Mobile Health Data: Investigating the data used by an mHealth app using different mobile app architectures

A dissertation

submitted to the Department of Computer Science,

Faculty of Science

at the University of Cape Town

In partial fulfillment of the requirements

for the degree of

Master of Science in Information Technology (MIT)

By

Faizel Faker

Supervised by

Prof. Hussein Suleman and Dr. Brian DeRenzi

6 December 2018

Declaration

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used the APA Convention for citation and referencing. Each contribution to, and quotation in, this essay from the works of other people has been attributed and has been cited and referenced.
- This thesis is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Full name and number of student: Faizel Faker – FKRFAI002

Signed by candidate

Signature _____ **Date:** 6 December 2018

Acknowledgments

I would like to thank my wife, Nehaal Faker, for her support and assisting me during this entire process.

I would also like to express my gratitude and to thank my supervisors, Professor Hussein Suleman and Dr Brian DeRenzi, for their guidance and support.

Lastly, I would like to thank my work colleagues and my place of employment, Traffic Management Technologies, who has sponsored, supported and enabled me to further my education.

Abstract

Mobile Health (mHealth) has come a long way in the last forty years and is still rapidly evolving and presenting many opportunities. The advancements in mobile technology and wireless mobile communication technology contributed to the rapid evolution and development of mHealth. Consequently, this evolution has led to mHealth solutions that are now capable of generating large amounts of data that is synchronised and stored on remote cloud and central servers, ensuring that the data is distributable to healthcare providers and available for analysis and decision making.

However, the amount of data used by mHealth apps can contribute significantly to the overall cost of implementing a new or upscaling an existing mHealth solution. The purpose of this research was to determine if the amount of data used by mHealth apps would differ significantly if they were to be implemented using different mobile app architectures.

Three mHealth apps using different mobile app architectures were developed and evaluated. The first app was a native app, the second was a standard mobile Web app and the third was a mobile Web app that used Asynchronous JavaScript and XML (AJAX). Experiments using the same data inputs were conducted on the three mHealth apps. The primary objective of the experiments was to determine if there was a significant difference in the amount of data used by different versions of an mHealth app when implemented using different mobile app architectures.

The experiment results demonstrated that native apps that are installed and executed on local mobile devices used the least amount of data and were more data efficient than mobile Web apps that executed on mobile Web browsers. It also demonstrated that mobile apps implemented using different mobile app architectures will demonstrate a significant difference in the amount of data used during normal mobile app usage.

Contents

List of Figures

List of Tables

List of Acronyms

ANC - Antenatal care

API – Application Programming Interface

APP - Application

ASHA - Accredited Social Health Activist

ASP – Active Server Pages

AVD – Android Virtual Device

CHW - Community Healthcare Workers

CSS – Cascading Style Sheets

DOM – Document Object Model

eHealth - Electronic Health

EHR - Electronic Health Record

EMR - Electronic Medical Record

GBD - Global Burden of Disease

HIV/AIDS - Human Immunodeficiency Virus / Acquired Immune Deficiency Syndrome

HTML – Hyper Text Mark-up Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

ICT - Information and Communication Technology

IDA - Iron Deficiency Anaemia

IEEE - Institute of Electrical and Electronic Engineers

IIS – Internet Information Services

IPv6 – Internet Protocol Version 6

JSON – JavaScript Object Notation

LTE – Long Term Evolution

M - Mean

MS - Milliseconds

MMR - Maternal Mortality Rates

MCH - Maternal and child healthcare

mHealth - Mobile Health

NCD - Non-Communicable diseases

p – P-Value

PDA - Personal Digital Assistant

POCT - Point-of-Care

SD – Standard Deviation

t – T-Value

SOCKS – Secure Sockets

UDP – User Datagram Protocol

WHO - World Health Organization

Chapter 1: Introduction

Mobile Health (mHealth) is the practice of using mobile devices and wireless mobile communication technology to assist and support the delivery of medical and healthcare services (Tachakra et al., 2003). It is a sub-component of Electronic Health (eHealth) (Perera, 2012) and is the most recent form of digital healthcare technology to have evolved (Tachakra et al., 2003).

Mobile health is providing many new opportunities and solutions in the battle against the world's most prevalent diseases (Kumar et al., 2013). It is bridging the gap between communities and public health systems (Braun et al., 2013), alleviating the healthcare service delivery burden experienced by numerous government health institutions, assisting in the collection of clinical health data (Free et al., 2010), assisting in the delivery of healthcare services, assisting in the monitoring and management of diseases, providing platforms to share healthcare information amongst clinicians and researchers and providing patients and individuals with advanced remote, real-time healthcare monitoring services and applications (Braun et al., 2013).

MHealth is providing many opportunities to solve current healthcare problems, however, implementing mHealth solutions successfully comes with many challenges (Kumar et al., 2013). This is especially evident when implementing mHealth apps in developing countries, where factors like user feedback, data collection and reporting, ease of use, user acceptance, availability of reliable infrastructure, government buy-in and the overall cost of the app are critical factors that can contribute to the success or failure of an mHealth app (Aranda-Jan et al., 2014).

1.1. Problem Statement and Motivation

According to Aranda-Jan et al. (2014), the success or failure of any mHealth app is largely dependent on it fulfilling its functional and non-functional requirements. Two pertinent

factors contributing to the success or failure of an mHealth app are (Aranda-Jan et al., 2014):

- **Overall Costs** – the overall cost of implementing and upscaling an mHealth app is a critical factor contributing to the success or failure of a mHealth solution. These costs include the cost of bandwidth and mobile data where the mHealth app will be used (Helios, 2017).
- **User Experience –** user experience is directly affected by the Internet communication speed and performance of an mHealth app (Aranda-Jan et al., 2014). Bandwidth, infrastructure and Internet speeds where the mHealth app will be used and the amount of data transferred between clients and servers all play a pivotal role in the user experience and acceptance of an mHealth app (Helios, 2017).

1.2. Thesis Statement

Modern mobile devices have powerful processing and large storage capacities enabling them to execute complex functions and advanced processes (Perera, 2012). The advancements in mobile devices, mobile app architectures and wireless mobile communication technology now make it possible to develop native mobile and Web based apps that do not require large amounts of data to be transferred to servers over mobile communication networks (Kumar et al., 2013). Advancements in mobile app architectures and development frameworks such as CommCare, Android Studio, Apache Cordova, NativeScript and Microsoft's Xamarin are also enabling mobile app developers to develop native, hybrid and Web based apps that look, feel and perform very similarly (Raboy, 2018).

However, mobile application architectures have advantages and disadvantages that include factors such as development costs, application development and delivery times and access to multiple skillsets required to develop a mobile application (Dalmasso etal., 2013). These advantages and disadvantages can contribute to selecting a mobile application architecture that is less data efficient than other mobile architectures (Dalmasso etal., 2013). It is recommended that businesses and organisations consider

several factors before formulating a mobile strategy, establishing a mobile presence and investing in a mobile app and mobile platform (Summerfield, 2018). The purpose of this research was to investigate the difference in the amount of data used when implementing an mHealth app as a native Android mobile app versus implementing it as a mobile Web based app on a modern mobile device and to answer the following question:

"Is there a significant difference in the amount of data used by an mHealth application that is used to manage maternal and neonatal health when implemented using a native message passing Android architecture versus using a mobile Web based architecture?"

1.3. Hypothesis

The amount of data used by an mHealth app that is used to manage maternal and neonatal health will be significantly different when implemented as a native mobile Android app versus implementing it as a Web based mobile app or an AJAX Web based mobile app.

1.4. Methodology

Three versions of an mHealth app used to manage maternal and neonatal health were developed and experimented with. The objective of the experiments was to analyse and determine if the amount of data used by the different versions of the mHealth app were significantly different when used during standard operational procedures and decision-making processes. The target audience of the apps was community health workers (CHW) who worked in local communities.

The first version of the app was a native Android app and comprised of two mobile components:

- The first component enabled users to capture and manage pregnant women's maternal and neonatal health care data and was developed with the CommCare mobile application development platform.
- The second component provided CHWs with a mobile user interface to perform data analysis that assisted with case management statistics, performance monitoring, trend spotting and decision making.

The second version of the app was an ASP.Net Web based application. The app executed requests using HTTP Web requests and the entire Web page was reloaded during each Web request.

The third version of the app was an ASP.Net Web based application using Asynchronous JavaScript and XML (AJAX). The app executed HTTP Web requests asynchronously and only the content of the Web page was dynamically updated without reloading the entire Web page during every Web request.

The Android app and the Web based apps provided the same functionalities. The data captured while performing experiments with the three different apps provided the information required to answer the research question and to test the thesis hypothesis.

1.5. Thesis Structure

1.5.1. Chapter 1: Introduction

Chapter 1 provides an overview of mHealth and information on how mHealth is contributing to the medical and healthcare industries. It also introduces some challenges that could hinder mHealth implementations and introduces a research problem, hypothesis and a methodology.

1.5.2. Chapter 2: Background and Literature Review

Chapter 2 provides an overview of mHealth, its evolution, current state of mHealth literature and identifies why, where, who and how mHealth is currently being used and its impact on healthcare industries. It introduces some of the technologies required and used to implement an mHealth app and provides an analysis on the current state, usage and performance of these technologies. It introduces the design features and gamification techniques currently employed to make mHealth apps more user friendly, to encourage positive user behaviour and the correct usage of an mHealth app. It introduces maternal and neonatal health as an industry that is currently benefitting from mHealth and the geographical regions where it is yielding positive results.

Lastly, it provides examples of how mHealth is contributing to the management of pregnant women's health and provides arguments as to how mHealth can improve antenatal health and infant mortality rates.

1.5.3. Chapter 3: Native, Web and Hybrid App Architectures

Chapter 3 provides an overview of mobile app architectures currently available for mobile app development. It introduces three well established mobile app architectures, their differences, advantages and disadvantages and the factors to consider when choosing a mobile app architecture.

1.5.4. Chapter 4: Research Methodology and Design

Chapter 4 provides an overview of the research design and research methodology and provides an explanation of the experiments implemented to carry out the research.

1.5.5. Chapter 5: Findings and Experiment Results

Chapter 5 presents the data and provides a statistical analysis of the data gathered from the experimental observations.

1.5.6. Chapter 6: Conclusions

Chapter 6 presents the research outcome, the answer to the research question and the factors that influenced the outcome of the experiments.

Chapter 2: Background and Literature Review

2.1. Introduction

Mobile computing, mobile communication technology and mobile smartphones have seen exponential growth in the last decade. According to Pieljko (2016), there were more than 4.61 billion mobile smartphone users worldwide in 2016 and it is predicted that this number will grow beyond 4.77 billion in 2018. Mobile smartphones have multiple functions and applications that include communication, GPS navigation, Internet browsing, social media and networking, digital and video photography, health and fitness tracking and have become ubiquitous in modern day life (Gupta, 2011). The advancements in mobile technology have enabled mobile application developers to create and provide a plethora of Web based and native mobile applications to many different industries, including the healthcare and Mobile Health (mHealth) industry.

Mobile Health is the practice of using mobile technology, mobile applications and mobile devices to enable, assist and support the delivery of medical and healthcare services (Free, et al., 2010). It is a component of Electronic Health (eHealth) (Perera, 2012) and targets a wide range of audiences that include community healthcare workers, medical professionals, doctors, clinicians, government healthcare services and the public in general. It provides these audiences mobile access to advanced medical and healthcare services and easy access to medical information (Perera, 2012).

The purpose of this background and literature review is to:

- Provide an overview on the evolution of mHealth.
- To assess and provide an overview of past mHealth studies and mHealth literature that is currently available.
- Describe the different uses of mHealth including how, where, what and who is using it.
- Analyse the impact that mHealth has had on the healthcare industry and the benefits of mHealth.
- Identify the benefits of using mobile gamification techniques and data visualisations to encourage and drive user behaviour in mHealth.

- Evaluate the differences and features in native mobile app architectures and mobile Web based app architectures.
- Evaluate the use of mHealth for managing maternal and neonatal health.

2.2. The Evolution of Mobile Health

MHealth is widely considered as the evolution from desktop telemedicine to using wireless mobile devices and mobile technology to provide advanced medical healthcare services (Istepanian, Jovanov, & Zhang, 2004).

The rapid improvements and developments of mHealth are assisting in the treatment of chronic diseases, generating public health data that is assisting in decision making, alleviating the burden on public health services and government health institutions and educating the general population on healthcare. Individuals with mobile smartphones are able to use mHealth technology to track fitness and activity levels, track weight loss, monitor diet and caloric consumption, monitor sleep patterns and manage chronic diseases (Fiordelli, Diviani, & Schulz, 2016).

Mobile Health is playing an important role in local communities where Community Health Workers (CHW) residing in local communities are using mHealth technology to provide advanced medical care, health assessments, advanced disease detection and prevention, pregnancy and prenatal monitoring (DeRenzi et al., 2011). CHWs are removing some of the healthcare service delivery burden from local government institutions that are already operating with constrained resources (Braun et al, 2013).

Mobile Health has made it possible for CHWs to operate in isolated locations and to provide quality healthcare services to people from lower income settings, people who are in isolated locations and people in developing countries, such as Ethiopia, Malawi and India (Kahn, Yang, & Kahn, 2010). The healthcare industry regards CHWs as the frontline for providing healthcare in local communities because they can conduct person-to-person healthcare assessments in local communities and CHW programs have proven to be very successful (Braun et al, 2013). Examples of well-run community healthcare programs using

mHealth technology resulted in a 40% reduction in malaria mortality rates in Ethiopia and reduced neonatal mortality rates by 34% in Bangladesh (DeRenzi, et al., 2016).

However, implementing an mHealth solution successfully comes with many challenges and constraints (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014). These include the lack of adequate mobile infrastructure, the cost of mobile infrastructure (Kahn, Yang, & Kahn, 2010), the lack of government involvement (Ali, Chew, & Yap, 2016) and the assimilation of mHealth technology by its target audience (Yu et al., 2006). These constraints have a direct impact on two pertinent factors that contribute to the overall success or failure of an mHealth solution: user acceptance or the adoption and acceptance of the mHealth app by its target audience; and the overall cost of implementing and upscaling of an mHealth app (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014).

2.2.1. Using Information and Communication Technology to assist in the delivery of Healthcare

Using Information and Communication Technology (ICT) in the delivery of healthcare services is not a new concept. In the early 1900s when radio revolutionised communication, innovators had already started envisioning how doctors could attend to patients over the radio (Fenston, 2011). In April 1924, Radio News Magazine ran a cover story titled “The Radio Doctor” with the cover of the magazine (see Figure 1) depicting a doctor examining a boy via a live video interface (Fenston, 2011).

Figure 1: The cover of Radio News magazine, April, 1924 (Field, 1996)

The advancements made in ICT have since made the prediction depicted in Figure 1 possible in the form of telemedicine, electronic health and mobile health.

Telemedicine, electronic health and mobile health are three forms of digital health technology that uses ICT to assist in the delivery of medical and healthcare services (Lupton, 2014).

2.2.2. Telemedicine

Literature suggests that the earliest reference to telemedicine first appeared in medical literature in 1950 (Field, 1996). Telemedicine is the use of information and communication technology to remotely provide healthcare services to people who are far from healthcare providers (Roine, Ohinmaa, & Hailey, 2001). This includes the sharing of electronic medical information between healthcare service providers by video, email, smartphones and any other forms of telecommunication technology. Telemedicine became popular in rural and isolated areas where access to healthcare services was limited. Telemedicine provided people in these locations with a solution to access medical specialists from afar (Roine, Ohinmaa, & Hailey, 2001). Early implementations of telemedicine were across

mountainous areas, islands and arctic regions (Field, 1996). According to Field (1996), telemedicine has 3 classification categories:

1. Remote Patient Monitoring - Remote monitoring of patients with chronic diseases such as sugar diabetes, epilepsy and heart disease.
2. Store and Forward - Enables healthcare providers to share patient information with other medical professionals in different regions.
3. Interactive Telemedicine - Enables physicians to communicate with patients in real time.

2.2.3. Electronic Health (eHealth)

The term eHealth is widely used amongst academics, business, medical professionals and the public in general. This has led to a lack of consensus and a precise definition of eHealth. According to Oh et al. (2005), in 2005 there were 10 different definitions of eHealth found in academic literature databases and 41 definitions for eHealth from a Google search. A concrete definition is out of scope of this research; however, a broad description of eHealth is the use of electronic communication, the Internet and related technologies to deliver healthcare services. EHealth is comprised of a variety of sub-components of digital health, which include (Mea, 2001):

1. Electronic Heath Records (EHR)
2. Electronic Medical Records (EMR)
3. Telemedicine
4. I.T. Health Systems
5. Consumer Healthcare Data
6. Virtual Healthcare
7. Big Data Systems
8. Mobile Health (mHealth)

2.2.4. Mobile Health

Mobile Health is the most recent iteration of digital healthcare and is defined as using mobile computing, mobile devices, mobile technology and mobile apps to provide

healthcare and medical services (Fiordelli, Diviani, & Schulz, 2016). This includes the use wearable mobile sensory technology to educate users about healthcare, chronic disease management, treatment support and remote monitoring.

Using mobile devices, mobile apps and mHealth technology is showing great potential in providing advanced medical care, especially in the low and middle-income markets (Fiordelli, Diviani, & Schulz, 2016). It is adding value in communities and alleviating some of the medical service delivery burden from government health institutions (Betjeman, Soghoain, & Foran, 2013). MHealth is assisting in data collection, providing healthcare information to individuals, managing chronic diseases and patient observations. It is also assisting in the battle against eight of the most prevalent global health conditions as listed by the World Health Organization's (WHO) Global Burden of Disease (GBD) list (Perez, De La Torre-Diez, & Lopez-Coronado, 2013). These diseases are:

1. Iron Deficiency Anaemia(IDA)
2. Hearing loss
3. Migraine
4. Low vision
5. Asthma
6. Diabetes mellitus
7. Osteoarthritis (OA)
8. Unipolar depressive disorder

MHealth literature demonstrates that the majority of mHealth research is carried out in rural locations, such as Africa, and the most researched topics are sexual health, reproductive health, HIV/AIDS, maternal and child health (Braun et al, 2013). It also demonstrates that the primary use of mHealth technology is for collecting field data, to convey healthcare education and to conduct person-to-person consultations (Braun et al, 2013). The low-cost, ease of use of mobile technology and the widespread use of mobile technology are some of the main factors that are driving mHealth solution implementations and mHealth research (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014). The benefits and success of mHealth were reported at every level of healthcare,

including government agencies, healthcare providers, medical professionals, patients and mHealth technology users. The successes in mHealth projects and research were largely due to the high acceptance and familiarity of mobile devices (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014).

According to Braun et al. (2013), removing paper-based systems and providing field workers and community healthcare workers with mHealth technology for data collection results in fewer data errors and data losses and a higher level of data quality. It also enables real time analysis and decision making (Braun et al, 2013). MHealth technology assists in improving the healthcare compliance levels of field workers, raises their standards of delivering healthcare services and helps in educating local communities about healthcare. It encourages the use of best practices and improves leadership and management in community healthcare workers (Braun et al, 2013).

However, there were also reports on mHealth failures and the current constraints affecting the successful implementation of mHealth technology. Many studies reported that the cost effectiveness of mHealth solutions are unknown and that mHealth technology only assists in data collection but does not help resolve the actual healthcare problems (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014). Limited funding in developing countries also make it difficult to implement long term mHealth solutions and mHealth research programs for extended periods (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014).

2.3. Mobile Health Applications

According to Terry (2015), there were more than 165,000 mHealth applications (apps) available in the Apple iTunes Store and Android Play Store in 2015. Their functionality ranged from managing and tracking health and fitness to managing non-communicable diseases (NCD), such as cardiovascular disease, diabetes and hypertension. Successful treatment of these NCDs is highly variable as success often depends on how vigilant patients are at self-care and adherence to a healthcare program (Perez, De La Torre-Diez, & Lopez-Coronado, 2013). Using mHealth apps to assist with self-care programs demonstrated to deliver very positive results (Terry, 2015). Most of these applications fall into the consumer health informatics (CHI) branch of healthcare and are designed for

monitoring, assisting or to inform users about their health statuses (Perez, De La Torre-Diez, & Lopez-Coronado, 2013).

2.4. MHealth App Gamification

Mobile app developers use gamification design and gamification techniques to drive and encourage user behaviour (Miller, Cafazzo, & Seto, 2016). Gamification is the use of gaming mechanics and design techniques in a non-video game context (Miller, Cafazzo, & Seto, 2016). This includes using game-based elements, such as badges, leaderboards and gaming strategies, such as challenges and tasks. These gaming elements drive user behaviour and reward users for achieving specific goals and objectives (Brigham, 2014). These are effective strategies to keep people's attention, engage users and for users to develop their skills (Mohamad, Salam, & Bakar, 2017). Users often find gamified apps less complicated and more engaging (Deterding, 2012) and they have been shown to increase users' participation in managing their health and increasing their levels of activity (Lee, Lee, & Lee, 2017). Implementing gamification techniques in mHealth apps promotes engagement in an entertaining manner, increases productivity and encourages healthy competition (Rashid & Suganya, 2017).

Gamified mHealth apps encourage their users to be self-involved and to self-manage their healthcare (Handel, 2011). The use of gamification design principles and techniques implemented in mHealth apps rewards self-involvement, self-management in patient healthcare (Zicherman & Cunningham, 2011) and the consistent use of mHealth apps (Miller, Cafazzo, & Seto, 2016). This improves and facilitates data collection, data quality, decision making and ultimately contributes to the overall success or failure of mHealth apps (Miller, Cafazzo, & Seto, 2016).

However, implementing gaming elements alone and over-simplifying apps can fail to engage users and have a negative effect on maintaining users' interests and experiences (Deterding, 2012). A successful app development strategy does not only incorporate gaming elements but also adheres to an overall gaming design strategy that is engaging, supportive, satisfying and challenging (Deterding, 2012). Mobile app developers achieve this by implementing the mechanics of gamification in a non-video game context (Miller,

Cafazzo, & Seto, 2016) . Literature demonstrates that the gamification provides users with entertaining experiences and has a direct effect on users' emotions and appeals to users' self-determination (Lee, Lee, & Lee, 2017).

2.5. MHealth Data Visualisation

Data visualisation and reporting provide users with overviews and analytics of their health performances, health statuses and assist in the tracking of their activities, achievements and behaviours. It encourages user performance and behaviour by providing constant reference points and analytics to improve and achieve an individual's health and fitness goals (Lobelo et al., 2007). Detailed data visualisation and reports also assists in decision making. Figure 2 is an example of a mobile data visualisation report displaying the Number of Cases Captured by Community Health Workers on an Android mHealth app.

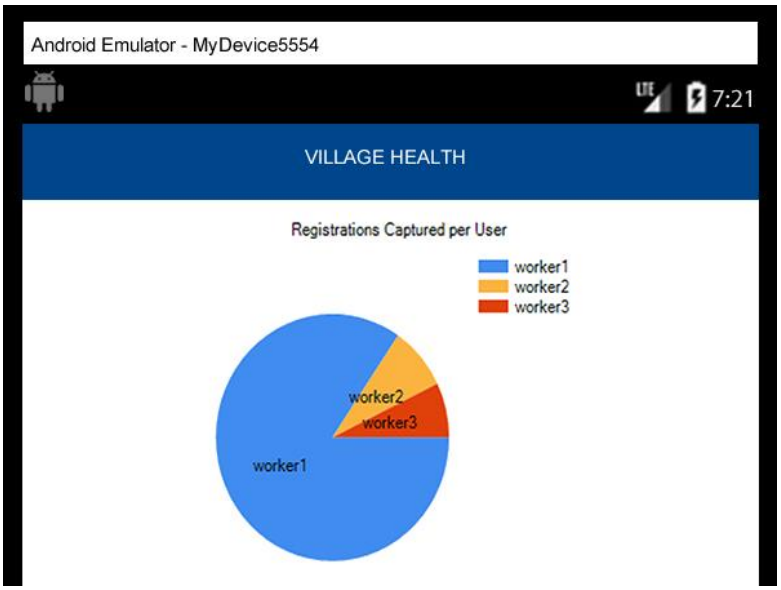


Figure 2: Village Health – Number of Cases Captured by Community Health Workers

2.6. MHealth in Developing Countries

The World Health Organisation states that eight of the most prevalent global health conditions are mainly found in underdeveloped and developing countries (Perez, De La Torre-Diez, & Lopez-Coronado, 2013) (Kahn, Yang, & Kahn, 2010). Mobile devices and wireless mobile communication technology are, however, becoming cheaper and more prevalent in remote areas of low-income and developing regions (DeRenzi, et al., 2011),

thus making these locations suitable for the implementation of mHealth solutions to treat prevalent and chronic diseases and to deliver healthcare.

MHealth technology is playing an important role in resolving the problem of reaching millions of people in different types of communities and locations. Providing community healthcare workers (CHW) with mobile devices and mHealth technology enables them to reach and supply much needed healthcare services to local communities (Kahn, Yang, & Kahn, 2010). It also assists government and public health institutions with documenting, tracking and control of infectious diseases while alleviating the burden on already constrained government hospitals, clinics, resources and healthcare facilities.

Tobacco, poor diets and low physical activity contributes to the causes of many non-communicable diseases, including sugar diabetes, heart and lung disease and cancer (Kahn, Yang, & Kahn, 2010). Using mHealth in conjunction with well-defined community health care programs can educate people and can assist in managing these diseases within the local communities and greater populations (Kahn, Yang, & Kahn, 2010).

2.7. Community Healthcare Workers and mHealth Technology

2.7.1. Community Health Workers

Community Healthcare Worker (CHW) is an umbrella term that encompasses a variety of trained healthcare aides such as Accredited Social Health Activists (ASHA) and midwives (Braun et al, 2013). CHWs provide healthcare services to their local communities (World Health Organisation, 2007). It is a concept that has existed for at least 70 years (World Health Organisation, 2007). CHWs can relate and understand local problems and local communities trust them when dealing with sensitive health issues (100 People Foundation, 2012). They have direct access to local communities, lower economic classes and individuals in isolated locations who often do not have the means to visit healthcare facilities (Braun et al, 2013) and are the bridge between healthcare systems and the local communities in which they serve (Braun et al, 2013).

CHWs provide healthcare services, such as healthcare education, contraceptive methods and home-based care to people living with AIDS (Braun et al, 2013), consequently

preventing disease and illnesses before they begin (Braun et al, 2013). They are playing a vital role by providing lower income families and individuals access to advanced healthcare and healthcare education in developing countries and are assisting the World Health Organisation (WHO) to reach its Millennium Development Goals (Velthoven et al, 2013) that include:

- Reducing Child Mortality
- Improve Maternal Health
- Combat HIV/AIDS, malaria and other diseases

2.7.2. Empowering Community Health Workers and mHealth Technology

CHWs from across the globe are using mobile phones and mobile technology to deliver healthcare services (Braun et al, 2013). According to Braun (2013), there are four key strategies to improve healthcare services delivered by CHWs equipped with mHealth technology:

1. Process improvement and technology development - Technological developments in mHealth ensures that CHWs capture healthcare data efficiently, with fewer errors, which results in consistent, regular and high-quality data.
2. Standards and guidelines - mHealth assists in ensuring that CHWs comply with the healthcare standards and guidelines while operating in the field. This is reinforced through decision support, alerts and reminder tools. For example, sending SMS message alert reminders to CHWs can be used to ensure that CHWs adhere to specific drug management schedules.
3. Education and training - mHealth technology provides CHWs in remote locations with access to educational and training resources. It also enables CHWs to create professional networks among CHWs, supervisors and healthcare institutions providing CHWs with real-time advice, healthcare information and support.
4. Leadership and management - mHealth technology enables communication between CHWs and their supervisors. This facilitates better leadership and management, especially with CHWs in remote locations.

