BSc (Hons) Computing Science

Client-Server Family Centre Application

Radoslaw Burkacki

B00309449

23rd March 2018

Supervisor: Ying Liang

Declaration

This dissertation is submitted in partial fulfilment of the requirements for the degree of Computing Science (Honours) in the University of the West of Scotland.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself. Following normal academic conventions, I have made due acknowledgement to the work of others.

Name:

Radoslaw Burkacki

Signature:

Date: 23th March 2018

Form to accompany Dissertation

To be completed in full

Surname: Burkacki
First Name: Radoslaw **Initials: RB**
Borrower ID Number: B00309449
Course Code: COMPSCI
Course Description: Computing Science (Hons)
Project Supervisor: Ying Liang
Dissertation Title: Client-Server Family Centre Application
Session: 2017/2018

Please ensure that a copy of this form is bound with your dissertation before submission

COMPUTING HONOURS PROJECT SPECIFICATION FORM

(Electronic copy available on Moodle Computing Hons Project Site)

Project Title: Client-Server Family Centre Application

Student: Radoslaw Burkacki **Banner ID: B00309449**

Supervisor: Ying Liang

Moderator: Michael McCready

Outline of Project:

Technologies like internet and GPS are available to anyone. Most of people now days do have a mobile device like smartphone or tablet that are capable of using internet and GPS. There are many families which would like to keep track of other family members for example keep track of current location of their child, communicate with them and to get an alert when their location is different from expected location for example child being at school. Currently there are **many** applications that allow families to use one of those features. My idea is to create **one** application which would include all of those features and some additional and it would be dedicated to families.

Main focus of the project is to create a family application which would have many features which could be used by family members. For example tracking each family member, chat, SOS and alerts. Those features would bring many benefits to families like safety of family members, communication and many more.

The proposed project aims to develop a client – server based software. This software would satisfy various family needs. The client for this application will be an application which will be run on android devices – smartphones or tablets. The server will be handling all communication between clients and database. Both client and server will be coded in JAVA.

A Passable Project will:

(I) Provide a technical review of development methods and technologies
(ii) Develop a client-server Family Centre Application with basic/popular functions/features such as family member tracking and chat
(iii) Make a workable application for both of the client and the server
(iv) Test the application

A First Class Project will:

(I) Develop Family Centre Application with additional features such as SOS
(ii) Develop a relational database for storing the data needed by the client
(iii) Connect the database with the Application
(iv) Implement security elements into the application (Authorization, Authentication, connection encryption and password hashing)
(v) Evaluate the developed application

Reading List:

Walls, C. 2015. Spring boot in action. New York: Manning Publications.

Singhal, M. and Shukla, A. 2012. Implementation of Location based Services in Android using GPS and Web Services. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2.

Al-Mazloum, A. and Omer, E. and Abdullah, M. F. A. 2013. GPS and SMS-Based Child Tracking System Using Smart Phone. World Academy of Science, Engineering and Technology, Vol:7, No:2.

Resources Required:

- PC.
- Intellij – IDE, java JDK, Android JDK,
- Database software.

Marking Scheme:

	Marks
Introduction	10
Background, research, chosen solution	10
Analysis	10
Design	20
Implementation	30
Testing	5
Conclusion	10
Self-evaluation	5

Signed:

Student	**Supervisor**	**Moderator**	**Year Leader**
Radoslaw Burkacki	**Ying Liang**	**Michael McCready**	**Henry Hunter**

IMPORTANT: ***By signing this form all signatories are confirming that any potential ethical issues have been considered and necessary actions undertaken and that Mark Stansfield (Module Coordinator) and Malcolm Crowe (Chair of School Ethics Committee) have been informed of any potential ethical issues relating to this proposed Hons Project.***

Acknowledgments

I would like to thank my supervisor Ying Liang for her guidance, support and kind advice throughout my honours project.

I would also like to thank all the participants which have completed the questionnaires based on which the research and the evaluation was made.

Abstract

The main aim of this project is to develop software which will be specifically designed to be used by families. Application will be based on client-server architecture and will use an Android application as the client and the JAVA server as the back-end software. The main goal of this software is to increase security of family members (especially children) and allow family member to use a range of functions including family member tracking, chat and SOS. Application supports various settings which allow users to customize the application to their needs that includes languages setting and marker colours setting. Each family member must register, login and then they are able to create or join new families. Family members can use the functionality of the application only on their family members.

Contents

1 Chapter 1: Introduction

In recent years' new technologies became available to people more than even, technologies like Internet and GPS are available to everyone that has a smartphone. Those technologies can bring many benefits for families like security or communication. Usage of application which will be created during this project will bring security to their family members (for example children), family members will be able to check location of other family members at any time, an SOS button will be available in the application which will notify other family members that something wrong has happened to the user that has pressed it. In this project I am going to do a research on technologies that can be used to develop a family centre application which will be specifically designed for families, research market and gather data from potential users and analyse it then I am going to fully design, implement and evaluate the application.

1.1 Project background

Family Centre Application is dedicated for family usage. This application will include authorization and authentication this means that before users will be able to use the application they will need to register using the mobile application, they will be able to choose family and then they will be able to use the features of application. The main screen of application will be showing a map with markers on it, each marker will be responsible for showing location of each member of the family. The map on main screen will be implemented using a map API.

1.2 Project aims

The project has four major aims:

- Discuss and research technologies that are suitable for this project
- Research if there is a need for application and what functions and aspects are the most important for potential users
- Choose technologies which will be used to develop the Client-Server Family Centre Application
- Analyse and Design Family Centre Application
- Implement application using chosen technologies- client and server
- Evaluate and test application

A literature review and technical review will be produced, they will include research about technologies which can be used for the project, it will compare two technologies which will be suitable for the project and discuss differences between them, and then choose technology that will be used for Family Centre Application. An object-oriented programming language must be chosen for development of the application. A research will be made. The aim of the research is to find out if potential users feel that there is a need for this type of application, find out what mobile platform is being used the most and what functions and aspects of mobile application are the most important for potential users. System requirements will be based on outcome of analysis of data collected.

1.3 Project objectives

- Research similar applications
- Research technologies that are suitable for the project
- Research market
- Literature review
- Technical review
- Research
- Analyse data gathered
- Identify system and user requirements
- Choose technologies that will be used to develop Family Centre application
- Analyse and design Family Centre application
- Implement application
- Test and evaluate application

1.4 Structure of report

This document will be split into ten chapters. First chapter – introduction will give a brief introduction to the reader about the project and set aims and objectives for the project.

Chapter two will contain a Literature review and technical review which will focus on technologies that can be used for the project and selection of technologies which will be used for implementation. It will include theoretical review, market review and discuss similar applications.

Chapter three will be about research. It will do a brief discussion about research methods and then discuss research methods used in this project. A research method will be chosen and distributed, data gathered will be analysed.

Chapter four will be focused on analysis of the application, various diagrams will be used to demonstrate how all the functions will work within the family centre application and user requirements will be listed and discussed.

Chapter five will be focused on designing the application, client – server communication architecture will be designed and discussed, list of end points will be provided with detailed explanation of each of them, designs of screens will be created and discussed, application wireframe will be created to illustrate how the application will be used and the database will be designed.

Chapter six will be about implementation of the software, details about implementation process will be provided in this chapter including list of functions with description of client and server sided software, communication patterns and each class will be discussed.

Chapter seven will focus on testing of the software, tables with testing data will be created.

Chapter eight will focus on evaluation of application, a questionnaire will be sent out to participants to find out about user experience and what aspects of application can be improved.

Chapter nine will contain conclusion of the project.

Chapter ten will contain self-evaluation.

2 Chapter 2: Literature review

2.1 Introduction to Chapter

There are many programming technologies that are being used by programmers now days. Selection of appropriate technology that will be used to develop software is a crucial step for a project that involves software development. There is no single programming technology that suits all requirements each technology is used for certain problems. The aim of this chapter is to: do literature review, review similar applications, research market, theoretical review, technical review and choose technologies which will be used for this project.

2.2 Application review

This review will contain information about several systems which has been identified as like Family Centre Application. It will include basic information about the software like targeted platforms, features and versions.

2.2.1 My Family

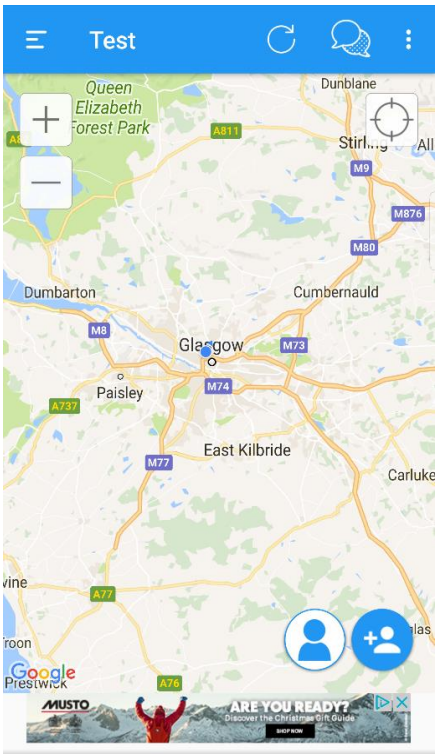

Figure 1- My Family application screen shot

This application is available for multiple platforms – android, IOS, windows phone and windows desktop. "My Family" does not require user to register. It allows users to create a family and add new family members to it. Two versions of the application are available -standard and premium. Standard application has limited features which include tracking and chat, advanced features are available only to premium members which costs £14.99 a year. It allows users to join to the family using a code which is available only to the user which has created the family. My Family is using google Maps API to show location of users, it has easy to use interface which can be adjusted. It has a wide range of settings which include change of theme, location timer setting, advertising and language. It is available in 18 languages although some issues with translation has been spotted as text sometimes is in Russian.

2.2.2 Family Locator

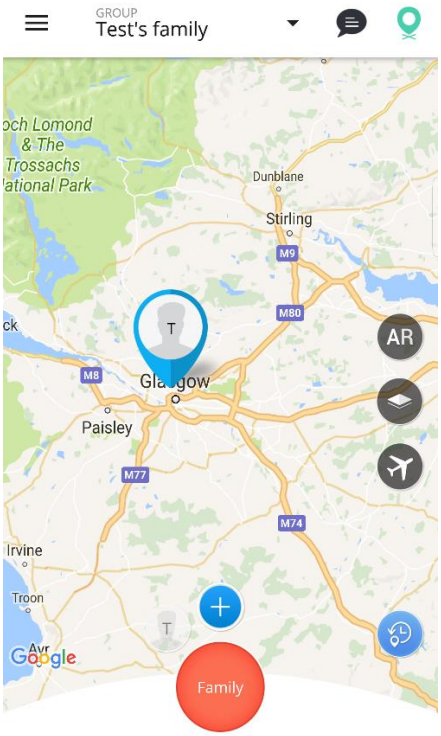

Figure 2- Family Locator application screen shot

Family Locator is available for two platforms – android and IOS. There are two versions of this application standard and premium, premium costs £9.99 for year and £12.49 for lifetime. Its features include chat, checking in, zone alerts. Premium members can use advanced features it includes real-time tracking which is not available in standard version. It uses google MAPs API for locating users. It has an easy to use interface. User can focus on each family member by pressing the image of user.

2.3 Market share

I have identified two mobile platforms which are dominating the market - Android and IOS, together they dominate the market with 99.1% market share. Android is the most popular mobile system with 86.2% market share and growing. IOS owns 12.9% of market share as of 2016. [Miler 2016] Family Centre Application must be available to as many people as possible due to this it should be developed for both systems although due to time constrains one platform must be chosen.

Operating System	2Q16 Units	2Q16 Market Share (%)	2Q15 Units	2Q15 Market Share (%)
Android	296,912.8	86.2	271,647.0	82.2
iOS	44,395.0	12.9	48,085.5	14.6
Windows	1,971.0	0.6	8,198.2	2.5
Blackberry	400.4	0.1	1,153.2	0.3
Others	680.6	0.2	1,229.0	0.4
Total	**344,359.7**	**100.0**	**330,312.9**	**100.0**

Figure 3- Market share

2.4 Theoretical review

2.4.1 Systems Development Life Cycle(SDLC)

Software development methodology is an approach to software development, each software development methodology has its own style, philosophy, pros and cons. Before the design stage can begin a software, development method must be chosen to ensure that development is well planned and structured. Some of popular development methods include: Waterfall, Agile, Scrum and Spiral model.

2.4.1.1 Waterfall Model

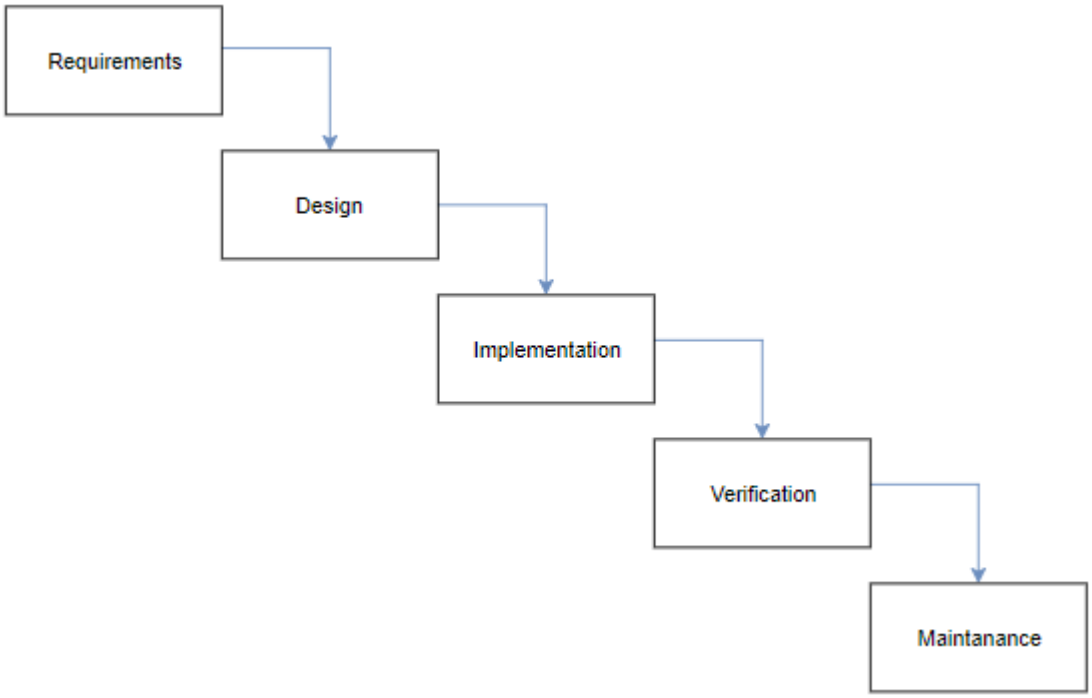

Figure 4- Waterfall model

The waterfall model is also known as linear-sequential lifecycle model. It follows a simple structure of phases, result of each phase is moved down to next level of development, before moving onto next stage the previous one must be completed. At the end of each stage it is necessary to check if work which has been done so far is according to the plan. Original model did contain 5 stages – requirements, design, implementation, verification and maintenance. The development of software using waterfall model begins with requirements stage, at this stage all detailed requirements must be captured. It is the most important stage of this model because if this stage will be carried out incorrectly then the benefits of this approach will be lost, project may fail because of that. Benefits of Waterfall model include:

- Simple and easy to understand
- Well documented
- Clearly defined stages
- Good for small projects where requirements are well defined

The disadvantages of this model are:

- No software is produced until implementation stage
- Bad for long and ongoing projects
- Not good for projects with requirements which can be changed in future (Kienitz, 2017)

2.4.1.2 Agile Model

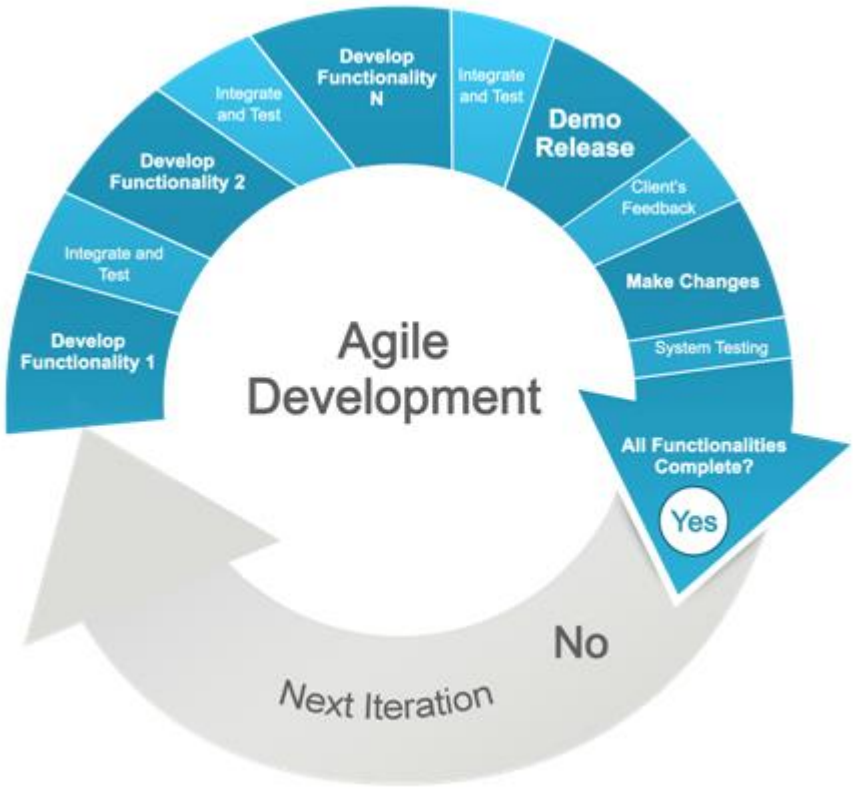

Figure 5- Agile model

The Agile software development model is very flexible; it is an opposite of waterfall model. Agile manifesto has been created by seventeen individuals which felt that there is a need for need for "an alternative to documentation driven, heavyweight software development processes" (Highsmith, 2001). Agile projects are completed in many iterations each iteration produces a separate part of a software.

Agile's manifesto values:

- Individuals and iterations over processes and tools
- Working software over documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan (Denning, 2016)

2.4.2 Object oriented programming

Object Oriented Programming (OOP) is a unique way of programming, it is used to process massive number of users. First appearance of object-oriented programming was in late 1950s at MIT. This way of programming is based on objects, software developer must create class which stands for object like user and then allocate appropriate attributes to them, when it is needed to create new user a new instance of a user can be created. OOP is using encapsulation for security of classes which means that attributes cannot be changed directly they have to be changed using function which has been predefined, variables inside of a class do have private status while the functions are public. Objects can inherit attributes and functions from other objects. Many languages support OOP like: C++, Python, c#, Java and many more. (Rouse, 2008)

2.4.3 Client – Server architecture

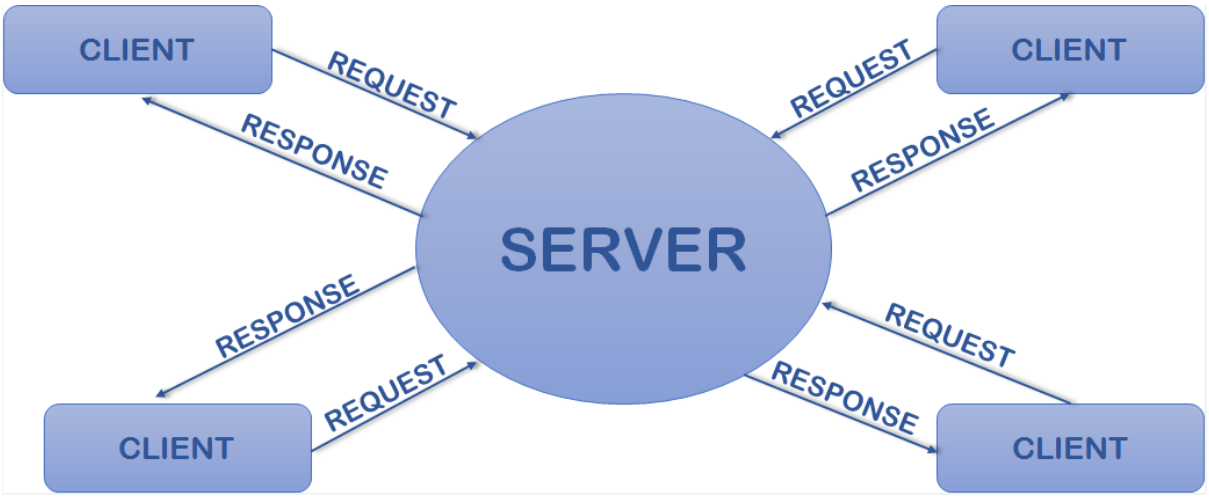

Figure 6- Client - server architecture

Client-server architectures is a distributed application structure. System which is using client-server architectures is based only on clients and servers. Clients and server are communicating via network in request-response pattern, first the client sends a request and then server is sending back response. Data which is being sent over the network is being send in JSON or XML. (Singh, 2017)

2.4.4 Version control tool

Complex software development projects will contain thousands of lines of code, in a real-life project many different developers will make changes to the code, it is essential to use version control software which will keep track of all the versions of code, information about which developer made changes to the code and to manage the code.

GitHub has been identified as one of the leaders in this field, they allow for creation of free accounts which then can use their services. GitHub allows its users to create repositories and upload code to it, all the versions of code which have been uploaded to GitHub can be seen at any time. GitHub provides two versions of software command line based and GUI, these tools can be used to manage the code and push(upload) and pull(download) the code, although it's also possible to use GitHub via their web site.

Family Centre Application will use GitHub as the version control software, each major update to the software will be uploaded to GitHub, 2 repositories have been created and they will be used to store code of Family Centre Application.

Family Centre Android Application is available on GitHub here:
https://github.com/RadoslawBurkacki/FamilyTrackingApplication

Family Centre Java Server is available on GitHub here:
https://github.com/RadoslawBurkacki/familycentre

2.5 Technical review

At the initial stage of the project it is important to decide what technologies are going to be used to create this project. Research has involved investigation into many technologies which are crucial for this project – server (backend), client(mobile), communication, map API, authorization and database.

2.5.1 Server (backend)

Two technologies have been identified which can handle Family Centre Application requirements as a backend – JAVA and C#. Both of those languages can process data fast and in real time, they are both object-oriented languages, they are high level languages and each of them has many frameworks. As Peter Sestoft said *"The Java and C# programming languages are managed languages and very similar: same machine model, managed platform, mandatory array bounds checks and so on"* (Sestoft, 2007), those languages are very similar, although one of the key differences is that JAVA is platform intendent, which means that it can be run on many systems like Linux or Windows while C# is tied to Microsoft platform only. Software which has been created using java must be run using Java Virtual Machine (JVM), JVM it is an abstract computing machine which allows computers to run java programs, thanks to JVM java is platform independent as JVM can be installed on most of the operating systems.

2.5.2 Client (frontend)

The other standalone part will be a mobile application. Before selection of technology which can be used to develop application, a market must be targeted, outcome of previous chapters has stated that android is the most used mobile operating system with 86.2% market share while IOS has 14.6%. Android applications can be coded using many programming languages like JAVA, C++ or Python. [Handy 2012].

2.5.3 Communication

Two communication file types have been identified, they are commonly used – JSON and XML. XML is more than just a communication file, it has many frameworks which can be used for many purposes while JSON is just a data format which is used for exchange of data. XML parasang process can be slower than JSON because DOM manipulation libraries require more memory to process large XML files. (Joshi, 2017)

2.5.4 Map API

Map API is a framework provided by its creator like Google, it is a set of tools that allows other developers to use services. To develop Family Centre application, it will be needed to use a map API which will allow to use a Map framework, Family Centre Application will need to show location of each family member on map. During the research I have identified two map API providers which are leaders in this market, Google Maps and Microsoft Bing Maps. Both are free to use. I haven't found many differences between them, the only difference which has been identified is that Google Maps has better documentation of their API. Android SDK does include a build-in templates for implementing Google Map API this can be used when creating the application.

2.5.5 Authorization

Each user of Family Centre Application will need to register and then login before using the application. To increase the level of security application will use JWT (Json Web Tokens). For the first time when user is logging in, android application will send email and password to server and if credentials will be correct then a token will be generated and sent back to the client, from that point client won't need to send password over the internet each time communicating, but a token will be used instead. Token has it's expire time so after some time generated token won't be usable, so it will need to be regenerated and again sent to user when logging in. (Roetman, 2017)

2.5.6 Database

Database is the most important part of a dynamic web application; it allows to store the data which has been gathered from users. Most commonly used databases are relational databases and NOSQL (Not Only SQL) databases. Relational databases are using tables to store data and they require detailed database model. Relational database uses columns and rows, each column is storing a pre-defined data and each row contains a unique instance.

2.5.7 Database security

Security of the data which is being stored in database is very important. Passwords which are stored in database cannot be stored as a plain text because in case if the database gets compromised then all the accounts which have been created for the Family Centre Application will be compromised.

2.5.7.1 Password hashing

Hashing is a type of algorithm which uses a mathematical function which expects any size of data to be passed into it and then it turns it into a fixed-length data. This process is not reversible. When user registers to use the Family Centre Application data is being sent from android application to the server, server is using hash function to hash the password then it's being saved to database.

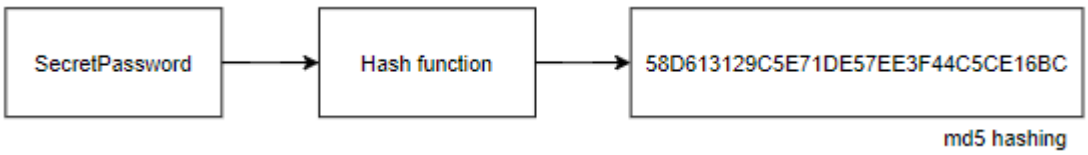

Figure 7- Hashing

There are many types of hashes like MD5, SHA0, SHA1, SHA2, SHA3 and Bcrypt. Each hashing type has its own hashing function and each of them will generate a different result for the same input.

