WitsCABS Documentation
Project 10
Group 10

Group Leader: 1171733 - Jason Stuart

Front End
886515 - Amine Boukrout
669006 - Rashad Akoodie

Back End
1064934 - Robert Basson
1171733 - Jason Stuart (GL)

September 30, 2017

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe what is required for the development of WITSCABS (a lift service similar to the one provided by Uber). This document is meant to convey how we have decided to develop our system and the functionality required for the first release. This will be achieved through a description of the scope of the software being designed, so that the project bounds are clearly defined, in order to prevent scope creep. The document will then for both the front and back ends describe their functional requirements.There will also be a description of the software required to be integrated with our software solution in order for it to be functional according to the requirements. In addition to this, a fully descriptive documentation of all programming techniques as well as software engineering techniques that we as a group will follow during development will be included. These techniques will be substantiated in order to provide a clear systematic approach to the complete design of our project. Furthermore, all resources consulted will be included as well.

1.2 Problem Statement

WITS students often require transport to get them from place to place, it could be to and from a place of residence, shopping centers, places to support wits sports, internet cafes etc. Wits would like to implement a transport system similar to Uber that Wits students are able to use as a means of travel.

1.3 Project Objective

The objective of this project is not only to learn and make use of specific design procedures and team management skills, but also to further our skills in development of software using specific techniques that will be used in industry. Further, the objective is to have a working prototype of the WitsCABS project by the end, using these skills so that we are more comfortable when going into industry.

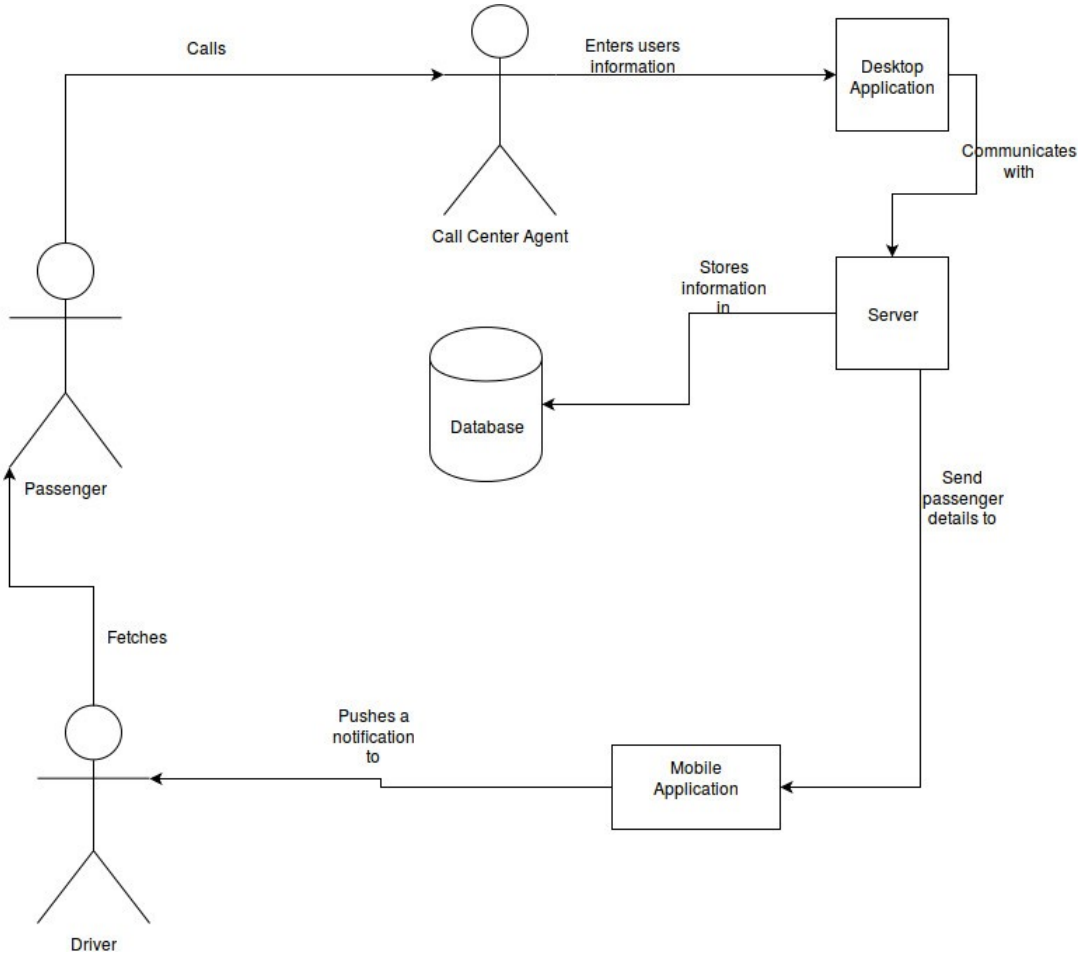
1.4 Scope

Our core objective is to design, develop and implement a software system which will be used to manage a fleet of taxi-cabs for a company (as per the project brief). Our company, WitsCABS Company, will be independently run and will employ our own staff/drivers. Vehicles however may either be owned by WitsCABS Company or by the drivers. The service will run based on service zones within the city of operation i.e. the city will be split into various sectors which will allow for more efficient ride time. Each driver will be equipped with a smart-phone embedded with GPS, maps and a navigational tracking system. All vehicles being dispatched will be on instruction from the 24-hour manned Dispatching Control Centre (DCC) which will be in direct contact with clients via calls to the centre.

