON an ideal data format for interchanging between various languages, for example when communicating between a Python based server and a Java based smartphone (JSON, 2020).

2.7. Smart Water Metering

The function of water metering essentially is to improve the balance between society's need to have access to potable water, a utility's need to receive payment for services rendered, and the joint responsibility of all to preserve finite water resources. Throughout the latter half of the 20th century as metering technology became increasingly viable, water utilities moved towards a more commercial approach to water service delivery through dedicated water bills, as opposed to cost recovery through rates and taxes. One of the main drivers behind this was the need to ring fence the financing of water services from other government activities (Boyle et al., 2013). Barraqué (2011) found that this move was also viewed as being more equitable for the customers, since the water bill was now linked to their actual consumption, rather than a flat rate or fee based on property size. According to Beal and Flynn (2014) a key aim of a smart water management system would be to build a reliable relationship between the customer and the utilities sector. A well designed and user-friendly application is beneficial to both the customer and service provider to detect any flaws that might exist in respect of the service rendered.

Boyle et al. (2013) found that more recently, the ever-increasing prevalence and affordability of technology has driven utilities' interest in intelligent metering as a means for implementing demand side management and water efficiency in an attempt to delay expensive infrastructure investments. The researchers expect that deployment of smart metering will transition from being predominantly "demonstration scale" towards a broader mainstream implementation in future. This evolution in water metering will bring to focus a number of issues, namely: the role of real-time data feedback; data ownership and privacy; data management and security; changing workforce skills required; and evaluating the costs versus benefits of implementation.

Smart metering systems are in essence an implementation of the IOT concept, since they typically contain all the core components thereof, namely: devices (e.g. smart water meter), network connectivity data processors, and user interfaces. Figure 2 shows the typical information flow of information in a smart metering system which integrates the various components associated with an IOT network.

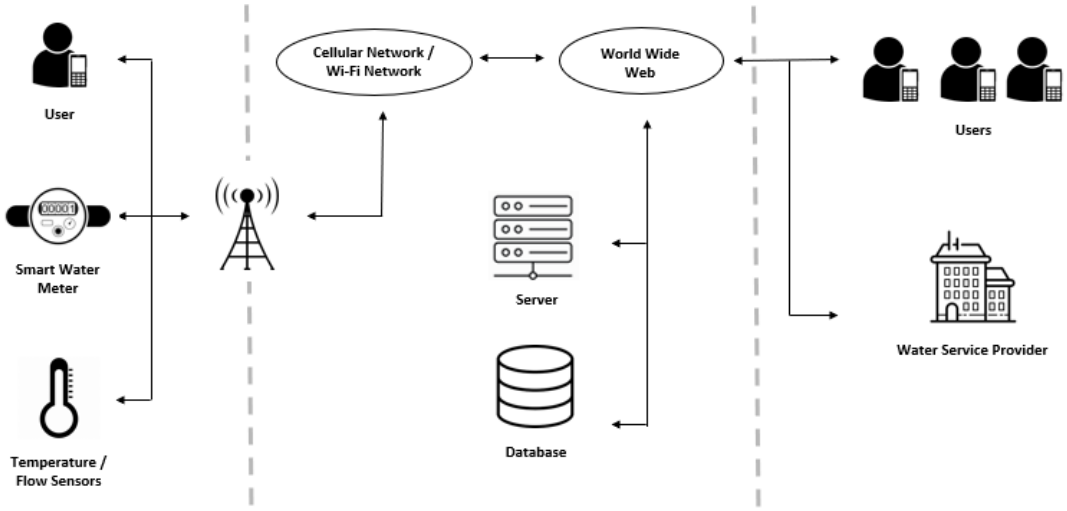


Figure 2: Typical flow of information in a smart metering system

Mudumbe and Abu-Mahfouz (2015) investigated a smart metering approach to water management in the South African context. The researchers noted that in South Africa, as with most other countries worldwide, utilities rely predominantly on manual water metering systems to manage water usage and supply. These readings are labour intensive, and the user may often be presented with an inaccurate bill due to a lag in reading or may be based on estimates of historical data. Despite the so-called data "big bang", and widespread implementation of smart technology seen in the energy sector, not much impact has yet been demonstrated in the case of water usage efficiency or effectiveness. Furthermore, the researchers developed a proof-of-concept smartphone application which allowed users to view their real-time data and historical data in visual format, and to compare cumulative usage graphs of two different periods. By enabling the user to view changes in their consumption patterns graphically, the user can then develop a better understanding of the driving factors influencing the change. For example, if the user has been attempting to reduce their overall consumption by fitting water-saving features and relying on harvested rain water, then the reduction will be reflected in the data and thus "rewarding" the user, and potentially reinforcing the behaviour. This "reward" of this feedback would potentially be increased if the data were accompanied by quantified financial savings. On the other hand, if the user observes an increase in usage, be it gradual or sudden, it could indicate water leakage due to an unattended tap or burst pipe.

In order to demonstrate a mobile application which had been developed, Mudumbe and Abu-Mahfouz (2015) installed a smart meter in an office building occupied by around 250 employees. The researchers posited that water consumption could be expected to occur mainly during working hours, i.e. between 08:00 and 16:30, and very little consumption outside these hours. The water meter data confirmed that, generally, virtually no water consumption occurred between 17:00 and 7:00. However, on one of the days during the trial period a constant consumption of approximately 10 litres per hour was detected overnight. This information prompted further investigation and a leakage was found outside the building, which was fortunately visible above ground.

Buffalo City Metropolitan Municipality in the Eastern Cape, South Africa, is currently in the process of installing MacroComm ICE-1-PULSE-6 devices at residences across the city. The ICE-1-PULSE-6 is a pulse meter device designed for easy retro-fitment to existing electrical, water flow and gas meters with a compatible pulse output, and allows for remote billing and leak detection notifications (ICE Catalogue, 2020). The device is approximately 93mm, has a capacity of 2800 mAh and an expected battery life of up to 10 years dependent on transmission frequency and environmental conditions. The photo shown in Figure 3 was taken on site by the author of this study, and shows a municipal water meter which has been fitted with an ICE-1-PULSE-6 transmitter. The devices reportedly transmit a meter reading to the municipality database once per day, however this information is not transmitted to the water user, who currently receives a monthly water bill which indicates the water consumed for the billing cycle. An optional application is available for the devices, which can be downloaded for a smartphone or accessed via the internet, and can receive alerts regarding water usage or potential leaks. The devices can be configured or upgraded via a technology known as near-field communication (NFC) which facilitates data transfer between two electronic devices over a distance of 4cm or less, in this case to allow a technician to set the information transmission frequency or to select parameters which will trigger a leak warning.

Figure 3: Household smart metering device

In Uganda, the ‘Mobile 4 Water’ application allows community members to report faulty water points via Short Message Service (SMS), providing water service providers with critical information regarding their supply network without requiring time and resource consuming field visits. The users on the other hand are incentivised by the promise of better service provision and accountability by prompting service providers to implement corrective action. Water vendors, who resell water from the utility network or private sources, play an important role in the provision of water services in many African countries where access to piped water limited, and service provision may be insufficient or inadequate. In Mauritania, for

example, it is estimated that 32 percent of the urban population are served by mobile water vendors (Ne et al., 2014)

M-Maji is a mobile phone-based information system widely used in Kenya, is an example of the of an effective smart water management system which greatly benefits the community. The application operates via Unstructured Supplementary Service Data (USSD) which is a protocol used by mobile phones to communicate with a service provider's computers or servers via text messages. This means that the application is compatible with virtually all mobile phones, and allows vendors to advertise their services, providing details regarding their selling price, their location, and whether or not the water is purified. The system also facilitates quality management by allowing buyers to report fraudulent vendors, thereby forewarning future buyers, and encouraging vendor accountability. Furthermore, the quality of the water advertised by vendors is verified by m-Maji staff during random monthly quality tests. Figure 4 demonstrates the functionality of the simple user interface for the m-Maji application.

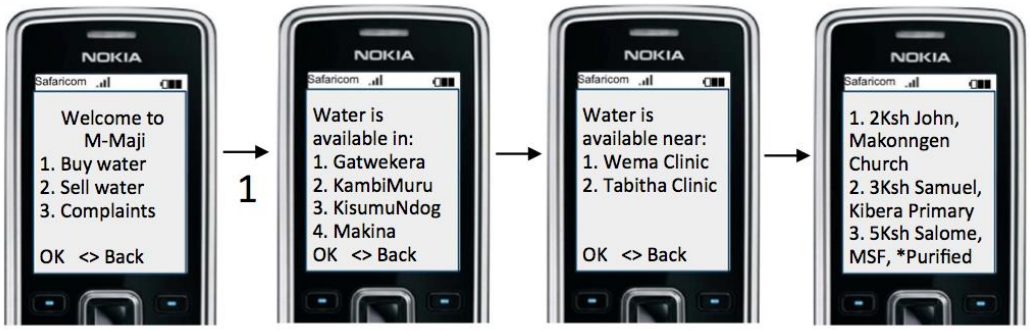


Figure 4: m-Maji application interface (M-Maji, 2012)

Nel et al. (2014) investigated challenges faced by water consumers and utilities in the water supply industry, and the extent to which smart metering and technology could address the needs of various stakeholders. Typically, a user's only feedback is a monthly water bill giving the cumulative consumption for the month, making it difficult to identify the effects of specific end-uses in their homes. By adopting smart metering practices, important data could then be accessed by users or disseminated automatically to relevant stakeholders to inform them of important events and help to understand consumption behaviour. Nique and Opala (2014) noted that the mobile industry has a tremendous presence and has become more accessible and affordable than both water and electricity services in the African continent, with GSM covering an estimated 76 percent of the African population. Due to ever-decreasing costs, mobile handsets are becoming an increasingly ubiquitous feature of society throughout Africa, and therefore being used as cost-effective tools for collecting and disseminating critical information.

2.8. Smartphone Applications

Developing a smartphone application for managing household water usage was a primary objective for this thesis project, and therefore a number of existing applications were reviewed in order to assist the

development process. Particular focus was placed on identifying and potentially emulating applications with simple user-friendly interfaces, investigating which type of servers and data transfer systems might be best suited for the functionality of the application, and to review available methods of graphically displaying data on smartphones.

A mobile device application and supporting cloud computing solution was developed and presented by Nel et al. (2014) as a proof-of-concept in order to demonstrate the potential of information technologies in the industry. The primary aim of the application was to assist demand-side management of residential household energy usage and water consumption. The application runs on Android operating system and allows users to monitor their hot water consumption in real-time, as well as view their historical usage data displayed graphically. Data is requested from a central server by implementing an API and suitable HTTP methods based on the task that the user is performing. The server can then return either the raw data to the user (e.g. instantaneous thermostat reading) or it could return data which has been processed to provide meaningful values (e.g. the average thermostat reading). The authors noted that providing excessive amounts of data may overwhelm or confuse the user, highlighting the importance of considering the 'user-friendliness' when designing an application. Their application was therefore divided into three tabs (interfaces), as shown in Figure 5 and Figure 6, each providing specific information as described in the following sections.

The 'Usage' tab, shown on the right in Figure 5, provides a summary of the most recent information reported by the data logging device, and includes an overview of the daily consumption as well as the associated cost (in Rands). The interface also includes a timestamp indicating when last the display values were updated, and users can manually refresh the data by clicking on a refresh icon in the action bar at the top of the screen. If the attempt to retrieve data from the server is unsuccessful (due to poor network connectivity etc.) then an error message is briefly displayed.

The 'Status' tab, shown on the left in Figure 5, summarizes the status and settings of the associated data logging device such as the current state of the water heating element (on/off) and the control mode which may be via a thermostat or a timer – the schedule of which can also be viewed.

Figure 5: Proof-of-concept application developed by Nel et al. (2014)

The 'Graph' tab provides graphical representations of daily usage data and allows users to select which date and which type of data they wish to view (e.g. water use or energy consumption). The interface is also interactive and allows users to select a point of interest on the graph to obtain the time and data values associated with that point, as shown in Figure 6.

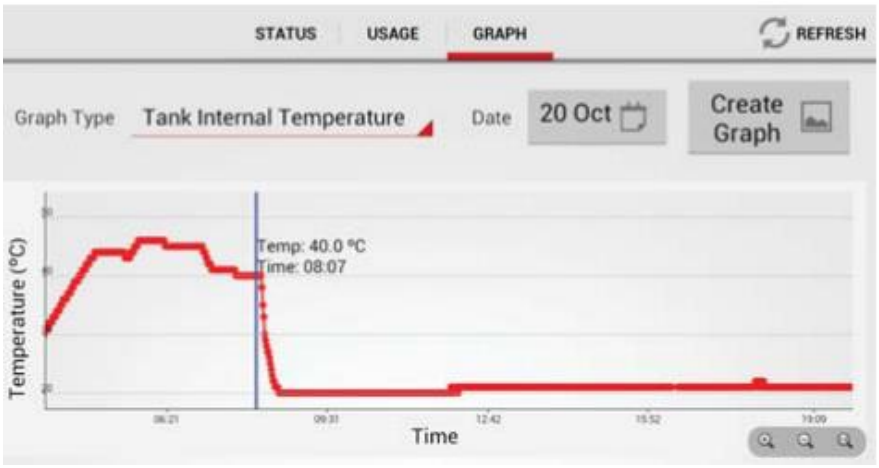


Figure 6: Graph tab implementing AChartEngine adapted from Nel et al. (2014)

The visual dials used in the Status and Usage tabs were created by implementing an open source library known as SpeedometerView, while the graphs implement an open source charting library known as AChartEngine. SpeedometerView and AChartEngine are both exclusively compatible with Android.

GraphView is library used for creating interactive graphs in Android applications. It is designed to be flexible in implementation and can display data as a line graph, bar graph or as points on a scatter plot. The data is rendered on the interface in real time, which means that it will update if data is changed or

modified. By default, the axes will be adjusted automatically to fit the data set, or alternatively developers may choose to manually set minimum and maximum range values. These parameters may also be overridden by the user if zoom is enabled, in which case the user can adjust the viewport scale using a two-finger touch gesture. The GraphView library is open source was found to be effective for the developed application, and was therefore implemented, as demonstrated further in Chapter 5.

3. Application Development

3.1. Overview

This chapter builds on the literature reviewed in Chapter 2, discussing the technical aspects of the application development and the various software components which were implemented in the final solution. The first step of the development process was to review popular mobile operating systems, including a comparison of the advantages and disadvantages of each. The application needed to be compatible with the flow sensors, water meters and database systems which were available, and therefore the hardware components are briefly discussed. Finally, the developed application needed to be given a practical name for listing on the relevant download platforms and is thus referred to as 'WaterWise' or hereafter.

3.2. Mobile Operating Systems

There are numerous operating systems available for mobile phone application development, however two operating systems – Android Operating System and iPhone Operating System hold almost the entire market share worldwide. Android, by far the most popular globally, holds roughly 84% of the market share for operating systems in South Africa, and 73% worldwide, with the remainder of the market held almost exclusively by Apple's iPhone Operating System (iOS), both locally and abroad (Mobile Operating System Market Share, 2020)

3.2.1. iPhone Operating System

iPhone Operating System (iOS) was created and developed by Apple Inc. to be used exclusively on its hardware and was first released in 2007 for iPhones. iOS is the most popular mobile operating system in the United States, with a market share of 58.2 % (Mobile Operating System Market Share, 2020), yet holds only 12.3% and 16.5% of the market share in Africa and Asia respectively. One of the main reasons for the relatively low adoption of the iOS platform in these latter two regions is that Android devices offer a broader price range and are generally more affordable in developing countries with a weak US dollar exchange rate.

iOS Software Development Kit (SDK) is the official platform for the development mobile application on iOS and is available for free download users of Mac personal computers but is not compatible with Windows operating systems. iOS SDK allows developers to build applications in the following languages: Objective C, C, C++, and Swift, but some companies have also created tools which facilitate iOS application development in other languages.

App Store is the official digital distribution system for iOS mobile applications. For developers to have their applications available on this platform, a yearly subscription fee of US$ 99 (R1659.93) must be paid.

3.2.2. Android Operating System

Android Operating System (hereafter referred to simply as 'Android') is a mobile operating system designed primarily for mobile devices such as smartphones first released in 2008. Android was developed by a consortium of engineers known as Open Handset Alliance and commercially sponsored by Google (Android, 2020). The operating system is based entirely on open source software and is supported by a large community of developers via various forums. Based on Androids' broad user base and open-source nature, it was decided that Android software was well-suited for developing the WaterWise application. Table 4 compares the most important aspects of Android and iOS operating systems.

Table 4: Comparison of popular mobile operating systems

Attribute	Android	iOS
Developer	Google, Open Handset Alliance	Apple Inc.
Global market share	74.1%	25.6%
First edition released	2008	2007
Programming languages	C, C++, Java, Kotlin	C, C++, Objective-C, Swift
Compatible SDK platforms	Linux, macOS and Windows	macOS
Software licensing	Open source	Proprietary
Official Application Store	Google Play	App Store
Cost to publish an application to official store	R419.15 (once off)	R1659.83 per year
Reference: (Comparison of Mobile Operating Systems, 2020)		

Android Studio is the official SDK for Android, and can be run on Windows, macOS and Linux. It supports code development in C, C++, Java, and Kotlin languages, the latter of which is a more modern and increasingly popular language. However, Java - the traditional language for Android software development - was selected for this application primarily due to its open-source eco-system and large support community which is active on forums such as GitHub and Stack Overflow. Android Studio provides the complete toolset for application development, including a virtual device emulator allowing the developer to test the applications functionality, and assisting the debugging process. In order to make an application available on the official Android application platform, Google Play, a once-off fee of US$ 25 (R419.15) must be paid, and thereafter the developer does not incur additional costs for uploading update versions of the application.

3.3. Water Meter Cameras

As a low-cost solution to remote water meter monitoring, camera installation at existing water meters was considered. There are numerous software packages available which are able to extract text characters from images of letters and numbers, a technology known as optical character recognition (OCR). Open-source software is freely available such purposes, and authors such as Nava-Ortiz et al. (2011) have investigated the automation of meter reading by implementing mobile phones and open-source OCR engines. Tesseract OCR engine was investigated for this study and images of water meters were tested to determine the accuracy of the software. It was found that complex feature extraction algorithms were required to isolate the objects of interest on a water meter and results obtained from the text extraction were found to be unreliable. Water meter OCR solutions were therefore not considered for further development.

3.4. Flow sensors

Sterne (2019) undertook a research project in which non-obtrusive flow sensor devices were developed to investigate trends in household water events. The devices are IOT compatible and consist of a vibration sensor, temperature sensor, and a micro-control unit (MCU) as shown in Figure 7.