2.5.7.2 Password salting

Password salting is a technique of adding additional data to the password before hashing it. This will protect the password from dictionary and rainbow table attacks.

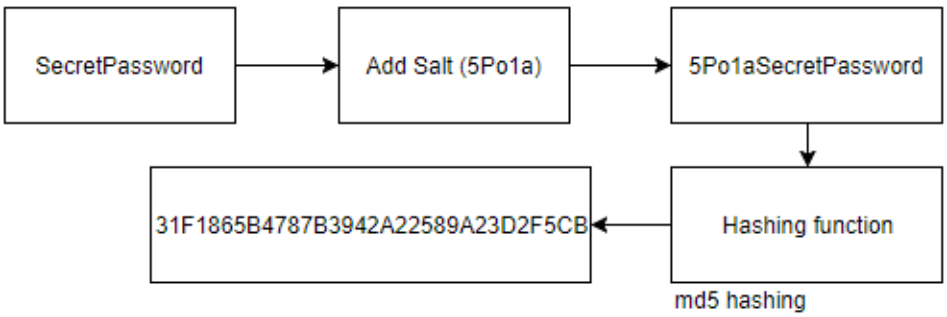

Figure 8- Hashing with salt

2.6 Selection and justification of techniques to be used for this project

Server

Family Centre Application will use JAVA version 1.8 for the development of the server. I have chosen to use JAVA because of its benefits which are:

- Object-oriented language
- Platform independence
- Performance
- Many frameworks and third-party libraries available

The server will also use Spring framework which is available for java projects. Json Web Tokens(JWT) will be used on server for authentication purpose. JWT will generate token for each user which has attempted to login with correct credentials.

Client

The client which will be used for Family Centre Application is going to be a mobile android application. The main reason for this choice is because android is dominating mobile market, most people are using android based mobile devices. Google has reported (Google, www.developer.android.com) that 99.3% android devices are running Android SDK version 15 (Ice cream Sandwich) or greater due to that android SDK version 15 will be used for development of android application, it will allow Family Centre Application to be compatible with 99.3% Android Smartphones.

Communication

Client – server communication will be done using an REST API which will be run on server. REST API will be used because it allows for easy future expansion of the application, for example desktop application, browser version or other mobile platform application can be created using REST API which will be developed for android application.

Communication between clients and server will be handled via JavaScript Object Nation(JSON). I have chosen JSON over XML because this project needs just a data-interchange format for communication which will be used for sending and receiving data. JSON has readable format which is important for testing stage and it can be faster than XML.

Map API

Google Map API is very well documented; this will be crucial as this project is depended on map API because main function which is family tracking will be based on it. It supports android SDK making it easier to implement map into android application due to that I have decided that Family Centre Application will use Google Map API to show location of users.

Database

Most of relational databases could have been used for this project. MySQL is globally known for being the most secure and reliable database system. (Branson, 2016) Family Centre Application will use database for storing: user data, last location coordinates and family details, it will do many queries to database, so the performance of database is important. I have decided to use MySQL because it is: free to use, open source, secure and has good performance.

Family Centre Application will use Bcrypt hashing algorithm to hash passwords which will be stored inside of the database. Bcrypt has a build-in random salting feature which means that random salt is being added to each password that is being hashed this means that It cannot be cracked by rainbow table attack because it is generating a random salt each time it is hashing password. It is slow – that is a benefit because if an attacker grants access to the database and run a brute force or dictionary attack then it will take forever to hash millions of passwords using Bcrypt.

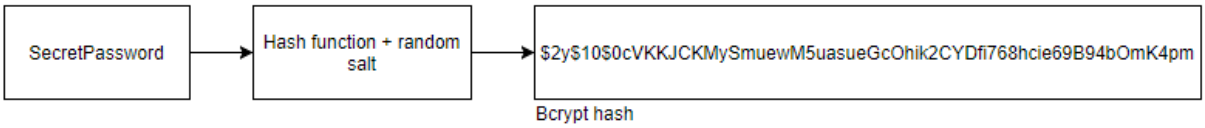

Figure 9- Bcrypt hashing

Bcrypt hash contains three separate parts which are being spited by "$" signs. First part which in this case is "2y" is the version of Bcrypt, second part is the cost parameter it specifies a key expansion iteration count as a power of two, and the third part is the hashed password with random salt.

Systems Development Life Cycle

Agile model has been chosen for this project because of its methodology. Software which is created using agile is created in iterations, so the software is created fast. Agile is very flexible so new functions can be implemented in to the software in future.

Systems Architecture

Family Centre Application will use Client-Server architecture. Client-Server architecture is using a centralized server, each client which will try to communicate with other clients will need to do it through the server. Data is stored on server which means that updates to data can be done very easily.

3 Chapter 3: Research methodologies

Research methodology is a process which is used to collect data about a subject, it is being used to gain a better understanding of the subject which the research is about.

3.1 Introduction to chapter

It is important to do a research at the early stage of the project. This chapter focuses on research, it will discuss two popular research methods qualitative and quantitative. Selection of a research method will be done in this chapter and it will be discussed. Research results will be provided as charts and results will be discussed, next chapters - design and analysis will be based on the outcome of this chapter.

3.2 Qualitative method

Qualitative method it is a way of gathering data from targeted people, involves talking to people in person and doing interviews to find out their opinion and experience, it seeks to answer and explain 'how' and 'why' questions. According to Saul McLeod (2008) "The aim of qualitative research is to understand the social reality of individuals, groups and cultures as nearly as possible as its participants feel it or live it. Thus, people and groups are studied in their natural setting." The data collection is done via observation and outcome of interviews. This method is the most time-consuming because it involves face to face meetings with targeted people. Qualitative data could include much more than just text it can be photos, videos and many more.

3.3 Quantitative method

The data which is produced by quantitative research method is always numerical and can be analysed using statistical and mathematical methods. The data is gathered using structured research instruments, main source of quantitative data is from surveys and observations, the data which is being recorded must be generalized. Main characteristics of quantitative method include:

- Clearly defined research questions
- Results as statistics and numbers

The disadvantage of this methods is the lack of communication with participant which is unable to justify his choices.

3.4 Research methods used for research

Research for this project will use quantitative method for data gathering. Questions will allow practitioners to answer questions by selecting options. A questionnaire was created using Google forms. It has been distributed to 48 participants via online surveys. The purpose of the questionnaire was to find out if potential users feel that there is a need for Family Centre Application, find out which features, and aspects of the application are valued the most by participants and what mobile operating system is being used by them. Questionnaire is available online: https://goo.gl/forms/U3QW86NzZZcfmrcD2

3.4.1 Questionnaire

Questionnaire is separated into 3 sections; each section has its own purpose. Section 1 is gathering data about participants, it is asking participants about age, gender and platform which they are using. Data gathered in this section will allow to filter data from next sections based on age, gender and platform of participants.

Section 2 contains a small description of Family Centre Application and asks participant if they feel that there is a need for this application. Based on outcome participant can be forwarded to section 3 or to the end of questionnaire.

Section 3 is about finding out what functions and aspects of mobile application are the most important for participants. This section requires participants to select on scale from 1 to 5 the importance of features or aspects of mobile application. Data gathered in this section is the most importance because it will allow me to know what to focus on and what to keep in mind during the development of Family Centre Application.

3.5 Research results

48 questionnaire responses have been received. The following charts have been generated from responses.

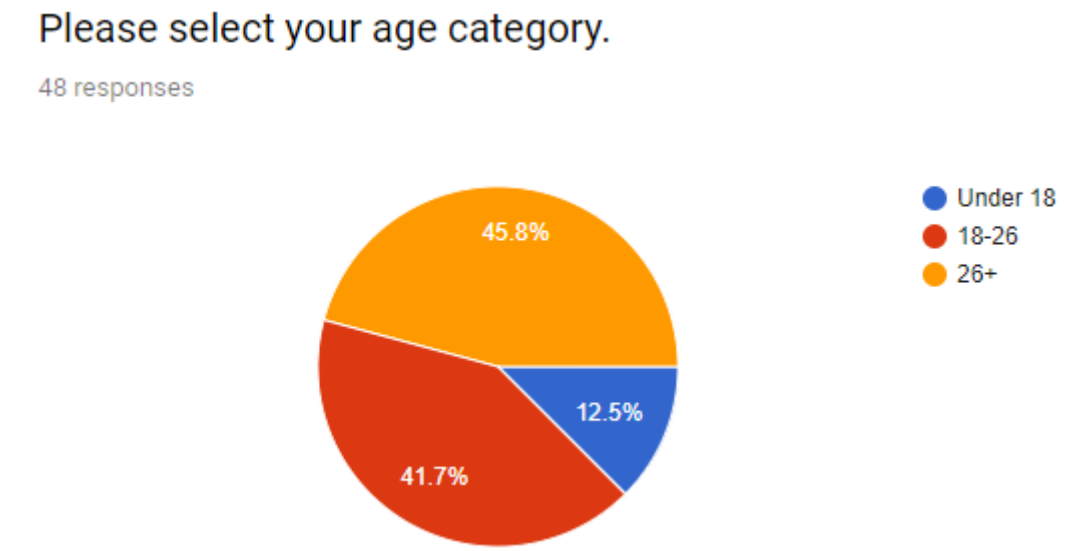

Figure 10- Questionnaire age result

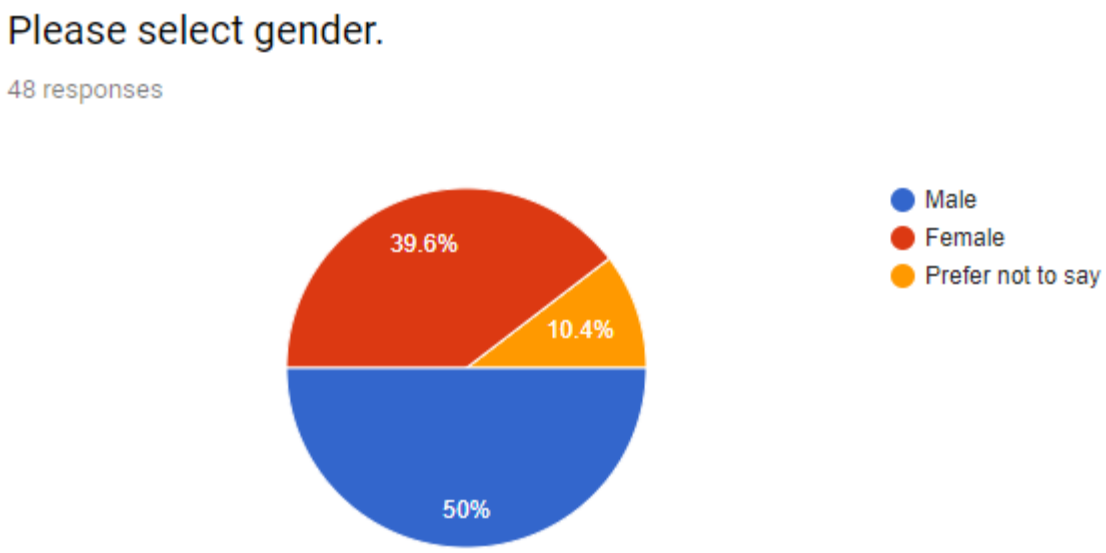

Figure 11- Questionnaire gender result

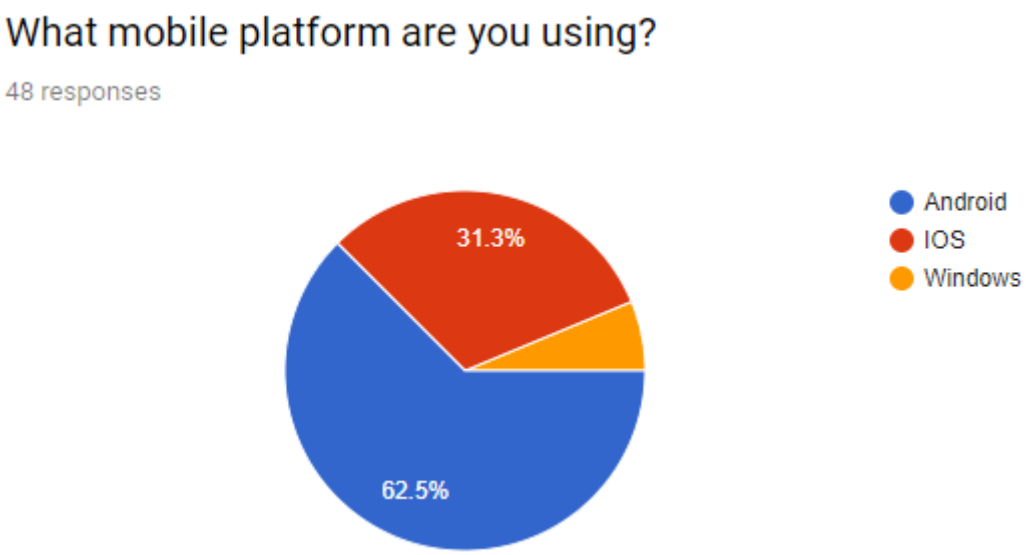

Figure 12- Questionnaire platform result

24(50%) of the 48 participants were male while 19(39.6%) were female, 5(10.4%) participants preferred not to say. 22(45.8%) of 48 participants were above 26 years, 22(41.7%) people have chosen category between 18 and 26, and only 6(12.5%) people were below 18 years old.

30(62.5%) participants out of 48 have stated that they are using Android mobile platform, 15(31.3%) participants have stated that they are using IOS mobile platform and only 3(6.3%) people are using windows mobile platform. Group of people which are using Android system is the largest containing 62.5% of participants.

Section 2 asked participants if they think that there is a need for Family Centre Application. A brief description about Family Centre Application have been displayed below the question.

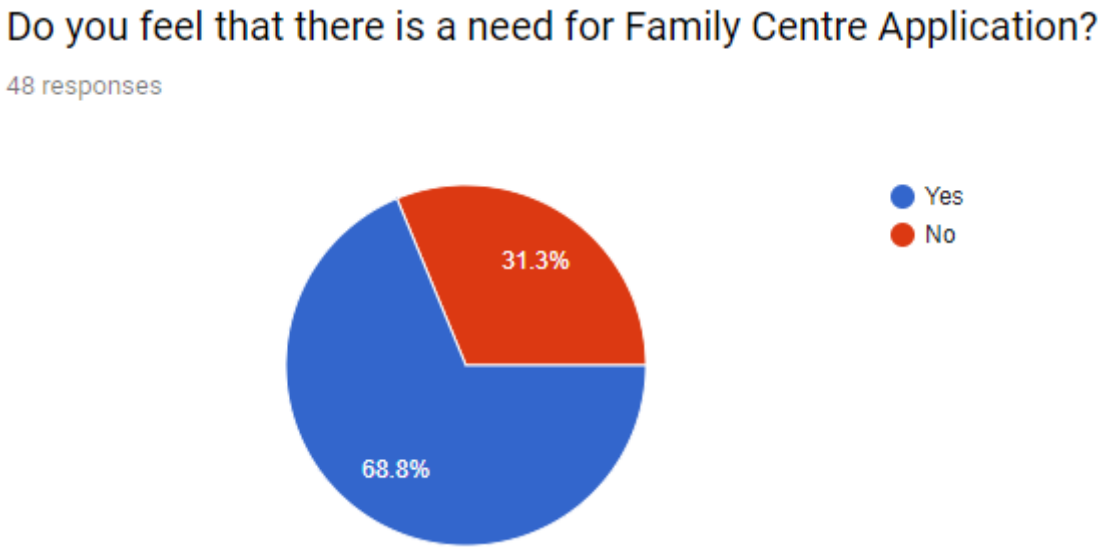

Figure 13- Questionnaire result

33(68.8%) have stated that they think that there is a need for this application. Participants which have selected yes to this question were taken to section 3, while 11(31.3%) participants which have answered no, have been moved to submit screen.

Section 3 asked participants which functions, and aspects of application are the most important for them. This section uses scale from 1-5. 1 is the least important for user and 5 is the most important. 33 participants have completed this section.

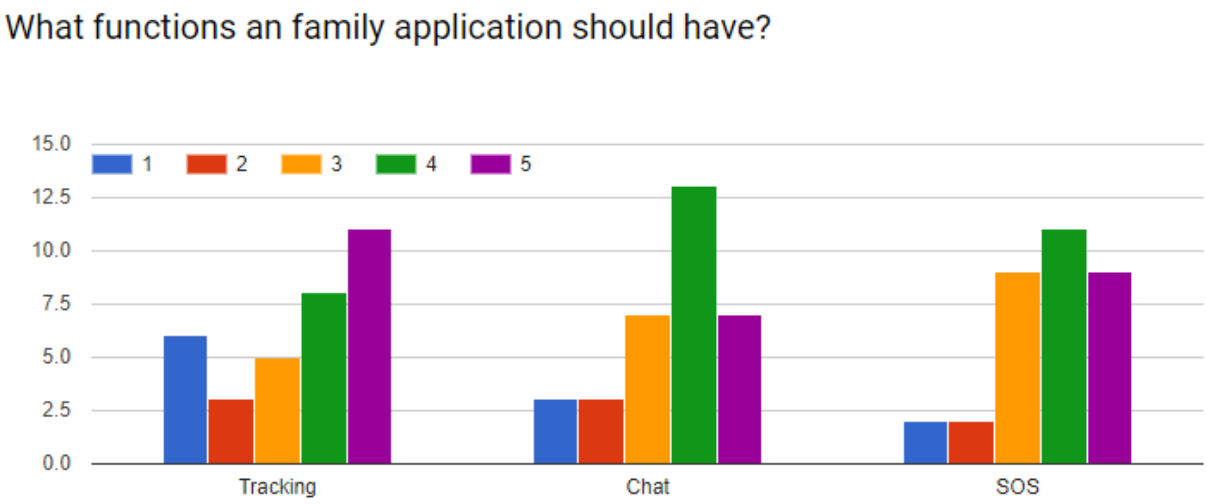

Figure 14- Questionnaire function results

Question 4 asked participants about functions which a Family Centre Application should have. This question allowed participants to use scale from 1 to 5 and they had to choose a number for each function.

The following table represents result of questionnaire. It is divided into 3 columns, 4-5(important), 3 (neutral) and 1-2(least important).

	4-5	3	1-2
Tracking	19	5	9
Chat	20	7	6
SOS	20	9	4

Table 1- Question 4 results

According to the table above the most important functions for participants are Chat (20) and SOS (20) followed by tracking (19).

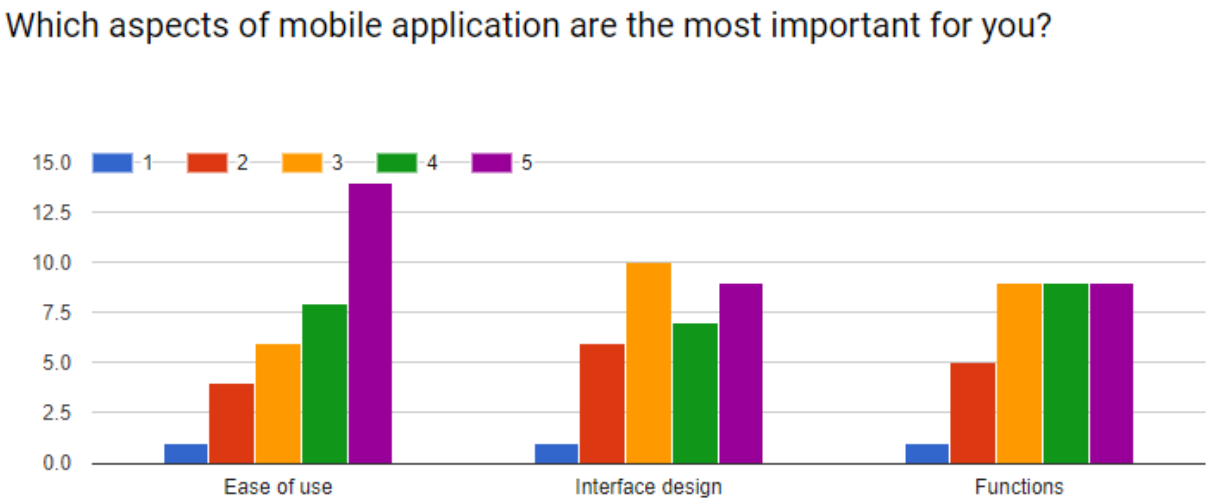

Figure 15- Questionnaire aspects results

Question 5 asked participants about aspects of mobile application. This question allowed participants to use scale from 1 to 5 and they had to choose number for 3 aspects.

The following table represents result of questionnaire. It is divided into 3 columns, 4-5(important), 3 (neutral) and 1-2(least important).

	4-5	3	1-2
Ease of use	22	6	5
Interface design	16	10	7
Functions	18	9	6

Table 2- Question 5 results

According to the table above the most important aspect of mobile application for participants (22) is Ease of use, then functions (18) and interface design (16).

4 Analysis

4.1 Introduction to chapter

In this chapter Family Centre Application will be analysed, to achieve that UML (Unified Modelling Language) will be used to create various diagrams which are widely used in software development to describe how system will work, what order are actions handled and what is the relationship is between the classes.

4.2 User requirements

To capture user requirements a questionnaire have been created via Google Forms and distributed to 44 participants. The main aim of the questionnaire was to find out if participants feel that there is a need for this application, but it also asked participants what features of the application they feel that are the most important for this type of application. Previous chapter has analysed the results of the questionnaire, last section of the questionnaire which asked participants about what functions are valued the most, a bar chart has been created and is available (figure 14).

Participants have stated that the most important function which family application should have is SOS (with 124 points), then Chat (with 117 points) and then Tracking (with 114 points).

The other question has asked participants, which aspects of the application are the most important for them, a bar chat has been created and is available (figure 15).

Participants have stated that the most important function which family application should have is Ease of use (with 129points), then Functions (with 123 points) and then Interface design (with 118 points)

4.2.1 Functional requirements

- Create relational database to store data
- Allow user to register and login
- Implement Authentication and Authorization systems
- Use Google Maps API to display family members location
- Application must be able to catch all errors, if they occur it must display details about error
- All-important alerts (user join family, SOS and chat message) must be displayed as notification with sound and vibration
- Creator of family must be able to remove other family members

4.2.2 Non-functional requirements

- Enhance security of the system via usage of tokens(JTW), data hashing(Bcrypt) and connection encryption(HTTPS)
- Accurate localization of each user
- Application must be easy to use
- All texts, input boxes and buttons must be easy to read and understand
- Client application available in two different languages
- Android application must be compatible with android devices which are using Android 4.0.3 or later version

4.3 Class diagram

Class diagram is widely used in software engineering. In the UML it is a static structure which is used to show classes, attributes, operations and connections between classes. I have identified the following classes, attributes and operations and created a class diagram based on that. It shows how the internal system of the Family Centre android application will work.

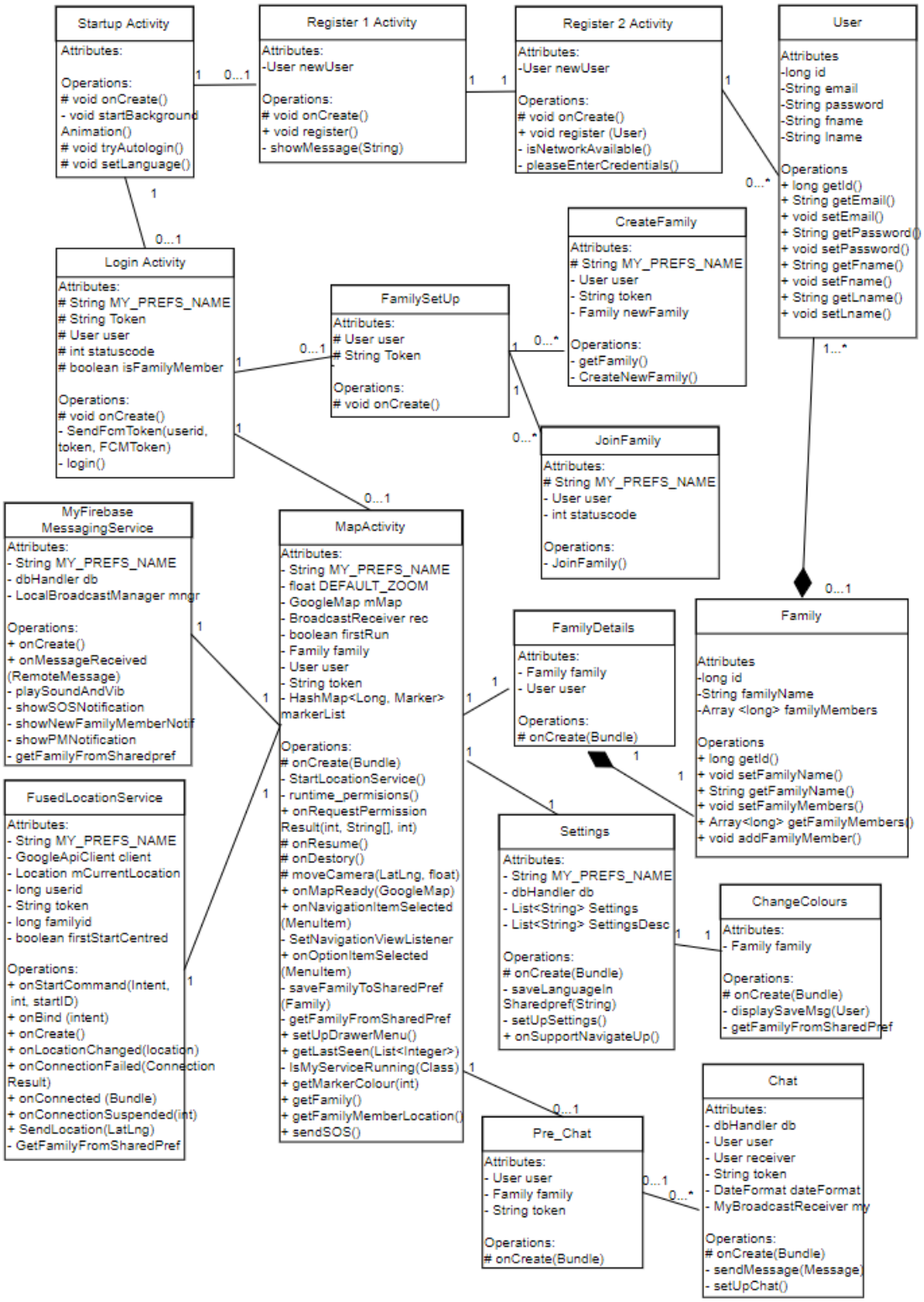

Figure 16- Class diagram

4.4 Use case diagram

Use case diagram is another part of UML which is used to show how basic user interactions with features occur and in what order. Use case diagram have been created for each function which is available to user within the family centre application.