The front-end of our system will comprise mainly of an android application used by the drivers which will be integrated with a GPS and mapping service such as Google maps to optimize routes taken as well as save time in the case of human error. A simple website will be set up as well to deal with customer queries/complaints and basic information about WitsCABS. To achieve this, we will be primarily using Java as a programming language as well as some HTML and Javascript. The front-end of our mobile android application will consist of the visual representation of the map being used as well as any turning signals or warning, messages etc. needing to be displayed on the screen, a login screen for the drivers where they would enter their credentials, a waiting screen for when they are awaiting a client. Furthermore, the drivers should communicate with the DCC. For this, a web-based application and/or application will need to be created for the DCC where they can view the entire city of operations. The front-end of this application would comprise of a form that will allow the DCC to capture customer details. It will then send these details to the back-end which will then determine which cab to assign.

The Back-end of our system will make use of Java as well as JSON. This will consist of storing all the drivers information (Names, license number, car registration, age etc.) as well as each trip he works as well as any necessary information about the passenger transported. It will also be integral to integrate the maps into finding which driver is best suited for the next pickup requested based on some sort of efficiency algorithm which we will design. This information would most probably need to be stored in a database and communicate with our mobile application using JSON. Another useful system to have would be to record every client that has rode with one of our

cabs before and perhaps assign a customer number or ID to use the following time he/she requests a ride.



1.5 Overview

The document will be broken up into multiple sections. As already mentioned before in the document, we have given an introduction as to why this document has been typed up, the objectives to be met, and the problem statement that describes the problem at hand. The scope of what the project is all about and a basic description of the required features needed to be implemented is also discussed above. We also discuss the relevant stakeholders next. In section 2, an analysis of how the project will be conducted is discussed. This includes a discussion of system architecture types, software required for the project, as well as how the team will be managed and handled during development. In section 3, a list of formal required functions will be provided for both front-end software as well as the back-end. Section 4 will then go into the design documentation, such as Database design, e.t.c. Section 5 will then contain relevant documentation related to Sprint Planning as well as Sprint Retrospectives for the SCRUM Methodology. Then The document will discuss how to evaluate the software for the project owner's use in Section 6. Finally a list of responsibilities and credentials is provided in section 7.

1.6 Stakeholders

The following people have an interest in this project:

1.6.1 Users

The future users of this project, being students that require transport within service areas, will be keen to use this software as transport is still a major issue in South Africa. This often leads to people not getting to their destination on time, a key factor for many. Hence these users will be keen to know how this software turns out.

1.6.2 Developers

Apart from the developers having to do this for their job, they too will be interested in making sure this project flows and is completed by the end of it, as it will further develop their skills as well as could help them further once the project is completed.

1.6.3 Project Management

Project Managers, the ones who handle the project at an administration level also have similar goals to developers.

1.6.4 Project Owners

These are the people who initiated the idea of this project and have since contacted us to implement such a project. This will provide good gains to them as it will improve the reputation of their company as they try to solve a common problem in South Africa.

2 Analysis

2.1 Team Administration

2.1.1 Team Management

Our team has decided to take the Agile SCRUM method as our SDLC (software development life cycle). We chose this approach based on multiple reasons. Due to the short time constraint given to us by the client, a rapid development approach is needed, one that is satisfied by Agile development.

We also require quite a bit of feedback from the client, hence this iterative approach is best as it allows us to be in constant communication with the client, with small changes in between, allowing us to quickly change anything the client is unhappy with. Another advantage of SCRUM specifically, is that we will constantly be meeting up each day either in person, or by Google Hangouts to check up on the group and what each member has done over the last day, allowing us to keep a high morale and high sense of worth in the team, even though the time requirement is strictly short. Since our team is highly motivated already and have experience in development, they can be trusted to handle their tasks efficiently, another benefit of the SCRUM process, where each developer has more responsibility than in other methodologies. Also, since our team is relatively small, SCRUM will also work, as there will not be as much time required to structure and organize the team as a whole, increasing productivity. Also, since SCRUM focuses more on the actual development and less on the documentation, since we are going to be showing each change to the client, it benefits the productivity even more.

We have also considered the disadvantages of SCRUM. We understand that there is a higher chance of Scope creep with Scrum, but this is why we have already defined the scope of the project in the beginning of this document, to define clear boundaries to the project, to prevent this from happening. We have also been very clear with our functional requirements needed for a successful project, to prevent inaccurate measurement of time estimations.

Furthermore, SCRUM has the possibility of failure, if at least 1 member loses interest or focus. This is why we have nominated a clear leader to take charge and to keep the team motivated, regardless of what may come. If a team member were to leave, this would also have a large inverse effect on the project, however this will not happen, as each member is fully committed to this project, as this project results in marks required to graduate.

Finally, it may be hard to quantify quality as the software is rapidly evolving, however since we do not have a quality control team, we can work on quality between each developer to keep the project from slipping quality wise.

2.1.2 Team Meetings

The team will conduct the project as follows:

> Each team member will be required to be at each sprint planning meeting every 2 weeks, from the start of the project. This meeting will be organized by the Group Leader, i.e. the scrum master.
>
> Each member will be required to be available for the daily stand up meeting, preferably in person, otherwise via Google Hangouts. Each member should be able to describe to the team what they have worked on over the past 24 hours. Thereafter each member will describe what they will work on for the next 24 hours.
>
> The scrum master will ensure that the project is moving at a reasonable pase and that all team members are motivated at all times.

2.1.3 Responsibilities of Team

Our current student assignment is as follows. Rashad and Amine will handle the mobile app. Robert will handle the frontend. Jason will handle the backend. As the project moves forward, this can change, depending on who needs what help at each time. Each member is responsible for providing constant communication and feedback on their portions as well as suggestions on others work. Jason will schedule any meetings required to advance the project in any way.

2.2 System Architecture Analysis

After thorough analysis of existing architectures and analyzing the functional requirements listed later in this document, the team has decided to make use of a 2 part Client - 1 part Server architecture. The first client software will be the mobile app that allows the drivers of the WITSCABS service to receive communication from the system as to who the driver should pick up, etc.

The other client software will be run on a desktop in the call center, so that as a client calls in, the operator can input all the details necessary for the system to compute the best driver to use. The server will be the bridge between these two client software services, where the server determines the best driver for the job that arrives and pushes the required information through to the driver on his mobile app.