MHealth is proving to assist CHWs in the collection of field-based healthcare data, to deliver healthcare education and to conduct person-to-person communication in developing regions and countries, such as Sub-Saharan Africa and India (Braun et al, 2013).

For example, India has been selecting and training women as ASHAs. From 2005 to 2013, India had already trained more than 750 000 ASHAs (World Health Organization, 2013) and has also been investing in mobile technology and providing ASHAs with mobile phones equipped with mHealth technology (World Health Organization, 2013). MHealth technology eliminated the tedious task of ASHAs having to transfer and capture patient visits and case records in huge book registers (100 People Foundation, 2012). MHealth is assisting the Indian government's national health programme by increasing access of healthcare to central and remote communities (World Health Organization, 2013). Table 1 illustrates the improvements in maternal mortality rates, infant mortality rates, pregnant women receiving prenatal care and births attended by skilled health staff realised by India from 1990 to 2015 (World Bank, 2017).

Table 1: *Improvements in Maternal and Neonatal Health in India from 1990 to 2015*

	1990	**1993**	**2000**	**2008**	**2015**
Maternal Mortality Ratio (modelled estimate, per 100 000 live births)	556		374		174
Infant Mortality Rate (per 1 000 live births)	88,4		66,6		36,2
Pregnant women receiving prenatal care (%)		61,9	61,8	75,8	
Births attended by skilled health staff (% of total)	34,2		42,5		52,8

ASHAs are contributing directly to the improvement of maternal and neonatal health and to reducing child mortality rates (World Health Organization, 2013). Equipping ASHAs with mHealth technology and mobile devices increases an ASHA's confidence, improves client engagement, increases the retention of healthcare knowledge and ensures consistent household visits are maintained by ASHAs (World Health Organization, 2013). MHealth

technology provides CHWs with advanced healthcare support, healthcare information and education via its mHealth apps and platforms (Dimagi, 2017). It enables CHWs to capture and display images, videos and healthcare information visually to patients with mobile devices or smartphones (Dimagi, 2017). It removes the tedious task of having to collect healthcare data on paper and reduces the risk of transcription errors (DeRenzi, et al., 2011), ultimately providing a more efficient and effective method of generating health records. MHealth apps using the global position system (GPS) functionality available on modern mobile devices are able to track CHWs' movements, to identify and map the locations of homes visited by CHWs and used to build datasets that can be used for decision making, tracking and management of disease outbreaks such as respiratory illnesses and maternal and neonatal health (Blaschke et al., 2009).

The improved communication and the timely exchange of information experienced by CHWs using mHealth technology leads to improved quality of healthcare, particularly in obstetrics emergencies (Lemay et al, 2012). This improvement in communication and quick and easy access to information provides CHWs with access to institutional information resources, peer information resources and access to supervisors, consequently resulting in CHWs who are more knowledgeable and up to date with the best medical practices and procedures (Lee, Kim, & Chib, 2011). Countries with more CHWs have lower child mortality rates, improved maternal health, improved immunisation programs and are more capable of managing non-communicable diseases (100 People Foundation, 2012). Well-defined and implemented community healthcare programs improve the productivity of health systems, lower the costs of medical services and reduce the number of people having to visit trained specialised healthcare professionals, such as surgeons (Braun et al, 2013). It also creates employment opportunities in local communities (World Health Organization, 2013).

2.8. Mobile Health in prenatal, maternal and child healthcare

In 2010 it was reported that a woman died every 90 seconds due to complications experienced during pregnancy or childbirth, resulting in more than 340 000 deaths per year (Hogan, et al., 2010). One of the key contributing factors to this high mortality rate is the long time that it takes to deliver emergency care services to women during obstetric emergencies (Noordam et el., 2011). These delays are mainly incurred during decision

making processes when dealing with obstetric emergencies, the time it takes to transport patients to healthcare facilities and the time it takes to receive the appropriate care from a trained healthcare professional at a healthcare facility (Noordam et el., 2011).

According to Tamrat and Kachnowski (2012), mHealth assists in minimising the time barriers and facilitates the provision of healthcare, especially during obstetric emergencies and referrals. MHealth has a positive impact on the management of maternal, prenatal and neonatal health, assists in pregnancy management training, pregnancy risk assessment, pregnancy intervention, decision making and can play an important role at every stage of pregnancy and obstetric emergencies (Tamara et al., 2014). It is also assisting the United Nations to achieve its number 4 and 5 Millennium Development Goals (MDG) of reducing child mortality rates and improving maternal health (Tamrat & Kachnowski, 2012).

Studies in Ghana, Mali, Uganda, Malawi and Sierra Leone demonstrated that using mHealth to support CHWs delivering healthcare services to pregnant women resulted in a significant reduction in maternal deaths and increased supervised birth rates (Noordam et al., 2011). In Uganda and Mali, CHWs equipped with mHealth technology were able to communicate directly with biomedical health systems enabling them to reduce delays during medical emergencies and decision making (Noordam et al., 2011). In Zanzibar, mHealth technology using short message services (SMS) to send basic health education, information and reminders of routine check-up appointments to pregnant women exhibited a reduction in delays during obstetric emergencies (Noordam et al., 2011).

Tamrat and Kachnowski (2012) analysed 34 reports on using mHealth in maternal and new-born health programs and reported that utilising mHealth technology in prenatal, maternal and child care has a positive impact on antenatal care, maternal childcare and maternal mortality rates. They also reported the following (Tamrat & Kachnowski, 2012):

- 55% of families use a mobile phone during pregnancy emergencies.
    - 72% of these called medical services
    - 57% of these inquired about medical advice
    - 21% of these inquired about financial aid

- MHealth can expedite obstetric emergency referrals.
- MHealth promotes positive prenatal behaviour amongst pregnant women.
- MHealth decreases maternal mortality rates.
- MHealth assists in decision making and identifying health facilities with the appropriate services.
- MHealth technology supports midwives during emergencies by providing them access to emergency procedural information.
- Pregnant women who use mHealth technology are more knowledgeable and confident.

The audience of maternal and neonatal mHealth solutions include pregnant women, physicians and community healthcare workers. Table 2 presents examples of mHealth apps that are currently available to pregnant women and a description of their functionalities and mobile platforms:

Table 2: *Examples of mHealth apps available for maternal and neonatal health*

App Name	Description and Function	Platform	Price
Hesperian	Hesperian provides its users information on how to stay healthy during pregnancy; how to recognise danger signs during pregnancy, birth and after birth; what to do when danger arises; when to refer a woman to emergency care; instructions for CHWs on how to take blood pressure readings, on how to treat someone and to stop bleeding (Smyth, 2012).	Android	Free with In-App Purchases
MommyMeds	MommyMeds provides users the ability to determine whether certain drugs are safe to	Android, IOS	Free with In-App Purchases

	use during pregnancy or breastfeeding (MommyMeds for Mothers, 2017).		
Smarter Pregnancy	Smarter Pregnancy assists in improving nutrition and diet during pregnancy. This is achieved by identifying and improving modifiable nutrition and lifestyle risk factors on a large scale. Consequently, reproductive and pregnancy results improved and perinatal morbidity and mortality rates are expected to be reduced (van Dijk, et al., 2017).	Web based	Free
Sprout	Sprout provides foetal development charts, timelines, to-do lists, a kick counter to keep track of one's pregnancy health and doctor visits (Sprout, 2017).	Android, IOS	Free with In-App Purchases

2.9. MHealth Challenges, Constraints and Failures

According to Gurupur et al. (2017), there are challenges and constraints that require consideration when implementing mHealth technology (Gurupur & Wan, 2017). Gurupur et al. (2017) categorised these challenges as shown in Table 3:

Table 3: *MHealth Challenges and Constraints*

Challenge	Key components	Definition
Usability	- User Interface Design	Refers to the complexities derived from designing and developing mHealth apps that support different mobile devices with different hardware specification. E.g. screen sizes, camera specifications. MHealth apps must be efficient, easy to learn and use, mitigate input errors and time taken to recover from errors, provide speed, performance, satisfaction and user acceptance.
Reliability	- Accuracy of the mHealth Records	MHealth data must be accurate with low error rates. Data shared amongst healthcare providers, mHealth apps and vendors must be in formats that can be easily assimilated.
System Integration	- Interoperability - System Design	MHealth apps may be required to share information, such as Electronic Health Records (EHT), with other healthcare service providers. These apps must be able to securely share and exchange information with other apps and services. MHealth apps should also be scalable and allow for integration with other systems.
Data Security and Privacy	- Confidentiality - Data Storage - Data Access	MHealth apps generate, store and share confidential health and patient data. Data must be securely stored in secure locations. Data must only be accessible using secure transmission channels.
Network Access	- Availability - Speed and Strength	The availability, speed and strength of wireless network services where the mHealth app will be used must be able to securely transmit and receive data.

Network access and network realiabilty were two challenges that attributed to the success or failure of an mHealth solution and required consideration when estimating the amount of data used by an mHealth application and its bandwidth requirements (Gurupur et al., 2017). These challenges affected the ability to provide secure wireless network access that was strong enough to transmit and receive data safely and securely and to provide accurate patient information without any data loss (Gurupur et al., 2017). Network realiabilty affects Internet speeds and fluctuations, can negatively impact an mHealth app's user experience and should be considered when estimating an app's bandwidth usage, identifying the app's target market and target markets locations (Helios, 2017).

Aranda-Jan, Mohutsiwa-Dibe and Loukanova analysed 44 studies on mHealth and reported that the long-term benefits of implementing mHealth technology in developing countries are uncertain and that mHealth technology only assists in data collection but does not help resolve the actual healthcare problems in these countries (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014).

MHealth technology implementations were dependent on the infrastructure availability in the areas where it was used. The lack of reliable wireless Internet access, the cost of Internet data and electricity were three of the primary factors that contributed to the failure of mHealth technology in developing countries (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014). Consequently, the long-term cost effectiveness and feasibility of mHealth technology remained unknown and the limited funding in developing countries made it difficult to implement mHealth technology programs for extensive periods (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014).

2.10. Summary

Mobile Health (mHealth) is assisting to combat non-communicable diseases and enabling CHWs to deliver healthcare services to local communities (Braun etal., 2013). It is also assisting women during pregnancy by providing them access to healthcare information, advanced healthcare services and tools to assist in the management of maternal, prenatal and neonatal health emergencies (Braun etal., 2013). Mobile gamification techniques and data visualisation are assisting with creating mHealth apps that are engaging and

encourage positive user behaviour, ultimately assisting with user acceptance of mHealth apps (Miller, Cafazzo, & Seto, 2016).

However, implementing mHealth apps present some challenges and constraints (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014). According to Gurupur & Wan (2017), these challenges includes the availability, speed and strength of wireless network services where the mHealth app will be used and the ability to securely transmit and receive data. These challenges contribute to pertinent factors, such as the overall costs of implementing an mHealth app and user experience and acceptance (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014), which can determine the success or failure of a mHealth app. Network access and network reliability are also challenges that require consideration as they can cause fluctuations in Internet speeds, intermittent network access and connectivity and can be negatively affected by the amount of data used by an mHealth app and the app's bandwidth resources (Helios, 2017).

Chapter 3: Native, Web and Hybrid Mobile App Architectures

Mobile app architecture is a set of design patterns and techniques used when developing structured mobile applications that are based on mobile industry and vendor specific standards (Pisuwala, 2018). There are currently three well established mobile app architectures: Native, Web and Hybrid (Charland & Leroux, 2011). These architectures are generally comprised of a presentation layer, business layer and data layer (Helios, 2017). Figure 3 is an overview of the various elements and layers of a mobile app's architecture.

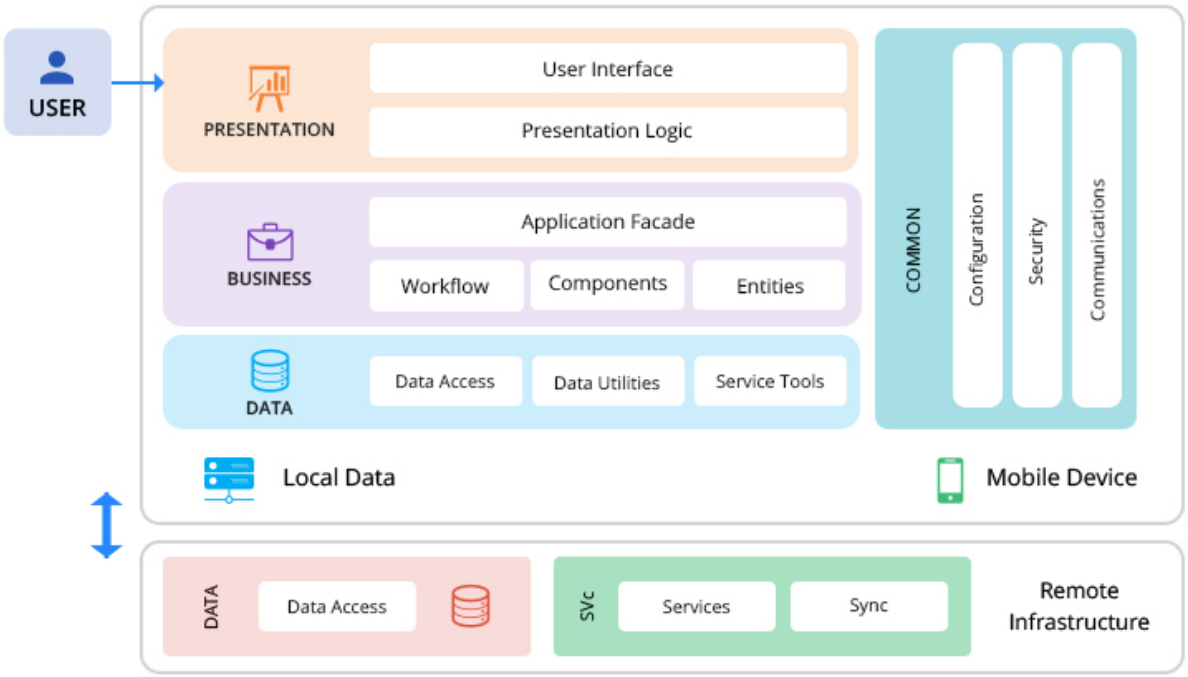


Figure 3: Mobile app architecture's elements and layers (Pisuwala, 2018).

At the start of the smartphone mobile app era, mobile application developers were obliged to develop mobile apps in the native language of the mobile app's intended mobile platforms (Wasserman, 2010). Different versions of these apps had to be developed for each type of targeted mobile device and mobile Operating System (OS). The native app development process was labour intensive, time consuming and expensive (Wasserman, 2010).

However, mobile app architectures and development platforms have rapidly evolved in recent years (Wasserman, 2010). Advancements in mobile technology, Mobile Development Frameworks (MDF), Cross-platform Development (CPD) tools and Web programming technologies have provided alternatives to developing explicit native mobile apps for specific mobile platforms (Jobe, 2013). Figure 4 displays an overview of the three established mobile app architectures (Charland & Leroux, 2011).

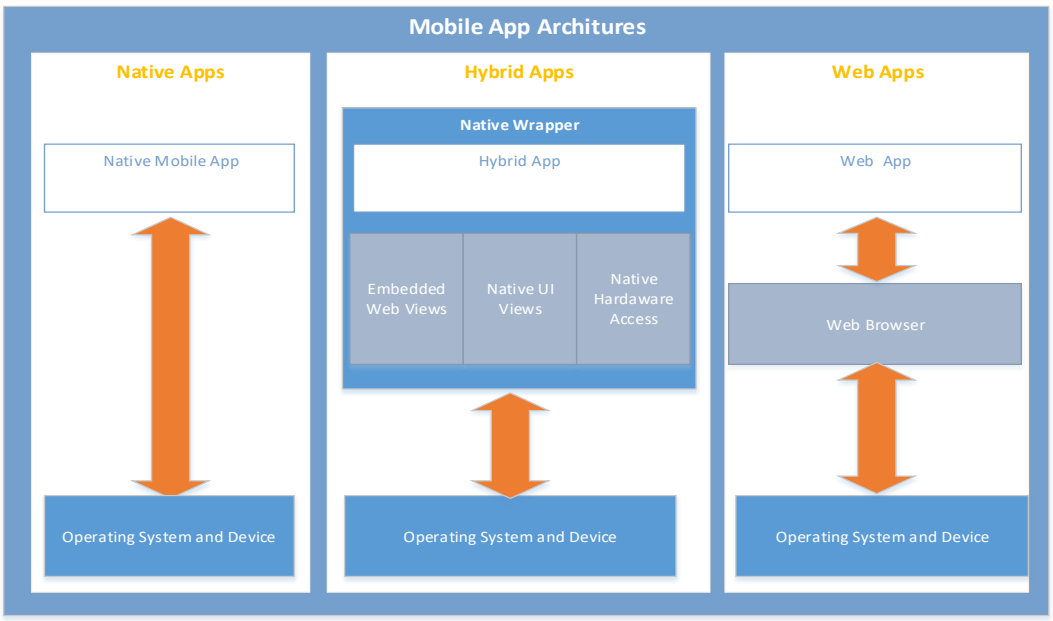


Figure 4: Native, Hybrid and Web App Architectures

According to Dalmasso et al. (2013), each mobile architecture has advantages and disadvantages. It is recommended that the high-level factors presented in Table 4 are considered when selecting a mobile architecture (Dalmasso et al., 2013).

Table 4: *High-level decision factors to consider when selecting mobile app architecture*

Criterion	Native	Responsive and Web	Hybrid
Quality of UX	Excellent	Very good	Not as good as
Quality of Apps	High	Medium	Medium to low
Potential users	Limited to specific mobile platforms	Maximum users. Including smartphones, tablets and other feature phones	More potential users of different platforms than
Development cost	High	Low	Medium to low
Security	Excellent	Depends on browser security	Not good
Supportability	Complex	Simple	Medium to
Ease of updating	Complex	Simple	Medium to
Time to market	High	Medium	Short
App extension	Yes	Yes	Yes

3.1. Native apps

Native apps are mobile applications that have been developed for specific targeted mobile platforms (Jobe, 2013). Native apps are usually downloaded from digital distribution platforms, such as the Apple App Store or Google Play Store (Serrano, Hernantes, & Gallardo, 2013). Native apps must be installed directly onto a mobile device before they can be used (Jobe, 2013).

Initially, native apps had to be developed in the specific programming language supported by the apps' target device platforms (Wasserman, 2010), for example, Objective C for Apple iPhone and Java for Android. Native apps will only execute on the mobile operating systems and platforms that they are developed for (Jobe, 2013). Figure 5 illustrates examples of programming languages and their supported mobile platforms.

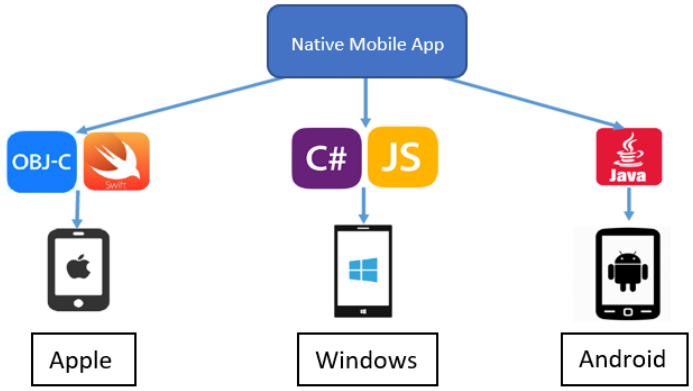


Figure 5: Examples of programming languages and their supported mobile platforms.

Table 5 illustrates a list of programming languages and skills that were required to develop a native app that targeted nine different mobile platforms (Charland & Leroux, 2011).

Table 5: *Mobile Operating System and Supported Programming Language*

OS	Programming Language
Apple iOS	C, Objective C
Google Android	Java

RIM BlackBerry	Java (J2ME)
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.NET
Window 7 Phone	.NET
HP Palm WebOS	HTML/CSS/JS
MeeGo	C, C++, HTML/CSS/JS
Samsung Bada	C++

However, since the introduction of Cross-Platform Development (CPD) tools such as Xamarin and React Native, mobile app developers are now able to develop native mobile apps in languages such as C# or JavaScript using a PC or a Mac computer (Microsoft, 2016). CPD tools enable mobile developers to write an app's code once and deploy the app on many different types of mobile devices (Redda, 2012). CPD tools can interpret and compile mobile app code into native assemblies for specific targeted devices (Microsoft, 2016).

Developing and supporting native apps usually requires larger budgets, multiple skill sets and longer time to develop but this is still recommended for high performance apps and mobile games because they are faster, more responsive and have full access to mobile device hardware and native APIs (Horda, 2017). Native apps also have full access to mobile device user interfaces, providing app users with smooth app interactivity features, such as multi-touch, double taps and pinch-spread functionality, ultimately delivering the best user experience (Jobe, 2013). Native apps can also be designed to function in an offline state when Internet connectivity is not available, slow and intermittent or not required (Dua, 2018).

3.2. Mobile Web Apps

Mobile Web apps are executed inside mobile Web browsers (Serrano, Hernantes, & Gallardo, 2013). No additional applications are required to be downloaded and installed on the mobile device. Web apps are hosted on Web servers and can be optimised for mobile devices using Web technologies such as HTML5, CSS3, JavaScript and Asynchronous JavaScript And XML (AJAX). AJAX enables Web apps to asynchronously send, receive and

exchange data with a Web server in various formats, e.g. JSON and XML, without having to reload the entire Web page (W3 Schools, 2018).

Web based apps are more compatible, accessible and easier to maintain than native apps and can be executed on most mobile devices using a Web browser (Horda, 2017). Most modern Web apps are developed using responsive Web design techniques, HTML5, JavaScript and Cascading Style Sheets (CSS) (Jobe, 2013). Responsive Web apps can detect the type of mobile device that they are being executed on and automatically adjust the Web app's content accordingly by using the most appropriate CSS for the mobile device's settings and screen resolution (Jobe, 2013). Web apps can be integrated with high-level mobile features such as QR codes and text messaging and provides access to a much larger target audience (Horda, 2017). Web apps are cheaper and easier to maintain because there is only one version of the Web app's code that requires maintenance, updates and support (Jobe, 2013).

However, Web based apps have limitations. Web based apps execute in mobile device Web browsers and are not able to leverage off mobile device hardware such as cameras, GPS and accelerometer sensors, ultimately limiting their possibilities and functionalities. According to Serrano et al. (2013), Web apps can be categorised as follows:

- **Generic Mobile and Responsive Mobile Web Apps**– Generic mobile Web apps are mobile versions of a desktop Web app that have been designed and optimised to be used on a mobile device. These Web apps query the Web browser's user-agent identifier to determine if they are being browsed from a desktop computer or a mobile device (Serrano, Hernantes, & Gallardo, 2013). If the Web app detects that it is being browsed from a mobile device then the user-agent is redirected to a mobile version of the Web app (Serrano, Hernantes, & Gallardo, 2013). Responsive Web apps are similar to generic mobile Web apps, but they also employ responsive Web design techniques (Horda, 2017). Responsive Web design is the practice of applying different or multiple Cascading Stylesheets (CSS) to support different types of mobile Web browsers and different types of mobile devices (Horda, 2017). Responsive Web apps determine the type of mobile device being used and then

apply the most appropriate CSS to transform and deliver the Web content in the most optimised format for the specific type of mobile device and its screen resolution (Serrano, Hernantes, & Gallardo, 2013). Determining the most appropriate CSS is done by the Web server serving the Web app and the CSS is applied at the server level, the client level or both (Serrano, Hernantes, & Gallardo, 2013). This enables the Web app to be used from different types of mobile devices without the loss of performance and user experience (Serrano, Hernantes, & Gallardo, 2013).

- **Hybrid Web Apps** – Hybrid Web apps are cross-platform mobile apps that combine elements of native and Web app technology (Serrano, Hernantes, & Gallardo, 2013). (Please see Section 3.3 Hybrid Web Apps.)

3.3. Hybrid Web Apps

Hybrid apps are Web based apps that are developed using technologies such as HTML5, JavaScript, AJAX and CSS. They are developed using Mobile Application Development Frameworks (MADF), such as Apache Cordova and Titanium, and Web technologies such as HTML5, JavaScript and CSS that are then embedded within a thin native layer container (Dalmasso etal., 2013). MADFs assist in the rapid development of cross-platform mobile applications (Dalmasso etal., 2013). The hybrid mobile app architecture enables developers to develop cross-platform mobile apps once and execute them on different types of mobile devices. This is achieved by embedding the Web app within a thin native layer container (Dalmasso etal., 2013). The native layer container provides access to mobile hardware, native APIs and platform features available on mobile devices (Dalmasso et al., 2013). Hybrid apps are downloaded from publishing platforms, such as Apple's iTunes Store, installed onto a mobile device and behave similarly to native apps (Serrano, Hernantes, & Gallardo, 2013). Hybrid apps address some of the challenges that native and mobile Web apps present and are faster and easier to develop than native apps (Dalmasso et al. 2013). However, hybrid apps do present challenges such as slow performance, poorer quality user experiences and lower levels of security (Serrano, Hernantes, & Gallardo, 2013).

Table 6 illustrates some factors to consider when choosing between a hybrid or Responsive/Web app architecture.

Table 6: *Factors to consider when choosing between a hybrid or Responsive/Web app (Serrano, Hernantes, & Gallardo, 2013)*

Considerations	**Hybrid**	**Responsive/Web**
Effort to support platforms and versions	Medium	Low
Device access capabilities	Full	Partial
User experience	Full	Medium
Performance	Very high	High
Upgrade in the client	Needed	Not needed
Ease of distribution	Medium	High
Approval cycle	Optional	Not required
Monetization in app store	Available	Not Available

Table 7 displays the app features, mobile hardware and native API access and UI interaction and gestures factors to consider when choosing between native, hybrid and Web app architectures.

Table 7: *Factors to consider when choosing between native, hybrid or Web app architecture (Serrano, Hernantes, & Gallardo, 2013).*

	Native	**Web**	**Hybrid**
App Features			
Graphics	Native APIs	HTML, Canvas, SVG	HTML, Canvas, SVG
Performance	Fast	Slow	Slow
Native look and feel	Native	Emulated	Emulated
Distribution	Appstore	Web	Appstore
Mobile Hardware and Native API Access			
Camera	Yes	No	Yes
Notifications	Yes	No	Yes
Contacts, calendar	Yes	No	Yes

Offline storage	Secure file storage	Shared SQL	Secure file system, shared SQL
Geolocation	Yes	Yes	Yes
Gestures and UI Interaction			
Swipe	Yes	Yes	Yes
Pinch/spread	Yes	No	Yes
Connectivity	Online and offline	Mainly online	Online and offline
Development skills	Objective C, Java, etc.	HTML5, CSS, and JavaScript	HTML5, CSS, and JavaScript

3.4. Selecting a mobile app architecture

Native, Web and Hybrid app architecture have advantages and disadvantages (Serrano, Hernantes, & Gallardo, 2013). It is recommended that businesses and organisations consider several factors before formulating a mobile strategy, establishing a mobile presence and investing in a mobile app architecture (Summerfield, 2018). These factors include:

- Aligning the mobile strategy with the business' strategy (Summerfield, 2018).
    - Does the business or organisation have a need for a native app, a Web mobile app or both?
    - Does the business have the time and resources to invest in a native app, Web app or both?
- The target audiences of the mobile strategy (Jobe, 2013).
    - Does the target audience require a Web based app or native mobile app?
    - Will the target audience require the mobile app to perform on many different types of mobile devices and mobile operating systems?
    - Will the app be used by a target audience in remote locations with intermitted network connectivity?
    - Is the cost of mobile data in the regions of your target audience very high?