Use case diagram part 1:

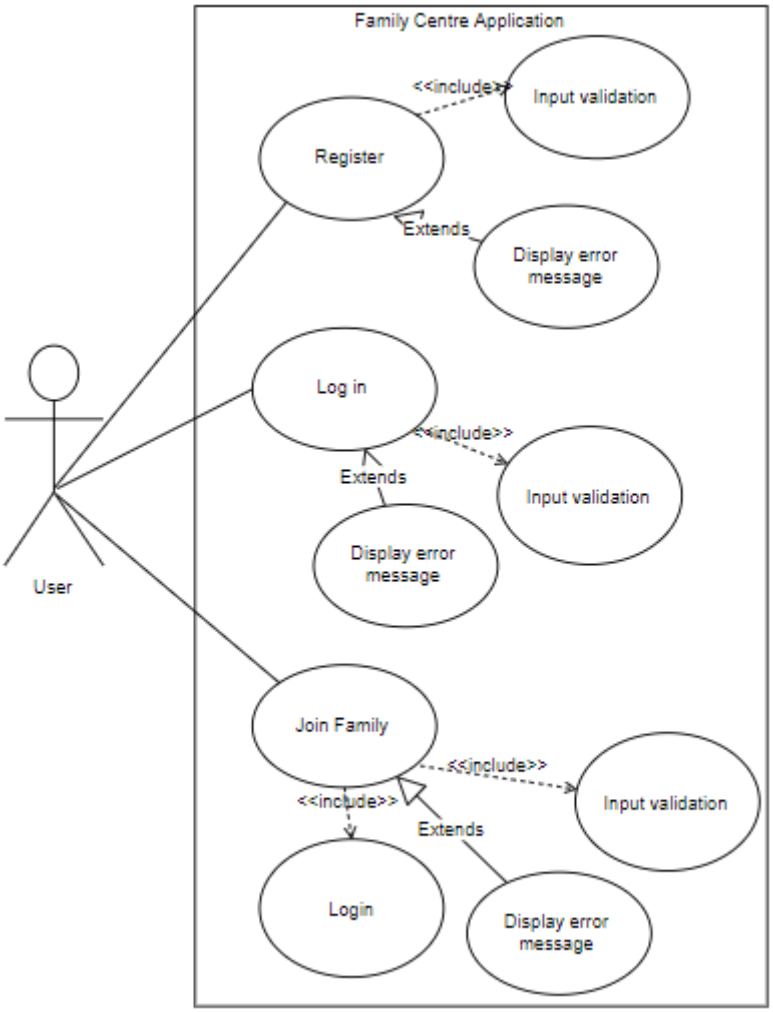

Figure 17- Use case diagram part 1

Use case diagram part 2:

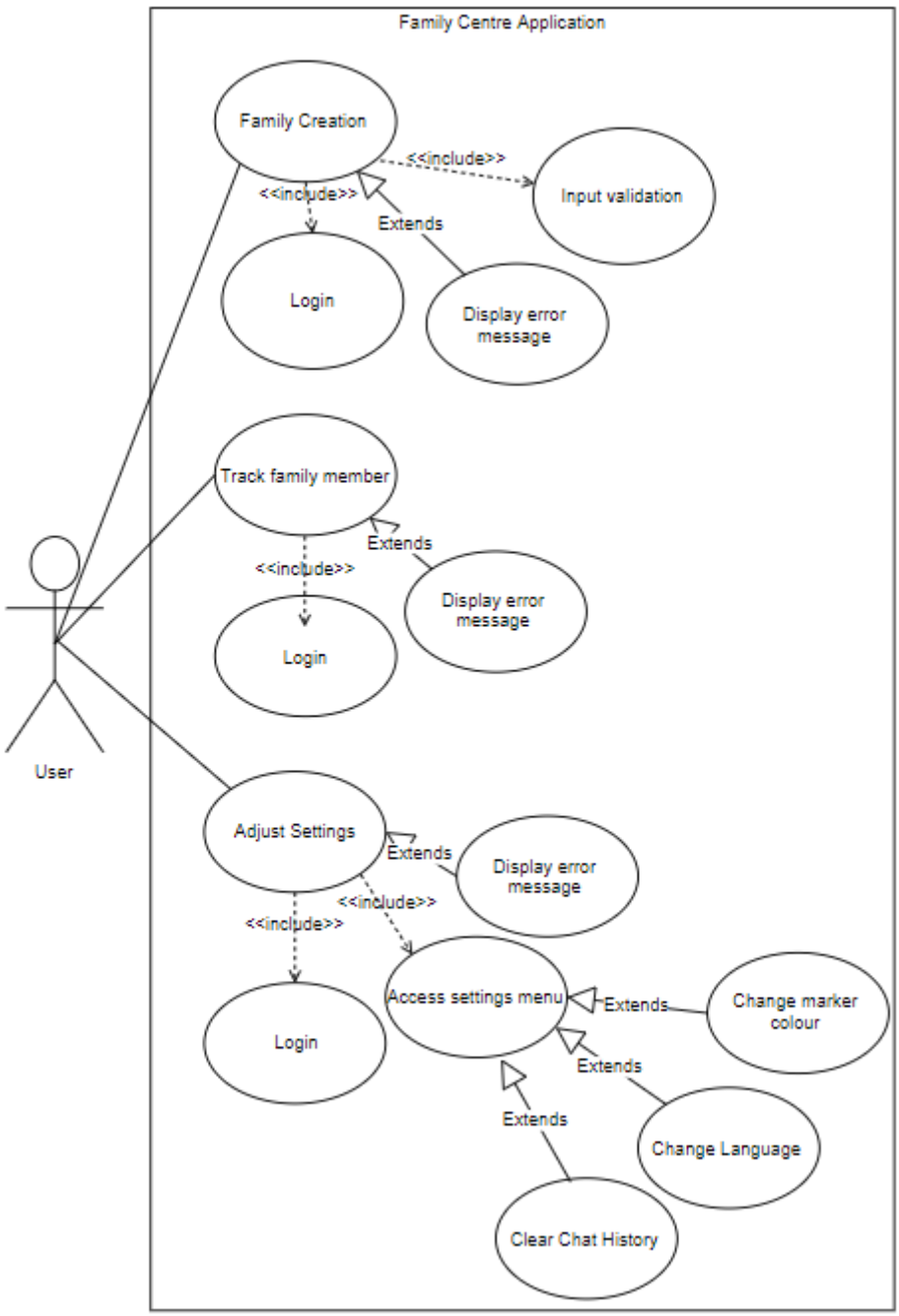

Figure 18- Use case diagram part 2

4.5 Sequence Diagram

Sequence diagram is an interaction diagram which is a part of UML and is used to shows how object interact with each other and in what order in time sequence (top to bottom). Each interaction is marked with a number, 1 is the interaction which starts the action. Sequence diagram have been used to show how basic functions of Family Centre Application work.

4.5.1 Register

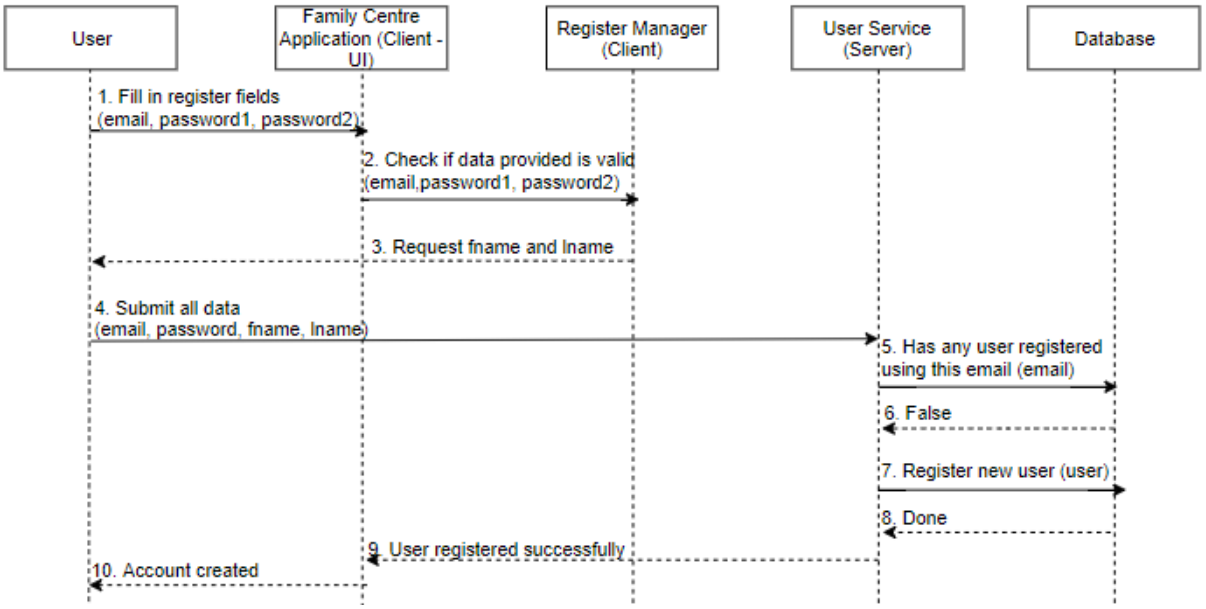

Figure 19- Register sequence diagram

1. User needs to start registration process by pressing "REGISTER" button, after that he will be moved to register part 1 screen which requires user to enter email, password and re-enter password, after that user needs to tap "NEXT" button.
2. Application will verify the data which has been provided and check if its valid, if the data is invalid then appropriate error message will be displayed if it is valid then user will be moved to next screen which is register part 2.
3. Register part 2 screen asks user to enter first name and last name.
4. After providing all required data user can tap on the create button, at this point the application is doing a POST request to the server and it is sending a JSON file which contains the data which has been provided by user in the register process, based on the response from the server an appropriate message will be displayed.
5. When server receives the request from client it checks in database if the email which has been provided in the request is not being already used by another user which would mean that this user already has an account.
6. If the response from database false which means that no account was found with same email.
7. Then server is adding new user to database.
8. Database returns message that new row was created.
9. Server sends response to client which states that user has been registered successfully.
10. "Account created" message is displayed.

4.5.2 Login

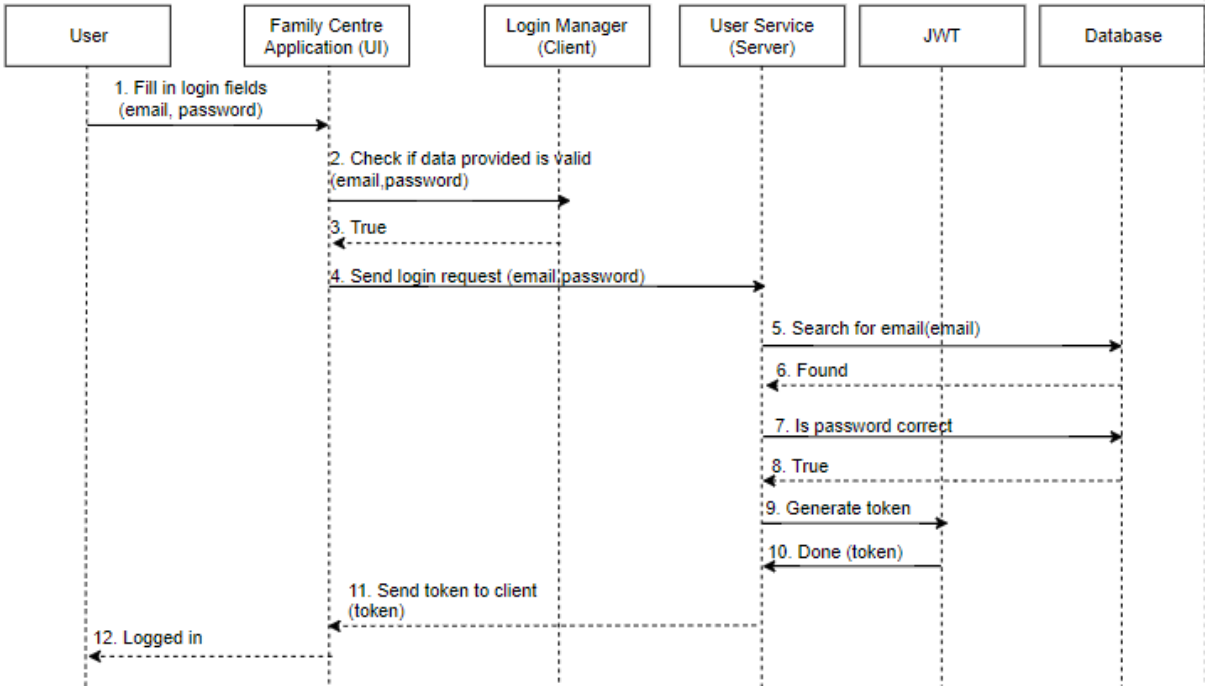

Figure 20- Login sequence diagram

1. User needs to start-up the application and tap on “SIGN IN” button, after that user will need to enter his email and password and tap on “SIGN IN” button.
2. Application will check if all fields have been filled and if the data is valid.
3. If the data is valid then
4. Application will send a POST request which sends data to server in JSON format.
5. When the server receives the request, it queries the database to check if the email is already in database.
6. If database found the email, then it returns true
7. Server is checking if the password which has been provided matches the password which is in database.
8. If that is true, then
9. Server is generating a JWT token which will be used for future communication.
10. Token is being returned.
11. Token is sent back to client.
12. User is logged in.

4.5.3 Create Family

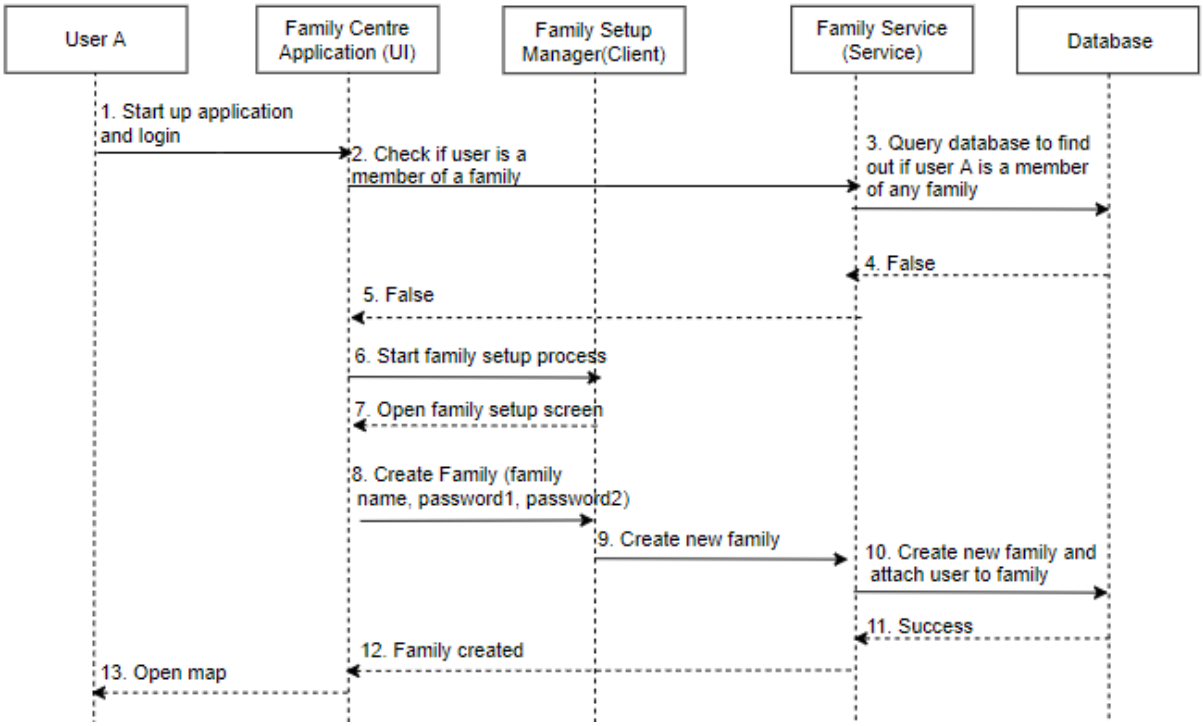

Figure 21- Create family sequence diagram

1. User needs to start-up the application and login.
2. Each time user logs in application is sending a request to server to check if that user is in any family.
3. Server receives request and it checks in database if this user is a member of any family
4. If not, then…
5. Server returns false to client
6. User is not a part of any family, starting family setup process
7. Opening family setup screen
8. User enters family name, password and re-enters password. Then user must press create new family button
9. POST request is sent to the server it contains JSON data with family details.
10. Server is saving new family and it attaches user to that family.
11. Database query successful
12. Server notifies client that family has been created successfully
13. Map is opened

4.5.4 Join family

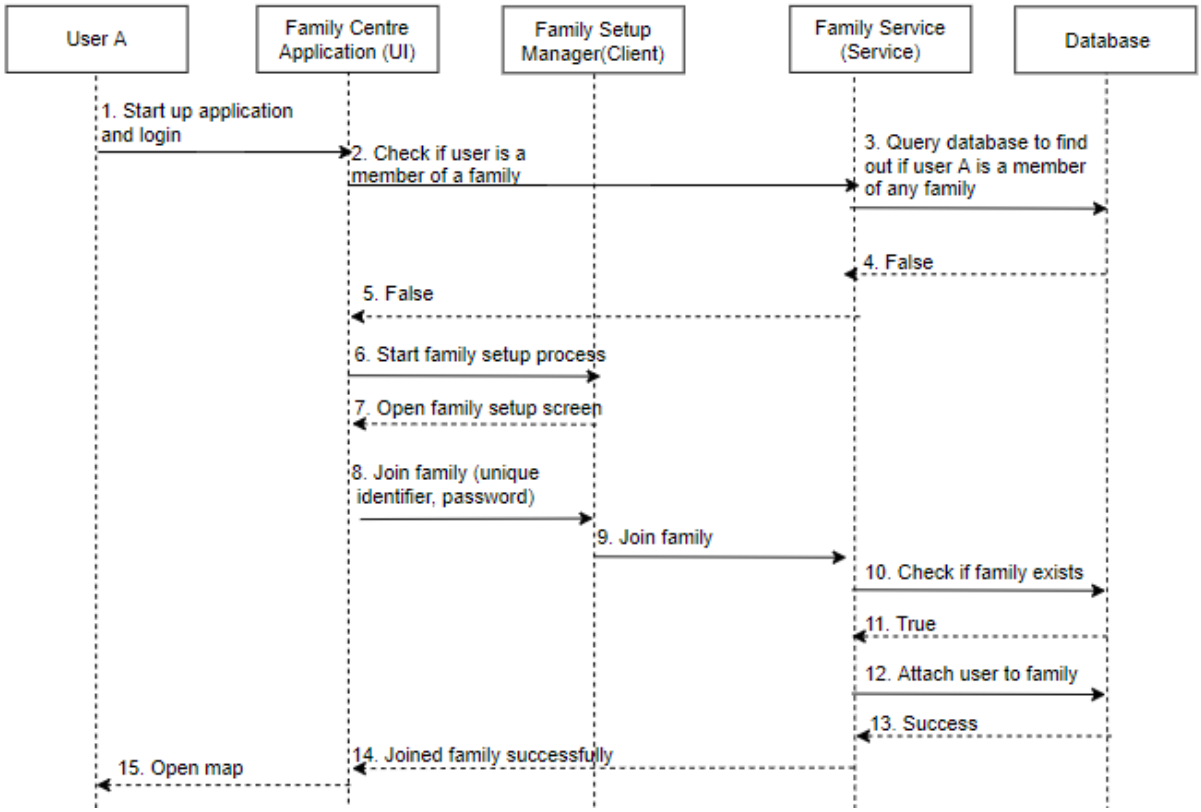

Figure 22- Join family sequence diagram

1. User needs to start-up the application and login.
2. Each time user logs in application is sending a request to server to check if that user is in any family.
3. Server receives request and it checks in database if this user is a member of any family
4. If not, then…
5. Server returns false to client
6. User is not a part of any family, starting family setup process
7. Opening family setup screen
8. User enters family unique identifier and password. Then user must press join family button.
9. POST request is sent to the server it contains JSON data user and family data.
10. Server checks is family exists.
11. Database returns true.
12. Server attaches user to the family, and it saves the relation.
13. Database returns success
14. Server sends response that user has joined family successfully.
15. Map screen is opened.

4.5.5 Send location coordinates

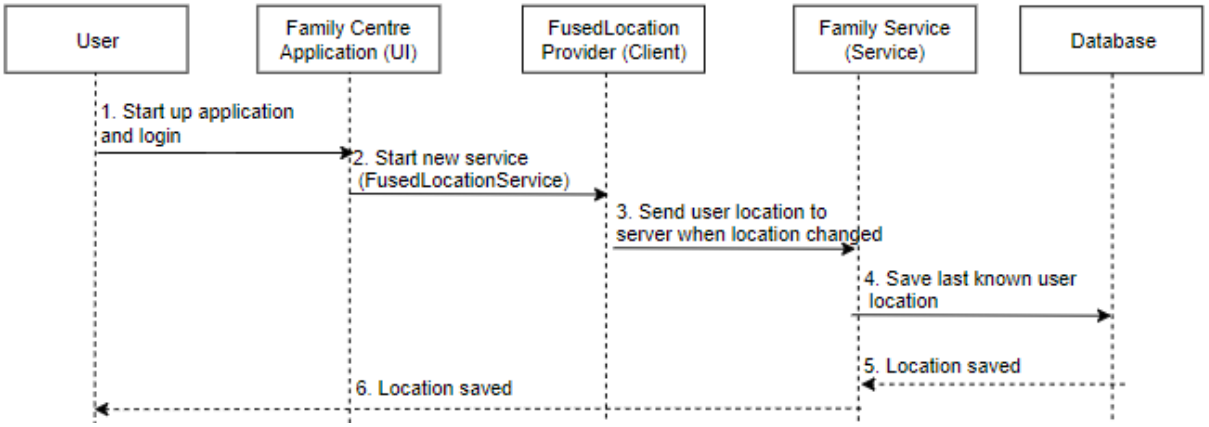

Figure 23- Send location coordinates sequence diagram

1. User needs to start-up the application and login.
2. After user logs in successfully new service is started. This service is used to send location to server each time user changes location, the coordinates are send in JSON file which consist of the longitude and latitude. To ensure that other family members can always locate this user, this service has been configured to work in any circumstances even when the phone is put to sleep, application is minimalised or even killed. This service can be only stopped by singing out of the application.
3. The service which has been started is sending data to the server each time the location of the device is changed.
4. When the server receives the location message it is saving the location to the database.
5. If the location was saved successfully then
6. Response is sent to the user which confirms that location has been saved.

4.5.6 Track family member

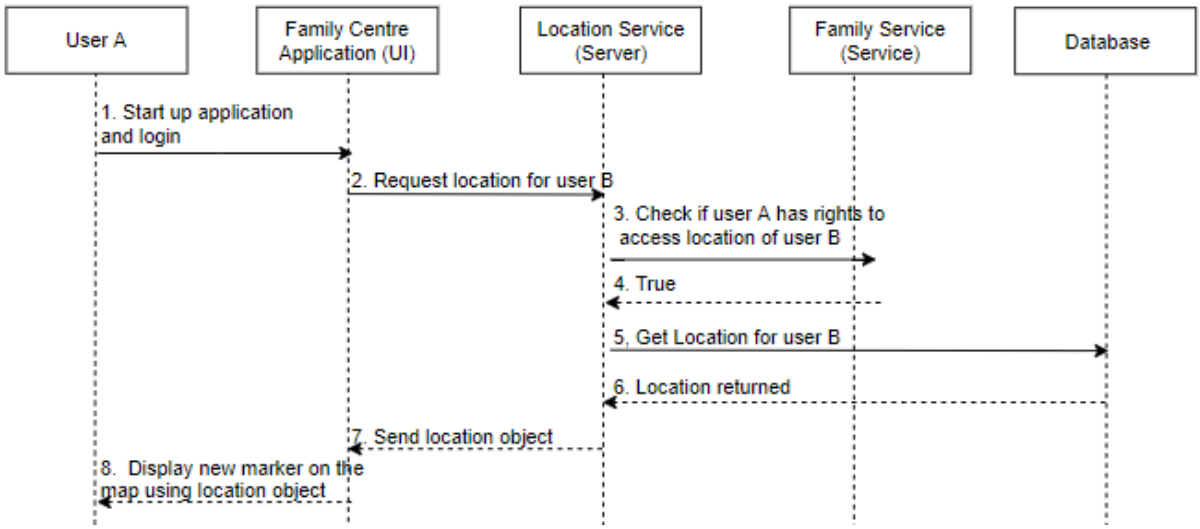

Figure 24- Track family member sequence diagram

1. User needs to start-up the application and login.
2. User needs to access the drawer menu, it can be done by taping button which is in top left corner or swiping from left to right. In the drawer menu there is a section called "Tracking ", in this section all users are listed, tapping on the name of the family member within the tracking section will display location of that user on the map. First a GET request will be sent to server.
3. When server receives the request, it will check if the user which is requesting location of other user is within the same family.
4. If that is true, then
5. Server will get the location of that user from the database.
6. If the database contains the location of that user, then it will return it.
7. Server will send location object to the client.
8. Application will create new marker on the map which will be representing the located user.