All this communication will be facilitated through an Internet connection, so when designing the system, it must take into consideration that an internet connection might not always be available for the drivers. There is no need to have any further server or client interfaces thereafter.

2.3 Analysis of Front-end System (Mobile App)

We have decided to make use of an android application as the primary user-accessible interface for our project. The reason for choosing a mobile application is quite obvious in our case as the drivers who will be using the application have to be in their vehicle at the time of using the application as well as moving in the vehicle while using it. This automatically squashes any idea of a desktop application or web based application as the accessibility would be a major issue. The possible downside to this would be that each driver requires a cellphone/tablet which will support the software WITSCabs as well as a mobile data connection.

The call centre will be using a desktop application in conjunction with the android application although the users will not be registered to the database but instead will have access to the database in a read-only capacity. This will allow the call-centre to access client details as well as drivers details which will in turn allow them to run statistics checks as well as be prepared in the case of an emergency. For this, it is clearly most beneficial to use a desktop application preferably with an internet connection as immediate run-time statistics as well as driver locations etc. are imperative.

For this purpose we have chosen to develop our software as an android application as it is the most common operating system friendly platform for a mobile application. Linking with this, it has been decided as a group that a predominant bulk of the code used should be Java as it is the most commonly used programming language for Android applications as well as the most familiar between each member of the group. Adding to this, we have decided to use IntelliJ idea as an IDE for our app development as it is popular, free to students and boasts many high-end features which will benefit the development of our project.

2.4 Analysis of Front-End System (Desktop App)

When a user wants to get a lift they have to call in and speak to a call center agent. The call center agent will then get the relevant information from the user and capture it in a desktop application. This application will communicate with a server which will capture the information in a database. The server then sends a notification to the mobile application so that a driver can pick up the user.

We have chosen to use a desktop application over a web application. The main reason for this is responsiveness. Web applications can become unresponsive if the internet connection is heavily used or if there are connectivity issues such as a damaged line, desktop applications have a more reliable response time as response time will be limited by the CPU. Desktop applications are also more secure than web applications which will be important as we store user information on a database.

The language chosen to develop the desktop app is Java. Java is platform independent so it will run no matter what operating system is on the call center computers. Java is an easy to learn object oriented language, this helps us to create reusable code. Another benefit of Java being easy to learn is its popularity, this means that should any problems arise it won't be difficult to find someone to fix these problems.

2.5 Analysis of Back-End System

The back-end will consist of two parts. The server application and the database. Our database will be developed using the DBMS MySQL. MySQL is a very commonly used and simple DBMS so any issues with the database can be dealt with quickly and easily. As MySQL is very easy it only requires basic statements in order to interact with the database. We have also chosen MySQL as it is free which helps reduce the cost needed to develop the full system. MySQL is also able to handle large amounts of data which is well suited for a popular application. The DBMS also includes options to add security which can help protect the data in the database from intruders.

The development of the back-end is important even though the user does not interact with it. We need a means to store user information and assign drivers. This back-end actually serves as the lifeline for the system. Hence this will also be developed with Java, since all the members of the team are most familiar with the language and how to implement networking to the highest efficiency possible.

2.6 Additional Software APIs Required

Due to the nature of the project, the implementation of the software application will require external software and most probably certain add-ons from the supporting software. One of the main supporting software that is required to actually implement the application is Google maps along with its various APIs. This is required since the application has to be integrate with Google maps in order to provide a GPS service to guide the driver to the location of the customer.

A type of a database management system (DBMS) will be required in order to keep relevant information about the operations of the business. Such information would include login details, customer details, driver details, trip details, etc. Most probably the DBMS that will be used is either MySQL or PostgreSQL. It is worthwhile to note that as the project progresses additional software requirements may arise, and in this regard the client will be notified.

2.7 Analysis of Inputs and Outputs

2.7.1 Inputs

The first type of input would be a phone call from a customer providing information such as the pickup location, customer name, and contact number. The dispatching control center (DCC) agent will enter the information provided by the customer into a database by using an interface that will run on a localised desktop at the DCC. Once the information is stored on the database, the system will allocate a customer to a driver based on whether or not a driver is available or not (by using an algorithm designed and agreed upon by the developers). The driver will then be provided with the GPS information of the customer which will be inputted into the embedded GPS on the driver's smartphone.

2.7.2 Outputs

Once the driver has been allocated, a notification will be sent firstly to the driver via the designed app to notify him/her about the customer?s assignment to that particular driver and basic information, such as the customer?s name and contact number, along with the physical address or GPS coordinates that the customer provided (this can be displayed the driver?s app as link which could launch the designed application for the company or the GPS app embedded in the smart phone). On the customer?s side, he/she will receive a SMS when the driver is in a predefined range from the address provided by the customer.

3 Functional Requirements

3.1 Front-End Mobile App

- Send Status updates to the server on whether this driver is available or not
- Send location updates so the server knows where you are when status is marked available.
- Be able to receive passenger information that will help the driver find the passenger.
- Find the quickest route to the destination.
- Update Driver details
- Register Driver onto the system.

