1. INTRODUCTION

1.1 Background

This Android Application is developed to track your Friends or Family or People travelling in groups towards a common destination. One of the traditional methods to get the current location of a person is to call that person & hope he/she gives you the correct information of the surroundings of their particular location. Another popular method is to ‘share live location’ on Whatsapp. While this is a good way to share 1’s location, it becomes a tedious task to track all the people at the same time. This Application overcomes the above problems & shows the live location of each user connected with each other in the Application in just a single click!

1.2 Objective

1. Get live location of connected users.
2. Share live location of connected users with eachother.

1.3 Purpose & Scope

1.3.1 Purpose

This Android Application provides facility to track Friends or Family or People travelling in groups towards a common destination. However, it can also be used by a user to track their loved ones to check if they reached their destination. User has the privilege to add, remove users from his/her account. User can also create a room & add other users to it. The people in the room can view each other’s location. Users can leave the room at any time if they wish too.

1.3.2 Scope

Scope of the project is :

- This app can be widely used by Bike Riders as they usually travel in groups.
- Also it can be used by anyone anywhere anytime.

1.3.3 Applicability

Ever wondered where the slowest rider of your group is? Or where in the world did the fastest rider of your group is? Or simply just where everyone is! Well, This is all now possible with this Android Application. An extention of Google Maps which allows you to keep track of your riding partners while on the tour. This application can also be used in various other scenarios like if your friends/family are coming at your place & you just like to know how far away they are from your place.

1.4 Achievements

- **Planned approach towards working :** The project will be working as planned. The data will be monitored, stored and maintained properly and will be user friendly.
- **Reliability :** The proposed system will be reliable, and will show correct data about other users.

1.5 Organisation of Report

The overall project is to develop an application which eases the user ability to track his family/friends. The remaining chapters will be discussing which technologies will be used to develop and test the proposed system. It will also discuss about requirements analysis and specifications and then Implementation of the system.

The remaining chapters of the project describes

- Technologies available for proposed system for Developing.
- Why Java-Kotlin ?
- Problems Definition.
- Requirements Specification
- Planning and Scheduling.
- System design
- Implementation

2. SYSTEM ANALYSIS

2.1 Existing System

Available technologies for developing the proposed system are as follows :

- C++
- C# (Xamarin)
- Html 5
- Hybrid Applications
- Java
- Kotlin

◆ **C++ :**

- *C++* is a middle-level programming language which can be used to develop Android Applications.
- Java, with the JVM-optimized byte-code, can generate pretty fast code, but native (i.e., machine code) can be faster and useful in areas such as gaming, physics simulations and signal processing.
- As C++ usually has no standard user Interface, the user-interface code is written in the native language and C++ used for the business logic.
- C++ has a smaller memory footprint, as it is nearer to the metal and has no garbage collection.
- C++ is a superset of C and compiles virtually all C programs, so it can reuse C software.

◆ C# (Xamarin) :

- C# is a programming language developed by Microsoft which can be used to develop Android Applications.
- Xamarin, a Microsoft owned software company has created a Cross Platform development tool which enables developers to develop iOS and Android apps in C# language.
- Xamarin is offered in different licenses from free to enterprise levels.

- The beauty of Xamarin is that despite the differences under the hood, Xamarin.iOS and Xamarin.Android (coupled with Microsoft's Windows SDKs) offer a seamless experience for writing C# code that can be re-used across all three platforms.
- Business logic, database usage, network access, and other common functions can be written once and re-used on each platform

◆ HTML5:

1. A HTML5 app refers to a mobile app built completely using HTML, CSS and Javascript only.
2. HTML5 apps are web apps and they must be run using the underlying OS browser.
3. A well written HTML5 app can be used even when the device is offline, or at the very least, show an error message.
4. HTML5 apps are portable across different OSes and device types.
5. HTML5 apps are generally cheaper to develop and maintain than native apps.

◆ Hybrid Applications :

- Hybrid apps are built using on language/framework like HTML5, CSS and Javascript and are then wrapped with native specific code for each desired mobile OS.
- A hybrid app is no different from a native app.
- Hybrid apps can be made available and distributed via the relevant app store, just like native apps.
- Hybrid apps have greater access to the native hardware resources than plain HTML5 apps, usually through the corresponding framework's own APIs.
- Popular hybrid app frameworks include Apache Cordova (formerly PhoneGap), Appcelerator Titanium, Appear IQ, CocconJS and Appzillon among others.