4.5.7 Send chat message

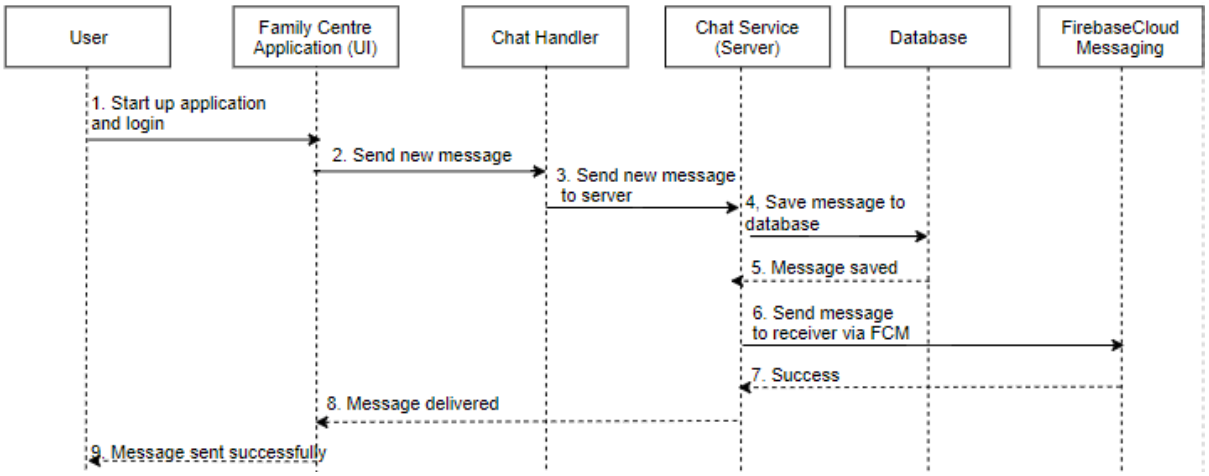

Figure 25- Send message sequence diagram

1. User needs to start-up the application and login.
2. User needs to access the drawer menu, it can be done by taping button which is in top left corner or swiping from left to right. In the drawer menu there is a row called "Chat", if user taps this row then the user will be moved to pre-chat screen which allows to select the receiver, after received is selected the chat opens, user needs to enter a message and press send button.
3. POST request is sent to the server, it contains JSON data which contains basic information about the sender, receiver and the message itself.
4. Server is saving the message to database.
5. Database returns "successful" message
6. Server sends a request to Firebase Cloud Messaging server and it passes the message object as a JSON and the receiver token.
7. Firebase Cloud Messaging returns success message
8. Client receives confirmation that message has been delivered.
9. Message sent successfully notification.

5 System Design

5.1 Introduction to chapter

This chapter of the report contains detailed description about the client – server architecture which will be used for this project, client and server architecture will be discussed and its functionality will be explained, each technology which will be used in either client or server will be listed and discussed. A table with all the endpoints which will be used for the communication will be provided with detailed explanation. The screen designs of the client application will be created and provided in this chapter and all the functionality of the user interface will be discussed.

5.2 Client – server communication

Client-Server Family Centre Application is using a centralised server which is handling all the requests from the client, it is used for all the functionality which is available to the client. The system is using two external services: Google Maps Services and Firebase Cloud Messaging which ensure that all the functionality works correctly within the Family Centre Application.

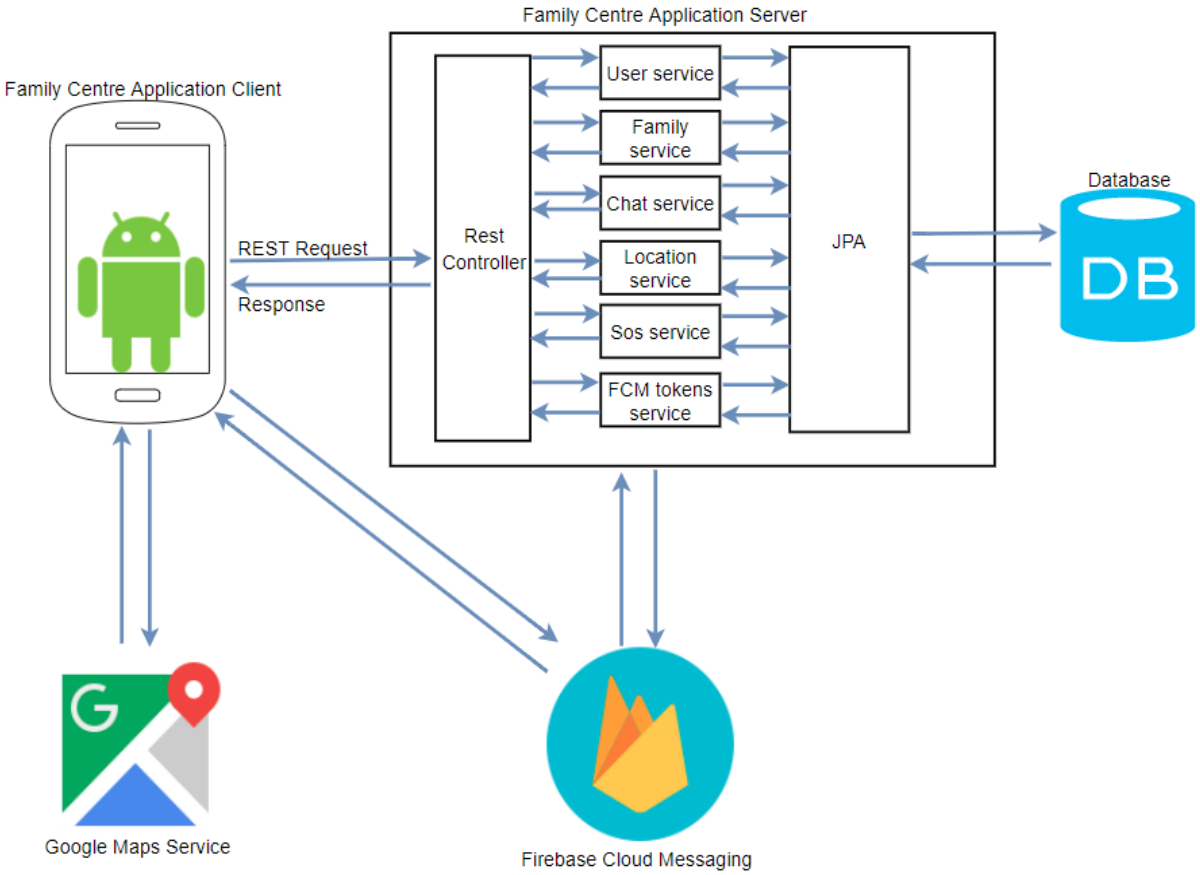

Figure 26- Client - server communication architecture

5.2.1 Firebase Cloud Messaging

Firebase Cloud Messaging service is used by both Client and Server, when client application is started it is getting a token from Firebase Cloud Messaging, then when user logins successfully to the application FCM token is sent to server and saved in database, this token is then user by Family Centre Application Server to send notifications and data to its clients.

5.2.2 Family Centre Application – Client

Android application will be coded in JAVA using IntelliJ IDEA IDE, it will be compatible with devices which use Android version 4.0.3 (Ice Cream Sandwich). Gradle will be used as the build system which will be used to import any external libraries, some of the libraries which will be used include:

- GSON – GSON is a library which has been developed by Google, it is used to translate JSON object into JAVA objects. This library is important for this project as all the data which will be sent from server to client and from client to server will be in JSON format, to use the data within the client it must be converted into JAVA objects, it can be achieved by using GSON.
- Okhttp – Okhttp is a HTTP client which can be used on Android devices to handle HTTP requests and responses. This client will be used within the android application for all the communication between the client and the server.

Android application will be communicating with 3 services in total, first one is the family centre application server which is responsible for all the functionality, the last two are the external servers which include the Google Map Services and Firebase Cloud Messaging.

5.2.3 Family Centre Application – Server

Family Centre Application Server will be coded using JAVA 1.8. The server will be created as a REST web service which will allow clients to communicate with it via HTTPS methods. Spring boot framework allows the developer to quickly create spring-powered applications which in this case is REST web service. In REST architecture all communication which occurs must be initiated by client, although for this project it is required to send important alerts to the client without their initiation, Firebase Cloud Messaging can be used by Family Centre Application server to send notifications and data to clients without initiation of communication by clients. When the server needs to send data or notification to its clients its sending a JSON request which contains tokens of users which the request needs to be sent to and the data, then all that data is sent to Firebase Cloud Messaging service which is passing the content of the request to the clients.

Maven is a build management tool which will be used on the server side to get all the libraries which will be used on the server side. Some of them include:

- MySQL connector java – it is a library which is used to connect java software to MySQL database.
- GSON – It has the same purpose as on the client, its used to convert JSON data to java objects,
- JWT – It's a library which will be used to manage and generate JWT tokens which will be used for the authorization.
- Okhttp – Okhttp client will be used to send requests to Firebase Cloud Messaging service from the server.
- Spring data – It's a JPA which allow for easy usage of database, all the queries will be done via Spring data

5.3 List of endpoints

This section contains all the endpoints which are used by the client to communicate with the server.

Endpoint	Method	Description
https://localhost:1000/users	POST	It's used to register users. Each request must provide a JSON body which will contain user object.
https://localhost:1000/users/{email}	GET	This endpoint is expecting an email to be passed in the URI. It's used to get user object from database, the database is returning user object which matches the email provided.
https://localhost:1000/users/	GET	Used to get list of all users. Returns JSON list of all users in the database.
https://localhost:1000/login	POST	It is expecting client to provide JSON body which contains of email and password. It is used to login user.
https://localhost:1000/families	POST	Each request must provide JSON body which must contain Family object. It is used to create new families.
https://localhost:1000/families/	GET	Used to get list of all families. Returns JSON list of all families.
https://localhost:1000/families/	POST	It is used to add user to a family. Each request must provide join family object which is used to link user to the family.
https://localhost:1000/families/{id}	HEAD	It is used to check if user is already a member of any family. It is expecting a number to be passed in the URI.
https://localhost:1000/families/by-user-id/{id}	GET	It is used to get family object via user id. It is expecting a number to be passed in the URI, the server is returning a family object to which this user is linked to.
https://localhost:1000/location /{userid}	GET	It is expecting a number to be passed in the URI which is a user id. It is used to get location of the user.
https://localhost:1000/families/ location/{userid}	POST	It is expecting a number in the URI (userid) and a JSON body which must be a LastKnownCoordinates object. This endpoint is used to save location of user.
https://localhost:1000/families/location/	GET	Used to get location of all users. Returns a list of all LastKnownCoordinates objects.
https://localhost:1000/sos/{id}	HEAD	It is expecting a number to be passed in the URI which is a user id. It is sending SOS to all family members of which are within the same family as the user which sent the request.
https://localhost:1000/chat/	POST	It is expecting JSON body which will contain Message object. It is allowing users to send messages.
https://localhost:1000/fcmtoken	POST	It is expecting JSON body which will contain FCMToken object. It is allowing clients to provide the FCMToken to the server.
https://localhost:1000/families/	DELETE	It is expecting JSON body which will contain Family Member object. It is used to delete family members from the family.

Table 3- Endpoints table

5.4 Screen designs

This section will contain basic screen designs of the application screens. Important controls have been split into three categories: buttons, text and input box. Controls which are buttons have been marked with red background, text controls which are used only to display text have been marked with green background and Input boxes which allow user to enter text or select options have been marked with blue background.

5.4.1 Start-up menu, login and register

Start-up screen includes two buttons which allow user to log in or register. Login and register screens are focused on getting data from user, user must select action (register or login) from start-up screen and then enter appropriate data, if registering then user needs to enter email, password, re-enter password, first name and last name and tap on create button, when logging in user needs to enter email and password then tap on login button.

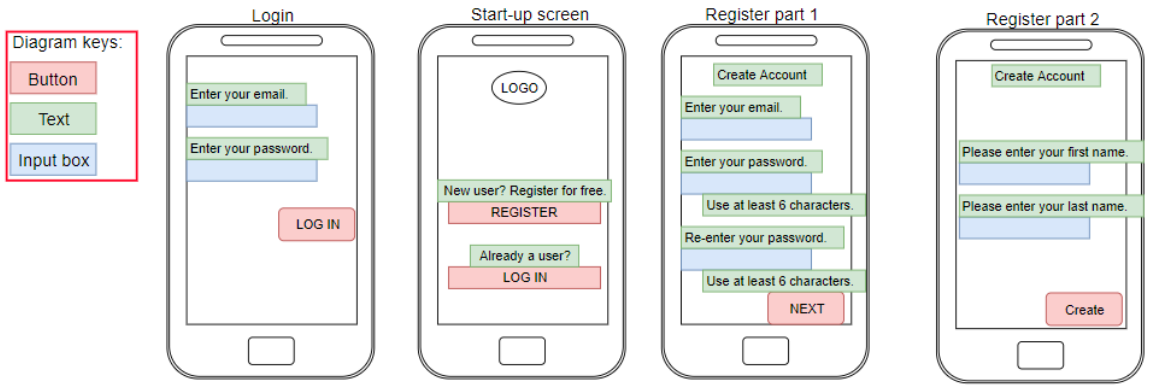

Figure 27- Login, start-up, register screen designs

5.4.2 Family setup, family creation and join family

These screens are very similar to login and register screens as they are used to get data from user, when user gets into family setup screen he can choose to join family or create new family, when creating new family user will need to enter family name, password and re-enter password and tap on create button, when joining family user will need to enter family id and password and then tap join button.

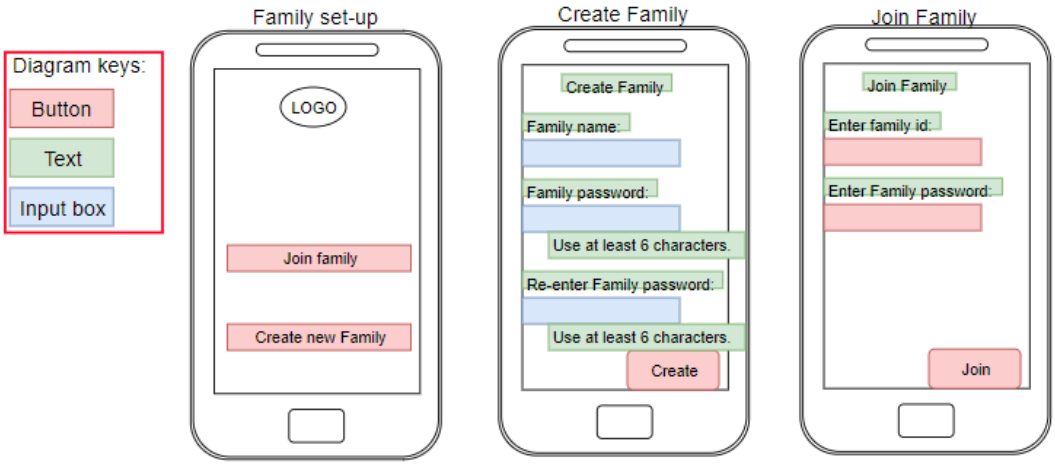

Figure 28- Family setup, create family and join family screen designs

5.4.3 Main screen(map) and side menu

This is the main screen of the application, after user logs in successfully this screen will be displayed, the side menu can be viewed by tapping button in top right corner or swiping from right to left.

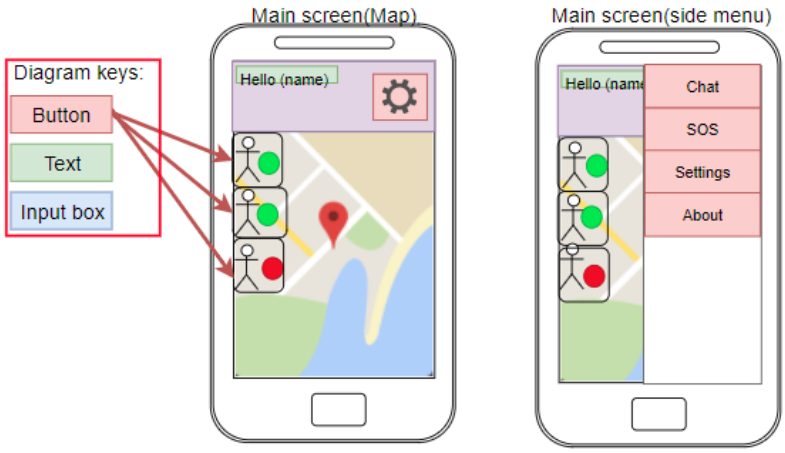

Figure 29- Main screen and side menu designs

5.4.4 Settings, language and marker colours

Settings screen will appear after users taps settings in the side menu, Language screen will be displayed when user taps on language button in settings screen, marker colours will be displayed when user taps on marker colours in settings screen. Marker colours contains of two input boxes, first one is used to select user, when user taps on it, a list of family members will be displayed in a popup list and to the right of it there is another input box which will display a popup list of colours which can be attached to the selected user.

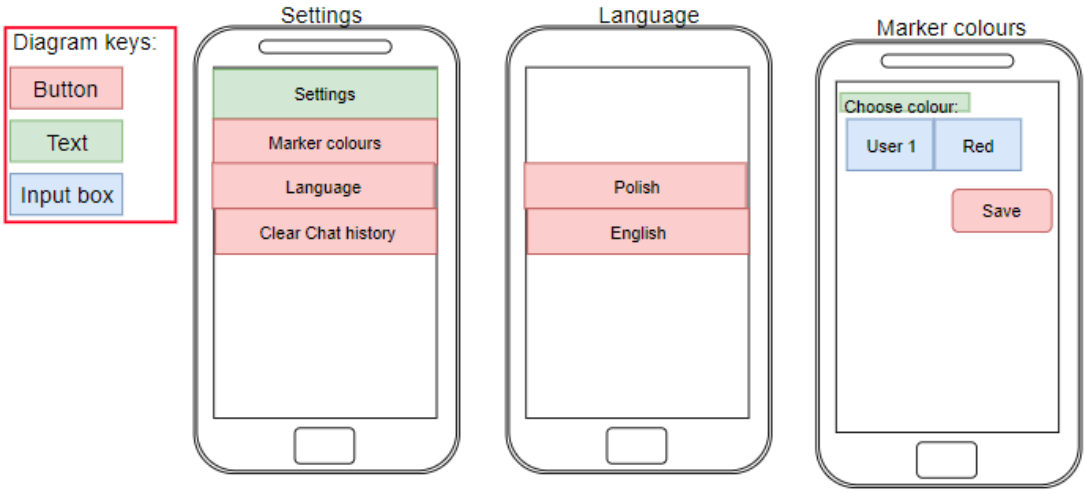

Figure 30- Settings, language and marker colours screen designs

5.4.5 Pre-chat and Chat

Pre-chat screen will be displayed after user taps on chat in the side menu, it allows user to select which family member he wants to send the message to, chat screen will appear after user selects the receiver. Chat screen is mostly covered in area which is used to display chat messages, on the bottom of it there is the input section which allows user to enter message and send it by taping on button.

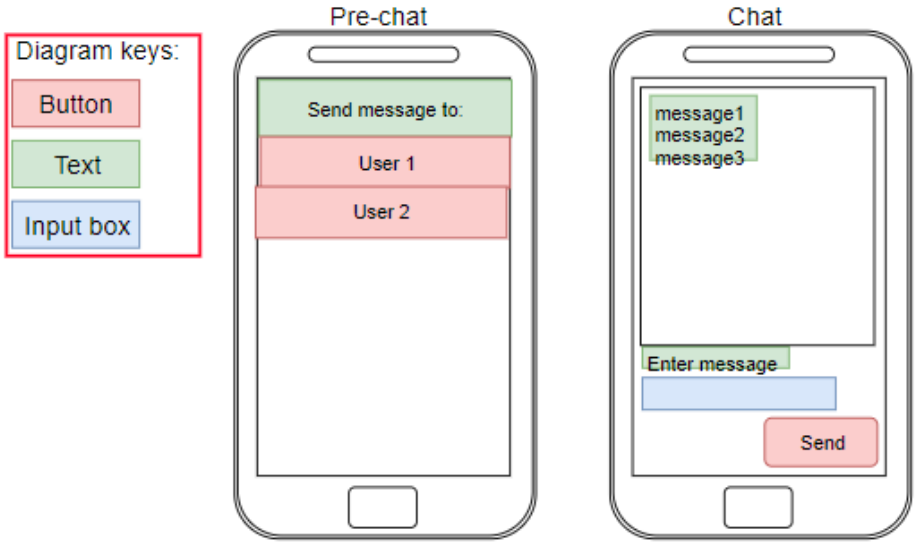

Figure 31- Pre-chat and chat screen designs

5.5 Android application Wireframe

Wireframe is an excellent way of designing a mobile application. It is a two-dimensional illustration which shows how the screens will be made, its shows where each object will be placed on the screen, relationships between screens, available functions and how they interact with each other.

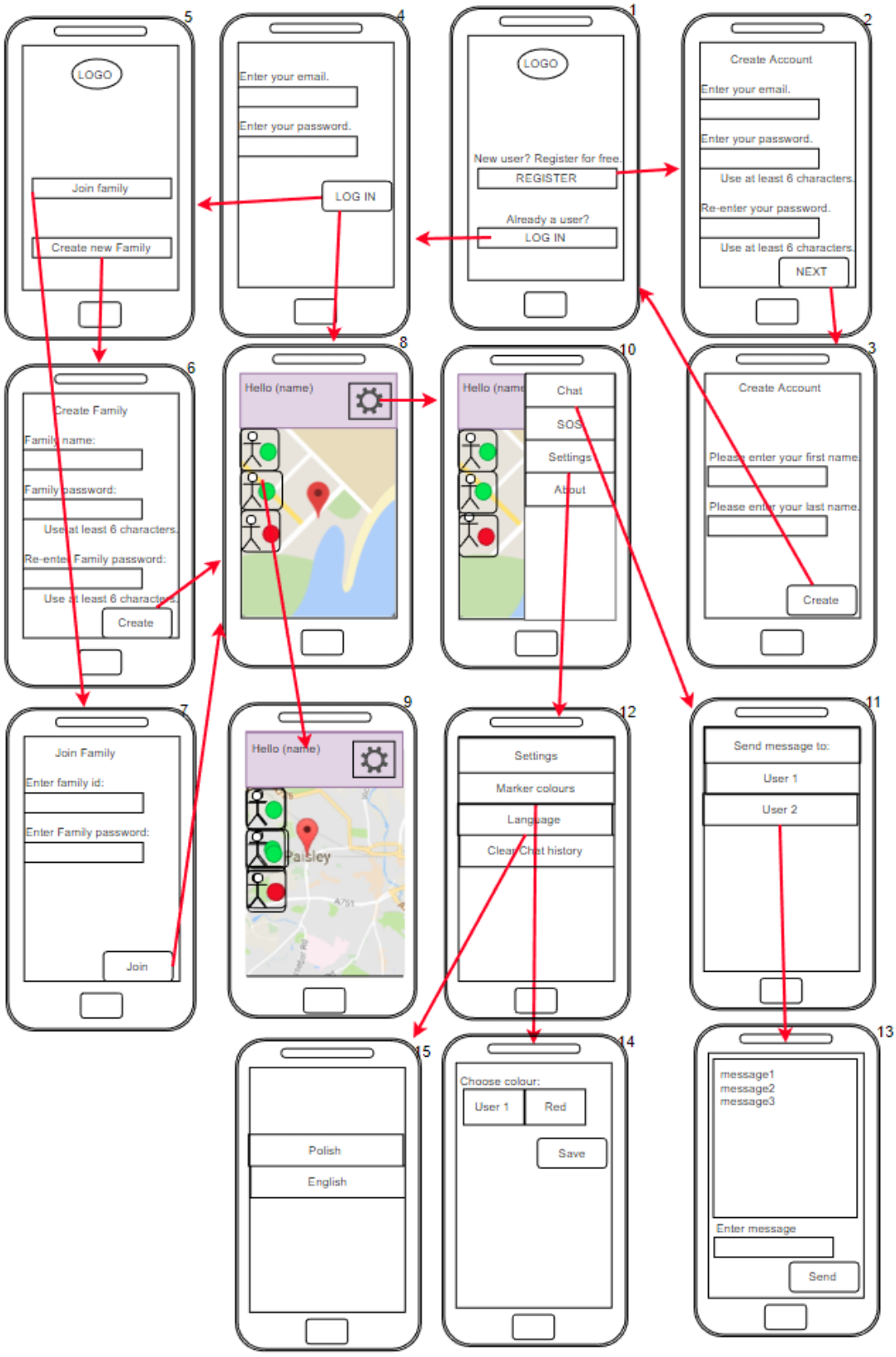

Figure 32- Application wireframe

5.6 Database design

5.6.1 Entity Relational Diagram

ERD (Entity Relational Diagram) is a data modelling technique that graphically shows how the database is structured, relations between the tables and the attributes.

