Implementing Speech-to-text Technologies for on-the-go CRM

Malik Mustapha

September 2016

Dissertation submitted in partial fulfilment for the degree of Master of Science in Information Technology

Computing Science and Mathematics
University of Stirling

Abstract

SuiteCRM is an open-source customer relationship management (CRM) tool built as a fork of SugarCRM. The software is intended to allow users to track business critical information and store the data on a database that is accessible to multiple personnel across an organisation. A typical use-case of the software would be accessing a customer account to record notes on a meeting that had been attended. This information is then stored and can be used or accessed at a later date. This is an advantageous piece of software for any business that is increasing its customer base and needs to keep up-to-date records of its customer's activities.

Whilst this software is undoubtedly beneficial within a business environment it has not adapted well to the ever evolving business world. This technology requires users to be connected to a laptop or desktop computer to input or retrieve data. In the past this was convenient, but in a day where most people have a smartphone the desktop method seems archaic and unrefined. Modern smartphones are pocket sized computers that are almost constantly connected to the web. The natural jump therefore is to enable sales professionals to have access to the CRM technology on their mobile devices.

This dissertation documents the development of an Android application that connects to SuiteCRM and is able to call CRUD (Create, read, update and delete) functions on data stored in the CRM database via a REST API. The application implements speech-to-text processing to navigate the UI (user-interface) and create the REST calls to the database. Implementing speech-to-text technologies is a primary focus of the project as it will allow end users to multi-task (travel) while accessing customer records.

The development of the application consisted of three main parts that presented various development challenges. The three components are; the speech-to-text processing, the Android SDK (Android Studio) and the REST API. The primary goal was to get these three components working harmoniously to provide a prototype that would allow the user to speak a set of relatively concrete (some slight prescribed flexibility) commands that the app would then turn into instructions that make the appropriate REST call to the database. After this problem was solved in a timely manner the project took a turn towards usability and end-user testing.

The project succeeded in its aim to harmonise the components, but then evolved into an exercise of refining the app so it was more easily navigable and user-friendly. The objectives for this project were constantly evolving, but the initial objectives for the project were met and this project has extended its initial plan and laid the way for future work in this area.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my University project except for the following:

Some of the code snippets in Chapter 6 have been adapted from online sources, the following code snippets have been amended from the referenced sources:

- Figure 14 & 16 – Speech-to-text processing [20].
- Figure 18 – AsyncTask [21].
- Figure 20 & 22 – SQLite database [22].
- Figure 23 – Google Maps intent [23]
- Figure 25 – Text-to-speech method [24]

Signature **Date**

Acknowledgements

I would like to use this section of the dissertation project to thank everyone that has helped or supported me throughout this project. It has been an enjoyable and educational experience that I will never forget and I am sure the lessons I have learnt throughout will be useful in my future career. In no particular order I would like to thank the following people:

- Dr Mario Kolberg – For being my dissertation tutor and providing constructive feedback whenever it was needed. An incredibly intelligent man that was able to explain things at all levels. Thank you for your guidance throughout this project.
- Paul Godley – Paul was my supervisor away from Mario and helped me with almost all stages of the project. Paul's assistance at SalesAgility was priceless and I am incredibly grateful that he was able to assist on this project.
- SalesAgility – For providing me with the opportunity to undertake this project and develop my skills within the company. I hope our paths cross again in the future.
- Coach Rob Orr – I only came to Stirling because Coach Robert Orr was able to give me the opportunity to come. He gained partial funding for my studies and for that I am eternally grateful; without his help I would never have gained this degree or obtained half the opportunities I have since coming to Scotland.
- I would also like to thank my girlfriend Grace for putting up with me and humouring my descriptions of things I couldn't fairly expect her to understand and for letting me demo my application to her on a daily basis. I would also like to thank my family who have supported me both emotionally and financially on this endeavour and to whom I owe a tremendous amount.

Table of Contents

List of Figures

1 Introduction

1.1 Background and Context

This project was undertaken with the aim of building a mobile application for SuiteCRM that was focussed on being able to utilise speech-to-text technology in a way that would enable the users of SuiteCRM to work on-the-go. The project would be a proof of concept that would act as a platform for SalesAgility, the developers of SuiteCRM, to give them an idea into the viability of developing their own mobile application.

SuiteCRM is an open-source CRM (customer relationship management) tool that aims to provide an enterprise level solution in an open-source environment. A CRM is a tool that tracks measurable aspects of business interactions with stakeholders of a business. SuiteCRM is not only a tool that tracks sales data but it can be used at all levels within an organisation to provide its users with up-to-date information regarding the organisation it is being used within.

Many of SuiteCRM's users are on-the-go individuals that need to travel for work, however there is not current solution for these users that need to interact with the CRM whilst on the go. There is a mobile site available to users that need to access SuiteCRM on their mobile devices, but it requires a high level of dexterity and concentration so is not a viable or comfortable long term solution for on-the-go SuiteCRM.

Building a fully functional, native android application for SuiteCRM that implemented voice-to-text technology was the primary objective of this project. The aim was to have an application that included SuiteCRM functionality but controlled by the users voice. This would then allow users to multi-task whilst operating SuiteCRM. The initial vision for the project was to enable a user to get in the car after a meeting and update the meeting notes whilst driving to the next meeting. The application was also envisioned to act somewhat as a personal assistant that would allow new meetings to be created by voice and to manage other SuiteCRM tasks via voice commands.

1.2 Objectives

Objectives for this project were established in a meeting between the developer and SalesAgility (the client) prior to any development. Creating objectives provided a clear path for the project to follow and allowed for concerns to be raised as to how certain elements would be tackled during the project.

The objectives of this project can be summarised as follows:

- Build an application that can connect to a SuiteCRM instance and perform CRUD operations on the database.
- Have an application capable of processing a user's voice input and using the data generated to perform operations within the application and on the SuiteCRM database.
- Utilise the SuiteCRM REST API to exchange data between the application and the SuiteCRM database.
- Test the application to gain a better understanding of what users would want from the application and to assess how well the developed features work.
- Gain user feedback that could contribute to future development for the application.
- Provide a proof of concept that SalesAgility can use to further their development in pursuing a mobile application.
- Learn how to program with android and work within a comprehensive development environment.
- Gain experience from working in a software development environment with SalesAgility.

These objectives provided a wide scope of activities from Android development to working with REST API's. This type of variety in a project would allow for the developer to improve their skills across a wider number of areas within development rather than just focusing on one aspect. This meant that the project also provided a beneficial experience for the developer.

The following components of the project were considered to be key milestones:

- Get the device to recognise user speech and generate accurate results that could be used to command the application.
- Establish a connection from the application to the SuiteCRM database via the REST API.

- Build a comprehensive android application that performs CRUD operations on the SuiteCRM database.

1.3 Achievements

This section details achievements and benefits of the project.

1.3.1 Tangible Achievements

This project was a success in terms of addressing the objectives that were established. A native android application was built that was capable of performing CRUD operations on the SuiteCRM database via the SuiteCRM REST API. The application developed is also capable of processing user speech to conduct searches and click buttons on the home screen. Being able to utilise speech-to-text technology demonstrated that voice technology is viable in the context of a CRM system.

1.3.2 Experience within a Development Environment.

The project succeeded in delivering a real development experience to the developer. Prior to commencement of this project the developer for this project had limited experience of applying their knowledge of Java programming to any noteworthy or tangible applications. This project allowed the opportunity to not only develop in Java and improve skills, but to develop an application that utilises several different technologies including Java, SQLite, PHP and XML. This is an achievement that could only have been accomplished within a complete development environment. Educational simulations of development do not compare to the real issues that are present in both the setting up and use of a multi-dimensional programming landscape.

1.3.3 Overview of Development Process

This project presented an opportunity to gain insight into each stage of the development process. Requirements elicitation during the initial stages allowed the developer to draw specifications based on requirements outlined by SalesAgility. Design was an ongoing process throughout development, having multiple components that needed to work together allowed the developer to improve their design skills as things needed to be developed with future work in mind. Implementation of the work enabled the developer to work on problem solving skills as well as gain hands on experience of new technologies.

1.4 Overview of Dissertation

This dissertation covers the following in each chapter:

Chapter 1 – Introduction: This chapter provides some background as to why this project was undertaken and the relevance as to why this project needed to be done. The projects objectives are outlined as well as the perceived achievements.

Chapter 2 – State-of-The-Art: This chapter looks at the current state of SuiteCRM and reinforces the need for this project and where it fits in as well as looking at the technologies used within the SuiteCRM application.

Chapter 3 – Methodology: This chapter details the processes and the rationale behind the development of the SuiteCRM application.

Chapter 4 – Requirements: This chapter outlines the use cases for the SuiteCRM mobile application as well as outlines the specifications of the application.

Chapter 5 – Design: This chapter looks at how the various components used within the SuiteCRM mobile application interact.

Chapter 6 – Implementation: This chapter shows how the Application was built and how some of the methods were written to solve problems faced during development.

Chapter 7 – Testing: This chapter shows how the application was tested to uncover flaws that were not realised at the time of development. It also helped to generate ideas regarding future development for the application.

Chapter 8 – Conclusion: This chapter summarises the achievements of the project and evaluates the outcomes of the development. This chapter also highlights future work for the project that didn’t fall within the scope of this project.

2 State-of-The-Art

This chapter details pertinent information regarding the existing SuiteCRM system and how it operates, as well as existing mobile solutions for SuiteCRM. Drawing a landscape of what already exists and how it fits together will make it easier to see where the need for a SuiteCRM mobile application comes from.

2.1 Background

SuiteCRM is a Customer Relationship Management (CRM) software that allows businesses to track customer interactions and store customer information. The benefit of a CRM is the connectivity it provides to a business's workforce. It allows managers to track almost any measurable aspect of interaction, as a tool it is useful across almost all business functions and can be used to measure most activities [1]. The CRM is built around modules [2] that each encapsulates a trackable business function. The most important module in the CRM is the Account module as it stores all the information pertaining to one particular customer. Customer information in the CRM is stored as modules, an example of this can be seen when creating a meeting; to create a meeting the account associated must be found first and then the meeting can be created. The meetings module alone will display all the existing meetings and even allow the creation of a new meeting, but when creating a new meeting it will need to be related back to an account. This level of connectivity is what makes the idea of a mobile application so attractive. Having the ability to input and amend business data on the road provides a system that has more up-to-date real time data.

SuiteCRM was created as a fork of SugarCRM [3] an enterprise level CRM that effectively discontinued its open source product by ceasing unpaid maintenance in favour for a fully paid subscription model. As it stands SuiteCRM is maintained by SalesAgility and they also provide support to users via GitHub forums. SuiteCRM has taken off where SugarCRM left-off, but in recent times they have wanted to experiment with a mobile solution in order to keep up with the paid services and provide the full enterprise level functionality which includes a mobile application.

2.2 System Architecture Overview

SuiteCRM consists of three separate parts; the client, the API and the database. This can be likened to the model-view-controller pattern [4] (Figure 1) within information systems architecture. This pattern ensures that the manipulation of the database (model) can be controlled by the REST API (controller) that is ultimately controlled by the user through the client (view). This style makes the system more robust as the controller is the only component interacting with the model. In the case of Suite CRM this is especially pertinent when considering multiple users. If two users had direct access to the database issues would occur around the state of information if they were being accessed simultaneously. This problem is compounded when applied on a larger scale. This section aims to discuss each component of the system and how it currently operates. This will then pave the way for discussions about how a dedicated SuiteCRM mobile application would fit into the landscape.

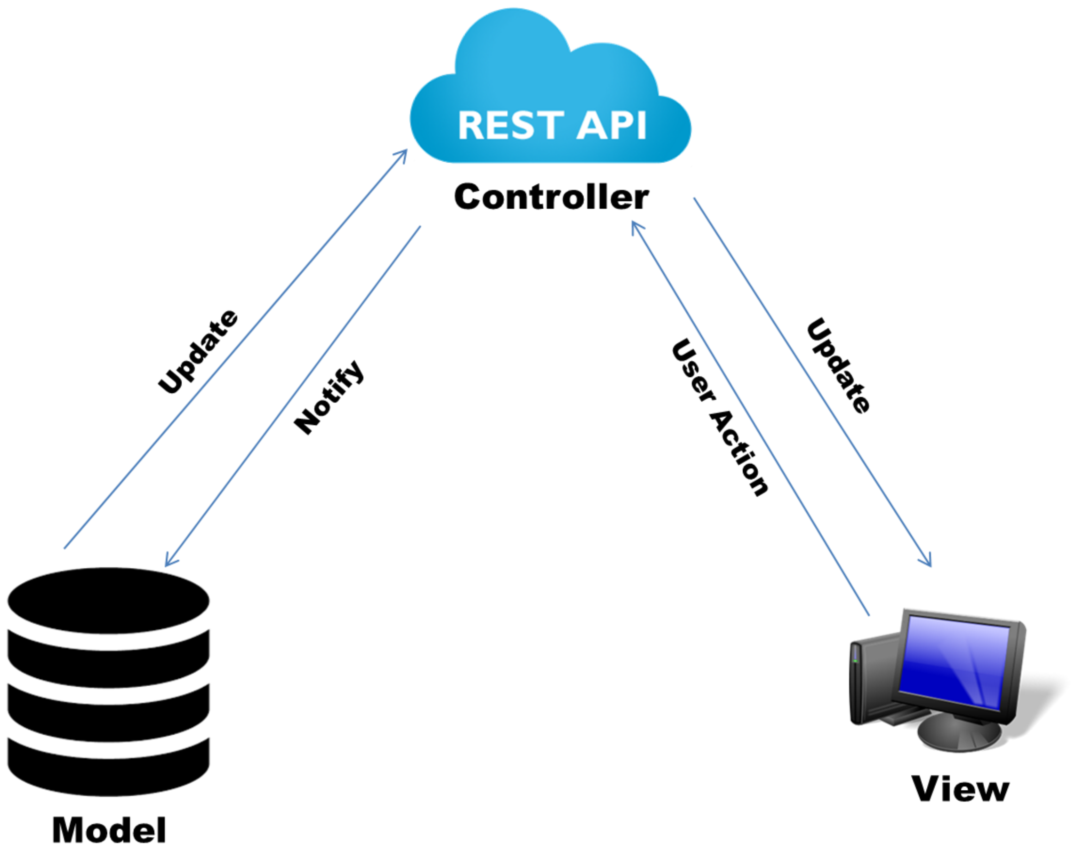


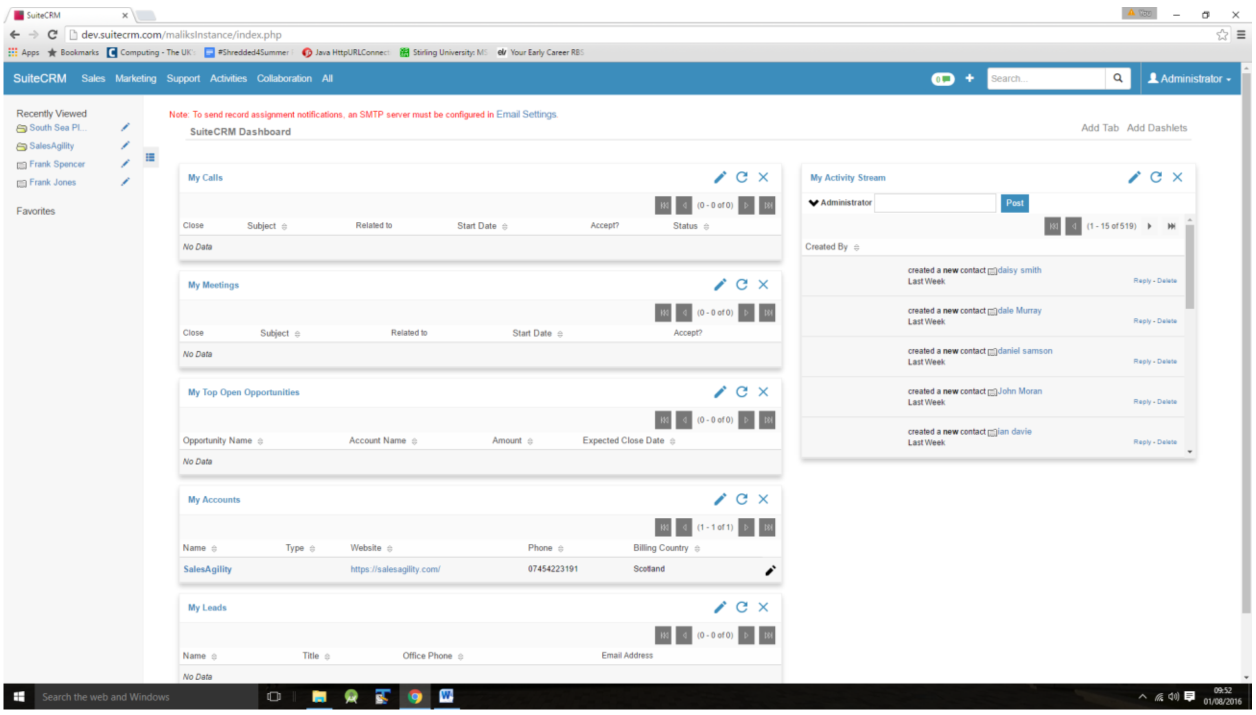
Figure 1 - Model View Controller

2.3 SuiteCRM View

This section will focus on the views available for SuiteCRM. The view is the part of the system that displays the data stored in the database. The view also allows users to interact with the system via controlled input methods. In the context of this dissertation the view will encompass the web client for desktop and any mobile solutions that are discussed.