◆ Java :

- Java is a Programming Language developed by James Gosling at Sun Microsystems.
- Java is the official programming language for Android app development.
- It is the most widely used programming language for android application development.
- Java itself is used by Google for large parts of the Android internals.
- Java has many frameworks and classes for features like networking, threading, IO operations and thus, programmers can leverage these qualities in their apps.

◆ **Kotlin :**

- Kotlin is a statistically typed programming language that runs on the Java virtual machine.
- Kotlin is a Java-based programming language and interoperable.
- As of now, Koltin is an official Android Programming Language.
- Kotlin is easy & simple to use.
- Kotlin is crisp, concise, and reduces a lot of much of the boilerplate code.

2.2 Proposed System

The Proposed Android Application will use Java-Kotlin :

Why Java-Kotlin ?

Both Java & Kotlin are now the official languages for Android Development. While Java is the oldest & most widely used language for Android Development, Kotlin being a newly developed language has taken the world of Android by storm. Kotlin is a Java based programming language which runs on JVM (Java Virtual Machine). Hence it is interoperable with Java. Because Kotlin generates Java bytecode, we can use our favorite Java frameworks and libraries in Kotlin. Kotlin is much more productive, less boilerplate code, concise & so it can be used in many areas of the project. And well, what better place to start learning Kotlin than this, eh?

Advantages of Native Development over Xamarin :

- Unlike Native Languages, Xamarin has slightly delayed support for the latest platform updates since its impossible for third-party tools to provide the immediate support for the latest Android releases.
- Native development makes extensive use of open source technologies. With Xamarin, you have to use only the components provided by the platform and some .Net open source resources.
- Xamarin community is significantly smaller than those of Android Native Community.
- When using Xamarin.Android to build mobile apps with truly native look and feel, we still need to write a platform-specific layer of code. Thus, at least a basic knowledge of native technologies (Java/Kotlin for Android) is required.

- Xamarin’s main benefit is the ability to share our code across the platforms. Yet, we can only share the logic, UI code will be mostly platform-specific. This makes building games, rich custom UI, or complex animations in Xamarin pretty pointless.
- Depending on their type and complexity, Xamarin apps are typically larger than native ones.

2.3 Requirement Analysis

2.3.1 Problem Definition

Since very long, it’s been a problem of getting an accurate location of one or many people trying to reach a particular location. Another problem is of when going on a trip on different vehicles, a certain vehicle takes a wrong turn & finds himself stranded away from his buddies. While the destination will still be the same, its just nice to know where you’re buddies are at the moment.

2.3.2 Requirements Specification

Aim

The aim of the project is to build a system that can track many people at a given time.

Objective

To develop an application which gets the location of the connected users ttravelling towards a common destination & shares it among them.

Introduction

Travelling has become one of the most important things in a person’s life in today’s world. Each day more & more people are buying a personal vehicle for their daily use & occasional long tours. While going on a Tour it is important for the tourer to know the live location of his friends.

Purpose

The purpose of this project is to effectively overcome the problem of tracking many people simultaneously at a given time.

Scope

- This app can be widely used by Bike Riders as they usually travel in groups.
- Also it can be used by anyone anywhere anytime.

2.3.3 Feasibility Study

The feasibility study of the system examines the practicability of the project. A feasibility study is performed to determine whether the solution considered to accomplish the requirements is practical and workable in the system. Information such as resource availability, cost estimation for system development, benefits of the system to the institution after it is developed and cost to be incurred on its maintenance are considered during the feasibility study.

The objective of feasibility study is as follows:

- To analyze whether the system will meet the requirements.
- To determine whether the system can be implemented using the current technology and within the specified budget and schedule.

There are three types of feasibility study named as Technical feasibility, Economic feasibility and Operational feasibility.