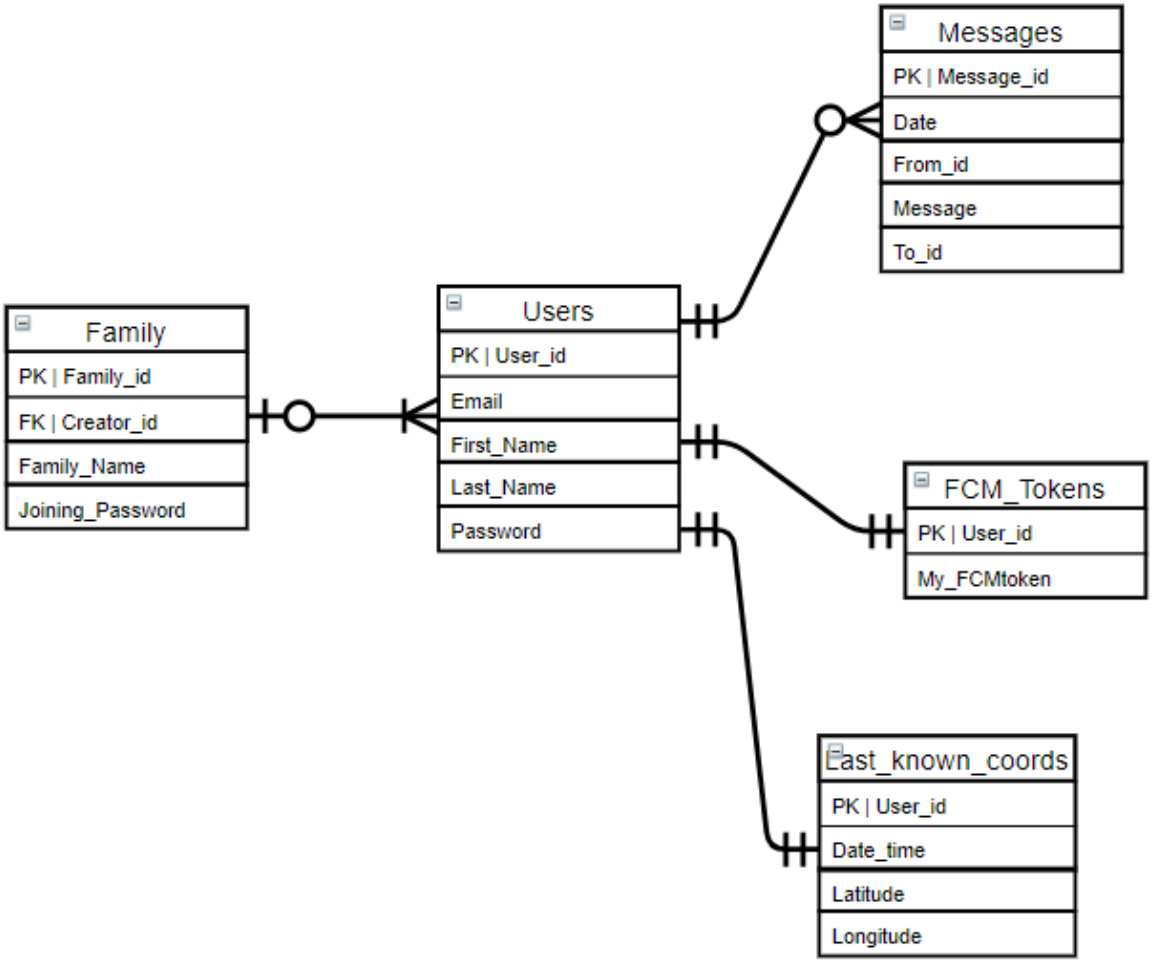

Figure 33- Entity relational diagram

5.6.2 Data dictionary

In database design data dictionary provides information about the tables, details about each table including what fields are used, identification of primary keys and foreign keys, identification of data type and field size.

User Table					
PK/FK	Field Name	Contents	Data Type	Field Size	Required
PK	User_id	User's unique identifier, uses auto increase.	Bigint	20	Y
	Email	User's unique email.	Varchar	50	Y
	First_Name	User's First name.	Varchar	35	Y
	Last_Name	User's Last name.	Varchar	35	Y
	Password	User's password.	Varchar	255	Y
Family Table					
PK	Family_id	Unique family identifier, uses auto increase.	Bigint	20	Y
FK	Creator_id	Foreign key which is used to link family to the user which created it.	Bigint	20	Y
	Family_Name	Family's name.	Varchar	35	Y
	Joining_Password	Family joining password.	Varchar	255	Y
FCM Tokens Table					
PK	User_id	User's unique identifier.	Bigint	20	Y
	My_FCMtoken	FCM	Varchar	255	Y
Messages Table					
PK	Message_id	Unique message identifier, uses auto increase.	Bigint	20	Y
	Date	Stores date and time of the message.	Varchar	255	Y
	From_id	Unique identifier of user which has sent the message.	Bigint	20	Y
	Message	Message text.	Varchar	255	Y
	To_id	Unique identifier of user which needs to receive the message.	Bigint	20	Y
Last_known_location					
PK	User_id	User's unique identifier, uses auto increase.	Bigint	20	Y
	Date_time	Stores date and time which was generated when location has been received.	Varchar	255	Y
	Latitude	Contains latitude of the coordinates.	Double	-	Y
	Longitude	Contains longitude of the coordinates.	Double	-	Y

Table 4- Data dictionary table

6 Implementation

6.1 Introduction to chapter

After the design and analysis of the application which happened in previous chapters, the implementation of the application can begin. This chapter will include implementation of the entire application including front-end(Client) and back-end(Backend).

6.2 Integrated development environments (IDE)

The application was developed via two different IDE's. Server side was developed using IntelliJ IDEA software, this software is not free to use, an educational licence has been obtained from IntelliJ IDEA web site. I have chosen this IDE for development of the server sided software because it makes coding easier and more efficient by smartly generating code and predicting errors.

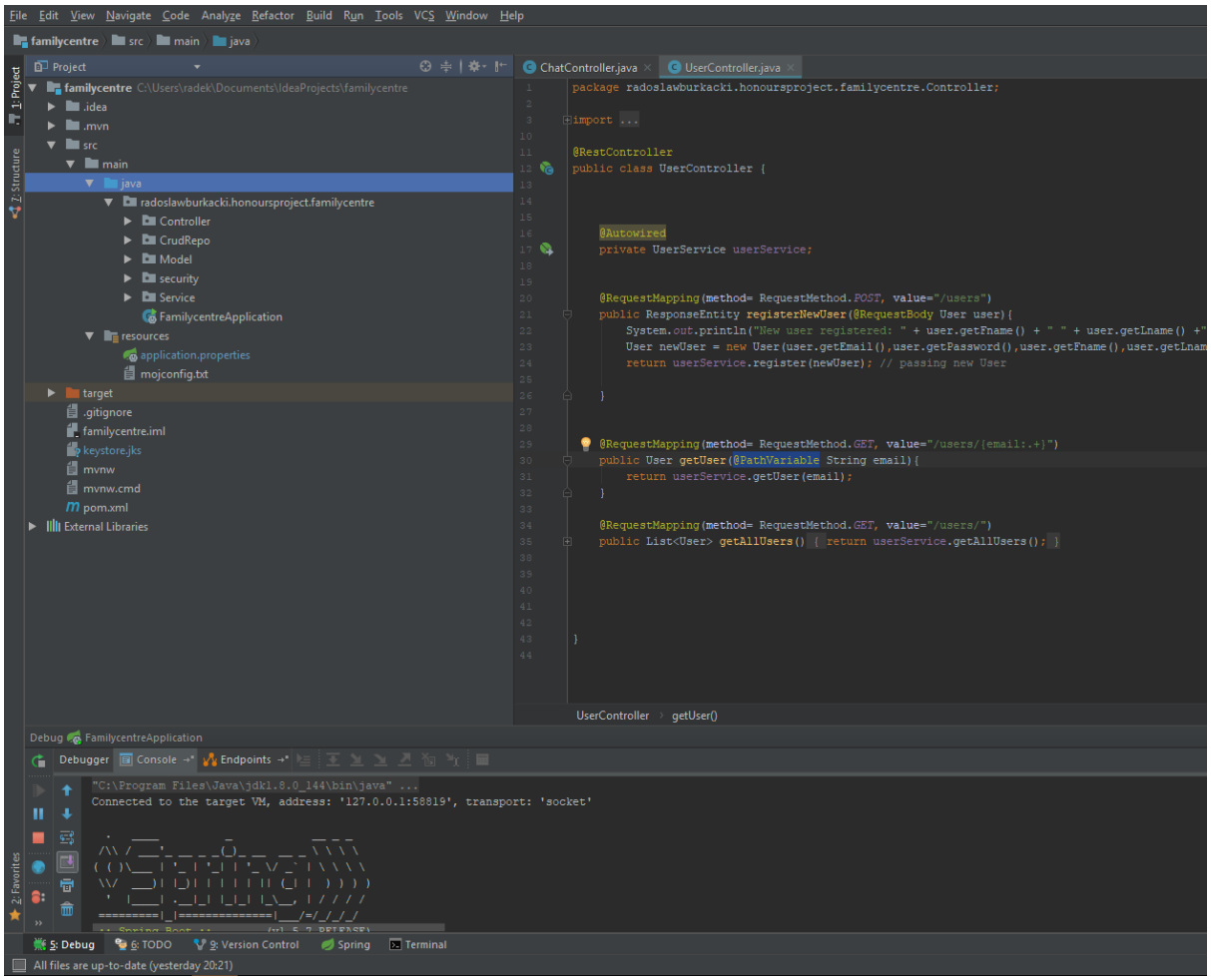

Figure 34- IntelliJ IDEA IDE

The android application was developed via android studio which is very similar to IntelliJ IDEA IDE because android studio is built upon the IntelliJ IDEA IDE. Android studio is the most popular IDE for android development and its released by Google, it comes with android emulator build in which can be used for testing purposes.

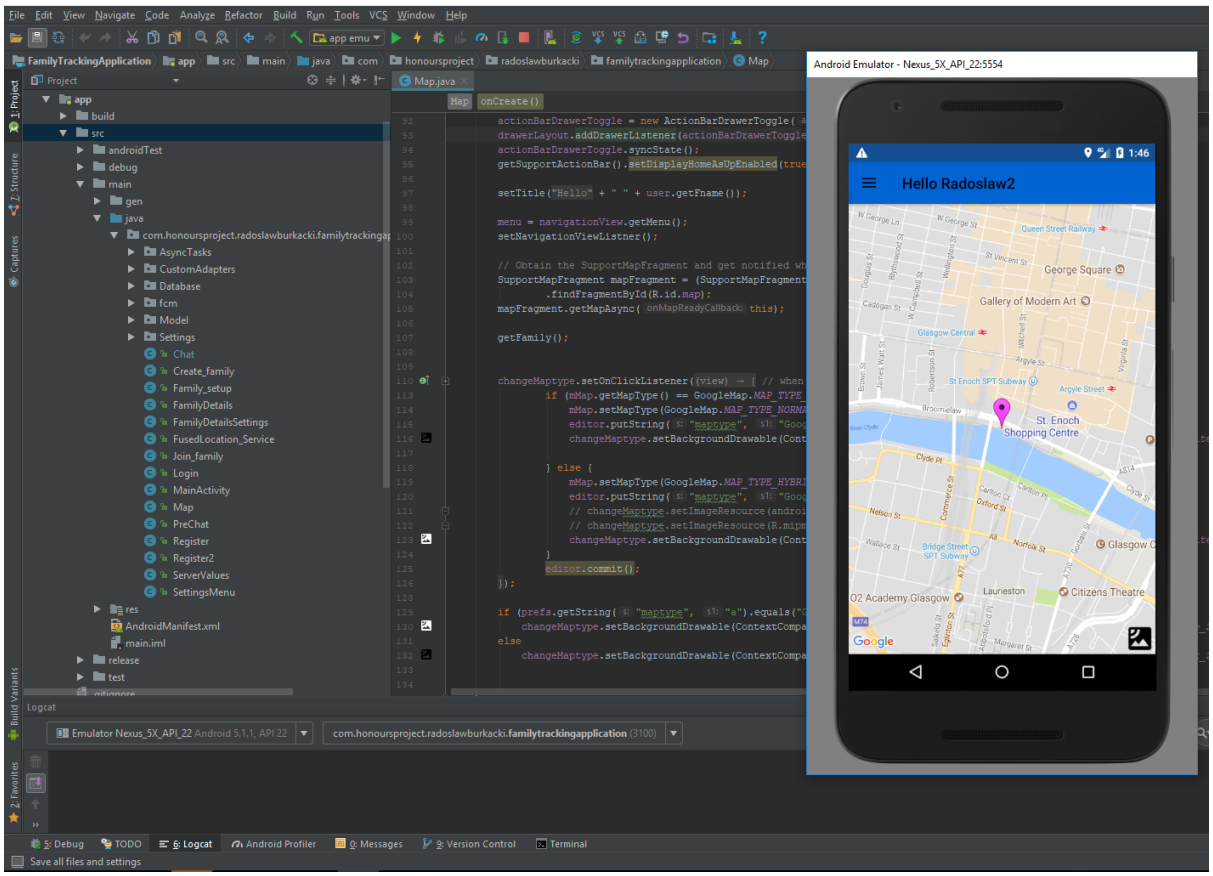

Figure 35- Android studio IDE and Android emulator

6.3 Implementation of connection encryption

To encrypt connection between client and server via SSL (HTTPS) it is required to create trust store which will contain a self-signed certificate, both can be created using key tool application which is available in the JAVA development kit.

To create key store and export certificate the following commands were used:

```
keytool -genkey -keyalg RSA -alias FCA -keystore C://Users/radek/Desktop/keystore.jks -storepass 123456 -validity 360 -keysize 2048

keytool -exportcert -rfc -alias FCA -keystore C://Users/radek/Desktop/keystore.jks -file C://Users/radek/Desktop/selfsigned.crt
```

After executing these commands two files appeared on the desktop: keystore.jks and selfsigned.crt, these files must be used to establish HTTPS connection between client and server.

Server was configured to use the key store file to establish HTTPS connection with clients, the following properties were added to the application.properites file:

```
server.ssl.key-alias=FCA
server.ssl.key-password=123456
server.ssl.key-store=keystore.jks
server.ssl.key-store-provider=SUN
server.ssl.key-store-type=JKS
```

After applying these properties and restarting server, server will boot up and it will accept only requests which use HTTPS.

In a real-life scenario there would be no need to do any additional configurations with the client, because a trusted certificate provided would be used to generate certificate although this solution costs around 100$ - 150$ per year. For purposes of this project a self-signed certificate was created, this means that the android application must be configured to use this certificate as a valid issuer. To achieve that an additional function within ServerValues.class was created, this class is responsible for initialization of custom made okhttp client which is configured to trust self-generated certificate, each time any request is sent to the server this function is used to initialize the okhttp client.

6.4 Implementation of request via Okhttp API

Okhttp client is used for all the communication between client and server and between server and firebase messaging service, all the code which is used for communication is between try and catch block which is used to catch any errors if they occur. Steps needed to send a request:

1. JSON object must be created and attributes and values must be added
2. Request Body object is created based on the JSON data,
3. OkHttpClient is created and initialised
4. Request object is created and initialised, URL, HTTP method and headers are settled
5. Creating Response object, sending request and saving data from response
6. Writing to console
7. Closing response objec

The following code is a template for all the communication using Okhttp client:

```
 final MediaType jsonMediaType = MediaType.parse("application/json"); // creating new MediaType
variable
    try {
      JsonObject jsonObject = new JsonObject(); // creating new json object
      jsonObject.addProperty("id", newUser.getId()); // adding attribute and value to json object
      jsonObject.addProperty("email", newUser.getEmail()); // adding attribute and value to json object
      jsonObject.addProperty("password", newUser.getPassword()); // adding attribute and value to json
object
      jsonObject.addProperty("fname", newUser.getFname()); // adding attribute and value to json object
      jsonObject.addProperty("lname", newUser.getLname()); // adding attribute and value to json object

      RequestBody requestBody = RequestBody.create(jsonMediaType, new Gson().toJson(jsonObject));

      OkHttpClient client = ServerValues.getOkHttpClient();

      Request request = new Request.Builder()
          .url(ServerValues.SERVER_ADDRESS + "/users")
          .post(requestBody)
          .addHeader("content-type", "application/json")
          .build();

      Response response = client.newCall(request).execute(); // setting value of response variable

      statuscode = response.code(); // setting statuscode variable equal to respone.code

      Log.d("https", statuscode + response.body().toString()); // displaying data into console

      response.body().close(); // closing response body

    } catch (Exception e) {
      Log.d("https", e.toString());
    }
```

6.5 JSON structures

In this section I will list all the JSON structures which will be used for the communication between the client and the server and between the server and the firebase cloud messaging service. All request which are sent contain header “content-type” which contains the following value: “application/json”, this header is stating that the body which came with request is in JSON format, some requests use additional header for authentication which is “authentication” and the value is

the JTW token. Example data will be entered in to the JSON objects. The IP address which will be used in this section is the IP address of the computer which is running the server software.

6.5.1 Client requests

This section will contain all the requests which are made by the android application.

6.5.1.1 Register

Register request is sent to server when user is registering in the client. There is one request which needs to be done to complete this action. This request is sending POST request to: https://192.168.0.10:1000/users

The following JSON data is passed with the request:

```
{
        "id":"0",
        "email":"1@exmaple.com",
        "password":"111111",
        "fname":"John",
        "lname":"Smith"
}
```

The response to that request is returning only HTTP code which can be either “Created” (201) which means that the user has been registered or “Conflict” (409) which means that the registration failed because user already exists.

6.5.1.2 Login

Login request is sent to server when user is logging in via the Android application. There are three requests which needs to be done to complete this action.

First the application is sending POST request to: https://192.168.0.10:1000/login

The following JSON data is passed with the request:

```
{
        "id":"0",
        "email":"1@exmaple.com",
        "password":"111111",
        "fname":"John",
        "lname":"Smith"
}
```

This request is returning HTTP code which can be “OK” (200) which means that the credentials were correct, and user will be logged in, if the response is “OK” then the response also contains additional header named “Authentication” which contain JWT token which is associated with that user and must be provided in future communication between client and server. If the credentials are wrong, then the response will contain “Unauthorized” (401) HTTP code.

All requests after user has logged in successfully must contain additional header which is “Authentication” and it must contain JTW token.

Next request is used to get instance of user which has just logged in. It is sending GET request to: https://192.168.0.10:1000/users/1@exmaple.com. This request is passing parameter in the URL instead of sending JSON data. Response to this request always returns JSON data which contains HTTP code “OK” (200) and instance of user object:

```
{
  "id":1,
```

```
  "email": "1@exmaple.com",
  "password": "",
  "fname": "John",
  "lname": "Smith"
}
```


Password attribute has been set to nothing on the server-side for security reasons.

The last request which is needed to complete login action is used to check if the user which is logging in is member of any family. It is sending HEAD request to: https://192.168.0.10:1000/families/by-user-id/1. This request is passing user id in the URL. Response this this request returns HTTP code and a JSON data which contains instance of family objects. If user is a member of any family then the request contains HTTP code "Found" (302) and family objects:


```
{
  "id": 1,
  "creatorId": 1,
  "familyName": "Smith",
  "joiningPassword": "",
  "familyMembers": [
    {
      "id": 1,
      "email": "1@example.com",
      "password": "",
      "fname": "John",
      "lname": "Smith"
    }
  ]
}
```


Passwords attributes has been set to nothing on the server-side for security reasons. If the user is not a member of any family then response contains HTTP code "Not Found" (404) and an empty object family object:


```
{
  "id": null,
  "creatorId": null,
  "familyName": null,
  "joiningPassword": null,
  "familyMembers": []
}
```


6.5.1.3 Create Family

Create Family request is sent to server when user is creating new family. There is one request which needs to be sent to complete this action. This request is sending POST request to: https://192.168.0.10:1000/families

The following JSON data is passed with the request:


```
{
  "id": 1,
  "creatorId": 1,
  "familyName": "Smith",
  "joiningPassword": "111111"
}
```

Response to this contains HTTP code which can be either “Created” (201) which means that new family has been created or “Conflict” (401) which means that family couldn’t be created.

6.5.1.4 Join Family

Join family request is sent to server when user is joining family. This request is sending POST request to: https://192.168.0.10:1000/families/

The following JSON data is passed with the request:

```
{
  "familyId":1,
  "familyPassword":"111111",
  "userId":"1"
}
```

Response to this request contains HTTP code which can be “Created” (201) only if user has joined the family successfully or “Conflict” (409) if user couldn’t be added to family.

6.5.1.5 Get family

Get family action is used to get family object which contains all family members. This request is sending GET request to: https://192.168.0.10:1000/families/by-user-id/1. This request is passing user id in the URL.

Response to this request contains HTTP code and JSON data which contains family object. If user is a part of the family then HTTP code returns is “Found” (302) then the request also contains JSON data with family object:

```
{
  "id":1,
  "creatorId":1,
  "familyName":"Smith",
  "joiningPassword":"",
  "familyMembers":[
    {
      "id":1,
      "email":"1@exmaple.com",
      "password":"",
      "fname":"John",
      "lname":"Smith"
    }
  ]
}
```

If user is not a member of family then HTTP code “Not Found” (404) is returned.

6.5.1.6 Send location

Send location action is used to send coordinates of user. This request is sending POST request to: https://192.168.0.10:1000/families/location/1. This request is passing user id in the URL and JSON data:

```
{
  "latitude":55.843420,
  "longitude":-4.429980
}
```

Response to this request contains of HTTP code which can be “OK” (200) if the location has been updated or “Conflict” (409) if location couldn’t be updated.

6.5.1.7 Track family member

Tracking family member request is sending GET request to: https://192.168.0.10:1000/families/location/1. This request is passing user id in URL.

Response to this request contains of HTTP code and JSON data. If used couldn’t be located then HTTP code contains “Not Found” (404), if user was located then response contains HTTP code “Found” (302) and the response contains JSON data:

```
{
  "id":1,
  "latitude":55.8750185,
  "longitude":-4.263724,
  "list":[
    0,
    0,
    0,
    0,
    0,
    2
  ]
}
```

List array which is inside of JSON data is used to determinate when user was located, there are 6 numbers in total, first number is responsible for year, second for month, third for days, fourth for hours, fifth for minutes and sixth for seconds.

6.5.1.8 Send SOS

SOS request is sending HEAD request to: https://192.168.0.10:1000/sos/1. This request is passing user id in URL. The response to this request contains of HTTP code which is used to determinate if the request was sent to other family members successfully.

6.5.1.9 Send private message

Send private message request is sending POST request to: https://192.168.0.10:1000/chat/. The request is passing JSON data which contains the message:

```
{
  "messageId":"0",
  "fromId":"1",
  "toId":"2",
  "message":"Hello",
  "date":"16/03/2018 18:00:00"
}
```

Response to that request contains of HTTP code which is set to “Created” (201) when message was sent successfully or “Conflict” (409) when message couldn’t be sent.

6.5.1.10 Send FCM token

Send FCM token request is used to pass FCM token of the client to server, to achieve that an POST request is sent to: https://192.168.0.10:1000/fcmtoken. The request contains JSON data:

```
{
  "messageId":"0",
  "fromId":"1",
```

```
  "toId": "2",
  "message": "Hello",
  "date": "16/03/2018 18:00:00"
}
```

6.5.1.11 Delete user from family task

This task is used to delete user from family, to achieve that an DELETE request is sent to: https://192.168.0.10:1000/families/. The request contains JSON data:

```
{
  "FamilyId": "1",
  "UserId":"2"
}
```

6.5.2 Server requests

This section will contain all requests which are made on the server, all these requests are sent to FCM service and each request must contain of two headers, first one is used to state that JSON data has been passed with request and its using “content-type” and the value is “application/json”, the other header is “Authorization” and the value is the token which has been obtained from FCM.

6.5.2.1 Send SOS notification

Send SOS notification action is used to notify all family members that one of them has sent SOS. The request is sent to: https://fcm.googleapis.com/fcm/send. The request contains JSON data:

```
{
  "messageId": "0",
  "fromId":"1",
  "toId": "2",
  "message": "Hello",
  "date": "16/03/2018 18:00:00"
}
```

FCM service is always sending with a JSON response to all the requests, it contains attributes with details about the request but also “success” and “failure” which state if the request was forwarded successfully or if it failed. JSON response:

```
{
  "multicast_id": 7634529082161767363,
  "success": 1,
  "failure": 0,
  "canonical_ids": 0,
  "results": [
    {
      "message_id": "0:1521241095153969%12cd9972f9fd7ecd"
    }
  ]
}
```

6.5.2.2 Send private message notification

Send private message notification action is used to send the message to the receiver. The request is sent to: https://fcm.googleapis.com/fcm/send. The request contains JSON data:

```
{
  "data":{
        "message": "example message"
        "fromid": "1"
        "toid": "2"
```

```
            "date": "16/03/2018 20:00"
    },
    "android":{
            "priority": "high"
    },
    "to": "exampleFCMToken"
}
```

The response from FCM contains exactly same JSON structured data as in Send SOS notification request.

6.5.2.3 Send new family member notification

Send new family member notification action is used to send notification to all family members, letting them know that new user has joined their family. The request is sent to: https://fcm.googleapis.com/fcm/send. The request contains JSON data:

```
{
    "data":{
            "newfamilymember": "John2 Smith2"
    },
    "android":{
            "priority": "high"
    },
    "to": "exampleFCMToken"
}
```

The response from FCM contains exactly same JSON structured data as in Send SOS notification request.

6.6 Client implementation

This section will focus on client implementation, it will discuss how android application screens are made, it will contain all the classes and functions which are used inside of the android application and each of them will be discussed and explained.

6.6.1 Android application screen build-up

Each screen which is displayed in android application is built of two files:

- Layout file – Layout file is used to create everything that is on the screen that user can see: text boxes, labels and buttons. It is using XML format to create all these controls.
- Java file – Java file is the mechanism which works in the background, it is used to get input from the text boxes, detect user button taps, connect to servers, set label values and many more…

6.6.2 List of classes and functions

This section will contain all the classes, their purposes and all the functions which are used inside of them. Classes will be split into 6 sub-sections including model classes which are used to store data, screen classes which used as background logic of the screen, services which are being run in background, asynchronous tasks classes which are used for communication between client and server and other classes.

6.6.2.1 Model classes

6.6.2.1.1 User class

User class is used to store all the data about the user including first name, last name, id, email and password. When new user is created a new instance of user class is created, data is taken from user and inserted into that instance of user.

List of functions:

Function	Purpose
public User()	Empty user constructor, its used to create empty user object.
public User(Long id, String email, String password, String fname, String lname)	It is a User constructor which expects 5 variables to be passed into it, these variables will be used to create new instance of user object.
public long getId()	Used to get user id.
public String getEmail()	Used to get user email.
public void setEmail(String email)	It is expecting 1 variable to be passed in to it, used to set email of user object.
public String getPassword()	Used to get user password.
public void setPassword(String password)	It is expecting 1 variable to be passed into it, used to set password of user instance.
public String getFname()	It is used to get user first name.
public void setFname(String fname)	It is expecting 1 variable to be passed into it, used to set first name of user instance.
public String getLname()	It is used to get user last name.
public void setLname(String lname)	It is expecting 1 variable to be passed into it, used to set last name of user instance.
public String toString()	Used to display all user attributed.