2.3.1 SuiteCRM on Desktop

SuiteCRM is a web application that is built to be run on a desktop computer. Running in the browser means that it does not need to be installed on a computer nor does it restrict users to specific computers. This level of portability is beneficial to the user as it makes the information accessible anywhere in the world provided they have access to a desktop or internet enabled device. As it is run in a web browser it does have mobile support, however this is not a complete on-the-go solution and will be discussed further in section 2.2.3.



Note: To send record assignment notifications, an SMTP server must be configured in Email Settings.

SuiteCRM Dashboard

My Calls (0 - 0 of 0)

Close	Subject	Related to	Start Date	Accept?	Status
No Data					

My Meetings (0 - 0 of 0)

Close	Subject	Related to	Start Date	Accept?
No Data				

My Top Open Opportunities (0 - 0 of 0)

Opportunity Name	Account Name	Amount	Expected Close Date
No Data			

My Accounts (1 - 1 of 1)

Name	Type	Website	Phone	Billing Country
SalesAgility		https://salesagility.com/	07454223191	Scotland

My Leads (0 - 0 of 0)

Name	Title	Office Phone	Email Address
No Data			

My Activity Stream

Administrator (1 - 15 of 519)

Created By

created a new contact daisy smith
Last Week Reply - Delete

created a new contact dale Murray
Last Week Reply - Delete

created a new contact daniel samson
Last Week Reply - Delete

created a new contact John Moran
Last Week Reply - Delete

created a new contact ian davie
Last Week Reply - Delete

09:52
01/08/2016

Figure 2 - SuiteCRM on Desktop

As can be seen by the screenshot in Figure 2 the view presents the information in a modular way (i.e. My Calls, My Messages etc.). This modularity makes it clear for users to manage their information by decomposing each activity into a module. Not only does it make it easier for users, but it creates opportunities for users to customise their view therefore making it a flexible tool.

The desktop web application is currently the most used and most supported way of using SuiteCRM, there are other ways of accessing the data via mobile devices etc. but this is still the most used route. With users however becoming more mobile and competitor products offering mobile solutions the desktop version will get left behind if it is not supplemented by an effective mobile solution.

2.3.2 SuiteCRM on Mobile

SuiteCRM as a web enabled CRM platform is supported on mobile devices, but only through the browser. This means that users are able to access information on the go provided they have an internet connection and that they are in a suitable place to navigate the website. A problem with navigating the browser on a mobile device is that it requires precise selections and a high level of dexterity from the user. This where a dedicated mobile application could be of use and provide the on-the-go functionality that SuiteCRM users need.

Currently SuiteCRM is only supported by one mobile application QuickCRM [5] developed by a French CRM consultancy 'NS-Team'. The application requires an expensive plug-in as well as an expensive licencing fee. This therefore exposes a gap in the market for a SuiteCRM mobile application that is provided by and maintained by the maintainers of SuiteCRM. The benefit of having the mobile application is that it provides users with the on-the go functionality that they are currently just not getting and are unable to get reasonably elsewhere.

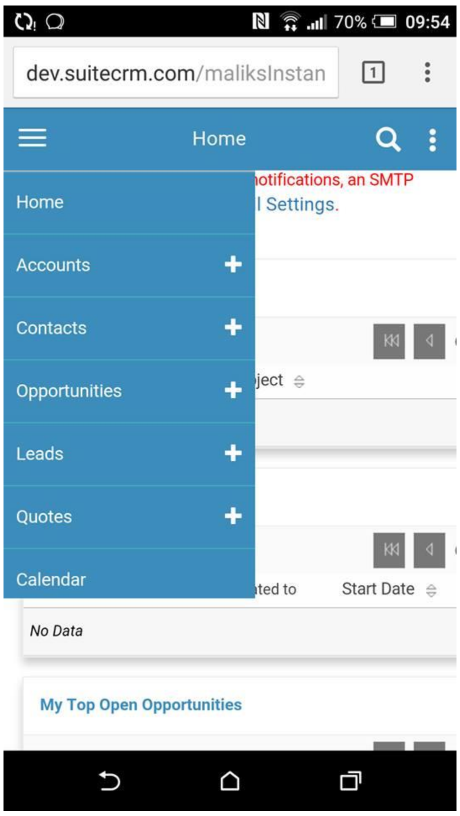
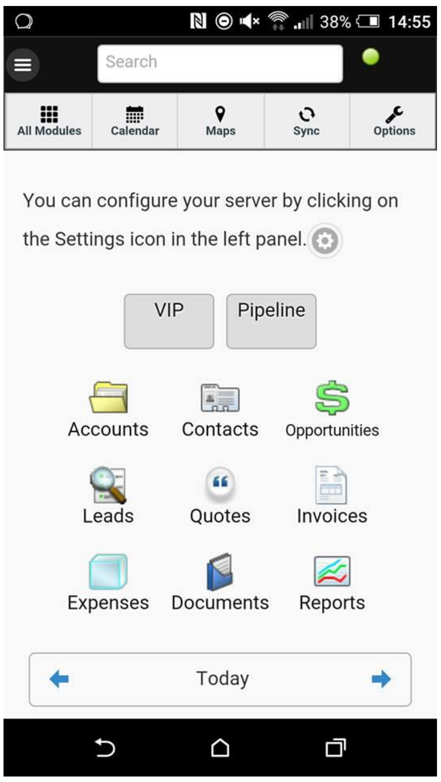


Figure 3 - SuiteCRM running on mobile browser

2.3.3 QuickCRM

Quick CRM is the only native application that allows interaction with a SuiteCRM system [5]. QuickCRM allows access to SuiteCRM records, but is also compatible with SugarCRM. QuickCRM provides a base plate for this project by demonstrating how a mobile application that interacts with SuiteCRM is achievable. The application seen in Figure 4 shows the landing page for the QuickCRM application. The layout is quite clear and displays functionality that is clearly useful in a CRM mobile application (access to customer relevant data) that utilises SuiteCRM. As in the desktop version of SuiteCRM there is an emphasis on the modules and they are presented for the user to interact with from the beginning. As solution QuickCRM is functional, however users are required to pay for a plugin that could cost a considerable amount if it needed to be deployed across an entire business.

QuickCRM was a useful tool that demonstrated many features that could be used within the SuiteCRM application and features such as Google Maps integration in QuickCRM certainly inspired the decision for the Maps functionality with the SuiteCRM application.



You can configure your server by clicking on the Settings icon in the left panel.

Figure 4 - QuickCRM application for accessing SuiteCRM

2.4 SuiteCRM Controller

This section highlights the REST API that facilitates communication between the SuiteCRM database (SuiteCRM model) and the view that has already been discussed. The role of the REST API is to manage the interaction between the user and the data they wish to access [6]. The REST API carries out functions on the data and is essentially the brains of the system. Its ability to call and modify data is what makes SuiteCRM an interactive application.

2.4.1 SuiteCRM REST API

An API (Application Programming Interface) abstracts the implementation of a software or system and allows other applications to utilise its features without disclosing the implementation [7]. An API acts as an interface for other applications to interact. Interactions could include anything from performing a function on some data (much like the android.speech API) to interacting with a different application altogether (Google Maps API).

REST APIs rely on an HTTP connection and utilise the HTTP verbs; GET, PUT, POST and Delete (CRUD cycle). Everything in a REST API call is made using the URL and the body of the request.

This project relies heavily on interaction with the SuiteCRM application. SuiteCRM utilises a REST API which serves its function much like any other API in that it provides a means of interacting [8], however the execution of how the interaction takes place is what differentiates it. REST (representational state transfer) is the term used to describe the type of interaction utilised by the API. REST as a method of web service interaction is preferred over SOAP (Simple Object Access Protocol) due to it being more lightweight and requiring less bandwidth [9]. This lower use of resources is what makes a REST API the preferred choice when dealing with mobile technologies such as the SuiteCRM application [10].

A good way to visualise the SuiteCRM REST API is to imagine it as sitting between the application and the CRM database like in Figure 5. This demonstrates how if the application needs information from the CRM database it consults the REST API and the REST API then makes the request to the CRM database that then serves up the relevant information before the REST API sends it back to the application. This layering system means that the REST API allows the CRM to be platform independent and therefore available to a wider variety of users.

In essence the REST API itself was a black-box throughout the development, and its inner workings are really beyond the scope of the project at hand, however it creates a clearer picture of how the application communicates with the system when it is understood.

Opportunities for future functionality of the application can be greatly enhanced when the REST API is understood in more detail.

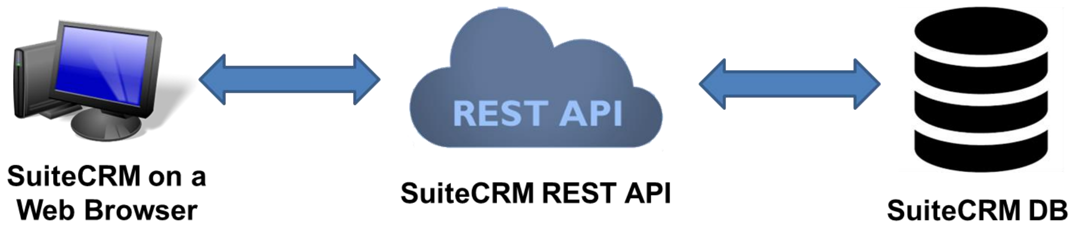


Figure 5 - Where the REST API sits in relation to the entire system.

2.4.1.1 Slim Framework

Slim framework is a micro framework that is used to build lean lightweight APIs. SuiteCRM's REST API employs the Slim framework as a routing tool. "At its core, Slim is a dispatcher that receives an HTTP request, invokes an appropriate call-back routine, and returns an HTTP response" [11]. This definition of the Slim framework best describes how it functions in a succinct way, it simplifies the API by essentially defining the call-back routine in a PHP file at the beginning of the HTTP request path. In the SuiteCRM REST API a typical call to the API looks like this, "http://dev.suitecrm.com/maliksInstance/Api/V8/module/Accounts" slim is invoked at the "/Api" where it takes over and runs the call-back routine. Slim Framework is not an essential part of this project, however discovering how it is used demystified the black-box nature surrounding how the REST API functioned.

2.5 SuiteCRM Model

This section aims to outline the SuiteCRM database that sits behind the REST API. This is where the information displayed to the view is stored. The SuiteCRM database does not do anything other than store information.

Under the hood exists a vast number of MySQL tables with interdependencies that tie them together and allow the controller to build a profile in the view and display it to the user. Every object in a table has an ID as its primary key; this ID can be found across all tables where information is related. The controller (REST API) is responsible for generating the SQL statements that produce the desired data to present in the view to the user.

2.6 Evidence based requirements

Based on the findings in sections 2.2 – 2.5 a list of core functional requirements could be drawn the highlight what the expectations for the finalised requirements could be. These requirements were manifested from areas where the SuiteCRM mobile application can hopefully fill a void and provide the functionality required by on-the-go SuiteCRM users. The core requirements based on research were as follows:

- Create a native android application that provides access to the same information available in the desktop version of SuiteCRM.
- Create a native android application capable of interacting with the SuiteCRM REST API.
- Provide an easily navigable UI that allows for simple selections and clear display of information.
- Enable users to navigate the application using voice commands to issue shortcuts or execute jobs.
- Integrate the application within its android environment by allowing direct access to other native applications such as the dialler, Google Maps and any local messaging applications.
- Provide an on-the-go SuiteCRM solution that acts as a proof of concept rather than a finished product.

2.7 Supporting Technology

2.7.1 Voice-to-Text

Voice-to-Text processing is the act of receiving a voice input from a person and translating the voice into text that can be used or viewed by a machine. This is usually achieved when the voice data reaches the server or device that is processing it. Speech is broken down into phonemes and processed by systems that are capable of recognising different words, patterns and even context. This technology is rapidly developing and becoming more accessible for developers with Google offering the service free with the android.Speech library available for use within mobile applications.

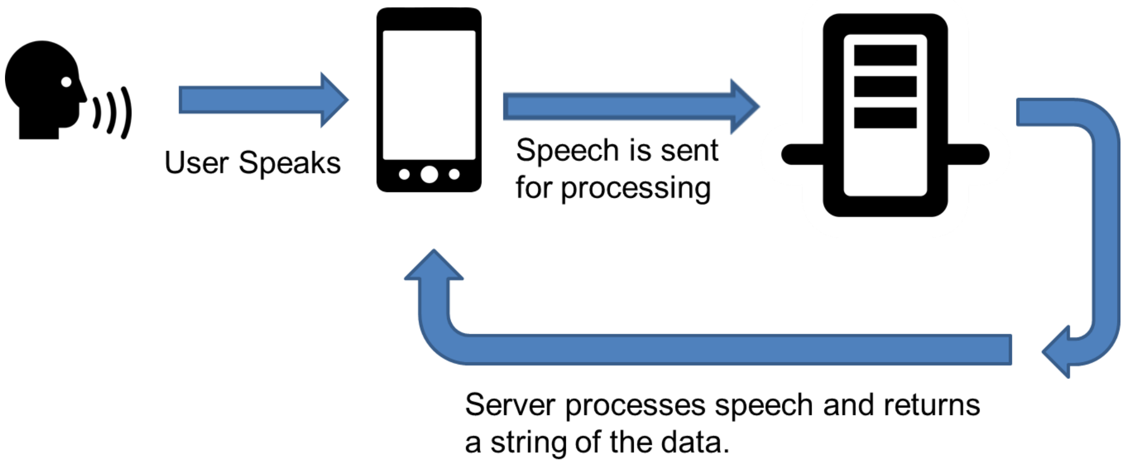

User Speaks
Speech is sent for processing
Server processes speech and returns a string of the data.

Figure 6: Voice-to-Text being processed by a remote server.

Figure 6 displays how the voice-to-text functionality supplied by android.speech works. The user’s voice input is recorded by the device before being sent to a server to be processed. Upon completion of processing the user’s voice, the server returns a string to the device that can then be used in any intended way. This form of voice processing relies on a steady internet connection and clear diction from the user.

This technology is also available in a more primitive solution that requires far more work from the developer. CMU (Carnegie Mellon University) developed CMU Sphinx [12], open source software capable of processing spoken language into text on a device. This software whilst incredibly capable is complex to work with and involves potentially having to build your own language model if you wanted to support users in unsupported languages. The built-in android solution for voice-to-text processing supports the widest user base and is the most flexible in terms of detecting different accents.

Within the SuiteCRM mobile application the use of voice-to-text processing would be used to enable the user in navigating the application with a particular focus on searching. Being able to accurately search for a record in the CRM by voice would allow the user to avoid having to type what they are looking for and would enable more multi-tasking. The role of voice-to-text processing in this project is to enable the user to multi-task whilst navigating the CRM or to at least speed up the tasks that would usually take them longer to type. Another area that voice-to-text would benefit the application is in the recording of meeting notes. By being able to dictate the notes by voice, it prevents the user from having to type out potentially lengthy notes.