1. **Technical Feasibility :** Technical feasibility examines the current resources (such as hardware and the software) and technology, which are required to accomplish user requirements in the system within the allocated time and budget. It also determines whether the relevant technology is stable and established. The technology which we will use for the development of the system is capable of achieving the objective of the system.
2. **Economic Feasibility :** Economic feasibility determines whether the required software is capable of generating financial gains for the institution. It involves the cost incurred on the system development, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out system development. As the estimated cost to be spent on the system development is less because most of the development software is available free of cost. Therefore, this system is technically and economically feasible.
3. **Operational Feasibility :** Operational feasibility determines the extent to which the required system performs a series of steps to solve the problems and user requirements. This

feasibility is dependent on human resources and involves visualizing whether the software will operate after it is developed and be operative once it is installed.

2.4 Software & Hardware Requirements

Developer Side

Software Requirements	Hardware Requirements
Android Studio 3.0 or higher Java jdk 5.0 or higher	4GB RAM or higher Intel Core i3 or higher Intel UHD 650 or higher

Client Side

Software Requirements	Hardware Requirements
Android v5.0 or higher	1GB RAM or higher Qualcomm Snapdragon 450 or higher Atleast 100MB free space

3. SYSTEM DESIGN

3.1 Module Division

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. An enterprise-level software application may contain several different modules, and each module serves unique and separate business_operations.

Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application. Modules are typically incorporated into the program (software) through interfaces.

For developing a proper working application, some basic modules need to be followed.

The modules of this project are :

1. Login
2. Show People
3. Send Friend Request
4. Notification
5. Accept / Decline Friend Request
6. See Friends Real-time Location

3.2 Planning & Scheduling

Planning and scheduling is a complicated part of software development. We planned our project in such a way that, all the small tasks, were covered thoroughly.

Gantt Chart

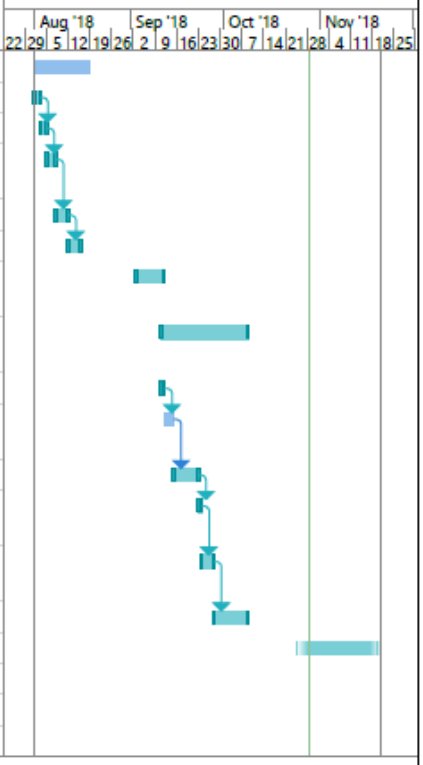
A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities.

Gantt charts are sometimes equated with bar charts. Gantt charts are usually created initially using an *early start time approach*, where each task is scheduled to start immediately when its prerequisites are complete. This method maximizes the float time available for all tasks.

In a progress Gantt chart, tasks are shaded in proportion to the degree of their completion: a task that is 60% complete would be 60% shaded, starting from the left. A vertical line is drawn at the time index when the progress Gantt chart is created, and this line can then be compared with shaded tasks. If everything is on schedule, all task portions left of the line will be shaded, and all task portions right of the line will not be shaded. This provides a visual representation of how the project and its tasks are ahead or behind schedule.

Linked Gantt charts contain lines indicating the dependencies between tasks. However, linked Gantt charts quickly become cluttered in all but the simplest cases. Critical Path Network Diagrams are superior to visually communicate the relationships between tasks. However, Gantt charts are often preferred over network diagrams because Gantt charts are easily interpreted without training, whereas critical path diagrams require training to interpret. Gantt chart software typically provides mechanisms to link task dependencies, although this data may or may not be visually represented. Gantt charts and network diagrams are often used for the same project, both being generated from the same data by a software application.