- The required features and functions of the mobile app (Jobe, 2013).
    - Will the mobile app be required to access mobile hardware features such as cameras, Global Positioning Systems (GPS) and mobile accelerometers?
    - Will the mobile app require instant and fast data transfers and version updates?
    - Will the mobile app be required to operate on many different types of mobile platforms and operating systems?
    - Will the app be required to operate in remote areas and in areas with outdated communications infrastructure?
- The available budget to develop, maintain and operate the mobile app (Horda, 2017).
    - Does the business or organisation have the budget to develop and maintain multiple versions of a native mobile app that targets different mobile operating systems such as Apple's IOS versions and different Android versions?
    - Is the mobile app required to be data efficient and use low amounts of data during operation?
- User Interface design (Helios, 2017)
    - Mobile app user interfaces must be intuitive, interactive and easy to use.
    - Badly designed UIs will negatively affect users' experiences.
- The available bandwidth and data usage costs in the target locations where the mobile app will be used (Helios, 2017).
    - Slow Internet can negatively affect users' experiences.
    - High Internet usage data costs can negatively affect users' experiences and ultimately determine the success or failure of a mobile app (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014).

3.5. Summary

There are currently three well established mobile app architectures: Native, Web and Hybrid (Charland & Leroux, 2011). Each mobile app architecture has its advantages and

disadvantages and presents its own factors to consider when selecting a mobile app architecture (Serrano, Hernantes, & Gallardo, 2013).

Native apps have full access and control over mobile device hardware and native mobile APIs, deliver the best performance and can operate in an online and offline state (Serrano, Hernantes, & Gallardo, 2013). They also provide excellent user experiences and higher levels of security (Dalmasso et al., 2013). However, native apps must be developed in specific programming languages for specific targeted mobile devices (Jobe, 2013) and must be installed onto a mobile device before they can be executed. Consequently, a version of the mobile app must be developed for each targeted mobile device, which could lead to higher development costs, increase in its time to market and make it complicated to support and to update (Dalmasso et al., 2013).

Web apps provide good user experiences, can be accessed from smartphones, tablets and feature phones, ultimately providing the maximum accessibility to potential users (Serrano, Hernantes, & Gallardo, 2013). They are the easiest to deploy and support and are usually the most cost effective mobile app architecture to implement (Dalmasso et al., 2013). However, mobile Web apps are not able to access mobile device hardware and APIs and this is the most insecure mobile app architecture to implement (Dalmasso et al., 2013).

Hybrid apps are cross-platform mobile apps that combine elements of native and Web app technology (Serrano, Hernantes, & Gallardo, 2013). They are Web apps that are embedded within a thin native layer container (Dalmasso et al., 2013). Hybrid apps address some of the challenges that native and mobile Web apps present but they present challenges such as slow performance, poorer quality user experiences and lower levels of security (Serrano, Hernantes, & Gallardo, 2013).

It is recommended that businesses and organisations consider the advantages and disadvantages such as performance, security, mobile hardware and API access that each mobile app architecture presents (Dalmasso et al., 2013) and factors such as, aligning the mobile strategy with the business' strategy, target audiences, development budget, user

interface requirements, bandwidth and infrastructure availability where the mobile app will be used, before investing in a mobile app architecture (Summerfield, 2018).

Chapter 4: Research Methodology and Design

4.1. Research Objective

The overall amount of data used by mHealth apps is a pertinent factor contributing to the challenges, successes or failures of mHealth apps (Aranda-Jan, Mohutsiwa-Dibe, & Loukanova, 2014). It can influence the long term operational cost of using the mobile app and have a negative effect on its users' experience and acceptance. The purpose of the research was to investigate and answer the following question:

"Is there a significant difference in the amount of data used by an mHealth application that is used to manage maternal and neonatal health when implemented using a native message passing Android architecture versus using a mobile Web based architecture?"

By monitoring, recording and analysing the amount of data transferred between mobile clients and servers while operating the mHealth app, one can determine and calculate the mean amount of data that the mHealth app will consume.

4.2. Research Methodology

The purpose of this research was to investigate the amount of data used by an mHealth app when implemented using a native mobile architecture and different mobile Web architectures. Three versions of an mHealth app called Village Health was created. The app is used to manage maternal and neonatal health. The different versions of the Village Health app were developed as:

1. A native Android mHealth app
2. A Web based mHealth app
3. A Web based mHealth app that uses Asynchronous JavaScript and XML (AJAX)

The three versions of the Village Health app provided the same functionalities and were tested with the same input data sets. The objective of the study was to measure the amount of data transferred between clients and servers while conducting standard operations, data analysis and decision-making processes by the three different types of mobile app architectures.

The data collected during the experiments was then analysed, compared and used to ascertain if there was a significant difference in the amount of data used by an mHealth app when implemented using a message passing Android architecture versus mobile Web based architectures. The experiment data captured by measuring the data transfers between clients and central servers provided the quantitative data required to evaluate the overall amount of data used when implementing a mobile health app that is used to manage maternal and neonatal health.

4.3. Application Design

A native Android based app, a mobile Web app and an AJAX enabled Web app used to manage maternal and neonatal health were developed:

1. **Native Android app:** The Android app was installed on a local mobile device. It transferred and stored data on a central database server. The solution queried and interacted with the central server and utilised the mobile device's computing resources and data tools to perform data analytics and charting directly on the mobile device.
2. **Web app:** The Web app was hosted on a Web server and comprised of three tiers:
    - Presentation Tier - Web based User Interfaces (UI).
    - Business Tier - Web API.
    - Data Tier – MS SQL Server and stored procedures.

    All business logic processes occurred on the Web server. Data generated by the Web app was stored on a central database. The Web interface enabled users to query and interact with the database. The Web interface communicated with the database using a Web API. The Web user interfaces and Web API were hosted on the same Web server. The Web server processed all client requests server-side and transferred responses back to the clients. Data analytic processes and chart visualisations occurred on the Web server and were transferred back to the client using standard HTTP requests and HTTP responses.

3. **AJAX Web app:** The AJAX Web app was similar to the Web app but used AJAX to asynchronously execute HTTP requests. All business logic processes occurred on the Web server. Data generated by the AJAX Web app was stored on a central database. The Web UI enabled users to query and interact with the database asynchronously without having to reload the entire Web page.

The native app used JSON format mark-up and only transferred the required data. (Please see Appendix I for an example of the native JSON data). The Web based apps transferred content data, request headers and response headers between clients and servers. (Please see Appendix C for an example of Web headers and content data sizes).

The different versions of the mHealth app provided the same end user functionalities but executed processes and transferred data differently.

4.3.1. Mobile Web Village Health Apps

The Web app and AJAX Web app provided users with Web based UIs that were compatible and accessible from most modern mobile devices that had a Web browser and an Internet connection. (Please see Appendix B for the Village Health Web interfaces). The Web apps had the same Web UIs and looked identical but used different methods to execute HTTP requests and responses. HTML header tags were used within the Web apps to disable all content caching and to ensure that the apps reloaded all website content.

The Web app reloaded the entire Web UI on each HTTP request. The AJAX Web app only loaded the Web user interface once then executed HTTP requests asynchronously without reloading the entire Web UI and only updated the parts of the Web UI that were affected.

The Web apps were developed with the following Internet technologies:

- **Microsoft Active Server Pages .NET (ASP.NET)** - ASP.NET is an open source Web framework used to build Web apps, services and websites (Microsoft, 2017).
- **Microsoft (MS) SQL Server 2017** - MS SQL Server is a relational database management system used for the storage and management of data (Rouse, 2018).

- **Transact Structured Query Language (T-SQL)** - Is Microsoft's extension to Structured Query Language (SQL) and is a database programming and querying language (Rouse, T-SQL , 2018).
- **Hyper Text Mark-up Language (HTML)** - HTML describes the structure of pages using a mark-up language. HTML elements are the building blocks of Web pages and are represented by tags (W3 Schools, 2017).
- **Cascading Style Sheets (CSS)** - CSS describes the style of an HTML document and how a HTML document's elements should be displayed (W3 Schools, 2017).
- **Microsoft Internet Information Services 7.0 (IIS 7.0)** - IIS is a secure and flexible Web server used for hosting Web applications, websites and services (Microsoft, 2017).
- **JavaScript** – JavaScript is a client-side scripting language used primarily for HTML and Web pages (W3 Schools, 2017).
- **Asynchronous JavaScript And XML (AJAX)** – AJAX enables Web pages to asynchronously send, receive and exchange data with a Web server in various formats, e.g. JSON and XML, without having to reload the entire Web page. This makes it possible to update parts of the Web page without reloading the entire Web page (W3 Schools, 2018).
- **C#** - Is a type-safe object-orientated language that runs on Microsoft's .NET Framework (Microsoft, 2015).
- **ASP.Net Web API** – Is a framework used to build HTTP services and RESTful applications on the .NET Framework and can be consumed from browsers and mobile devices (Microsoft, 2017).

Communication between Web clients and servers occurred via. the Hypertext Transfer Protocol (HTTP) and used HTTP requests and HTTP responses. HTTP is an application layer protocol and is the foundation of data communications on the Web (W3 Consortium, 1999). HTTP uses the Transmission Control Protocol (TCP), a transport layer protocol, to send and receive data (Clark, Jacobson, & Salwen, 1989). TCP uses the Internet Protocol (IP), an Internet layer protocol, that enables the successful delivery of data packets from a source host to a destination host (Clark, Jacobson, & Salwen, 1989). Business logic

processes, database queries and data analytic processes initiated by requesting clients were executed on the Web server and the responses were sent back to the clients over HTTP. ASP.Net charting Web controls were used to generate data visualisations on the Web server, which were then outputted to HTML data visualisation Web pages.

Figure 6 illustrates the Web apps' case registration and navigation interfaces (Please see Appendix B for the Web based Village Health app interfaces).

Figure 6: Village Health case registration and navigation user interface

4.3.2. Native Village Health App

The native version of the Village Health app was comprised of two separate app components (Please see Appendix A for the native Village Health Mobile Interfaces):

a) A CommCare native app component. CommCare is an open source mobile application development platform that enables individuals and organisations to develop and deploy native mobile applications (CommCare, Inc., 2017).
b) A native data analysis app component developed with the Android Studio Integrated Development Environment (IDE) and the Java programming language. The primary function of this analysis component was to present mobile users with an easy to read, graphical representation of the collected data, ultimately assisting users in trend spotting and decision making.

The CommCare app stored its data locally and synchronised its data to the CommCare central servers only when necessary. The native data analysis app queried the data stored on the database servers via a Web API. The retrieved data was then used to generate charts directly on the mobile device. The following is a list of technologies were used to develop the native solution:

- CommCare Mobile Development Platform.
- Android Studio Integrated Development Environment (IDE).
- Java Programming Language
- JavaScript Object Notation (JSON)
- Android Virtual Device (AVD) emulator.
- ASP.Net Web API – Web API is a framework used to build HTTP services that can be consumed by different client such as Web browsers and mobile devices (Microsoft, 2017).

Figure 7 illustrates the home screen and navigation interface of the CommCare mHealth app developed with the CommCare mobile development platform (Please see Appendix A for the native Village Health app's screen interfaces).

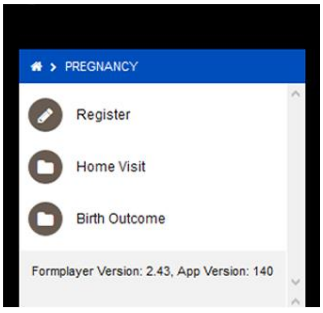


Figure 7: Village Health mHealth app home screen

4.4. Data Transfer Monitoring

The data input and output processes were monitored and the amount of data transferred during these processes were recorded. The recorded data provided the information required to calculate the overall amount of data used during processes.

The data transfer monitoring and capturing was achieved by using network management technologies such as proxy servers, network management software, network monitoring and network debugging technologies and Web browser developer tools that included:

- **Wireshark** – Wireshark is an open source network protocol analyser that is used for network analysis, network troubleshooting, network performance monitoring and network fault detection (Wireshark , 2017).
- **Telerik Fiddler** – Fiddler is a Web debugging proxy technology that enables its users to record, inspect and debug traffic from any type of Web browser (Telerik, 2017).
- **ProxyCap** – ProxyCap enables its users to redirect a computer's network connections through a proxy server. It supports protocols such as SOCKS, HTTPS, TCP, UDP and IPv6 (ProxyCap, 2017). (Please see Appendix E for the ProxyCap interface and configuration settings).
- **RawCap** – RawCap is a network sniffer that uses raw sockets and can sniff any interface that has an IP address (Netresec, 2017).
- **Web Browser Developer Tools** – Web browsers such as Firefox Developer Edition provide tools that can be used for debugging, network packet transfer monitoring, performance monitoring and browser data storage monitoring (Orgerea, 2017).

The experiments were conducted with a WIFI connection on an 8 megabits per second (Mbps) asymmetric digital subscriber line (ADSL) wireless local area network, a third generation (3G) mobile telecommunication connection and a second generation (2G) mobile telecommunication connection. The following data variables were captured:

- Overall data transfer sizes and its content types, including:
    - Data Content Bytes
    - Request Header Bytes
    - Response Header Bytes

4.5. Common Environment Configurations

The experiments were conducted with the Android Virtual Device (AVD) emulator on a computer system with the following specifications:

System Manufacturer: Hewlett-Packard
Processor: Intel Core i7-7500 CPU @2.90GHz
System Memory: 8GB
Operating System: Windows 10 Enterprise
System Type: 64-bit Operating System, x64 based processor

Network management software, Web browsers and mobile emulation software including Android Virtual Device (AVD), Wireshark, RawCap, ProxyCap, Telerik Fiddler and Firefox were installed on the computer system and appropriately configured for the experiments.

The following settings were configured and applied to the software for the experiments:

- ProxyCap was configured to ensure that all network traffic on ports 8888, 80, 443, 60001 and 60002 were intercepted.
- Wireshark, RawCap, Telerik Fiddler and Firefox Developer Edition networking tools were all configured to record and monitor all data transfer sizes between mobile clients and servers. (Please see Appendix F for the Firefox and Appendix D for the Wireshark filter configurations and sample data and Appendix F for Firefox filter configurations and sample data).
- The native Village Health app was installed onto the system via the AVD mobile emulator.
- The Web based Village Health solution experiments were conducted using the AVD Web browser.
- The data transfer sizes and data transfer times were captured, stored and analysed using Microsoft Excel and XLSTAT.

4.6. Native, Web and AJAX Web App Experiments

Data experiments on the native and Web apps were conducted using the AVD mobile emulator Web browser.

Figure 8 illustrates a high-level overview of the native, Web and AJAX Web apps' input request and response processes that occurred between mobile clients and the application servers. It also shows where the data transfer monitoring occurred.

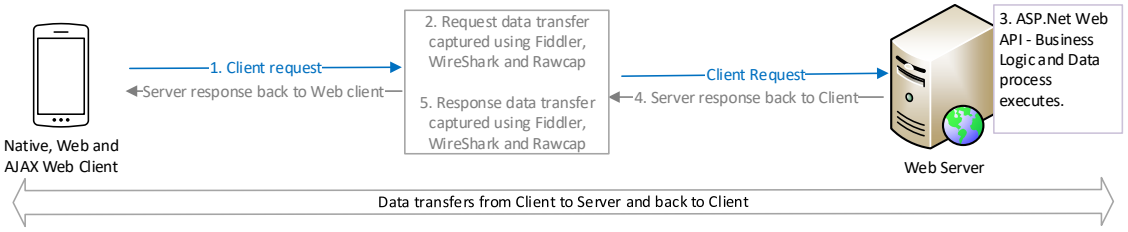


Figure 8: High level overview of the native, Web and AJAX Web Village Health apps input request and response processes.

Figure 8 represents the following sequence:

1. A mobile client executing the native, Web or AJAX Web app initiates a request.
2. The request data sizes are monitored and captured.
3. The request is handled by the server.
4. The server sends a response back to the mobile client.
5. The response data sizes are monitored and captured.

Data requests were sent from the native, Web and AJAX Web mobile clients. The requests and responses were monitored and recorded using RawCap, Wireshark, Fiddler and Firefox Developer Tools.

A dataset was created by having informal interviews with 20 women who had children. The interview included the following questions (Please see Appendix A for the full list of questions and the Village Health user input interfaces):

1. What is your name and surname?

2. What village or area do you live in?
3. Do you have children?
4. Did you experience any sick days during pregnancy?
5. How many of your children are male?
6. How many of your children are female?

The “Register a New Case” operational process available on the different versions of the mobile app was selected as the standard operational process used during the experiments because the process enabled the capturing of all the data acquired during the interviews and allowed for the greatest variation of data inputs accepted by the mobile app. The answers supplied by the women provided the data required and was used as inputs into the mobile apps’ interfaces.

Data experiments were conducted with all versions of the app using the Android Virtual Device (AVD) emulator executing on a laptop (Please see Section 4.5 for the laptop specifications). The experiments were conducted using WIFI, 3G and 2G mobile telecommunication connections. Identical data sets were inputted into the three different versions of the app in order to intercept and record the amount of data transferred between clients and servers. An example of the data input records can be seen Appendix G. The amount of data transferred between clients and servers were monitored and recorded. This included the input data, request and response header data and content data.

Figure 9 provides an overview of the data flow sequence between mobile clients and servers demonstrating an example of the data statistics that were captured.

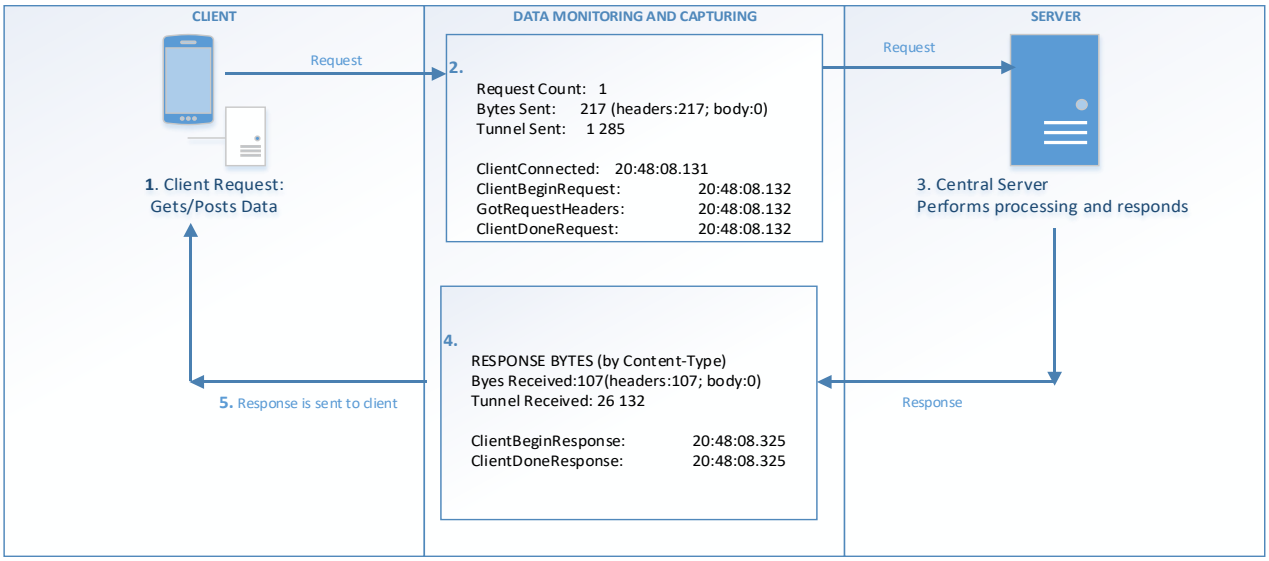


Figure 9: Village Health data flow sequence displaying and example of the data statistics that was recorded.

Figure 9 represents the following sequence:

1. Clients send a Get or Post data requests to a central server.
2. The header requests and data sizes are intercepted using Wireshark, RawCap and Fiddler and recorded.
3. The server handles the requests, executes the necessary processes to generate responses to the incoming requests and sends the responses back to the client.
4. The response data sizes are intercepted using Wireshark, RawCap and Fiddler and recorded.
5. The responses are sent to the requesting clients.

4.7. Experiment Procedures

The following are step by step instructions of the experiment procedure:

1. Ensure that the relevant network connection type is correctly configured i.e. Wi-Fi, 3G or 2G, and that the software used in the experiments are correctly configured for monitoring and intercepting all data communications between clients and servers.
2. Run the AVD mobile emulator on a laptop.
3. Native and mobile Web apps - Navigate to the “Register a New Case”:

a. Native app - Using the AVD mobile emulator, run the native based Village Health app, login into native Village Health app and select “Register” from the menu.
b. Web apps – Using the AVD mobile emulator, run the AVD Web browser and navigate to the Village Health website, login to the Web app and select “Register a New Case” from the menu.

4. Complete the required input data fields and screens for the “Register a New Case” process. (Please see Appendix A for the native Village Health app “Register a New Case” interfaces and Appendix C for the Web based app “Register a New Case” interfaces).
5. Input the required data and complete the “Register a New Case” process by submitting the data to the central server.
6. Use Wireshark and Fiddler to collect and record the data sizes transferred between clients and servers during the “Register a New Case” process.

7. Record the amount of data transferred between clients and servers when executing the “Register a New Case” process in Microsoft Excel.

Twenty data combinations were inputted into the “Register a New Case” user interface process screens. The data combinations were derived from the answers collected during the informal interviews that were conducted with the twenty women who had children. Figures 10, 11 and 12 are examples of the data attributes that were monitored and recorded with the various network monitoring tools.