The intention behind implementing voice-to text is to make using the CRM a more autonomous task. The less work the end user has to do manually the better. Being able to input data vocally removes the need to be staring down at the keyboard.

A potential difficulty with a system that is entirely dependent on voice-to-text is the level of errors that are still existent within this technology and the inabilities to get the context 100% correct [13]. While a spelling error in a meeting note is probably not a catastrophic event or a deterrent for use. It could cause larger problems if the intended actions were not understood correctly by the application during a search or if data was altered due to not being able to translate the users' needs correctly.

2.7.2 Operating System

The entirety of the technical part of this dissertation project was undertaken on a Linux machine running Ubuntu (version 4.8.4). This OS (operating system) was chosen as it was the OS of choice for development within SalesAgility due to it being open source. Differences in UI (user interface) compared to more mainstream OS's such as Microsoft's Windows and Apple's Mac OS meant that part of the development process included getting used to the way the new OS worked. Whereas all operating systems are text-based at their core many these days have what is known as a GUI (Graphical User Interface) that enables the user to point-and-click & drag-and-drop the files or links they wish to access. Ubuntu marries together the idea of a powerful text-based command-line OS and the popular GUI OS's that are widely used today. However it is essential for any Ubuntu user to have a fundamental understanding of the Linux Terminal if they wish to achieve anything meaningful on the machine.

The first task in this dissertation project was downloading and installing the IDE as well as the JDK (Java Development Kit) in order to begin development of the application itself. This involved using the Linux Terminal to check and download the latest version of the JDK from (http://www.oracle.com) and also to download Android Studio (Androids official IDE).

2.7.3 IDE

As mentioned in the previous section Android Studio was selected as the development IDE. This was due to a number of factors. Android Studio is the official IDE for android development released by Google in 2013[14]. It was developed by Google to compete with Eclipse the popular IDE for android development at the time. Eclipse was not the most stable platform for android development and many users reported the IDE crashing. Part of the reason for this is that Eclipse was a much larger program than Android Studio and required more system resources as a result. This meant that lower powered machines were not equipped to run Eclipse to its full potential and were more likely to suffer from unresponsive behaviour.

Being developed by Google (the developers of android itself) Android Studio is the logical IDE to use. Android Studio is also incredibly capable out of the box and features available are also built to aid rapid development of android applications. An example of such a feature is the ability to drag and drop activity elements (buttons, text fields etc.) onto the GUI. This enables the developer to design a relatively polished looking GUI in minutes. Android Studio also features the Gradle, a tool that builds the .apk (Android Package Kit) file and pushes it to the target device via the ADB (Android Debug Bridge) for execution. Having a wide variety of tools that work harmoniously all under one roof made Android Studio a very attractive IDE when choosing.

Since Eclipse has become outdated for Android development the amount of in-date support available online is dwindling. Support for Android Studio is increasing every day and many of the forums are incredibly active. There is also a rich repository of libraries available at (https://developer.android.com/index.html) that provided development support for many elements used in this project. Whenever there was a moment of ambiguity throughout development of the project, the solution was usually available in the android developer’s website. Anything that was not detailed on the android developer’s web page regarding the functionality of Android Studio was usually available at stack overflow (http://stackoverflow.com/). The amount of support available for Android Studio was the primary reason for choosing it as the projects IDE.

A central component of the android IDE is the emulator. The android emulator simulates a real android device on the developers machine (Run Apps on the Android Emulator, 2016) and allows rapid development of applications by bypassing the need to test on physical device. Development of the SuiteCRM application was run on Nexus 5X device through the emulator. This is the default virtual device, by using the default device the intention was to remove any features that would be device dependant. The Nexus line of consumer products run on stock android that comes straight from Google. Many mobile phone manufacturers place their own altered version of android on their devices; for example both Sony and HTC run android on their consumer mobile devices, but they are aesthetically different. The changes that manufactures make are not only cosmetic, but many embed their own applications to differentiate their product. For this reason the emulator used the default Nexus 5X device as not to be affected by device dependent factors.

The emulator allows the developer to see what the application would look like on a real life target device and allows the simulation of different scenarios and different screen sizes. Being able to adjust the environment of the device allows for a faster and more rounded development as testing a device in perfect conditions does not reflect real use by real users. Conditions that can be manufactured by the emulator include; changing network type for data, roaming status for both data and voice calls, battery status including level, health and charging status as well as simulating interruptions such as receiving a text or phone call.

2.7.4 Setting up the IDE

Setting up Android Studio is a relatively straightforward process, once installed a new project is created, at this point the name of the application is set (can be changed later) and the project location is established on the system. At this point the API level is set that dictates the age of device the application will run on. The lower the API level, the fewer the number of features available to it. As this project requires features such as voice recognition and google maps integration the target SDK (Software Development Kit) was version 23. The minimum SDK version however is API level 15. Whilst the application would work on older devices, these models would soon become unsupported over time as the application grew and more advanced technologies were utilised. Once the project has been set-up the developer is able to begin writing code.

2.8 Prescribed Development Tools

This section briefly discusses each of the prescribed development tools. These are tools that are used by SalesAgility in the development of SuiteCRM. During the project use of these tools was default and already in place ready to use. This section just highlights the role each component played in the development process.

2.8.1 Development fork of SuiteCRM

A useful tool throughout the development stage was having a closed instance of SuiteCRM that could be amended and altered by the application without having catastrophic effects on a real version. This enabled full care-free testing of the CRUD life-cycle on the test data available on the closed instance. The fork of SuitCRM that was used was loaded onto the development machine and all calls through the REST API were going to localhost (the development machine). After development was completed the SuiteCRM instance was pushed onto the development server(http://dev.suitecrm.com/) for testing. Testing was the first time the application was connecting to SuiteCRM over an internet connection from a mobile device.

2.8.2 LAMP

LAMP stands for Linux, Apache, MySQL and PHP. The definition of LAMP also spans Perl and Python, but throughout this project Perl and Python are not used. The LAMP architecture is widely viewed as the best way to build an enterprise-level web application [15]. SuiteCRM is developed and used in a LAMP environment that is by its very nature open source, this reinforces the idea that open source platforms are capable of building enterprise level applications. The next few sections will detail the various LAMP constructs that were utilised in the development of the SuiteCRM mobile application.

2.8.3 Apache HTTP Web Server

The purpose of a HTTP (Hypertext Transfer Protocol) web server is to serve information based on HTTP requests made. The Apache HTTP server is the software installed on a machine that fulfils HTTP requests and returns the relevant object. Since April 1996 Apache’s http web server has been the number one web server on the internet [16]. For the majority of the development of the SuiteCRM application Apache’s HTTP server has been installed on the local machine and the HTTP requests have been aimed at the local host. This meant that during development there was no dependency on an internet connection and the quality of the connection did not impact the development process. Later in the development phase of the project a dedicated application instance of the SuiteCRM was placed on the server to simulate real world conditions.

2.8.4 MySQL

As can be seen in the requirements matrix (Appendix 4) SuiteCRM requires the user to have a database in place (Maria DB, MySQL or SQL Server). SalesAgility machines utilise MySQL due to its fully featured open source nature [17]. SuiteCRM stores all of its data in MySQL tables that can then be queried by the REST API that fires SQL statements at the database in order to retrieve the results.

2.8.5 phpMyAdmin

phpMyAdmin is a browser based software tool built to handle the administration of MySQL databases over the web. SalesAgility use this tool to manage their internal version of SuiteCRM, however for the purpose of this paper it is being mentioned to provide a more informed view of how the application communicates with SuiteCRM and the underlying structure of the CRM and how it is managed. phpMyAdmin provides a GUI that allows visualisation of data as well as providing a WYSIWYG (what you see is what you get) interface that allows the performance of operations on the tables in the SuiteCRM database.

2.8.6 PHP Storm

PHP Storm is an IDE for PHP development. It is the primary IDE used at SalesAgility as the majority of their work is done in PHP. For the purpose of this project PHP Storm was only used to view the functionality of the REST API and gain an understanding of how it worked. The primary function used within PHP Storm was the debugger, it was used to detect if the application was utilising the REST API and that the calls being made by the application were correct.

2.9 Elected Development Tools

This section covers the development tools that were selected for aiding development. These tools were not strictly part of the SalesAgility setup and were chosen by the developer. Each subsection will briefly mention the role that each component played and how it contributed to the development of the SuiteCRM application.

2.9.1 Sqliteman

Sqliteman is an Ubuntu application that allows the user to visualise a SQlite3 database. Sqliteman was used to view the contents of a SQlite3 database that was created during the development of the application. Visualising the database enables the developer to have a better understanding of the shape of their data and how it is being stored. In the context of this application, a relatively small amount of data was being stored, but being able to visualise how the data was being stored enabled an understanding of how the data would be updated and amended throughout the project. Having this resource available helped development significantly.

2.9.2 GitHub

A dedicated GitHub repository was set up to store the project. This reduced the risk of losing work done on the project and also gave the project more portability in terms of when and where work could be done on it.

GitHub is Git repository; a Git is a computing term for a VCS (Version Control System) that enables developers to store branches (different versions) of their source code. For this project, GitHub was used as a portable storage solution negating the need for any physical media such as memory sticks or discs. This not only made the project accessible at any time, but protected it from loss or damage. Using GitHub also provided experience in what is a widely used tool within software development or any area where version control is important.

3 Methodology

This short chapter aims to provide some context as to how development of the application was conducted. The processes that were followed and the general principles that enabled progressive development will be detailed here.

As a lot of learning was being done at the same time as development there were many surprises along the way and many implementations did not work as intended. Therefore as can be seen in Figure 7 the software development cycle consisted of gaining requirements, designing a solution, implementing the solution and testing it. This was an iterative cycle that was repeated incrementally for each requirement in order to build a robust solution.

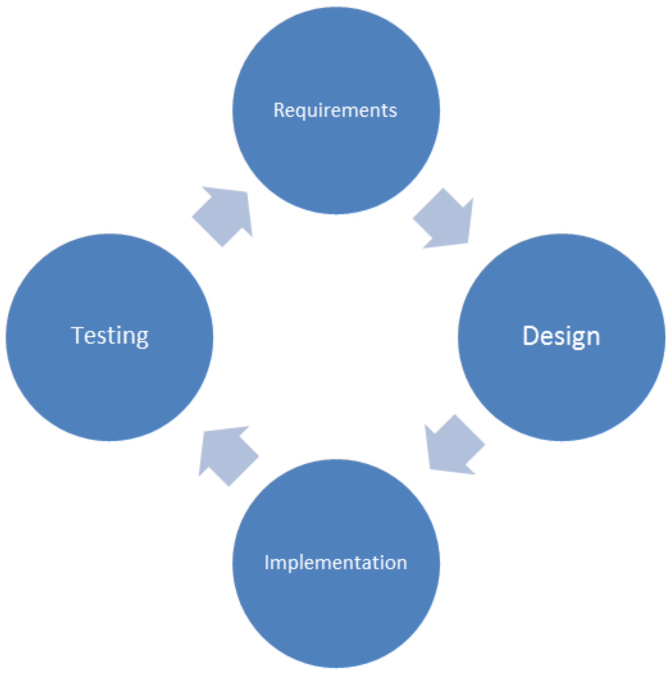


Figure 7 – Development cycle throughout the development phase

This development style created a set of dynamic sub-requirements that were constantly changing due to priority and the manifestation of novel ideas. Having to constantly evaluate the necessity of certain sub-requirements and features within the application kept development interesting and avoided getting hung-up on any one particular element.

An incremental and iterative hybrid development style was used for developing the application. The main focus was to get the three main components working together (Android, REST API and voice-to-text) then to add features piece by piece to provide a well-rounded prototype of a SuiteCRM application.

This development style led to multiple proto-type applications that would perform a dedicated function. The implementation of these proto-type applications were then brought together to build a functional SuiteCRM application.

This project fell very much into the camp of agile development that focussed on the iterative nature of software development. Many of the features included in the final SuiteCRM application were not requirements initially, they were features that were added on after being tested or scrutinized by the client. The agile methodology as outlined in the agile manifesto [18] sees software development as much more of an ongoing process that welcomes change rather than a highly structured, inflexible process that is recommended in methodologies such as the waterfall model where everything needs to be meticulously planned and thoroughly documented. The agile methodology focusses on the production of good software developed in a timely manner. This project certainly followed these principles.

4 Requirements

The agreed upon objective for this project was to create a prototype SuiteCRM application in android that is capable of voice-to-text translation. This provided a lot of scope for what direction the project could go in. It was agreed that the application would need to utilise the android SDK, voice-to-text libraries and the SuiteCRM REST API. These three areas were the building blocks of the project for which a more rounded SuiteCRM application would be built around.

Due to the explorative nature of the development process many features added to the application became requirements as the scope of the application grew and the technology was better understood. For example in the initial stages there was no provision for an offline mode, but when alternative solutions were looked at, offline mode seemed like a natural progression. Offline mode was added because of the mobile nature of the product; users that are out on work may not always be in areas with network connection, making the application redundant. By adding offline support it means that users can still view recently accessed information without connection to a network.

Throughout the project sub-requirements were added and removed as it developed. Android Studio proved to be an excellent development tool that made some features very easy to include therefore they were added to the requirements. An example of where this happened was with the autoLink feature in the layout tool. It is possible to give a TextView link capability in the layout builder in Android Studio. Instead of writing onClickListeners or creating links, Android Studio allows you to check a box that states that a TextView is a link. This feature was going to be used for clicking on customer email addresses or web pages to take the user through, but due to other programming difficulties this feature was handled by a separate button. That is how requirements were subject to change throughout development.

The final requirements for the SuiteCRM application were:

- Be supported on android devices.
- Connect to SuiteCRM via the new REST API.
- Perform CRUD operations on the SuiteCRM System.
- Allow users to search for records via voice command.
- Integrate the SuiteCRM application with Google maps technology.
- Enable users to still use the application offline.

- Allow users to search for specific records.
- Enable users to use interact with the SuiteCRM data via other native applications (i.e. native email client, dialler or web browser)

Part of the initial requirements analysis that was conducted was use case modelling, this is method is used to give development more focus towards the user while developing. Use case modelling looks at real world uses for an application and focuses on what their desired outcomes will be. The main element of use case modelling is to see how the user interacts with the application.

4.1 Use Cases

The use cases contained within this project are from the perspective of the user. The application is simply a view to the information in the model so there are no administrator views. The application is intended for end users only therefore any administrative processes should be undertaken on the desktop version of SuiteCRM.

This section aims to identify and justify use cases for the SuiteCRM application based on the requirements outlined in the chapter 2. For a summary of the use cases the use case diagram in Figure 8 shows what the end user can expect to achieve when interacting with the SuiteCRM mobile application.

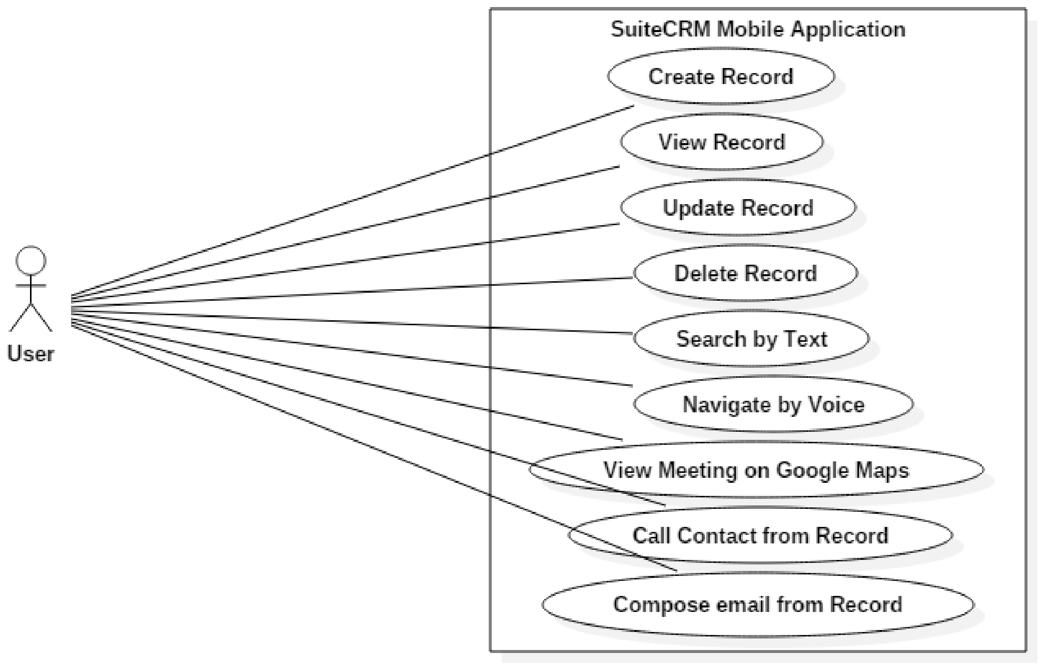


Figure 8 - Use case diagram for SuiteCRM application

This section will look to build on the use cases shown in Figure 8 and state the case for why it would be used by a user. The order reflected in the diagram will be followed throughout the remainder of this section.