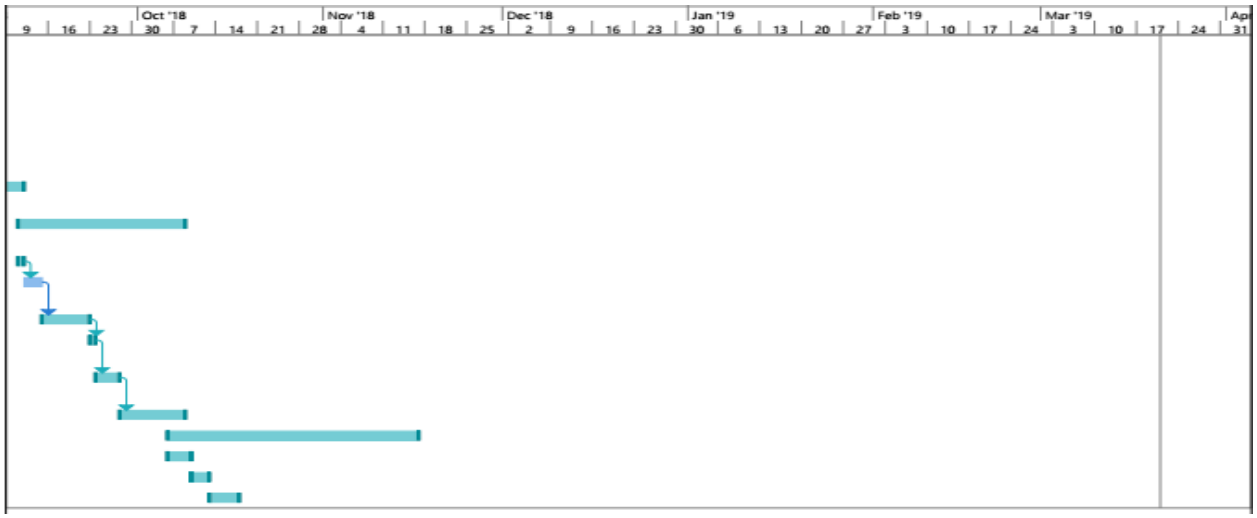
ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**CHAPTER 1: INTRODUCTION**	15 days	Wed 8/1/18	Sat 8/18/18	
2		1.1 Background	2 days	Wed 8/1/18	Thu 8/2/18	
3		1.2 Objectives	2 days	Fri 8/3/18	Sat 8/4/18	2
4		1.3 Purpose, Scope, and Applicability	3 days	Sun 8/5/18	Tue 8/7/18	3
5		1.4 Achievements	4 days	Wed 8/8/18	Sat 8/11/18	4
6		1.5 Organisation of Report	4 days	Sun 8/12/18	Wed 8/15/18	5
7		**CHAPTER 2: SURVEY OF TECHNOLOGIES**	7 days	Mon 9/3/18	Tue 9/11/18	
8		**CHAPTER 3: REQUIREMENTS AND ANALYSIS**	20 days	Tue 9/11/18	Mon 10/8/18	
9		3.1 Problem Definition	1 day	Tue 9/11/18	Tue 9/11/18	
10		3.2 Requirements Specification	3 days	Wed 9/12/18	Fri 9/14/18	9
11		3.3 Planning and Scheduling	7 days	Sat 9/15/18	Sat 9/22/18	10
12		3.4 Software and Hardware Requirements	1 day	Sun 9/23/18	Sun 9/23/18	11
13		3.5 Preliminary Product Description	4 days	Mon 9/24/18	Thu 9/27/18	12
14		3.6 Conceptual Models	7 days	Fri 9/28/18	Mon 10/8/18	13
15		**CHAPTER 4: SYSTEM DESIGN**	20 days			
16		**4.1 Sequence Diagram**				
17		**4.2 Use Case Diagram**				
18		**4.3 State Transition**				