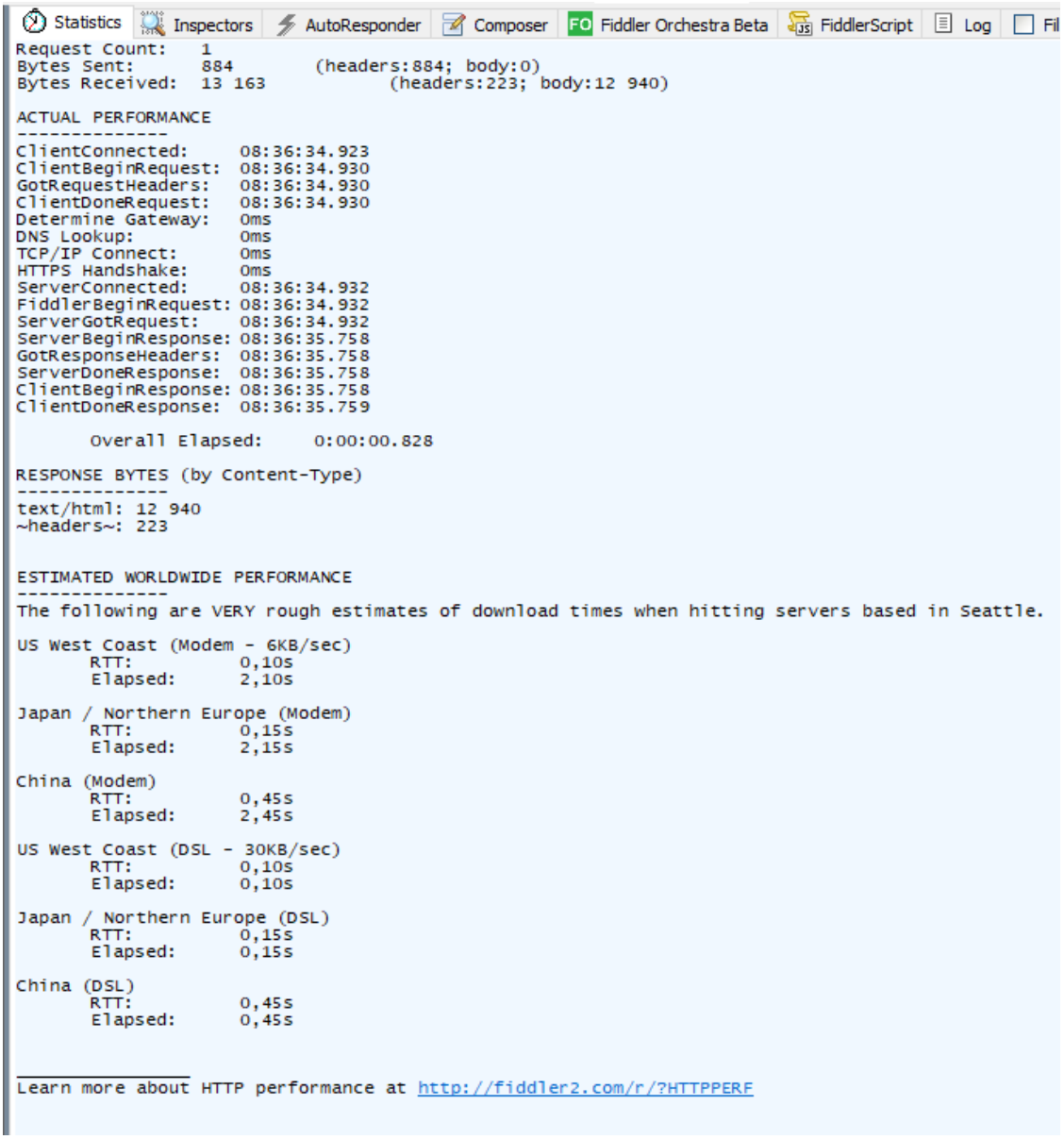


Figure 10: Fiddler data transfer statistics

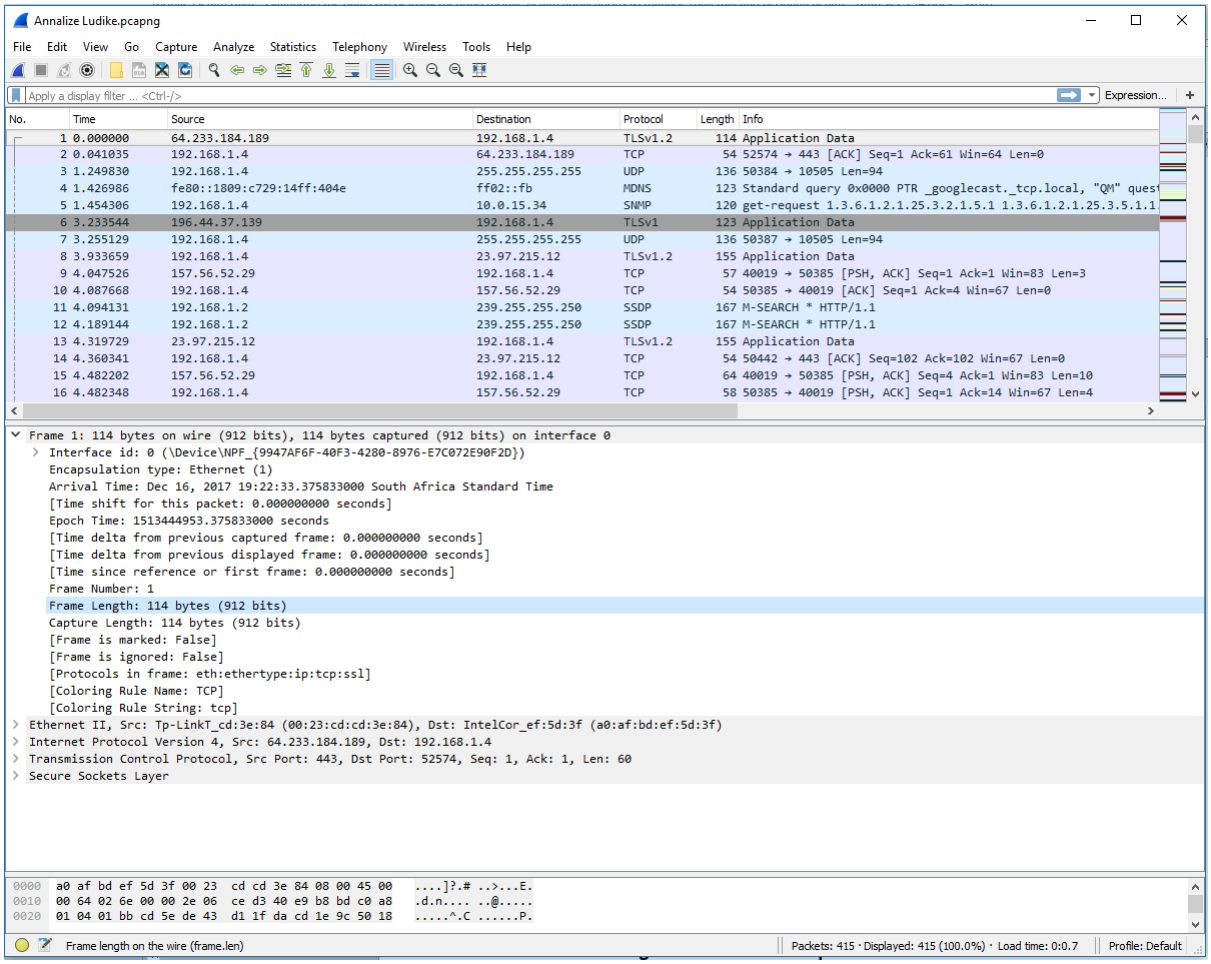


Figure 11: Wireshark Network Analyser record

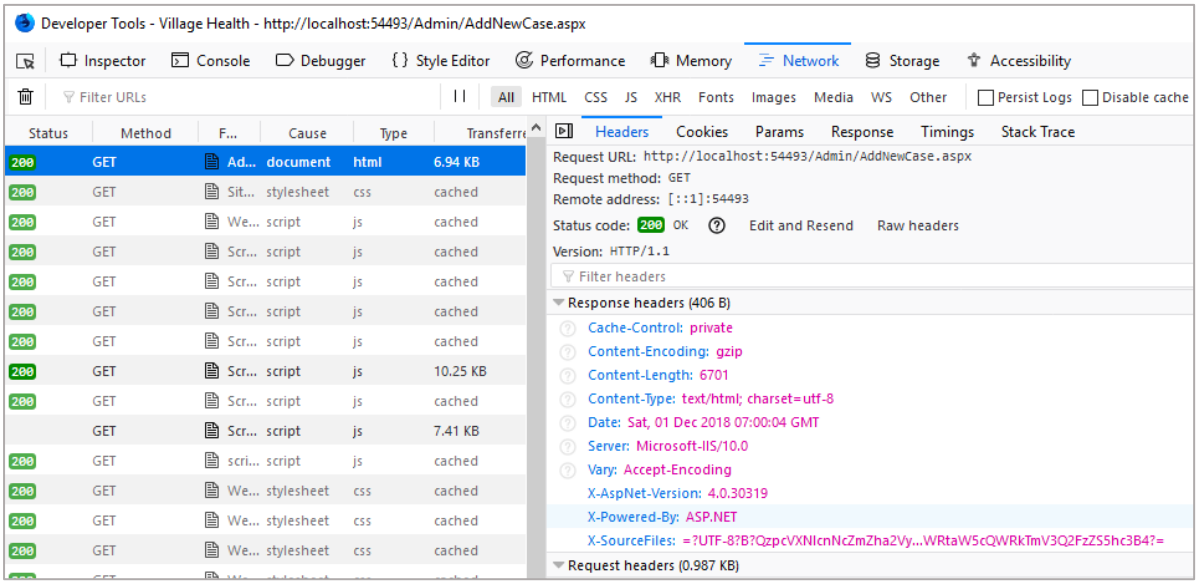


Figure 12: Firefox Web Developer Tools - network performance and data sizes

Chapter 5: Findings and Experiment Results

This chapter presents the results and findings of the research experiments. The experiments were designed to intercept, capture and analyse the amount of data transferred between clients and servers over networks during the standard usage of an mHealth app. The experiments provided the data required to determine the mean amount of data used while performing a functional process of an mHealth app, ultimately providing the data required to determine if there was a significant difference in the amount of data used by a mobile app that is implemented using different mobile app architectures.

The experiments were designed and conducted to test the following hypothesis:

Different versions of an mHealth app that have been implemented using a native mobile architecture, a standard mobile Web architecture and mobile Web architecture that uses technologies such as AJAX will demonstrate significant differences in the total amount of data used when performing a standard functional process such as registering a new case on an mHealth pregnancy monitoring app.

The total amount of data used by the mobile app was calculated by adding the amount of data sent and the amount of data received by the mobile app when performing processes. The native app sent and received data using the TCP protocol and the Web apps used the HTTP application layer protocol that used the TCP protocol to send and receive data. The calculation used to determine the overall amount of data used by the native and Web apps is expressed in the following formula:

Total Amount of Data Sent + Total Amount of Data Received = Total Amount of Data Used

The data was sent and received using the TCP protocol and were comprised of TCP headers, IP headers and payloads. A TCP header is comprised of TCP overhead data that is required to enable the secure and reliable transfer of data. Headers can contain TCP packet information such as packet sources, destinations, packet sequence numbers and checksums to ensure that the data that is being transferred is not corrupted. The size of a TCP header can range from 20 to 60 bytes and contains ten default fields totalling 20 bytes

and optionally contain additional data of up to 40 bytes (Clark, Jacobson, & Salwen, 1989). These fields include the following sequences and sizes (Mitchell, 2019) :

1. Source TCP port Number (2 bytes)
2. Destination Port Number (2 bytes)
3. Sequence Number (4 bytes)
4. Acknowledgment Number (4 bytes)
5. TCP Data Offset (4 bits)
6. Reserved Data (3 bits)
7. Control Flags (Up to 9 bits)
8. Window Size (2 bytes)
9. TCP Checksum (2 bytes)
10. Urgent Pointer (2 bytes)
11. TCP Optional Data (0-40 bytes)

A payload is comprised of the application data that needs to be transferred. Payloads can contain information such as application input data and form post data. (Please see Figure 24 for an example of a TCP data packet dissection).

5.1. Native App Data Usage Analysis

Figure 13 illustrates the amount of data sent, received and the total amount of data used when registering new cases using the native app.

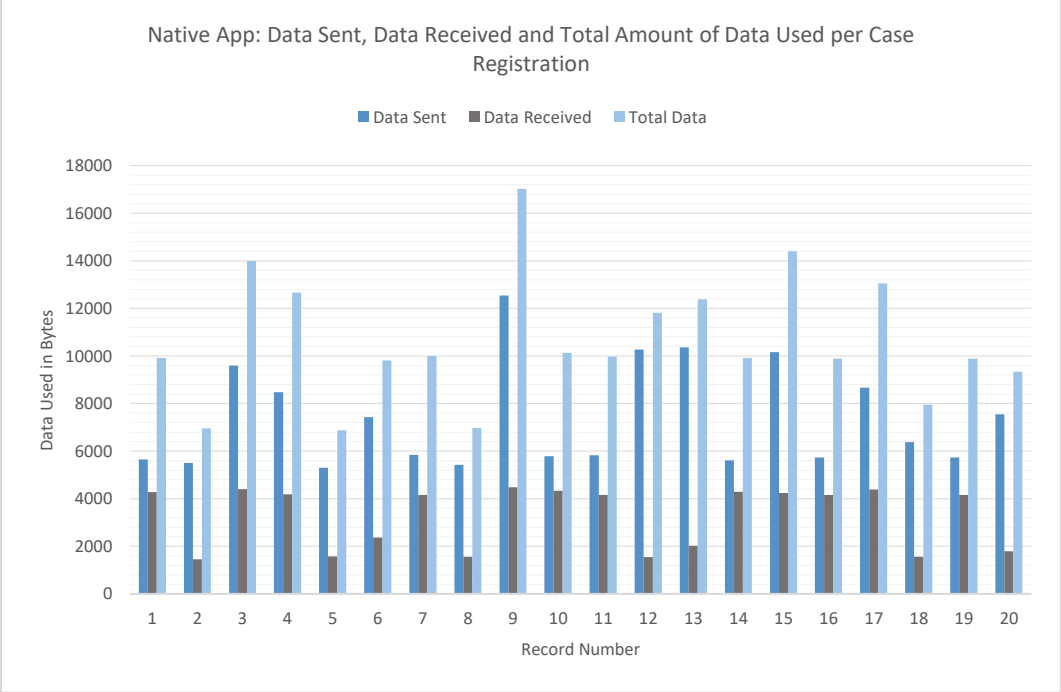


Figure 13: Native App - Data sent, data received and total amount of data used.

Figure 14 shows that the mean amount of data sent by the native app was greater than the mean amount of data received and makes up most of the total amount of data used by the native app during the register a new case process.

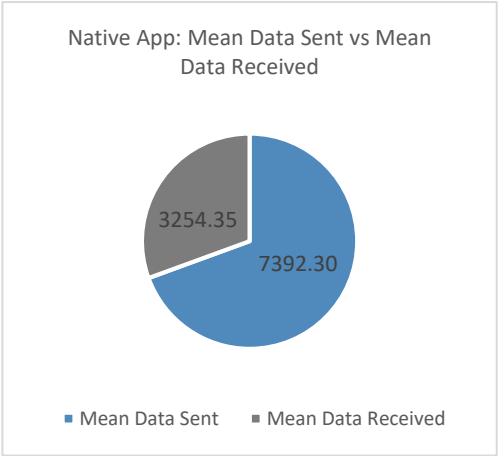


Figure 14: Native App: Data sent vs. Data received.

The mean amount of data sent by the native app was 7 392,3 bytes and the mean amount of data received by the native app was 3 254,35 bytes.

Figure 15 illustrates the mean amount of overhead data versus the mean amount of payload data transferred by the native app when performing standard procedures with the native app. Overhead data was 10,7% of the total amount of data used by the native app.

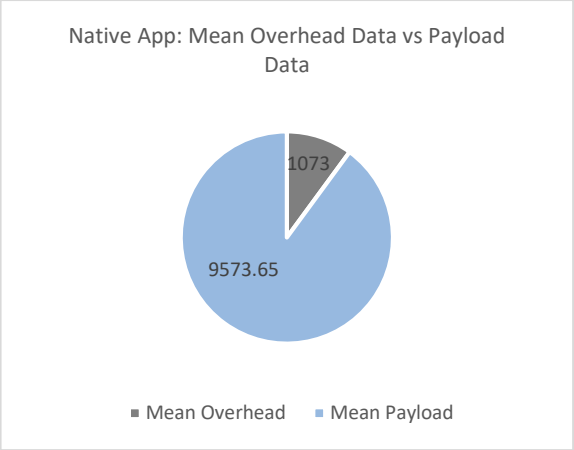


Figure 15: Native App: Mean Overhead Data sent vs. Payload Data.

Table 8 illustrates the data statistics produced by the native app when registering a new case.

Table 8: *Native App Data Statistics in Bytes: Data used by the native app when registering a new case*

	Data Sent	**Data Received**	**Total Data**	**Overhead Data**	**Payload**
Mean	7 392,3	3 254,35	10 646,65	1 073	9 573
Standard Deviation	2 186,48	1 290,67	2 661,87	235,33	2 502,84
Median	6 112	4 152	9 949	1 016	9 573,65

The total amount of data used by the native app when executing the register a new case process was the combined amounts of all the data packets, including headers and the payload data sent and data received between the client and server. Figure 16 illustrates the data communication sequence of record 4 of the native app's experiment.

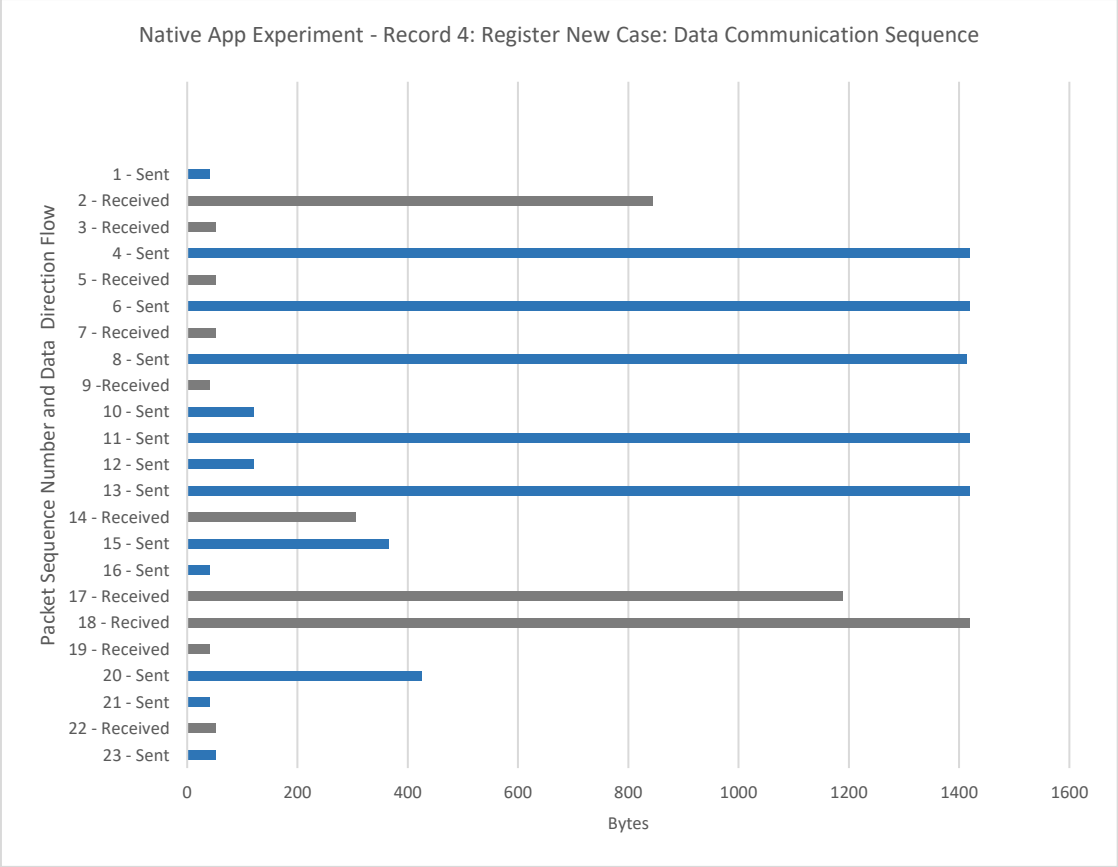


Figure 16: Native App – Data communication sequence of "Register a New Case" process.

Table 9 provides information about the individual data packets from record 4 of the native app experiment's data communication sequence.

Table 9: *Native App Experiment - Record 4. Native App Data Packet Communication Sequence Information*

Packet Sequence Number	Time	Data Flow Direction	Source	Destination	Protocol	Total Length (Bytes)	Packet Information
1	12:46:44 AM	Sent	192.168.1.4	146.20.47.186	TCP	40	51288 > 443 [ACK] Seq=5005 Ack=3601 Win=16384 Len=0
2	12:46:44 AM	Received	146.20.47.186	192.168.1.4	TCP	845	Application Data
3	12:46:44 AM	Received	146.20.47.186	192.168.1.4	TCP	52	443 > 51288 [ACK] Seq=2796 Ack=5005 Win=43008 Len=0 SLE=3632 SRE=4852
4	12:46:44 AM	Sent	192.168.1.4	146.20.47.186	TLSv1	1420	[TCP Retransmission] 51288 > 443 [PSH, ACK] Seq=3472 Ack=2796 Win=17152 Len=1380
5	12:46:44 AM	Received	146.20.47.186	192.168.1.4	TCP	52	443 > 51288 [ACK] Seq=2796 Ack=3472 Win=39936 Len=0 SLE=3632 SRE=5005
6	12:46:44 AM	Sent	192.168.1.4	146.20.47.186	TLSv1	1420	[TCP Retransmission] 51288 > 443 [PSH, ACK] Seq=2092 Ack=2796 Win=17152 Len=1380
7	12:46:44 AM	Received	146.20.47.186	192.168.1.4	TLSv1	52	[TCP Window Update] 443 > 51288 [ACK] Seq=2796 Ack=2092 Win=36864 Len=0 SLE=3632 SRE=5005
8	12:46:44 AM	Sent	192.168.1.4	146.20.47.186	TCP	1413	Application Data

9	12:46:44 AM	Received	146.20.47.186	192.168.1.4	TLSv1	40	443 > 51288 [ACK] Seq=2796 Ack=2092 Win=34816 Len=0
10	12:46:45 AM	Sent	192.168.1.4	146.20.47.186	TLSv1	120	51288 > 443 [PSH, ACK] Seq=3552 Ack=2796 Win=17152 Len=80 [TCP segment of a reassembled PDU]
11	12:46:45 AM	Sent	192.168.1.4	146.20.47.186	TCP	1420	51288 > 443 [ACK] Seq=2172 Ack=2796 Win=17152 Len=1380 [TCP segment of a reassembled PDU]
12	12:46:45 AM	Sent	192.168.1.4	146.20.47.186	TCP	120	51288 > 443 [PSH, ACK] Seq=2092 Ack=2796 Win=17152 Len=80 [TCP segment of a reassembled PDU]
13	12:46:45 AM	Sent	192.168.1.4	146.20.47.186	TCP	1420	51288 > 443 [ACK] Seq=712 Ack=2796 Win=17152 Len=1380 [TCP segment of a reassembled PDU]
14	12:46:45 AM	Received	146.20.47.186	192.168.1.4	TCP	306	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
15	12:46:45 AM	Sent	192.168.1.4	146.20.47.186	TCP	366	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
16	12:46:45 AM	Sent	192.168.1.4	146.20.47.186	TLSv1	40	51288 > 443 [ACK] Seq=386 Ack=2530 Win=17408 Len=0
17	12:46:45 AM	Received	146.20.47.186	192.168.1.4	TCP	1189	Certificate, Server Hello Done
18	12:46:46 AM	Received	146.20.47.186	192.168.1.4	TCP	1420	Server Hello
19	12:46:46 AM	Received	146.20.47.186	192.168.1.4	TCP	40	443 > 51288 [ACK] Seq=1 Ack=386 Win=30720 Len=0
20	12:46:46 AM	Sent	192.168.1.4	146.20.47.186	TCP	425	Client Hello
21	12:46:46 AM	Sent	192.168.1.4	146.20.47.186	TCP	40	51288 > 443 [ACK] Seq=1 Ack=1 Win=17408 Len=0
22	12:46:47 AM	Received	146.20.47.186	192.168.1.4	TLSv1	52	443 > 51288 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK_PERM=1 WS=1024
23	12:46:47 AM	Sent	192.168.1.4	146.20.47.186	TCP	52	51288 > 443 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1

The Total Length column illustrated in Table 9 displays the total data size of the packets and represents the total amount of data used by the packet. The total length is comprised of TCP header data, IP header data and a TCP payload. The TCP payload is comprised of the TCP segment data but can also contain the Secure Socket Layer (SSL) Transport Layer Security (TLS) “Hello” data. The TLS handshake protocol is responsible for the authentication and establishment of the secret keys and key exchange that enable clients and servers to communicate and resume secure sessions (IBM, 2018).

Figure 18 illustrates the packet information of the individual packet sequence number 6 as displayed in Table 9 showing the TCP header, ID header and TCP payload and segment sizes.

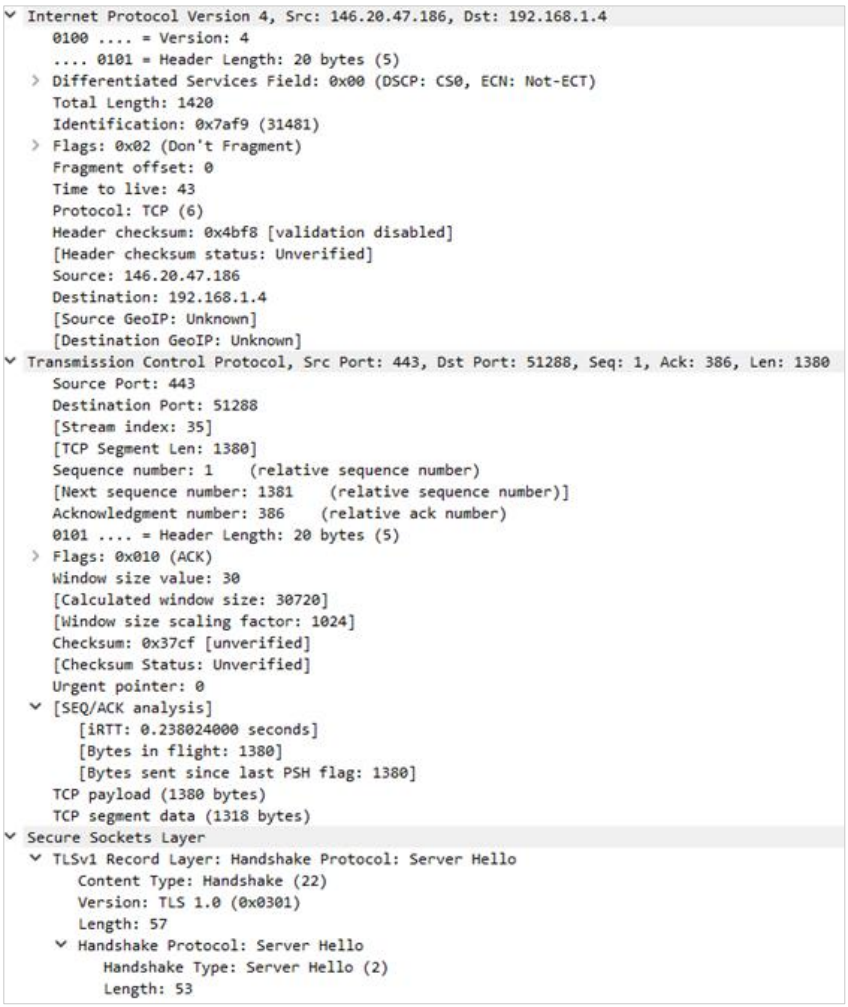


Figure 17: Native App - Packet dissection packet 6 from Table 9

Figure 17 illustrates the details of a sent data packet displaying the IP header size of the data packet was 20 bytes, the TCP header size was 20 bytes and the TCP payload was 1 380 bytes. The total length or data size of the packet was 1 420 bytes (Please see Appendix G for an example of the data inputs and data in JSON format that was sent to the server by the native app). It was observed that the TCP header and IP header sizes were 20 bytes across all the data packets and that the TCP payload sizes varied. The total length of the packet or total amount of data used can be calculated with the following formula:

(TCP Header + IP Header) + TCP Payload = Total Length

5.2. Web App Data Usage Analysis

Figure 18 illustrates the amount of data sent, data received and the total amount of data used when registering new cases using the Web app.

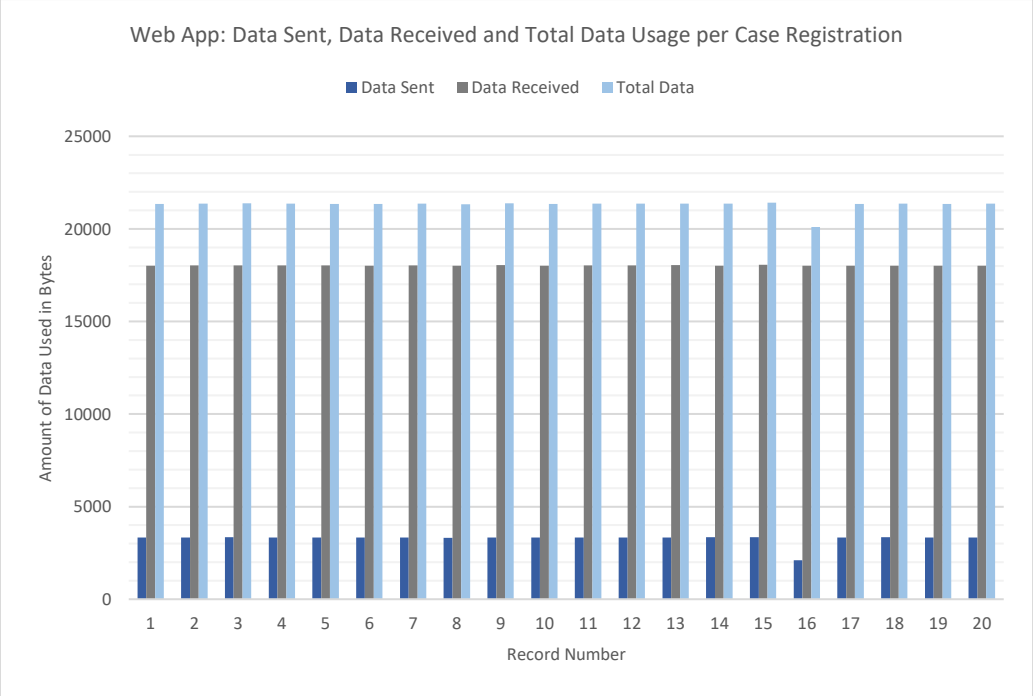


Figure 18: Web App - Data sent, data received and total amount of data used.

Figure 19 illustrates that the mean amount of data received by the Web app was greater than the amount of data sent and made up most of the total amount of data used by the Web app during the register a new case process.

The mean amount of data sent by the Web app was 3 275,25 bytes and the mean amount of data received by the Web app was 18 024 bytes.

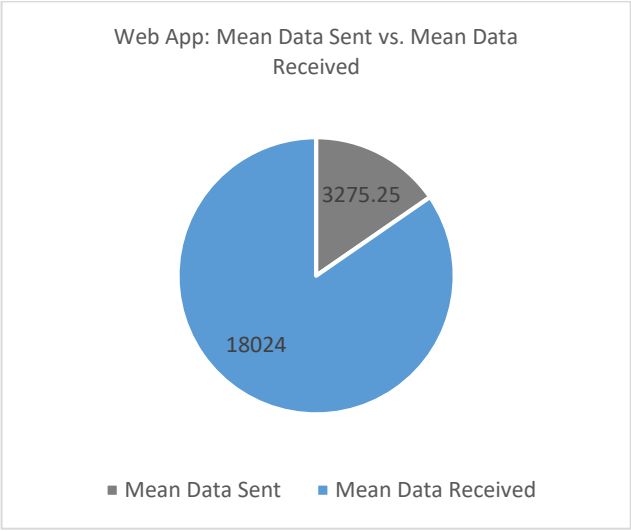


Figure 19: Web App: Data sent vs. Data received.

Figure 20 illustrates the mean amount of overhead data versus the mean amount of payload data transferred by the Web app. Overhead data was 4,49% of the total amount of data used by the Web app.