4.1.1 CRUD Operations

4.1.1.1 Create a Record

Record creation is an important use case feature. The users are likely to make new contacts when out on the road. The SuiteCRM application allows records to be created on the move and updated to the centralised SuiteCRM system. The creation tool is relatively limited, but allows for basic information that could be found on a business card to be entered. Being able to add records to the CRM on the go ensures that the SuiteCRM system is as up-to-date as possible.

4.1.1.2 Viewing a Record

As a business use case, viewing a record is essentially the primary use case for the SuiteCRM mobile application. It allows the user to view information about their workflow or their clients. Being able to view this information on the go eliminates the need for a laptop or desktop connection whilst on the move. The main objective for this project was arguably to mobilise SuiteCRM data and enable users to access it more easily on the go.

4.1.1.3 Updating Record Information

The business environment is an ever changing environment and information is changing constantly. This idea alone states the use case for why the application allows the user to amend information on the move. Examples of why information may need to be updated are if a contact changes their email address or an account changes address. These are all changes that need to be recorded as soon as possible to prevent potential errors in communication.

4.1.1.4 Deleting a Record

In large organisations with many employees making many interactions with multiple clients, there are always instances of duplicate records or out dated records. The SuiteCRM application allows users to delete records on the move. This can enable fast amendments if accidental records have been entered or if a record needs to be removed from the system. However this feature maybe a bit too powerful to have on a touch screen device as important records could get deleted by accident. The feature was included to demonstrate the full CRUD cycle on a mobile device.

4.1.2 Search by Text

Searching by text is the more reliable alternative to the voice search feature. From a business use case, you couldn't have the voice search without the text search as it operates as a backup to the voice search if environmental conditions do not permit voice search. Not only is it necessary as backup, but not all users would want to search by voice. If in a public area or travelling on public transport, being limited to a voice search may make the user feel uncomfortable. Providing a typed search function allows the user to choose whichever method they prefer.

4.1.3 Voice App Navigation

The voice navigation allows users to make home screen selections based on voice commands given. The clear use case for this feature is if the user is required to multi-task while using the application. One example could be walking down a busy street or driving; the user needs to search for a record, but doesn't want to be distracted by looking down and typing on the keyboard. With the touch of a single button, the user is able to either make home screen selections or search for a contact or account.

4.1.4 Native Application Integration

4.1.4.1 View a Meeting on Google Maps

On the go end-users are usually travelling to multiple locations throughout the day, meaning they spend a lot of time following satellite navigation or inputting information to a satellite navigation device. With an android device with GPS capabilities users are able to use their smartphone as the satellite navigation device. To cut a step out of the process of opening Google Maps and typing in the address, users of the SuiteCRM mobile application will be able to either select the Google Maps button on the AccountResult activity or click the compass button. The two buttons do similar but different jobs. The Google Maps button shows the record location on google maps by initiating a new intent to open Google Maps and passes the record address into the search parameters. Whereas the compass button passes the same parameters through a similar Google Maps intent, but places the words "navigate to" at the beginning of the search query. Using "navigate to" boots up the Google Maps application in navigation mode that uses the devices GPS to start the navigation from the location of the users device.

4.1.4.2 Calling a Contact from a Record

Both the Meetings and Contacts records allow the user to call the relevant telephone number straight from the application. This feature bypasses the user having to copy and paste a number, or write it down and place it in the dialler. By clicking the telephone button on the AccountResult page the devices dialler will open with the number of the contact pre-loaded in the input field.

4.1.4.3 Send an Email from AccountResult Page

The ability to send emails from the SuiteCRM application allows it to start acting as a one size fits all in the field SuiteCRM solution. The envelope button on the AccountResult page allows the user to use any email or messaging client they chose by opening a selector of all available mail clients. The intent used to move to a mail client also allows information to be sent to the appropriate application. In the prototype application that has been developed as part of this project, the subject line and main body of the email were populated with test data to demonstrate options available to users. In the finished product users would be able to customise what the default message and subject is if they wanted a default option. The use case for this would be if the user was running late to a meeting they would quickly be able to email them with a standardised message rather than wasting time trying to create an email.

5 Design

As per the requirements the application needs to perform certain tasks, but before they could be implemented the use cases need to be scrutinised and a method drawn up for how each use case would be executed. This section of the paper aims to outline the approach taken to how each component of the application would interact with one another and how they will be executed in the final product. This part of the paper also aims to provide a good idea as to how each part will work. For a general overview of the application architecture see Figure 9.

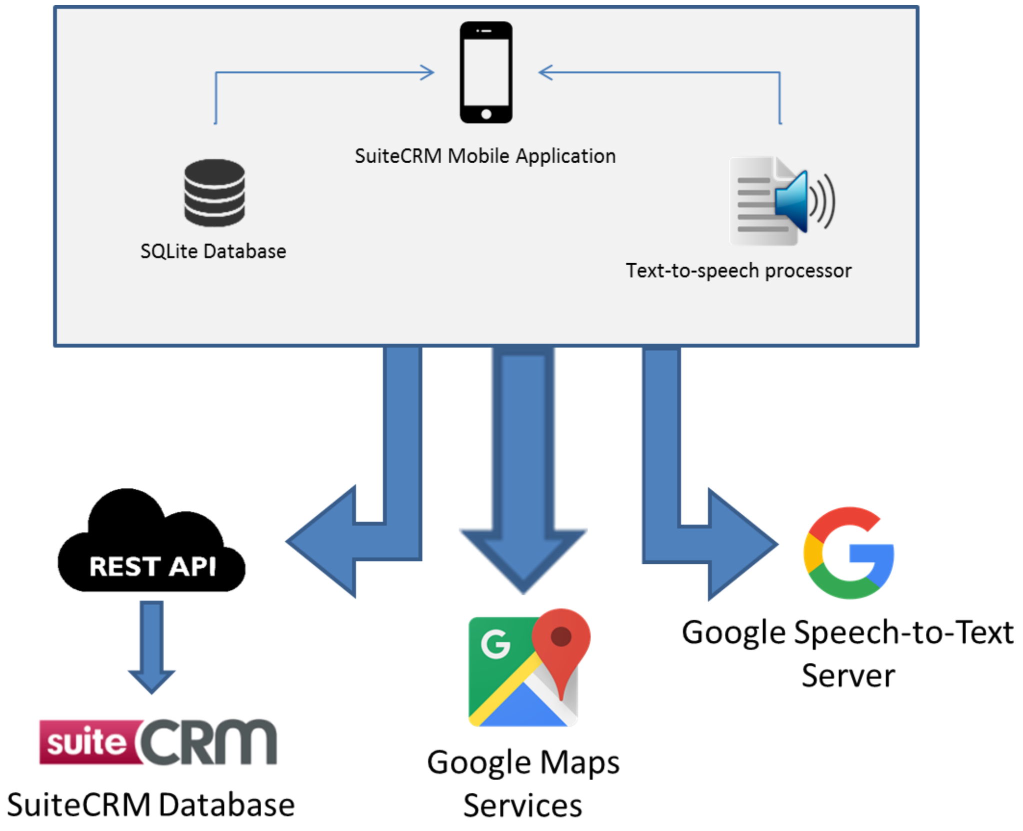


Figure 9 – Overview of SuiteCRM mobile application system

5.1 GUI and UX Design

GUI and user experience design are an important feature in application development. The user never gets to see what is under the GUI, so as far as they are aware what they see represents everything the application is capable of. It is important the GUI tells the user everything they need to know about the application and that it allows them to fully utilise its features. With the SuiteCRM application many of the design choices for the GUI were based on the QuickCRM application (2.2.3) and a fundamental understanding of the eight golden rules outlined by Ben Schneiderman [19].

Schneiderman outlined the following ‘Eight Golden Rules’

- Strive for consistency.
- Enable frequent users to use shortcuts.
- Offer informative feedback.
- Design dialogue to yield closure.
- Offer simple error handling.
- Permit easy reversal of actions.
- Support internal locus of control.
- Reduce short-term memory load.

Particular components of the SuiteCRM application GUI represent these rules better than others, but an attempt to follow these guidelines can be seen in the following sections.

5.1.1 Strive for Consistency

The theme used for the application GUI was an attempt to imitate the aesthetic of the web application. The use of rectangular boxes that matched the colour scheme was the most direct attempt to maintain consistency across the platforms. Not only on an aesthetic level, but by utilising the same module names and displaying them on the home screen such as in Figure 12 was an attempt to make the two different platforms as homogeneous as possible in how they are presented to the user.

5.1.2 Enable Frequent Users to Use Shortcuts

The entire inclusion of the speech-to-text functionality is to enable users, frequent or otherwise to perform actions in a more autonomous and expedited fashion. Once a user increases their efficacy with the application, voice navigation will become a more fluid experience in comparison to a user that has infrequent use or experience with the apps speech search functionality. Enabling users to become more fluent with this feature will present many shortcut like opportunities for users. Especially as the speech-to-text functionality improves.

5.1.3 Offer Informative Feedback

It is important that users understand how their actions will be reflected in the application. Usually the user will receive feedback in the form of their successful request providing the desired result. It is equally as important however to communicate to the user when something hasn’t gone right either. Toasts are used in this project to indicate to the user that their request is not successful; however this is not enough on its own. If the user is attempting to request new data from the CRM database and no signal is available to the device, the user is made aware of the lack of signal via a Toast and the offline information is made available. Using the toasts means that the user is constantly aware of what is going on or why particular things didn’t work.

5.1.4 Design Dialogue to Yield Closure

This refers to the creation of a journey for the user and the establishment of a beginning, middle and end for each use case. The SuiteCRM application strives to achieve this by creating a similar path for each use case. By doing this the user knows where they are at all times and how many steps they need to make to perform the next action. By having a defined path structure within the application users can feel confident when navigating the application.

5.1.5 Offer Simple Error Handling

The SuiteCRM application has been designed to be a relatively safe environment for the user. There are very few avenues that the user can follow to commit irreversible changes. This means that users can be confident navigating the application knowing that they will not cause any system wide damage. One design feature of SuiteCRM is the delete capability. The user is currently able to delete a record with a single click of a button which may lead to accidental deletions, however this action is reversible as the deletion only marks the record as deleted and does not delete any valuable information.

5.1.6 Permit Easy Reversal of Actions

The application allows users to rectify errors by enabling the user to edit and update changes on each record. This means that if information is entered to a contact record incorrectly then it can be amended swiftly and easily via the edit and update buttons on the AccountResult page.

5.1.7 Support Internal Locus of Control

At no point should the user feel out of control. The SuiteCRM application has been designed with this in mind and the navigation of the application has been built so that outcomes of actions are predictable.

5.1.8 Reduce Short-term Memory Load

Regarding the design of the voice-to-text functionality the user only needs to remember the keywords. This is designed with regular users in mind; keywords are established based on activities within the application. For example if a user wants to find a particular record they would say “search” followed by their specific search request.

5.2 UI Mock-ups and Prototyping

This section contains a mock-up of the UI that was created in the early stages of development. The mock-up in Figure 10 looks very different to the final product and this reflects the changing nature of the requirements and what was expected by the client.

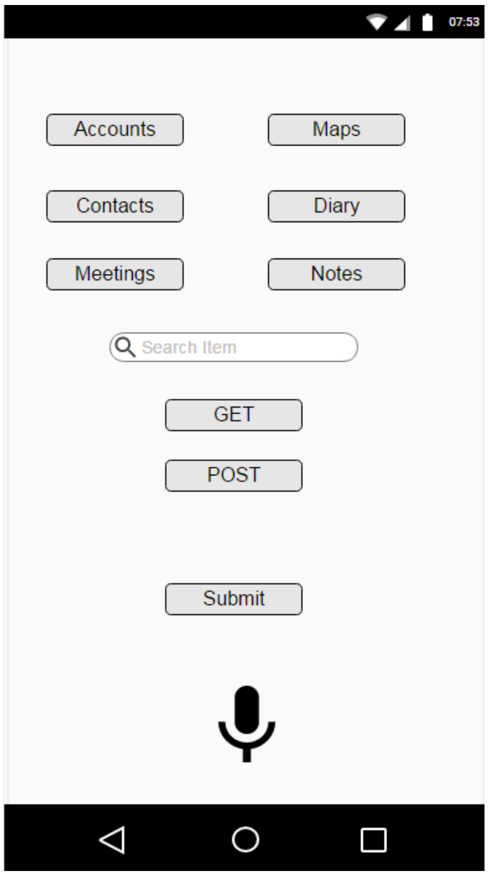


Figure 10 - Initial mock-up made in a photo editor

5.3 Offline Support with SQLite

The application utilises a SQLite database in order to store data that can be used when the device is not an area covered by data signal. The SQLite database has a very simple design that consists of two fields called type and data that are both of type String. This design allows for easy visualisation of the data and prevents complications. The role of the SQLite database is to store the objects that get retrieved each time the relevant module button is clicked. That means that when the Accounts button is clicked and an Accounts object is returned by the CRM, the entire Accounts object overwrites the contents of the data column where the type is Accounts. This means that the offline SQLite database is full of the most recent information that the user has accessed. A sequence diagram for the interaction can be seen in Figure 11.

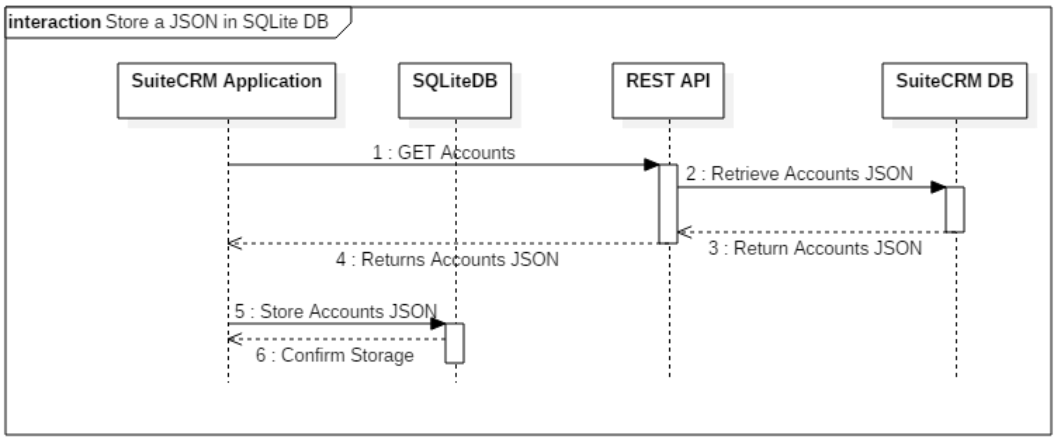


Figure 11 - Interaction diagram showing how SQLite database is updated online

When the application is offline the system needs to be able to recognise that it is offline and retrieve the stored data from the SQLite database. When the device has no network connection instead of making a request to the REST API the application will run an SQL query on the SQLite database and send the result to the appropriate method to continue the running of the application as if it had retrieved a result from the REST API.