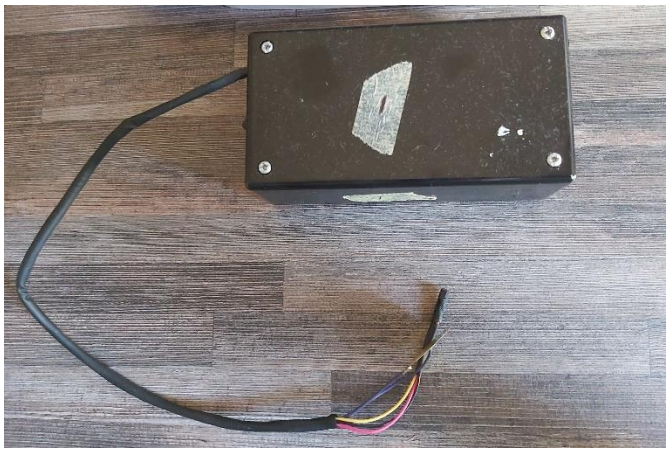
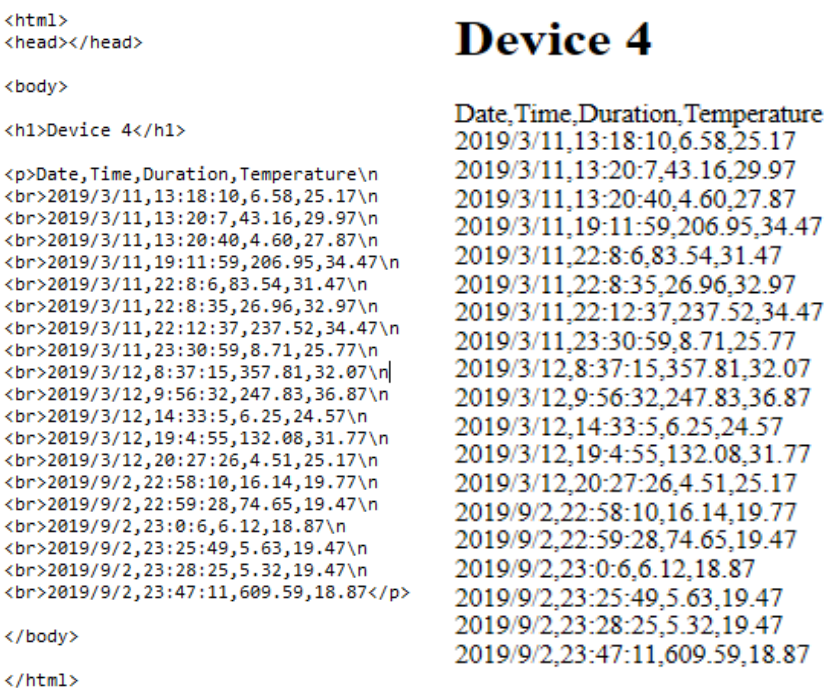


Figure 7: Image of one of the flow devices used in this study

The vibration sensor is attached to a pipe at the end use (e.g. tap, dishwasher, etc.) and detects vibrations in the pipe during water flow events. The mechanical vibrations are then converted into an electrical signal, which is received by the MCU, which in turn records the duration of the flow event. The temperature is also recorded along with the date and time at the beginning of the flow event. The data is stored locally on a memory card, to be transmitted daily to an online server via an established Wi-Fi connection. The device data was intended to be accessed by the user via a web-based application, and the devices were therefore programmed to automatically post the data to a host website in HTML format at 24-hour intervals. Figure 8 shows the device data posted in HTML formatting on the host website (left), and the same data viewed in a web browser (right).



```
<html>
<head></head>

<body>

<h1>Device 4</h1>

<p>Date,Time,Duration,Temperature\n
<br>2019/3/11,13:18:10,6.58,25.17\n
<br>2019/3/11,13:20:7,43.16,29.97\n
<br>2019/3/11,13:20:40,4.60,27.87\n
<br>2019/3/11,19:11:59,206.95,34.47\n
<br>2019/3/11,22:8:6,83.54,31.47\n
<br>2019/3/11,22:8:35,26.96,32.97\n
<br>2019/3/11,22:12:37,237.52,34.47\n
<br>2019/3/11,23:30:59,8.71,25.77\n
<br>2019/3/12,8:37:15,357.81,32.07\n
<br>2019/3/12,9:56:32,247.83,36.87\n
<br>2019/3/12,14:33:5,6.25,24.57\n
<br>2019/3/12,19:4:55,132.08,31.77\n
<br>2019/3/12,20:27:26,4.51,25.17\n
<br>2019/9/2,22:58:10,16.14,19.77\n
<br>2019/9/2,22:59:28,74.65,19.47\n
<br>2019/9/2,23:0:6,6.12,18.87\n
<br>2019/9/2,23:25:49,5.63,19.47\n
<br>2019/9/2,23:28:25,5.32,19.47\n
<br>2019/9/2,23:47:11,609.59,18.87</p>

</body>

</html>
```


Figure 8: Flow sensor data in HTML format (left) and viewed on a web browser (right)

The host website is password protected and does not have any API available for the application to utilize for retrieving the data in a suitable format. Therefore, in order to obtain the data loggings, the application server developed for WaterWise included functionality which acted as an intermediatory for providing access to the database. This component of the server pass in the username and password to the website, and then copy all the raw data in HTML formatting - a process known as “web crawling” or “web scraping”.

Table 5: Typical flow sensor data

Date [yyyy-mm-dd]	Time [hh:mm:ss]	Duration [s]	Temperature [^{O}C]
2019-09-16	09:32:11	9.3	22.1
2019-09-16	11:01:32	7.2	22.6
2019-09-16	13:59:02	4.1	23.1
2019-09-16	20:56:13	23.2	21.1
2019-09-16	23:11:11	8.9	20.7

Although connectivity to a smart metering device was key to the functionality and intended outcomes of the application, it was decided that access to data provided by one of the flow devices developed by Sterne (2019) may provide useful information to the user at a relatively low cost. The flow devices do not require any plumbing for installation and are placed at points of interest to monitor the activity of specific end-use applications such as showers, washing machines or taps. Note that the devices do not measure the volume of water consumed but record the duration of events and average temperature of the water used

over the duration, which is used to distinguish between hot-water and cold-water events. The intention of the flow devices is to help users become more aware of water usage at specific end points.

3.5. Smart Meters

There are several types of smart meters available on the market with various mechanisms of data transfer. For this study, simulated data was used for demonstration purposes as outlined in Chapter 4.

Each data entry contained a meter code for identifying the residence, a date and time stamp, and a meter reading in litres, as shown in Table 6.

Table 6: A sample of the data entries for the site used in this investigation

Meter code	Date / Time Stamp [dd-mm-yyyy hh:mm:ss]	Meter reading [L]
1010-003-0686	05-09-2016 08:20:56	222227
1010-003-0686	05-09-2016 08:27:11	222228
1010-003-0686	05-09-2016 08:33:41	222229
1010-003-0686	05-09-2016 08:39:57	222230
1010-003-0686	05-09-2016 08:46:26	222231
1010-003-0686	05-09-2016 08:52:41	222232
1010-003-0686	05-09-2016 08:59:12	222233
1010-003-0686	05-09-2016 09:06:26	222234

Smart meter data, recorded at high temporal and volumetric resolution, and provided in real-time via an application such as WaterWise, would empower the user to better manage their water consumption, and in turn may help utilities to manage overall water demand. The applications and benefits to the user are discussed further in Chapter 5.

3.6. Application Server

Much of an application's functionality is provided by a computing server, which forms part of the application back-end. The primary function of the server is to provide a database which the application communicates with when retrieving data, while at the same time providing some measure of data security. As part of this research project, an account was created with a free website known as PasteBin, which provides a basic application programming interface (API) functionality. This allows the developer to load content onto a site which can then be accessed by a third-party user, provided the user has the relevant URL and API key. In the WaterWise application, the user is requested to enter the URL and API key, which the application then forwards to the server when requesting data. The server uses the API key to identify which device data is being requested then returns the JSON format if it is available. If the API

key is invalid the server will return an empty JSON object and an error message, which the application relays onto the user.

Simulated smart meter data was used for this study, as will be elaborated upon further in Chapter 4. Therefore, the data was uploaded to the server and made available to the application for demonstration purposes. The flow sensor data was designed to be viewed on a web application, and it is not possible to be retrieved directly from the devices. Therefore, in order to view or retrieve the data, the analyst would be required to open the host site with a web browser and manually interact with HTTP login form, which would not be desirable from the standpoint of application functionality. Therefore, a home computer was implemented to run a web scraping algorithm, written in Python coding language, which would perform the following automated steps:

1. Launch a web browser to open the device host site
2. Submit the username and password keys using HTML requests
3. Navigate to the resource file containing the data for the required device
4. Retrieve the contents of the page and store it as a String
5. Convert the raw string data into a list by using *split()* function. As seen in Figure 8, each line begins with the markup script “
“ and therefore serves as an identifier for each data entry
6. Once the data is split into individual entries, an iterator can be used to run through each element in the list and remove remaining markup script e.g. “\n”, “<p>”, etc.
7. Each object in the list is then be split again at the comma (“,”) separator to create a series of values, thereby creating a set with the following format:

   [“Date”, “Time”, “Duration”, “Temperature”]

   e.g. “01-09-2016”, "7:02:17", "9.3", "19.3"]
8. The values in the set are then paired with respective keys, to create a JSON object with a final formatting as follows:

```
{
        “Date”: “01-09-2016”,
        “Time”: "7:02:17",
        “Duration”: "9.3",
        “Temperature”: "19.3"
}
```

9. Use the web browser to open the host website (pastebin.com) and upload the JSON objects, as a JSON array, to the unique URL allocated for the flow device.

The host site will then function as the application server for the WaterWise application and will only allow access to the data if the application request carries the correct device API key. Figure 9 shows a screenshot of the host website containing data for one of the flow devices used in this study. The address bar at the top of the browser contains the URL which is unique to that specific device. In this case, the URL is

comprised of the hostname (pastebin.com) and the API key ("/raw/KzsDeuRy") which is case sensitive. A user who wishes to access the data for this device will therefore require this key.

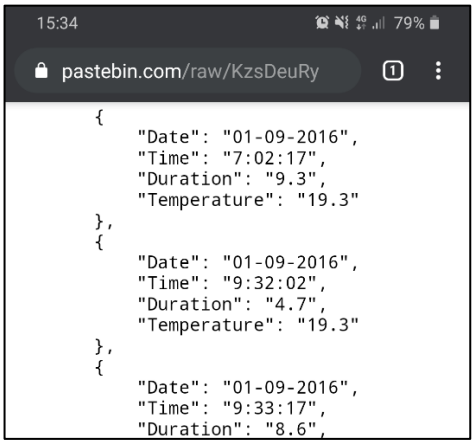


Figure 9: Screenshot showing sample data in JSON format on a host website

A similar process was followed for uploading water meter data to the server. When the application requests data from the server URL, hosted by PasteBin, it is read using native Java class methods without requiring conversion to an alternative format.

3.7. WaterWise: Application Components

This section provides an overview of the various software components implemented to build the application. These components are comprised of Java objects, classes, and methods, XML files, data files, as well as various other resources which provided functionality to the application. Minimal coding was included in the following sections and chapters, but relevant coding resources were instead added in Appendices.

The following platforms were implemented in the development the WaterWise application: Java Platform Standard Edition 8 (Java SE8), and Android API Level 29 (also known as Android 10). These platforms provide important building components which formed the foundation of the application.

3.7.1. Java Development Kit

Android Studio, the official development platform for Android software, currently employs a native Java Development Kit (JDK) built on the Java SE8 library and includes all basic functions for working with primitive datatypes and objects. However, for the sake of compactness only the most relevant components functions are discussed in this section, and are summarized in Table 7.

. **Table 7:** Summary of important classes and methods implemented from JDK 8 packages

Name	Description
URL	Objects of *URL* class contain pointers to indicate the location of a resource on the World Wide Web, e.g. flow device data on an online server.
HttpURLConnection	*HttpURLConnection* is an abstract class which provides methods to make a request to an HTTP server, initiating a session. *HttpURLConnection* is typically used in conjunction with *InputStream* and *InputStreamReader* to process data from a web server. In the WaterWise application it was implemented to retrieve flow meter and flow sensor data from the web server.
InputStream	*InputStream* is an abstract class which provides methods for receiving and holding a sequence of data elements, in the form of bytes, from a resource, e.g. an *HttpURLConnection.*
InputStreamReader	*InputStreamReader* class is used as a bridge to receive bytes from an *InputStream* and decode them into characters of a specified charset, usually the default charset used by the platform. This data can then be stored by a primitive data type such as a *String*.
FileOutputStream	A *FileOutputStream* is used for writing data to a file. In the developed application it was implemented together with *OutputStreamWriter* to write JSON data to text files (.txt)
OutputStreamWriter	*OutputStreamWriter* class is a bridge from character streams to byte streams, producing data of a specified *charset.*
try/catch	The *try* statement is used for creating a protected block of code which is tested for possible errors and throws an *Exception* message if an error occurs. The *catch* statement receives an *Exception* message if an error occurs and contains a block of code to be executed in such an event. The *try/catch* statements are used in pairs and help prevent an application from crashing.
Exception	*Exception* is an abstract class and is used along with *try/catch* statements to create *handlers* which manage errors within a given block of code. Implementations of the *Exception* class return a message to the runtime system of a computer indicating that a disruption has occurred to the normal flow of information, and to terminate the operation.
JSONObject	The *JSONObject* interface is used for creating instances of objects with JSON key/value mapping formats. Instances can be created using a buffered *InputStream* object, or by using the *add* method to add key/value pairs. Note that the "value" may be another nested JSON object or array.
Note: Italics is used to indicate that a term refers to an object, class, interface, or method in the Java programming environment.	
Reference: (All Classes, 2020)	

3.7.2. Android Libraries

Android 10 is currently the latest version of the operating system, released in September 2019, and was used for development of the WaterWise application. Android ensures backwards compatibility when releasing new versions of their operating system, meaning that packages and components designed for older versions of their software are always compatible with newer releases, but not necessarily vice versa. Therefore, some of the newer packages implemented for this application will not be compatible with smartphones running on older software.

Most modern smartphones are characterized by large, interactive touchscreens which are an integral component for application functionality and Android has many packages available which contain classes specifically designed for managing screen layout and user interaction. The following core Android packages were implemented extensively in the development of the WaterWise application: *view*, *widget*, *os*, *application*, *database*, and *content*.

The *view* package contains mainly abstract classes and interfaces which define the fundamental screen layout components and user interaction events. These can then be used by developers to build custom activities and interfaces. The *widget* package contains mostly visual user interface components which are implementations of individual components from the *view* library (Documentation for App Developers, 2020). Widgets simplify the development process since they provide most of the basic interactive functionality required for the application, which the developer would otherwise have to build from scratch. The relevant *view* and *widget* components which were utilized for development are summarized in Table 8.

Table 8: User interface components implemented from the Android 10 Platform

Name	Description
Layout	A *Layout* resource is an XML document (.xml file) which defines the architecture for a user interface in an *activity*, or a component of a user interface in an *activity*. All elements in the layout are built using a hierarchy of *view* objects.
TextView	A *TextView* is a user interface element which displays text to the user. The text may be fixed, or it may be editable to allow the user to enter a username or password etc.
Button	A *Button* is user interface element which the user can tap to perform an action.
ImageView	An *ImageView* displays image resources and supports most image formats. An *ImageView* may be customized to be clickable and function in the same way as a *Button.*
ProgressBar	*ProgressBar* is used to create graphical elements which update the user of the progress of an operation.

Toast	A *Toast* is a *view* containing a short message for the user. *Toast* is designed to be unobtrusive and disappears after a short interval.
LinearLayout	*LinearLayout* holds other nested *views* arranged either as a single column or a single row
TableLayout	*TableLayout* arranges nested *views* into rows and columns. Each row is defined by *TableRow* objects which in turn arrange nested elements into rows.
Note: Italics is used to indicate that a term refers to an object, class, interface, or method in the Java programming environment.	
Reference: (All Classes, 2020)	

The *os* package provides basic operating system services, some of which dictate how the application interacts with the smartphones' hardware and software. The *content* package contains classes used to access and publish data within the application, including API which allow content sharing between different application components and tools to manage application resources such as layout files, string data, media and images. The functions of the specific classes of interest are listed in Table 9.

Finally, the *app* package contains high-level classes which bind the Android application components together. *Activity* and *Notifications* are two of the core Android components which are defined in this package. An *Activity* is responsible for providing the screen for user interaction, and therefore forms the foundation upon which other user interface components operate. An *Activity* can also be used for starting a new *activity*, such as opening a new user interface, opening a web browser, or launching another smartphone application (Documentation for App Developers, 2020).

Table 9: Other components implemented from the Android 10 Platform

Name	Description
Package*: app*	
Activity	An *activity* is broadly defined as any task that a user can do. Objects of this class are responsible for creating a window in which the user interface is placed. The two most important attributes of an *activity* object are its *layout* resource file and its java class file. The *layout* file contains *view* and *widget* components for rendering the user interface, and the java file contains coding which governs the behaviour of the interface.
onCreate	*onCreate* is a method called when starting an *activity* and is where most class methods and variables are initialized, such as the *activity's* layout and event listeners (e.g. a button click).

Notifications	The *Notifications* class is used to modify how the application's notifications appearance and behaviour. *Notifications* are a critical tool for communicating with the user and optimizing application functionality.
Package*: content*	
Context	*Context* is an abstract class contains functions whose implementation allows access to application-specific resources.
ContentValues	Objects of the *ContentValues* class provide data authentication functionality and are called when updating databases. An object of this class represents a single row object to be relayed into the next open position in the database.
Intent	An *intent* is an abstract description of an operation to be performed, and objects of this class provide the facility for launching a new *activity*.
SharedPreferences	*SharedPreferences* interface provides method for saving application data to the phone's memory in the form of key/value pairs. *SharedPreferences* can only store the following Java datatypes: *boolean, float, int, long, String* and *Set<String>*.
Package*: os*	
AsyncTask	*AsyncTask* is an abstract class which must be subclassed to create asynchronous task, i.e. runs a computational thread concurrently with the user interface thread. This allows an application to perform tasks such as retrieving data from an online server, while still allowing the user to interact with the interface. In the WaterWise application, *AsyncTask* was used to create a method called *DownloadTask* which connects to the internet and retrieves flow device and water meter data.
Bundle	A *bundle* is used together with an *Intent* to pass data between activities
Note: Italics is used to indicate that a term refers to an object, class, interface, or method in the Java programming environment.	
(Documentation for App Developers, 2020).	

3.7.3. AndroidX

AndroidX is a relatively new Android support library released by Jetpack, compatible with Android API level 28 (Android 9.0) and higher. AndroidX offers full backwards compatibility with older Android components and has superseded some of the packages in the original support library, as developers migrate towards full adoption of AndroidX. The most important class of interest from this package is *AppCompatActivity* which is implemented when creating an activity and allows newer platform features to be used on older versions of Android software.

AndroidX package includes a new widget called *RecyclerView* which generates a scrollable list of elements, is more flexible and dynamic than older widgets such as *CardView* or *ListView*, which have since been deprecated by Android. Each element in a *RecyclerView* is contains its own set of *view* objects structured according to its layout file and is populated in runtime (i.e. while the user interface thread is active) by implementing an adapter, which is responsible for providing the views that represent items in a data set. In WaterWise a class called *DataListAdapter* was created to bind data to recycler views.

3.7.4. Android Manifest

Every Android application project must have an *AndroidManifest.xml* file which describes important information about the project and is used by the operating system to run the application (Documentation for App Developers, 2020). The manifest file is required for declaring the following:

- The project name and version which is used for generating a unique application identifier tag on Google Play and phone system
- The activities which will be run on the application, as well as specifying the main activity which is to be opened by default when the application is launched
- The permissions which the application needs in order to access other applications or protected regions of the system. Permissions may include internet access, camera access, or access to increased heap memory
- Details of the hardware and software compatibility requirements for the application. This information is important for listing on the Google Play store.

The manifest file for the WaterWise application is shown in Appendix A

3.7.5. Application Activities

As defined in Table 7, an activity is any task which the user can do. In this application, an activity was created for each user interface. Each activity has its own XML layout and Java class resource which share the same name as the activity, e.g. the “Tariffs” activity has a layout file named tariffs.xml which dictates the layout of the interface and a Java file named tariffs.java which contains instructions regarding the functionality of the interface. The following list summarizes the application activities which were created as well:

- Main Activity – Main menu which opens upon launching the application
- Flow Device – Table of latest flow sensor data
- Water Meter – Summary of recent smart water meter data and water bill
- Tariffs – Table of IBT blocks and rates
- Graphs – Interactive water usage graphs
- Customer – Table of customer details

The coding for all activities, written in Java programming language, is included in Appendix B. Coding for the respective XML layouts is included in Appendix C.

3.8. WaterWise: Application Structure

In order to simplify the overall application structure, the demonstration version of the app allowed for connectivity to only one smart water meter, one flow meter, and is only capable of calculating the cost of water bills for users in Stellenbosch Municipality. Data is pre-processed by the application server into a suitable JSON format to reduce the processing load on the application. WaterWise obtains JSON data from the application server by exploiting structured API calls. The JSON files are then stored in phone memory as cache data, allowing for offline usage of the application. The data in the user interface is presented in graphical or tabular format.

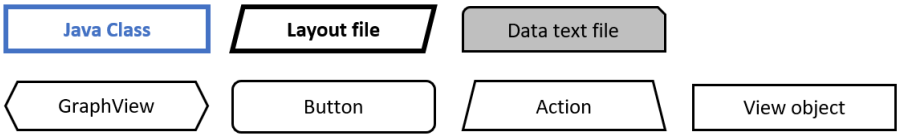


Figure 10: User interface elements keys

This section will describe the structure of each *activity* in WaterWise application in terms of its components and functions, as well as the relationship between the various activities.

Figure 10 shows the keys which will be used to represent the *activity* components in flow diagrams throughout the rest of this chapter.

3.8.1. Main Activity

Upon launching the application, the user will be taken to Main Activity, which functions as a main menu. This activity will utilize MainActivity.java and activity_main.xml resource files for interface functionality and rendering. All data files necessary for the application will be loaded from the server at this point, and written to text files (.txt) in JSON format to be stored on phone memory. The layout file contains a GraphView object which will display a graph of the water usage for the current day.