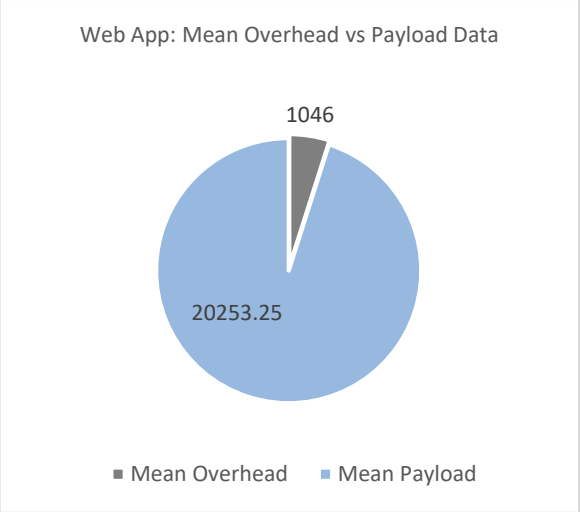


Figure 20: Web App: Overhead Data sent vs. Payload Data

Table 10 illustrates the data statistics produced by the Web app when registering a new case.

Table 10: *Web App Data Statistics in Bytes: Data used by the Web app when registering a new case*

	Data Sent	**Data Received**	**Total Data**	**Overhead Data**	**Payload**
Mean	3 275,25	18 024	21 299,25	1 046	20 253,25
Standard Deviation	277,67	13,04	280,4	0	280,4
Median	3 336	18 019	21 360,5	1046	20 314,5

The Web app executed processes using HTTP requests and responses and employed HTTP POST methods to send data to the server. The HTTP POST method transfers input data by storing the data in the body of the HTTP request (W3Schools, 2018). The HTTP request data sizes were comprised of the request headers and POST data. The HTTP response returned by the Web app server was comprised of the response headers and the response

body data. The Web app's response body was comprised of the entire Web page's HTML mark-up. Figure 21 illustrates an overview of the data communication sequence from record 4 of the Web app's experiment.

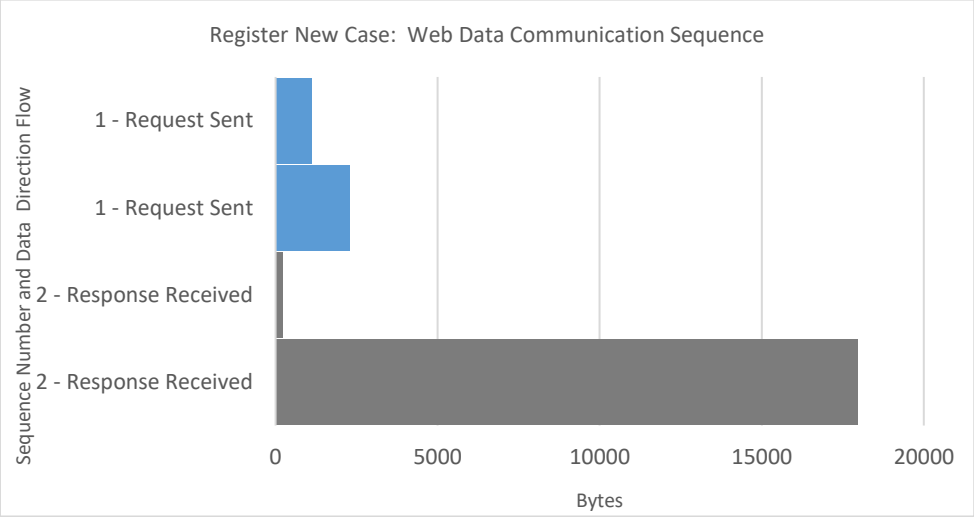


Figure 21: Web App – Data communication sequence of the "Register a New Case" process.

Table 11 provides an overview of the HTTP request and response data from record 4 of the Web app experiment's HTTP request and response information.

Table 11: *Web App Experiment - Record 4. Web App HTTP Request and Response Information*

Sequence	Data Direction	Total Length (Bytes)	Data Information
1	Request Sent	565	**Request Headers:** POST http://localhost:60003/Admin/AddNewCase.aspx HTTP/1.1 Host: localhost:60003 Proxy-Connection: keep-alive Content-Length: 1854 Cache-Control: max-age=0 Origin: http://localhost:60003 Upgrade-Insecure-Requests: 1 Content-Type: application/x-www-form-urlencoded User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/ 70.0.3538.77 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 Referer: http://localhost:60003/Admin/AddNewCase.aspx Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9 Cookie: ASP.NET_SessionId=qj2stwkqyyciiwzjbkm5ji2x
	Request Sent	2280	**Content-Data:** ctl00_ScriptManager_TSM: __EVENTTARGET: __EVENTARGUMENT: __VIEWSTATE: /wEPDwUKMjEwNTU3MTAxOA9kFgJmD2QWAgIDD2QWCAIDDw8WAh4EVGV4dAUHd29ya2VyMWRkAgg PFCsAAhQrAAIPFgIeF0VuYWJsZUFqYXhTa2luUmVuZGVyaW5naGQQFgJmAgEWAhQrAAJkEBYDZgIBAgIWAx QrAAJkZBQrAAJkZBQrAAJkZA8WA2ZmZhYBBXRUZWxlcmlrLldlYi5VSS5SYWRQYW5lbEl0ZW0sIFRlbGVyaWsu V2ViLlVJLCBWZXJzaW9uPTIwMTIuMS40MTEuNDAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGljS2V5VG9rZW4

			9MTIxZmFlNzgxNjViYTNkNBQrAAIPFgIeCEV4cGFuZGVkaGQQFgFmFgEUKwACZGQPFgFmFgEFdFRlbGVyaWsuV2ViLlVJLlJhZFBhbmVsSXRlbSwgVGVsZXJpay5XZWIuVUksIFZlcnNpb249MjxMi4xLjQxMS40MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj0xMjFmYWU3ODE2NWJhM2Q0DxYCZmYWAQV0VGVsZXJpay5XZWIuVUkuUmFkUGFuZWxJdGVtLCBUZWxlcmlrLldlYi5VSSwgVmVyc2lvbj0yMDEyLjEuNDExLjQwLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPTEyMWZhZTc4MTY1YmEzZDRkFgICAQ8PFgIfAmhkZAIKDw8WAh8BaGRkAg0PDxYCHgdWaXNpYmxlaGRkGAEFHl9fQ29udHJvbHNSZXF1aXJlUG9zdEJhY2tLZXlfXxYBBRJjdGwwMCRSYWRQYW5lbEJhcjFhVrCZRqzUeMyTEtaBHZ+faVxumj52eN0AJqz73ifdfA== __VIEWSTATEGENERATOR:480CC50 __EVENTVALIDATION:/wEdAAtcgD9FrgqPY+lwo/Wcju0CYFcQmUEeeOvwSdB9SuJnzsQc5SiMnxOW9c8OikLoIExT1ImyhbPhR2qtjAiwhQgloDiuygTsKXPocVIGy6fO+mgRFWJiCP2Fos6SPq3mRlVQ858bpNBkPKtZAf485eAtYQn2PeJDWiEXvyZU5cSaoENMhGMhp4xxjxDm8R7dxHmxavMUpzxnYxXh5rJZ9H8kVtW0xc9VMwvRz8qvLyKqxWir/gOgG4M929Lxb2kjdALzajxDn4CiGTlv9iQCU6d ctl00_RadPanelBar1_ClientState: {"expandedItems":["0"],"logEntries":[],"selectedItems":[]} ctl00$ContentPlaceHolder1$txtWomanName: [PATIENT_NAME] ctl00$ContentPlaceHolder1$txtVillageName: Athlone ctl00$ContentPlaceHolder1$ddlHasChildren: 1 ctl00$ContentPlaceHolder1$ddlIsSick: 1 ctl00$ContentPlaceHolder1$txtBoys: 3 ctl00$ContentPlaceHolder1$txtGIrls: 1 ctl00$ContentPlaceHolder1$txtLMSD: 05/05/2017 ctl00$ContentPlaceHolder1$btnSaveWoman: Save Woman's Details
2	Response Received	223	**Response Headers:** HTTP/1.1 200 OK Cache-Control: private Content-Type: text/html; charset=utf-8 Server: Microsoft-IIS/10.0 X-AspNet-Version: 4.0.30319 X-Powered-By: ASP.NET Date: Fri, 09 Nov 2018 06:24:56 GMT Content-Length: 17975
	Response Received	17975	**Response Body:** <!DOCTYPE html> <html> <head id="ctl00_Head1"><title>Village Health</title> <link href="../Styles/Site.css" rel="stylesheet" type="text/css" /><script src="../Scripts/script.js" type="text/javascript"></script> ... </body></html>

Sequence 1 in Table 11 illustrates the data sent by the Web app to the server by the HTTP request. The HTTP request data sent contains the header data, session state data and the content-data. The content-data in the request displayed in Table 11 was 2 280 bytes and was comprised of the POST data containing the form input data, the session data and the event data. The response body received in the response displayed in Table 11 was 17 975 bytes and was comprised of the entire Web page's mark-up which was reloaded every time the Web page was submitted. The Total Length column illustrated in Table 11 displays the data sizes of the request and response data. The total amount of data used by the Web app when executing the register new case process was derived from the sum of the values from the Total Length column displayed in Table 11. The total amount of data used by the Web app to process record 4 of the Web app's experiments was 21 593 bytes.

Figure 22 and Table 12 provides a detailed breakdown of the data communication sequence generated by record 4 of the Web app's experiment illustrating the HTTP application layer protocol communications and the TCP transport layer protocol communications between the client and the server.

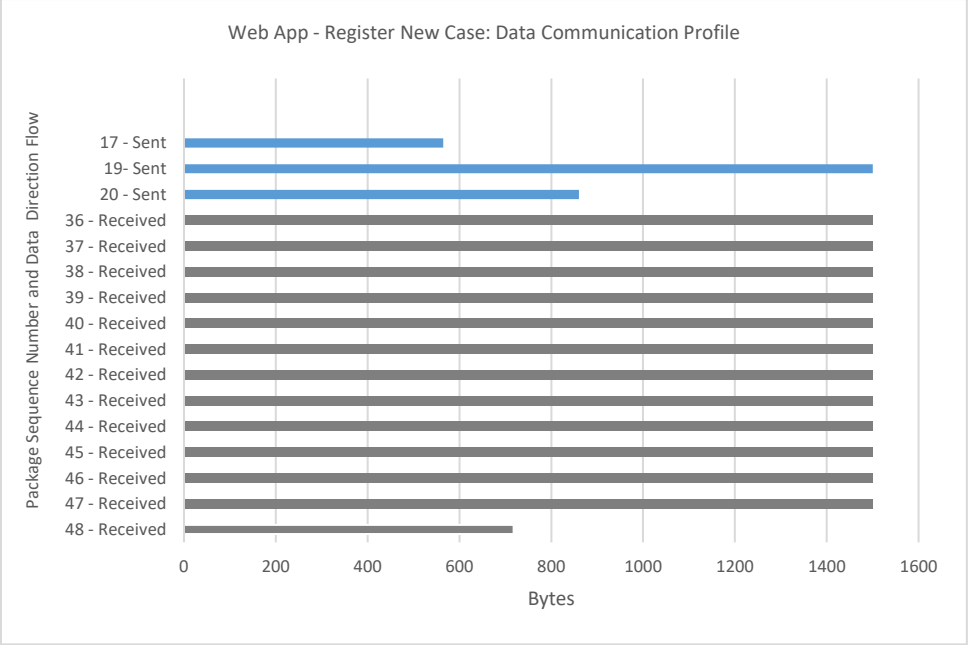


Figure 22: Web App – Detailed data communication sequence of "Register a New Case" process.

Table 12 provides information about the individual data packets from record 4 of the Web app experiment's data communication sequence.

Table 12: *Web App Experiment - Record 4. Web App Data Packet Communication Sequence Information*

Packet Sequence	Time	Source	Data Direction	Destination	Protocol	Total Length	Info
17	09:44:40 PM	127.0.0.1	Sent	127.0.0.2	TCP	565	50482 > 60003 [PSH, ACK] Seq=1 Ack=1 Win=2053 Len=525 [TCP segment of a reassembled PDU]
19	09:44:40 PM	127.0.0.1	Sent	127.0.0.2	TCP	1500	50482 > 60003 [ACK] Seq=526 Ack=1 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
20	09:44:40 PM	127.0.0.1	Sent	127.0.0.2	HTTP	860	POST /Admin/AddNewCase.aspx HTTP/1.1 (application/x-www-form-urlencoded)
36	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=1 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
37	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=1461 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
38	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=2921 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
39	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=4381 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
40	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=5841 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
41	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=7301 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
42	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=8761 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
43	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=10221 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
44	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=11681 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
45	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=13141 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
46	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=14601 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
47	09:44:40 PM	127.0.0.2	Received	127.0.0.1	TCP	1500	60003 > 50482 [ACK] Seq=16061 Ack=2806 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
48	09:44:40 PM	127.0.0.2	Received	127.0.0.1	HTTP	668	HTTP/1.1 200 OK (text/html)

Figure 23 illustrates the individual packet information of packet sequence number 20 as displayed in Table 12 and shows the reassembled TCP segment packet sequence numbers and payload sizes of the HTTP request.

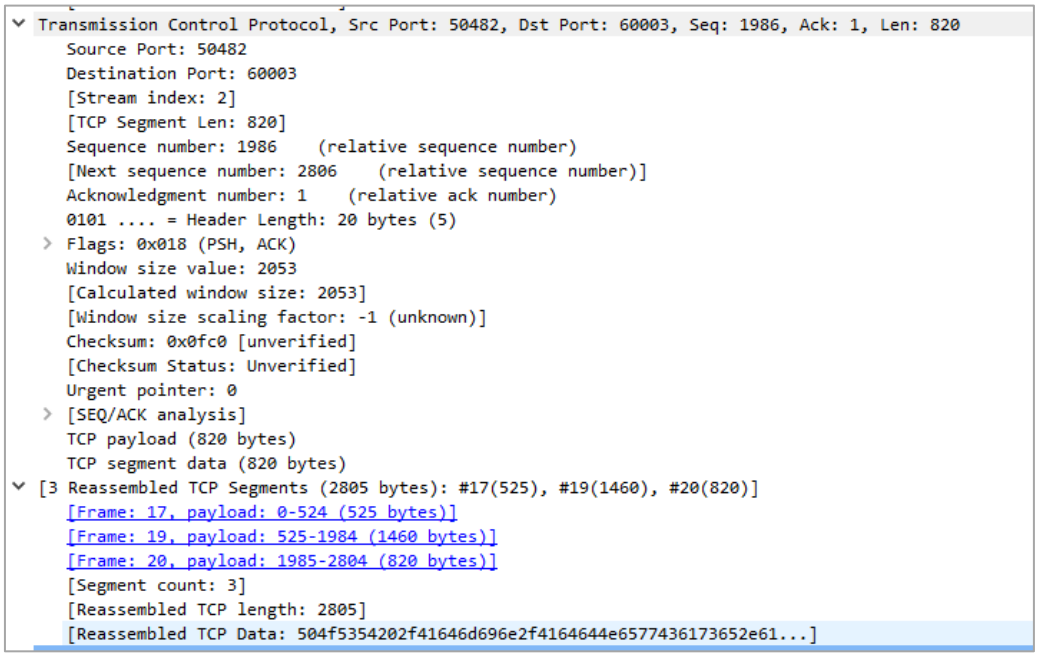


Figure 23: Packet dissection of packet 20 from Table 12.

The total HTTP request data size was the combined values of all the reassembled IP Headers and TCP segments' headers and payloads illustrated in Figure 23.

Figure 24 illustrates the packet information of the individual packet sequence number 48 as displayed in Table 12 and shows the reassembled TCP segment packet sequence numbers and payload sizes of the Web app's HTTP response.

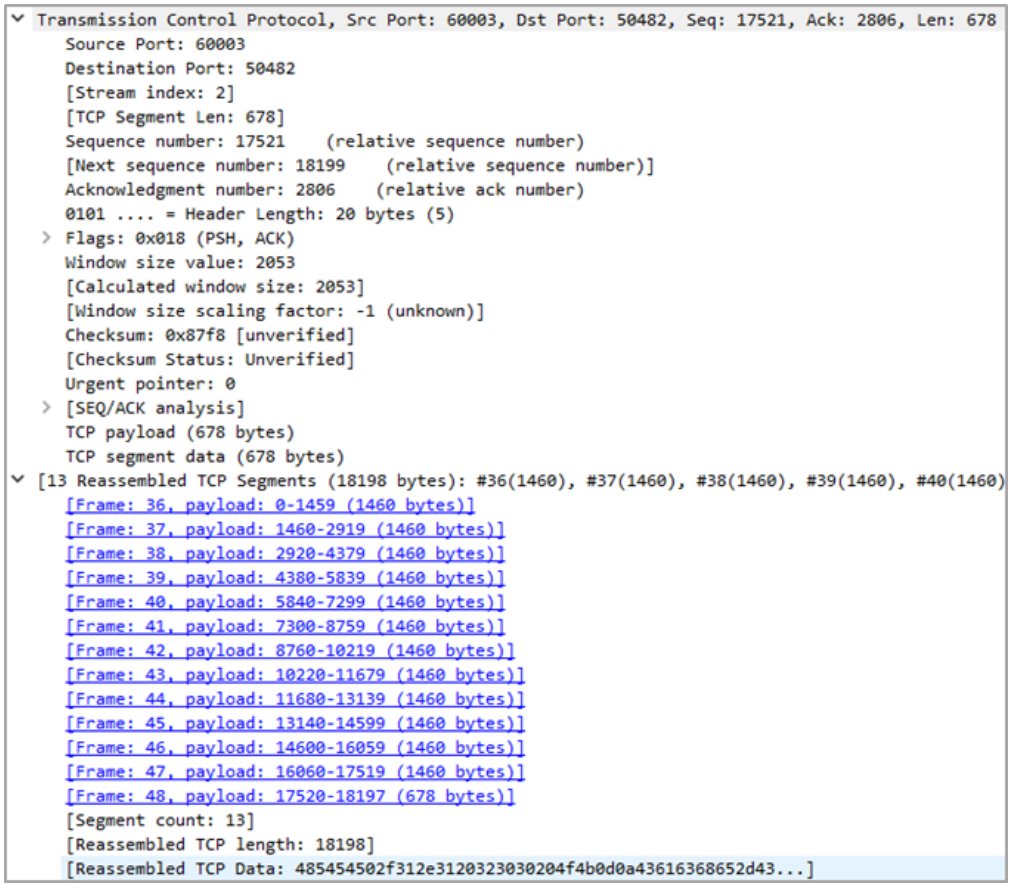


Figure 24: Packet dissection of packet 48 from Table 12.

The total HTTP response data was the combined values of the IP headers and the reassembled TCP segments' headers and payloads illustrated in Figure 24.

It was observed that the high total length data sizes generated by the HTTP responses were caused by the entire Web page's mark-up being outputted to the response stream on every Web page submission during the Web app experiments. Consequently, the Web app demonstrated the poorest performance and was the least data efficient of the three versions of the mobile app.

5.3. AJAX Web App Data Analysis

Figure 25 illustrates the amount of data sent, data received and the total amount of data used when registering new cases using the AJAX Web app.

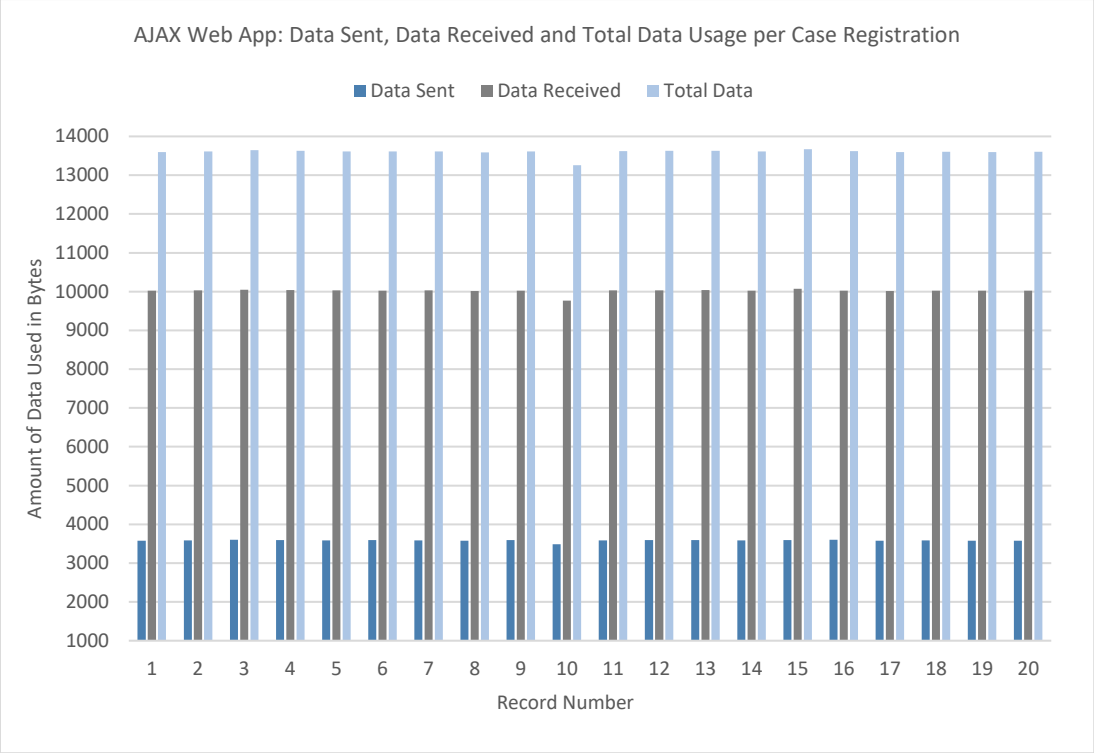


Figure 25: AJAX Web App - Data sent, data received and total amount of data used.

Figure 26 illustrates that the mean amount of data received by the AJAX Web app was greater than the amount of data sent and made up most of the total amount of data used by the AJAX Web app.

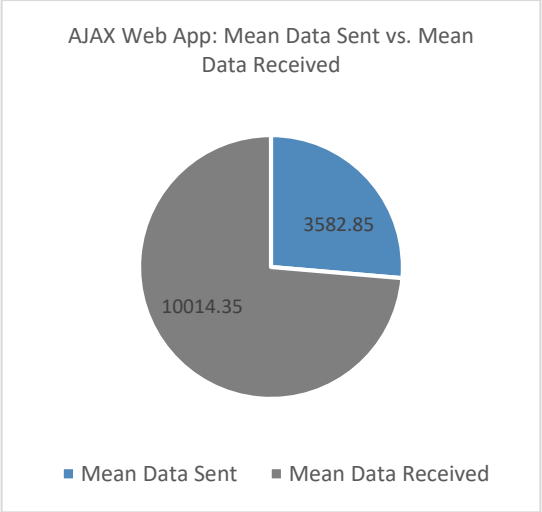


Figure 26: AJAX Web App: Data sent vs. data received.

Figure 27 illustrates the mean amount of overhead data versus the mean amount of payload data transferred by the AJAX Web app when performing standard procedures. Overhead data was 7,78% of the total amount of data used by the AJAX Web app.

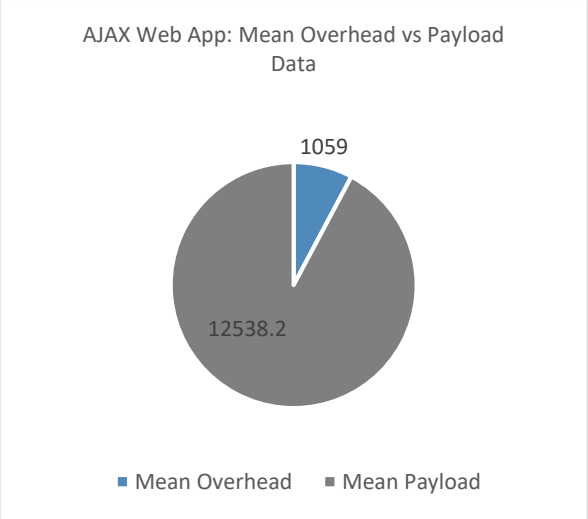


Figure 27: AJAX Web App: Mean Overhead vs Payload Data.

Table 13 illustrates the data statistics produced by the AJAX Web app when registering a new case.

Table 13: *AJAX Web App Data Statistics in Bytes: Data used by the Web app when registering a new case*

	Data Sent	Data Received	Total Data	Overhead Data	Payload
Mean	3 582,85	10 014,35	13 597,2	1 059	12 538,2
Standard Deviation	24,12	59,56	82,7	0,04	82,97
Median	3 587,5	10 021	13 612,5	7,77	12 553,5

The AJAX Web app executed HTTP requests and responses asynchronously and employed HTTP POST methods to send the data to the server. The HTTP request data sizes were comprised of the request headers and POST data. The HTTP response returned by the Web app's server was comprised of the response headers and the response body data.

However, in contrast to the Web app, the AJAX Web app response body only contained the mark-up required to update the affected Web page and its Web controls and not the entire Web page's mark-up and did not reload the entire Web page when submitting data.

Figure 28 illustrates a data communication sequence of record 4 of the AJAX Web app's experiment.

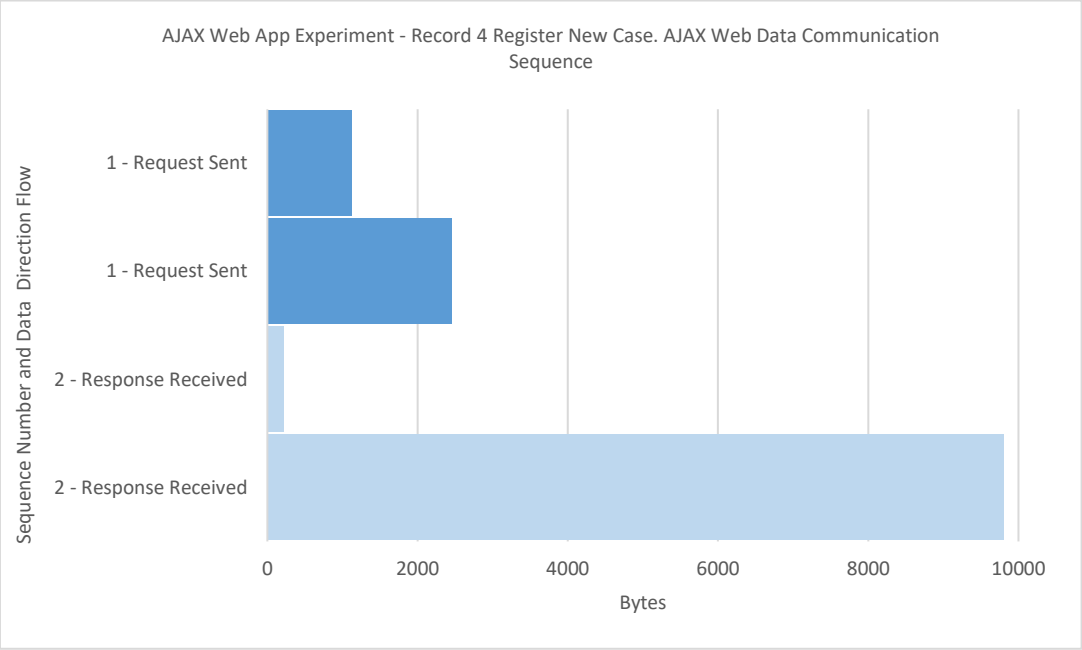


Figure 28: AJAX Web App – Data communication sequence of "Register a New Case" process.