5.3.1 Database Design

The database used for offline storage in this application is primitive in its design and is only meant to store module names and the corresponding data that is returned from the SuiteCRM database. The database only consists of one table that has two fields; the type field contains the module names that are used within the application and the data field contains the module data that is returned from the SuiteCRM database and then stored as a String. This design is only fit for the purpose of the prototype application as this method would require too much storage if it needed to store all the records of the CRM. A solution that would be applicable to real usage would need to scale much better than the solution used in this project. Not only would the database need to have a better design, but it should enable data that is changed offline to be uploaded when the user comes back online.

5.3.1.1 Type & Data

The two fields in the table are designed to contain strings that are used to be processed when offline only. There is no way of cross referencing tables in the database and generating intelligent queries. The table only exists for the purpose of persistent storage when the user is in an area without sufficient mobile data coverage. The values that can be expected in the type field are the module titles such as Accounts, Contacts or Meetings and they each have a corresponding value in the data field. The data field contains the information of each record in the related module, this will be stored as a string that will be processed at run-time to provide the relevant information.

5.4 The Role of the REST API

The REST API is utilised as a server of information for the SuiteCRM database as can be seen in Figure 5. This allows the application to be independent of changes to SuiteCRM as the calls the application makes are only ever to the REST API and there is no direct communication between the Application and the SuiteCRM database.

As per the requirements the REST API's primary function within this project was to provide CRUD functionality to the application. For the search functionality the search method from the REST API was utilised also, this was done in order to prevent processing the data on the device. The decision to use the search function of the REST API was to save resources as far as user data is concerned. To conduct the search on the device, the user would need to request all of the user information and then process it; this would require downloading a lot of irrelevant data. By running the search through the API the search query is passed to the API where the data is sorted and only the relevant data is sent to the user's device. This was a consideration made when deciding on how to conduct the search.

The REST API was treated as a black-box in terms of its full functionality, but it was explored slightly in order to find functions that it had that could help support the application fulfil the specifications outlined in the requirements.

5.5 Speech-to-Text

Speech-to-text provided an opportunity to design a system that could process the user’s voice and turn it into a series of strings that when processed performed a particular task. In this project the full potential of this technology was not discovered, but a framework was built that could be expanded on in the future. The basic framework of how the speech-to-text worked is outlined in Section 2.2.1. This allows for future developments to be made to the keywords structure that currently exists. The keyword structure that has been built relies on the user saying a specific set of words in a specific order based on what they want to achieve. For example a user could press the microphone button and say “Search, SalesAgility” this will then run a search because the first word is search. The current settings dictate that the first word establishes the path and then the second word could be a command related the chosen path. In this example the keyword is search then SalesAgility is the search term. This could be amended in future to support more specific instructions such as “Search, Contact, Tom.” This will run a search on Contacts with the search term “Tom”. The limitation of the voice-to-text as a tool within the SuiteCRM is limited to the capabilities of the REST API, but in future specific functions would need to be written to support speech-to-text functionality. This is because it would need to account for different phrasing from less experienced users.

5.6 Managing Activities with Intents

Intent is a term used within android development that signifies the description of an action that will be performed. In the context of this project intents are used to manage the information that passes between activities. This can be anything from a JSON object that carries important data to be displayed on the next screen, to a location that needs to be viewed on google maps. Using intents to manage the data passed to different activities ensures that the application is logical in how it presents data to the user. Passing data between activities using intents also means that data can be maintained within the system and does not need to be refreshed via the network every-time the user switches activity. This semi-persistent storage makes the application more economical of the user’s data and prevents frequent use of mobile data.

5.7 Handling Exceptions

It is important that the user is aware of the state of the application whilst it is running. This is especially important when something goes wrong. When the user makes a selection and the outcome they anticipated does not occur they need to be made aware of what when wrong and potentially why. As outlined in Section 6.1.3 Toasts are used to keep the user informed of any error or failure to complete a task. Using Toasts is an informative way of providing the user with immediate feedback on the status of their request. However it is important the user is only notified if there has been an error, it is not worth notifying the user of the success if their desired outcome is visible on screen. It is useful to display feedback to the user of a success if they have made a change that is not visible on screen. Toasts are therefore displayed if a user successfully creates a record or deletes a record, these are both occasions where the user doesn’t see explicit feedback on the status of the action they attempted to perform.

6 Implementation

This chapter looks to portray how different elements of the SuiteCRM application were realised on a practical level. The structure of this chapter is such that it will act as a walkthrough showing the different features available in the application before going into more detail on each component and speaking about the individual implementation of particular features and methods. It is important during this chapter that a general overview is given first in order contextualise the implementation and provide insight into why things were done the way they were.

6.1 SuiteCRM Mobile Application Walkthrough

This section will act as a guide through performing the core functionalities of the application and will use screen shots as a guidance tool.

6.1.1 Home Screen

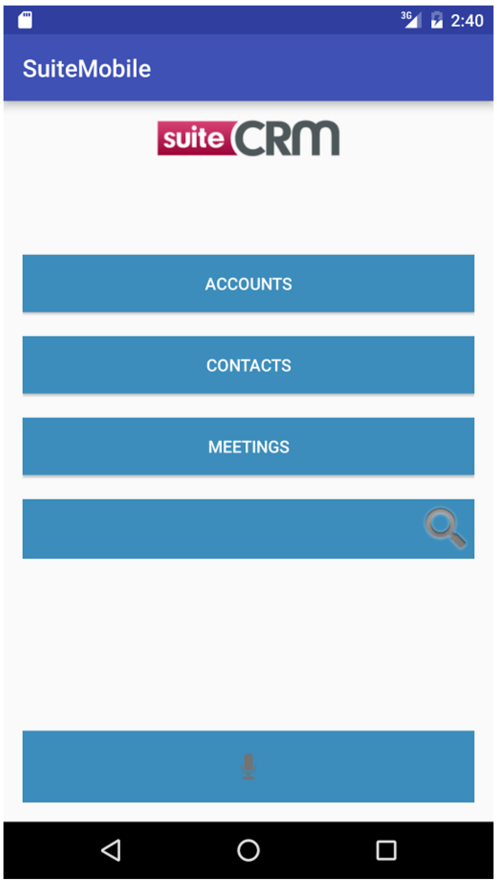


Figure 12 - Home Screen of SuiteCRM Mobile Application

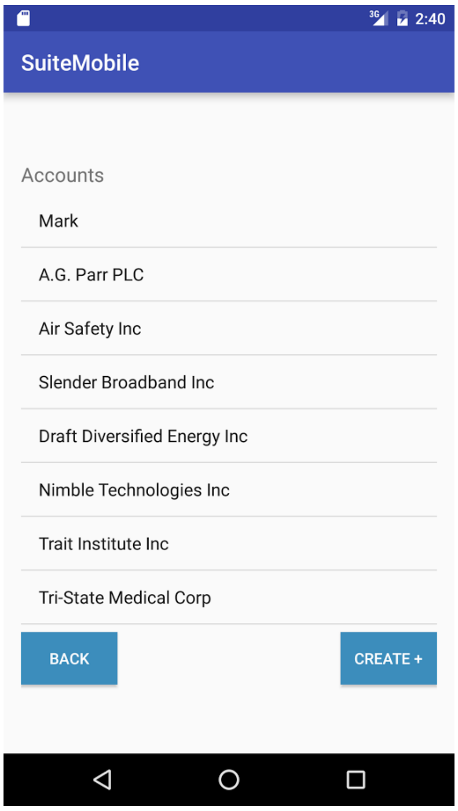
The screenshot in Figure 12 shows the screen that greets users when they open the application. This screen is basic and hosts five buttons that enable the user to interact with the application. There are the module buttons:

- Accounts - Takes the user to a ListView where all the Accounts are viewable.
- Contacts - Takes the user to a ListView where all the Contacts are viewable.
- Meetings - Takes the user to a ListView where all the Meetings are viewable.

The screen also host an EditText that allows the user to type in the search query they wish to search and a magnifying glass icon that acts as a button to confirm the search.

The final button is the microphone bar at the bottom of the screen that allows the user to initiate a voice command.

6.1.2 ListView



Accounts

Mark

A.G. Parr PLC

Air Safety Inc

Slender Broadband Inc

Draft Diversified Energy Inc

Nimble Technologies Inc

Trait Institute Inc

Tri-State Medical Corp

Figure 13 - ListView displaying Results

The ListView is where users can view the records stored in the module they selected on the home screen. This page consists of a ScrollView containing a ListView as well as two buttons for navigating to other areas of the application. If a user selects an item from the ListView they are then taken to the result page where the data from their selection is displayed.

6.1.3 Create a Record

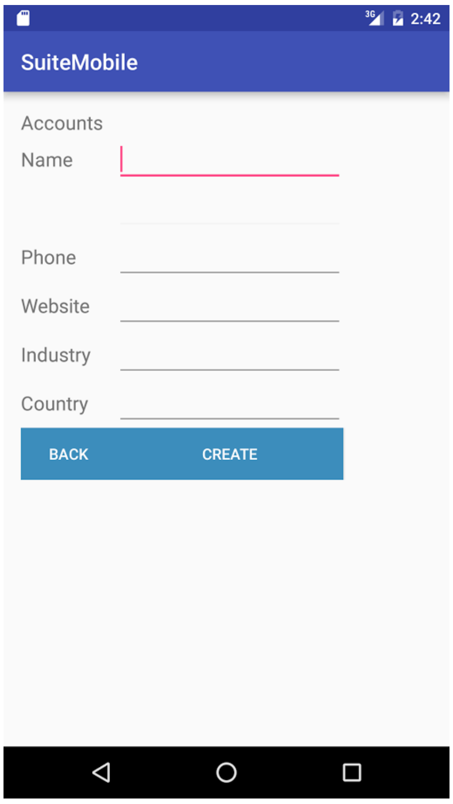


Figure 14 - Create record page

The create record page allows the user to fill in the appropriate fields and create a record. This page works by selecting the module name of the ListView page to dictate what the type of module being created is, and based on this information the information fields can be labelled appropriately. The Create button when selected takes the user back to the ListView and displays a toast confirming the creation of the record. Confirmation is given when the application receives a 200 response code from the SuiteCRM server confirming that the POST request was successful. If the Creation is not possible a toast is displayed saying that creation couldn’t be completed and to try again. The back button just returns the user to the ListView screen.

6.1.4 Search by Text

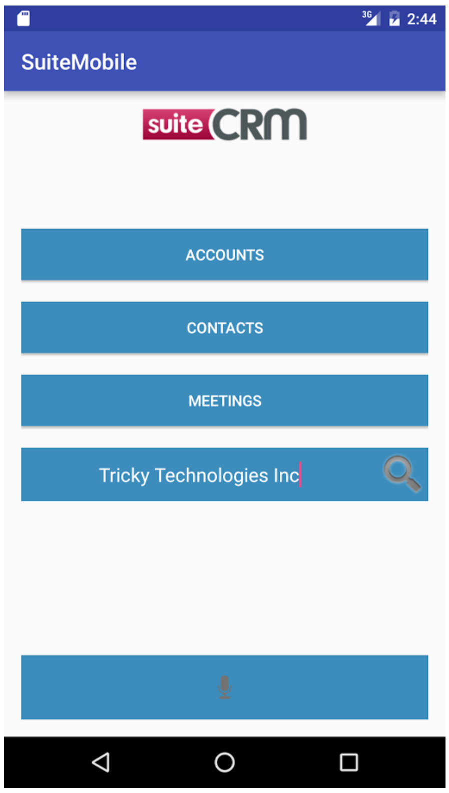


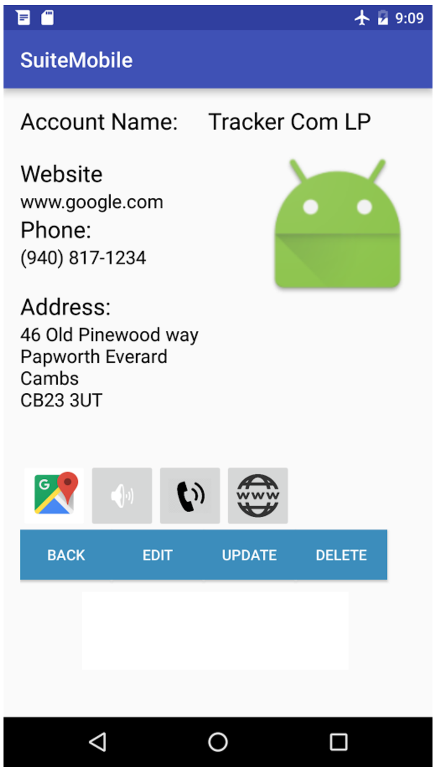
Figure 15 - Home screen showing the text search box populated

Search by text is located on the home screen and allows users to search SuiteCRM for any Accounts or contracts stored in the database. The reason this is located on the home screen is to provide a reliable and accessible alternative to the voice search that is also located on the home screen. Using the search by text function will take the user to the ListView page with the search results being displayed in a list and ordered by type (Account or Contact). The magnifying glass was used as it has become somewhat of a universal search symbol across mobile technology.

6.1.5 Navigating the Result Page

The results page offers the user more selections and options than any other page in the application. The page enables the user to modify and delete data as well as navigate to other applications (or places). The data that is displayed on the result page is dependent on the module it is representing. In the screenshot in Figure 16 it is possible to see that the account name, website, telephone number and address are the four fields displayed. This information can be edited and committed to the server via the edit and save buttons at the bottom of the screen. The record can also be marked for deletion with the delete button.

Some of the functionality on this page takes the user away from the application and these buttons are displayed as images. The Google Maps button on the left of the display will take the user to Google Maps and display the accounts address on the map. The phone button to the right of the Google Maps button will take the user to the dialler where the phone number of the account will be inserted and ready to call. The envelope button opens a list of local mail clients for the user to choose and will populate the subject and body of the message with a small message. This demonstrates that it could be used as an on the go tool for contacting customers. The www button on the end of the row enables the user to view the customer's website or the website that is entered into the website field on the application. This page offers limited functionality, but allows the users to perform all necessary operations on the client's data.


9:09
SuiteMobile

Account Name: Tracker Com LP

Website
www.google.com
Phone:
(940) 817-1234

Address:
46 Old Pinewood way
Papworth Everard
Cambs
CB23 3UT

Figure 16 -AccountResult page showing an accounts information

6.2 Creating the GUI

The GUI of the application was developed using Android Studio's drag and drop GUI editor (Figure 17). This tool is exceptionally useful for the rapid development of GUIs as it automatically generates the XML that represents the GUI created in the editor. This tool was used to create every activity that can be seen in the application as it is a relatively trouble free way of building attractive GUIs. This tool also allows the developer to control the parameters of each component on the screen. This is useful as it means that the developer does not need to be particularly proficient at XML.

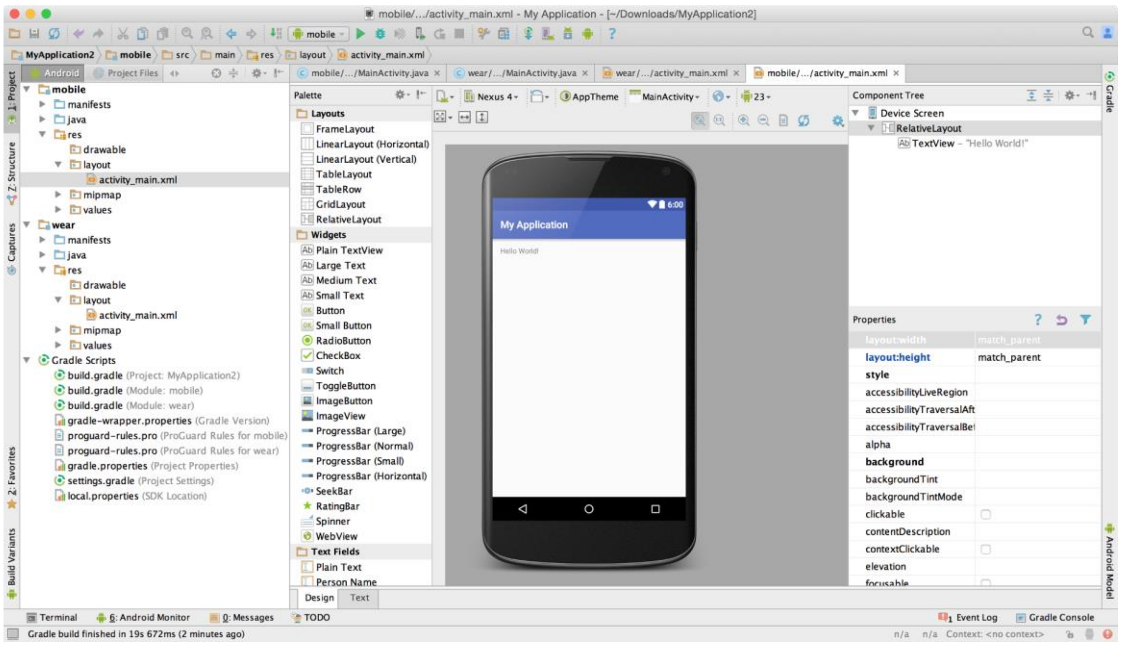


Figure 17 - Android Studio's drag and drop GUI editor

Each layout is stored in an XML file and is loaded at the beginning of the onCreate method in each activity. This allows each activity to be customised quickly and easily if needed. In this project each activity was associated with a layout, the AccountResult activity appears differently depending on the module that is selected. This is done by modifying the status of the components that are visible on the page. For example, in AccountResult the buttons are displayed differently for different modules; using the setVisible method for the ImageButtons it is possible to pick and choose which buttons are visible to the user.