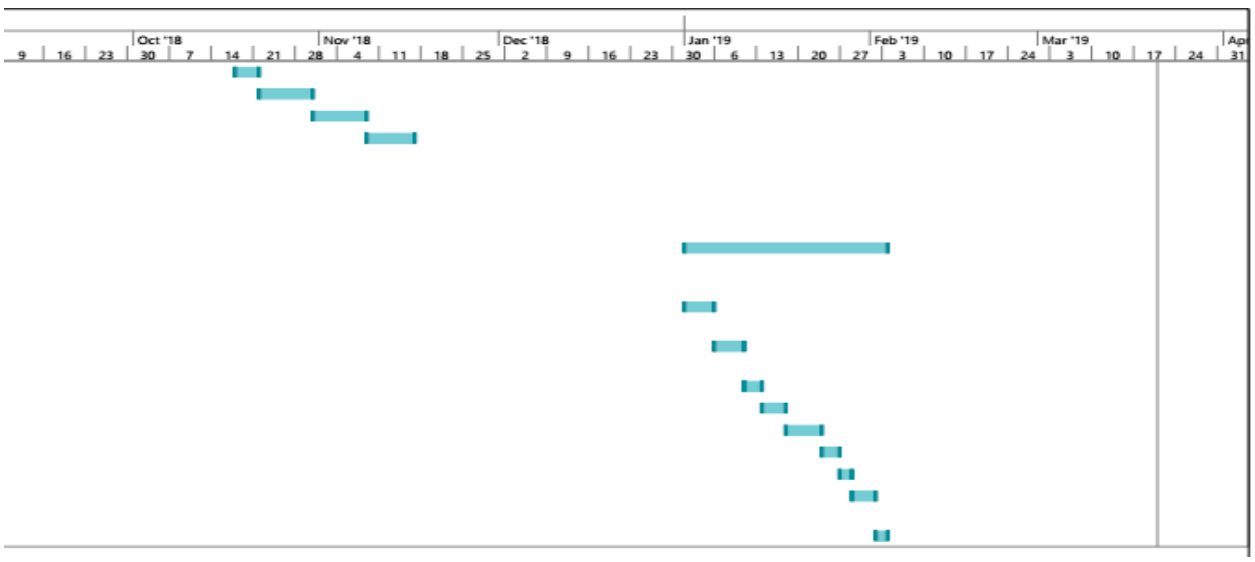


Project: Project1
Date: Mon 10/29/18

Task	Inactive Summary	External Tasks
Split	Manual Task	External Milestone
Milestone	Duration-only	Deadline
Summary	Manual Summary Rollup	Progress
Project Summary	Manual Summary	Manual Progress
Inactive Task	Start-only	
Inactive Milestone	Finish-only	

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
19	**4.4 Component Diagram**	3 days	Thu 10/18/18	Sun 10/21/18		
20	**4.5 Collaboration Diagram**	7 days	Mon 10/22/18	Tue 10/30/18		
21	**4.6 Activity Diagram**	7 days	Wed 10/31/18	Thu 11/8/18		
22	**4.7 DFD Diagram**	6 days	Fri 11/9/18	Fri 11/16/18		
23						
24						
25						
26						
27	**CHAPTER 5: IMPLEMENTATION AND TESTING**	25 days	Tue 1/1/19	Sun 2/3/19		
28	5.1 Implementation Approaches	5 days	Tue 1/1/19	Sat 1/5/19		
29	5.2 Coding Details and Code Efficiency	5 days	Sun 1/6/19	Thu 1/10/19		
30	5.2.1 Code Efficiency	2 days	Fri 1/11/19	Sun 1/13/19		
31	5.3 Testing Approach	4 days	Mon 1/14/19	Thu 1/17/19		
32	5.3.1 Unit Testing	4 days	Fri 1/18/19	Wed 1/23/19		
33	5.3.2 Integrated Testing	3 days	Thu 1/24/19	Sat 1/26/19		
34	5.3.3 Beta Testing	2 days	Sun 1/27/19	Mon 1/28/19		
35	5.4 Modifications and Improvements	4 days	Tue 1/29/19	Fri 2/1/19		
36	5.5 Test Cases	2 days	Sat 2/2/19	Sun 2/3/19		





ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
37	**CHAPTER 6: RESULTS AND DISCUSSION**	7 days	Fri 2/15/19	Mon 2/25/19		
38	6.1 Test Reports	5 days	Fri 2/15/19	Thu 2/21/19		
39	6.2 User Documentation	2 days	Fri 2/22/19	Mon 2/25/19		
40	**CHAPTER 7: CONCLUSIONS**	14 days	Fri 3/1/19	Wed 3/20/19		
41	7.1 Conclusion	3 days	Fri 3/1/19	Tue 3/5/19		
42	7.1.1 Significance of the System	4 days	Wed 3/6/19	Mon 3/11/19		
43	7.2 Limitations of the System	4 days	Tue 3/12/19	Fri 3/15/19		
44	7.3 Future Scope of the Project	4 days	Sat 3/16/19	Wed 3/20/19		

Fig.3.2.1 Gantt Chart

PERT Chart

A program evaluation review technique chart, known as a PERT chart, is a graphical illustration or representation of a project's schedule, which shows the sequence of tasks to be performed. PERT charts aid in determining the critical path of tasks and help in completing the project within a given time frame.