Table 14 provides more information about the HTTP request and HTTP response data from record 4 of the AJAX Web app experiment's data communication sequence.

Table 14: *AJAX Web Experiment - Record 4: AJAX Web App Data Packet Communication Sequence Information*

Sequence	Data Direction	Total Length (Bytes)	Data Information
1	Request Sent	1121	**Request Headers:** POST /Admin/AddNewCase.aspx HTTP/1.1 Host: localhost:60001 Connection: keep-alive Content-Length: 2034 Origin: http://localhost:60001 X-Requested-With: XMLHttpRequest Cache-Control: no-cache X-MicrosoftAjax: Delta=true User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 Accept: */*

			Referer: http://localhost:60001/Admin/AddNewCase.aspx Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9 Cookie: ASP.NET_SessionId=kxrs044fsvndarqz01xaf23w
	Request Sent	2463	**Content-Data:** ctl00$ScriptManager: ctl00$ctl00$RadAjaxPanel1Panel\|ctl00$ContentPlaceHolder1$btnSaveWoman ctl00_ScriptManager_TSM: __EVENTTARGET: ctl00$ContentPlaceHolder1$btnSaveWoman __EVENTARGUMENT: __VIEWSTATE: /wEPDwUKMjEwNTU3MTAxOA9kFgJmD2QWAgIDD2QWCAIDDw8WAh4EVGV4dAUHd29ya2VyMWRkAggP FCsAAhQrAAIPFgIeF0VuYWJsZUFqYXhTa2luUmVuZGVyaW5naGQQFgJmAgEWAhQrAAJkEBYDZgIBAgIWAxQ rAAJkZBQrAAJkZBQrAAJkZA8WA2ZmZhYBBXRUZWxlcmlrLldlYi5VSS5SYWRQYW5lbEl0ZW0sIFRlbGVyaWsuV 2ViLlVJLCBWZXJzaW9uPTIwMTIuMS40MTEuNDAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGljS2V5VG9rZW49 MTIxZmFlNzgxNjViYTNkNBQrAAIPFgIeCEV4cGFuZGVkaGQQFgFmFgEUKwACZGQPFgFmFgEFdFRlbGVyaWsu V2ViLlVJLlJhZFBhbmVsSXRlbSwgVGVsZXJpay5XZWIuVUksIFZlcnNpb249MjAxMi4xLjQxMS40MCwgQ3VsdHV yZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj0xMjFmYWU3ODE2NWJhM2Q0DxYCZmYWAQV0VGVsZXJpay5X ZWIuVUkuUmFkUGFuZWxJdGVtLCBUZWxlcmlrLldlYi5VSSwgVmVyc2lvbj0yMDEyLjEuNDExLjQwLCBDdWx0d XJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPTEyMWZhZTc4MTY1YmEzZDRkFgICAQ8PFgIfAmhkZAIKDw8WA h8BaGRkAgwPZBYEAgEPDxYCHgdWaXNpYmxlaGRkAgMPZBYCAg8PDxYCHhFVc2VTdWJtaXRCZWhhdmlvcm hkZBgBBR5fX0NvbnRyb2xzUmVxdWlyZVBvc3RCYWNrS2V5X18WAQUSY3RsMDAkUmFkUGFuZWxCYXlxuzH 7TXj6wEZA/1YsjyI3f8w7gj3ZMUSdWQi2igJrz9U= __VIEWSTATEGENERATOR: 5480CC50 __EVENTVALIDATION: /wEdAAt8JIZHorywunYL4skIRdE0YFcQmUEeeOvwSdB9SuJnzsXQc5SiMnxOW9c8OikLoIExT1ImyhbPhR2qtjAi whQgloDiuygTsKXPocVIGy6fO+mgRFWJiCP2Fos6SPq3mRIVQ858bpNBkPKtZAf485eAtYQn2PeJDWiEXvyZU5 cSaoENMhGMhp4xxjxDm8R7dxHmxavMUpzxnYxXh5rJZ9H8kVtW0xc9VMwvRz8qvLyKq7cBFkUH/DRJQYro3 JOSKncB0pOn22mFLS9GUIUdfiVC ctl00_RadPanelBar1_ClientState: {"expandedItems":["0"],"logEntries":[],"selectedItems":[]} ctl00$ContentPlaceHolder1$txtWomanName: [PATIENT_NAME] ctl00$ContentPlaceHolder1$txtVillageName: Athlone ctl00$ContentPlaceHolder1$ddlHasChildren: 1 ctl00$ContentPlaceHolder1$ddlIsSick: 1 ctl00$ContentPlaceHolder1$txtBoys: 3 ctl00$ContentPlaceHolder1$txtGIrls: 1 ctl00$ContentPlaceHolder1$txtLMSD: 05/05/2017 __ASYNCPOST: true RadAJAXControlID: ctl00_RadAjaxPanel1
2	Response Received	223	**Response Headers:** HTTP/1.1 200 OK Cache-Control: private Content-Type: text/plain; charset=utf-8 Server: Microsoft-IIS/10.0 X-AspNet-Version: 4.0.30319 X-Powered-By: ASP.NET Date: Fri, 09 Nov 2018 09:02:59 GMT Content-Length: 9812
	Response Received	9812	**Response Body:** 1|#||4|5396|updatePanel|ctl00_ctl00_RadAjaxPanel1Panel| <!-- 2012.1.411.40 --> <table border="0" width="100%"><tr><td style="width:15px"></td> <td valign="center"> <B>SUCCESS</B> </td> </tr> <tr> <td colspan="2" style="height:15px"/> </tr> <tr> <td></td> <td>[PATIENT] has been registered. </td> </tr> <tr><td></td><td> <input type="hidden" name="ctl00$apErrorStack_AccordionExtender_ClientState" id="ctl00_apErrorStack_AccordionExtender_ClientState" value="-1" /> </td> </tr</table> <div id="btnClosecommuncationToolMessages" lass="btnClosecommuncationToolMessages"> <div id="btnClosecommuncationToolMessages2" class="btnClosecommuncationToolMessages2"> <input type="button" value="Close" style="cursor:pointer;" onclick="closeControl('ctl00_pnlCommunicationMessages' <ContentTemplate><table border="0"> <tr> <td colspan="2" class="columnHeaders">Add New Woman</td> </tr> <tr> <td style="width:180px" valign="top" class="columnHeaders">Woman's Name :</td> <td><input name="ctl00$ContentPlaceHolder1$txtWomanName" type="text" value="Faatimah Karriem" id="ctl00_ContentPlaceHolder1_txtWomanName" style="width:300px" /></td> </tr> <tr> <td valign="top" class="columnHeaders">Village Name :</td> <td> <input name="ctl00$ContentPlaceHolder1$txtVillageName" type="text" value="Athlone" id="ctl00_ContentPlaceHolder1_txtVillageName" style="width:300px;" /> </td></tr> <tr> <td valign="top" class="columnHeaders">Has Children : </td> <td><select name="ctl00$ContentPlaceHolder1$ddlHasChildren" id="ctl00_ContentPlaceHolder1_ddlHasChildren"> <option selected="selected" value="1">Yes</option><option value="0">No</option></select> </td> </tr> <tr> <td valign="top" class="columnHeaders">Is Sick Today : </td> <td> <select name="ctl00$ContentPlaceHolder1$ddlIsSick" id="ctl00_ContentPlaceHolder1_ddlIsSick"><option selected="selected" value="1">Yes</option> <option value="0">No</option></select>

</td> </tr><tr> <td valign="top" class="columnHeaders">Number of Boys :</td> <td> <input name="ctl00$ContentPlaceHolder1$txtBoys" type="text" value="3" id="ctl00_ContentPlaceHolder1_txtBoys" style="width:300px;" /> </td> </tr> <tr> <td valign="top" class="columnHeaders">Number Of Girls :</td> <td> <input name="ctl00$ContentPlaceHolder1$txtGIrls" type="text" value="1" id="ctl00_ContentPlaceHolder1_txtGIrls" style="width:300px;" /> </td> </tr> <tr><td valign="top" class="columnHeaders">Last Menstrual Date :</td><td> <input name="ctl00$ContentPlaceHolder1$txtLMSD" type="text" value="05/05/2017" id="ctl00_ContentPlaceHolder1_txtLMSD" style="width:300px;" /> </td> </tr> <tr> <td colspan="2" class="columnHeaders" style="height:25px"> <input type="button" name="ctl00$ContentPlaceHolder1$btnSaveWoman" value="Save Woman's Details" onclick="javascript:__doPostBack('ctl00$ContentPlaceHolder1$btnSaveWoman','')" id="ctl00_ContentPlaceHolder1_btnSaveWoman" /> </td> </tr></table> </ContentTemplate> |0|hiddenField|__EVENTTARGET||0|hiddenField| .... });

Table 14 illustrates that the AJAX Web app had more attributes in its request content-data than the Web app. These attributes included:

- __ASYNCPOST: true
- RadAJAXControlID: ctl00_RadAjaxPanel1

Figure 29 and Table 15 displays a detailed breakdown of the data communication sequence generated by record 4 of the AJAX Web app's experiment illustrating the HTTP application layer protocol communications and the TCP transport layer protocol communications between the client and the server.

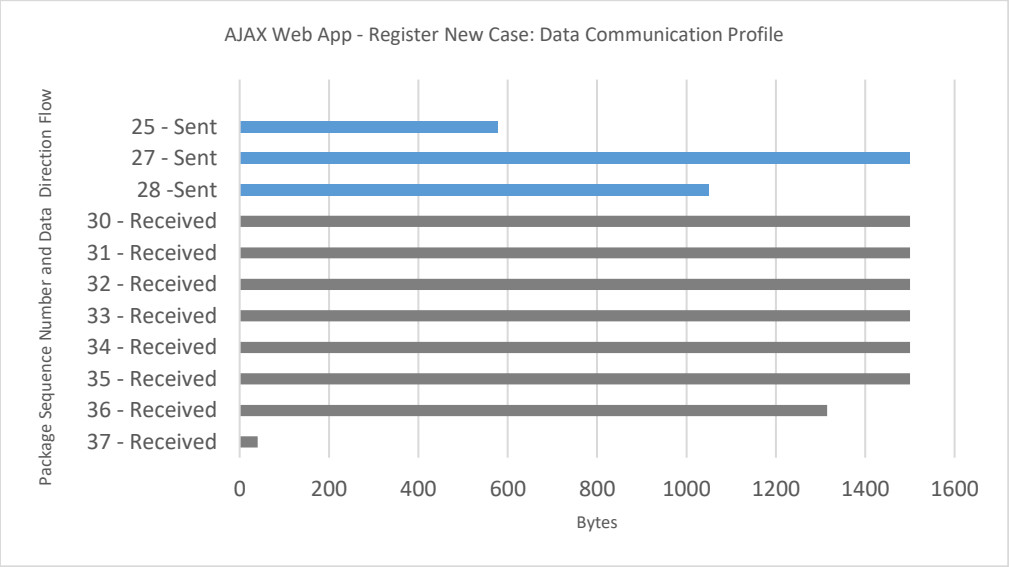


Figure 29: AJAX Web App – Detailed data communication sequence of the "Register a New Case" process.

Table 15 provides information about the individual data packets from record 4 of the AJAX Web app experiment's data communication sequence.

Table 15: *AJAX Web App Experiment - Record 4. Web App Data Packet Communication Sequence Information*

Packet Sequence Number	Time	Source	Data Direction	Destination	Protocol	Length	Total Length	Packet Information
25	45:57.7	127.0.0.1	Sent	127.0.0.1	TCP	578	578	59325 > 60001 [PSH, ACK] Seq=1 Ack=1 Win=2053 Len=538 [TCP segment of a reassembled PDU]
27	45:57.7	127.0.0.1	Sent	127.0.0.1	TCP	1500	1500	59325 > 60001 [ACK] Seq=539 Ack=1 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
28	45:57.7	127.0.0.1	Sent	127.0.0.1	HTTP	1050	1050	POST /Admin/AddNewCase.aspx HTTP/1.1 (application/x-www-form-urlencoded)
30	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	1500	1500	60001 > 59325 [ACK] Seq=1 Ack=3009 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
31	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	1500	1500	60001 > 59325 [ACK] Seq=1461 Ack=3009 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
32	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	1500	1500	60001 > 59325 [ACK] Seq=2921 Ack=3009 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
33	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	1500	1500	60001 > 59325 [ACK] Seq=4381 Ack=3009 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
34	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	1500	1500	60001 > 59325 [ACK] Seq=5841 Ack=3009 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
35	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	1500	1500	60001 > 59325 [ACK] Seq=7301 Ack=3009 Win=2053 Len=1460 [TCP segment of a reassembled PDU]
36	45:57.7	127.0.0.1	Received	127.0.0.1	HTTP	1315	1315	HTTP/1.1 200 OK (text/plain)
37	45:57.7	127.0.0.1	Received	127.0.0.1	TCP	40	40	59325 > 60001 [ACK] Seq=3009 Ack=10036 Win=2053 Len=0

Figure 30 illustrates the individual packet information of packet sequence number 28 as displayed in Table 15 and shows the reassembled TCP segment packet sequence numbers and payload sizes of the AJAX Web app's HTTP request.

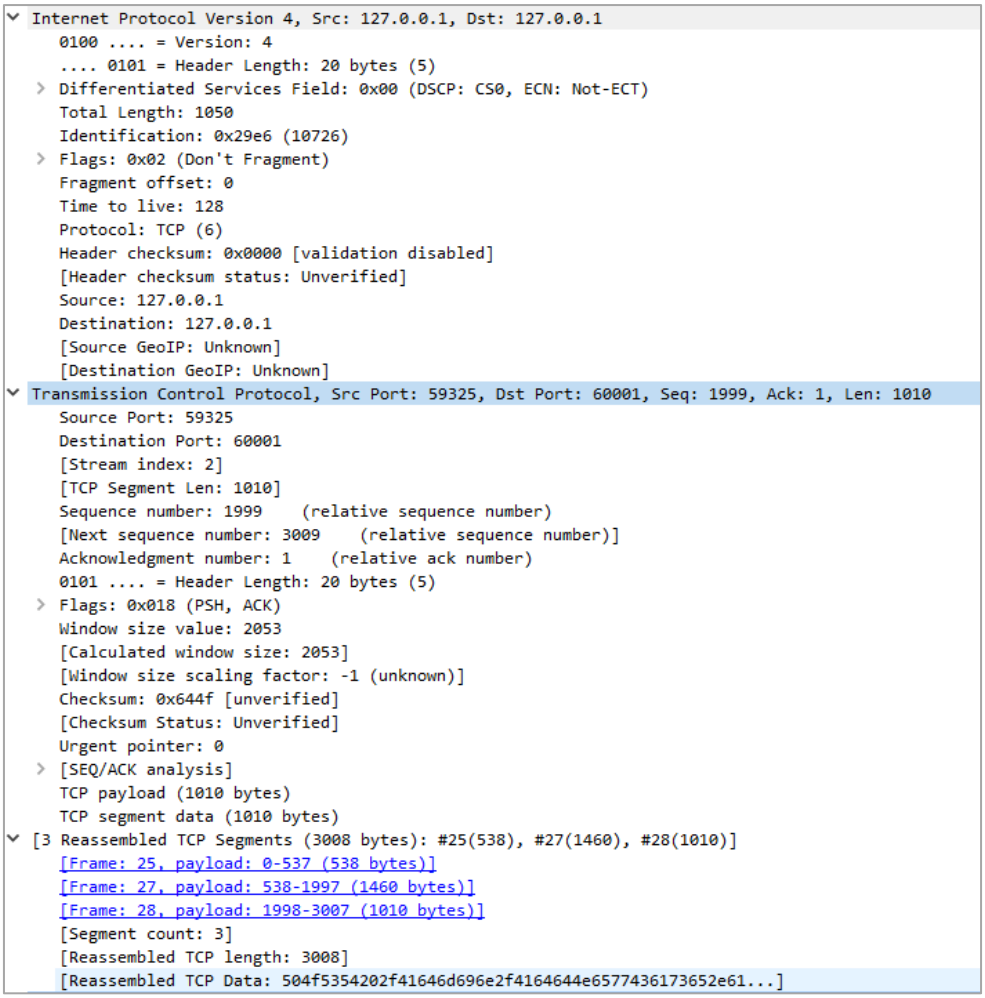


Figure 30: AJAX Web - Packet dissection of packet 28 from Table 15.

Figure 31 illustrates the individual packet information of packet sequence number 36 as displayed in Table 15 and shows the reassembled TCP segment packet sequence numbers and payload sizes of the AJAX Web app's HTTP response.

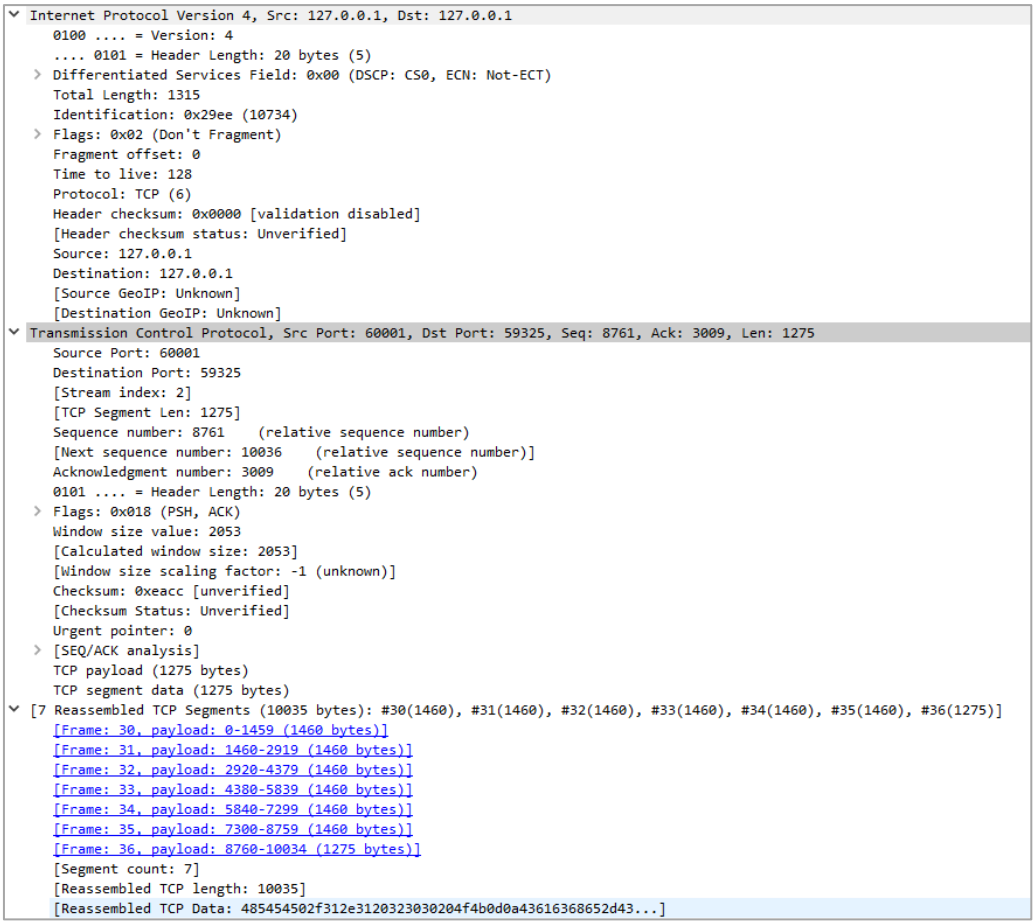


Figure 31: Packet dissection of packet 36 from Table 15.

5.4. Native, Web and AJAX Web - Total Amount of Data Used Analysis

Figure 32 illustrates the total amount of data used in bytes recorded while registering a new case with all three versions of the mobile app.

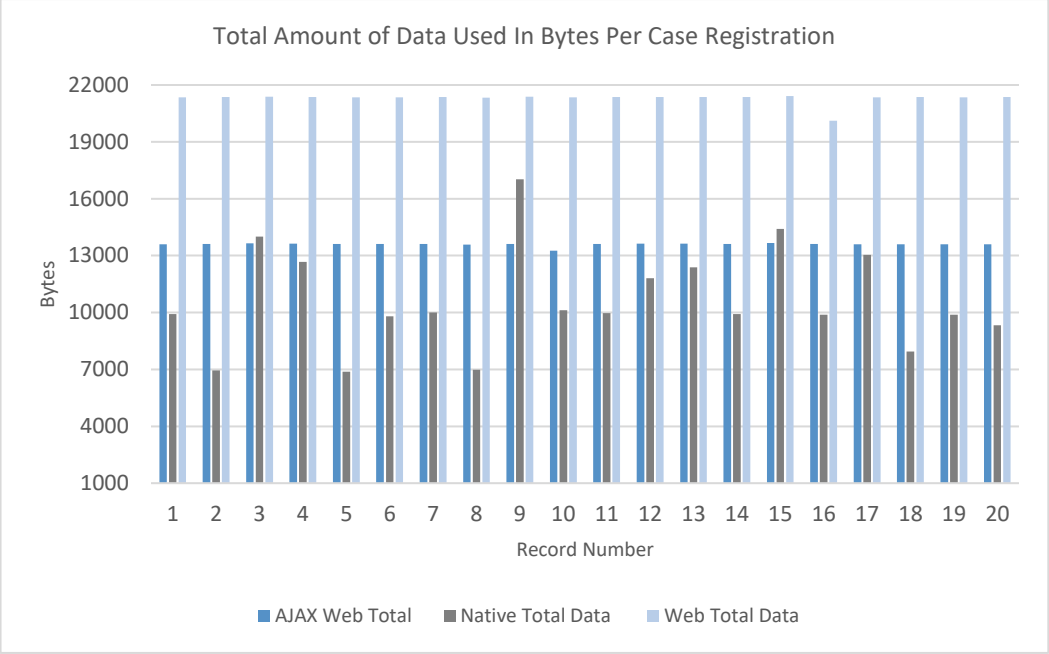


Figure 32: Native, Web and AJAX Web – Total Amount of Bytes per Case Registration

Table 16 illustrates the differences in the mean amounts of data sent and data received by the different versions of the mHealth app when registering a new case.

Table 16: *Data Usage Comparison: Mean amounts of data sent and data received.*

Mobile App Architecture	Mean Data Sent (Bytes)	Mean Data Received (Bytes)
Native App	7 392,3	3 254,35
AJAX Web App	3 582,85	10 014,35
Web App	3 275,25	18 024

The native app sent the greatest amount of data and the Web app sent the least amount of data. There were instances demonstrated where the native app used more data that the AJAX Web app. The larger data sizes sent by the native app can be attributed to the additional JSON text generated by transforming the input data into JSON format. The Web app received the greatest of amount of data and the native app received the least amount of data. The AJAX Web app sent more data than the Web app but less than the native app and received more data the native but less data than the Web app.

Table 17 displays the mean values of the total amount of data used per case registration for the native app, Web app and AJAX Web app.

Table 17: *Comparison of native and Web data statistics – Amount of Data Used Data Statistics by Case Registration*

	Native App	AJAX Web App	Standard Web App
Mean (Bytes)	10 646,65	13 597,2	21 299,25

The mean of the total amount of data used by the AJAX Web app was 2 950,55 bytes greater than the native app's. The mean of the total amount of data used by the Web app was 21 299,25 bytes and was 7 702,75 bytes greater than the AJAX Web app's and 10 652,6 bytes greater than the native app's. The data statistics illustrated in Table 17 and Table 15 suggests that the native app was more data efficient than the Web app and the AJAX Web app. It also illustrates that the AJAX Web app was more data efficient than the Web app.

However, further statistical analyses were required to determine if there was a significant difference in the total amount of data used by the different versions of the mobile app. Normality tests were conducted on the native app, the Web app and the AJAX Web app's recorded experiment data to determine if their experiment data followed normal distributions.

Table 18 illustrates the normality tests results for the native app, the AJAX Web app and the Web app (Please see Appendix J, K and L for detailed test results).

Table 18: *Native, AJAX Web App and Web App Normality Test Results: Amount of Data Used per Case Registration*

	Native App	AJAX Web App	Web App
p-value	0,14	< 0,0001	< 0,0001
Alpha	0,05	0,05	0,05
W	0,92	0,43	0,28
Standard Deviation	2 661,87	82,9	280,4
Mean	10 646,65	13 3597,2	21 299,25
Normality Distribution	Normal	Non-normal	Non-normal

5.5. Native, AJAX Web and Web App Nonparametric Test Results

The normality test results for the three different versions of the mobile app reported that the experiment data of the native app followed a normal distribution, but the experiment data of the Web app and the AJAX Web app did not follow a normal distribution.

Consequently, nonparametric tests were conducted on the native app, the Web app and the AJAX Web app's experiment data to determine if there was a significant difference in the amount of data used by the different versions of the mobile app.

Table 19 presents the results of the nonparametric tests (Please see Appendix M, N and O detailed test results.).

Table 19: *Native App, AJAX Web App and Web APP - Nonparametric Test Results*

Test Type	1st Sample	2nd Sample	Result	Conclusion
Mann-Whitney U	Web App	AJAX Web App	< 0,0001	Significantly Different
Mann-Whitney U	Web App	Native App	< 0.0001	Significantly Different
Mann-Whitney U	Native	AJAX Web	< 0.0001	Significantly Different

The results of the nonparametric tests reported that there was a significant difference in the amount of data used by the three different versions of the mobile health app.

5.6. Summary

The experiments revealed that the native app transferred data between clients and servers differently to the mobile Web apps. The native app sent and received individual data packets with each data packet containing a TCP header, IP header and a TCP payload. The native app transformed input data into JSON format (Please see Appendix G for an example of the JSON formatted input data), before transferring it to the server. The Web app used HTTP methods to send and receive data using the HTTP application layer protocol and the TCP transport layer protocol to transport the individual TCP data packets. The AJAX Web app used asynchronous HTTP methods to send and receive data using the HTTP

application layer protocol and the TCP transport layer protocol to transport the individual TCP data packets.

The native app sent more data than the Web app and the AJAX Web app. The larger data sizes sent by the native app can be attributed to the additional JSON text generated by transforming the input data into JSON format. However, leveraging off the resources of the mobile device and having the native app installed locally on the mobile device meant that no additional mark-up or UI controls were required to be downloaded and rendered to view, update and display user interfaces on the native app. Consequently, the total amount of data received by the native app was lower than the Web app and the AJAX web, ultimately resulting in the native app using the least amount of data.

The Web apps used HTTP requests and POST methods, which contained request headers and content-data to send requests and received HTTP responses, which contained response headers and response bodies. The AJAX Web app's requests had more attributes in its content-data than the Web app's requests. The additional attributes in the AJAX Web requests indicated that the requests were asynchronous posts and included the following:

- __ASYNCPOST: true
- RadAJAXControlID: ctl00_RadAjaxPanel1

Consequently, the mean amount of the data sent by the AJAX Web app was greater than the mean amount of the data sent by the Web app.

It was observed that the mean amount of the data received by the AJAX Web app was less than the mean amount of the data received by Web app. This can be attributed to the fact that the Web app responses contained the entire Web page's mark-up in its response bodies whereas the AJAX Web app's responses only contained the necessary mark-up to update the AJAX Web app's Web page and its Web controls.

The statistical analysis test results demonstrated that the native app had the highest percentage of overhead data, however, it used the least amount of overall data and was the most data efficient. The AJAX Web app used more data than the native app but less

data than the Web app. The Web app demonstrated to have the lowest amount of overhead but used the greatest amount of data when used to perform standard functional processes. The nonparametric test results reported that there was a significant difference in the total amount of data used by the mobile app when implemented using different mobile app architectures.