6.1 Implementing Speech-to-Text

This projects initial intention was to create an application that utilised speech to text to navigate the application and perform tasks dictated by the user. The final implementation managed to capture the essential element of this and was capable of translating user's speech into a set of tasks that performed the desired action. This was achieved using android.speech; android.speech is androids built in speech-to-text API. Figure 6 in section 2.7.1 shows how the API works by sending the voice data to a Google server where it is then processed into a string that is then returned.

6.1.1 Processing User Speech

To implement this feature a method has been created called startSpeechToText() this method initiates RecognizerIntent the class that controls speech input. As seen in Figure 18 putExtra is used to set the parameters for the intent. startActivityForResult is then called and the intent is placed in alongside a SPEECH_RECOGNITION_CODE so that the result can be paired with the request later on. At this point a dialogue box appears and prompts the user to speak (Figure 19).

```
 private void startSpeechToText() {
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
               RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT,"Say \"search\" followed by your
request.");
```

Figure 18 - startSpeechToText() method for setting up speech-to-text intent.

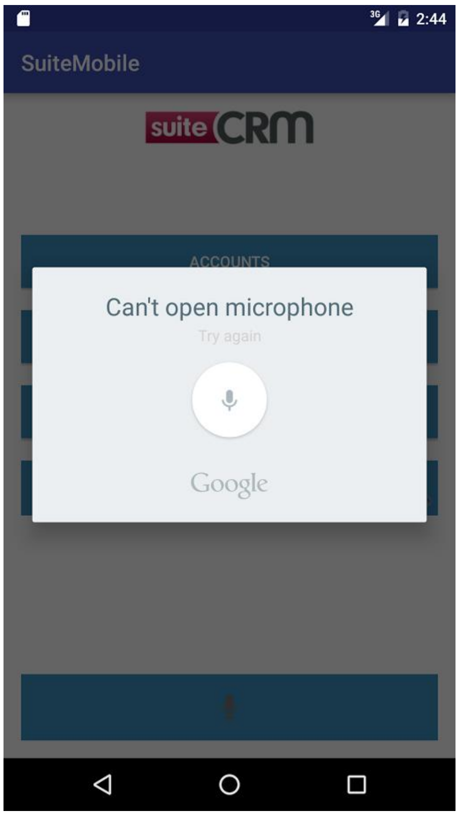


Figure 19 - Screenshot of the dialogue box for the speech-to-text intent

The startActivityForResult() method invokes the onActivityResult() method that uses a switch statement to check if the requestCode matches the SPEECH_RECOGNITION_CODE, if this is the case and the conditions of the if statement are met, the content of data (the text returned from the server) is stored in an ArrayList containing 1 string at index 0. At this point a new String is created called text; this String contains the data stored in the first index of the result ArrayList (Figure 20).

```
switch (requestCode) {
    case SPEECH_RECOGNITION_CODE: {
        if (resultCode == RESULT_OK && null != data) {
            ArrayList<String> result = data
                .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            String text = result.get(0);
```

Figure 20 - Switch statement used to build String from result of processed speech

6.1.2 Processing the Result

Once text is initialised the String can be used in any way to perform tasks. At this point there are many different options for processing the text to perform actions; however the method chosen reflected the style that was wanted for issuing commands to the application.

The command() method uses an array of keywords to dictate what action will be performed. As can be seen in Figure 21, the keywords supported are the names of the modules on the home screen and search. In this prototype, search is the term in the command method that processes the data further, but each keyword could be built on to create a host of different outcomes based on commands given. This method also makes it easy to create new commands. As can be seen in Figure 21 the command method works by identifying the first word to choose which path it will take and then allows further processing at that point. In the context of search the application loops through each index in the array and places a space in between them to build the searchTerm which is used as the query String in the search.

```
private void command(String words){
      String[] s = decompose(words);
      String searchTerm = "";
      AsyncRestRequest restcall = new AsyncRestRequest();
      SearchRequested searchRequested = new SearchRequested();
      if (s[0].equals("accounts")){
              restcall.execute("Accounts");
      }
      else if (s[0].equals("contacts")){
          restcall.execute("Contacts");
      }
      else if (s[0].equals("meetings")){
          restcall.execute("Meetings");
      }
      else if (s[0].equals("search")){
          for (int i = 1; i<s.length; i++){
              searchTerm += s[i] + " ";
          }
          searchTerm = searchTerm.trim();
          searchRequested.execute(searchTerm);
          String did = "";
      }
  }
```

Figure 21 - Code snippet showing the command method

6.2 Async Task

AsyncTask allowed the application to utilise another thread to perform the HTTP calls and communicate with the REST API. The benefit of having this was to leave the UI (User Interface) thread free so the application would remain responsive. If AsyncTask was not utilised, the application would be non-responsive and users may think that the application had crashed.

AsyncTask is an android class that consists of four methods that are essential to its structure; they are:

- doInBackground()
- onPostExecute()
- onPreExecute()
- onProgressUpdate()

```
private class SearchRequested extends AsyncTask<String,String,String> {

        @Override
        protected String doInBackground(String... params) {
            return "Search Results";
        }
        @Override
        protected void onPostExecute(String result) {
        }
        @Override
        protected void onPreExecute() {
        }
        @Override
        protected void onProgressUpdate(String... text) {
        }
    }
```

Figure 22 - Shell of AsyncTask class without internal implementation

AsyncTask is able to control how each of its methods operates so that the developer does not need to worry about handling threads. The class is also capable of calling the correct methods in the correct order. In Figure 22 the construction of the AsyncTask can be seen, it is worth noting that the class being used is extended by AsyncTask to inherit its functionality.

6.2.1 doInBackground()

This method is responsible for carrying out the desired task, in the context of the SuiteCRM application, this method is used to make calls to the REST API. This method handles the connection in its entirety and processes the information that is returned from the connection.

6.2.2 onPostExecute()

This method is important as it is what sends the information back to the UI thread. Methods from the main thread can be called in this method and the results are implemented in the main thread.

6.2.3 onPreExecute()

In this application this method is never used. Its role is to provide some context for the doInBackground() method. Any information that is essential to the completion of the AsyncTask should be declared in this method.

6.2.4 onProgressUpdate()

This method was never utilised in the development of this application. This method can send information to the main thread that can be used to provide users with progress reports on the background activity that is being run.

6.3 Implementing the CRUD Cycle

CRUD standing for create, read, update and delete covers the core functions of the application. It is important as these four operation cover almost all eventualities for how a user may wish to utilise the data stored in the SuiteCRM database. Create in the context of the SuiteCRM application means to make a new record to be stored in the database. Read simply means to view a record and perform no action other than viewing it. The role of update is to amend a previous record and delete serves to remove the data. Each of these actions is supported by HTTP verbs that are used to contextualise the requests to the REST API. The HTTP requests used in the SuiteCRM application are GET, POST and PUT. Delete is not used as the method for deletion in SuiteCRM is to mark an object for deletion via the PUT verb. The role of these verbs is detailed further in the remainder of this section.

6.3.1 HTTP GET

HTTP GET is the verb used to “request data from a specified resource”[*] in the context of this application GET is used to retrieve record information from the specified URL that is contained within the request.

6.3.2 HTTP POST

HTTP POST is the opposite to the GET method that instead of requesting data it submits data to be stored at a specific address. In the SuiteCRM application this verb is used to support the create function where records are created. Within the SuiteCRM system using POST to create an Account record for example; will result in the generation of a new ID. This is how the system manages the creation of new records.

6.3.3 HTTP PUT

The PUT verb is used to amend existing records on the SuiteCRM system. The PUT verb essentially overwrites the information stored in the location you are sending it to. In the SuiteCRM application the PUT verb is used to amend information stored in a record or to delete the record. The delete function works by updating the ‘deleted’ field in the records data

JSON array stored in the returned JSON object. From that point on the object or record deleted will still exist, however it will be stored with all the records where the ‘deleted’ value is equal to 1.

6.4 The Role of Intents

Intents are a construct of android programming that allows the developer to open new activities whilst passing information between activities. In the context of this application intents are behind all the activities. The speech-to-text function launches an activity to process the speech; switching between activities is done via intents and intents are used to switch between applications such as Google maps and the dialler. Intents allow the application to be truly native in how they operate. Without intents it would not be possible to create a dynamic application that fluently switch between tasks.

6.4.1 Exchanging Data Using Intents

As seen already in this dissertation; intents are used to transfer useful information between activities. putExtra is the method used for attaching information that will be passed between intents. The example in Figure 23 is of the displayData() method that initiates a new activity and passes the relevant data through using the EXTRA_MESSAGE as the constant.

```
private void displayData(String title, String data){
        Intent dataDisplayIntent = new Intent(this,Meetings_Page.class);
        dataDisplayIntent.putExtra(EXTRA_MESSAGE, data);
        dataDisplayIntent.putExtra(EXTRA_MESSAGE1,title);
        dataDisplayIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(dataDisplayIntent);
    }
```

Figure 23 - Code snippet showing how putExtra is used to exchange data between activities.

6.5 Creating a Database with SQLite

SQLite was used to construct a table that would facilitate the use of the SuiteCRM application whilst offline. The table is created the first time the user runs the application, the table is built using the createDatabase() method (Figure 24). This method initialises a SQLite database called mydatabase that creates a database called OfflineStorage once the database has been created a table called OfflineJson is builtr with the column headings Type and Json. The two column titles reflect the content that is to be stored in the corresponding cells below. Type signifies the module name and the Json column will store the JSON objects that are returned when the relevant button is pushed during the applications usage. An example of the OfflineJson table can be seen in Figure 25. The createDatabase() method is only executed however if there is no instance of the OfflineStorage database. If this database exists the current database continues to be used.

```
public void createDatabase(){
        SQLiteDatabase mydatabase = openOrCreateDatabase("OfflineStorage", MODE_PRIVATE,
null);
    try {
  mydatabase.execSQL("CREATE TABLE IF NOT EXISTS OfflineJson(Type VARCHAR,Json VARCHAR);");
        mydatabase.execSQL("INSERT INTO OfflineJson VALUES('Accounts','[]');");
        mydatabase.execSQL("INSERT INTO OfflineJson VALUES('Contacts','[]');");
        mydatabase.execSQL("INSERT INTO OfflineJson VALUES('Meetings','[]');");
    }catch (Exception e){
        System.out.println(e);
    }
}
```

Figure 24 - Code snippet of the createDatabase() method used to initialise the SQLite database when the user first opens the application.

OfflineJson	
Type	**Json (Containing JSON objects)**
Accounts	Id:,Contact,Telephone,Website
Contacts	Name,Telephone,Mobile,Address
Meetings	Date&Time,Location,Description,Contact

Figure 25 - Table representing the structure of OfflineJson table

6.5.1 Updating the SQLite Database

The purpose of having the SQLite database is to ensure that the user can still access recent information when there is no internet connection. This requires keeping the SQLite database up-to-date so that old and potentially incorrect information isn't displayed. This is done during the asynchronous GET request that is run when one of the home screen buttons has been pressed. As an example, if the Accounts button is pressed the system will make a GET request from the SuiteCRM database for all the records under the accounts module. These will be returned in a large JSON object that is then stored via the updateDatabase() method in the Json column where the type is equal to Accounts. The exact implementation can be seen in Figure 26.

```
public void updateDatabase(String module, String json) {
        SQLiteDatabase mydatabase = openOrCreateDatabase("OfflineStorage",
android.content.Context.MODE_PRIVATE, null);
        String query = "UPDATE OfflineJson SET Json = '"+json+"' where type =
'"+module+"';";
        mydatabase.execSQL(query);
        mydatabase.close();
    }
```

Figure 26 - Code snippet updateDatabase() method

6.6 Utilising Google Maps

The use of 3rd party applications such as google maps helped to make the application seem even more native to the android platform. This was done using intents as described earlier. Google provide an easy to use API that makes sending information to Google maps a simple process that only involves a couple of lines of code as can be seen in Figure 27. The results make for a seamless integration as can be seen in the screen shot in Figure 28.

```
    private void onMap(String location){
        Uri mapsIntentUri = Uri.parse("geo:0,0?q=".concat(location));
        Intent mapIntent = new Intent(Intent.ACTION_VIEW, mapsIntentUri);
        startActivity(mapIntent);
    }
```

Figure 27 - Code snippet Google Maps intent

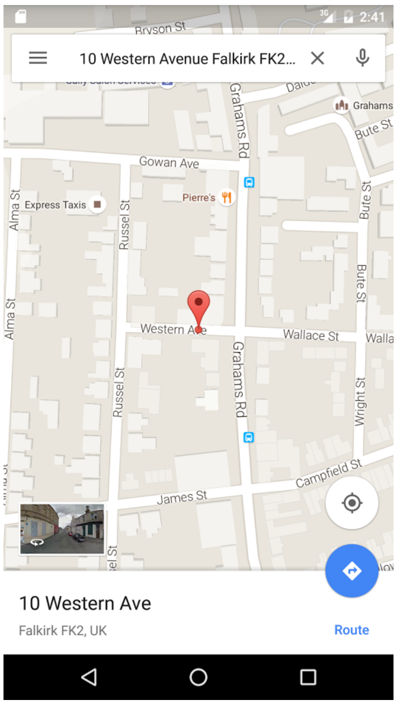


Figure 28 -Google Maps search after onMap() method is called

6.7 Text-to-Speech

Text to speech was not a requirement for the system, but was a feature that was added as it seemed to provide a tangible benefit at very little cost. The android library android.tts was used to read the content of the EditText located in the AccountResult page. The code showing the implementation of this feature is shown below in Figure 25.

```
public void onClick (View v){
                    ttobj.speak(body,TextToSpeech.QUEUE_FLUSH,null,"utteranceId");
            }
```

Figure 29 – Text-to-Speech implementation

This feature allows users that cannot afford to be focussing on the screen to hear playback of the information stored in the notes field of a meeting or the address field of an accounts/contacts page. This feature could be expanded as an accessibility tool to enable users that have sight impairments to use the software. Text-to-speech also has applications in aiding the speech-to-text process. This could be implemented by reading options to the user then prompting a response.

6.8 Working with the Device

Programming in android presents its own challenges that required understanding before all the implementations could take place. This section aims to briefly talk about different android specific nuances that sparked intrigue during development.

6.8.1 Android Activity Stack

The android activity stack or back stack is the way that activities are managed within android. When intent is used to start another activity the activity that has been called replaces the current activity and sits on top of it. This means that if the back button was pressed the previous activity would be restored. If the back button was pressed again the new activity would not be called and this is because of the way that android stacks its activities in a first in last out way as can be seen in Figure 26.