3.2 Front-End Desktop Application

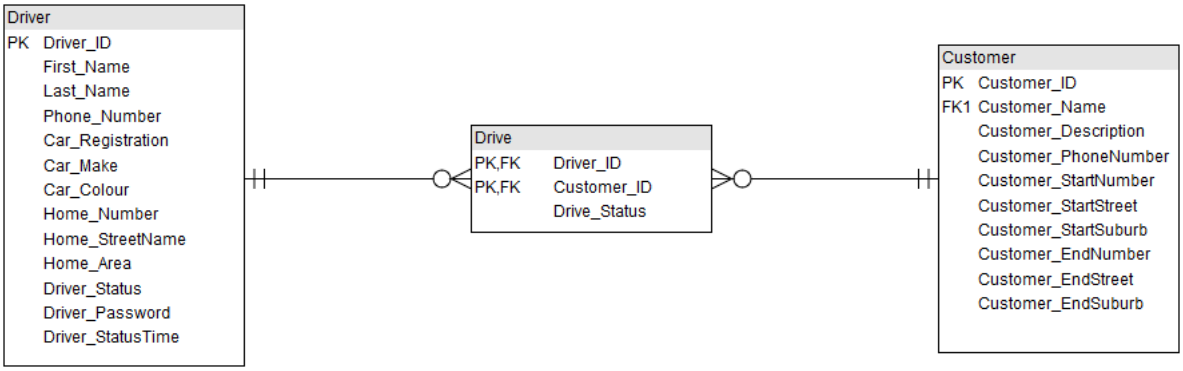
- Create new passenger
- Insert passenger into database via the server
- Edit record
- Push and save information to database

3.3 Back-End Server

- Accept incoming connections from desktop applications, which contains details related to the passenger in need of transport.
- Analyze which service zone the passenger is located in.
- Run an algorithm to find the closest driver ready to pick up a passenger
- Send a push message to this specific driver with the details of the passenger to fetch as well as their destination.
- Allow drivers to mark if available or not via a status change
- Be notified when a driver is close to pick up point via SMS.
- Be notified when passenger is there at their destination, to automatically mark drivers as complete, i.e. the driver is transitioning back to their house or service taxi rank. Thereafter the driver can mark they are ready for another job.

4 Design Documentation

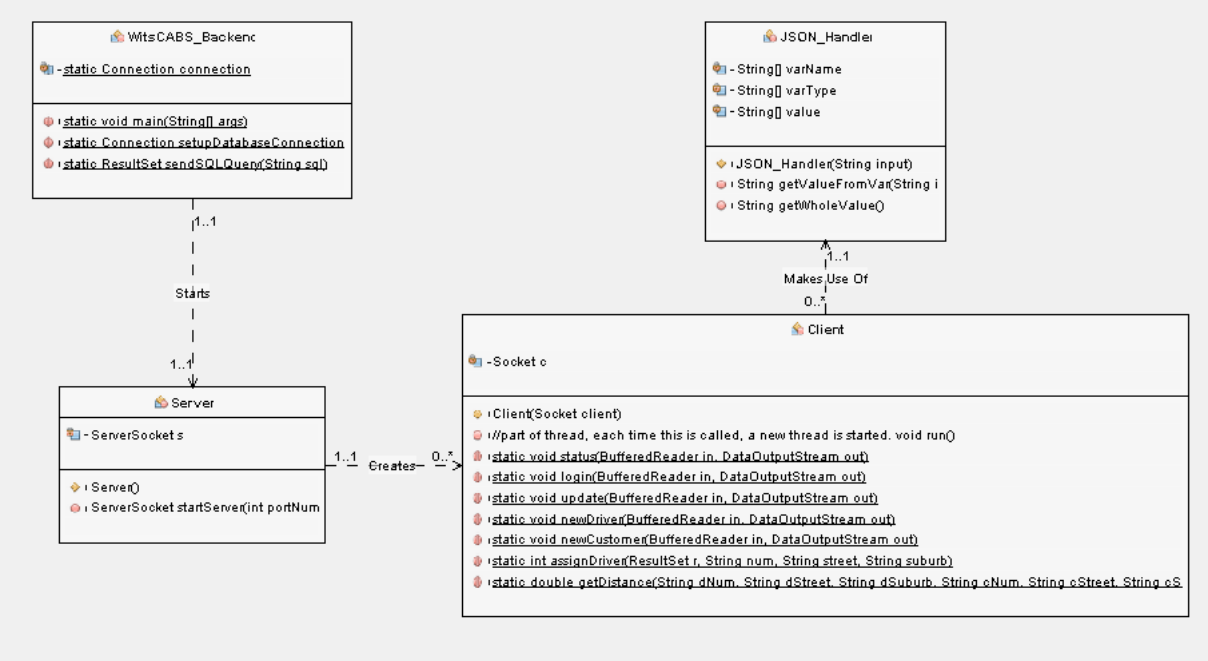
4.1 Database Design



The approach taken here is to achieve as minimalistic approach, storing the least amount of information possible. Hence we just have 3 tables, one for the customer and their details, one for the driver and their details and a bridge entity to match the two together. To simplify things, we decided that if a driver is part of a rank, their home address details become the ranks address. We also made it that the customers do not have to each time log in, they just call and create a new customer each time for convenience.