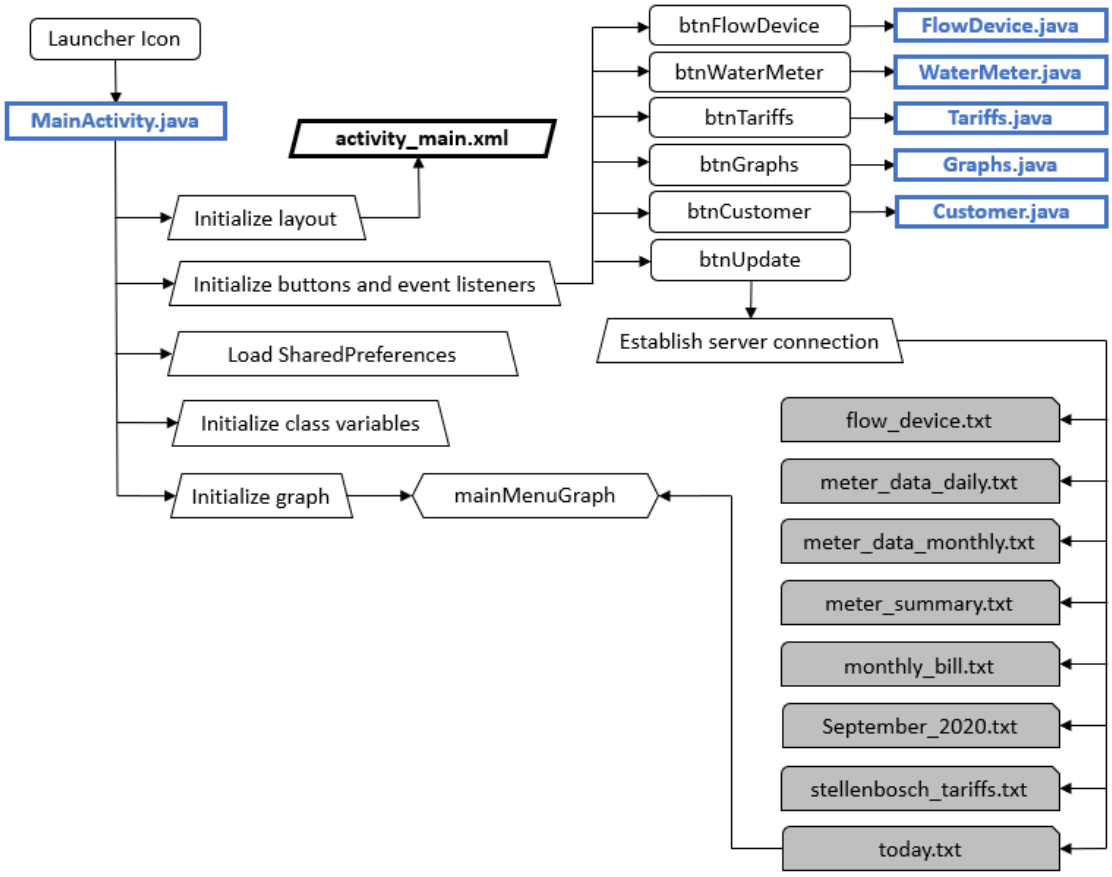


Figure 11: Main Menu activity flow diagram

The Main Menu interface provides a panel of buttons for navigating to the other available activities as shown in Figure 12. If one of the buttons is clicked, an *intent* will be called to launch the requested activity.

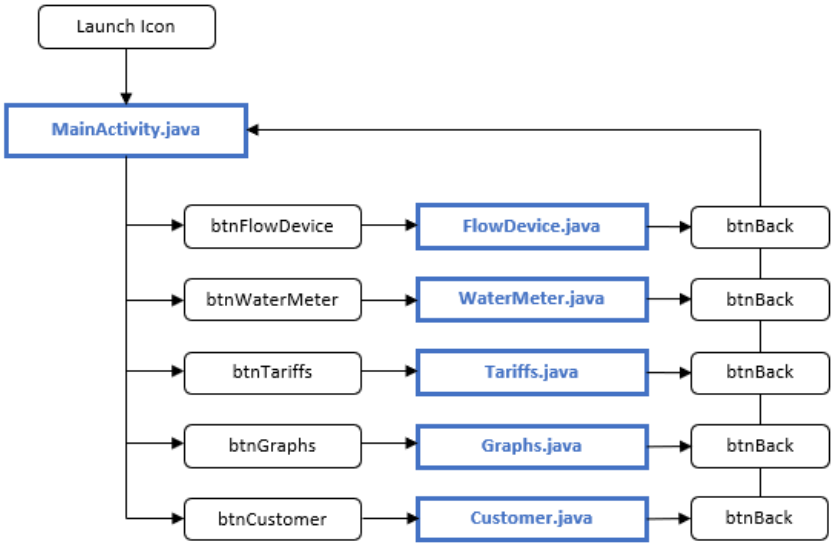


Figure 12: Navigating the user interface activities

3.8.2. Flow Device

The purpose of the Flow Device activity is to provide the user with direct access to data from one of the flow devices described in Section 3.4. When this activity is launched FlowDevice.java and flow_device.xml will be utilized to update the user interface, and initialize all *Layouts*, *SharedPreferences*, variables and *Views* as applicable. Figure 13 shows a schematic representation of the functionality of the Flow Device activity.

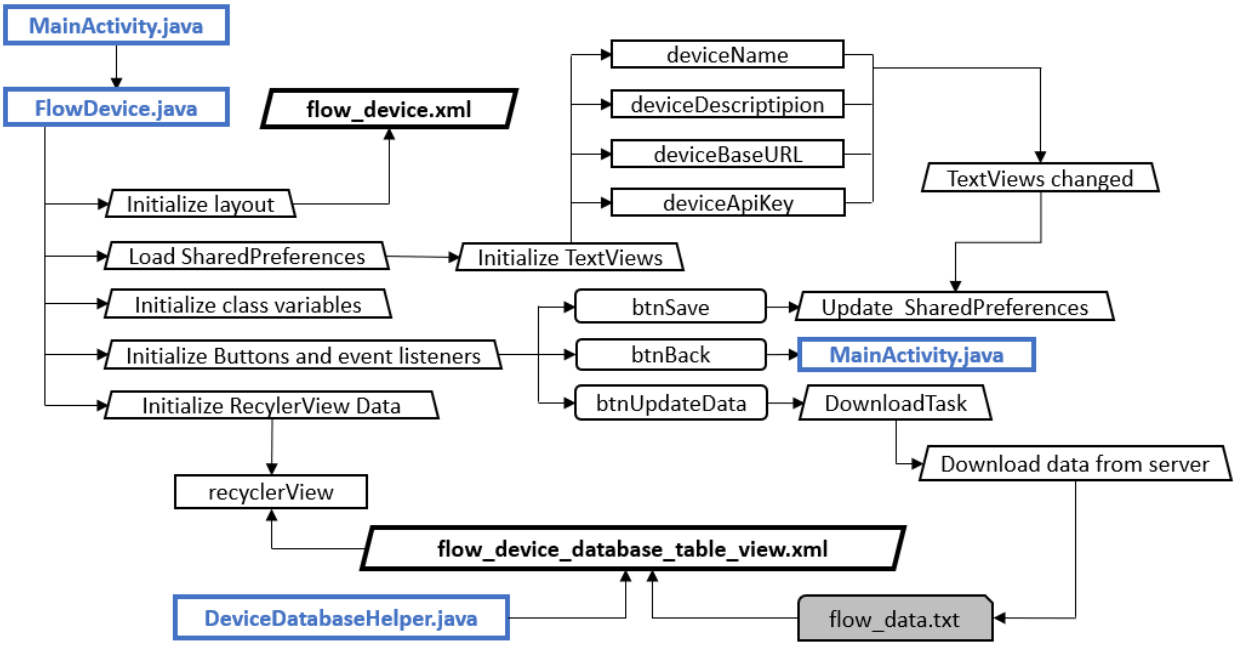


Figure 13: Flow Device activity flow diagram

The activity interface provides text fields which the user may use to update the following device attributes: device description; the URL for the device data; and the device API key. If the text fields are updated, then the 'Save' button must be clicked in order to save the updated attributes to phone storage. The latest data from the connected flow device will be displayed through *RecyclerView*, and may be refreshed by clicking the 'Update' button.

3.8.3. Water Meter

The Water Meter activity will function as an interface for a connected smart meter and will provide a summary of important data which has been pre-processed by the application server. The Water Meter interface has limited interactive capability, and it's structure is summarized in Figure 14.

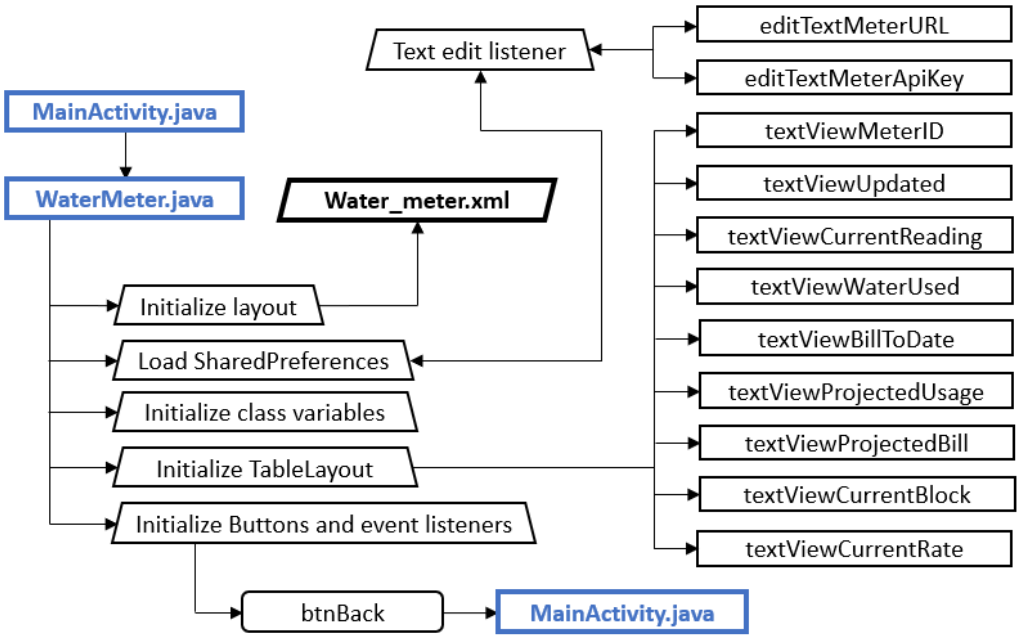


Figure 14: Water Meter activity flow diagram

3.8.4. Tariffs

The Tariffs is an informative interface with limited interactive capability, containing a *TableLayout* which displays the billing structure for the local water utility. An interface flow diagram is show in Figure 15.

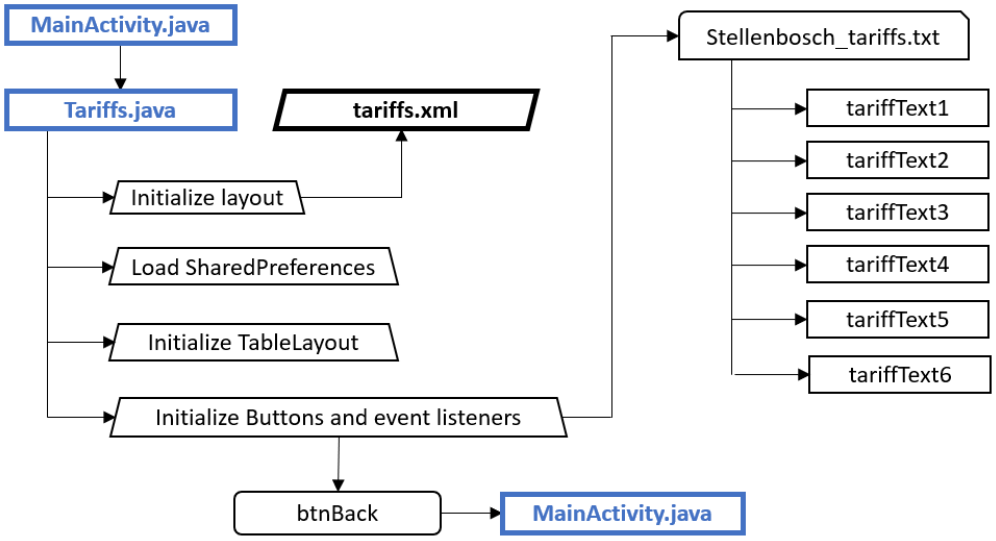


Figure 15: Tariff activity flow diagram

3.8.5. Graphs

The Graphs is an interactive activity which allows the user to view a more detailed history of their water usage. Customer.java and customer.xml files are utilized for executing this activity. The activity uses a GraphView to display historical data by month or by yea. A summary of the activity structure is shown in Figure 16.

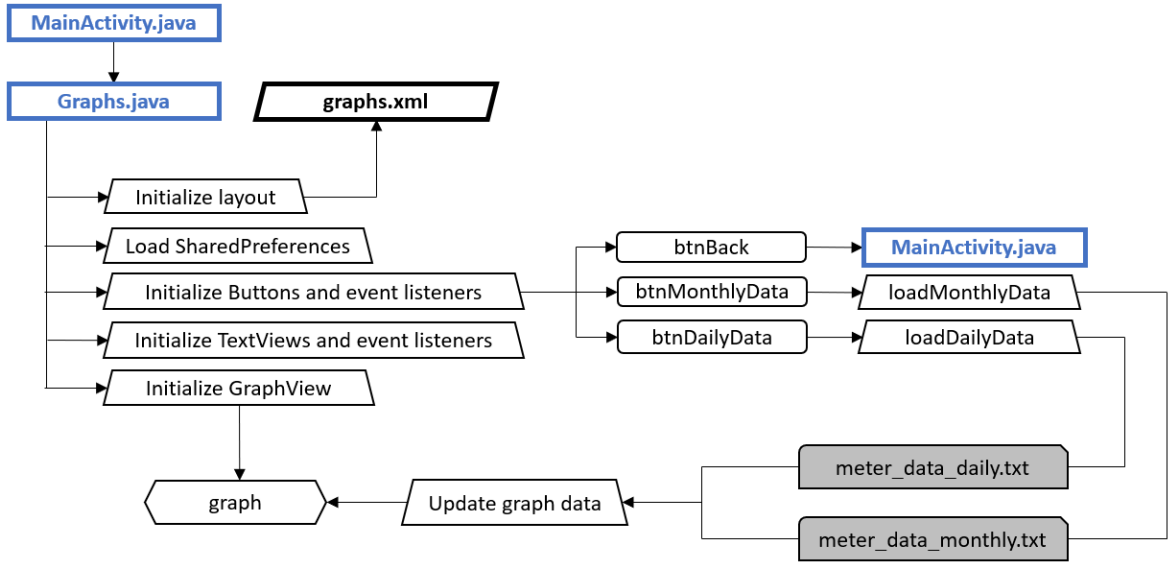


Figure 16: Graph activity flow diagram

3.8.6. Customer Data

Customer Data activity was designed to facilitate local storage of pertinent customer information. The activity utilizes Customer.java and customer.xml upon launching to render the interface, and utilizes the SharedPreferences interface to store data.

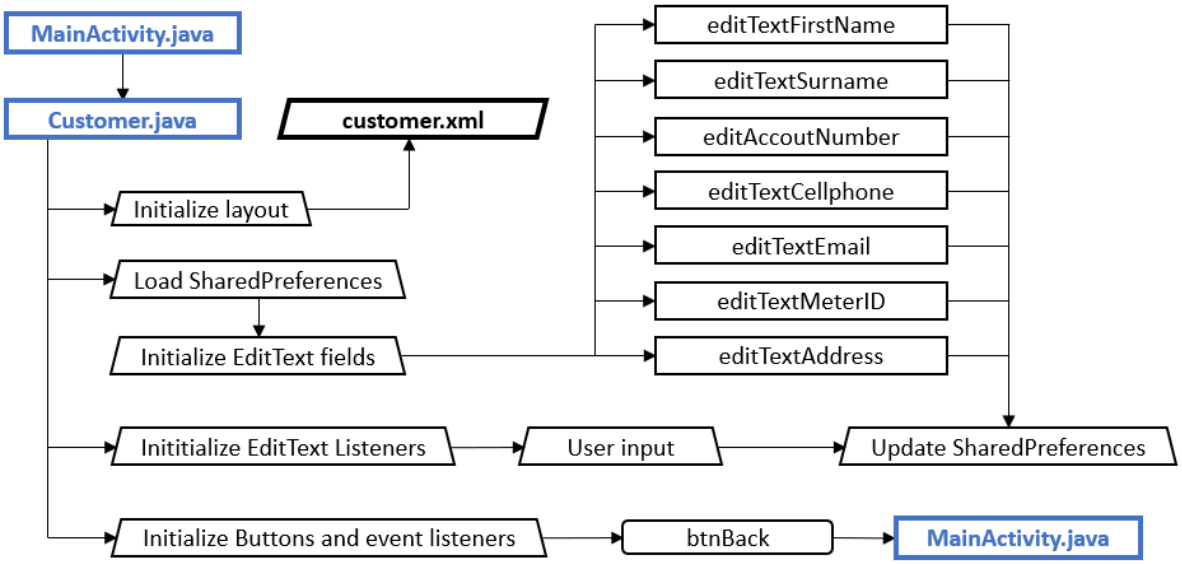


Figure 17: Customer activity flow diagram

4. Data Analysis

Two sets of data were collected and analysed for this research project, and the procedures followed for analysing these two datasets are discussed separately in this chapter. The second dataset comprised of water meter readings, obtained from a previous study by the Stellenbosch University Civil Engineering Department and is discussed in Section 4.1. The first dataset comprised of water tariffs rates for selected municipalities in South Africa, extracted from various sources and is discussed in Section 4.2.

4.1. Smart Water Meter Data

4.1.1. Data Filtering and Cleaning

In order to demonstrate the capabilities of the developed smartphone application, it was decided to create simulated database to represent smart water meter recordings of a typical household. Secondary data was obtained from a previous study by the Stellenbosch University Civil Engineering Department, comprised of water meter loggings recorded at a four-person household within an estate complex in Johannesburg, South Africa. The data was stored on an Excel spreadsheet, had a temporal resolution of 15 second, and spanned a period of 17 months, from September 2016 to January 2018. This data was ideal for analysis due to its high resolution and due to the fact that the household been verified to ensure that no leaks were present.

A single data entry contained only a meter reading, a date/time stamp, and the unique home code to identify the site. The smart meter which recorded the data would read the meter every 15 seconds, and then log the average between this interval, e.g. if the meter reading incremented from 10000 to 10002 over the 15 second interval, the logger would record a reading of 10001, with litres being the unit of measurement., A sample of the data obtained from the site is shown in Table 6.

In total the dataset contained 79431 individual data entries spanning a period of 511 days, including 214 non-null days and an overall average consumption of 746.6 L/d. However, there were a number of gaps in the data - presumably due to device failure - which required correcting. Between the period of April 2017 and October 2017, virtually no data was available. It was therefore decided to build a simulated dataset as follows:

- Data entries where the time difference between the first recording and the last recording for the day was less than 18 hours were discarded, as it was assumed that these days would not have captured all water use events taking place in that day and would thus be unreliable. 59 days were discarded using this parameter. The average daily consumption for these days was 302.7 L, significantly less than the overall average. After discarding incomplete days, the average daily consumption for the remaining days in the dataset was 940.5 L/d.

- The majority of the data was recorded between October 2016 and September 2017, and so this period was used as the hypothetical analysis period.
- Days without data were "filled in" randomly using days from the cleaned dataset which represented the same day of the week and from within the same season.
- No data was available for the winter period of the analysis (June – August), which is typically the period associated with the lowest daily consumption. Data for this period was thus generated using randomly distributed data from autumn (March – May) and spring (September – November), provided they represented the same day of week to be simulated. The values for this period were sorted, from highest consumption per day to lowest consumption per day, and the upper quartile values were then replaced with random values from the lowest consumption days of the overall dataset.

The methods and assumptions used for building the dataset were considered to be acceptable for the objectives of this investigation, since it was not crucial that the simulated data be representative of overall consumer behaviour. Rather, the aim was to demonstrate the capabilities of the smartphone application which had been developed, and how the high-resolution data might be able to assist users with managing their water usage and quantifying the potential savings of altered water-use behaviour.

4.2. Water Tariff Analysis

As discussed in Section 2.5.2, one of the strategies which water utilities can implement to manage consumer demand is to apply tariffs to curtail excessive water use, also known as pecuniary approaches. In South Africa, IBT structures are implemented by most water service providers. For this investigation, the most recent IBT structures of eight different municipal water service providers were investigated to compare domestic consumer rates in South Africa. Typically, municipalities have a standard domestic IBT structure which is applied during non-drought periods, and one or more IBT structures which are applied during periods of when water restrictions are in place. For this investigation, only two of these structures were considered for analysis: the standard IBT rates, and the emergency IBT rates applied during most severe water restrictions.

The municipal IBT structures for the following cities were analysed:

- Buffalo City, Eastern Cape
- City of Cape Town, Western Cape
- City of Johannesburg, Gauteng
- eThekweni, Kwa-Zulu Natal
- Mangaung, Free State
- Nelson Mandela Bay, Eastern Cape
- Tshwane, Gauteng.