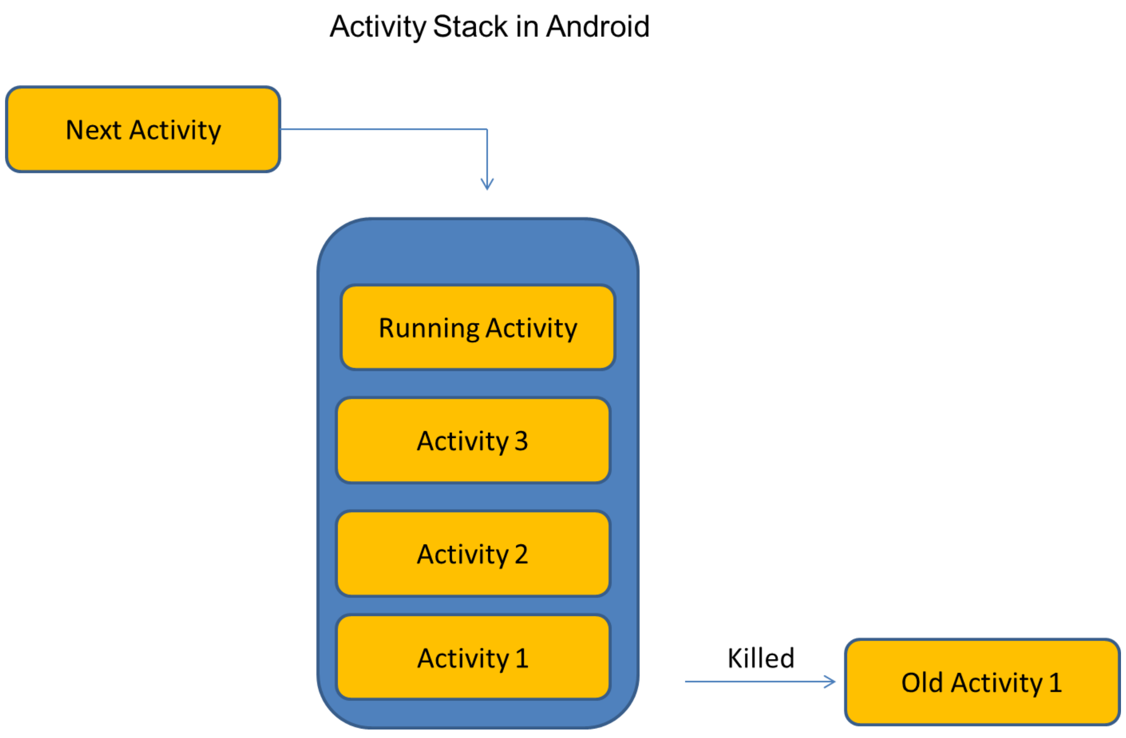


Figure 30 – Android activity stack showing how activities are managed

This was a critical consideration when the navigation of the application was being designed. When deleting a record in the application it returns the user to the ListView from the AccountResult activity. This couldn't be done via a simple termination of the current activity as it would lead the user to the previous activity where the deleted result still exists. To implement this correctly intent constant FLAG_ACTIVITY_CLEAR_TOP was used to remove previous instances of this activity and everything in-between it and the current activity. Therefore when the back button is click in the new activity the natural activity of the home screen is reached when back is pushed.

7 Testing

Testing can be used as a method of benchmarking how an application performs against the expectations of the developer and the client. In the context of this project, testing was conducted to see if the application was as functional from a user perspective. As the development team consisted of one developer and two supervisors, it was difficult to assess the functionality of the application as the development team was too close to the project and knew how it worked. Testing provided the opportunity to gain a fresh perspective on how the application was perceived by users. Testing also allows for any bugs to be discovered and addressed.

7.1 Developmental Testing

Developmental testing was conducted by the developer during implementation of the solution. This consisted of unit testing individual methods, integration testing of a group of methods or classes and system testing when the application was being tested for bugs whilst all the components were together. Compatibility testing was also conducted due to the fact that many android devices are different and many devices have different screen sizes that will give the GUI a different look. This testing was conducted to identify consistency across different devices.

7.1.1 Unit Testing

Unit testing is responsible for testing each independent component of the application and this was conducted during implementation of the SuiteCRM mobile application. This type of testing was conducted for each method or feature added to the application. This ensured that each piece was working and therefore meant that all the methods would work together when put together. The benefit of this testing is that it can be done without any external help and can just be adopted into the development process.

7.1.2 Integration Testing

This type of testing is the natural progression of unit testing and is involves testing all of the individual units that make up certain components of the application. In the context of the SuiteCRM mobile application an example could be ensuring that the speech-to-text processing worked and that an interaction with the REST API could be automatically managed. Integration testing was conducted anytime two or more elements from the application had to work together or communicate. It was commonplace throughout the project to encounter bugs at this testing point as different components were not always automatically set up. One such example was using the debugger in PHP Storm to ensure that the application was hitting the right points in the API.

7.1.3 System Testing

System testing was the testing of the whole application and this is where the questionnaire and user testing came in. Before the user tests could be done it was important that the developer checked the system and attempted as many use cases as possible in as many combinations conceivable. This was done to detect any bugs that may be present during the user testing.

7.1.4 Compatibility Testing

Android is deployed on over 1.4 billion devices worldwide [25] and many of them run on different hardware and altered versions of android. This means that there is plethora of different android combinations. As the end users will likely be using a variety of different devices it is important that as many devices as possible are tested in order to be sure that it will work for as many users as possible.

The compatibility testing conducted as part of this project was done on four different devices and was only assessing the impact on the GUI of using different devices. The devices used were a 7-inch Nexus tablet, a 5.2-inch Nexus 5X smartphone, an HTC (One) M8 with a 5.2 inch display and an Alcatel OneTouch Pixi 3 with a 3.2-inch display. Screenshots can be seen of the home screen on each device in Appendix 2. In general the application responded well to the different devices, however the Alcatel was running (KitKat) an older version of android than the SuiteCRM application was developed for and this resulted in a few crashes and features that didn't work. The application was developed for Marshmallow (the most recent android version) so failure on old models was to be expected. The aim of testing on this device was to see how the GUI responded to the small screen-size.

7.2 User Testing

7.2.1 Sample

The sample used was a mixture of 13 individuals from the SalesAgility office. The sample consisted of technical (software developers) and non-technical (sales) people as to get a good all-round impression. Test subjects had not previously used the application nor were they involved in the development in any way. This was important as having experience of how the application worked would skew perceptions of the application. All the users were familiar with SuiteCRM as they used it in their day-to-day roles. This was seen as a balanced and reliable testing group as real world users would also be familiar with the SuiteCRM system. Test subjects were also asked some qualifying questions to assess the efficacy levels with mobile technology (more details can be seen in Appendix 1).

7.2.2 Questionnaire & User Testing

User Testing was administered via supervised use of the application that included questions and tasks for the respondents to complete. The questions were conducted in a questionnaire style and filled out on behalf of the respondents. This was done so that respondents could focus on using the application and not tracking down their answers. The questioning was separated into four separate stages each with a distinct objective. A copy of the questionnaire can be seen in Appendix 1.

The first stage was qualifying questions as mentioned in section 7.1. This part of the testing was to gain insight into the people taking the tests, this information was useful for understanding how different users interacted with the application and how factors such as regional accents affected the outcome of the speech recognition function.

The second stage of the testing was observing how the users interacted with the application when being asked to perform some pre-defined use cases as can be seen in Section A of Appendix 1. This section of the testing was included to assess how the GUI of the application worked in providing visual cues to assist the user as well as core functionality. Help was only provided in a few cases where users were stuck as what to do. This part of the testing also assessed how the application functioned and allowed the tester to assess flaws in the application.

The third stage of testing and the fourth were related in terms of gaining user feedback. The third section focussed on quantitative data that assessed aspects of the application based on user experience. The fourth section however focussed on qualitative data that asked the test subject to provide more detailed answers as to what they did and did not like about the SuiteCRM application and what they would change about the application. These sections provided the best feedback in terms of how the project could be improved in the future and where basic errors were made in development.

7.2.3 Usability Testing

The closest formal usability testing strategy adopted was hallway testing in that the participants were not related to the project in any way other than working for SalesAgility. This testing allowed some bugs to be discovered. For example if a selection is made twice before the first selection is executed it will load two instances of the selection. This was not discovered during system testing as the developer did not think to be impatient and make multiple selections. This type of testing was good for uncovering small usability bugs such as this.

8 Conclusion

8.1 Summary

SalesAgility wanted to build a proof of concept, native android application that was a realistic on the go solution for SuiteCRM. The project had to implement speech-to-text in a way that end users could use the application to navigate and access data stored on the CRM. The application would also utilise their recently developed REST API that was developed to make SuiteCRM a truly RESTful system. The application needed to be on Android and work harmoniously with devices to create a seamlessly native experience. This was achieved by enabling features such as setting up emails from a local mail client or calling contacts direct from the application.

SuiteCRM has many moving parts and for one person to capture every element of it into a mobile application in a short period of time was a near impossible task. The approach that was adopted was to build a fragment of what could grow into a much larger application. This meant severely reducing the number of available features and modules to work with. Accounts, contacts and meetings were the three modules that were chosen as they represented a good base of use cases to work with from a user perspective. Implementing the solution was also not about scale, the fact that it was a proof of concept project as opposed to a fully realised solution is justification enough for not introducing more modules.

The voice-to-text solution provided by android for the solution was immensely useful and ready to go out of the box. The search for an appropriate voice-to-text processor displayed lots of open source solutions, but none that were as convenient or accurate as android.speech proved to be. The android.speech library enables a wide range of users with a variety of accents to use the device to search and navigate data in the application with limited need to repeat themselves. This solution made the most sense for this project as it was quick to implement and worked very well.

The final solution produced met the criteria established in the requirements and generated new scope to build on such as a fully realised offline system and the idea that the speech to text could be implemented in a more natural way by utilising the text-to-speech feature also. By meeting the requirements the project was in essence a success, but as the remainder of this chapter will go on to detail is that the project could have done things differently from an development perspective.

8.2 Evaluation

This section aims to critically evaluate the outcomes of the project and the processes that were undertaken in its fulfilment. The structure of this evaluation will be such that the process will be evaluated first and then the final product.

Development of the SuiteCRM mobile application started by looking at what components would be best for the job. The first component to be sourced was a speech-to-text processor. This component was seen as a critical element as it was thought that this would take the majority of time to implement. The search started by looking at many open source speech-to-text engines that involved a lot of heavy lifting in terms or programming the engine to recognise particular words. This method seemed over the top and retrospectively looking back on it would have been worthy of a project of its own. This meant that a lightweight quick to implement solution was needed. Androids built-in solution android.speech was selected as the obvious choice as it was a quick to implement library that was backed up by a tried and tested speech-to-text processor powered by Google. This concluded the search for a speech-to-text processor as it was the most reliable and elegant to implement as can be seen in chapter 6.

Another challenge was to understand how the REST API functioned. As this component was mandatory there were no alternative solutions to communicate with the SuiteCRM database. However getting to grips with how RESTful web services worked provided a learning opportunity. If a similar project were undertaken in the future it would be advisable to gain an understanding of RESTful web services prior to using them in a solution.

Android development was the key obstacle to overcome during this project. The developer was inexperienced with Android, but did have a fundamental grasp of key Java constructs such as class composition, reading and writing to a file as well as handling different data types in Java. Whilst these skills were a good platform, they were by no means the complete set of skills required to succeed within an Android environment. A stronger understanding of Android prior to the commencement of the project would have resulted in a project that was more connected in terms of the data available in the application. This would have been because more time could have been spent utilising the REST API rather than learning the fundamentals of Android programming.

Throughout development the SuiteCRM mobile application was tested through the Android Studio emulator and was accessing an instance of SuiteCRM that was located on the machines localhost, meaning that for much of the development and developmental testing the application was in sheltered conditions. A repercussion of this is that when the application was connecting to a real life instance of SuiteCRM via the internet there were delays in the transmission of information that had not existed previously. This led to extended waiting times between button clicks that relied on a connection to the centralised CRM. This raised some issues in user testing with impatient users accidentally making multiple selections and loading several instances of the same thing. If this project were to be conducted again development of the online functions should have been done in a sheltered but live environment as to reflect real life conditions and make the developer aware of any issues that could arise in this situation.

The SQLite database that stores records for offline use was a feature that was added on as a proof of concept. The idea that users that would be on the go and may need to access information when they are in areas that have limited mobile network cover was identified during the requirements process. However during this project the solution that was realised was enough for the purpose of a small scale dataset, but would not scale for much larger quantities of data. The solution used in this project was simply to provide the functionality and realise the concept. A more sophisticated database would need to be constructed in order to house large amounts of data or a process that was able to identify useful information and store it for users. A sophisticated solution would have detracted from the key objectives of the project which is why the solution used was implemented.

On balance SalesAgility were pleased to see how SuiteCRM translated to a mobile application and also saw how the idea could be developed further to transform their users experience with SuiteCRM. The initial scope of the project was quite large and the direction in which to take it was quite general, but as different features for the application were added the project took a very natural shape and the application had a very clear set of business use cases.

The developer was also able to learn a lot through the opportunities that the project provided. Previous experience the developer had received was very fragmented in terms of learning about different systems in silos and never really working in a harmonious environment. This project allowed the developer to work in an environment where PHP was being used in conjunction with JAVA, XML and SQL. The level of independence involved in this project also allowed problem solving skills to develop as the amount of assistance with minor to middling problems was limited.

8.3 Future Work

This section will discuss future work for this project in a compartmental way. First, the limitations to this project will be outlined and reasons for why some areas were not fully explored will be explained. Finally this section will discuss what a future solution should seek to achieve.

8.3.1 Limitations

The nature of this dissertation project is that there is a three month window to develop a project and write it up. In this time-frame the developer must also learn a significant number of skills that were unfamiliar. This means that throughout the process there was a tight schedule for each activity meaning that there is not always enough time to realise every idea that is manifested. Throughout this project the requirements evolved based on client demands. Whilst this reflects the real situation of changing customer requirements it proved to make development a dynamic environment that made planning difficult and budgeting time and managing expectations became significant challenges. If this project was given the time and resources that it needs, there is no doubt that a reliable and fully featured SuiteCRM application could have been built.

Another limitation to the project was the fact that everything had to be open source. SalesAgility develop in an (almost) entirely open source environment from the IDEs to the operating systems. This meant that paid technologies such as Google Maps were not available in their most embedded form. This is a small limitation as it does not affect the functional quality of the application; however it does detract from the idea of having an embedded and truly native application.

8.3.2 Next Steps

8.3.2.1 Module Support

A next step for the SuiteCRM application would be to increase the number of modules available and to enable the modules to interact. For example if the user was looking at a contact the user should be able to create a meeting with that contact. This will give the app more functionality and would then start to make the application more of an alternative to the SuiteCRM desktop version. Although the application is not intended to ever fully replace the SuiteCRM desktop version, the more functionality it adopts the more of a resource it is for its on-the-go-users.

8.3.2.2 Built in Speech-to-Text

A dedicated speech-to-text processor could also be developed for the SuiteCRM mobile solution as the android.speech library certainly has limitations in as much as the context of SuiteCRM data is not the same as natural language. The application got stuck a few times when a customer’s name wasn’t an English name or an accounts name contained a symbol (e.g. Black & Young). Building a purpose built speech-to-text library would allow the model to be more adaptive in the CRM context.

8.3.2.3 GUI Improvements

As can be seen in Appendix 3 mock-ups of the GUI were made to reflect how the application would look and fit in with the current SuiteCRM theme. These designs made in-house by SalesAgility show a more sophisticated layout that is more in line with the type of thing you see in enterprise quality software. The implementation of a new GUI is important as this is the part the user sees. User's perceptions of an application's aesthetics will reflect how they perceive the functionality to be also; therefore updating the GUI in the SuiteCRM application would be important to improving the user experience.

8.3.2.4 Full Offline Functionality

As it stands the offline feature is only useful for a small scale application or a user with a small number of clients. To make this application scalable, the offline support function would need to support potentially millions of records. This could be done by allowing users to commit records to their devices memory or by caching a certain number of records based on SuiteCRM search history. The natural evolution of this service would be to then allow the user to amend information while offline and those changes to be committed to the central SuiteCRM when the user come online.