Table 5- List of functions (User. Class)

6.6.2.1.2 Family class

Family class is used to store all the data about the family including id, creator id, family name, joining password and family members. When new family is created a new instance of family class is created, data is taken from user and inserted into family instance.

List of functions:

Function	Purpose
public Family()	Empty family constructor, its used to create empty family object.
public Family(Long creatorId, String familyName, String joiningPassword, List<User> familyMembers)	It is a Family constructor which expects 4 variables to be passed into it, these variables will be used to create new instance of family object.
public long getId()	Used to get family id.
public Long getCreatorId()	It is used to get creator id.
public void setCreatorId(Long creatorId)	It is used to set creator id, it is expecting 1 variable to be passed into it.
public String getFamilyName()	It is used to get family name.
public void setFamilyName(String familyName)	It is used to set family name, it is expecting 1 variable to be passed into it.
public String getJoiningPassword()	It is used to get joining password.
public void setJoiningPassword(String joiningPassword)	It is used to set joining password, it is expecting 1 variable to be passed into it.
public List<User> getFamilyMembers()	It is used to get all family members.
public void setFamilyMembers(List<User> familyMembers)	It is used to set family members, it is expecting 1 variable to be passed into it.
public void addFamilyMember(User u)	It is used to add single family member to family, it is expecting user object to be passed into it.
public String toString()	Used to display all attributes and values of family.

Table 6- List of functions (Family. Class)

6.6.2.1.3 Message class

Message class is used to store all the data about the message including message id, from which user the message was sent id, to which user the message must be sent id, message body and date. When new message is created a new instance of message class is created, data is taken from user and inserted into message instance.

List of functions:

Function	Purpose
public Message ()	Empty message constructor, its used to create empty message object.
public Message (long messageId, long fromId, long toId, String message, String date)	It is a Message constructor which expects 5 variables to be passed into it, these variables will be used to create new instance of message object.
public long getMessageId()	It is used to get message id.
public long getFromId()	It is used to get from id.
public long getToId()	It is used to get to id.
public String getMessage()	It is used to get message text.
public String getDate()	It is used to get date.
public void setMessageId(long messageId	It is used to set message id, it is expecting 1 variable to be passed into it.
public void setFromId(long fromId)	It is used to set from id, it is expecting 1 variable to be passed into it.
public void setToId(long toId)	It is used to set to id, it is expecting 1 variable to be passed into it.
public void setMessage(String message)	It is used to set message id, it is expecting 1 variable to be passed into it.
public void setDate(String date)	It is used to set date, it is expecting 1 variable to be passed into it.
public String toString()	Used to display all attributes and values of message.

Table 7- List of functions (Message. Class)

6.6.2.3 Screen classes

6.6.2.3.1 Main Activity class

Main activity class is used as background mechanism for the start-up screen. It contains function "onCreate" which is used only in classes which are used to display content on the screen.

List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible calling function which sets language, starts autologin, for creating variables out of visual controls and for creating on click listeners to buttons which are triggered when buttons are clicked.
protected void tryAutologin()	This function is responsible for checking if user has previously logged in and if its possible to autologin user when application is started.
protected void setLanguage()	It is responsible for checking if user has settled preferred language if user did then that language is loaded and used in all activities.
protected void startBackgroundAnimation()	It is responsible for starting up background animation.
protected void onResume()	It is called when application is resumed from background.
protected void onPause()	It is called when application is minimalised into background.

Table 8- List of functions (Main Activity. Class)

6.6.2.3.2 Register1 class

Register1 class is used as a background mechanism for registration part 1 screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible creating variables out of visual controls, all the logic behind the user interface and for creating on click listeners to buttons which are triggered when buttons are clicked.
private void showMessage(String msg)	This function is expecting 1 variable to be passed into it, it is used to display toast message on the screen.

Table 9- List of functions (Register1. Class)

6.6.2.3.3 Register2 class

Register2 class is used as a background mechanism for registration part 2 screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are

	clicked, for checking if device is connected to internet and for sending request to server.
private void pleaseEnterCredentials()	It is used to display toast message on the screen.
private boolean isNetworkAvailable()	It is used to check if device is connected to internet.
public void register()	This function is used for starting RegisterTask which is responsible for registration of user.
public void processFinish(Integer statuscode)	This function is used to get data from RegisterTask class, it is called when RegisterTask has finished its job. It provides the status code which was taken from the response to the registration request and based on that register2 class can determinate if the user was registered successfully or not.

Table 10- List of functions (Register2. Class)

6.6.2.3.4 Login class

Login class is used as a background mechanism for login screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
private void startBackgroundAnimation()	This function is used to start background animation.
protected void onResume()	It is called when application is resumed from background.
protected void onPause()	It is called when application is minimalised into background.
public void SendFcmToken(Long userid, String token, String FCMtoken)	This function is used to send fcm token to server, it is creating instance of class SendFCMTokenTask and its executing it. It is expecting 3 variables to be passed into it.
public void login()	This function is used to login user, it is creating new instance of loginTask which is used to login user and its executing it.
public void SendFcmToken(Long userid, String token, String FCMtoken)	This function is used to send fcm token to server, it is creating new instance of SendFCMTokenTask and executing it. This function is expecting 3 variables to be passed into it.
public void processFinish(String token, int statuscode, User user, boolean isUserFamilyMember)	This function is used to get data from loginTask class, it is called when loginTask has finished its job. It provides 4 variables including token, status code, user object and Boolean variable, based on the values of these an appropriate message or new screen will be displayed.

Table 11- List of functions (Login. Class)

6.6.2.3.5 Family setup class

Family setup class is used as background mechanism for family set up screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.

Table 12- List of functions (Family setup. Class)

6.6.2.3.6 Join family class

Join family class is used as background mechanism for joining family screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
public void JoinFamily()	This function is creating new instance of JoinFamilyTask and its executing it.
public void processFinish(int statuscode)	This function is used to get data from JoinFamilyTask class, it is called when JoinFamilyTask has finished its job. It provides the status code which was taken from the response to the join family request and based on that join family class can determinate if the user has joined family successfully or not.

Table 13- List of functions (Join family. Class)

6.6.2.3.7 Create family class

Create family class is used as background mechanism for family creation screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
private void GetFamily()	It is used to create new instance of GetFamilyTask class and to execute it.
private void createNewFamily()	It is used to create new instance of createNewFamily class and to execute it.
public void processFinish(Family f, int statuscode)	This function is used to get data from GetFamilyTask class, it is called when GetFamilyTask has finished its job. It provides the status code and family object which was taken from the response to the get family request. Based on the values of these variables

	appropriate screen or message is displayed.
public void processFinish(int stautscode)	This function is used to get data from CreateFamilyTask class, it is called when CreateFamilyTask has finished its job. It provides the status code which was taken from the response to the create family request. Based on the value of status code appropriate screen or message is displayed.

Table 14- List of functions (Create family. Class)

6.6.2.3.8 Map class

Map class is used as background mechanism for Map screen which is the main screen of the application, this class is the biggest because all the functionality is in this class. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
private void startLocationService()	This function is used to start up the service which is responsible for sending location updates to the server
private boolean runtime_permissions()	It is used to check if used has given the application appropriate permissions if then then user is asked to do so.
protected void onResume()	It is called when application is resumed from background.
public void onRequestPermissionsResult()	It is used to check if used has given appropriate permission.
protected void onDestroy()	This function is called when application is destroyed, it is stopping receivers when application is destroyed.
private void moveCamera(LatLng latLng, float zoom)	This function is used to move camera over the map. It is expecting two variables to be passed into it- coordinates and zoom value.
public void onMapReady(GoogleMap googleMap)	This function is called when the map is ready to use, it is expecting google map object.
public boolean onNavigationItemSelected(@NonNull MenuItem item)	This function is called when user taps one of the menu options on the drawer menu (side menu). It is responsible for carrying appropriate action like displaying settings menu based on user input.
private void setNavigationViewListner()	Used to set listener in the drawer menu.
private void saveFamilyToSharedPreferences(Family f)	It is used to save data about family into shared preferences storage.
private Family getFamilyFromSharedPreferences()	It is used to retrieve family data from shared preferences storage.
public void setUpDrawerMenu()	It is responsible for adding options to the drawer menu.
public String getLastSeen(List<Integer> lastseenList)	This function is expecting list of integers to be passed into it. It is used to get a string which will tell the user when a member which has been tracked

	has sent location to server.
private boolean isMyServiceRunning(Class<?> FusedLocation_Service)	This function is expecting an instance of a class to be passed. It is used to check if the location service is already running.
private Float getMarkerColour(int userid)	It is expecting a variable to be passed into it. This function is reading shared preferences storage to find out what colour has been attached to a given user and its returning the colour as float value.
public void getFamily()	It is used to create new instance of GetFamilyTask class and to execute it.
public void getFamilyMemberLocation(long familyMemberId)	It is used to create new instance of getFamilyMemberLocation class and to execute it. Its expecting a variable to be passed into it.
public void sendSOS()	It is used to create new instance of sendSOS class and to execute it.
public void processFinish(Family f, int statuscode)	This function is used to get data from getFamilyTask class, it is called when getFamilyTask has finished its job. It provides the status code and family object.
public void processFinish(int statuscode, LatLng coordinates, long familyMemberId, List<Integer> list)	This function is used to get data from getFamilyMemberLocation class, it is called when getFamilyMemberLocation has finished its job. It provides 4 variables which are then used to locate the user and the time when he was located.
public void processFinish(Integer statuscode)	This function is used to get data from SendSOSTask class, its called when SendSOSTask has finished its job. It provides a status code based on which message is displayed if SOS has been sent successfully or not.

Table 15- List of functions (Map. Class)

6.6.2.3.9 Pre-chat class

Pre-chat class is used as background mechanism for setting up chat conversation. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.

Table 16- List of functions (Pre-chat. Class)

6.6.2.3.10 Chat class

Chat class is used as background mechanism for chat screen. This class is responsible for displaying all the chat messages and getting messages to send from user. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
protected void onResume()	It is called when application is resumed from background.
protected void onPause()	It is called when application is minimalised into background.
private void sendMessage(Message m)	This function is expecting a message object to be passed into it. It is creating new instance of SendChatMessageTask and its executing it.
public void processFinish(int statuscode)	This function is used to get data from SendChatMessageTask class, it's called when SendChatMessageTask has finished its job. It provides a status code based on which message is displayed if message has been sent successfully or not.
private void setUpChat()	It is responsible for setting up chat message screen, when a chat screen is displayed it is getting messages from database and its displaying them in the right order.

Table 17- List of functions (Chat. Class)

6.6.2.3.11 Settings menu

Settings menu class is used as background mechanism for settings menu screen. This class is responsible for displaying all the options which are in options menu and for getting the input from user. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
private void saveLanguageToSharedpref(String arg)	It is called when user is saving new language preference. It is saving language preference to shared preference storage.
private void setUpSettings()	It is used to set up settings menu, it is setting up all the menu labels.

Table 16- List of functions (Settings menu. Class)

6.6.2.3.12 Marker Colour

Marker colour class is used as background mechanism for marker colour screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.
private void displaySaveMessage(User u)	This function is responsible for displaying message which tells user that colour has been save successfully for user X. It is expecting user object to be passed into it.
private Family getFamilyFromSharedPreferences()	Gets family object from shared preference storage.

Table 17- List of functions (Marker colours. Class)

6.6.2.3.13 Family Details

Family details class is used as background mechanism for family details screen. It is responsible for all the logic which is behind the user interface. List of functions:

Function	Purpose
protected void onCreate(Bundle savedInstanceState)	This function is called when the activity is displayed on the screen. This function is responsible for creating variables out of visual controls, all the logic behind the user interface, for creating on click listeners to buttons which are triggered when buttons are clicked.

Table 18- List of functions (Family details. Class)

6.6.2.4 Service classes

6.6.2.4.1 Fused location service class

This class is used to send location updates to the server. It is running in the background even If the phone is put to sleep or application is killed. This service is started when user is logged in to the application and can be stopped only by logging out of the application. List of functions:

Function	Purpose
public int onStartCommand(Intent intent, int flags, int startId)	When service is started then this function is executed. It is used to configure the Google Api Client which is used to get the location and to configure the service settings.
public void onCreate()	This function is called when class is initialised. It is used to get data from shared preference and to set up the location service.
public void onLocationChanged(Location location)	This function is called when user changes location. This function is calling another function which is sending location coordinates to the server.
public void onConnected(Bundle arg0)	This function is run when user is connected. It is checking if user has given permissions to access user's location.
public void onConnectionSuspended(int i)	This function is called when connection is suspended, it is reconnecting
public void SendCoordinates(LatLng latLng)	This function is creating instance of SendCoordinatesTask which is used to send coordinates to the server and its executing it.
private Family getFamilyFromSharedPreferences()	Used to get data from shared preferences.

Table 19- List of functions (Fused location service. Class)

6.6.2.4.2 My firebase messaging service

This class is running in the background of the application. It is listening to FCM service messages, if any message is sent to the device from FCM then this service will receive it and trigger appropriate function. List of functions:

Function	Purpose
public void onCreate()	This function is responsible for creating new instance of database handler and broadcaster which is used to send data to other classes.
public void onMessageReceived(RemoteMessage remoteMessage)	This function is triggered when a message has been received from FCM. It contains a large block of if statements which are used to determine what actions must be carried out based on the data received and then it is calling an appropriate function.
private void playSoundandVib()	Used to play default notification sound and to vibrate.
private void showSOSnotification(RemoteMessage remoteMessage)	Used to show SOS notification.
private void showNewFamilyMemberNotification(String newFamilyMemberName)	Used to show new family member notification.
private void showPMNotification(String message, String senderid, String toid, String date)	Used to show new message notification.
private Family getFamilyFromSharedPreferences()	Used to get family from shared preferences.

6.6.2.4.3 My firebase instance id service

This class is used to get firebase cloud messaging token.

List of functions:

Function	Purpose
public void onTokenRefresh()	Used to get FCM token.

Table 20- List of functions (My firebase instance id service. Class)

6.6.2.5 Asynchronous task classes

There are 10 asynchronous task classes in this application, each of them is used to send different requests to the server. The structure and internal behaviour of these classes is the same in all these classes only the values of JSON requests and the URL to which the request is sent are changed, so in this sub-section only Register Task will be described.

6.6.2.5.1 Register Task

Register Task class is used to send request to server which is used to register new user.

List of functions:

Function	Purpose
public RegisterTask(AsyncResponse delegate,Context context, User newUser)	It is the constructor of the class, it is expecting 3 variables to be passed into it. It is setting value of class variables to value of variables which were passed.
protected Integer doInBackground(User... param)	This function is started when the task is executed. It is building a JSON request file, its adding headers to the request, then creating Okhttp client, then it is sending the request and it is saving the response.
protected void onProgressUpdate(String... values)	This function is called on each update.
protected void onPostExecute(Integer statuscode)	This function is called when the task is finished. It is used to return status code to the class which has started the task.

Table 21- List of functions (Register task. Class)

6.6.2.6 Other classes

There are two classes in this sub-section. Server values which has been created to store data which will be accessible to all other classes, and the other one is the database handler which allows all classes to access database data.

6.6.2.6.1 Server Values

Server values class is responsible for storing two variables which are used across whole application, including server address and shared preferences name. Each class which is accessing shared preferences storage or is sending request to the server is using this class to get the values.

6.6.2.6.2 Db Handler

Db handler class is used to interact with the SQLite database which is stored on the android device, it is used to create database, read from database and write data to it.

List of functions:

Function	Purpose
public dbHandler(Context context)	This function is a constructor which expects a context object to be passed into it. It is used to set up the database.
public void onCreate(SQLiteDatabase db)	This function is called when the class is initialised. It is expecting SQLiteDatabase object to be passed into it, it is creating new table called table messages.
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	This function is responsible for making sure that database version is correct.
public void addMessage(Message m)	This function is used to add messages to database, it is expecting message object to be passed into it.
public Message getMessage(int id)	Used to get message from database.
public List<Message> getAllMessagess()	Used to get all messages from database.
public List<Message> get20lastMessages(long fromid, long toid)	Used to get 20 last messages.
public void removeAllMesagess()	Used to delete all messages from database.

Table 22- List of functions (Db handler. Class)

6.6.3 Android application screen end-result

This section will contain screen shots of family centre application which is running on android emulator. All these screen shots are from the final version of the application which was produced in the implementation stage, although they may change in the evaluation stage.

6.6.3.1 Start-up, login, register part 1 and register part 2

These screen shots are the results of the implementation. Going from left to right, first screen shot is the login screen, then the main menu screen, then register part 1 screen and the last screen is register part 2. In the first two screens (login and start-up screens) there is a background animation running which is doing a colour animation.

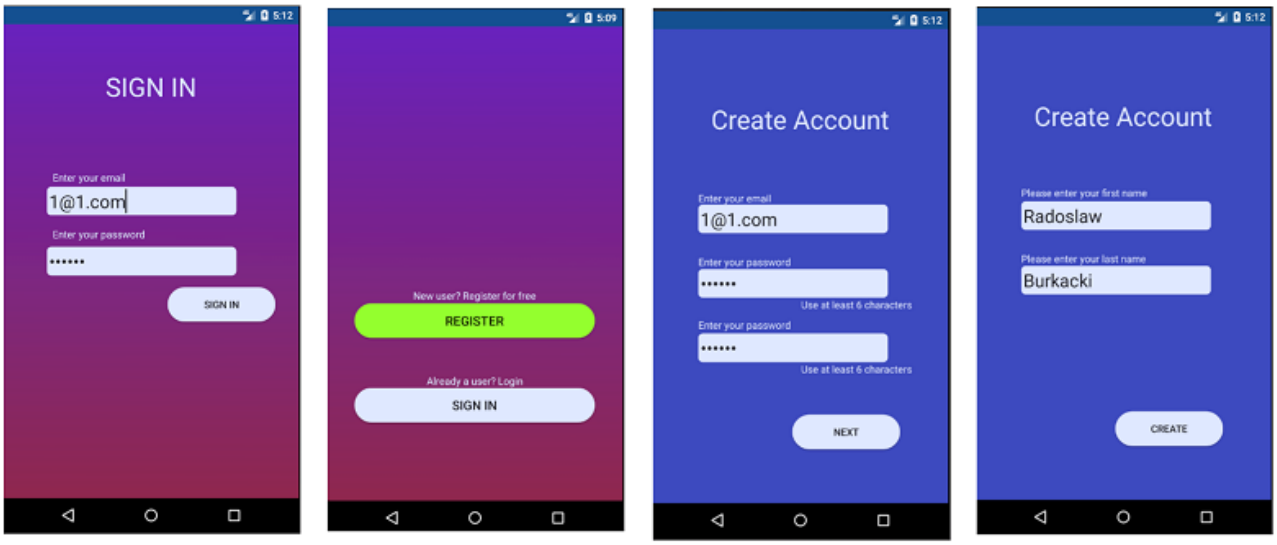

Figure 36- Start-up, login, register screens implementation

6.6.3.2 Family set-up, join family and create new family

Going from left to right, first screen is the family set-up screen, then is the join family screen and the last one is the family creation screen.

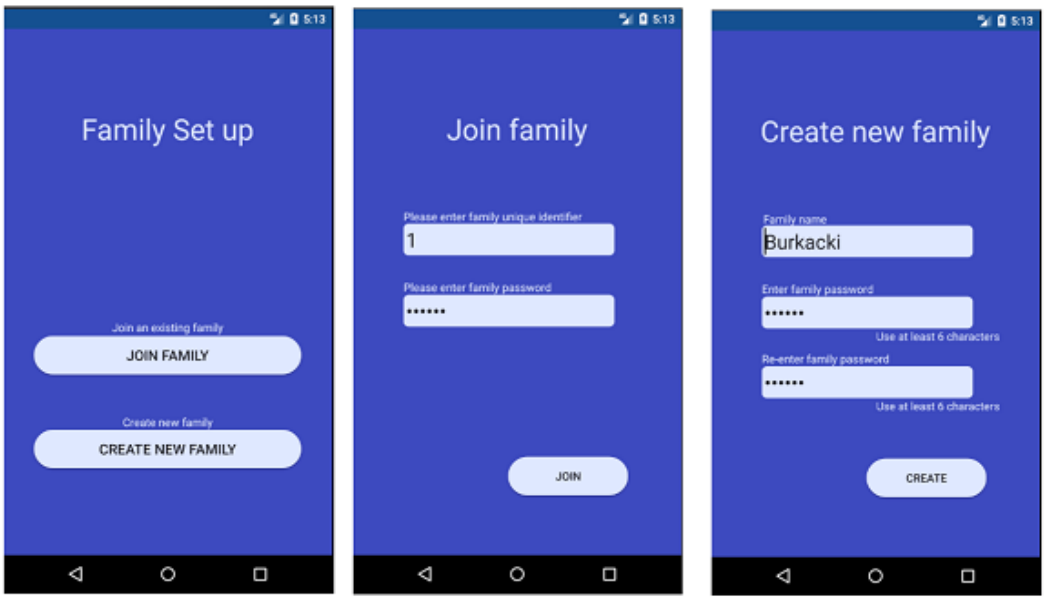

Figure 37- Family setup, family creation and join family screens implementation

6.6.3.3 Main screen(map), family details, drawer menu and SOS

Going from left to right on the left side there is the family details screen which is displaying all the information about the family this screen is displayed after user taps on my family button in the drawer menu, the next one is the screen which is displayed after user created family, map has focused on the location of the user and a message has been displayed, the next screen is the drawer menu which contain all the options available to the user, the next screen is the screen which appears after user taps on SOS option. It is asking user if he is sure that he wants to send the SOS.

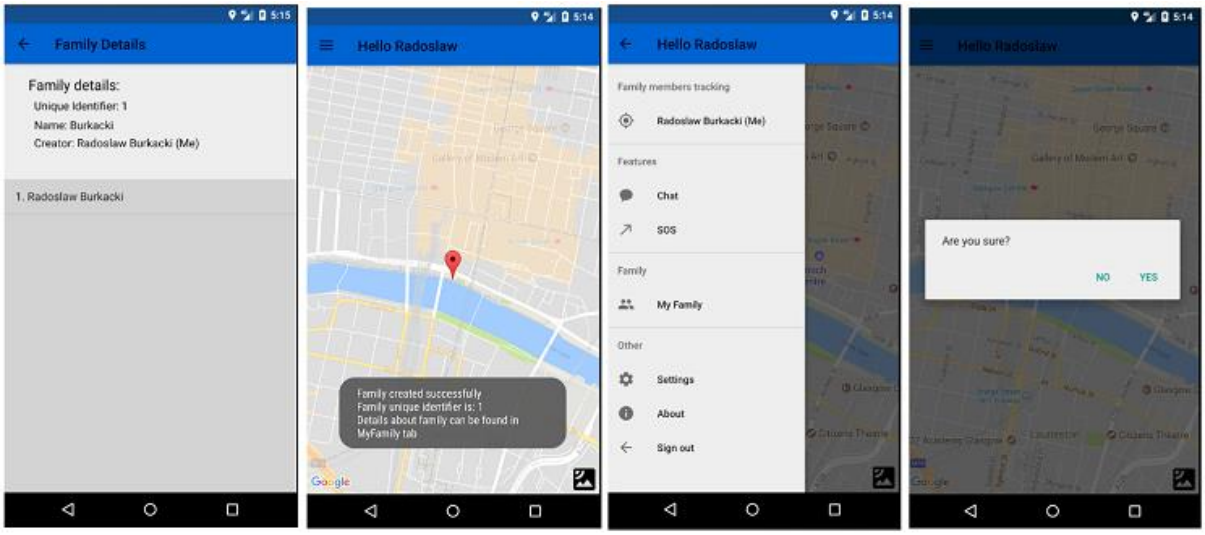

Figure 38- Map, family details, SOS and side menu screens implementation

6.6.3.4 Pre-chat and chat

Going from left to right the first screen which appears after user taps on chat button in the drawer menu, it allows user to choose a family member, second screen is the chat screen all messages are listed here and on bottom side of the screen there is a section which allows user to send messages.

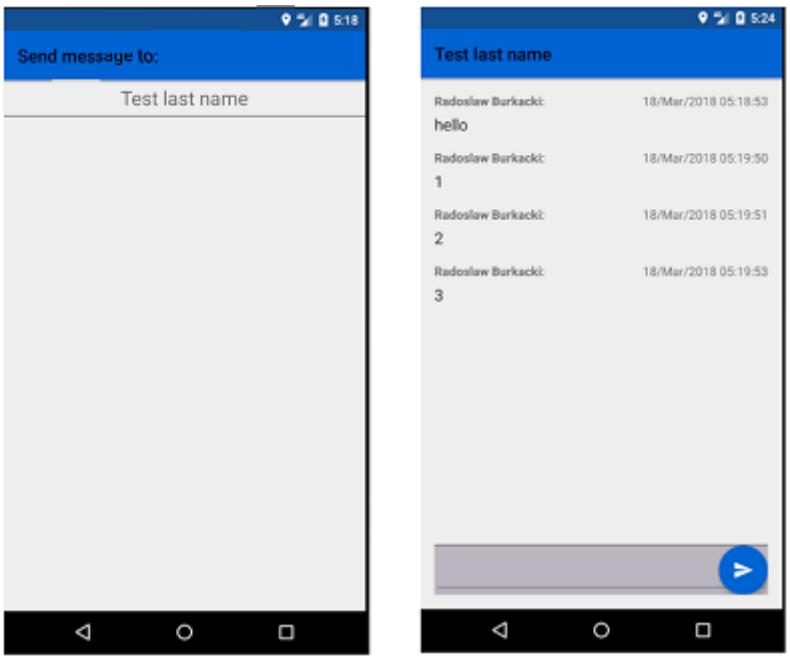

Figure 39- Pre-chat and chat screens implementation

6.6.3.5 Settings menu, clear chat history, marker colours

Going from left to right the first screen is asking user if he is sure that he wants to delete all the messages, this screen is displayed after user taps on clear chat history setting. The next screen is the settings menu screen it is displayed after user clicks on settings in the drawer menu. The last one is the marker colours.