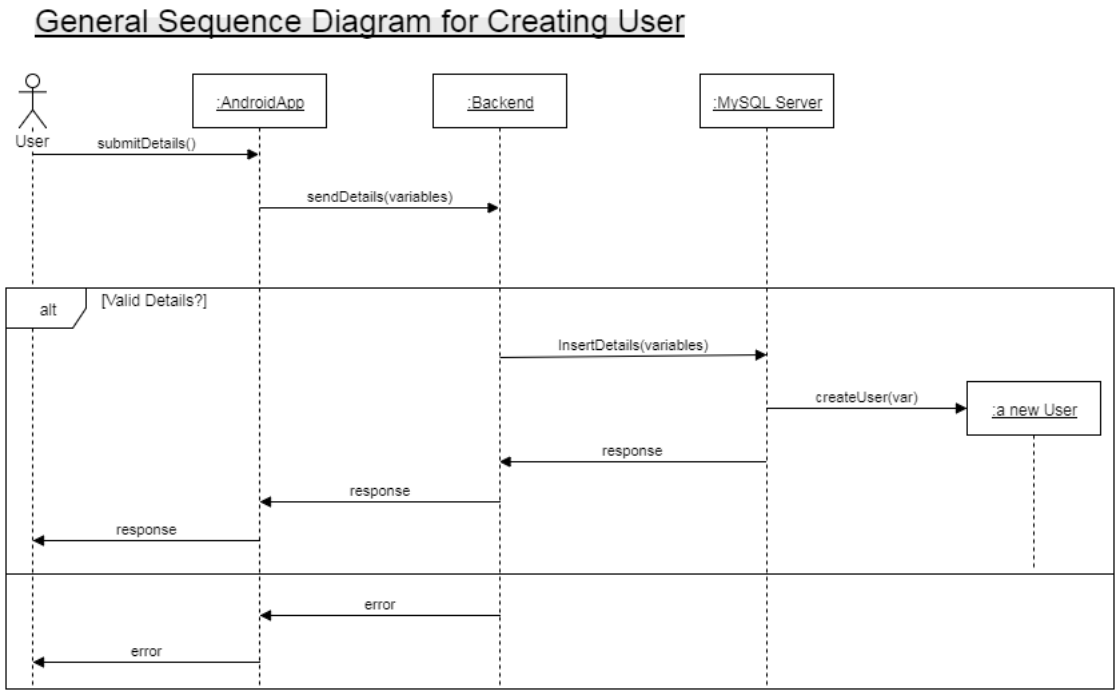
4.2 Server Design

4.2.1 Class Diagram

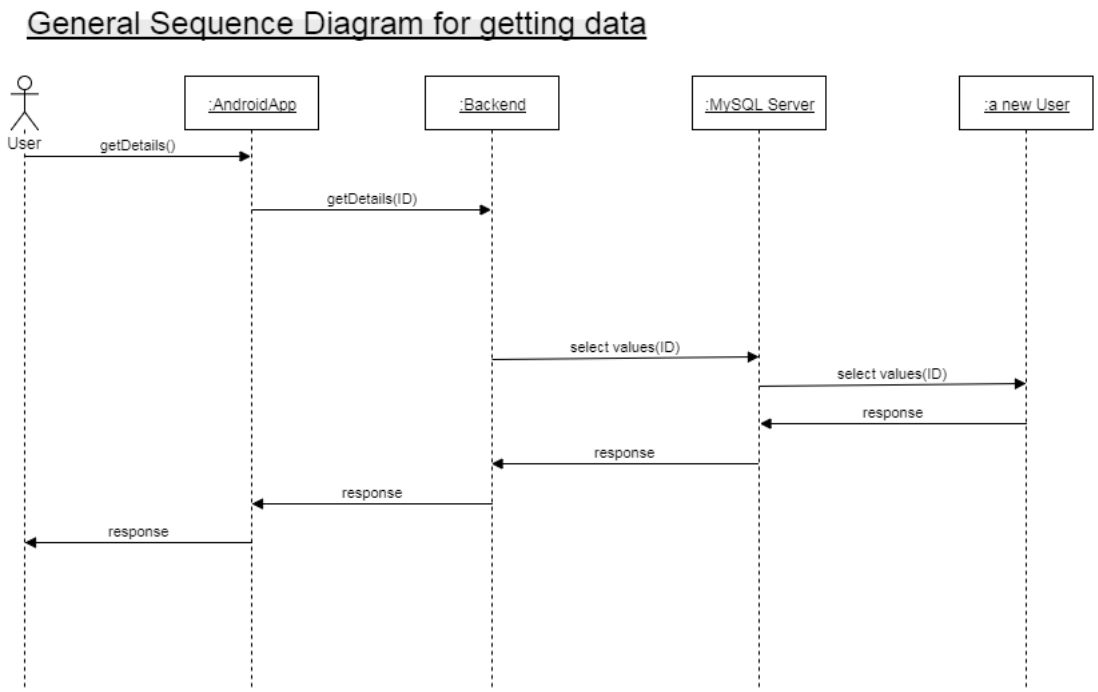


When designing the server, we as a group decided that it would be best to make sure efficiency was our main priority, especially due to the networking specifics required. Hence we went for a minimalist approach and kept all networking as simple as possible. We also made use of threads to improve response time when multiple queries came in at once, as each thread could handle a request at the same time. Further, we decided to separate the code into multiple classes and methods to make the code more understandable and easier to read, as well as make sure that there is upgradability and maintainability with the code, for further development. The image above describes the class diagram that links all the classes together.

4.2.2 Sequence Diagrams

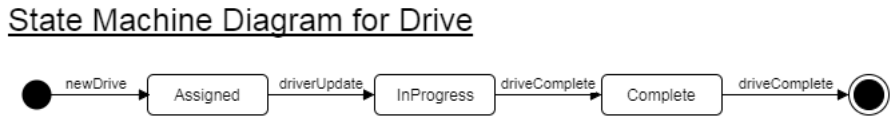


The above diagram shows the general procedure that is followed when either a new customer or driver is created. The steps that the applications take are outlined in this diagram above.



Similarly, the above diagram shows how the front end apps query the server for any data they need. They again, follow the same general strategy as outlined in the sequence diagram.