8.3.3 Summary of Future Work

Due to the ever-growing base of use cases for SuiteCRM the future work of the SuiteCRM mobile application is a constantly evolving list of features that could support users. The development of this application as part of the dissertation project has encouraged SalesAgility to push forward with the development of a SuiteCRM application that will perform the core functionality of the SuiteCRM web application. This is evidence that this project has therefore inspired actual change within SalesAgility and prompted the next stage of development for the concept of a SuiteCRM mobile application.

References

[1] "What is customer relationship management (CRM) ? - Definition from WhatIs.com", *SearchCRM*, 2016. [Online]. Available: http://searchcrm.techtarget.com/definition/CRM [Accessed: 23- Aug- 2016].

[2] J. Mackin, *SuiteCRM for Developers*. Online: Leanpub, 2015, p. 4.

[3] "SuiteCRM", *Wikipedia*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/SuiteCRM [Accessed: 23- Aug- 2016].

[4] "MVC Architecture - Google Chrome", *Developer.chrome.com*, 2016. [Online]. Available: https://developer.chrome.com/apps/app_frameworks [Accessed: 23- Aug- 2016].

[5] "SugarCRM and SuiteCRM Mobile on iPhone and Android - QuickCRM - Addons for Sugar and SuiteCRM", *Quickcrm.fr*, 2016. [Online]. Available: http://www.quickcrm.fr/mobile/en/ [Accessed: 23- Aug- 2016].

[6] J. Mackin, *SuiteCRM for Developers*. Online: Leanpub, 2015, p. 46.

[7] "What is application program interface (API)? - Definition from WhatIs.com", *SearchExchange*, 2016. [Online]. Available: http://searchexchange.techtarget.com/definition/application-program-interface [Accessed: 23- Aug- 2016].

[8] J. Mackin, *SuiteCRM for Developers*. Online: Leanpub, 2015, p. 89.

[9] M. Rouse, "What is REST (representational state transfer)? - Definition from WhatIs.com", *SearchSOA*, 2014. [Online]. Available: http://searchsoa.techtarget.com/definition/REST [Accessed: 23- Aug- 2016].

[10] E. Wilde and C. Pautasso, *REST*. New York: Springer, 2011, p.2-5.

[11] "Documentation", *Slim Framework*, 2016. [Online]. Available: http://www.slimframework.com/docs/ [Accessed: 23- Aug- 2016].

[12] "CMUSphinx Wiki [CMUSphinx Wiki]", *Cmusphinx.sourceforge.net*, 2016. [Online]. Available: http://cmusphinx.sourceforge.net/wiki/ [Accessed: 10- Jul- 2016].

[13] P. Aleksic, M. Gohdsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach and P. Moreno, "Bringing Contextual Information to Google Speech Recognition", 2016.

[14] "Download Android Studio and SDK Tools | Android Studio", *Developer.android.com*, 2016. [Online]. Available: https://developer.android.com/studio/index.html [Accessed: 13- Jun- 2016].

[15] J. Gerner, *Professional LAMP*. Indianapolis, IN: Wiley Pub., 2006.

[16] D. Group, "About the Apache HTTP Server Project - The Apache HTTP Server Project", *Httpd.apache.org*, 2016. [Online]. Available: https://httpd.apache.org/ABOUT_APACHE.html [Accessed: 07- Jul- 2016].

[17] "Novell Doc: NW 6.5 SP8: Novell MySQL Administration Guide - Benefits of MySQL", *Novell.com*, 2016. [Online]. Available: http://www.novell.com/documentation/nw65/web_mysql_nw/data/aj5bj52.html [Accessed: 23- Aug- 2016].

[18] K. Beck et al, "Manifesto for Agile Software Development", *Agilemanifesto.org*, 2001. [Online]. Available: http://agilemanifesto.org/ [Accessed: 17- Aug- 2016].

[19] E. Wong, "Shneiderman's Eight Golden Rules Will Help You Design Better Interfaces", *The Interaction Design Foundation*, 2016. [Online]. Available: https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces [Accessed: 23- Aug- 2016].

[20] R. Tamada, "Android Speech To Text Tutorial", *androidhive*, 2014. [Online]. Available: http://www.androidhive.info/2014/07/android-speech-to-text-tutorial/ [Accessed: 01- Jun- 2016].

[21] Android, "AsyncTask | Android Developers", *Developer.android.com*. [Online]. Available: https://developer.android.com/reference/android/os/AsyncTask.html [Accessed: 19- Jun- 2016].

[22] "How to work with SQLite in android", *Stackoverflow.com*, 2016. [Online]. Available: http://stackoverflow.com/questions/32229040/how-to-work-with-sqlite-in-android . [Accessed: 15- Jul- 2016].

[23] G. Developers, "Google Maps Intents", *Google Developers*, 2016. [Online]. Available: https://developers.google.com/maps/documentation/android-api/intents . [Accessed: 06- Jun- 2016].

[24] "Android Text To Speech Tutorial", *www.tutorialspoint.com*, 2016. [Online]. Available: http://www.tutorialspoint.com/android/android_text_to_speech.htm . [Accessed: 02- Aug- 2016].

[25] J. Callaham, "Google says there are now 1.4 billion active Android devices worldwide", *Android Central*, 2015. [Online]. Available: http://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide . [Accessed: 24- Aug- 2016].

Appendix 1- User Questionnaire

<u>SuiteCRM Voice Enabled Mobile Application Questionnaire</u>

Thank you for participating in this testing session for the prototype SuiteCRM mobile application. This testing is being conducted as part of an MSc project at the University of Stirling in conjunction with SalesAgility. Any response you give will be used in-part to either make recommendations for further development or to qualify decisions made in the process.

Before testing begins, please be aware that this is a very early stage prototype and many features still require significant improvement. This is not intended to be a release quality product but we are looking for feedback that can help the project get there. The questions you will be asked are based on usability and are looking to understand how a variety of users navigate the application and how easy or difficult they find performing fundamental tasks. Please follow the instructions as best as you can and provide feedback wherever possible. You will now be taken through the questions and observed whilst you use the application. Please feel free to ask questions at any point.

Native Language:
Frequent user of mobile apps:
Do you own a Smartphone:
Accent:
Gender:
Level of Background Noise:
Other:

<u>Section A - Testing</u>

1. Go to Accounts and select "Tracker Com LP"…. Edit the email Address to www.salesagility.com and update the record. Click on the www button to visit the web page.

2. Go to Contacts and select "Create+"… Create a Contact and remember the Name.

3. Using the search by voice button, please speak the name of the contact you created.

4. When you have found the contact you created Delete it.

5. Go to Meetings and select "South Sea Plumbing Products". Have the device read out the notes section and the edit the notes to say whatever you wish.

6. Staying on Meetings use the phone button to open the dialler (Don't call the number in the dialler) Navigate back to the meetings page using the back button in the bottom left corner.

7. Open the location of your meeting with the Google maps button, this displays the location of the meeting.

8. Click the navigation button to open the satellite navigation feature. Head back using the back button in the bottom left corner.

9. Please take a minute to explore the app and see what features are available.

<u>Section B - Questionnaire</u>

Please rank the following questions between 1 & 5 based on how much you agree with the following statements.

1 = Strongly disagree

2 = Disagree

3 = Neither agree nor disagree

4 = Agree

5 = Strongly agree

1. Navigating the application was easy and everything was where I expected it to be.

   ____

2. There were no issues navigating the application and each action had a predictable outcome.

   ____

3. The voice search was able to translate what I said and generate an accurate search based on what I wanted.

   ____

4. All the functions (create, retrieve, update and delete) on the records appeared to work as I would expect.

   ____

5. When the application read out the contents of the meeting notes I thought this was a feature I would use.

   ____

6. The inclusion of third party applications such as Google maps for navigation were a useful feature.

   ____

Section C - Feedback

In this section, please provide relevant feedback to the questions providing as much detail as possible.

1. What features did you like about the application and why?

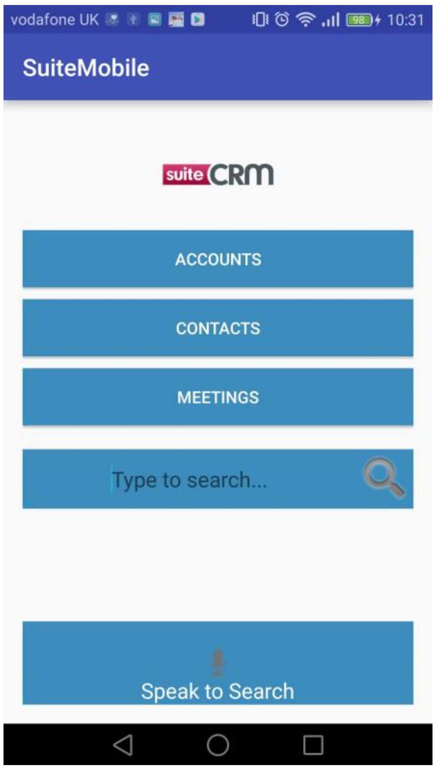
2. What did you not like about the application and why?

3. What did you think of the user interface and how would you make changes if at all?

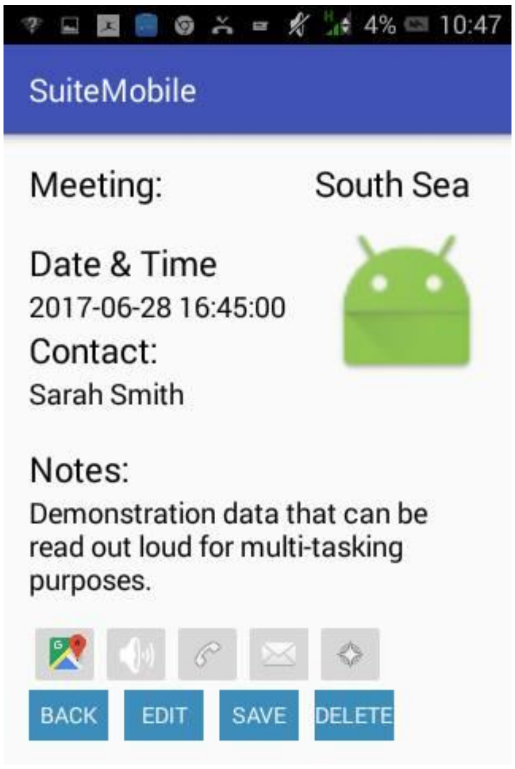
4. Please provide any additional feedback based on what you have seen and things you would like to see included in future.

5. Would this application be of any benefit to you or how you use SuiteCRM in your current role?

Appendix 2 – Compatibility Testing Screenshots



5.2" Display Nexus 5X



Meeting: South Sea

Date & Time

2017-06-28 16:45:00

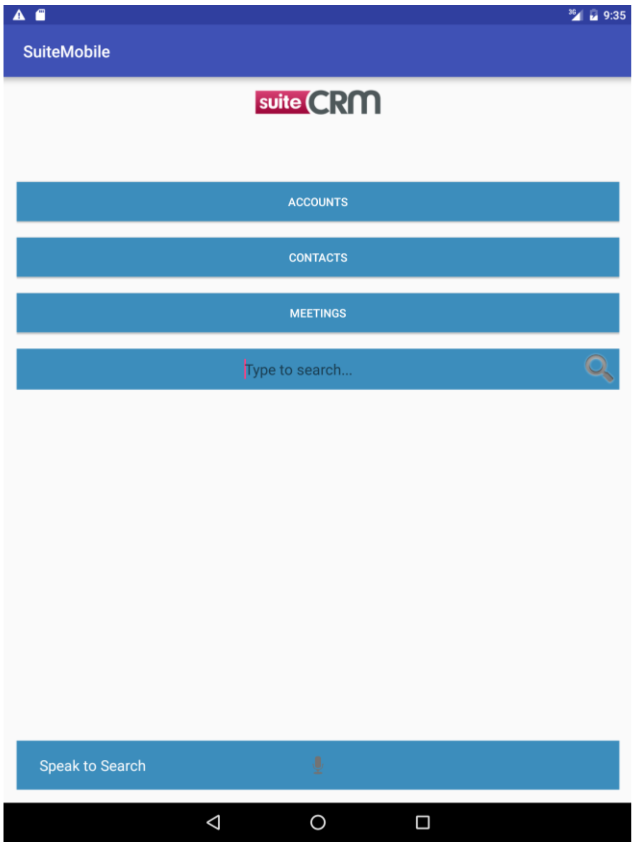
Contact:

Sarah Smith

Notes:

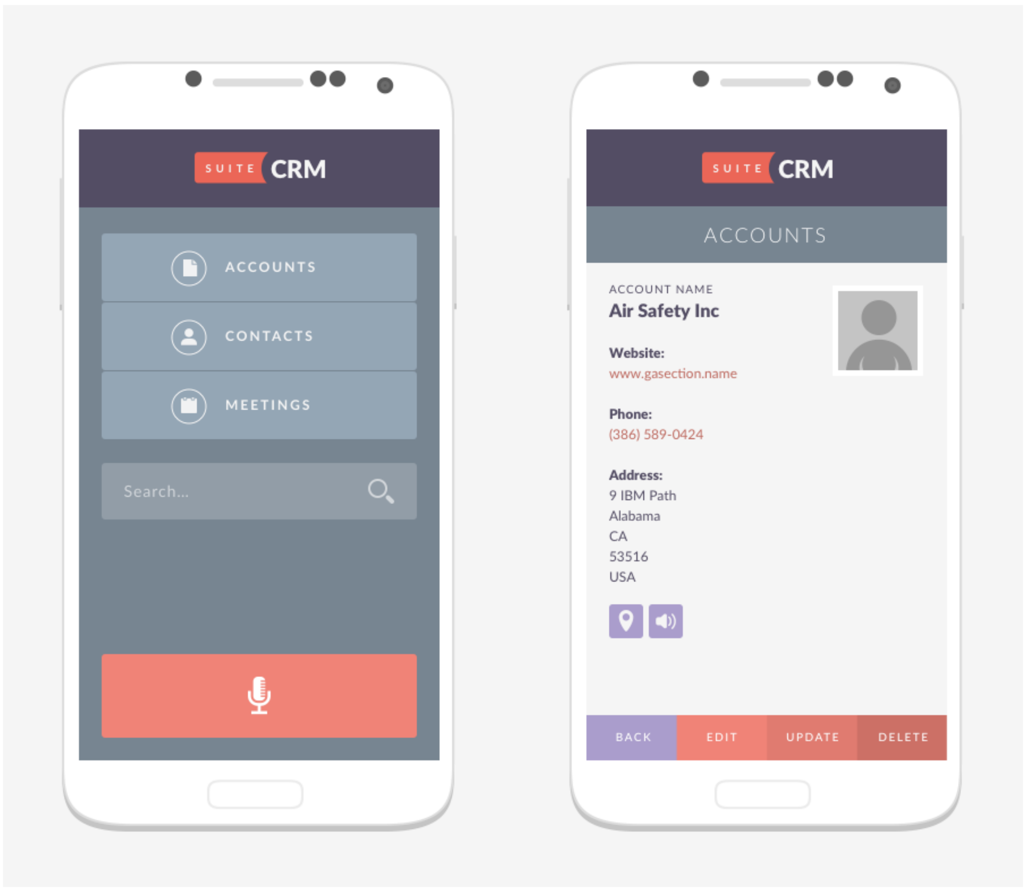
Demonstration data that can be read out loud for multi-tasking purposes.

3.2" Display Alcatel Pixi 3.5



7" Nexus Tablet – SuiteCRM expands to fit the screen.

Appendix 3 – Future Work Screenshots



The screenshots above were produced in house by SalesAgility to display what the application could look like in the future.

Appendix 4 – SuiteCRM Requirements Matrix

Platform	
Linux, Unix, Mac OS	Any version supporting PHP
Windows	Windows Server 2008+
PHP	5.5, 5.6, 7.0
Web Server	
Apache	2.2, 2.4
IIS	8, 8.5
Database	
MariaDB	5.5, 10, 10.1
MySQL	5.5, 5.6
SQL Server	SQL Server 2008+
Browsers	
Chrome	43+
Firefox	38+
IE	11 (compatibility mode not supported)
Edge	26
Safari	6+