Chapter 6: Conclusions

The purpose of this research was to determine if there would be a significant difference in the amount of data used by an mHealth app when implemented as a native mobile app, a mobile Web app or with a mobile app architecture that uses technologies such as AJAX, HTML5, CSS and Mobile Development Frameworks (MDF) and to answer the following question:

"Is there a significant difference in the amount of data used by an mHealth application that is used to manage maternal and neonatal health when implemented using a native message passing Android architecture versus using a mobile Web based architecture?"

It was hypothesised that implementing an mHealth app using different mobile architectures would demonstrate a significant difference in the amounts of data used by the different versions of the mHealth app.

Experiments and statistical analysis were conducted to determine if the differences in the amounts of data used by different versions of a mobile app that was developed using different mobile architectures were significantly different or not.

The statistical analysis of the experiment results reported that the experiment data distributions of the native app followed a normal distribution, but the AJAX Web app and the Web app's experiment data did not follow a normal distribution. Consequently, nonparametric tests were conducted with the experiment data combinations to determine if the amount of data used by the different versions of the mobile app were significantly different.

6.1. Mobile Application Architectures

The amount of data used by a mobile app can have a direct effect on a mobile app's performance, its user experience, its implementation and long-term operational costs and can ultimately contribute to its success or failure. The amount of data that a mobile app will use during standard operational processes and the mobile app architecture are factors

to consider when formulating a mobile strategy, establishing a mobile presence and investing in a mobile app.

A native version, a mobile Web version and a mobile AJAX Web version of the same mHealth app were developed and experimented with. The three mobile app architectures evaluated in these experiments sent, received and queried data on a central server and database while performing system processes such as submitting data. The three mobile app architectures required Internet connectivity to communicate with a central server, consequently consuming data and bandwidth.

6.2. Experiment Results

The experiment data generated and recorded during the experiments provided the data required to analyse and compare the amount of data used by the different versions of the mHealth app and to determine if there was a significant difference in the amounts of data used by each version of the mHealth app. The experiments demonstrated that the native mobile app sent and received data differently to the Web app and the AJAX Web app. The native app sent and received data packets that contained TCP headers, IP headers and TCP payloads. The Web app and AJAX Web app used the HTTP application layer protocol and sent HTTP requests that contained headers and content-data to send data and received HTTP responses that contained headers and response bodies and used TCP to transport the individual TCP data packets. The experiments demonstrated that the native app formatted data inputs with JSON and sent more data than the AJAX Web and Web app. The AJAX Web app's HTTP requests contained more attributes and resulted in greater content-data sizes than the Web app's requests.

The experiments also demonstrated that the Web app's response bodies contained the entire Web page mark-up which was reloaded on each Web form submission whereas the AJAX Web app's response bodies only contained the mark-up that was affected by the asynchronous AJAX Web form request. Consequently, the Web app's response body data sizes were greater than the AJAX Web app's response body data sizes and resulted in the Web app using a greater amount of data when performing processes.

The different methods used to send and receive data by the different versions of the mobile app demonstrated contrasting amounts of data used to perform mobile app processes. The Web app used the greatest amount of data when executing processes. The combined data size of reloading the entire Web page during HTTP requests and responses, which included content, request and response data generated by the standard Web app, was greater than the data sizes generated by the native app and the mobile AJAX Web app. The AJAX Web App used less data than the standard mobile Web app but used a greater amount of data than the native app when executing processes. The AJAX Web app performed HTTP requests asynchronously and dynamically reloaded and updated the Web page's content and not the entire Web page.

6.2.1. Native App

The native app demonstrated the highest data overhead, 10.7%, and sent more data than the AJAX Web app and the Web App. The large data sent sizes were attributed to the JSON formatting applied to the data inputs by the native app.

However, the native app transferred less overall data than the Web app and AJAX Web app, ultimately, demonstrating that it was the most data efficient during standard operation.

6.2.2. AJAX Web App

The AJAX Web app demonstrated the second highest data overhead, 7.78%, and the second-best performance from three versions of the mobile app. It used less overall data than the Web app but used more overall data than the native app.

6.2.3. Web App

The Web app demonstrated the lowest data overhead, 4.49%, but also demonstrated the poorest performance and used the greatest amount of overall data when executing the register a new case process.

The statistical analyses reported that there was a significant difference in the amount of data used by the app when implemented using a native mobile architecture, a standard

mobile Web architecture or a mobile Web architecture that uses Web technologies such as AJAX, HTML5 and CSS.

6.3. Future Work

Mobile phones and mobile development technologies are rapidly evolving. Frameworks such as Angular, NodeJS and Web technologies such as Single-Page Applications (SPA), Progressive Web Apps (PWA) and services workers, now make it possible to develop mobile applications using Web technologies that can easily be deployed, updated, maintained, executed on many different devices and have full access to the mobile device's hardware and APIs. Like native apps, PWAs are installable, can live on a user's mobile device's home screen, can engage users with push notifications and using service workers can operate in a disconnected state.

A possible research question is:

"Can a mobile application developed using technologies such Progressive Web Apps, Single-Page Applications and service workers be as data efficient as a native version of the same mobile application?"

Bibliography

100 People Foundation. (2012, August 20). *Frontline Health Workers*. Retrieved from YouTube: https://www.youtube.com/watch?v=5PkL7qeUL-o

Ali, E. E., Chew, L., & Yap, K. Y.-L. (2016). Evolution and current status of mHealth research: A systematic review. *BMJ Innovations*, 1-8.

AliveCor. (2017). *Strokes are preventable. Guard your heart.* Retrieved from AliveCor: https://www.alivecor.com/

Appelboom, G., Camacho, E., Abraham, M. E., Bruce, S. S., Dumont, E. L., Zacharia, B. E., . . . Connoly Jr, E. S. (2014). Smart wearable body sensors for patient self-assessment and monitoring. *Archives of Public Health*, 2-9.

Aranda-Jan, C. B., Mohutsiwa-Dibe, N., & Loukanova, S. (2014). Systematic review on what works, what does not work and why of implementation of mobile health (mHealth) projects in Africa. *BioMed Central*, pp. 1-15.

Betjeman, T. J., Soghoain, S. E., & Foran, M. P. (2013). mHealth in Sub-Saharan Africa. *International Journal of Telemedicine and Applications*, 1-7.

Blaschke, S., Bokenkamp, K., Cosmaciuc, R., Denby, M., Hailu, B., & Short, R. (June 2009). *Using mobile phones to improve child nutrition surveillance in Malawai.* UNICEF Malawi and UNICEF Innovations.

Braun, R., Catalani, C., Wimbush, J., & Israelski, D. (2013). Community Health Workers and Mobile Technology: A Systematic Review of the Literature. *Plus One*, Vol.8 pp.1-6.

Brigham, T. J. (2014). An Introduction to Gamification- Adding Game Elements for Engagement. *Medical Reference Services Quarterly*, 471-480.

Charland, A., & Leroux, B. (2011). Mobile Application Development: Web vs. Native. *ACMQUEUE*, 49-53.

CommCare, Inc. (2017, September 1). *Welcome To CommCare*. Retrieved from CommCare: https://www.commcarehq.org

Dalmasso, I., Datta, S. K., Bonnet, C., & Nikaein, N. (2013). Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tool. *IEEE*, 323-328.

DeRenzi, B., Borriello, G., Jackson, J., Kumar, V. S., Parikh, T. S., Virk, P., & Lesh, N. (2011). Mobile Phone Tools for field based health care workers in low income countries. *Mount Sinai Journal of Medicine*, Vol.78 p.406-418.

DeRenzi, B., Wacksman, J., Dell, N., Lee, S., Lesh, N., Borriello, G., & Ellner, A. (2016, June 3). Closing the Feedback Loop: A 12-month Evaluation of ASTA, a Self-Tracking Application for ASHAs. Ann Arbor, Michigan, USA.

Deterding, S. (2012). Gamification: Designing for Motivation. *Social Mediator*, 14-18.

Dimagi. (2017, January 30). *Health workers in Kaushambi, India talk about using mHealth apps* . Retrieved from YouTUbe: https://www.youtube.com/watch?v=oJOSYmPJ528

Dua, K. (2018, March 7). *A Guide to Moile App Development: Web vs. Native vs. Hybrid*. Retrieved from ClearbridgeMobile.com: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/

Eng, D., & Lee, J. (2013). The Promise and Peril of Mobile Health - Applications for Diabetes and Endocrinology. *Pediatric Diabetes 14*, pp. 231-238.

Fenston, J. (2011, December 14). *Telemedicine: Adventures in time and space* . Retrieved April 25, 2017, from KBIA: http://kbia.org/post/telemedicine-adventures-time-and-space#stream/0

Field, M. J. (1996). *Telemedicine: A Guide to Assessing Telecommunications in Health Care.* Washington D.C.: National Academies Press.

Fiordelli, M., Diviani, N., & Schulz, P. J. (2016). Mapping mHealth Research:A Decade of Evolution. *JOURNAL OF MEDICAL INTERNET RESEARCH*, Vol.15 pp 1-14.

Fitbit. (2017). *Surge: The #1 selling GPS watch in the US.* Retrieved from Fitbit: https://www.fitbit.com/surge

Flaming, A., Canty, M., Javetski, G., & Lesh, N. (2016). *The CommCare evidence base for frontline workers.* Dimagi.com.

Free, C., Phillips, G., Felix, L., Galli, L., Patel, V., & Edwards, P. (2010, October 6). The effectiveness of M-health technologies for improving health and health services: a systematic review protocol. *BioMed Central*, pp. 1-7.

Ghamari, M., Janko, B., Sherratt, S., Harwin, W., Piechockic, R., & Soltanpur, C. (2016). A Survey on Wireless Body Area Networks for eHealthcare Systems in Residential Environments. *Sensors*, 1-33.

Gupta, G. K. (2011, June). Ubiquitous Mobile Phones are becoming indispensible. *Bits & Bytes*, pp. Vol.2 pp 32-33.

Gurupur, V. P., & Wan, T. T. (2017). Challenges in implementing mHealth interventions: a technical perspective. *mHealth*, 1-45.

Hamari, J., & Koivisto, J. (2013). Social Motivations To Use Gamification: An Empirical Study Of Gamifying Exercise. *Association of Information Systems* , 1-13.

Handel, M. J. (2011). Mobile Health - Using Apps for Health and Wellness. *mHealth* , Vol.7 pp. 256-261.

Hatouri, K. (2011). "Gamification" from the perspective of service marketing. *ResearchGate*, 1-9.

Helios. (2017, August 25). *Why Mobile App Architecture is Vital for Mobile App Development?* Retrieved from Helios: https://blog.heliossolutions.in/mobile-app-architecture-vital-mobile-app-development/

Hogan, M. C., Foreman, K. J., Naghavi, M., Ahn, S. Y., Wang, M., Makela, S. M., . . . Murray, C. J. (2010). Maternal mortality for 181 countries, 1980-2008: A systematic analysis of progress towards Millennium Development Goal 5. *Lancet*, Vol. 375 p. 1609-1623.

Horda, T. (2017, January 1). *Websites and Mobile Apps*. Retrieved from rubygarage.org: https://rubygarage.org/blog/mobile-app-vs-mobile-website

IBM. (2018). *An overview of the SSL or TLS handshake*. Retrieved from IBM Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm

IEEE. (2000, October 9). IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. New York, New York, USA.

Istepanian, R. H., Jovanov, E., & Zhang, Y. T. (2004). Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*, Vol.8 pp. 405-413.

Istepanian, R. S., Jovanov, E., & Zhang, Y. T. (2004). Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE,*, Vol.8 pp 405-413.

Jobe, W. (2013). *NATIVE APPS VS. MOBILE WEB APPS.* Stockholm: iJIM.

Kahn, J. G., Yang, J. S., & Kahn, J. S. (2010). 'Mobile' Health Needs And Opportunities In Developing Countries. *Health Affairs*, Vol. 29 pp. 254-261.

Khan, A. H., Qadeer, M. A., Ansari, J. A., & Waheed, S. (2009). *4G as a Next Generation Wireless Network.* Aligarh: International Conference on Future Computer and Communication.

Khan, J. Y., Yuce, M. R., & Karami, F. (2008). Performance Evaluation of a Wireless Body Area Sensor Network for Remote Patient Monitoring. *IEEE*, 1266-1269.

Kumar, S., Nilsen, W., Abernethy, A., Atienza, A., Patrick, K., Pavel, M., . . . Swendeman, D. (2013). Mobile Health Technology Evaluation: The mHealth Evidence Workshop. *American Journal of Preventive Medicine*, pp. 228-236.

Kumar, S., Nilsen, W., Pavel, M., & Srivastava, M. (2013, January). Mobile Health: Revolutionizing Healthcare Through Trans-disciplinary Research. *IEEE Computer Society*, pp. 228-35.

Lee, C., Lee, K., & Lee, D. (2017). Mobile Healthcare Applications and Gamification for Sustained Health Maintenance. *MDPI*, 1-12.

Lee, J.-S., Su, Y.-W., & Shen, C.-C. (2007). A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee and Wi-Fi. *IEEE Industrial Electronics Society*, 46-51.

Lee, S., Kim, J.-N., & Chib, A. (2011). Midwives' Cell Phone Use and Health Knowledge in Rural Communities. *Journal of Health Communication*, 1006-1023.

Lemay, N. V., Sullivan, T., Jumbe, B., & Perry, C. P. (2012). Reaching Remote Health Workers in Malawi: Baseline Assessment of a Pilot mHealth Intervention. *Journal of Health Communication*, 105-117.

Lobelo, F., Kellie, H. M., Tejedor, S. C., Pratt, M., McConnel, M. V., Martin, S. S., & Welk, G. J. (2007). The Wild Wild West: A Framework to Integrate mHealth Software Applications and Wearables to Support Physical Activity Assessment, Counseling and Interventions for Cardiovascular Disease Risk Reduction. *Science Direct*, 584-594.

Lupton, D. (2014). Critical Perspectives on Digital Health Technologies. *Sociology Compass*, 1-17.

Mea, V. D. (2001). What is e-Health (2): The death of telemedicine? *Journal of Medical Internet Research* , 1-2.

Microsoft. (2015, 07 20). *Introduction to the C# Language and the .NET Framework*. Retrieved from Microsoft.com: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Microsoft. (2016). *Visual Studio Tools for Xamarin*. Retrieved from Microsoft.com: https://visualstudio.microsoft.com/xamarin/

Microsoft. (2017). *A flexible & easy-to-manage web server...* Retrieved from IIS.net: https://www.iis.net

Microsoft. (2017). *ASP.NET Web API*. Retrieved from Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx

Microsoft. (2017). *Get Building*. Retrieved from ASP.NET.com: https://www.asp.net/

Microsoft. (2018). *TLS Handshake Protocol*. Retrieved from www.microsoft.com: https://docs.microsoft.com/en-us/windows/desktop/secauthn/tls-handshake-protocol

Miller, A. S., Cafazzo, J. A., & Seto, E. (2016). A Game Plan: Gamification design principles in mHealth applications for chronic disease. *Health Informatics Journal*, Vol.22 pp. 184-193.

Mohamad, N. M., Salam, S., & Bakar, N. (2017). An Analysis of Gamification Elements in Online Learning to Enhance Learning Engagement. *International Conference of Computing and Informatics*, 452-460.

MommyMeds for Mothers. (2017). Retrieved from MommyMeds: http://mommymeds.com/mobile-apps/mommymeds

Netresec. (2017). *RawCap*. Retrieved from NetResec RawCap: http://www.netresec.com/?page=RawCap

Noordam, C. A., Kuepper, B. M., Stekelenburg, J., & Milen, A. (2011). Improvement of maternal health services through the use of mobile phones. *Tropical Medicine and International Health*, Vol. 16 p. 622-626.

Oh, H., Rizo, C., Enkin, M., Jedad, A., & Phil, D. (2005). *What Is eHealth : A Systematic Review of Published Definitions.* Toronto: Centre for Global eHealth Innovation.

Olzak, T. (2006, December 1). *Secure your Bluetooth wireless networks and protect your data*. Retrieved from TechRepublic: http://www.techrepublic.com/article/secure-your-bluetooth-wireless-networks-and-protect-your-data/6139987/

Orgerea, S. (2017, May 23). *How to Use Web Browser Developer Tools*. Retrieved from Lifewire: https://www.lifewire.com/web-browser-developer-tools-3988965

Pebble. (2016). *Pebble Fit & Smart*. Retrieved from Pebble: https://www.pebble.com/pebble-2-smartwatch-features

Perera, C. (2012). The Evolution of E-Health – Mobile Technology and mHealth. *Journal of Mobile Technology in Medicine*, 1-2.

Perez, B. M., De La Torre-Diez, I., & Lopez-Coronado, M. (2013). Mobile Health Applications for the Most Prevalent Conditions. *Journal of Medical Internet Research*, Vol.15 pp.1-19.

Pieljko, P. (2016, April 12). *16 mobile market statistics you should know in 2016*. Retrieved April 17, 2017, from DeviceAtlas.com: https://deviceatlas.com/blog/16-mobile-market-statistics-you-should-know-2016

Pisuwala, U. (2018). *Everything you need to know about mobile app architecture*. Retrieved from Peerbits: https://www.peerbits.com/blog/all-about-app-architecture-for-efficient-mobile-app-development.html

ProxyCap. (2017). *ProxyCap - Making your system proxy aware*. Retrieved from ProxyCap: http://www.proxycap.com/

Raboy, N. (2018, April 30). *3 Of The Best Frameworks For Mobile App Development In 2018*. Retrieved from www.thepolyglotdeveloper.com: https://www.thepolyglotdeveloper.com/2018/04/3-best-frameworks-mobile-app-development-2018/

Rashid, M. B., & Suganya, P. (2017). Gamification: An Initiative to Increase Engagement and Performance in Education. *International Journal of Advance Research, Ideas and Innovations in Technology*, Vol.3 p7-16.

Redda, Y. A. (2012, June). Cross Platform Mobile Applications Development. Norway.

Roine, R., Ohinmaa, A., & Hailey, D. (2001). Assessing Telemedicine: A systematic review of the literature. *CMAJ*, 765-771.

Rouse, M. (2018). *Microsoft SQL Server*. Retrieved from techtarget.com: https://searchsqlserver.techtarget.com/definition/SQL-Server

Rouse, M. (2018). *T-SQL* . Retrieved from techtarget.com: https://searchsqlserver.techtarget.com/definition/T-SQL

Salesforce. (2016, June). *Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options*. Retrieved from Salesforce.com: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Serrano, N., Hernantes, J., & Gallardo, G. (2013). Mobile Web Apps. *IEEE Software*, 22-27.

Shameli, A., Althoff, T., Saberi, A., & Leskovec, J. (2017). How Gamification Affects Physical Activity - Large-scale Analysis of Walking Challenges in a Mobile Application. *International World Wide Web Conference Committee*, 455-463.

Smyth, J. M. (2012, May 12). *Hesperian Health Guides Releases Open Copyright Apps*. Retrieved from Open Health News: http://www.openhealthnews.com/hotnews/hesperian-health-guides-releases-open-copyright-apps

Sprout. (2017). *Sprout Pregnancy*. Retrieved from Sprout: http://sprout-apps.com/sprout-pregnancy-iphone-app/

Statista. (2018). *Share of Android OS of global smartphone shipments from 1st quarter 2011 to 1st quarter 2018*. Retrieved from Statista - The Statistics Portal: https://www.statista.com/statistics/236027/global-smartphone-os-market-share-of-android/

Summerfield, J. (2018, July 27). *Mobile Website vs. Mobile App: Which is Best for Your Organization?* . Retrieved from www.hswsolutions.com: https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/

Tachakra, S., Wang, X. H., Istepanian, R. S., & Song, Y. H. (2003). Mobile e-Health: The Unwired Evolution of Telemedicine. *Mobile e-Health: The Unwired Evolution of Telemedicine*, Vol.9 pp. 1-11.

Tamara, P., Kraschnewski, J., Chuang, C., Poole, E., & Reddy, M. (2014). Information, Sharing and Support in Pregnancy: Addressing Needs for mHealth Design. *Computer-Supported Cooperative Work and Social Computing*.

Tamrat, T., & Kachnowski, S. (2012). Special Delivery: An analysis of mHealth in Maternal and Newborn Health Programs and their outcomes around the World. *Maternal and child health journal*, Vol. 16 p. 1092-1101.

Telerik. (2017). *Telerik* . Retrieved from Telerik Fiddler: https://www.telerik.com/fiddler

Terry, K. (2015, September 18). *Number of Health Apps Soars, but Use Does Not Always Follow*. Retrieved April 22, 2017, from MedScape: http://www.medscape.com/viewarticle/851226

Thiebes, S., Lins, S., & Basten, D. (2014). Gaming Information Systems - A synthesis of gamification mechanics and dynamics. *Association for Information Systems*, 1-18.

Tomlinson, M., Rotheram-Borus, M., Swartz, L., & Tsai, A. (2013, February 12). Scaling Up mHealth: Where Is the Evidence? *PLOS Medicine Volume 10*, pp. 1-5.

van Dijk, M. R., Oostingh, E. C., Koster, M. P., Willemsen, S. P., Laven, J. S., & Steegers-Theunissen, R. P. (2017). The use of the mHealth program Smarter Pregnancy in preconception care: rationale, study design and data collection of a randomised comtrolled trial. *BioMed Central*, 1-7.

Velthoven, M. H., Car, J., Zhang, Y., & Marusic, A. (2013). New ideas for mHealth data collection implementation in low–and middle–income countries. *Journal of Global Health*, Vol.3 p.1-3.

Viatom. (2017). *Viatom Checkme Pro*. Retrieved from Viatom: http://www.viatomtech.com/checkme-pro

W3 Schools. (2017). *CSS Tutorial*. Retrieved from W3 Schools: https://www.w3schools.com/css/default.asp

W3 Schools. (2017). *HTML Introduction*. Retrieved from w3schools.com: https://www.w3schools.com/html/html_intro.asp

W3 Schools. (2017, 2017). *JavaScript Tutorial*. Retrieved from W3 Schools: https://www.w3schools.com/js/default.asp

W3 Schools. (2017). *PNG (Portable Network Graphics)*. Retrieved from W3 Schools: https://www.w3.org/Graphics/PNG/

W3 Schools. (2017). *W3 Schools*. Retrieved from https://www.w3schools.com/js/js_json_intro.asp: https://www.w3schools.com/js/js_json_intro.asp

W3 Schools. (2018, August 5). *AJAX Introduction*. Retrieved from w3schools.com: https://www.w3schools.com/xml/ajax_intro.asp

W3Schools. (2018). *HTTP Request Methods*. Retrieved from W3Schools: https://www.w3schools.com/tags/ref_httpmethods.asp

Wasserman, A. I. (2010). Software Engineering Issues for Mobile Application Development. *ACM*, 397-400.

Wireshark . (2017). *Wireshark Homepage*. Retrieved from Wireshark: https://www.wireshark.org/

Withings. (2017). *Wireless Blood Pressure Monitor*. Retrieved from Withings Part of Nokia: https://www.withings.com/eu/en/products/blood-pressure-monitor

World Bank. (2017, June 12). *India Health Stats*. Retrieved from The World Bank: http://data.worldbank.org/topic/health?locations=IN

World Health Organisation. (2007, January). *Community health workers:What do we know about them? The state of the evidence on programmes, activities, costs and impact on health outcomes of using community health workers.* Retrieved from World Health Organisation: http://www.who.int/healthsystems/round9_7.pdf

World Health Organization. (2013). *Assisting community Health workers in India.* Geneva: World Health Organization.

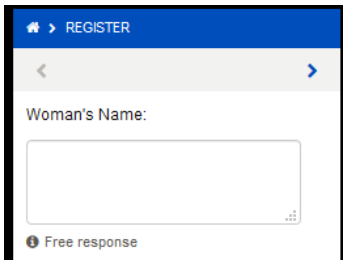
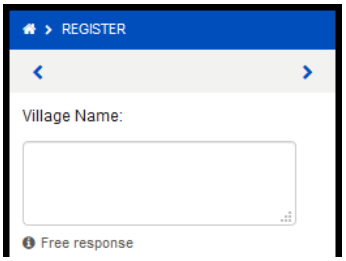
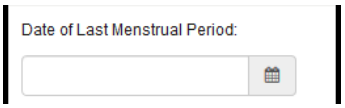
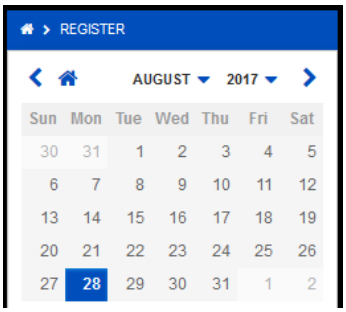
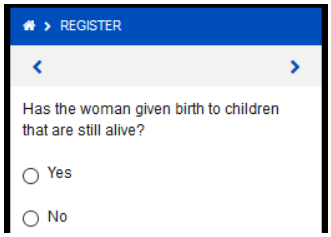
Yu, P., Ming, X. W., Hui, Y., & Guo, Q. X. (2006). The Challenges for the Adoption of mHealth. *IEEE*, 181-186.

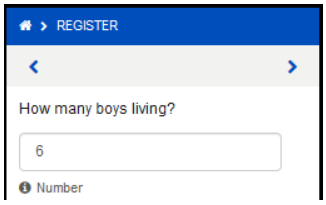
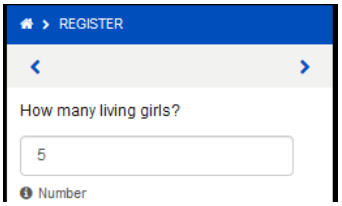
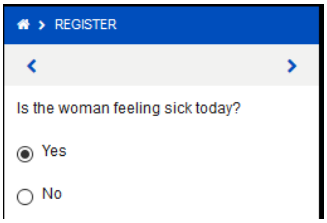
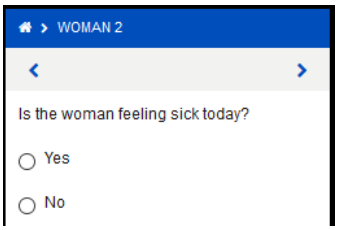
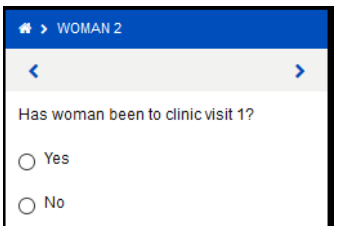
Zicherman, G., & Cunningham, C. (2011). *Gamification by Design.* Sebastopol: O'Reilly.