4.2.3 State Machine Transition Diagram

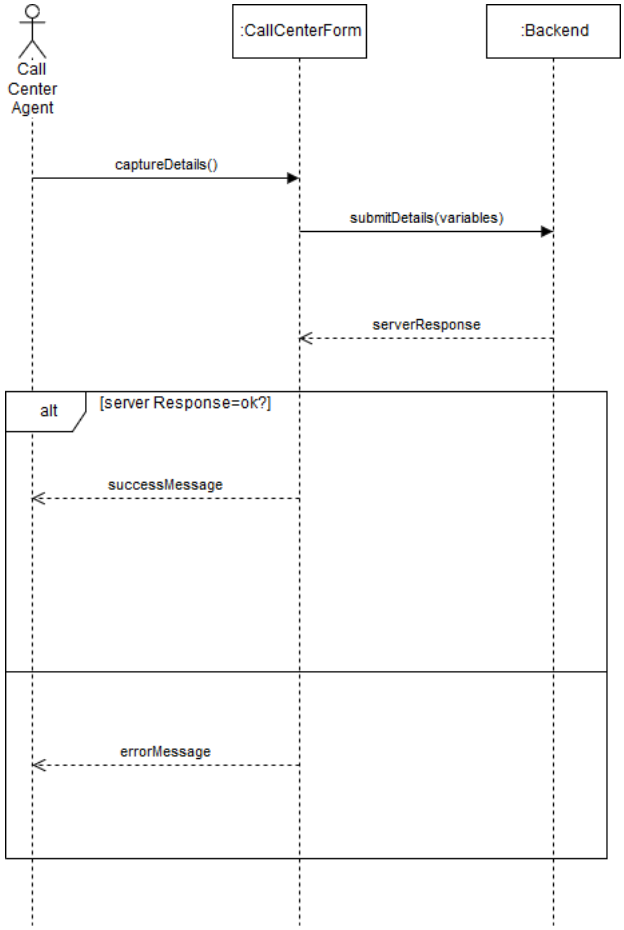


The above diagram shows how the state of a drive progresses as events occur. When the customer is newly created from the call center, they are "assigned" to a driver, but the driver does not know this, until the app refreshes with the server. Once the driver has the refreshed information, i.e. the drive has been transferred to a driver, then the drive is "InProgress". Finally, once the drive is "completed", it is marked as so, and archived.

4.2.4 Driver Assignment Algorithm

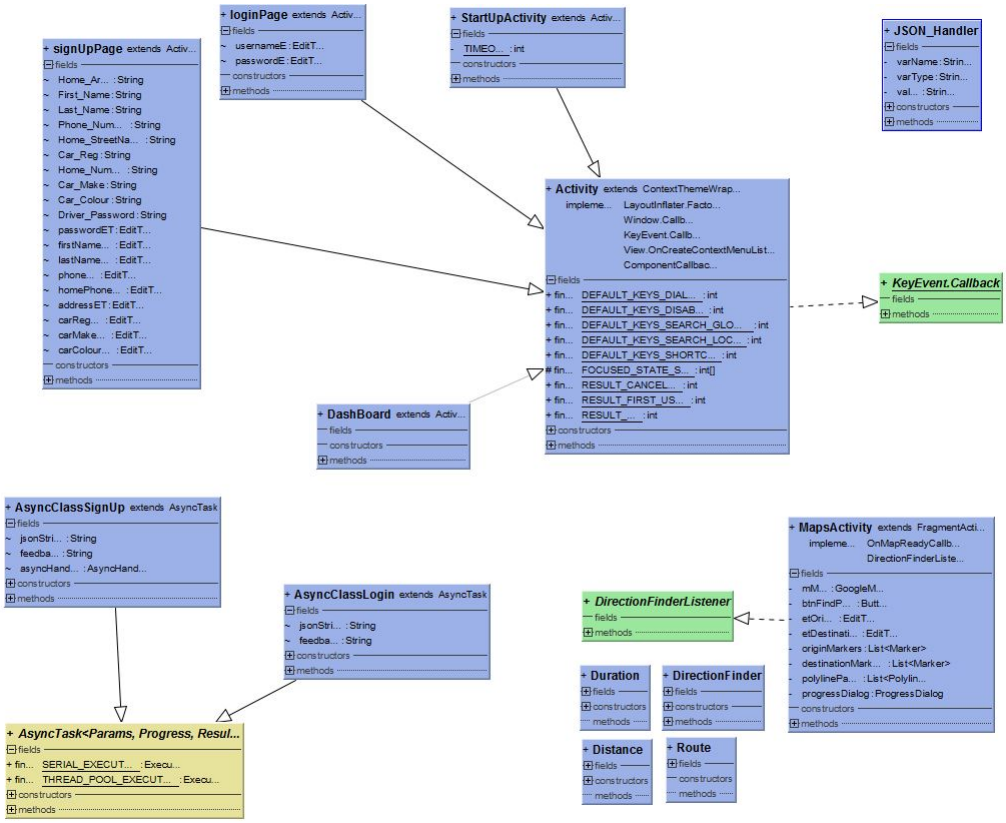
Apart from the previously mentioned and discussed design diagrams, it is critical to discuss the design of the algorithm that chooses which driver to dispatch to each customer. The server gets a resultSet from the database of all drivers ready for a drive in the service zone related to the customers starting position. It then, using the distanceMatrix API provided by google calculates which driver will get to the customer first in terms of time taken (including traffic delays e.t.c.) and assigns that driver to go fetch the customer. If the closest drivers are located at the Service Rank, the algorithm picks the driver that marked their status as available first at the service rank. It also then marks the driver status as unavailable and the server then waits for the driver to get the customer details.

4.3 Call Center Design



The diagram shows the process that occurs when a user calls the call-center and the call-center agent captures their details on the form. The steps that need to occur are outlined above.

4.4 Android App Design



As any other android applications, the application consists of activities which are standalone and completely independent from one another. (To clarify terminology, an activity is a screen that displays on the android device in use.) The application also consists of several auxiliary classes which assist in the functionalities of the main activities (i.e. the activities). The WitsCabs android application has five activities in total, wherein three out of the five activities are the core activities of the application. Each of the five activities are standalone as stated previously, and thus each of the five activities constitute to different and specific functionalities of the application. Each of the activities are linked to a single .xml file which helps to display the visual aspect of the activity (the GUI) on the device.
The different (main) activities along with their functionalities are described below:

- StartUpActivity: this activity is linked to the startupactivity.xml. This specific activity is launched when the application is launched. The StartUpActivity displays an image along with text that reads “Welcome to WitsCabs!”, and automatically times out after 7 seconds to start up

the next activity which is the login page for the application.