The data was obtained from various sources, published by the respective municipalities, and consolidated into one Microsoft Excel worksheet, as shown in Table 3. Each municipality had different block thresholds, that is, the quantity in kilolitres per month (kL/m) which water users had to consume to progress into a higher tariff varied between the municipalities considered, and thus the rates could not be compared directly. Therefore, cumulative water cost was plotted against total water consumed for households in the various municipalities, with results presented in Figure 18 and Figure 19.

Figure 18 compares the cost of water for domestic users in various municipalities during non-drought periods.

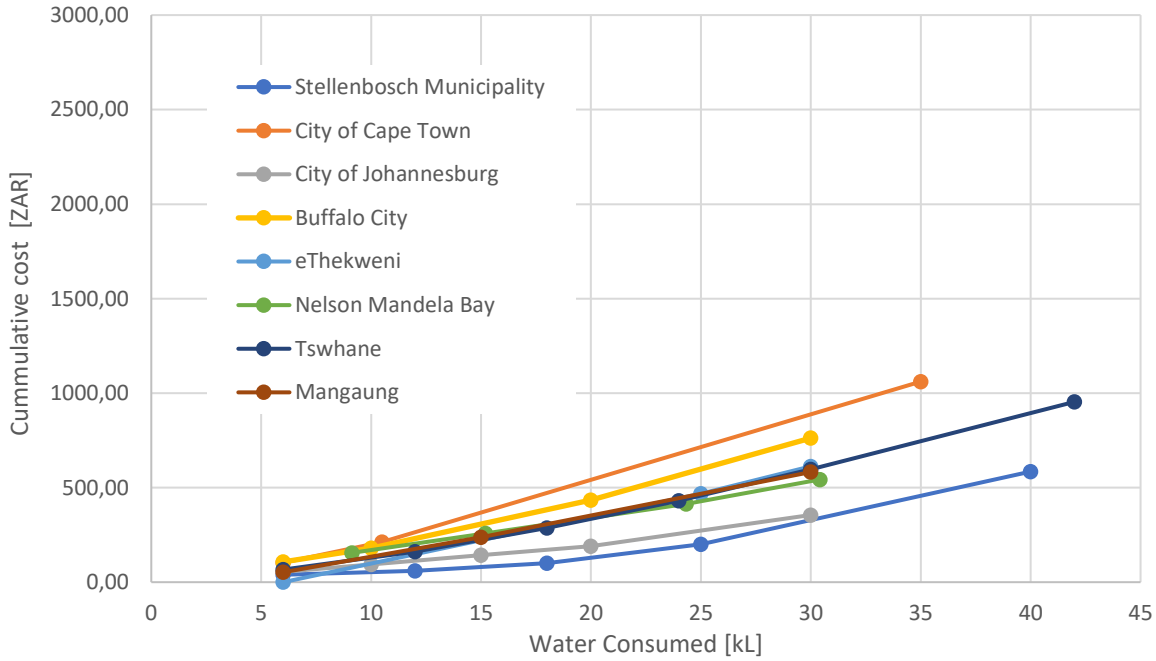


Figure 18: Comparison of standard water tariffs for municipalities in South Africa

Figure 19 compares the cost of water for domestic users in various municipalities when severe water restrictions are in place. Note that no emergency rates were available for Buffalo City, eThekweni, Tshwane and Mangaung.

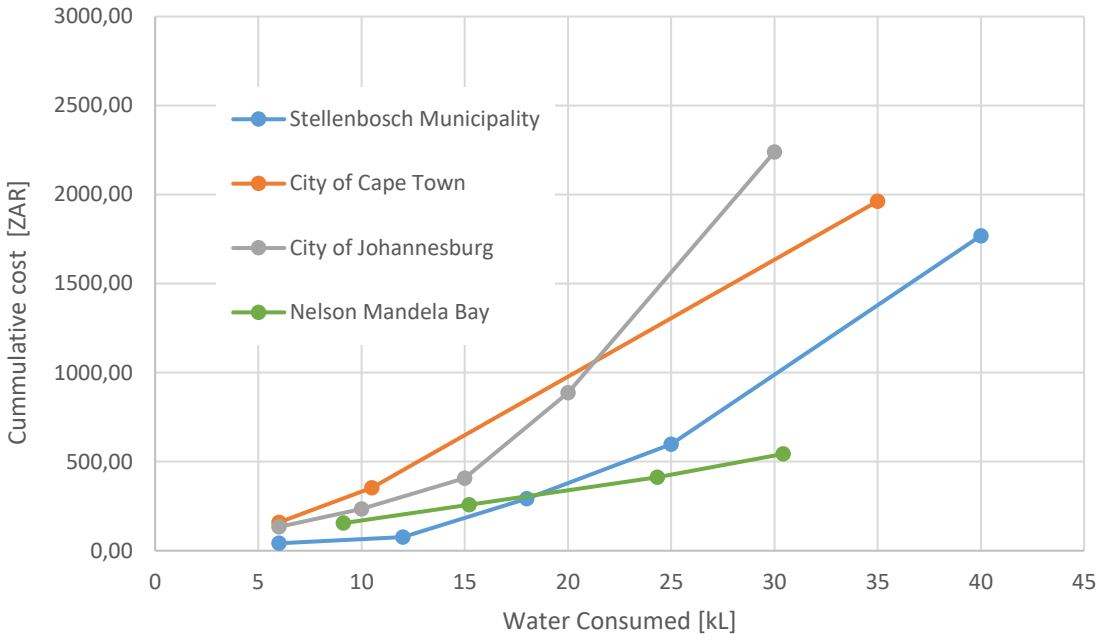


Figure 19: Comparison of water restriction tariffs for municipalities in South Africa

In Chapter 4.1 a simulated database was created to represent the water consumption of a typical four person household in South Africa, with an average daily consumption of 858 L, giving 215 L consumed per person per day. These results were then used to generate a typical water bill each of the municipalities considered in this analysis, as shown in Table 10. Typical bills are also calculated for periods of water restriction, where such tariff structures are in place. The values shown exclude additional fixed charges and levies applied for water services, which varies between various municipalities.

Table 10: Typical monthly water bills for various municipalities in South Africa

Municipality	**Water Bill**	
	Normal tariffs applied	**Restriction tariffs applied**
Buffalo City	R 632.07	N/A
City of Cape Town	R 750.72	R 1374.14
City of Johannesburg	R 480.91	R 1753.56
eThekweni	R 497.56	N/A
Mangaung	R 491.66	N/A
Nelson Mandela Bay	R 449.96	R863.36
Stellenbosch	R 440.44	R 1129.87
Tshwane	R 487.59	N/A
Average:	**R514.12**	**R1280.23**

4.3. Data Formatting

JavaScript Object Notation (JSON) was identified in previous chapters as a suitable data format for this application, due to its compatibility across various programming languages. JSON objects are both - readable by computers and humans, and data can thus be verified by users.

The formatting process used for converting the flow sensor device data into JSON is described in Chapter 3.8.2. All other data which was to be used by the app was stored in table format in Excel spreadsheets, and therefore needed to be converted into JSON separately, which was achieved by implementing an online parsing tool. Figure 20 shows an Excel spreadsheet table containing sample data for demonstrating the formatting process. The row 1 and row 2 from the table are highlighted for reference, and is labelled as 'Object 1' and 'Object 2', respectively.

Date	Time	Duration	Temperature	
01-09-2016	07:02:17	9.3	19.3	Object 1
01-09-2016	09:32:02	4.7	19.3	Object 2
01-09-2016	09:33:17	8.6	19.7	
01-09-2016	10:10:32	9.9	19.7	

Figure 20: Sample data in Excel spreadsheet

Figure 21 from Figure 20 after being reformatted. Each row from the original table corresponds to a single JSON object. The collection of JSON objects is referred to as a JSON array.

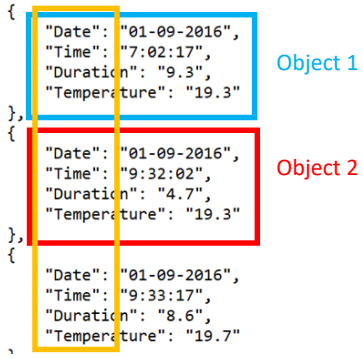


Figure 21: Sample data in JSON format

The data, in JSON format, is stored in text files (.txt) and uploaded to the application server website. This data is then accessible by the WaterWise application via API calls. The text files containing the application JSON data is show in Appendix E.

5. WaterWise Application

A proof-of-concept application, named WaterWise, was developed to provide household water users with a tool to manage consumption and achieve water conservation. In Chapter 5.1, the developed application is presented to demonstrate how the target aims and objectives, discussed in Section 1.4, have been addressed. Thereafter in Chapter 5.2 the application is further discussed in terms of its usefulness for household water users, and how the application aims to advance water conservation, with reference to the literature reviewed.

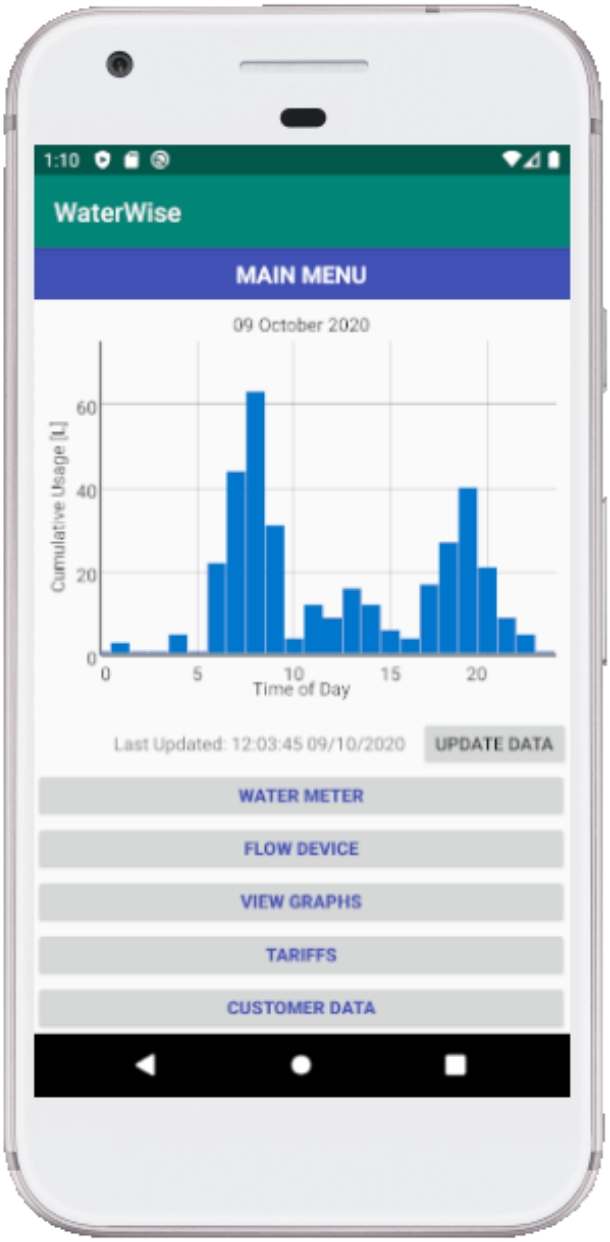


Figure 22: Main Activity Interface

5.1. Proof of Concept

5.1.1. Main Menu

Upon launching the application, the user will be taken to the main menu, as shown in Figure 22. The interface contains a graph displaying the user's water usage for the current day at an hourly resolution, as well as a panel of buttons which are used to access other application activities.

Literature reviewed noted that interfaces which are overly complex are likely to confuse or overwhelm the user, and so the layout of the main menu was designed to be as simple and as user-friendly as possible. The interface buttons facilitate intuitive navigation of the application, allowing the user to easily find the information which they are looking for.

The first time the user opens the application, permission for internet access will be requested, and the user will be able to select whether the application is able retrieve data using mobile data or Wi-Fi, or whether data may only be updated when connected to a Wi-Fi network. Thereafter WaterWise will automatically connect to the application server when launched and retrieve all necessary data.

5.1.2. Water Meter

The Water Meter activity interface provides the user with a table of important data regarding their water use for the current billing cycle. Although smart meters are capable of recording flow at a high temporal and volumetric resolution, the resulting dataset may be cumbersome and impractical. Therefore, it was decided that this interface would provide only a summary of useful information which is likely to be of interest to the user.

A screenshot of the meter data interface is shown on the left of Figure 23. The table heading indicates the monthly billing cycle currently in progress, which the application assumes to start on the first day of each month and ends the last day of that month. This may not reflect billing cycle recorded of the water utility's database, and will be addressed when considering future development of the application. The first three items in the table, namely 'Meter ID', 'Last Updated', and 'Meter Reading' are provided primarily as a means of data verification. If the user has access to their water meter, a manual reading can be taken, and the identification number (or serial number) can be compared against the Meter ID reflected in the application. If discrepancies should exist, it may indicate a device error or possible data corruption. The user can also check the date and timestamp to verify that the data is up to date.

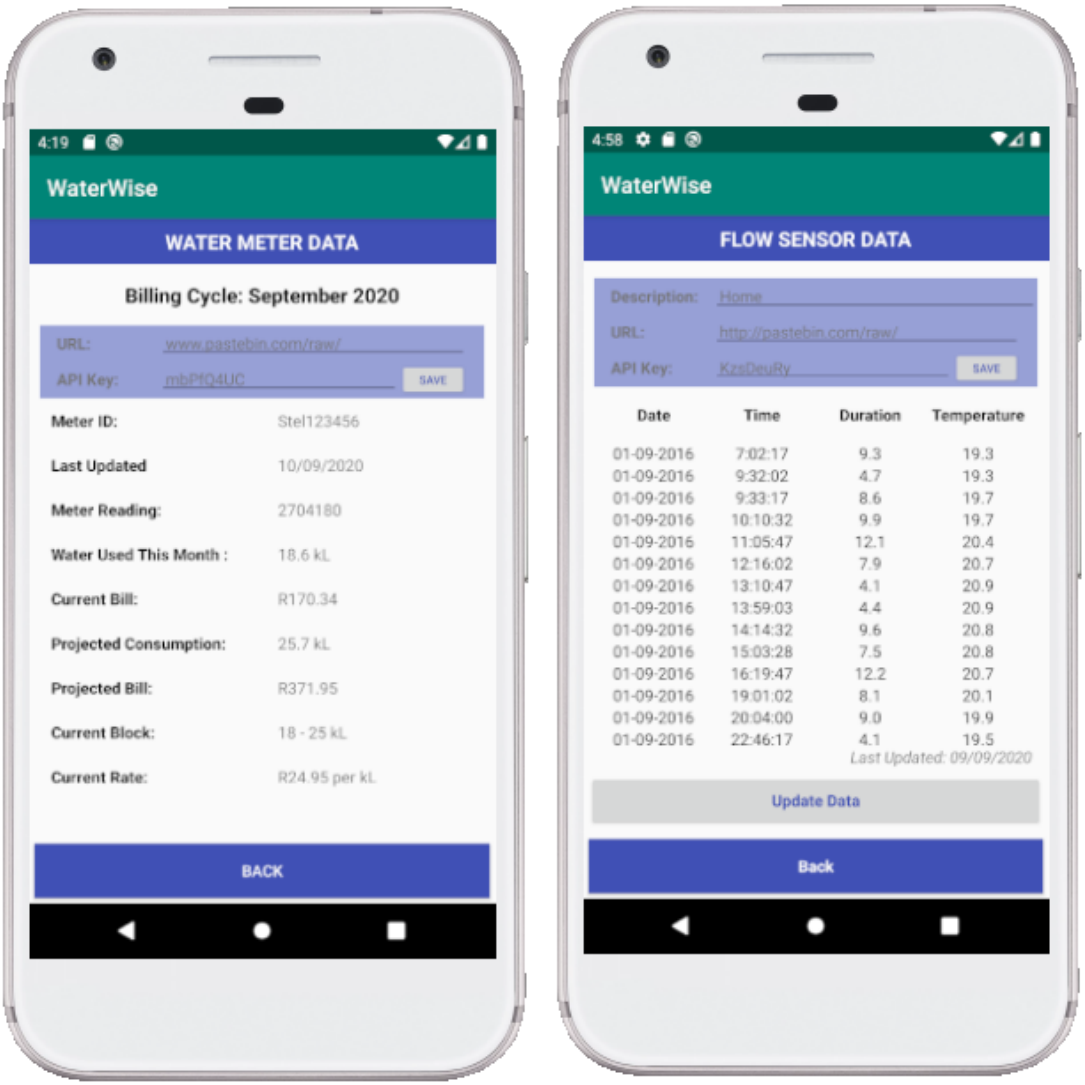


Figure 23: Water Meter interface (left) and Flow Device interface (right)

In order to calculate the monthly water bill, the application assumes that the user is non-indigent, and therefore does not qualify for free basic water. However, it is envisioned that future versions of the application would provide more versatility in this regard, allowing the user to select which tariff structure is applicable for their household. The application presents the tentative water bill based on the water used for the current water cycle, implementing the IBT structure of the appropriate water service provider. Note that many water service providers apply different IBT structures during periods of water restrictions, and this is managed by the application back-end. The Tariffs interface, shown on the right of Figure 23, allows the user to view the tariff structure used for calculating their water bill.

The Main Activity interface also informs the user which block rate is currently being used to calculate their water bill. Although many users may be aware of IBT structures in place, they are unlikely to be aware of which rates are applicable to their household at a given point in time. It is posited that by providing the user with their current rate, which will change as they progress into higher tariff blocks, the

user will develop an appreciation of the financial impacts of moving into higher pricing tiers. Furthermore, the application extrapolates the average daily consumption to estimate the total water which will be consumed for the billing cycle, calculate a projected estimate of the month's water bill, and determine which rate would be applicable in that threshold.

5.1.3. Flow Device

The Flow Device activity was developed with the intention of supplementing the smart metering capabilities of the application. The devices which were developed for a research study by Sterne (2019), are placed at specific end-points of interest to record activity. Although the devices are primarily intended for use in research, they are relatively inexpensive in comparison to smart metering systems, easy to install, and are capable of providing meaningful data household water users.

Below are a few examples of implementations which could provide users with meaningful feedback:

- If the user aims to save water by reducing their shower durations for example, then the WaterWise application can be used to monitor daily usage, and if desired, send a notification to the user to inform them of water use events exceeding a selected threshold
- Remote monitoring of irrigation systems
- Remote monitoring of rainwater tank usage
- Remote monitoring of borehole pumping activity.

The flow devices are programmed to upload the data to a host website, which users would have to open in a web browser to view. The interface provided by the WaterWise application, shown in Figure 23, makes it simpler for the user to access the data, and users may find table easier to read compared than HTML markup formatting on the host website, as shown previously in Figure 8.

5.1.4. Customer Details

This interface is used for storing important customer information on the phone device. It is envisaged that in future versions of the application, this data could be used as a bridge to facilitate efficient communication between water utilities and their customers.