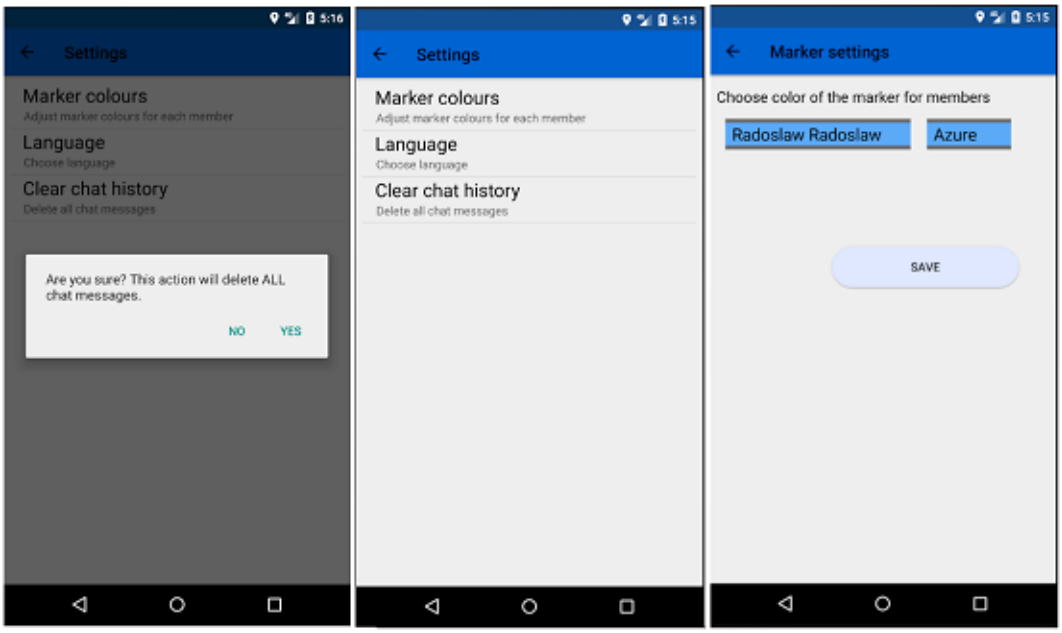

Figure 40- Settings, clear chat history and marker colours screens implementation

6.6.3.6 Language setting, main menu(map) in Polish and drawer menu in Polish

Going from left to right the first screen is allowing user to pick language this pop-up menu is displayed after user taps on language in the settings menu, the next screen is the main menu which is displayed after user chooses language, this screen is displayed in Polish, last screen is the drawer menu which is also displayed in Polish.

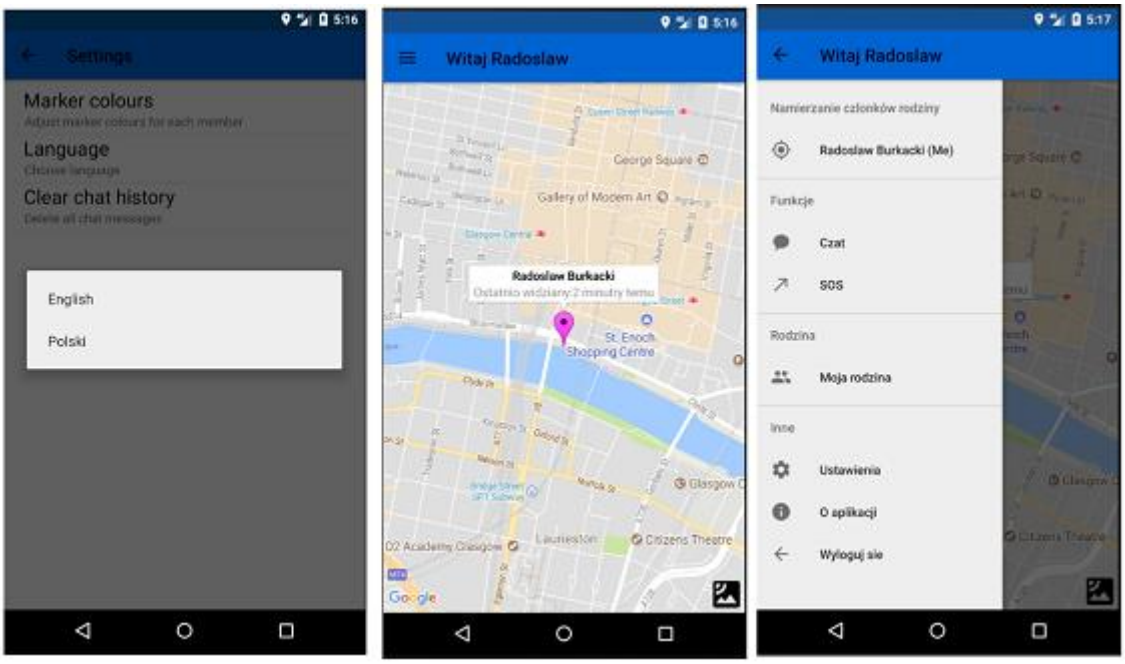

Figure 41- Language screens implementation

6.6.4 Notification implementation

All the data which will be sent to the client via FCM service will be sent as notification. Each notification will trigger sound and vibration to let the user know that new notification has been received. Received notifications can be in the notification section. Each notification within the family centre application is implemented via this code:

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this)
        .setAutoCancel(true)
        .setContentTitle("Title")
        .setContentText("Notification description")
        .setSmallIcon(R.drawable.common_google_signin_btn_icon_dark);