Appendix A – Native Village Health App: Examples of the Inputs, User Interfaces and Data Types

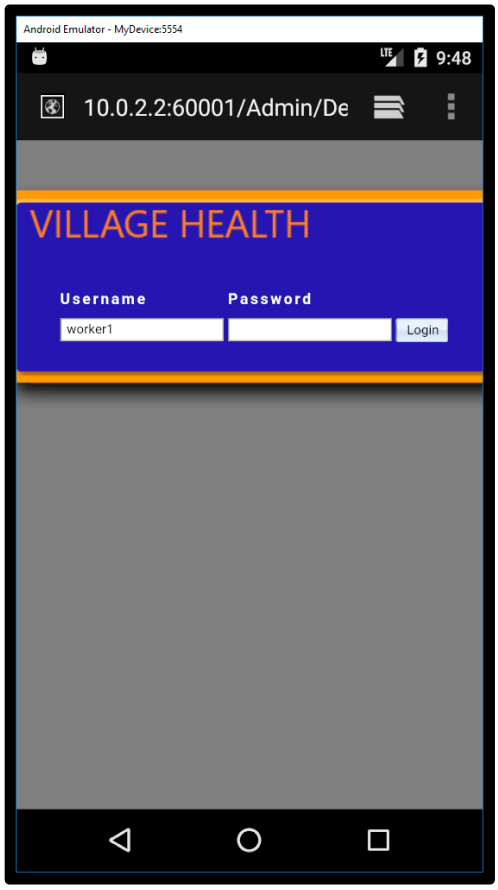
The following tables display the data input questions, data type definitions, business logic and the user screen interfaces for the Village Health Android solution:

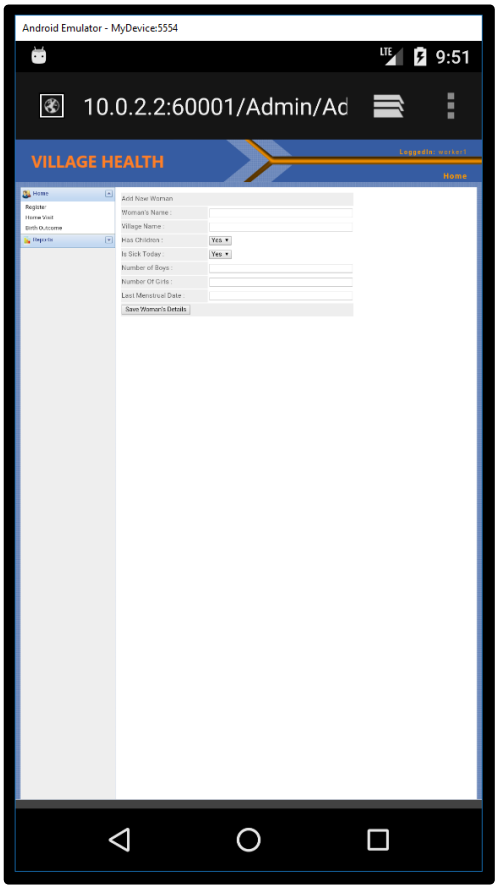
Register a new case

Question (Is Required)	**Data Type**	**Conditions and Logic**	**User Interface**
Woman's Name (Required)	Text		
Village Name (Required)	Text		
Date of Last Menstrual Period (Required)	Date	Calculate and display the expected due date message: "The woman's EDD is:"	 
Has the woman given birth to children that are still alive (Required)	Boolean		

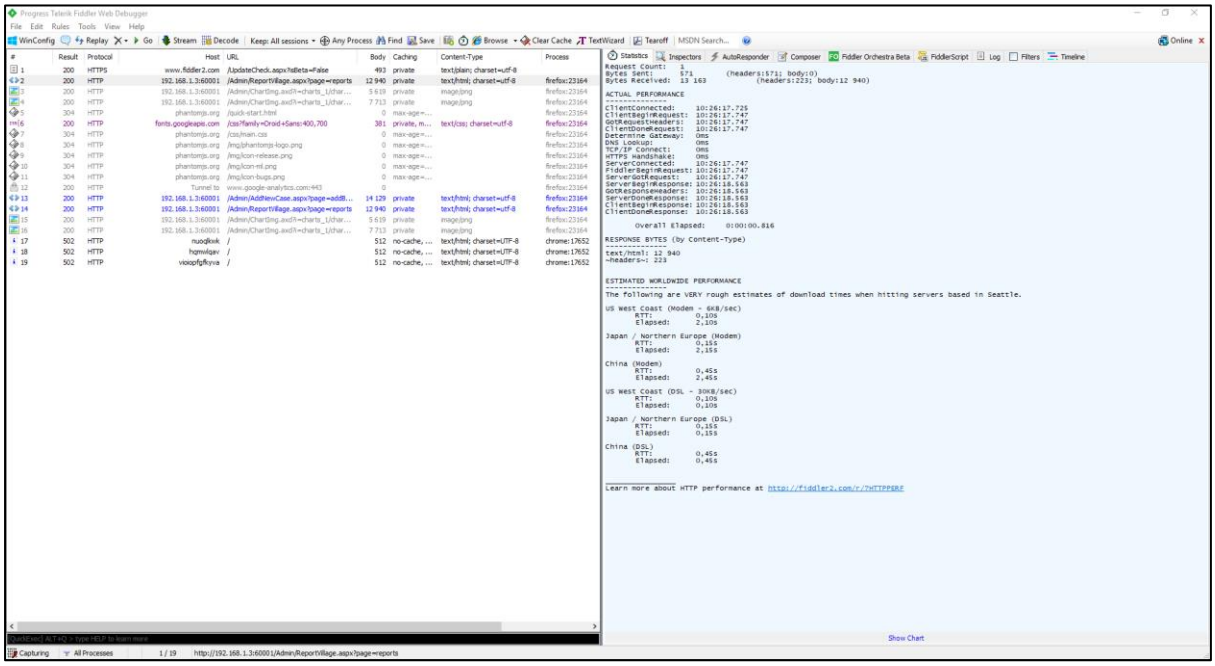
How many living boys?	Integer	If value > 5, display “High Risk” message	
How many living girls?		If value > 5, display “High Risk” message	
Is the woman feeling sick today?	Boolean	Remind the pregnant woman to go to the clinic for her check-up!	
Is the woman feeling sick today?	Boolean		
Has woman been to clinic visit 1?	Boolean	Display if visit 1 != yes	

Appendix B – Village Health Web Apps User Interfaces: Login and Case Registration



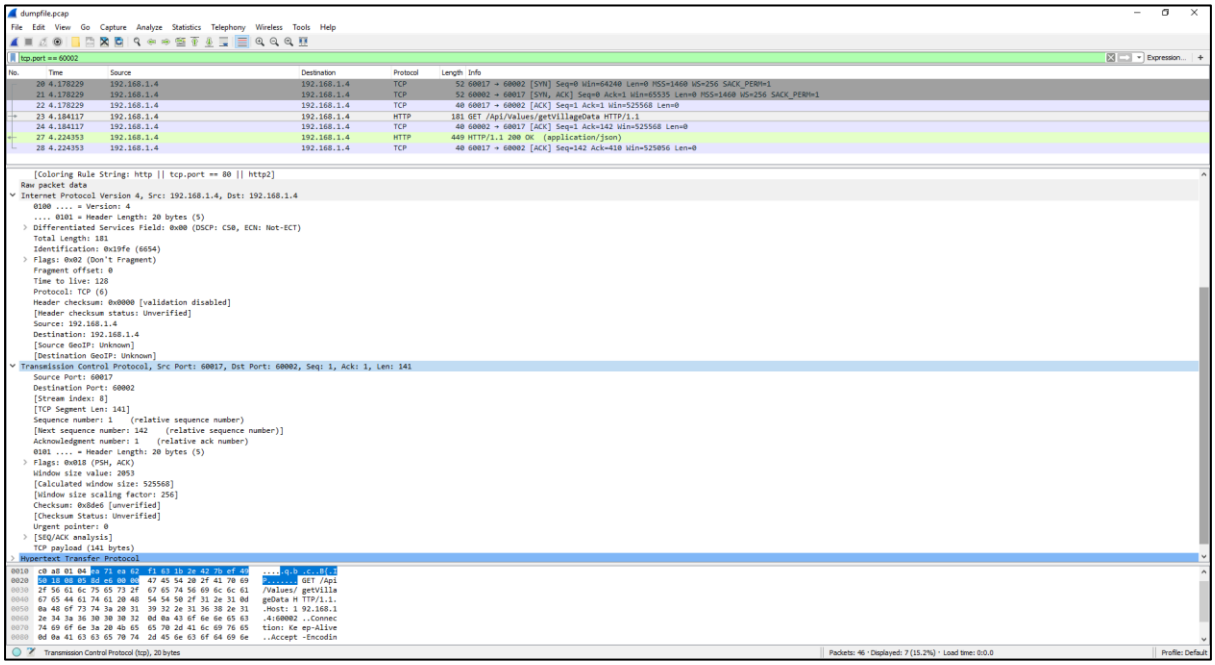


Appendix C – Fiddler Web Debugger: Network Statistics

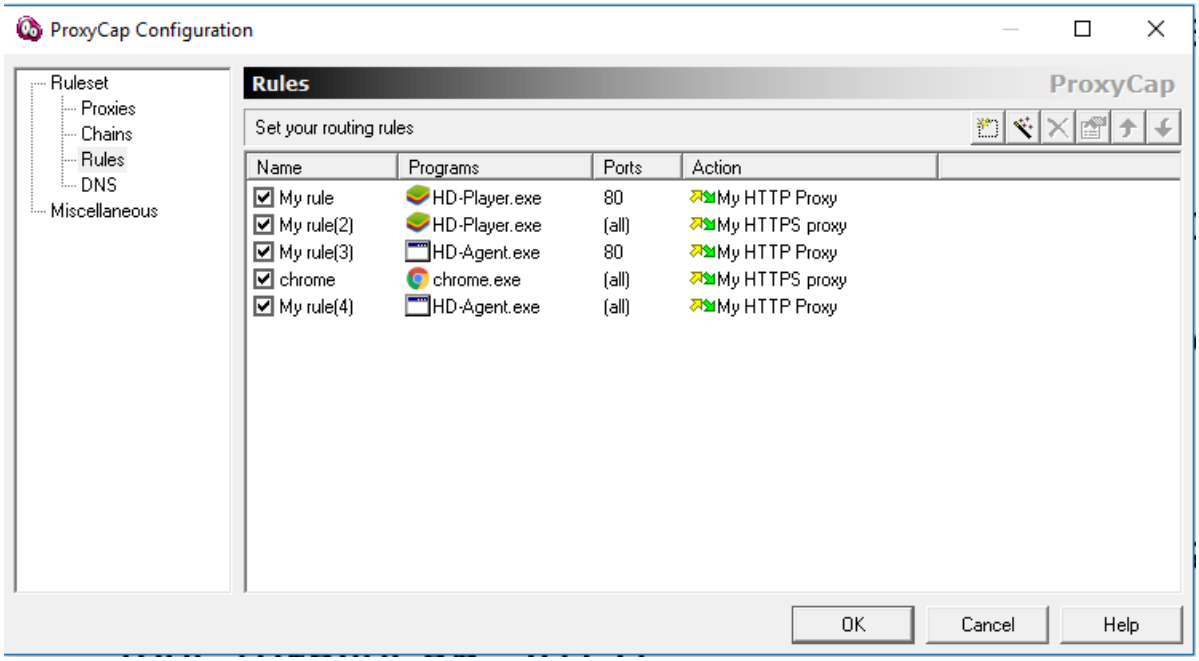


```
Request Count:          1
Bytes Sent:             540   (headers:540; body:0)
Bytes Received:         14 008(headers:223; body:13 785)

ACTUAL PERFORMANCE
--------------
ClientConnected:        20:15:03.025
ClientBeginRequest:     20:15:03.238
GotRequestHeaders:      20:15:03.238
ClientDoneRequest:      20:15:03.238
Determine Gateway:      0ms
DNS Lookup:             0ms
TCP/IP Connect:         0ms
HTTPS Handshake:        0ms
ServerConnected:        20:15:03.035
FiddlerBeginRequest:    20:15:03.238
ServerGotRequest: 2     0:15:03.238
ServerBeginResponse:    20:15:03.491
GotResponseHeaders:     20:15:03.491
ServerDoneResponse:     20:15:03.491
ClientBeginResponse:    20:15:03.491
ClientDoneResponse:     20:15:03.491

Overall Elapsed:  0:00:00.253

RESPONSE BYTES (by Content-Type)
--------------
text/html: 13 785
~headers~: 223
```

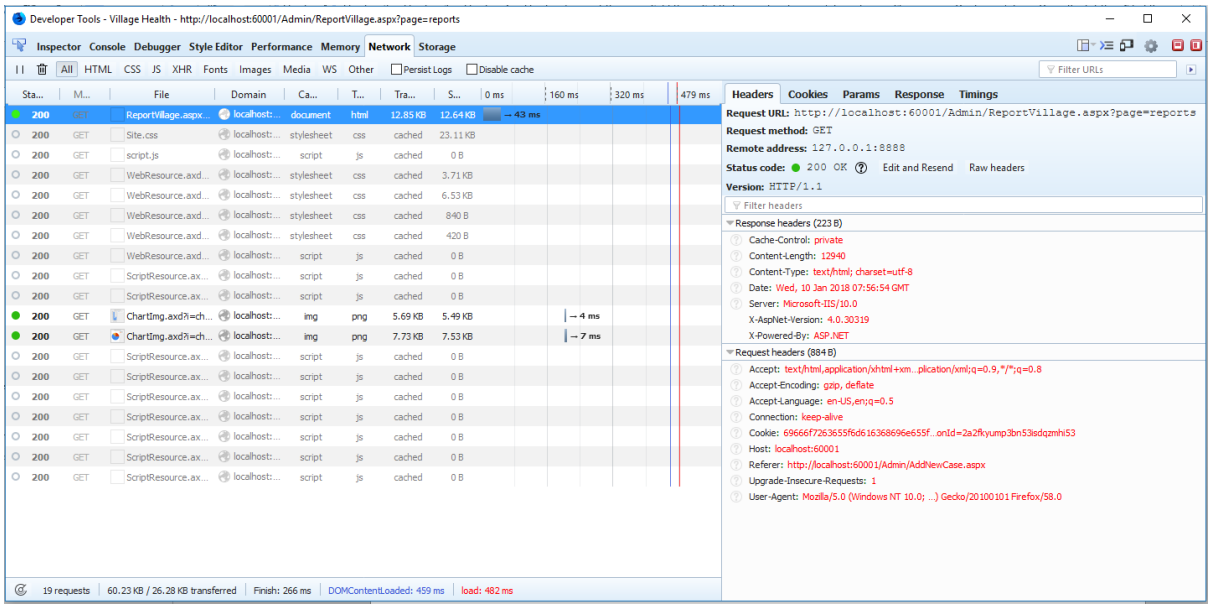
Appendix D – Wireshark: Network Analyser Interface



Appendix E – ProxyCap Configuration Settings



Appendix F – Firefox Network Analyser



Appendix G – Village Health: Data Input Dataset Sample Record


```
[
 {
  "Woman's Name": "[FIRST_NAME]",
  "Total Children": 3,
  "Village Name": "Athlone",
  "Last Menstrual Period": "01/11/2017",
  "Alive Children": "Yes",
  "Boys": 2,
  "Girls": 1,
  "Sick": 0,
  "Sick Today": 0,
  "Request Size": "",
  "Response Size": "",
  "Bytes Sent": "",
  "Number of Responses": "",
  "Overall Elapsed Time": ""
 }
]
```

Appendix H – CommCare Report JSON Data Sample (No. of Forms Submitted by Worker)


```
[
 {
  "User": "worker1 \"FirstName1 Surname1\"",
  "# Forms Submitted": 28,
  "Avg # Forms Submitted": 4,
  "Last Form Submission": "2017-11-15",
  "# Cases Created": 19,
  "# Cases Closed": 5,
  "# Active Cases": 24,
  "# Total Cases": 24,
  "% Active Cases": 100
 },
 {
  "User": "worker2 \"FirstName2 Surname2\"",
  "# Forms Submitted": 5,
  "Avg # Forms Submitted": 0,
  "Last Form Submission": "2017-11-15",
  "# Cases Created": 5,
  "# Cases Closed": 0,
  "# Active Cases": 5,
  "# Total Cases": 5,
  "% Active Cases": 100
 },
 {
  "User": "worker3 \"FirstName3 Surname3\"",
  "# Forms Submitted": 4,
  "Avg # Forms Submitted": 0,
  "Last Form Submission": "2017-11-16",
  "# Cases Created": 4,
  "# Cases Closed": 0,
  "# Active Cases": 4,
  "# Total Cases": 4,
  "% Active Cases": 100
 }
]
```

Appendix I – Wireshark: Network Record JSON Data


```
[
 {
  "_index": "packets-2017-12-16",
  "_type": "pcap_file",
  "_score": null,
  "_source": {
   "layers": {
    "frame": {
     "frame.interface_id": "0",
     "frame.interface_id_tree": {
      "frame.interface_name": "\\Device\\NPF_{9947AF6F-40F3-4280-8976-E7C072E90F2D}"
     },
     "frame.encap_type": "1",
     "frame.time": "Dec 16, 2017 18:30:14.081693000 South Africa Standard Time",
     "frame.offset_shift": "0.000000000",
     "frame.time_epoch": "1513441814.081693000",
     "frame.time_delta": "0.779449000",
     "frame.time_delta_displayed": "0.000000000",
     "frame.time_relative": "2.453060000",
     "frame.number": "25",
     "frame.len": "66",
     "frame.cap_len": "66",
     "frame.marked": "0",
     "frame.ignored": "0",
     "frame.protocols": "eth:ethertype:ip:tcp",
     "frame.coloring_rule.name": "TCP SYN\/FIN",
     "frame.coloring_rule.string": "tcp.flags & 0x02 || tcp.flags.fin == 1"
    },
    "eth": {
     "eth.dst": "00:23:cd:cd:3e:84",
     "eth.dst_tree": {
      "eth.dst_resolved": "Tp-LinkT_cd:3e:84",
      "eth.addr": "00:23:cd:cd:3e:84",
      "eth.addr_resolved": "Tp-LinkT_cd:3e:84",
      "eth.lg": "0",
      "eth.ig": "0"
     },
     "eth.src": "a0:af:bd:ef:5d:3f",
     "eth.src_tree": {
      "eth.src_resolved": "IntelCor_ef:5d:3f",
      "eth.addr": "a0:af:bd:ef:5d:3f",
      "eth.addr_resolved": "IntelCor_ef:5d:3f",
      "eth.lg": "0",
      "eth.ig": "0"
     },……]
```

Appendix J – Shapiro- Wilk Test Results: Standard Web – Amount of Data Used per Case Registration

Summary statistics:

Variable	Observations	Minimum	Maximum	Mean	Standard Deviation
AJAX Web App	20	20 110	21 411	21 299,25	280,40

Shapiro-Wilk test (AJAX Web App):

W	0,28
p-value	<0,0001
alpha	0,05

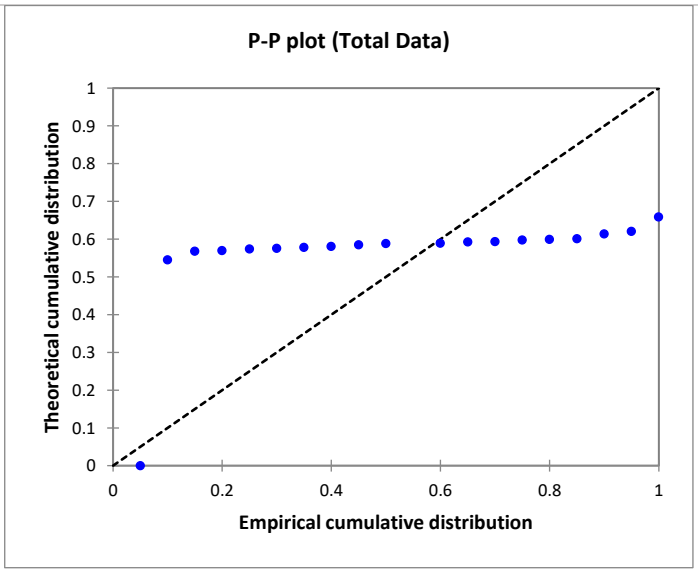
Test interpretation:

H0: The variable from which the sample was extracted follows a Normal distribution.

Ha: The variable from which the sample was extracted does not follow a Normal distribution.

As the computed p-value is lower than the significance level alpha=0.05, one should reject the null hypothesis H0, and accept the alternative hypothesis Ha.

P-P plot (Standard Web App):



Appendix K – Shapiro- Wilk Test Results: AJAX Web - Amount of Data Used per Case Registration

Summary statistics:

Variable	Observations	Minimum	Maximum	Mean	Standard Deviation
AJAX Web App	20	13 253	13 3666	13 593,2	82,97

Shapiro-Wilk test (AJAX Web App):

W	0,43
p-value	<0,0001
alpha	0,05

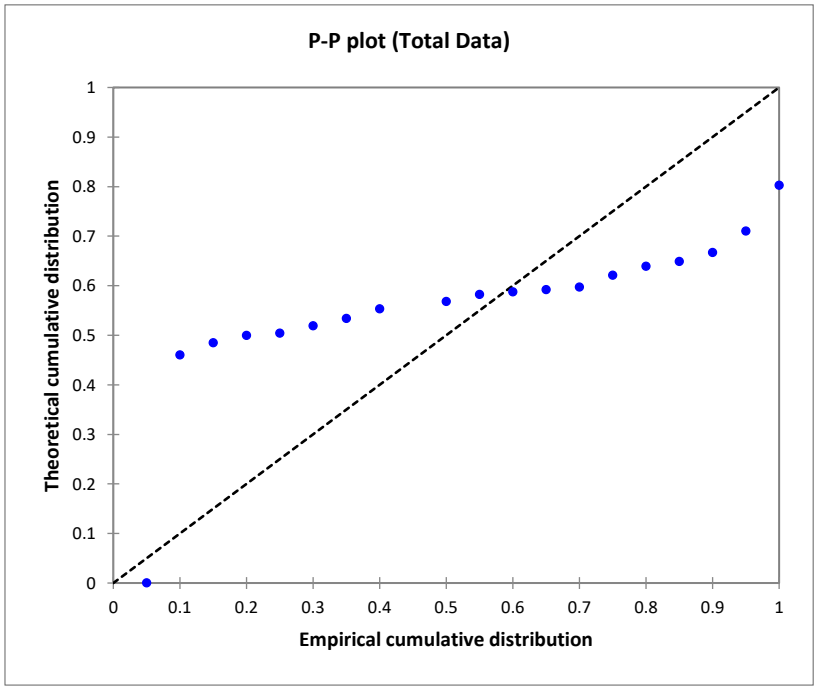
Test interpretation:

H0: The variable from which the sample was extracted follows a Normal distribution.

Ha: The variable from which the sample was extracted does not follow a Normal distribution.

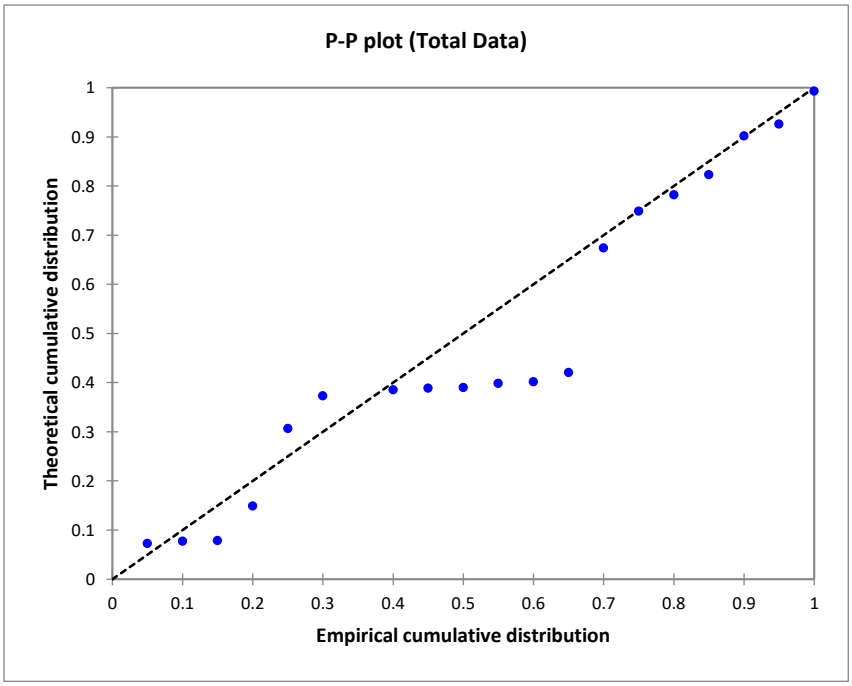
As the computed p-value is lower than the significance level alpha=0.05, one should reject the null hypothesis H0, and accept the alternative hypothesis Ha.

P-P plot (AJAX Web App):



Appendix L – Shapiro- Wilk Test Results: Native App - Amount of Data Used per Case Registration

Summary statistics:					
Variable	**Observations**	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Native App	20	6 871	17 019	10 646,65	2 661,87
Shapiro-Wilk test (Native Web App):					
W	0,92				
p-value	0,14				
alpha	0,05				
Test interpretation:					
H0: The variable from which the sample was extracted follows a Normal distribution.					
Ha: The variable from which the sample was extracted does not follow a Normal distribution.					
As the computed p-value is greater than the significance level alpha=0.05, one cannot reject the null hypothesis H0.					
P-P plot (Native App):					



Appendix M – Mann-Whitney U Test Results: Native App and Web Based - Amount of Data Used

Summary Statistics:

Variable	Observations	Minimum	Maximum	Mean	Standard Deviation
Web App	20	20 110	21 411	21 299,25	280,4
Native App	20	6 871	17 019	10 646,65	2 661,4

Mann-Whitney test / Two-tailed test:		
U	0	
Expected value	200	
Variance (U)	1 366,41	
p-value (Two-tailed)	< 0,0001	
alpha	0,05	
Test Interpretation:		
H0: The difference of location between the samples is equal to 0.		No Difference
Ha: The difference of location between the samples is different from 0.		**Is Difference**

As the computed p-value is lower than the significance level alpha=0.05, one should reject the null hypothesis H0, and accept the alternative hypothesis Ha.

The risk to reject the null hypothesis H0 while it is true is lower than 0.01%.

Ties have been detected in the data and the appropriate corrections have been applied.

Appendix N – Mann-Whitney U Test Results: AJAX Web App and Web App - Amount of Data Used

Summary Statistics:					
Variable	**Observations**	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Web App	20	20 110	21 411	21 299,25	280,4
AJAX Web App	20	13 253	13 666	13 597,2	89,97

Mann-Whitney test / Two-tailed test:	
U	400
Expected value	200
Variance (U)	1 366,15
p-value (Two-tailed)	< 0,0001
alpha	0,05
Test Interpretation:	
H0: The difference of location between the samples is equal to 0.	No Difference
Ha: The difference of location between the samples is different from 0.	**Is Difference**

As the computed p-value is lower than the significance level alpha=0.05, one should reject the null hypothesis H0, and accept the alternative hypothesis Ha.

The risk to reject the null hypothesis H0 while it is true is lower than 0.01%.

Ties have been detected in the data and the appropriate corrections have been applied.

Appendix O – Mann-Whitney U Test Results: Native App and AJAX Web App -Amount of Data Used

Summary Statistics:					
Variable	**Observations**	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Native Total Data	20	6 871	17 019	10 646,65	2 661,87
AJAX Web App	20	13 253	13 666	13 597,2	82,97

Mann-Whitney test / Two-tailed test:	
U	60
Expected value	200
Variance (U)	1 366,41
p-value (Two-tailed)	< 0,0002
alpha	0,05
Test Interpretation:	
H0: The difference of location between the samples is equal to 0.	No Difference
Ha: The difference of location between the samples is different from 0.	**Is Difference**

As the computed p-value is lower than the significance level alpha=0.05, one should reject the null hypothesis H0, and accept the alternative hypothesis Ha.

The risk to reject the null hypothesis H0 while it is true is lower than 0.02%.

Ties have been detected in the data and the appropriate corrections have been applied.