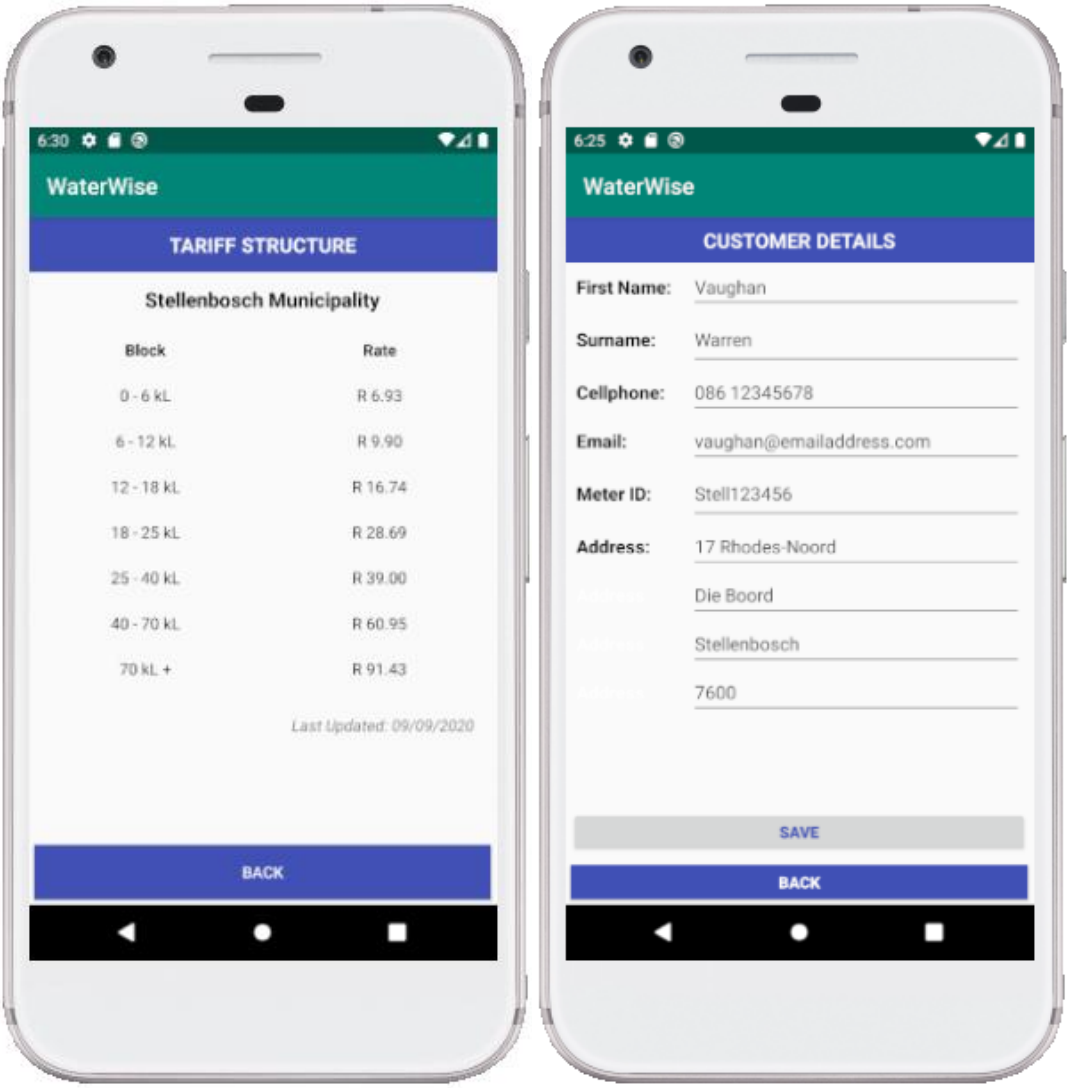


Figure 24: Tariffs interface (left) and Customer Details interface (right)

5.1.5. Graphs

The Graphs interface provides users with information regarding their historical consumption in graphical format. Users can view consumption data for a specific month aggregated by day, as shown on the left of Figure 16, or alternatively they can view their annual consumption data for a selected year aggregated by month. The graph is interactive, allowing users to zoom in or out using finger gestures.

Figure 25: Graphs interface data at a daily resolution (left) and monthly resolution (right)

5.1.6. Notifications

Notifications are powerful tools for enhancing user engagement. Android OS provides useful tools which allow developers to easily create customized notifications for their applications. Three categories of notifications were created for the WaterWise application, namely: Updates, Warnings, and Nudges.

'Warnings' would be issued to a user if water usage exceeded a selected threshold, with the intention of prompting the user to investigate a potential leak. Various parameters could be used to trigger a warning notification, and appropriate thresholds would vary between households. For demonstration purposes, this application was programmed to issue a 'high consumption' warning if the smart meter detected usage exceeding 100 L over an hour period, or would issue a 'leak' warning if more than 35 L usage was registered between the hours 23h00 and 05h00, as shown in Figure 26. The user can tap on the notification panel, and will be directed to the water meter data activity to inspect recent usage.

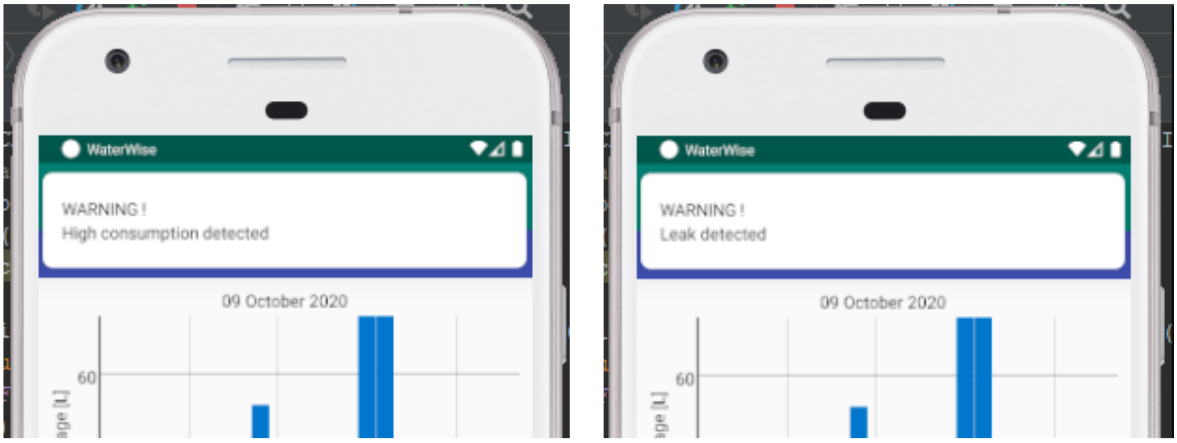


Figure 26: Warning notification

'Update' notifications would be used to provide the user with consumption information at set intervals (e.g. daily, weekly, or monthly) providing a summary of their consumption over the reporting period, as shown in Figure 27. At a daily interval, the application can provide feedback regarding the quantity of water consumed for the day, as well as the associated cost. The application can also notify the user at the end of each billing cycle and inform them of their month's consumption and provide a provisional water bill.

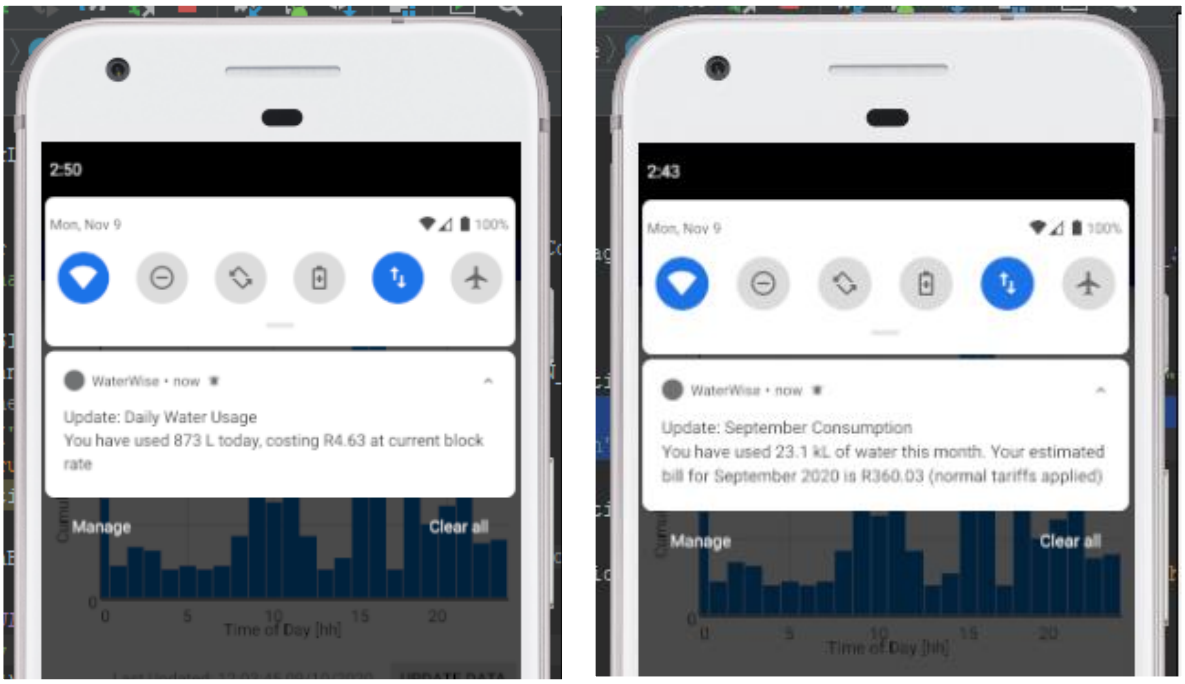


Figure 27: Update notifications

'Nudge' warnings were designed to actively encourage user consumption. A list of briefs, informative messages was compiled for demonstration, containing 'social norm' and 'neighbourhood comparison' type information with the heading '*Did you know?'*. The notifications were programmed to be sent at

random intervals during the day, and would provide users with a baseline against which to compare their own usage.

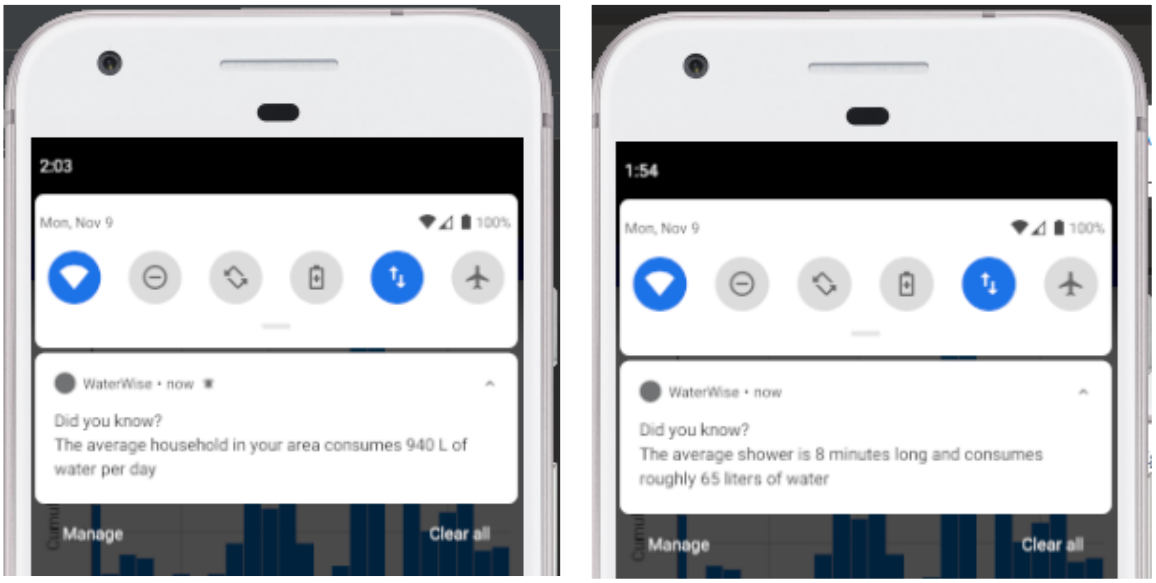


Figure 28: Behavioural intervention messages

Android allows users to turn off application notifications if they do not to receive communication alert updates, since users may find such features to be obtrusive or unhelpful.

5.2. Discussion

In this section the application is discussed in terms of its ability to address the needs which were identified in literature. In the literature review an overview was developed of the various factors influencing water demand and typical household end-uses. Thereafter, water demand management strategies were reviewed, and two broad categories were identified, namely: pecuniary approaches and communication approaches.

When considering the factors influencing water demand, many can be considered fixed for the purposes of this application. That is, there is no scope for influencing the price of water, household size, weather, or water supply pressure. On the other hand, it the WaterWise application could be used to address post-meter leakage and consumer behaviour.

Studies by authors such as DeOreo et al. (1996), Couvelis and Van Zyl (2012), and Lugoma et al. (2012) have noted the prevalence post-meter leakage on household sites both internationally and locally in South Africa, and it is estimated that on-site leakage may contribute to up to 6.0% of overall household water demand. Currently, subterranean leaks may remain undetected for years, especially if the flow is relatively low. High resolution meter data, accessed via developed application, may help the user to identify signs of leaks early if water use is observed while the home is unoccupied or if persistent flow is recorded overnight. Literature suggests that significant water savings could be achieved through simple leak management.

Authors such as Datta et al. (2015) have presented findings which strongly suggest that simple behavioural interventions can prompt water users to significantly reduce consumption. Various behavioural 'nudges' were identified, of which the following are considered applicable to this application: 'plan-making', 'public good', 'neighbourhood comparison' and 'social norm'. Simply providing users with consumption feedback in real-time has been found to prompt a reduction in consumption (Stewart et al., 2013). The application currently utilizes a set of pre-selected informative messages which are displayed to the user in random notifications. This functionality could be expanded or modified to implement specific behavioural interventions.

WaterWise could also be used to achieve water savings by augmenting existing pecuniary strategies. In theory, IBT structures are implemented by water utilities to curtail the demand of high-consumption water users. However, it is debatable whether these measures are effectively able to curb excessive consumption since the contribution many household end-uses may be difficult for users to quantify. Furthermore, water bills are issued after each billing cycle and so users are not "penalised" for excessive consumption at the time of usage. Adoption of smart technologies in the water sector has also been identified as key for implementing ToU tariffs to reduce demand and optimizing overall network performance. By notifying users of their consumption in real time, and helping to quantify costs as they are incurred, users may become more inclined to conserve water.

6. Conclusion

6.1. Summary

For this research project, an extensive literature review was undertaken, focusing primarily on water demand and management strategies. Available literature suggests that there is considerable scope for reducing household consumption, particularly by via implementation of behavioural interventions or 'nudges'.

Research and development of a smartphone application was undertaken to demonstrate how smart metering hardware could be implemented in an IOT type system to help household consumers conserve water resources. Through a review of existing knowledge and literature, various software and hardware products were identified, as well as user needs to be considered. Two major application development platforms were identified, namely, Android and iPhone Operation System (iOS). Android was favoured for the development of this application due to its broad user base, open-source environment, and active support forums. The application, named WaterWise, was successfully developed and capable of processing water meter data, flow sensor data, and billing data as well as other useful consumption information. A simulated database was created and used to demonstrate the functionality of the WaterWise interface. A demonstration version is available for download from official platforms and contains the following key features: water usage summary, information regarding water tariffs and monthly water bill, historical water usage graphs, and warnings notifications (e.g. high usage or leaks) as well as various 'nudges' to encourage conservation.

6.2. Future Research and Development Needs

This study focussed primarily on the development of front-end software, with very little emphasis placed on back-end architecture. However, in reality, a robust back-end is crucial for providing application functionality and data security. Indeed, data computing and data processing have been identified as core components of any IOT system. Server functionality in this project was provided by a host website which facilitated data transaction via basic API calls. A brief review of literature suggested that Representational State Transfer (REST) style architecture is currently favoured by software developers and would be well suited for building a versatile back-end interface for the WaterWise application. The developed application currently only interacts with the server in order to *retrieve* device data and does not provide the user with a means of device control. It is envisioned that a more sophisticated back-end would facilitate two-way communication with connected devices and improved customer-client interaction.

A number of additional functionalities could be built into the WaterWise application to improve its usefulness to the user. For example, the current version of the application only facilitates connection to one water, one flow sensor device, and implements a single set of billing tariffs. Ideally, the application should be expanded to allow users to add numerous devices and indicate which tariffs are applicable to

them based on which type of consumer they are (domestic, commercial, municipal etc.) and who their water service provider is. Furthermore, the application could be modified for compatibility with other types of devices such as pressure management devices, OCR based meter reading devices, geyser thermostats as well as smart electricity and gas meters. In South Africa, municipalities function as the utility responsible both water and electricity provision and thus it might be practical to combine the smart metering systems for these services into a single user interface, which could potentially be used by utilities to automate the meter reading process. Consideration should be given to the implications of adopting such a system, particularly the changing skillset which would be required from the work force.

Improved analytical capabilities of the WaterWise application could provide significant benefit to the user in a number of ways. For demonstration purposes, a leak detection was implemented in the application to alert the user when night-time flow exceeded a selected threshold or if unusually high flow rates were detected. However more complex algorithms or flow analysis software could be incorporated, along with machine learning tools, to identify specific end-uses which significantly contribute to their overall consumption. Data provided by such analysis tools could be further used to draw up a 'water budget' for the consumer, perhaps notifying the user if one or more end-uses exceeds its allocated consumption for the day or week.

WaterWise provides a useful platform for future research and development. Communication approaches to water demand management have been found to be very effective in reducing household consumption. Comparative norm approaches, whereby users are given feedback comparing their water usage to others in their neighbourhood or city, have been found to be particularly effective in reducing consumer water demand (Datta et al., 2015). The application was programmed to send notifications containing informative messages to encourage water conservation. The current functionality could be modified by utilities to convey conservation messages to target consumers during periods of water scarcity. This modification could also be used as a tool by researchers to gather data for testing the effects of various behavioural interventions in trial studies. It was noted in the literature review that ToU tariffs are not common in the water sector, and research is required in this field. Wide-scale implementation of smart meter applications such as WaterWise could be used not only for providing users with feedback, but also for collecting useful data which could then be used to deriving ToU tariff structures, or designing specific intervention for targeted users. Finally, future research and development should focus on data security and data privacy, to ensure that the user's best interest are protected as the smart metering industry grows.

References

All Classes. 2020. [Online].
Available: https://docs.oracle.com/javase/8/docs/api/, [Accessed 5 August 2020]

Allcott, H., 2011. Social norms and energy conservation. *Journal of public Economics*, *95*(9-10), pp.1082-1095.

Android. 2020. [Online].
Available: https://en.wikipedia.org/wiki/Android_(operating_system), [Accessed 23 July 2020]

API. 2020. [Online].
Available: https://en.wikipedia.org/wiki/api, [Accessed 13 August 2020]

Arbúes, F., Barberan, R. and Villanua, I., 2004. Price impact on urban residential water demand: A dynamic panel data approach. *Water Resources Research*, *40*(11).

Bamezai, A. and Lessick, D., 2003. Is system pressure reduction a valuable water conservation tool. *Preliminary evidence from the Irvine Ranch Water district*.

Barraqué, B., 2011. Is Individual Metering Socially Sustainable? The Case of Multifamily Housing in France. *Water Alternatives*, *4*(2), pp. 223-244

Beal, C.D., Stewart, R.A. and Fielding, K., 2013. A novel mixed method smart metering approach to reconciling differences between perceived and actual residential end use water consumption. *Journal of Cleaner Production*, *60*, pp.116-128.

Beal, C. and Flynn, J., 2014. The 2014 review of smart metering and intelligent water networks in Australia and New Zealand. *Report prepared for WSAA by the Smart Water Research Centre, Griffith University*.

Booysen, M.J., Visser, M. and Burger, R., 2019. Temporal case study of household behavioural response to Cape Town's “Day Zero” using smart meter data. *Water research*, *149*, pp.414-420.

Boyle, T., Giurco, D., Mukheibir, P., Liu, A., Moy, C., White, S. and Stewart, R., 2013. Intelligent metering for urban water: A review. *Water*, *5*(3), pp.1052-1081.

Brick, K., De Martino, S. and Visser, M., 2017. Behavioural nudges for water conservation: Experimental evidence from Cape Town. *Preprint*.

Buffalo City Metropolitan Municipality Tariff Book Index. 2020. [Online] Available: https://buffalocity.gov.za/cm/uploads/documents/, [Accessed 17 August 2020]

Burls, N.J., Blamey, R.C., Cash, B.A., Swenson, E.T., al Fahad, A., Bopape, M.J.M., Straus, D.M. and Reason, C.J., 2019. The Cape Town "Day Zero" drought and Hadley cell expansion. *Npj Climate and Atmospheric Science*, *2*(1), pp.1-8.

Carragher, B.J., Stewart, R.A. and Beal, C.D., 2012. Quantifying the influence of residential water appliance efficiency on average day diurnal demand patterns at an end use level: A precursor to optimised water service infrastructure planning. *Resources, Conservation and Recycling*, *62*, pp.81-90.

City of Cape Town: Residential Water Tariffs. 2019. [Online]. Available: http://resource.capetown.gov.za/documentcentre/documents/forms%2c%20notices%2c%20tariffs%20and%20lists/l3-watersanitationrestrictiontariffs.pdf [Accessed 7 September 2020]

City of Johannesburg: Tariffs. 2019. [Online]. Available: https://www.joburg.org.za/documents_/documents/tariffs/tariffs/coj%202019%20approved%20tariffs.pdf [Accessed 4 August 2020]

City of Tshwane: Supply of Water Tariff. 2019. [Online]. Available: http://www.tshwane.gov.za/sites/departments/public-works-and-infrastructure/watersanitation/documents/, [Accessed 11 September 2020]

Comparison Of Mobile Operating Systems. 2020. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems, [Accessed 3 June 2020]

Couvelis, F. and van Zyl, J., 2012. *Apparent water losses related to municipal metering in South Africa* (No. 1998/1, p. 12). WRC Report.

Datta, S., Miranda, J.J., Zoratto, L., Calvo-González, O., Darling, M. and Lorenzana, K., 2015. *A behavioral approach to water conservation: evidence from Costa Rica*. The World Bank.

DeOreo, W.B., Heaney, J.P. and Mayer, P.W., 1996. Flow trace analysis to access water use. *Journal-American Water Works Association*, *88*(1), pp.79-90.