PERT charts were first developed by U.S. Navy in 1950 to support very large and complex projects during the cold war era. To complete a given project within a specified time, the factors to be evaluated include the shortest time the project can be completed and deciding on the activities to be completed first, allowing the project to be finished in the shortest time. These activities are represented in the PERT chart as a critical path.

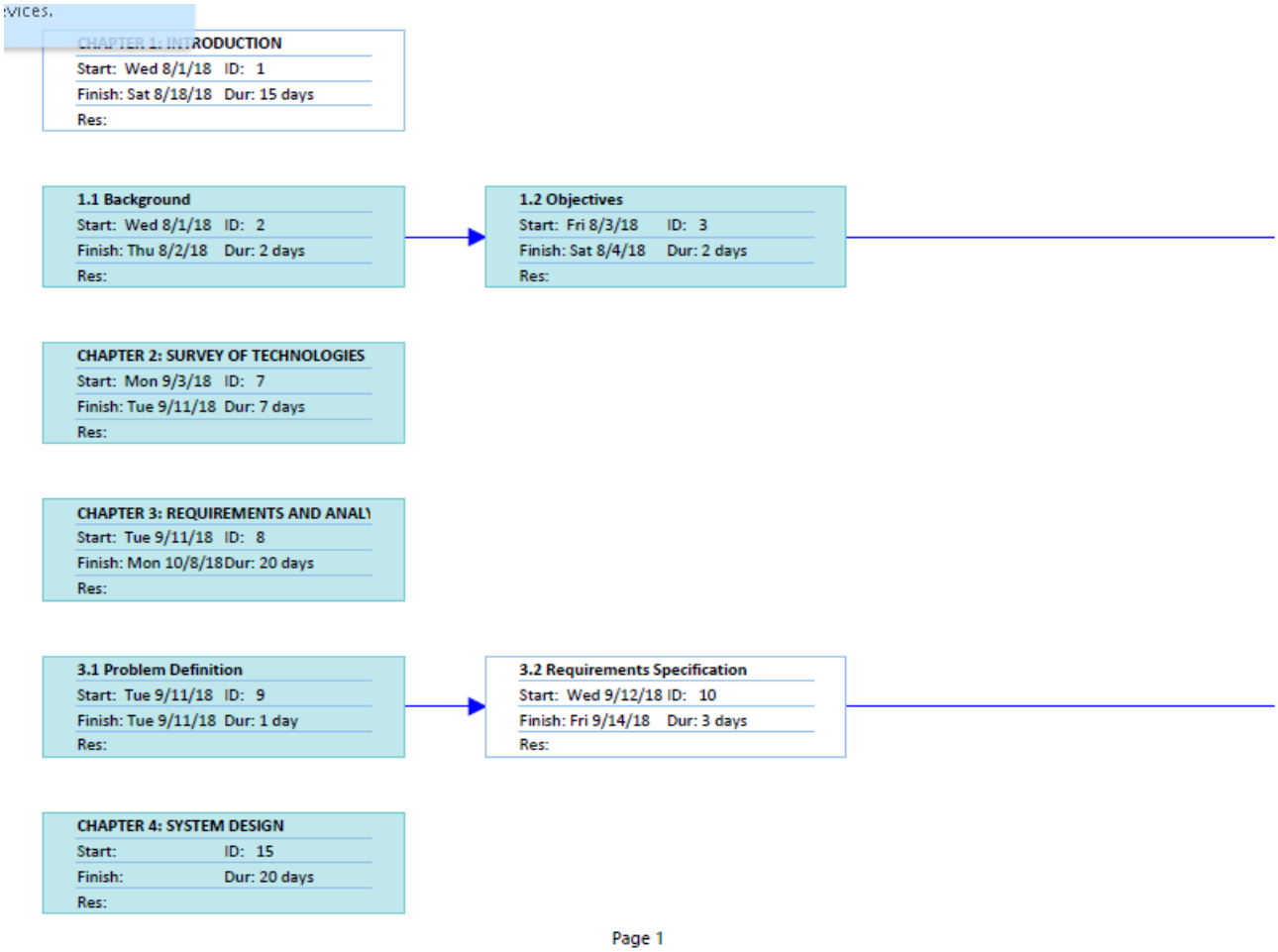


Fig.3.2.2 PERT Chart