- loginPage: this activity is linked to the loginlayout.xml and is launched automatically after the StartUpActivity. This activity allows the user (i.e. the driver) to login to the application by inserting his/her username and password in the provided text boxes followed by clicking the login button, and also provides a button which the user can click to register as a driver at WitsCabs. Depending on which button is clicked the application will lead to a different activity. If the login button is clicked, the application will then lead to the dashboard of the application and if the register button is clicked, the application will lead to the register activity.

- signUpPage: this activity is linked to the signuplayout.xml and is launched when the "Register" button is clicked in the loginPage activity. This activity consists of several text boxes which allows the user to insert his/her information into. Note that these text boxes have "hints" which indicates to the user what information is required to be entered by the user at each text box. At the bottom of the application there is a button named "Register" to officially register the user, and once clicked and if successful the application will lead back to the loginPage activity and if unsuccessful the application will stay on the same activity and display a not registered message via a toast.

- DashBoard: this activity is linked to the dashboardlayout.xml and is launched when the user is successfully logged in, as well as this activity is the "main activity" after logging in. In this activity is where the user can change his availability status, there is an analogue clock that displays the time, a text view section which displays the details of the passenger that has been allocated to the specific driver, and there is a "Go To Maps" button which once clicked the application leads to the MapsActivity.

Note: all of the above activities extends from the Activity superclass.

- MapsActivity: this activity is linked to the activity_maps.xml and is launched when the "Go To Maps" button is clicked in the dashboard. The main purpose of the activity is to give directions to the driver to get to the passenger, and to get the passenger to his/her destination. This activity makes use of the google maps API and google directions API, along with the auxiliary predefined modules by google. These modules include DirectionFinder determines the route of the trip required,

DirectionFinderListener which listens when directions are requested, Duration calculates and stores the duration of the current trip, Route stores the route.

Auxiliary classes:

- AsyncClassSignUp and AsyncClassLogin: initiates the communication with the back-end server in order to send the required registration and login information respectively to the server, and receives a response from the server. Based on the response, certain decisions are made in the signUpPage and loginPage respectively. Please note that the IP address when declaring the Socket has to be altered based on the machine that the back-end server is being run on.

- JSON_Handler: converts an array of strings to a JSON string in order to send data to the server using a JSON string format. This class was created by Jason for use in the server, but we made use of it in the android app as well.

Important Notes:
The actual application is not fully as required due time constraints and issues with the google API, however basic functionality of the app is operating.

5 SCRUM Documentation

Note: Each standard sprint we implemented has a length of 1 week. It should also be noted that each day at 19:00 the team would do a daily stand-up meeting via Google Hangouts wherein each member would describe what they have done, the problems they faced, as well as get any answers related to their concerns if they needed to do so.

5.1 Sprint 1: 7 September 2017

5.1.1 Sprint Planning Documentation

Task	Assigned
Create Database Structure	Jason
Implement Server Socket for backend	Jason
Implement Threading for server	Jason
Implement Call Center App	Robert
Link Call Center and Backend Together	Robert And Jason
Construct Android Screens	Amine and Rashad
Analyze Google API for use	Jason

5.1.2 Sprint Retrospective

Overview
All Tasks Completed Successfully Except for the Android Screens. Amine and Rashad encountered a strange bug that would not allow them to compile their code, hence they have been held up and have not met the deadline. Their team has gone to get assistance with the bug from staff who are knowledgeable with android development.
Robert Successfully completed the Call Center Screens within 2 days, but the holdup was implementing the networking code as we had issues with the proxy blocking our connection. We found a workaround and now have a working connection.
Jason installed and created the mysql database as discussed in the planning stages of the project. He also implemented the server networking code and Threading code as well. He also helped Robert connect the applications together. He further went on to analyze the google api and even implemented it into the server code ahead of schedule.

Negatives

Currently the android team is behind schedule due to the bug. Hence it is critical that they can recover from this situation and catch up. This means they will have to work extra hard this next week. Also the networking methodology was not clearly described by Jason until later on in the week, as he was still implementing the code, hence the networking of Robert's code could not be implemented yet.

Positives

Jason managed to get ahead of schedule and implement slightly more than what was required for the week. This puts us in a relatively good spot as more time can go into making sure the server is working to its optimal pace. Communication within the team on each day was also excellent, with each member checking in on google hangouts for the daily standup meeting. The android team also immediately let all involved know that they had run into this IDE bug.

5.2 Sprint 2: 15 September 2017

5.2.1 Sprint Planning Document

Task	Assigned
Implement all Server functionality required for both front-ends	Jason
Finish Android Screens	Amine and Rashad
Implement networking code for android app to server	Amine and Rashad
Implement functionality of basic functions for android prototype	Amine and Rashad
Implement Google Maps API into android app	Amine and Rashad
Touch up beauty aspects of Call center app	Robert
Test Connection and functionality between server and call center	Jason and Robert
Begin basic testing of server functionality and find bugs	Jason

5.2.2 Sprint Retrospective

Overview
This sprint went much better than the previous one. While the android team is still behind, they feel that they can complete the job by the end of sprint 3. They managed to add the required functionality and implement the Google maps api into the app. They also managed to complete the android screens. They now just need to add the networking code. Robert finished commenting and tidying up the code for the call-center app. Jason did bug testing excluding the android side of things on the server.
Negatives
Once again, the android team encountered some strange bugs with the IDE while developing the app, this slowed them down and in the end they could not finish their networking portion of the code. They are however more than ever determined to sidestep these problems to the best of their ability and make sure that all can be done as quickly and efficiently as possible.
Positives
Our group has begun to work more with each other and less on their own, speeding up the development process. Furthermore, it has become more clear to each member exactly what is required of them. Every member is motivated and positive they can finish all content in the next sprint.