Documentation for Android Developers. 2020. [Online].
Available: https://developer.android.com/docs, [Accessed 01 September 2020]

eThekweni Municipality: Water Tariffs. 2020. [Online]. Available:
http://www.durban.gov.za/resource_centre/services_tariffs/, [Accessed 29 August 2020]

Ferraro, P.J. and Price, M.K., 2013. Using nonpecuniary strategies to influence behavior: evidence from a large-scale field experiment. *Review of Economics and Statistics*, *95*(1), pp.64-73.

Gaudin, S., 2006. Effect of price information on residential water demand. *Applied economics*, *38*(4), pp.383-393.

GraphView. 2020 [Online].
Available: https://github.com/jjoe64/GraphView [Accessed 17 July 2020]

Gurung, T.R., Stewart, R.A., Beal, C.D. and Sharma, A.K., 2015. Smart meter enabled water end-use demand data: platform for the enhanced infrastructure planning of contemporary urban water supply networks. *Journal of Cleaner Production*, *87*, pp.642-654.

Heinrich, M., 2007. *Water end use and efficiency project (WEEP)*. Branz, Judgeford, New Zealand.

Hui-ting, Q. and Yi-jie, L., 2011, July. Research of peak and valley period partition approach on statistics. In *2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT)* (pp. 1774-1779). IEEE.

Hypertext Transfer Protocol. 2020. [Online].
Available: https://en.wikipedia.org/wiki/hypertext_transfer_Protocol [Accessed 2 May 2020]

ICE Catalogue. 2019. [Online].
Available: https://realtelematics.co.za/wp-content/uploads/2019/07/ice-catalogue_august2019.pdf
[Acessed 29 September 2020]

iOS. 2020. [Online]. Available: https://en.wikipedia.org/wiki/ios [Accessed 04 February 2020]

Jacobs, H.E., 2008. *A conceptual end-use model for residential water demand and return flow* (Doctoral dissertation, University of Johannesburg).

Jansen, A. and Schulz, C.E., 2006. Water demand and the urban poor: a study of the factors influencing water consumption among households in Cape Town, South Africa. *South African Journal of Economics*, *74*(3), pp.593-609.

JSON. 2020. [Online]. Available: https://en.wikipedia.org/wiki/json [Accessed 01 July 2020]

Loh, M. and Coghlan, P., 2003. *Domestic water use study in Perth, Western Australia, 1998-2001* (pp. 1-235). Perth: Water Corporation.

Lugoma, M.F.T., Van Zyl, J.E. and Ilemobade, A.A., 2012. The extent of on-site leakage in selected suburbs of Johannesburg. *Water SA*, *38*(1), pp.127-132.

M-Maji. 2012. [Online]. Available: https://mmaji.wordpress.com/ [Accessed 02 March 2020]

Makki, A.A., Stewart, R.A., Panuwatwanich, K. and Beal, C., 2013. Revealing the determinants of shower water end use consumption: enabling better targeted urban water conservation strategies. *Journal of Cleaner Production*, *60*, pp.129-146.

Mangaung Metropolitan Municipality: General Tariffs. 2019. [Online].
Available: http://www.mangaung.co.za/wp-content/uploads/, [Accessed 13 August 2020]

Mayer, P.W., DeOreo, W.B., Opitz, E.M., Kiefer, J.C., Davis, W.Y., Dziegielewski, B. and Nelson, J.O., 1999. Residential end uses of water..

McKenzie, R.D. and Lambert, A.O., 2002. ECONOLEAK user guide. *South Africa Water Research Commission, WRC Report TT169/02*.

Meyer, D.N., 2018. *The effect of controlled pressure adjustment on consumer water demand in an urban water distribution system* (Masters dissertation, Stellenbosch: Stellenbosch University).

Mobile Operating System Market Share. 2020. [Online].
Available: https://gs.statcounter.com/os-market-share/mobile/ [Accessed 22 June 2020]

Mudumbe, M.J. and Abu-Mahfouz, A.M., 2015, July. Smart water meter system for user-centric consumption measurement. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)* (pp. 993-998). IEEE.

Nava-Ortiz, M., Gómez, W. and Díaz-Pérez, A., 2011, October. Digit recognition system for camera mobile phones. In *2011 8th International Conference on Electrical Engineering, Computing Science and Automatic Control* (pp. 1-5). IEEE.

Ncube, M. and Taigbenu, A.E., 2016. Consumption characterisation and on-site leakage in Johannesburg, South Africa. In *Proceedings of the IWA Water Loss Conference 2016*.

Nel, P.J.C., Booysen, M.J. and Van Der Merwe, B., 2014. ICT-enabled solutions for smart management of water supply in Africa.

Nelson Mandela Bay Municipality Tariff Book Index. 2020. [Online].
Available: http://nelsonmandelabay.gov.za/datarepository/documents/, [Accessed 15 August 2020]

Nique, M. and Opala, K., 2014. The synergies between mobile, energy and water access: Africa. *Energy*, *32*(34), p.36.

Praskievicz, S. and Chang, H., 2009. Identifying the relationships between urban water consumption and weather variables in Seoul, Korea. *Physical Geography*, *30*(4), pp.324-337.

Ramos, A., Gago, A., Labandeira, X. and Linares, P., 2015. The role of information for energy efficiency in the residential sector. *Energy Economics*, *52*, pp.S17-S29.

Reason, C.J.C., Rouault, M., Melice, J.L. and Jagadheesha, D., 2002. Interannual winter rainfall variability in SW South Africa and large scale ocean–atmosphere interactions. *Meteorology and Atmospheric Physics*, *80*(1-4), pp.19-29.

Representational State Transfer. 2020. [Online].
Available: https://en.wikipedia.org/wiki/representational_state_transfer, [Accessed 11 July 2020]

Roberts, P. 2005. 2004 Residential End Use Measurement Study, *Yarra Valley Water Ltd, Mitcham, Victoria*

Smith, J.A., 2010. How much water is enough? Domestic metered water consumption and free basic water volumes: The case of Eastwood, Pietermaritzburg. *Water SA*, *36*(5).

Sousa, P.M., Blamey, R.C., Reason, C.J., Ramos, A.M. and Trigo, R.M., 2018. The 'Day Zero'Cape Town drought and the poleward migration of moisture corridors. *Environmental Research Letters*, *13*(12)

Stellenbosch Municipality: Tariffs, 2019. [Online].
Available: https://www.stellenbosch.gov.za/documents/, [Accessed 12 September 2020]

Sterne, N.L., 2019. *Characterisation of household water use events using a non-intrusive sensor* (Doctoral dissertation, Stellenbosch: Stellenbosch University).

Stewart, R.A., Willis, R.M., Panuwatwanich, K. and Sahin, O., 2013. Showering behavioural response to alarming visual display monitors: longitudinal mixed method study. *Behaviour & Information Technology*, *32*(7), pp.695-711.

Veck, G.A. and Bill, M.R., 2000. Estimation of the residential price elasticity of demand for water by means of a contingent valuation approach. *Water Research Commission Report*, (790/1), p.00.

Venizelou, V., Philippou, N., Hadjipanayi, M., Makrides, G., Efthymiou, V. and Georghiou, G.E., 2018. Development of a novel time-of-use tariff algorithm for residential prosumer price-based demand side management. *Energy*, *142*, pp.633-646.

Willis, R.M., Stewart, R.A., Giurco, D.P., Talebpour, M.R. and Mousavinejad, A., 2013. End use water consumption in households: impact of socio-demographic factors and efficient devices. *Journal of Cleaner Production*, *60*, pp.107-115..

Willis, R.M., Stewart, R.A., Panuwatwanich, K., Williams, P.R. and Hollingsworth, A.L., 2011. Quantifying the influence of environmental and water conservation attitudes on household end use water consumption. *Journal of environmental management*, *92*(8), pp.1996-2009.

World Health Organization (WHO), 2003. Domestic water quantity, service level and health. *WHO, Geneva, Switzerland*.

Xia, F., Yang, L.T., Wang, L. and Vinel, A., 2012. Internet of things. *International journal of communication systems*, *25*(9), p.1101.

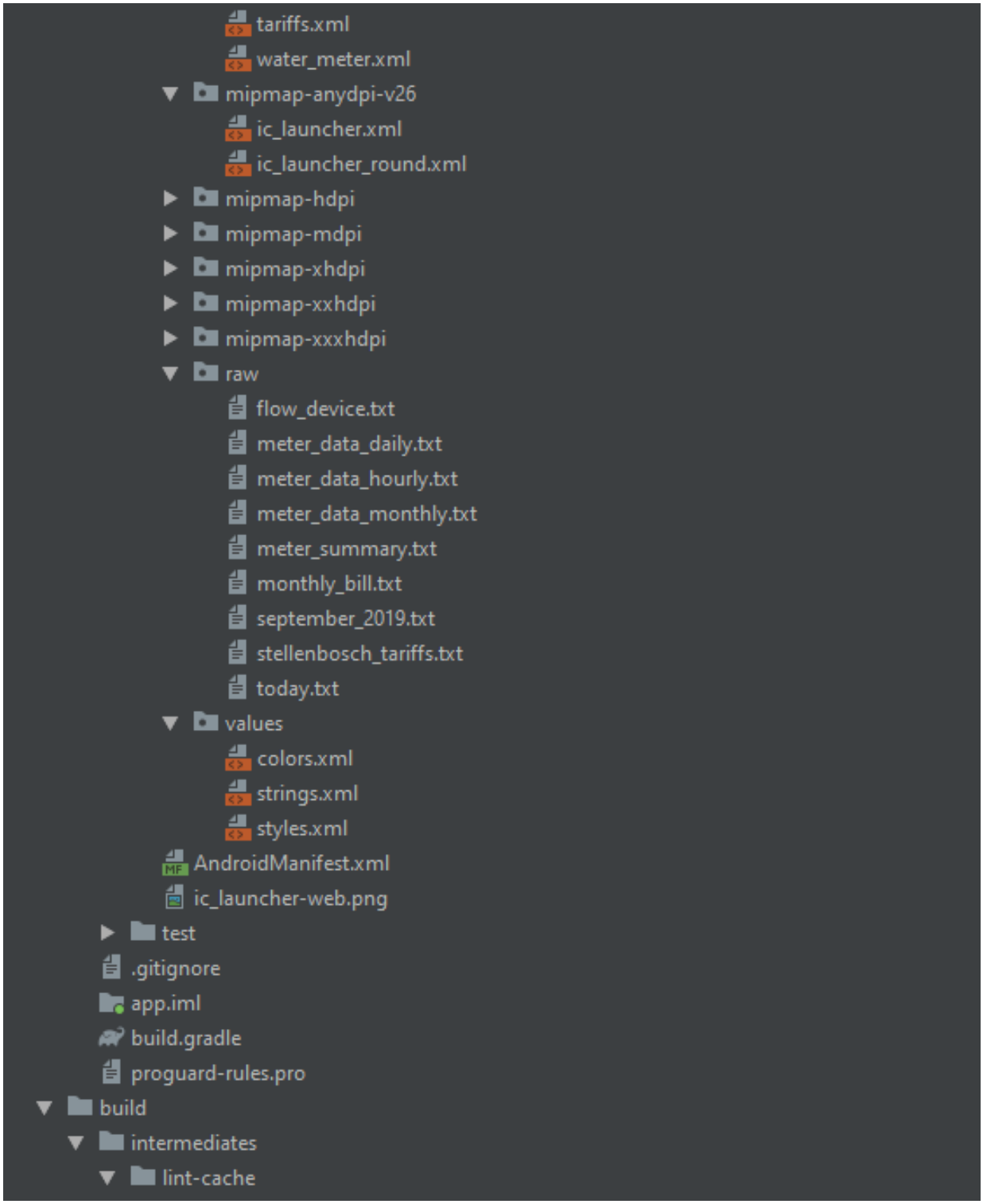
XML. 2020. [Online]. Available: https://en.wikipedia.org/wiki/xml [Accessed 18 March 2020]

Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M., 2014. Internet of things for smart cities. *IEEE Internet of Things journal*, *1*(1), pp.22-32.

.

Ziervogel, G., 2019. Unpacking the Cape Town drought: lessons learned. *Cities support programme| Climate resilience paper. African Centre for Cities, Febraury*.

Appendix A – WaterWise Project File Structure



Appendix B: Android Manifest

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="release.waterwise"
    android:versionCode="4"
    android:versionName="4.0">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:label="WaterWise"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"

        android:usesCleartextTraffic="true"
        android:largeHeap="true">

        <activity android:name="release.waterwise.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="release.waterwise.Flow_Device.FlowDevice" > </activity>

        <activity android:name="release.waterwise.WaterMeter" ></activity>

        <activity android:name="release.waterwise.Tariffs" ></activity>

        <activity android:name="release.waterwise.Graphs" > </activity>

        <activity android:name="release.waterwise.Customer" > </activity>

    </application>

</manifest>
```

Appendix C – Java Classes

MainActivity.java

```
package release.waterwise;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.BarGraphSeries;
import com.jjoe64.graphview.series.DataPoint;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import release.waterwise.Flow_Device.FlowDevice;

public class MainActivity extends AppCompatActivity {

    Button btnGraphs;
    Button btnFlowDevice;
    Button btnTariffs;
    Button btnWaterMeter;
    Button btnCustomer;
    Button btnUpdate;

    GraphView graph;

    // Shared preferences variables
    //--------------------------------------------------------------------
    public SharedPreferences sharedPreferences;

    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    // Load sharedPreferences
    //--------------------------------------------------------------------
    sharedPreferences = this.getSharedPreferences("release.waterwise", MODE_PRIVATE);

    // Create active buttons
    //--------------------------------------------------------------------
    btnGraphs = findViewById(R.id.buttonGraphViewer);
    btnFlowDevice = findViewById(R.id.buttonFlowDevices);
    btnTariffs = findViewById(R.id.buttonWaterBill);
    btnWaterMeter = findViewById(R.id.buttonWaterMeter);
    btnCustomer = findViewById(R.id.buttonCustomer);
    btnUpdate = findViewById(R.id.buttonUpdate);

    // Create graph
    //--------------------------------------------------------------------
    graph = findViewById(R.id.mainMenuGraph);
    updateCurrentDailyGraph();

    // Initiate button OnClickListeners
    //--------------------------------------------------------------------
    viewFlowSensor();
    viewTariffs();
    viewWaterMeter();
    viewDetailedGraphs();
    viewCustomerDetails();
    updateData();
}

public void updateData(){
    Log.d("Device model data update method", "Initiated");

    btnUpdate.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    //  Retrieve Water Meter Data from server
                    try {
                        DownloadTask task = new DownloadTask();
                        task.execute("www.pastebin.com/raw/mbPfQ4UC");
                    }
                    catch (Exception e) {
                        Log.d("updateData: ", "Failed to update");
                        e.printStackTrace();
                    }

                    //  Retrieve Flow Device Data from server
                    try {
                        DownloadTask task = new DownloadTask();
                        task.execute("www.pastebin.com/raw/KzsDeuRy");
```

```
                }
                catch (Exception e) {
                    Log.d("updateData: ", "Failed to update");
                    e.printStackTrace();
                }

                //  Retrieve miscellaneous data from server
                try {
                    DownloadTask task = new DownloadTask();
                    task.execute("www.pastebin.com/raw/aJdolEryi");

                }
                catch (Exception e) {
                    Log.d("updateData: ", "Failed to update");
                    e.printStackTrace();
                }
            }
        }
    );

}


private class DownloadTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {

        String result = "";
        URL url;
        HttpURLConnection urlConnection = null;

        try {
            url = new URL(urls[0]);

            urlConnection = (HttpURLConnection) url.openConnection();
            InputStream in = urlConnection.getInputStream();
            InputStreamReader reader = new InputStreamReader(in);

            int data = reader.read();

            while (data != -1) {
                char current = (char) data;
                result += current;
                data = reader.read();
            }

            return result;

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    return null;
}

@Override
protected void onPostExecute(String result) {
    super.onPostExecute(result);

    //  ----------------------------------------
    //  Text files to be written
    //  ----------------------------------------
    //  1.) flow_device.txt
    //  2.) meter_data_daily.txt
    //  3.) meter_data_hourly.txt
    //  4.) meter_data_monthly.txt
    //  5.) meter_summary.txt
    //  6.) monthly_bill.txt
    //  7.) september_2020.txt
    //  8.) stellenbosch_tariffs.txt
    //  9.) today.txt
    //  ----------------------------------------


    //  1.) flow_device.txt
    try {
        FileOutputStream os = new FileOutputStream("flow_device.txt");
        OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
        BufferedWriter bufferedWriter = new BufferedWriter(osw);

        bufferedWriter.write(result);
        bufferedWriter.close();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    //  2.) meter_data_daily.txt
    try {
        FileOutputStream os = new FileOutputStream("meter_data_daily.txt");
        OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
        BufferedWriter bufferedWriter = new BufferedWriter(osw);

        bufferedWriter.write(result);
        bufferedWriter.close();

    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
//  3.) meter_data_hourly.txt
try {
    FileOutputStream os = new FileOutputStream("meter_data_hourly.txt");
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);

    bufferedWriter.write(result);
    bufferedWriter.close();

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

//  4.) meter_data_monthly.txt
try {
    FileOutputStream os = new FileOutputStream("meter_data_monthly.txt");
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);

    bufferedWriter.write(result);
    bufferedWriter.close();
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

//  5.) meter_summary.txt
try {
    FileOutputStream os = new FileOutputStream("flow_device.txt");
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);

    bufferedWriter.write(result);
    bufferedWriter.close();

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

//  6.) monthly_bill.txt
try {
    FileOutputStream os = new FileOutputStream("monthly_bill.txt");
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);
```

```
    bufferedWriter.write(result);
    bufferedWriter.close();

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

//  7.) september_2020.txt
try {
    FileOutputStream os = new FileOutputStream("september_2020.txt");
    OutputStreamWriter os = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);

    bufferedWriter.write(result);
    bufferedWriter.close();

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

//  8.) stellenbosch_tariffs.txt
try {
    FileOutputStream os = new FileOutputStream("stellenbosch_tariffs.txt");
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);

    bufferedWriter.write(result);
    bufferedWriter.close();

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

//  9.) today.txt
try {
    FileOutputStream os = new FileOutputStream("today.txt");
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-16");
    BufferedWriter bufferedWriter = new BufferedWriter(osw);

    bufferedWriter.write(result);
    bufferedWriter.close();
```

```
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}


// Button OnClickListeners
//----------------------------------------------------------------------
public void viewFlowSensor(){
    btnFlowDevice.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent = new Intent(getApplicationContext(), FlowDevice.class);
                    startActivity(intent);
                }
            }
    );
}

public void viewWaterMeter(){
    btnWaterMeter.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Log.d("Water Meter Button", "...clicked...");
                    Intent intent = new Intent(getApplicationContext(), WaterMeter.class);
                    startActivity(intent);
                }
            }
    );
}

public void viewTariffs(){
    btnTariffs.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent = new Intent(getApplicationContext(), Tariffs.class);
                    startActivity(intent);
                }
            }
    );
}

public void viewDetailedGraphs(){
    btnGraphs.setOnClickListener(
            new View.OnClickListener() {
                @Override
```

```
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), Graphs.class);
                startActivity(intent);
            }
        }
    );
}

public void viewCustomerDetails(){
    btnCustomer.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), Customer.class);
                startActivity(intent);
            }
        }
    );
}

// Update graph to most recent data
//------------------------------------------------------------------------
public void updateCurrentDailyGraph(){

    String jsonStringToday = null;

    // set manual X bounds
    graph.getViewport().setXAxisBoundsManual(true);
    graph.getViewport().setMinX(0);
    graph.getViewport().setMaxX(24);

    // set manual Y bounds
    graph.getViewport().setYAxisBoundsManual(true);
    graph.getViewport().setMinY(0);
    graph.getViewport().setMaxY(75);

    // activate zooming and scrolling
    graph.getViewport().setScalable(true);
    graph.getViewport().setScrollable(true);
    graph.getViewport().setScalableY(true);
    graph.getViewport().setScrollableY(true);

    // set titles
    graph.setTitle("09 October 2020");
    graph.getGridLabelRenderer().setHorizontalAxisTitle(" Time of Day [hh]");
    graph.getGridLabelRenderer().setVerticalAxisTitle(" Cumulative Usage [L]");

    //------------------------------------------------------------------------
    //      Read data text files
    //------------------------------------------------------------------------

    try {

        InputStream is = this.getResources().openRawResource(R.raw.today);
        int size = is.available();
```

```
            byte[] buffer = new byte[size];
            is.read(buffer);
            is.close();

            jsonStringToday = new String(buffer, "UTF-8");
            Log.d(" 1....Testing JSON String", jsonStringToday);

            // Convert to array
            //---------------------------------------------------------------------

            JSONObject jsonbject = new JSONObject(jsonStringToday);

            JSONArray jsonArray = jsonbject.getJSONArray("Today");

            // Create data points from JSON array
            //---------------------------------------------------------------------