    NotificationManager manager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

    manager.notify(0, builder.build());
```

Notification has been highlighted with red colour:

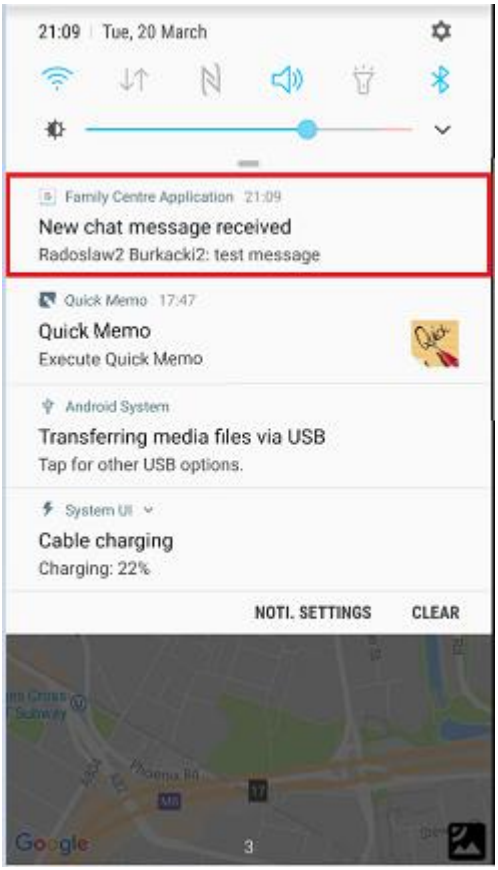

Figure 42- Notification screen implementation

6.7 Server implementation

This section will focus on server implementation, it will contain all classes and functions which are used to build server sided software which is responsible for handling all the functionality which is available to users.

6.7.1 List of classes and functions

Classes will be split into six sub-sections including: model which is used to store data, controller which is an entry point for all the requests, service which is responsible for the logic of the software, Crud repository which is responsible for communication with database and table creation and other.

6.7.1.1 Model classes

Model classes are used to store data, they can be used to create user object and then get user data and store it inside of that object. Model classes on server-side are using spring annotations which are used for various purposes including:

- @Entity – Is used to state that a table needs to be created based on that class.
- @Table – Used to choose table name.
- @Id – States that the attribute is the id.
- @GeneratedValue – States that the value must be generated (id auto increase).
- @Column – Used to state that field is unique and can be used to choose name for column.
- @Transparent – Attribute will be ignored
- @JsonCreator – Used to create JSON file.

6.7.1.1.1 User class

User class is used to store data about the user. Spring annotations are used to create table out of User class.

List of functions:

Function	Purpose
public User()	Empty user constructor, its used to create empty user object.
public User(Long id, String email, String password, String fname, String lname)	It is a User constructor which expects 5 variables to be passed into it, these variables will be used to create new instance of user object.
public long getId()	Used to get user id.
public String getEmail()	Used to get user email.
public void setEmail(String email)	It is expecting 1 variable to be passed in to it, used to set email of user object.
public String getPassword()	Used to get user password.
public void setPassword(String password)	It is expecting 1 variable to be passed into it, used to set password of user instance.
public String getFname()	It is used to get user first name.
public void setFname(String fname)	It is expecting 1 variable to be passed into it, used to set first name of user instance.
public String getLname()	It is used to get user last name.
public void setLname(String lname)	It is expecting 1 variable to be passed into it, used to set last name of user instance.

Table 23- List of functions (User. Class)

6.7.1.1.2 Family class

Family class is used to store data about family. Spring annotations are used to create table out of Family class. List of functions:

Function	Purpose
public Family()	Empty family constructor, its used to create empty family object.
public Family(Long creatorId, String familyName, String joiningPassword, List<User> familyMembers)	It is a Family constructor which expects 4 variables to be passed into it, these variables will be used to create new instance of family object.
public long getId()	Used to get family id.
public Long getCreatorId()	It is used to get creator id.
public void setCreatorId(Long creatorId)	It is used to set creator id, it is expecting 1 variable to be passed into it.
public String getFamilyName()	It is used to get family name.
public void setFamilyName(String familyName)	It is used to set family name, it is expecting 1 variable to be passed into it.
public String getJoiningPassword()	It is used to get joining password.
public void setJoiningPassword(String joiningPassword)	It is used to set joining password, it is expecting 1 variable to be passed into it.
public List<User> getFamilyMembers()	It is used to get all family members.
public void setFamilyMembers(List<User> familyMembers)	It is used to set family members, it is expecting 1 variable to be passed into it.
public void addFamilyMember(User u)	It is used to add single family member to family, it is expecting user object to be passed into it.
public String toString()	Used to display all attributes and values of family.

Table 24- List of functions (Family. Class)

6.7.1.1.3 Message Class

Message class is used to store data about message. Spring annotations are used to create table out of Message object. List of functions:

Function	Purpose
public Message ()	Empty message constructor, its used to create empty message object.
public Message (long messageId, long fromId, long toId, String message, String date)	It is a Message constructor which expects 5 variables to be passed into it, these variables will be used to create new instance of message object.
public long getMessageId()	It is used to get message id.
public long getFromId()	It is used to get from id.
public long getToId()	It is used to get to id.
public String getMessage()	It is used to get message text.
public String getDate()	It is used to get date.
public void setMessageId(long messageId	It is used to set message id, it is expecting 1 variable to be passed into it.
public void setFromId(long fromId)	It is used to set from id, it is expecting 1 variable to be passed into it.
public void setToId(long toId)	It is used to set to id, it is expecting 1 variable to be passed into it.
public void setMessage(String message)	It is used to set message id, it is expecting 1 variable to be passed into it.
public void setDate(String date)	It is used to set date, it is expecting 1 variable to be passed into it.

6.7.1.1.4 FCM Token class

FCM Token class is used to store data about FCM tokens. Spring annotations are used to create table out of FCM Token object. List of functions:

Function	Purpose
public FCMToken()	Empty FCMToken constructor, its used to create empty FCMToken object.
public Long getUserId()	Used to get user id.
public void setUserId(Long userId)	Used to set user id.
public String getMyFCMToken()	Used to get FCM token.
public void setMyFCMToken(String myFCMToken)	Used to set FCM token.
public String toString()	Used to print all data about the object.

Table 25- List of functions (FCM token. Class)

6.7.1.1.5 Last Known Coordinates class

Last Known Coordinates class is used to store data about location coordinates. Spring annotations are used to create table out of Last Known Coordinates object. List of functions:

Function	Purpose
public LastKnownCoordinates()	Empty LastKnownCoordinates constructor, its used to create empty LastKnownCoordinates object.
public long getId()	Used to get id.
public void setId(long id)	Used to set id.
public double getLatitude()	Used to get latitude.
public void setLatitude(double lat)	Used to set latitude.
public double getLongitude()	Used to get longitude.
public void setLongitude(double longitude)	Used to set longitude.
public void setDateTime(LocalDateTime dateTime)	Used to set date and time.
public void getDifference()	Used to calculate value based on which system can detect when the location update was sent by client.

Table 26- List of functions (Last known coordinates. Class)

6.7.1.1.6 Join Family class

Join Family class is used to join user to families. List of functions:

Function	Purpose
public long getFamilyId()	Used to get family id.
public void setFamilyId(long familyId)	Used to set family id.
public String getFamilyPassword()	Used to get family password.
public void setFamilyPassword(String familyPassword)	Used to set family password.
public Long getUserId()	Used to get user id.
public void setUserId(Long userId)	Used to set user id.
public String toString()	Used to print all data from this object instance.

Table 27- List of functions (Join family. Class)

6.7.1.1.7 Family Members class

Family members class is used to store data about family members. Spring annotations are used to create table out of family members object. This object is used to link users with families. List of functions:

Function	Purpose
public FamilyMember()	Empty FamilyMember constructor, its used to create empty family Member object.
public FamilyMember(Long memberId, Long familyId)	It is a family members constructor which expects 2 variables to be passed into it, these variables will be used to create new instance of family members object.
public Long getMemberId()	Used to get member id.
public void setMemberId(Long memberId)	Used to set member id.
public Long getFamilyId()	Used to get family id.
public void setFamilyId(Long familyId)	Used to set family id.

Table 28- List of functions (Family members. Class)

6.7.1.2 Controller classes

Controller classes are the entry point for each request which is coming in from the client application. In these classes spring annotation are used:

- @RestController – This annotation lets spring framework know that this class it the controller.
- @Autowirder – Its used for dependency injection.
- @RequestMapping – This annotation is telling spring framework that mapping is requested, this annotation is used to choose which HTTP method will be accepted and what is the URL address for it.
- @RequestBody – Used to convert JSON data to objects.
- @PathVariable – Used to accept data from URL.

6.7.1.2.1 User controller class

User controller is responsible for handling user-based actions, like registering, login in and getting list of all users. List of functions:

Function	Purpose
public ResponseEntity registerNewUser(@RequestBody User user)	Used to register new user.
public User getUser(@PathVariable String email)	Used to get user from database.
public List<User> getAllUsers()	Used to get all users from database.

Table 29- List of functions (User controller. Class)

6.7.1.2.2 Chat controller class

Chat controller is used to save messages to database. List of functions:

Function	Purpose
public ResponseEntity SaveChatMessage(@RequestBody Message m)	Used to save chat message.

Table 30- List of functions (Chat controller. Class)

6.7.1.2.3 Family controller class

Family controller is used to manage families. List of functions:

Function	Purpose
public ResponseEntity createFamily(@RequestBody Family family)	Used to save new family to database.
public List<Family> getAllFamilies()	Used to get list of all families from database.
public ResponseEntity addUserToFamily(@RequestBody JoinFamily jf)	Used to add new user to family.
public ResponseEntity CheckIfUserIsFamilyMemberById(@PathVariable long id)	Used to check if user is a member of any family.
public ResponseEntity getFamilyByUserID(@PathVariable long id)	Used to get family by user id.
public ResponseEntity removeUserFromFamily(@RequestBody JoinFamily jf)	Used to delete user from family.

Table 31- List of functions (Family controller. Class)

6.7.1.2.4 FCM controller class

FCM controller is used to save FCM tokens to database. List of functions:

Function	Purpose
public void saveFCMToken(@RequestBody FCMToken fcmToken)	Used to save FCM token to the database.

Table 32- List of functions (FCM controller. Class)

6.7.1.2.5 Location controller class

Location controller is used to manage user coordinates. List of functions:

Function	Purpose
public ResponseEntity getLocationCoordinates(@PathVariable Long userid)	Used by clients to get location coordinates of a family member.
public ResponseEntity addNewLocationCoordinates(@PathVariable Long userid, @RequestBody LastKnownCoordinates lastKnownCoordinates)	Used to save location coordinates to database.
public List<LastKnownCoordinates> getAllUsers()	Used to get coordinates of all users.

Table 33- List of functions (Location controller. Class)

6.7.1.2.6 SOS controller class

SOS controller is used by clients to send SOS. List of functions:

Function	Purpose
public ResponseEntity sendSOS(@PathVariable long id)	Used to send SOS notification to all family members.

Table 34- List of functions (SOS controller. Class)

6.7.1.3 Service classes

Service classes are responsible for all the logic which is behind the software. Each of these classes is not communicating directly with the database, these classes use instances of repository classes which are used for communication with database.

6.7.1.3.1 User service class

This class is responsible for handling all the logic with user actions. List of functions:

Function	Purpose
public ResponseEntity register (User user)	This function is used to register new user.
public User getUser(String email)	This function is used to get user object.
public List<User> getAllUsers()	This function is used to get all users.

Table 35- List of functions (User Service. Class)

6.7.1.3.2 Chat service class

This class is responsible for all logic which is behind the chat actions. List of functions:

Function	Purpose
public ResponseEntity addMessage(Message m)	Used to save message to database.

Table 36- List of functions (Chat Service. Class)

6.7.1.3.3 Family service class

This class is responsible for all logic which is behind the family-based actions. List of functions:

Function	Purpose
public ResponseEntity CheckIfUserIsFamilyMemberById(Long id)	It is used to check if user is already a member of family via user id.
public ResponseEntity createFamily(Family family)	It is used to create new family.
public ResponseEntity getFamilyByUserID(long id)	Used to get family by user id.
public ResponseEntity addUserToFamily(JoinFamily jf)	Used to add family to database.
public void sendNotificationAboutNewUser(Long familyid, Long userid)	Used to send notification which lets users know that new user has joined their family.
public List<Family> getAllFamilies()	Used to get all families.
public ResponseEntity removeUserFromFamily(long famid, long id)	Used to remove user from family.

Table 37- List of functions (Family Service. Class)

6.7.1.3.4 FCM service class

This class is responsible for handling the FCM tokens and for sending all the requests to the FCM. List of functions:

Function	Purpose
public void saveFCMToken(FCMToken fcmToken)	Used to save FCM token to database.
public void sendSOS(List<String> FCMtokenList, String name)	Used to send SOS notification, this class is using ok http client to send request to FCM.
public void sendNewUserNotification(List<String> FCMtokenList, String membername)	Used to send notification about new user, this class is using ok http client to send request to FCM.
public void sendPrivateChatMessage(Message m)	Used to send notification about new private chat message, this class is using ok http client to send request to FCM.
public void sendUserRemovedNotification(List<String> FCMtokenList, User user)	Used to send notification about removed user from family, this class is using ok http client to send request to FCM.

6.7.1.3.5 Location service class

This class is responsible for all logic behind the location coordinates service. List of functions:

Function	Purpose
public ResponseEntity saveCoordinates(LastKnownCoordinates lastKnownCoordinates)	Used to save coordinates to the database.
public ResponseEntity getLastKnownCoordinates(Long userid)	Used to get coordinates of family members.
public List<LastKnownCoordinates> getAllLastKnownCoordinates()	Used to get coordinates of all users.

Table 38- List of functions (Location service. Class)

6.7.1.3.6 SOS service class

This function is responsible for logic behind SOS action. List of functions:

Function	Purpose
public ResponseEntity sendSOS(long userid)	Used to send SOS to all family members.

Table 39- List of functions (SOS service. Class)

6.7.1.4 Crud Repository interfaces

All interfaces which are in this section are extending curd repository. All the functions which are within these interfaces are used for the direct communication with the database.

6.7.1.4.1 Family Member Repository interface

This interface is responsible for all the communication to database regarding family member objects. List of functions:

Function	Purpose
public boolean existsByMemberId(Long id)	Used to communicate with database, it checks if user is already in the database.
public FamilyMember findFamilyMemberByMemberId(Long id)	Used to communicate with database, it is used to find family member via member id.
public List<FamilyMember> findFamilyMemberByFamilyId(long id)	Used to communicate with database, it is used to find family member via family id.
public void removeFamilyMemberByMemberId(long id)	Used to communicate with database, it is used to remove family member via member id.

Table 40- List of functions (Family member repository. Class)

6.7.1.4.2 Family Repository interface

This interface is responsible for all the communication to database regarding family objects. List of functions:

Function	Purpose
public Family findFamilyById(long id)	Used to communicate with database, it is used to find family via id.

Table 41- List of functions (Family repository. Class)

6.7.1.4.3 FCM Token Repository interface

This interface is responsible for all the communication to database regarding FCM token objects. List of functions:

Function	Purpose
public FCMToken findFCMTokenByUserId(Long id)	Used to communicate with database, it is used to find FCM token via user id.
public FCMToken findByMyFCMToken(String fcmtoken)	Used to communicate with database, it is used to find FCM token via fcm token.

Table 42- List of functions (TCM token repository. Class)

6.7.1.4.4 Last known coordinates repository interface

This interface is responsible for all the communication to database regarding last known coordinates objects. List of functions:

Function	Purpose
public LastKnownCoordinates findLastKnownCoordinatesById(Long id);	Used to communicate with database, it is used to find Last known coordinates via user id.

Table 43- List of functions (Last known coordinates repository. Class)

6.7.1.4.5 User repository interface

This interface is responsible for all the communication to database regarding user objects. List of functions:

Function	Purpose
public User findUserByEmail(String email);	Used to communicate with database, it is used to find user via user email.

public User findUserById(Long id);	Used to communicate with database, it is used to find user via user id.

Table 44- List of functions (User repository. Class)

6.7.1.5 Other

6.7.1.5.1 Main class (Family centre application)

The last class is the main class which is called when the application is started. It contains two annotations:

- @SpringBootApplication – Lets spring framework know that this application is using spring boot.
- @Bean – States that next function is a bean.

List of functions:

Function	Purpose
public static void main(String[] args)	This function is called when the server-side software is stared, it is starting spring application.

Table 45- List of functions (Main class. Class)

7 Testing

7.1 Device compatibility

This section will be focused on the compatibility of android application. In the user requirements it was stated that the application must be compatible with android API 15 operating systems and above. Three different android-based devices will be used for device compatibility testing. Application will be tested on the following devices:

- Samsung Galaxy S7 Edge based on Android 7.0 Nougat (API 24)
- Xiaomi Red Mi Note 4 based on Android 7.0 Nougat (API 24)
- Sony Xperia SP based on Android 4.3 Jelly Bean (API 18)

Each device will be used to do various actions, the following table represents test actions and the results which have been captured via using each device.

Test action	Samsung S7 Edge (API 24)	Xiaomi Red Mi Note 4 (API 24)	Sony Xperia SP (API 18)
Register	Pass	Pass	Pass
Login	Pass	Pass	Pass
Family Join	Pass	Pass	Pass
Family Creation	Pass	Pass	Pass
Map usage	Pass	Pass	Pass
Family member tracking	Pass	Pass	Pass
Family details	Pass	Pass	Pass
Sending location	Pass	Pass	Pass
Chat	Pass	Pass	Pass
SOS	Pass	Pass	Pass
Settings menu	Pass	Pass	Pass
Marker colours setting	Pass	Pass	Pass
Language setting	Pass	Pass	Pass
Clear chat history setting	Pass	Pass	Pass
Notification	Pass	Pass	Pass
Sound and vibration	Pass	Pass	Pass
User interface	Pass	Pass	Pass

Table 46- Compatibility testing table

7.2 Test cases

This section will focus on testing the android application. Aspects which will be tested in the testing phase include: input validation and error handling.

7.2.1 Registration testing

This section will focus on testing registration process. Testing will check if application is validating user input and if application is displaying appropriate messages in each error case.

Test scenario	Test case	Test data	Expected result	Actual result	Pass/ Fail
Verify register part 1 error handling	Invalid email and passwords	Email: “” Password: “” Re-password: “”	Message: “Please fill in all boxes”	Message: “Please fill in all boxes”	Pass
	Valid email, invalid passwords	Email: “1@1.com” Password: “” Re-password: “”	Message: “Please fill in all boxes”	Message: “Please fill in all boxes”	Pass
	Valid email, invalid passwords	Email: “1@1.com” Password: “11111” Re-password: “11111”	Message: “Password is too short”	Message: “Password is too short”	Pass
	Valid email, invalid re-password	Email: “1@1.com” Password: “111111” Re-password: “222222”	Message: “Passwords are not identical”	Message: “Passwords are not identical”	Pass
	Invalid email, Valid passwords	Email: “example.com” Password: “111111” Re-password: “111111”	Message: “Email is invalid”	Message: “Email is invalid”	Pass
	Valid Email and password	Email: “1@1.com” Password: “111111” Re-password: “111111”	Register part 2 screen displayed(Success)	Register part 2 screen displayed(Success)	Pass
Verify register part 2 error handling	Invalid first name and last name	First name: “” Last name: “”	Message: “Please enter first name and last name”	Message: “Please enter first name and last name”	Pass
	Valid First name and invalid last name	First name: “Radoslaw” Last name: “”	Message: “Please enter first name and last name”	Message: “Please enter first name and last name”	Pass
	Invalid first name and valid last name.	First name: “” Last name: “Burkacki”	Message: “Please enter first name and last name”	Message: “Please enter first name and last name”	Pass

	Valid last name and first name	First name: "Radoslaw" Last name: "Burkacki"	Main menu screen displayed and message: "Success! New account was created"	Main menu screen displayed and message: "Success! New account was created"	Pass
Verify register email usage for two different accounts	Valid data	Email: "1@1.com" Password: "111111" Re-password: "111111" First name: "Radoslaw" Last name: "Burkacki"	Message: "Success! New account was created"	Message: "Success! New account was created"	Pass
	Invalid email (its already in use) and valid data	Email: "1@1.com" Password: "111111" Re-password: "111111" First name: "Radoslaw" Last name: "Burkacki"	Message: "Fail! Email is already in use"	Message: "Fail! Email is already in use"	Pass
Verify register part network error handling	Attempt to register without internet connection	No internet connection(disabled)	Message: "Fail! Check your internet connection"	Message: "Fail! Check your internet connection"	Pass

Table 47- Register testing results

All registration tests have been passed successfully.

7.2.2 Login and family setup testing

This section will focus on testing login and family setup process. Testing will check if application is validating user input and if application is displaying appropriate messages in each error case.

Test scenario	Test case	Test data	Expected result	Actual result	Pass/ Fail
Verify login error handling	Invalid email and password	Email: "" Password: ""	Message: "Make sure that all boxed are filled in"	Message: "Make sure that all boxed are filled in"	Pass
	Invalid email format and valid password	Email: "2.com" Password: "111111"	Message: "Please fill in all boxes"	Message: "Please fill in all boxes"	Pass
	Valid email and wrong password	Email: "1@1.com" Password: "111112"	Message: "You have entered wrong email/password. Please try again."	Message: "You have entered wrong email/password. Please try again."	Pass
	Valid email and password	Email: "1@1.com" Password: "111111"	Message: "Login successful"	Message: "Login successful"	Pass
Verify login network error handling	Attempt to register without internet connection	No internet connection(disabled)	Message: "Fail! Check your internet connection"	Message: "Fail! Check your internet connection"	Pass
Verify if proper screen is displayed after user logs in	Login attempt	User is not a family member	Family setup screen displayed	Family setup screen displayed	Pass
		User is a member of family	Map screen displayed	Map screen displayed	Pass
Verify family creation error handling	Invalid name and passwords	Family name: "" Passwords:""	Message: "Make sure that all boxes are filled in correctly"	Message: "Make sure that all boxes are filled in correctly"	Pass
	Valid family name and password invalid re-password	Family name: "Burkacki" Password:"111111" Re-password:"111112"	Message: "Make sure that all boxes are filled in correctly"	Message: "Make sure that all boxes are filled in correctly"	Pass
	Invalid family name and valid passwords	Family name: "" Passwords:"111111"	Message: "Make sure that all boxes are filled in correctly"	Message: "Make sure that all boxes are filled in correctly"	Pass
	Valid family name and passwords	Family name: "Burkacki" Password:"111111" Re-password:"111111"	Message: "Family created successfully"	Message: "Family created successfully"	Pass
Verify family joining error	Invalid data	Id: "" Password: ""	Message: "Make sure that all boxes	Message: "Make sure that all boxes are	Pass

handling			are filled in correctly"	filled in correctly"	
	Valid id and invalid password	Id: "1" Password: "111112"	Message: "Wrong family join password"	Message: "Wrong family join password"	Pass
	Family id (not existing) and random password	Id: "2" Password: "123456"	Message: "Family does not exist"	Message: "Family does not exist"	Pass
	Valid family id and valid password	Id: "1" Password: "111111"	Message: "You have joined family"	Message: "You have joined family"	Pass

Table 48- Login and family setup testing results

All login tests have been passed successfully.

7.2.3 Main screen (Map) testing

For this test 2 users have been registered and joined the same family.

Test scenario	Test case	Test data	Expected result	Actual result	Pass/ Fail
User interface responsiveness testing	Testing drawer menu access via button	Tap on menu button	Drawer menu opens	Drawer menu opens	Pass
	Testing drawer menu access via swipe (left to right)	Swiping from left to right	Drawer menu opens	Drawer menu opens	Pass
	Testing tracking my location	Logged in as user A, tapping on user A in tracking menu	User A location focused on map	User A location focused on map	Pass
	Testing tracking of other users	Logged in as user A, tapping on user B in tracking menu	User B location focused on map	User B location focused on map	Pass
	Moving map (swiping)	Swiping on map.	Map moving.	Map moving.	Pass
	Zooming map	Zooming gesture	Map zooming	Map zooming	Pass
	Testing button which is overlaid on the map	Button press	Map type change	Map type change	Pass
	Chat button(menu) testing	Chat item tapped	Pre-Chat screen displayed	Pre-Chat screen displayed	Pass
	SOS button(menu) testing	SOS item tapped	Pop-up message displayed	Pop-up message displayed	Pass
	My family button(menu) testing	My Family item tapped	Family details screen displayed	Family details screen displayed	Pass
	Settings button(menu) testing	Settings item tapped	Settings screen displayed	Settings screen displayed	Pass
	About button(menu) testing	About item tapped	About screen displayed	About screen displayed	Pass
	Sign out button(menu) testing	Sign out item tapped	Application closed	Application closed	Pass

Table 49- Map screen testing results

All tests have been passed successfully.

7.2.4 Chat testing, settings testing

For this test 2 users have been registered and joined the same family. Both users have been logged in.

Test scenario	Test case	Test data	Expected result	Actual result	Pass/ Fail
Chat functionality	Sending message	User A sends message to user B. Message 'test123'	User B received notification with sound and vibration, message "test123" displayed in chat	User B received notification with sound and vibration, message "test123" displayed in chat	Pass
Chat input validation	Chat input testing	Message ""	Message cannot be send as it is null	Message cannot be send as it is null	Pass
Settings functionality	Testing marker colours setting	Colour marker for user B set to blue & saved.	User B marked on map with blue marker.	User B marked on map with blue marker.	Pass
	Testing language setting	Language changed to Polish	All labels, messages displayed in Polish.	All labels, messages displayed in Polish.	Pass
	Clearing chat history.	Cleared chat history.	Chat with user B contains no messages.	Chat with user B contains no messages.	Pass
Family details functionality	Deleting user B from family.	Deleted user B.	User B removed from family details list and from drawer menu.	User B removed from family details list and from drawer menu.	Pass

Table 50- Chat and settings testing results

All tests have been passed successfully.

8 Evaluation

The finished software will be evaluated via feedback from potential users, if this method will find out that there is a need for improvement of any aspect of application, then it will be implemented in this chapter and documented.

8.1 Evaluation research method

To gather feedback data about the finished application from users a quantitative method will be used. A questionnaire will be created and sent out to participants, the aim of the questionnaire is to find out about user experience while using the application, the compatibility and stability of application and if there is need to improve any aspect of application.

8.2 Questionnaire

Questionnaire has been created via google forms and is available here:
https://goo.gl/forms/eCPoWJ0RIdFXIzIS2

Each participant which has taken part in the questionnaire was asked to download (link available in the questionnaire) the android application and to try it, in order to allow participants to use the application the back-end software had to be available to the internet requests, to achieve that software has been deployed on a personal computer on a network which had access to the internet, router has been configured that if any requests are sent to the router on port 1000 then these request will be passed to the personal computer which had the family centre application server software running.

8.2.1 Ethics approval of questionnaire

Before the questionnaire could be distributed to participants I had to make sure that there are no ethical issues with it, so the questionnaire has been first sent out to the supervisor for approval, this questionnaire has been approved by the supervisor, so it is ready for distribution to participants.

8.2.2 Questionnaire content

The questionnaire contains brief description of the software and its purpose, the link to the application is available in that description. Questionnaire contains one section which 4 questions in total, each participant must answer all of them. First 3 questions allow user to select answer in scale from 1(disagree) – 5(agree). The last question allow user to pick answer. Questionnaire contains the following content:

- Family Centre Application did work on my device without any issues – Participant is able to select number from 1(disagree) – 5(agree).
- I enjoyed using Family Centre Application – Participant can select number from 1(disagree) – 5(agree).
- User interface was easy to use– Participant can select number from 1(disagree) – 5(agree).
- Which of these aspects of the application can be improved? – Participant can choose from 4 suggested options- Ease of use, User interface, Functionality and family management.

8.2.3 Questionnaire results

32 questionnaire responses have been received. The following charts have been generated from responses.

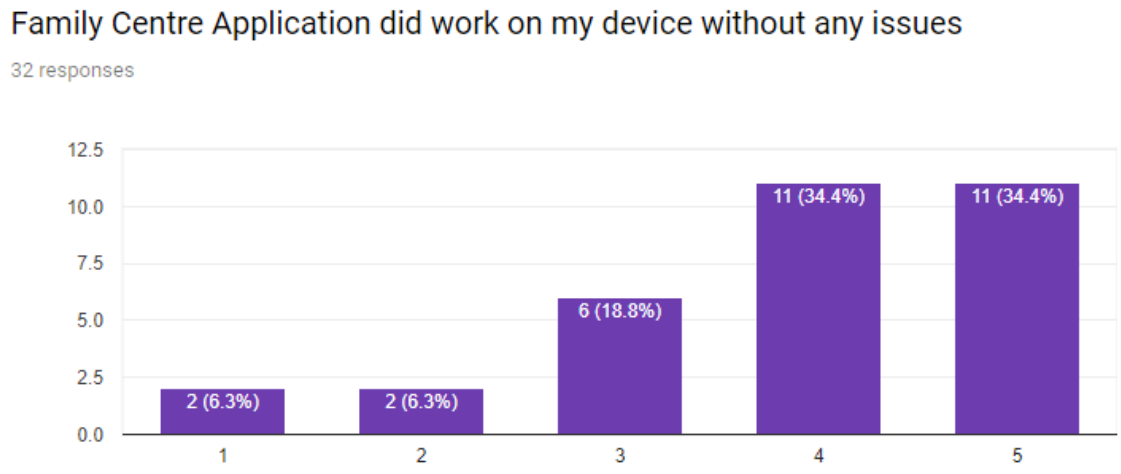

Figure 43- Evaluation - question 1 results

Question 1 asked participants about the experience which they had while using the application.

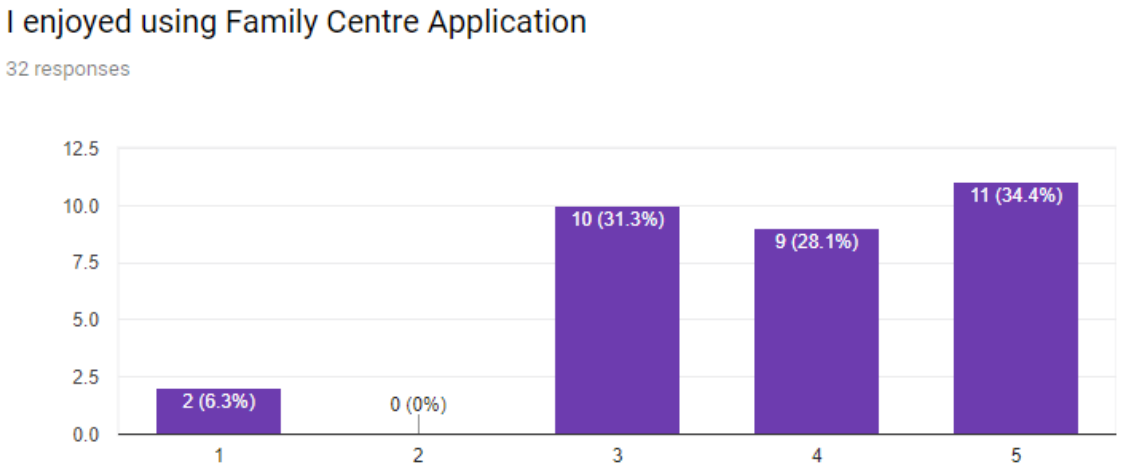

Figure 44- Evaluation - question 2 results

Question 2 asked participants if they have enjoyed using the application.

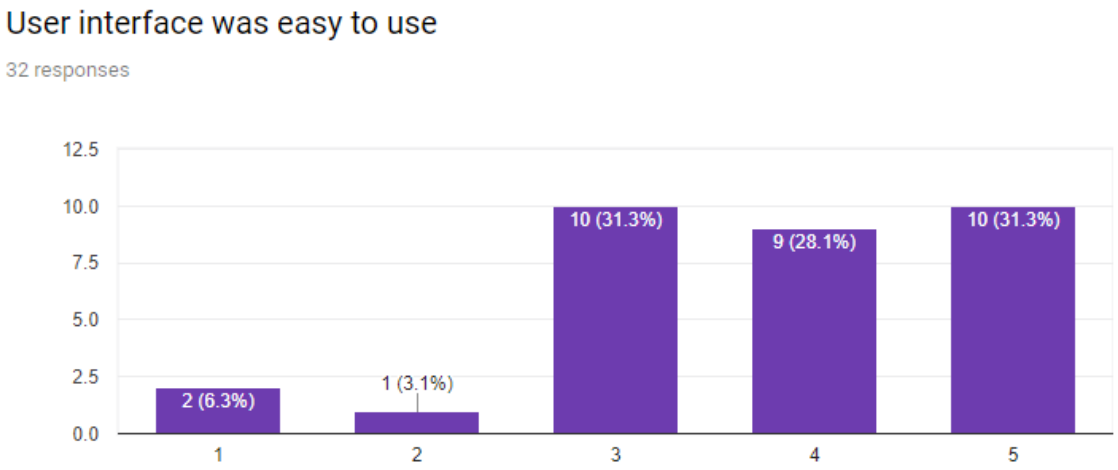

Figure 45- Evaluation - question 3 results

Question 3 asked participants if they think that the user interface was easy to use.

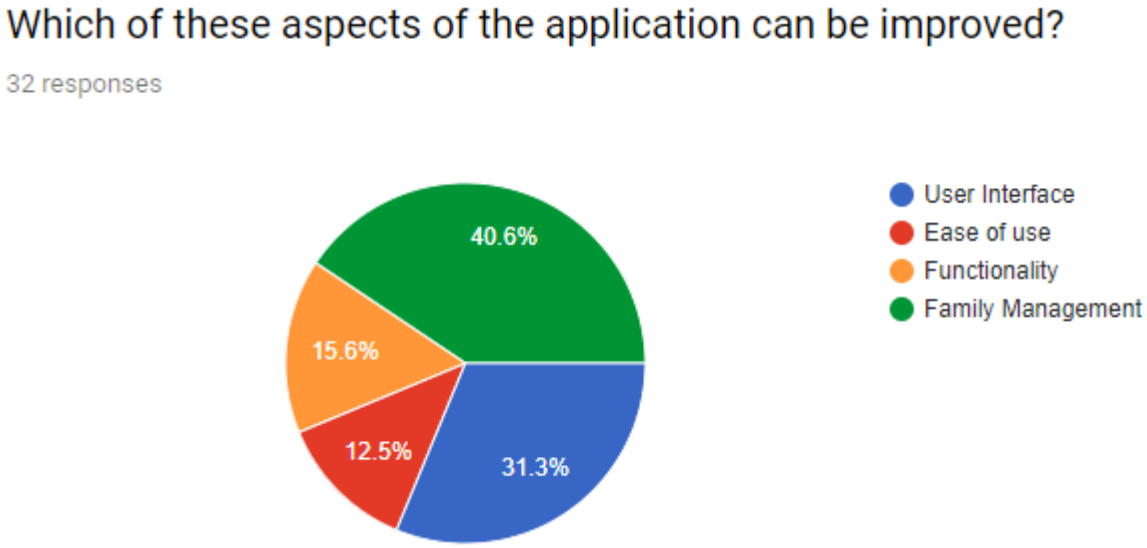

Figure 46- Evaluation - question 4 results

Question 4 asked participants what aspect of application should be improved.

The following table represents the results of the questionnaire:

	1	2	3	4	5
Question 1	2(6.3%)	2(6.3%)	6(18.8%)	11(34.4%)	11(34.4%)
Question 2	2(6.3%)	0(0.0%)	10(31.3%)	9(28.1%)	11(34.4%)
Question 3	2(6.3%)	2(6.3%)	10(31.3%)	9(28.1%)	10(31.3%)
	Functionality	Ease of use	User Interface	Family Management	
Question 4	5(15.6%)	4(12.5%)	10(31.3%)	13(40.6%)	

Table 51- Evaluation questionnaire result summary

8.2.4 Analysis of questionnaire results

According to question 1 results 22(68.8%) participants did not experience any issues, 6(18.8%) participants experience some issues and only 4(12.6%) experiences issues with the application.

According to question 2 results 20(62.5%) participants did enjoy usage of the application, 10(31.3%) participants stated that they have somewhat enjoyed the application and only 2(6.3%) participants did not enjoy the application.

According to question 3 results 19(59.3%) participants stated that they think that interface was easy to use, 10(31.3%) participants stated that application was somewhat easy to use and 4(12.3%) participants stated that they think that application is not easy to use.

According to question 4 results 5(15.6%) participants stated that functionality can be improved, 4(12.5%) stated that ease of use can be improved, 10(31.3%) participants stated that user interface can be improved and 13(40.6%) participants stated that family management can be improved.

Results state that the application needs to be improved that includes the user interface and the family management, these two aspects where highlighted by participants as the ones which need more attention.

8.3 Improvements plan

Two aspects of the application must be improved – user interface and family management. The following additional functions/improvements will be added to the application:

- Chat visibility improvements- Currently the application is using simple list items which are listed in one row. It's hard to recognise which messages was sent by user A and which message was sent by user B. To improve this a system of chat bubbles will be implemented, user which has sent the message will see his message on the right side of the screen and the background of the bubble will be blue, and the messages which are sent by other family members will be on the left side of the screen and the background will using orange colour.
- Family management – A function will be implemented to remove family members, only the creator will be able to use this function.

It's time to implement these features, results will be available in the next section.

8.4 Additional features implemented

As planted in the previous section two additional features has been implemented.

Going from left to right, first screen is the family details screen, at this screen the creator is able to tap on each family member this will trigger an popup list(second screen) to appear which will allow creator to remove user or to cancel the action, if user taps on remove user from family button then the list will disappear and family list will be updated(screen 3). Last screen is the chat screen which contains new bubble system implemented, for presentation purposes two messages has been sent. Message on the right side with blue background is the message which has been sent by user A and the other message which is on the left with orange background is the message which has been sent by user B.

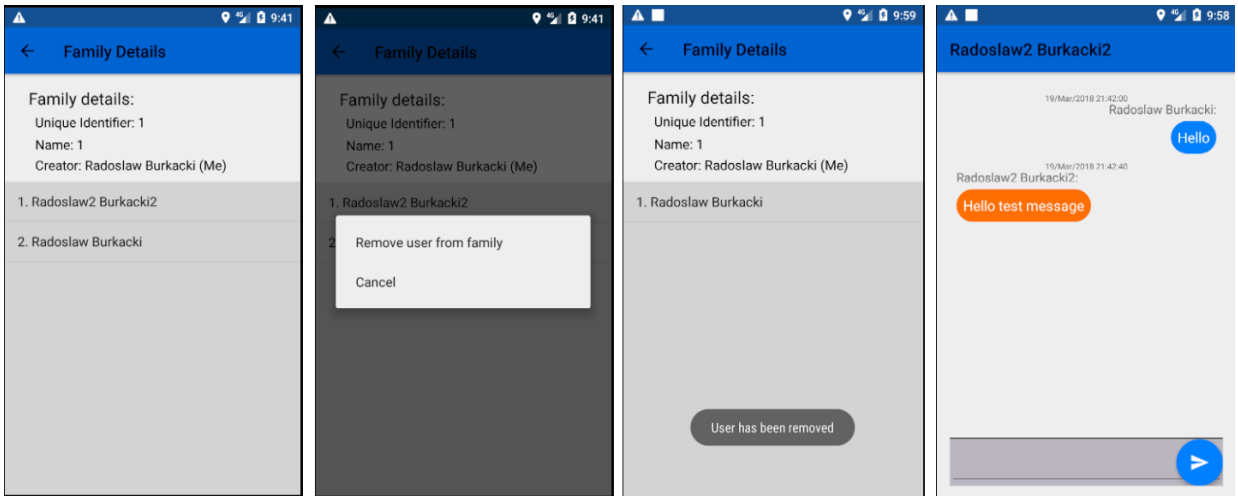

Figure 47- Evaluation software improvements

9 Conclusion

The main goal of the project was to develop client-server-based software which will be designed for family usage, many objectives had to be completed to achieve this final goal including: planning, research, design, analysis, implementation, testing and evaluation.

First class project goals (according to project specification) have been achieved:

- Client-Server family centre application implemented with additional features
- MySQL relational database has been developed
- Server software is connected to database and is using it to store data
- Planned security elements have been implemented (authorization, authentication, connection encryption and password hashing) and an additional one which is token-based communication
- Application has been evaluated

The software is focused on family usage, application requires users to register and login after that they can use a range of family-friendly functions including: family member tracking, SOS and chat. Users can adjust settings to their needs via the settings menu. Application is available in two languages – English and Polish and it is compatible with android devices which are using Android 4.0.3 or later.

The software which has been created has utilised current technologies and up to date security standards including: Spring framework, Android, Firebase Cloud Messaging, Google Maps, password hashing and salting, connection encryption, authentication, authorization and token-based communication, all these standards are used by professional developers around the world.

Two questionnaires have been created via Google forms and sent out to participants, the first one was used in the research stage to find out if there is a need for this type of application, based on the results of this questionnaire user requirements were captured, the second questionnaire was used in the evaluation stage to get feedback from users about the application, results from this questionnaire did let me know that some improvements of the application were needed including family management feature and user interface improvement which has been implemented.

Final state of the software has been fully tested and its running smoothly on Android devices which are based on android Ice Cream Sandwich (API 15) or later, all the functionality within the application are working correctly without any issues and are bug-free, although the application has been tested only on modern devices with use high resolution displays and one of the newest android versions, it is expected that old devices with low resolution displays and size might encounter some problems with text overlapping in some parts of the application(register, login and family details screens).

This application has potential for future expansion. Many things can be added to its features like family member location prediction or invite system (via email), but the most important thing for future development of this project would be to release an equivalent application for IOS system which is the second mostly used mobile platform, this would ensure that all family members will be able to use the application, because some family members could be using android devices and other might be using IOS devices.

Whole development process has been documented via git hub service, each major software update/improvement has been uploaded to git hub:

- Final android software can be found: https://github.com/RadoslawBurkacki/FamilyTrackingApplication
- Final server software can be found: https://github.com/RadoslawBurkacki/familycentre

10 Self-evaluation

At the start of this project when I had to choose project type I was sure that I want to develop software in a professional development process and to learn new technologies, I had experience in programming using JAVA, but it only included web applications and desktop applications, I had no previous experience with development of mobile applications which I always wanted to learn but never could motivate myself to do so, this project allowed me to learn about android application creation process and the structure of android applications.

In total I had five official meetings with my supervisor which helped me with the structure of this project. I have done a presentation to my supervisor and moderator which has demonstrated the idea behind the application and software in action.

Over the period of the honours project many challenges had to be overcome to ensure that the project succeeds. The biggest challenge which occurred during this project was the development of the android application, as I previously mentioned I had no experience in android application development, it took me lots of additional time to learn about android application development process, the techniques which should be used in the development process, the libraries and the activity uses.

The other challenge was the time management, balancing workload of the honours project, doing other modules and learning android application development was difficult as it required lots of time. I have managed to overcome this challenge by setting objectives/milestones every week for each of these activities, ensuring that these objectives were completed in a given week was key to success of this project and my studies.

This project was the most advanced and complex software development project which I have ever created, in future I will use this project as a demonstration of my skillset to the potential employers.

I think that the honours project went well and have been completed successfully according to the project specification, I have used all knowledge and skills which I learned during my studies to create this project to as high an academic standard as I could.

References:

Kim, L. (2015) 10 Most popular Programming Languages Today. [Online] Available: https://www.inc.com/larry-kim/10-most-popular-programming-languages-today.html [Accessed: 09 October 2017]

Sestoft, P. (2007) Numeric performance in C, C# and Java. IT University of Copenhagen Denmark. [Online] Version 0.7.1 of 2007-02-28, http://www.itu.dk/people/sestoft/smp2/csharpspeed.pdf [Accessed: 09 October 2017]

Miler, C. (2016) Latest Gartner data shows iOS vs Android battle shaping up much like Mac vs Windows. [Online] https://9to5mac.com/2016/08/18/android-ios-smartphone-market-share/ [Accessed: 09 October 2017]

Handy, G. (2012) What Language Should You Build Your App With? [Online] Available: http://mashable.com/2012/07/11/language-app/#zyAoelFSKgq7 [Accessed: 10 October 2017]

Google Platform Versions. [Online] Available: https://developer.android.com/about/dashboards/index.html [Accessed: 11 October 2017]

Kienitz, P. (2017) The pros and cons of Waterfall Software Development. [Online] Available: https://www.dcslsoftware.com/pros-cons-waterfall-software-development/ [Accessed: 17 October 2017]

Highsmith, J. (2001) History: The Agile Manifesto. [Online] Available: http://agilemanifesto.org/history.html [Accessed: 17 October 2017]

Singh, V. (2017) Client Server Architecture and HTTP Protocol. [Online] Available: http://toolsqa.com/client-server/client-server-architecture-and-http-protocol/ [Accessed: 17 October 2017]

Denning, S. (2016) Explaining Agile. [Online] Available: https://www.forbes.com/sites/stevedenning/2016/09/08/explaining-agile/#3b6614ef301b [Accessed 17 October 2017]

Rouse, M. (2008) object-oriented programming(OOP). [Online] Available: http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP [Accessed: 17 October 2017]

Branson, T. (2016) 8 Major Advantages of Using MySQL. [Online] Available: https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html [Accessed: 18 October 2017]

McLeod, S. (2008) Qualitative vs Quantitative. [Online] Available: https://www.simplypsychology.org/qualitative-quantitative.html [Accessed: 13 November 2017]

Joshi, V. (2017) WHY YOU SHOULD BE USING JSON VS XML. [Online] Available: https://blog.cloud-elements.com/using-json-over-xml [Accessed: 14 November 2017]

Roetman, J. (2017) How and when to use JSON Web Tokens for your services. [Online] Available: https://blog.codecentric.de/en/2017/08/use-json-web-tokens-services/ [Accessed: 14 November 2017]

Obugyei, E. (2016) Android Networking Tutorial: Getting Started. [Online] Available: https://www.raywenderlich.com/126770/android-networking-tutorial-getting-started [Accessed: 22/03/2017]

Connors, J. (2015) Installing Trusted Certificates into a Java Keystore. [Online] Available: https://blogs.oracle.com/jtc/installing-trusted-certificates-into-a-java-keystore [Accessed: 22/03/2017]

Khan, B. (2016) Firebase Cloud Messaging Tutorial for Android. [Online] Available: https://www.simplifiedcoding.net/firebase-cloud-messaging-tutorial-android/ [Accessed: 22/03/2017]