5.3 Sprint 3: 22 September 2017

5.3.1 Sprint Planning Document

Task	Assigned
Implement networking code for android app to server	Amine and Rashad
Bug test server code with android app	Whole Team
Advanced testing of server functionality and find bugs	Jason
Begin overall test cases and find bugs in all software	Whole Team

5.3.2 Sprint Retrospective

Overview
The team is pleased to say that the networking code between the android app and server now successfully runs for some functions, meaning that the app is now in a demonstrable state. This took a lot of effort to program as we were constantly facing new challenges, but through perseverance, a strong quality of the team, it has been done. The main focus of this week however was bug testing, which the team did an admirable job finding these small but significant bugs in the software code. Testing also occurred which was a huge success.
Negatives
The team could not identify any Negative aspects relating to this week. All went smoothly.
Positives
The team has been putting in extra hours behind the scenes to make sure that the project deadline is met, even though these hours are not officially scholar hours. This shows that the team is dedicated and motivated to give their all in making sure this project completes successfully.

5.4 Mini Sprint 4: 29 September 2017 - 1 October 2017

During this time period, a major focus was made on neatening the document up into a final document that is legible and understandable. It should be noted that the document was worked on at all points during the development stages and evidence of this is provided on Github in the repository by analyzing commits and their contents.

6 Evaluation

6.1 Compiling And Running Code

6.1.1 Desktop Application

The Desktop Application to be used in the call center will be a Java program (as previously mentioned). So in order to compile it an IDE that supports Java will be required eg Eclipse or Netbeans. For the sake of this document it is assumed that Netbeans will be used. Further it is required that you have the latest JDK and JRE software installed from Oracle.
Netbeans has a built in compiler that is easy to use. On the ribbon at the top of screen there are two hammers, one hammer has a brush. These hammers are used to call "build" and "clean and build". Clicking either one of them will compile the code. The difference between the two is that clean and build completely recompiles the code, deleting former versions.
After compiling a .jar file should be present in the build folder of the Java project. This .jar file can be transferred to computers in the call center and will allow the application to run without the need for an IDE.It will require the use of the command prompt.
To run the code using an IDE there are multiple options, you can right click the main class and select run file, or click the green play button in the top ribbon. Either of these options will run the code if the code has been compiled or compile and then run the code. To run from the command line without the IDE installed, navigate to the file in the command prompt and then type: java -jar CallCenter.jar

6.1.2 Server

Similarly to the Call Center Desktop Application, the server has also been constructed using Java as the programming language in order to more easily maintain compatibility with all the other programs. So in order to compile the code, you can follow a similar approach to 6.1.1 for the desktop application. Note it is required that you have installed the latest java JDK as well as JRE software. There is however an extra step required to setup the database. One needs to install mySQL and install it to the system using the default settings. When setting up user details, set the root password to "jason". Keep the port number the default. Furthermore, make sure to restore the current database from the repository code to the server using mysql -u root -p WitsCABS ¡ WitsCABS.sql where this command is run in the same directory as the sql backup. Finally, to make sure that you can receive connections from client programs, make sure to give them your current computer ip address to input into their program code for testing. Note: it is critical to be on a network other than the Wits network as the proxy service and network policies interfere with the correct running of the server. To run the server, follow the similar setup as for the desktop application, i.e. navigate to the .jar file in the command prompt using cd and then type: java -jar WitsCABS_Backend.jar

6.1.3 Android Application

The Android application has been developed by using IntelliJ Idea as an IDE with Java as a programming language. Java has been chosen as it is the most familiar programming language between the group as well as the most used among android developers. Before installing the .apk file, one has to install the latest version of the Java jdk (1.8) and the android sdk (API 23). In order to install the app on a mobile device, one would be required to install IntelliJ Idea or Android Studio. We will assume that we're using IntelliJ Idea in this description. The first thing to do would to navigate to the project structure. In the project structure, choose the jdk and sdk you downloaded on your system. When using a mobile device to run the application, make sure that developer options are enabled. Do this by clicking 10 times on the build number under my device info in the settings of the android device. Once this is completed, enable USB Debugging in the developer options. You are now ready to install the WitsCabs app. Simply click on the "Run" tab on the top of the screen and choose "Run Application" with the mobile device connected to the pc running IntelliJ Idea via a usb cable. When prompted to choose which device to run the application on, choose the name of the android device being used and the application .apk file will be installed on the device. Once this is completed, all that is needed is a Wi-Fi or cellular connection on the device for the app to work. Note: once installed, the application can be run from the android device without having to install it again.

7 Credentials

7.1 The Credentials

- Purpose by Robert, Rashad, and Jason (Section 1.1)
- Problem Statement by Robert (Section 1.2)
- Project Objective by Jason (Section 1.3)
- Scope by Amine (Section 1.4)
- Image Process Diagram by Robert
- Overview by Jason (Section 1.5)
- Stakeholders by Jason (Section 1.6)
- Team Management by Jason (Section 2.1.1)
- Team Meetings by Jason (Section 2.1.2)
- Responsibilities of team by Jason (Section 2.1.3)
- System Architecture Analysis by Jason (Section 2.2)
- Analysis of Front-end System (Mobile App) by Rashad (Section 2.3)
- Analysis of Front-End System (Desktop App) by Robert (Section 2.4)
- Analysis of Back-End System by Robert (Section 2.5)
- Additional Software APIs Required by Amine (Section 2.6)
- Analysis of Inputs and outputs by whole team (Section 2.7)
- Front-End Mobile App to be handled by Amine and Rashad (Section 3.1)
- Front-End Desktop Application to be handled by Robert (Section 3.2)
- Back-End Server to be handled by Jason (Section 3.3)
- Database Design by Jason (Section 4.1)
- Server Design by Jason (Section 4.2)
- Call Center Design by Robert (Section 4.3)
- Android App Design by Amine (Section 4.4)
- SCRUM Documentation by Jason as Group Leader
- Evaluation, compiling and running Desktop Application by Robert (Section 6.1.1)
- Evaluation, compiling and running Server and Database by Jason (Section 6.1.2)

- Evaluation, compiling and running Android Application by Rashad (Section 6.1.3)