            DataPoint[] set = new DataPoint[24];

            for (int i = 0; i < jsonArray.length(); i++) {
               JSONObject jsonOb = jsonArray.getJSONObject(i);

               set[i] = new DataPoint(Double.parseDouble(jsonOb.get("Hour").toString()),
Double.parseDouble(jsonOb.get("dL").toString()));

               Log.d(" 2....Testing JSON Date String", " " + i);
            }

            // Add points to graph
            //---------------------------------------------------------------------
            BarGraphSeries<DataPoint> series = new BarGraphSeries<DataPoint>(set);
            graph.addSeries(series);

         } catch (IOException | JSONException ex) {
            ex.printStackTrace();
         }
      }
   }
```

WaterMeter.java

```
package release.waterwise;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;


public class WaterMeter extends AppCompatActivity {

    // Shared preferences variables
    //----------------------------------------------------------------------
    public SharedPreferences sharedPreferences;
    protected String spMeterBaseURL;
    protected String spMeterApiKey;
    //----------------------------------------------------------------------

    Button backButton;
    Button save;

    EditText meterBaseURL;
    EditText meterApiKey;

    TextView deviceID;
    TextView lastUpdate;
    TextView lastReading;
    TextView currentUsed;
    TextView currentBill;
    TextView projectedConsumption;
    TextView projectedBill;
    TextView currentBlock;
    TextView currentRate;

    JSONObject meterSummary = null;

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.water_meter);

        // Load sharedPreferences
```

```
    //----------------------------------------------------------------------
    sharedPreferences = this.getSharedPreferences("release.waterwise", MODE_PRIVATE);
    spMeterBaseURL = sharedPreferences.getString("meterBaseURL", "www.pastebin.com/raw/");
    spMeterApiKey = sharedPreferences.getString("meterApiKey", "mbPfQ4UC");

    backButton = findViewById(R.id.back2MainMenu);
    save = findViewById(R.id.save1);

    meterBaseURL = findViewById(R.id.meterURL);
    meterApiKey = findViewById(R.id.meterApiKey);

    deviceID = findViewById(R.id.textViewMeterID);
    lastUpdate = findViewById(R.id.textViewUpdated);
    lastReading = findViewById(R.id.textViewCurrentReading);
    currentUsed = findViewById(R.id.textViewWaterUsed);
    currentBill = findViewById(R.id.textViewBillToDate);
    projectedConsumption = findViewById(R.id.textViewProjectedUsage);
    projectedBill = findViewById(R.id.textViewProjectedBill);
    currentBlock = findViewById(R.id.textViewCurrentBlock);
    currentRate = findViewById(R.id.textViewCurrentRate2);

    backToMain();
    updateSharedPreferences();
}



public void backToMain() {
    backButton.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                    startActivity(intent);
                }
            });
}

public void updateSharedPreferences(){

    save.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    spMeterBaseURL = meterBaseURL.getText().toString();
                    sharedPreferences.edit().putString("meterBaseURL", spMeterBaseURL).apply();

                    spMeterApiKey = meterApiKey.getText().toString();
                    sharedPreferences.edit().putString("meterApiKey", spMeterApiKey).apply();
                }
            });
}

public void loadData() {
```

```
        String jsonString = null;
        InputStream is = this.getResources().openRawResource(R.raw.meter_summary);

        // Read data text files
        //---------------------------------------------------------------------------

        try {

            int size = is.available();
            byte[] buffer = new byte[size];
            is.read(buffer);
            is.close();

            jsonString = new String(buffer, "UTF-8");
            Log.d(" 1....Testing JSON String", jsonString);

            meterSummary = new JSONObject(jsonString);

            deviceID.setText(meterSummary.get("Device ID").toString());
            lastUpdate.setText(meterSummary.get("Last Updated").toString());
            lastReading.setText(meterSummary.get("Meter Reading").toString());
            currentUsed.setText(meterSummary.get("Water Used This Month").toString());
            currentBill.setText(meterSummary.get("Current Bill").toString());
            projectedConsumption.setText(meterSummary.get("Projected Consumption").toString());
            projectedBill.setText(meterSummary.get("Projected Bill").toString());
            currentBlock.setText(meterSummary.get("Current Block").toString());
            currentRate.setText(meterSummary.get("Current Rate").toString());

        } catch (IOException | JSONException ex) {
            ex.printStackTrace();
        }

    }
}
```

FlowDevice.java

```
package release.waterwise.Flow_Device;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import release.waterwise.DataListAdapter;
import release.waterwise.MainActivity;
import release.waterwise.R;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;

import java.util.ArrayList;

public class FlowDevice extends AppCompatActivity{

    protected JSONArray flowDeviceArray;
    protected ArrayList<String> deviceData;

    private RecyclerView recyclerView;

    protected Button btnUpdateData;
    protected Button btnBack;
    protected Button btnSave;

    protected TextView deviceDescriptionEditText;
    protected TextView deviceURLEditText;
    protected TextView deviceAPIEditText;

    // Shared preferences variables
    //--------------------------------------------------------------------
    public SharedPreferences sharedPreferences;

    protected String deviceDescription;
    protected String deviceBaseURL;
    protected String deviceApiKey;
```

```
protected boolean deviceInitialized;

//----------------------------------------------------------------------

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.flow_device);

   sharedPreferences = this.getSharedPreferences("release.waterwise", MODE_PRIVATE);

   deviceDescription = sharedPreferences.getString("deviceDescription", "Home");
   deviceBaseURL = sharedPreferences.getString("deviceBaseURL", "http://pastebin.com/raw/");
   deviceApiKey = sharedPreferences.getString("deviceApiKey", "KzsDeuRy");
   deviceInitialized = sharedPreferences.getBoolean("deviceInitialized", false);

   btnUpdateData = findViewById(R.id.dataUpdateButton);
   btnBack = findViewById(R.id.backDeviceView2DeviceManager);

   deviceDescriptionEditText = findViewById(R.id.deviceDescription);
   deviceURLEditText = findViewById(R.id.deviceURL);
   deviceAPIEditText = findViewById(R.id.deviceApiKey);

   btnSave = findViewById(R.id.save);

   deviceDescriptionEditText.setText(deviceDescription);
   deviceURLEditText.setText(deviceBaseURL);
   deviceAPIEditText.setText(deviceApiKey);

   // Initialize buttons
   saveSharedPreferences();
   updateData();
   back();

}

public void updateData(){
   btnUpdateData.setOnClickListener(
         new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               loadData();
            }
         }
   );
}

public void loadData(){

   Log.d("loadData()", "...method initiated...");

   try {
      String jsonStringFlowDevice;

      InputStream is = this.getResources().openRawResource(R.raw.flow_device);
```

```
            int size = is.available();
            byte[] buffer = new byte[size];
            is.read(buffer);
            is.close();

            jsonStringFlowDevice = new String(buffer, "UTF-8");

            // Convert to array
            //-------------------------------------------------------------------------

            JSONObject jsonbject = new JSONObject(jsonStringFlowDevice);
            flowDeviceArray = jsonbject.getJSONArray("Sample");

        }
        catch (IOException | JSONException ex) {
            ex.printStackTrace();
        }


        updateArrayList();

    }

    public void updateArrayList() {

        deviceData = new ArrayList<>();

        JSONObject temp = null;

        if(flowDeviceArray.length() == 0){
            // Show message
            showMessage("Error", "Database empty");
            return;
        }

        for (int i = 0; i < flowDeviceArray.length(); i++){

            try {
                temp = flowDeviceArray.getJSONObject(i);
            }
            catch (JSONException e) {
                e.printStackTrace();
            }

            try {

deviceData.add(temp.get("Date").toString()+","+temp.get("Time")+","+temp.get("Duration")+","
+temp.get("Temperature"));
            }
            catch (JSONException e) {
                e.printStackTrace();
            }
        }

        loadRecyclerViewData();
```

```
    }

    public void loadRecyclerViewData(){

        DataListAdapter dataListAdapter = new DataListAdapter(this, deviceData);
        recyclerView = findViewById(R.id.deviceRecycvlerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setAdapter(dataListAdapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
    }

    public void back(){
        btnBack.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                        startActivity(intent);
                    }
                }
        );
    }

    public void showMessage(String title, String message){

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setCancelable(true);
        builder.setTitle(title);
        builder.setMessage(message);
        builder.show();
    }

    public void saveSharedPreferences(){

        btnSave.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {

                        deviceDescription = deviceDescriptionEditText.getText().toString();
                        sharedPreferences.edit().putString("deviceDescription", deviceDescription).apply();

                        deviceBaseURL = deviceURLEditText.getText().toString();
                        sharedPreferences.edit().putString("deviceBaseURL", deviceBaseURL).apply();

                        deviceApiKey = deviceAPIEditText.getText().toString();
                        sharedPreferences.edit().putString("deviceApiKey", deviceApiKey).apply();
                    }
                }
        );

    }
}
```

Customer.java

```
package release.waterwise;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;

public class Customer extends AppCompatActivity{

    // Shared preferences variables
    //--------------------------------------------------------------------
    public SharedPreferences sharedPreferences;
    protected String spFirstName;
    protected String spSurname;
    protected String spCellphone;
    protected String spEmail;
    protected String spMeterID;
    protected String spAddress1;
    protected String spAddress2;
    protected String spAddress3;
    protected String spAddress4;
    //--------------------------------------------------------------------


    EditText firstName;
    EditText surname;
    EditText cellphone;
    EditText email;
    EditText meterID;
    EditText address1;
    EditText address2;
    EditText address3;
    EditText address4;

    Button back;
    Button save;

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.customer);

        firstName = findViewById(R.id.editTextFirstName);
        surname = findViewById(R.id.editTextSurname);
        cellphone = findViewById(R.id.editTextCellphone);
        email = findViewById(R.id.editTextEmail);
        meterID = findViewById(R.id.editTextMeterID);
        address1 = findViewById(R.id.editTextAddressLine1);
        address2 = findViewById(R.id.editTextAddressLine2);
```

```
    address3 = findViewById(R.id.editTextAddressLine3);
    address4 = findViewById(R.id.editTextAddressLine4);

    sharedPreferences = this.getSharedPreferences("release.waterwise", MODE_PRIVATE);

    spFirstName = sharedPreferences.getString("firstName", "Vaughan");
    spSurname = sharedPreferences.getString("surname", "Warren");
    spCellphone = sharedPreferences.getString("cellphone", "086 12345678");
    spEmail = sharedPreferences.getString("email", "vaughan@emailaddress.com");
    spMeterID = sharedPreferences.getString("meterID", "Stell123456");
    spAddress1 = sharedPreferences.getString("address1", "17 Rhodes-Noord");
    spAddress2 = sharedPreferences.getString("address2", "Die Boord");
    spAddress3 = sharedPreferences.getString("address3", "Stellenbosch");
    spAddress4 = sharedPreferences.getString("address4", "7600");

    save = findViewById(R.id.save);
    back = findViewById(R.id.backButton2mainMenu);

    updateSharedPreferences();
    back();
}

public void updateSharedPreferences(){

    save.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    spFirstName = firstName.getText().toString();
                    sharedPreferences.edit().putString("firstName", spFirstName).apply();

                    spSurname = surname.getText().toString();
                    sharedPreferences.edit().putString("surname", spSurname).apply();

                    spCellphone = cellphone.getText().toString();
                    sharedPreferences.edit().putString("cellphone", spCellphone).apply();

                    spEmail = email.getText().toString();
                    sharedPreferences.edit().putString("email", spEmail).apply();

                    spMeterID = meterID.getText().toString();
                    sharedPreferences.edit().putString("meterID", spMeterID).apply();

                    spAddress1 = address1.getText().toString();
                    sharedPreferences.edit().putString("address1", spAddress1).apply();

                    spAddress2 = address2.getText().toString();
                    sharedPreferences.edit().putString("address2", spAddress2).apply();

                    spAddress3 = address3.getText().toString();
                    sharedPreferences.edit().putString("address3", spAddress3).apply();

                    spAddress4 = address4.getText().toString();
                    sharedPreferences.edit().putString("address4", spAddress4).apply();
                }
```

```
            }
        );
    }

    public void back(){
        back.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                        startActivity(intent);
                    }
                }
        );
    }

}
```

Tariffs.java

```
package release.waterwise;

import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;

public class Tariffs extends AppCompatActivity {

    TextView rate1;
    TextView rate2;
    TextView rate3;
    TextView rate4;
    TextView rate5;
    TextView rate6;
    TextView rate7;

    JSONObject tariffs = null;

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.tariffs);

        rate1 = findViewById(R.id.tariffText2);
        rate2 = findViewById(R.id.tariffText3);
        rate3 = findViewById(R.id.tariffText4);
        rate4 = findViewById(R.id.tariffText5);
        rate5 = findViewById(R.id.tariffText6);
        rate6 = findViewById(R.id.tariffText7);
        rate7 = findViewById(R.id.tariffText8);

        loadData();
    }

    public void loadData() {

        String jsonString = null;
        InputStream is = this.getResources().openRawResource(R.raw.stellenbosch_tariffs);

        // Read data text files
        //--------------------------------------------------------------------------
        try {
            int size = is.available();
            byte[] buffer = new byte[size];
```

```
            is.read(buffer);
            is.close();

            jsonString = new String(buffer, "UTF-8");
            Log.d(" 1....Testing JSON String", jsonString);
            tariffs = new JSONObject(jsonString);

            rate1.setText(tariffs.get("0 - 6 kL").toString());
            rate2.setText(tariffs.get("6 - 12 kL").toString());
            rate3.setText(tariffs.get("12 - 18 kL").toString());
            rate4.setText(tariffs.get("18 - 25 kL").toString());
            rate5.setText(tariffs.get("25 - 40 kL").toString());
            rate6.setText(tariffs.get("40 - 70 kL").toString());
            rate7.setText(tariffs.get("70 kL +").toString());

        }
        catch (IOException | JSONException ex) {
            ex.printStackTrace();
        }
    }
}
```

Graphs.java

```
package release.waterwise;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.BarGraphSeries;
import com.jjoe64.graphview.series.DataPoint;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.InputStream;

public class Graphs extends AppCompatActivity {

    JSONArray meterDataDaily;
    JSONArray meterDataMonthly;

    Button dailyData;
    Button monthlyData;
    Button back;

    GraphView graph;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.graphs);

        dailyData = findViewById(R.id.buttonHistoricalDaily);
        monthlyData = findViewById(R.id.buttonHistoricalMonthly);
        back = findViewById(R.id.backButton2mainMenu);

        graph = findViewById(R.id.graph1);
        graph.removeAllSeries();


        // Initiate button OnClickListeners
        //----------------------------------------------------------------------
        backToMain();
        dailyGraph();
        monthlyGraph();

    }

    // Button OnClickListeners
```

```
//-----------------------------------------------------------------------
public void backToMain() {
   back.setOnClickListener(
         new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               Intent intent = new Intent(getApplicationContext(), MainActivity.class);
               startActivity(intent);
            }
         }
   );
}

public void dailyGraph() {
   dailyData.setOnClickListener(

         new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               loadDailyData();
            }
         }
   );
}
public void monthlyGraph() {
   monthlyData.setOnClickListener(
         new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               loadMonthlyData();
            }
         }
   );
}

public void loadDailyData() {

   String jsonString = null;
   InputStream daily = null;

   // Graph settings
   //-----------------------------------------------------------------------

   graph.removeAllSeries();

   // set manual X bounds
   graph.getViewport().setXAxisBoundsManual(true);
   graph.getViewport().setMinX(0);
   graph.getViewport().setMaxX(31);

   // set manual Y bounds
   graph.getViewport().setYAxisBoundsManual(true);
   graph.getViewport().setMinY(0);
   graph.getViewport().setMaxY(1500);
```

```
        // set titles
        graph.setTitle("September 2019 Water Usage");
        graph.getGridLabelRenderer().setHorizontalAxisTitle(" Day of Month ");
        graph.getGridLabelRenderer().setVerticalAxisTitle(" Water Used [L]");

        //-------------------------------------------------------------------------
        //      Read data text files
        //-------------------------------------------------------------------------

        daily = this.getResources().openRawResource(R.raw.september_2019);

        try {

            int size = daily.available();
            byte[] buffer = new byte[size];
            daily.read(buffer);
            daily.close();
            jsonString = new String(buffer, "UTF-8");


            //-------------------------------------------------------------------------
            //      Convert to array
            //-------------------------------------------------------------------------
            JSONObject jsonObject = new JSONObject(jsonString);
            meterDataDaily = jsonObject.getJSONArray("Daily");

            // Create data points from JSON array
            //-------------------------------------------------------------------------
            DataPoint[] set2 = new DataPoint[meterDataDaily.length()];

            for (int i = 0; i < meterDataDaily.length(); i++) {
                JSONObject jsonOb = meterDataDaily.getJSONObject(i);
                set2[i] = new DataPoint(Double.parseDouble(jsonOb.get("Day").toString()),
Double.parseDouble(jsonOb.get("dL").toString()));
            }

            // Add points to series
            //-------------------------------------------------------------------------
            BarGraphSeries<DataPoint> series2 = new BarGraphSeries<DataPoint>(set2);

            // Create graph and add series
            //-------------------------------------------------------------------------
            graph.addSeries(series2);

        }
        catch (IOException | JSONException ex) {  ex.printStackTrace();        }
    }

    public void loadMonthlyData() {

        String jsonString = null;
        InputStream monthly = null;

        graph.removeAllSeries();
```

```
        // set manual X bounds
        graph.getViewport().setXAxisBoundsManual(true);
        graph.getViewport().setMinX(0);
        graph.getViewport().setMaxX(13);

        // set manual Y bounds
        graph.getViewport().setYAxisBoundsManual(true);
        graph.getViewport().setMinY(20);
        graph.getViewport().setMaxY(30);

        // set titles
        graph.setTitle("2019 Monthly Water Consumption");
        graph.getGridLabelRenderer().setHorizontalAxisTitle(" Month ");
        graph.getGridLabelRenderer().setVerticalAxisTitle(" Total Usage[kL]");

        //-------------------------------------------------------------------------
        //      Read data text files
        //-------------------------------------------------------------------------
        monthly = this.getResources().openRawResource(R.raw.meter_data_monthly);

        try {
            int size = monthly.available();
            byte[] buffer = new byte[size];
            monthly.read(buffer);
            monthly.close();
            jsonString = new String(buffer, "UTF-8");

            //-------------------------------------------------------------------------
            //      Convert to array
            //-------------------------------------------------------------------------
            JSONObject jsonObject = new JSONObject(jsonString);
            meterDataMonthly = jsonObject.getJSONArray("Monthly");

           DataPoint[] set = new DataPoint[12];

            for (int i = 0; i < meterDataMonthly.length(); i++) {
                JSONObject jsonOb = meterDataMonthly.getJSONObject(i);
                set[i] = new DataPoint(Double.parseDouble(jsonOb.get("Month").toString()),
Double.parseDouble(jsonOb.get("kL").toString()));
            }
            // Add points to series
            //-------------------------------------------------------------------------
            BarGraphSeries<DataPoint> series = new BarGraphSeries<DataPoint>(set);

            // Create graph and add series
            //-------------------------------------------------------------------------
            graph.addSeries(series);
            } catch (IOException | JSONException ex) {        ex.printStackTrace();           }
    }
}
```

DataListAdapter.java

```
package release.waterwise;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class DataListAdapter extends RecyclerView.Adapter<DataListAdapter.DataViewHolder>{

    private Context context;
    private ArrayList<String> deviceData;

    public DataListAdapter(Context context, ArrayList<String> deviceData){
        this.context = context;
        this.deviceData = deviceData;
    }

    public class DataViewHolder extends RecyclerView.ViewHolder{

        TextView dateTextView, timeTextView, durationTextView, temperatureTextView;
        LinearLayout dataEntryParentLayout;

        public DataViewHolder(@NonNull View itemView) {
            super(itemView);

            this.dateTextView = itemView.findViewById(R.id.dateTextView);
            this.timeTextView = itemView.findViewById(R.id.timeTextView);
            this.durationTextView = itemView.findViewById(R.id.durationTextView);
            this.temperatureTextView = itemView.findViewById(R.id.temperatureTextView);
            this.dataEntryParentLayout = itemView.findViewById(R.id.dataEntryParentLayout);
        }
    }

    @NonNull
    @Override
    public DataListAdapter.DataViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.flow_device_database_table_view, parent,
false);
        DataViewHolder dataViewHolder = new DataViewHolder(view);

        return dataViewHolder;
    }

    @Override
```

```
    public void onBindViewHolder(@NonNull DataListAdapter.DataViewHolder holder, int position) {

        String[] temp = deviceData.get(position).split(",");

        holder.dateTextView.setText(temp[0]);
        holder.timeTextView.setText(temp[1]);
        holder.durationTextView.setText(temp[2]);
        holder.temperatureTextView.setText(temp[3]);
    }

    @Override
    public int getItemCount() {
        if(deviceData != null) return deviceData.size();

        else return 0;
    }
}
```

Appendix D – Layout Files

Activity_main.xml

```
?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/heading"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#3F51B5"
            android:padding="8dp"
            android:text="MAIN MENU"
            android:textAlignment="center"
            android:textColor="#FFFFFF"
            android:textSize="18sp"
            android:textStyle="bold" />

        <com.jjoe64.graphview.GraphView
            android:id="@+id/mainMenuGraph"
            android:layout_width="match_parent"
            android:layout_height="200dip"
            android:layout_margin="10dp"
            android:layout_weight="1"
            android:paddingRight="5dp" />

        <TableLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TableRow
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TextView
                    android:id="@+id/lastUpdated"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
```

```
            android:paddingRight="1dp"
            android:text="Last Updated: 12:03:45 09/10/2020"
            android:textAlignment="textEnd" />

        <Button
            android:id="@+id/buttonUpdate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:minHeight="16dp"
            android:text="Update Data" />
    </TableRow>
</TableLayout>

<Button
    android:id="@+id/buttonWaterMeter"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="1dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_weight="0"
    android:minHeight="30dp"
    android:text="Water Meter"
    android:textColor="#3F51B5"
    android:textStyle="bold" />

<Button
    android:id="@+id/buttonFlowDevices"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="1dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_weight="0"
    android:minHeight="30dp"
    android:text="Flow Device"
    android:textColor="#3F51B5"
    android:textStyle="bold" />

<Button
    android:id="@+id/buttonGraphViewer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="1dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_weight="0"
    android:minHeight="30dp"
    android:text="View Graphs"
    android:textColor="#3F51B5"
    android:textStyle="bold" />

<Button
    android:id="@+id/buttonWaterBill"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="1dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_weight="0"
        android:minHeight="30dp"
        android:text="Tariffs"
        android:textColor="#3F51B5"
        android:textStyle="bold" />

    <Button
        android:id="@+id/buttonCustomer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="1dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginBottom="10dp"
        android:minHeight="30dp"
        android:text="Customer Data"
        android:textColor="#3F51B5"
        android:textStyle="bold" />

  </LinearLayout>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Water_meter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/heading1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F51B5"
        android:padding="8dp"
        android:text="WATER METER DATA"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:text="Billing Cycle: September 2020"
        android:textAlignment="center"
        android:textColor="#000000"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:background="#883F51B5"
        android:paddingRight="15dp">

        <TextView
            android:id="@+id/textView77"
            android:layout_width="90dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15dp"
            android:text="URL: "
            android:textColor="#747474"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/meterURL"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="textPersonName"
```

```
        android:padding="6sp"
        android:text="www.pastebin.com/raw/"
        android:textColor="#747474"
        android:textSize="14sp" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:background="#883F51B5"
    android:paddingRight="15dp">

    <TextView
        android:id="@+id/textView88"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:text="API Key: "
        android:textColor="#747474"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/meterApiKey"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:padding="6sp"
        android:text="mbPfQ4UC"
        android:textAppearance="@style/TextAppearance.AppCompat.Small"
        android:textColor="#747474" />

    <Button
        android:id="@+id/save1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:minWidth="60dp"
        android:minHeight="30dp"
        android:text="Save"
        android:textColor="#3F51B5"
        android:textSize="10sp" />

</TableRow>

<TableRow
    android:id="@+id/row1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textViewDeviceID1"
        android:layout_width="200dp"
```

```
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Meter ID:"
        android:textColor="#000000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewMeterID"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="Stel123456" />

</TableRow>

<TableRow
    android:id="@+id/row3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textViewUpdated1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Last Updated"
        android:textColor="#000000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewUpdated"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="10/09/2020" />

</TableRow>

<TableRow
    android:id="@+id/row4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textViewCurrentReading1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Meter Reading:"
        android:textColor="#000000"
        android:textStyle="bold" />
```

```
    <TextView
        android:id="@+id/textViewCurrentReading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="2704180" />

</TableRow>


<TableRow
    android:id="@+id/row5"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textViewLitersUsed1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Water Used This Month :"
        android:textColor="#000000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewWaterUsed"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="18.6 kL" />

</TableRow>

<TableRow
    android:id="@+id/row6"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textViewBillToDate1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Current Bill:"
        android:textColor="#000000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewBillToDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="R170.34" />
```

```
</TableRow>

<TableRow
   android:id="@+id/row7"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

   <TextView
      android:id="@+id/textViewProjectedUsage1"
      android:layout_width="200dp"
      android:layout_height="wrap_content"
      android:layout_marginLeft="10dp"
      android:padding="10dp"
      android:text="Projected Consumption:"
      android:textColor="#000000"
      android:textStyle="bold" />

   <TextView
      android:id="@+id/textViewProjectedUsage"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:padding="10dp"
      android:text="25.7 kL" />

</TableRow>

<TableRow
   android:id="@+id/row8"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

   <TextView
      android:id="@+id/textViewProjectedBill1"
      android:layout_width="200dp"
      android:layout_height="wrap_content"
      android:layout_marginLeft="10dp"
      android:padding="10dp"
      android:text="Projected Bill:"
      android:textColor="#000000"
      android:textStyle="bold" />

   <TextView
      android:id="@+id/textViewProjectedBill"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:padding="10dp"
      android:text="R371.95" />

</TableRow>

<TableRow
android:id="@+id/row9"
android:layout_width="match_parent"
android:layout_height="wrap_content">
```

```
    <TextView
        android:id="@+id/textViewBlock1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Current Block:"
        android:textColor="#000000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewCurrentBlock"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="18 - 25 kL" />

</TableRow>

<TableRow
    android:id="@+id/row10"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textViewCurrentRate1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:padding="10dp"
        android:text="Current Rate:"
        android:textColor="#000000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewCurrentRate2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="R24.95 per kL" />

</TableRow>


<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1" />

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

<Button
```

```
        android:id="@+id/back2MainMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:background="#3F51B5"
        android:text="Back"
        android:textColor="#FFFFFF" />

</LinearLayout>
```

Flow_device.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <EditText
            android:id="@+id/deviceName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:background="#3F51B5"
            android:gravity="center|center_horizontal"
            android:inputType="textPersonName"
            android:padding="8dp"
            android:text="FLOW SENSOR DATA"
            android:textColor="#FFFFFF"
            android:textSize="18sp"
            android:textStyle="bold" />

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="15dp"
            android:layout_marginRight="10dp"
            android:background="#883F51B5"
            android:gravity="left|center_vertical">

            <TextView
                android:id="@+id/textView6"
                android:layout_width="90dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15dp"
                android:text="Description: "
                android:textColor="#747474"
                android:textStyle="bold" />

            <EditText
                android:id="@+id/deviceDescription"
                style="@style/Widget.AppCompat.EditText"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
```

```
        android:allowUndo="false"
        android:padding="6sp"
        android:text="Device Description: "
        android:textAllCaps="false"
        android:textColor="#747474"
        android:textIsSelectable="false"
        android:textSize="14sp" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:background="#883F51B5"
    android:paddingRight="15dp">

    <TextView
        android:id="@+id/textView7"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:text="URL: "
        android:textColor="#747474"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/deviceURL"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName"
        android:padding="6sp"
        android:text="www.pastebin.com/raw/"
        android:textColor="#747474"
        android:textSize="14sp" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:background="#883F51B5"
    android:paddingRight="15dp">

    <TextView
        android:id="@+id/textView8"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:text="API Key: "
```

```
        android:textColor="#747474"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/deviceApiKey"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:padding="6sp"
        android:text="KzsDeuRy"
        android:textAppearance="@style/TextAppearance.AppCompat.Small"
        android:textColor="#747474" />

    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:minWidth="60dp"
        android:minHeight="30dp"
        android:text="Save"
        android:textColor="#3F51B5"
        android:textSize="10sp" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:layout_weight="0">

    <TextView
        android:id="@+id/dateText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center_horizontal"
        android:text="Date"
        android:textColor="#020000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/TimeText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center_horizontal"
        android:text="Time"
        android:textColor="#020000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/durationText"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center_horizontal"
        android:text="Duration"
        android:textColor="#020000"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/temperatureText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center_horizontal"
        android:text="Temperature"
        android:textColor="#020000"
        android:textStyle="bold" />
</TableRow>

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/deviceRecycvlerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:paddingLeft="15dp"
    android:paddingRight="15dp"/>

<TextView
    android:id="@+id/textViewLastUpdated2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    android:text="Last Updated: 09/09/2020"
    android:textAlignment="textEnd"
    android:textStyle="italic" />

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0">

        <Button
            android:id="@+id/dataUpdateButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="Update Data"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            android:textColor="#3F51B5"
```

```
            android:textStyle="bold" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0">

            <Button
                android:id="@+id/backDeviceView2DeviceManager"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_margin="5dp"
                android:layout_weight="1"
                android:background="#3F51B5"
                android:text="Back"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:textColor="#FFFFFF"
                android:textStyle="bold" />

        </TableRow>

    </TableLayout>

</LinearLayout>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Flow_device_database_table_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/dataEntryParentLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/dateTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="1992-08-24"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/timeTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="20:34:01"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/durationTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="100"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/temperatureTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="100"
            android:textColor="#000000" />
    </TableRow>
</LinearLayout>
```

Customer.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/headingTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F51B5"
        android:padding="8dp"
        android:text="CUSTOMER DETAILS"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        android:textStyle="bold" />


    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/textView1"
                android:layout_width="100dp"
                android:layout_height="20dp"
                android:layout_marginLeft="10dp"
                android:text="First Name:"
                android:textColor="#000000"
                android:textSize="16sp"
                android:textStyle="bold" />

            <EditText
                android:id="@+id/editTextFirstName"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginRight="10dp"
                android:layout_weight="1"
                android:ems="10"
                android:inputType="textPersonName"
                android:text="Vaughan"
                android:textColor="#525252"
                android:textSize="16sp" />
        </TableRow>

        <TableRow
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:text="Surname:"
        android:textColor="#000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/editTextSurname"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Warren"
        android:textColor="#525252"
        android:textSize="16sp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView10"
        android:layout_width="100dp"
        android:layout_height="20dp"
        android:layout_marginLeft="10dp"
        android:text="Cellphone:"
        android:textColor="#000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/editTextCellphone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="086 12345678"
        android:textColor="#525252"
        android:textSize="16sp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView11"
        android:layout_width="100dp"
        android:layout_height="20dp"
        android:layout_marginLeft="10dp"
        android:text="Email:"
        android:textColor="#000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="vaughan@emailaddress.com"
        android:textColor="#525252"
        android:textSize="16sp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView8"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:text="Meter ID:"
        android:textColor="#000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/editTextMeterID"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Stell123456"
        android:textColor="#525252"
        android:textSize="16sp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
```

```
        <TextView
            android:id="@+id/textView4"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Address:"
            android:textColor="#000000"
            android:textSize="16sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/editTextAddressLine1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="17 Rhodes-Noord"
            android:textColor="#525252"
            android:textSize="16sp" />
    </TableRow>

    <TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView5"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Address"
            android:textColor="#FFFFFF"
            android:textSize="16sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/editTextAddressLine2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="Die Boord"
            android:textColor="#525252"
            android:textSize="16sp" />

</TableRow>
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
```

```
        <TextView
            android:id="@+id/textView6"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Address"
            android:textColor="#FFFFFF"
            android:textSize="16sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/editTextAddressLine3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="Stellenbosch"
            android:textColor="#525252"
            android:textSize="16sp" />

    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView7"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Address"
            android:textColor="#FFFFFF"
            android:textSize="16sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/editTextAddressLine4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="7600"
            android:textColor="#525252"
            android:textSize="16sp" />

    </TableRow>

</TableLayout>

<Button
```

```
        android:id="@+id/save"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="0"
        android:minHeight="30dp"
        android:text="Save"
        android:textColor="#3F51B5"
        android:textStyle="bold" />

    <Button
        android:id="@+id/backButton2mainMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="0"
        android:background="#3F51B5"
        android:minHeight="30dp"
        android:text="Back"
        android:textColor="#FFFFFF"
        android:textStyle="bold" />

</LinearLayout>
```

Tariffs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/headingText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|center_vertical"
        android:background="#3F51B5"
        android:gravity="center_horizontal|center_vertical"
        android:minHeight="48dip"
        android:text="TARIFF STRUCTURE"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/municipalityName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|center_vertical"
        android:gravity="center_horizontal|center_vertical"
        android:minHeight="48dip"
        android:text="Stellenbosch Municipality"
        android:textAlignment="center"
        android:textColor="#000000"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:minHeight="40dp">

        <TextView
            android:id="@+id/blockText1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:gravity="center"
            android:text="Block"
            android:textColor="#000000"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tariffText1"
            android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="Rate"
        android:textColor="#000000"
        android:textStyle="bold" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="0 - 6 kL"
        android:textColor="#000000" />

    <TextView
        android:id="@+id/tariffText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 6.93"
        android:textColor="#000000" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="6 - 12 kL"
        android:textColor="#000000" />

    <TextView
```

```
        android:id="@+id/tariffText3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 9.90"
        android:textColor="#000000" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="12 - 18 kL"
        android:textColor="#000000" />

    <TextView
        android:id="@+id/tariffText4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 16.74"
        android:textColor="#000000" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="18 - 25 kL"
        android:textColor="#000000" />
```

```
    <TextView
        android:id="@+id/tariffText5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 28.69"
        android:textColor="#000000" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="25 - 40 kL"
        android:textColor="#000000" />

    <TextView
        android:id="@+id/tariffText6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 39.00"
        android:textColor="#000000" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="40 - 70 kL"
        android:textColor="#000000" />
```

```
    <TextView
        android:id="@+id/tariffText7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 60.95"
        android:textColor="#000000" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:minHeight="40dp">

    <TextView
        android:id="@+id/blockText8"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="70 kL +"
        android:textColor="#000000" />

    <TextView
        android:id="@+id/tariffText8"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:gravity="center"
        android:text="R 91.43"
        android:textColor="#000000" />

</TableRow>


<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:layout_margin="5dp"
    android:layout_weight="1">

    <TextView
        android:id="@+id/textViewLastUpdated"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:text="Last Updated: 09/09/2020"
```

```
            android:textAlignment="textEnd"
            android:textStyle="italic" />
    </TableRow>

    <Button
        android:id="@+id/back2WaterBill"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:background="#3F51B5"
        android:text="Back"
        android:textColor="#FFFFFF" />

</LinearLayout>
```

Graphs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/headingTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F51B5"
        android:padding="8dp"
        android:text="GRAPHS"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        android:textStyle="bold" />

    <com.jjoe64.graphview.GraphView
        android:id="@+id/graph1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"
        android:layout_weight="1"
        android:paddingRight="5dp" />

    <Button
        android:id="@+id/buttonHistoricalDaily"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Data by Month"
        android:textColor="#3F51B5" />

    <Button
        android:id="@+id/buttonHistoricalMonthly"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Data by Year"
        android:textColor="#3F51B5" />

    <Button
        android:id="@+id/backButton2mainMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="0"
        android:background="#3F51B5"
        android:text="Back"
        android:textColor="#FFFFFF"
        android:textStyle="bold" />

</LinearLayout>
```

Appendix E – Data Files

Flow_device.txt


```
{
   "Sample": [
            {
               "Date": "01-09-2020",
               "Time": "7:02:17",
               "Duration": "9.3",
               "Temperature": "19.3"6
            },
            {
               "Date": "01-09-2020",
               "Time": "9:32:02",
               "Duration": "4.7",
               "Temperature": "19.3"
            },
            {
               "Date": "01-09-2020",
               "Time": "9:33:17",
               "Duration": "8.6",
               "Temperature": "19.7"
            },

---------- TRUNCATED ----------

            {
               "Date": "01-09-2020",
               "Time": "16:19:47",
               "Duration": "12.2",
               "Temperature": "20.7"
            },
            {
               "Date": "01-09-2020",
               "Time": "19:01:02",
               "Duration": "8.1",
               "Temperature": "20.1"
            },
            {
               "Date": "01-09-2020",
               "Time": "22:46:17",
               "Duration": "4.1",
               "Temperature": "19.5"
            }
   ]
}
```

Meter_data_daily.txt


```
{
   "Daily": [
      {
         "Day": "1",
         "dL": "791",
         "Sum": "791"
      },
      {
         "Day": "2",
         "dL": "427",
         "Sum": "1218"
      },
      {
         "Day": "3",
         "dL": "894",
         "Sum": "2112"
      },
      {
         "Day": "4",
         "dL": "493",
         "Sum": "2605"
      },
```


---------- TRUNCATED ----------


```
      {
         "Day": "362",
         "dL": "471",
         "Sum": "310126"
      },
      {
         "Day": "363",
         "dL": "616",
         "Sum": "310742"
      },
      {
         "Day": "364",
         "dL": "1004",
         "Sum": "311746"
      },
      {
         "Day": "365",
         "dL": "703",
         "Sum": "312449"
      }
   ]
}
```

Meter_data_hourly.txt


```
{
   "Hourly": [
      {
         "Date": "9/1/16",
         "Hour": "0",
         "dL": "27"
      },
      {
         "Date": "9/1/16",
         "Hour": "1",
         "dL": "9"
      },
      {
         "Date": "9/1/16",
         "Hour": "2",
         "dL": "14"
      },
      {
         "Date": "9/1/16",
         "Hour": "3",
         "dL": "13"
      },
```

---------- TRUNCATED ----------

```
      {
         "Date": "8/31/17",
         "Hour": "20",
         "dL": "56"
      },
      {
         "Date": "8/31/17",
         "Hour": "21",
         "dL": "17"
      },
      {
         "Date": "8/31/17",
         "Hour": "22",
         "dL": "46"
      },
      {
         "Date": "8/31/17",
         "Hour": "23",
         "dL": "41"
      }
   ]
}
```

Meter_data_monthly.txt


```
{
  "Monthly": [
    {
      "Month": "1", "kL": "26.045"
    },
    {
      "Month": "2", "kL": "24.146"
    },
    {
      "Month": "3", "kL": "24.937"
    },
    {
      "Month": "4", "kL": "26.991"
    },
    {
      "Month": "5", "kL": "27.386"
    },
    {
      "Month": "6", "kL": "25.621"
    },
    {
      "Month": "7", "kL": "28.456"
    },
    {
      "Month": "8", "kL": "27.949"
    },
    {
      "Month": "9", "kL": "25.058"
    },
    {
      "Month": "10", "kL": "24.546"
    },
    {
      "Month": "11", "kL": "25.552"
    },
    {
      "Month": "12", "kL": "25.762"
    }
  ]
}
```

Meter_summary.txt

“Summary”: [

{

"Device ID": "Stel123456",

"Last Updated": "10/09/2020",

"Meter Reading": "2704211",

"Water Used This Month": "18.6 kL",

"Current Bill": "R188.13",

"Projected Consumption": "25.7 kL",

"Projected Bill": "R371.95",

"Current Block": "18 - 25 kL",

"Current Rate": "R24.95 per kL"

}

]

September_2019.txt


```
{
   "Daily": [
      {
         "Day": "1",
         "dL": "791",
         "Sum": "791"
      },
      {
         "Day": "2",
         "dL": "427",
         "Sum": "1218"
      },
      {
         "Day": "3",
         "dL": "894",
         "Sum": "2112"
      },
      {
         "Day": "4",
         "dL": "493",
         "Sum": "2605"
      },
```

----------TRUNCATED----------

```
      {
         "Day": "27",
         "dL": "1141",
         "Sum": "22027"
      },
      {
         "Day": "28",
         "dL": "1104",
         "Sum": "23131"
      },
      {
         "Day": "29",
         "dL": "1024",
         "Sum": "24155"
      },
      {
         "Day": "30",
         "dL": "903",
         "Sum": "25058"
      }
   ]
}
```

Stellenbosch_tariffs.txt

“Stellenbosch”: [
    {
        "0 - 6 kL": "R 6.93",
        "6 - 12 kL": "R 9.90",
        "12 - 18 kL": "R 16.74",
        "18 - 25 kL": "R 28.69",
        "25 - 40 kL": "R 39.00",
        "40 - 70 kL": "R 60.95",
        "70 kL +": "R 91.43"
    }
]

Today.txt


```
{
   "Today": [
            {
               "Date": "01-01-2017",
               "Hour": "00",
               "dL": "27"
            },
            {
               "Date": "01-01-2017",
               "Hour": "01",
               "dL": "9"
            },
            {
               "Date": "01-01-2017",
               "Hour": "02",
               "dL": "14"
            },
```

---------- TRUNCATED ----------

```
           {
               "Date": "01-01-2017",
               "Hour": "20",
               "dL": "25"
            },
            {
               "Date": "01-01-2017",
               "Hour": "21",
               "dL": "36"
            },
            {
               "Date": "01-01-2017",
               "Hour": "22",
               "dL": "15"
            },
            {
               "Date": "01-01-2017",
               "Hour": "23",
               "dL": "16"
            }
   ]
}
```