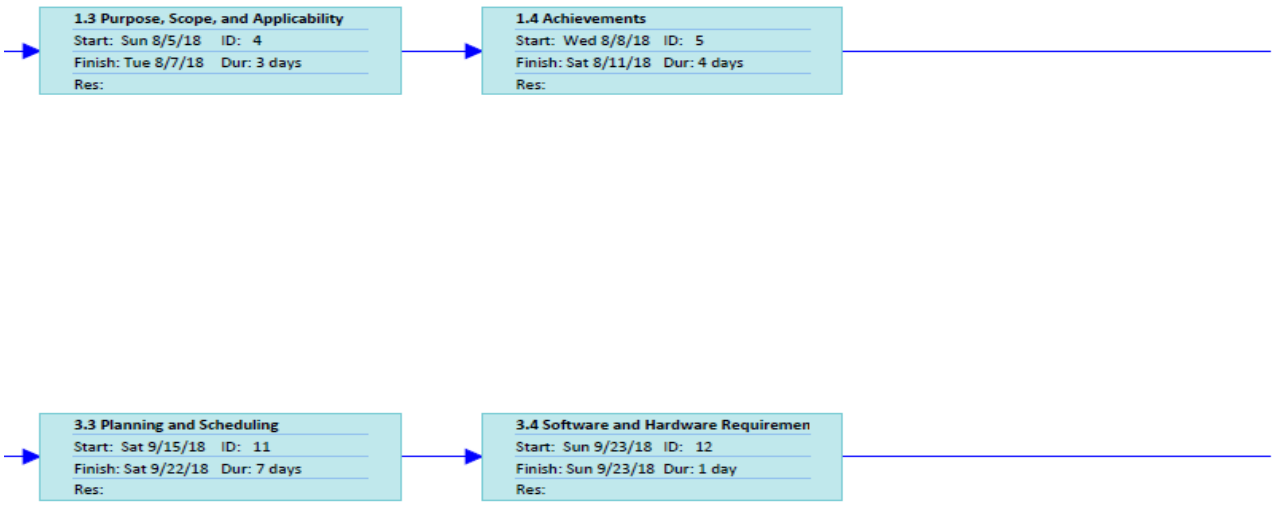


Fig.3.2.2 PERT Chart

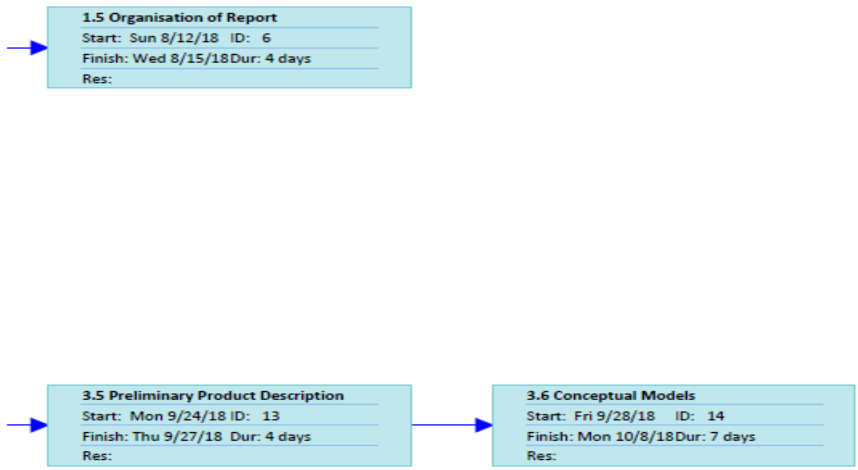


Fig.3.2.2 PERT Chart

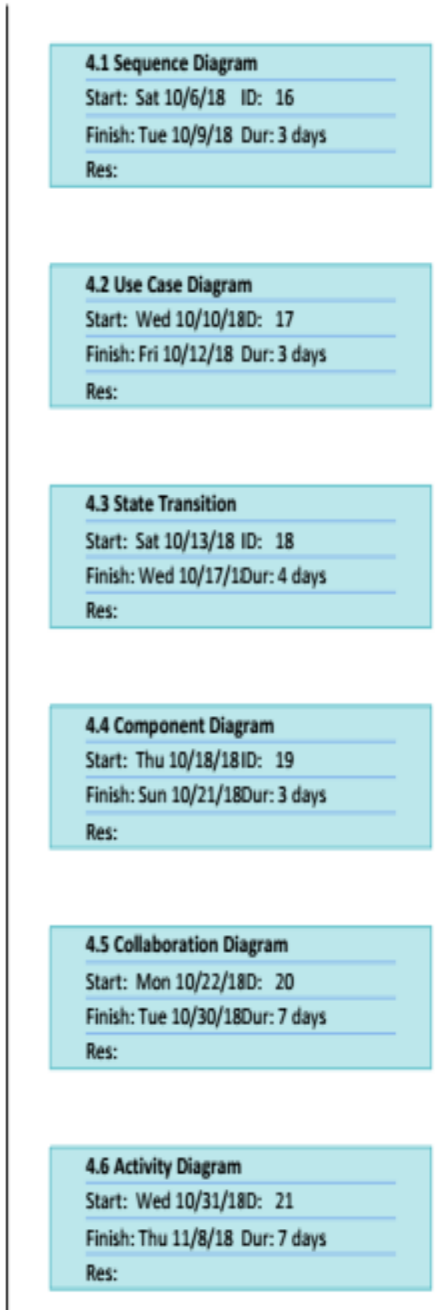


Fig.3.2.2 PERT Chart

Fig.3.2.2 PERT Chart

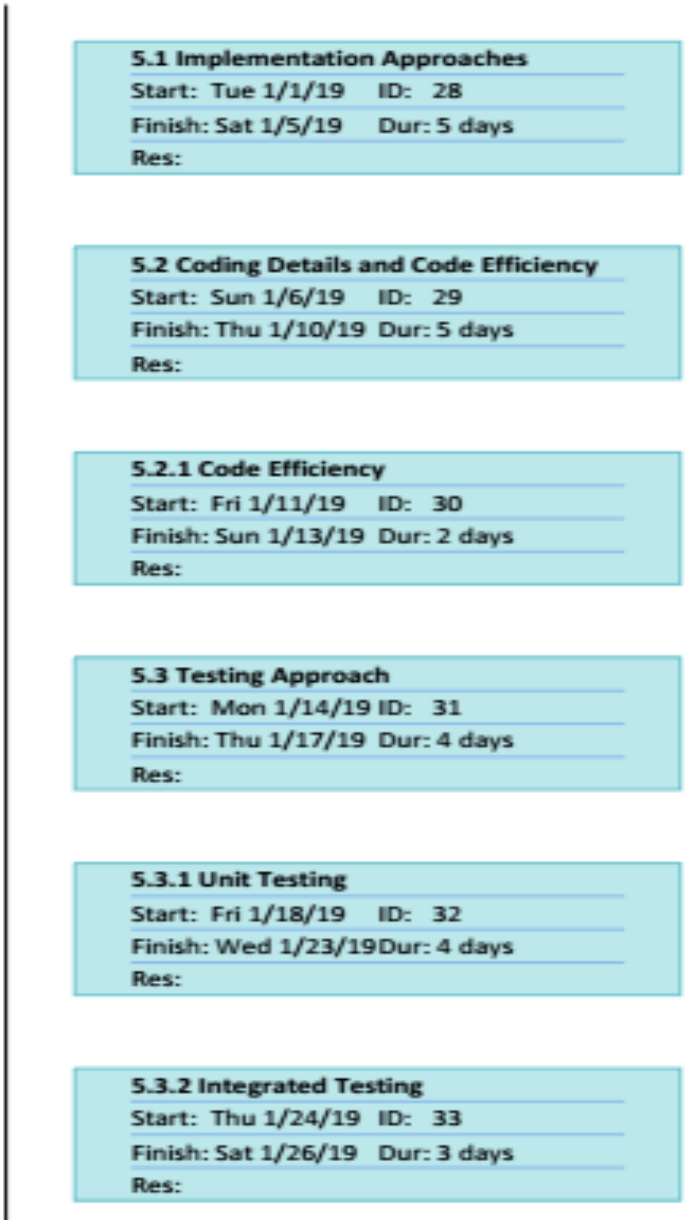


Fig.3.2.2 PERT Chart

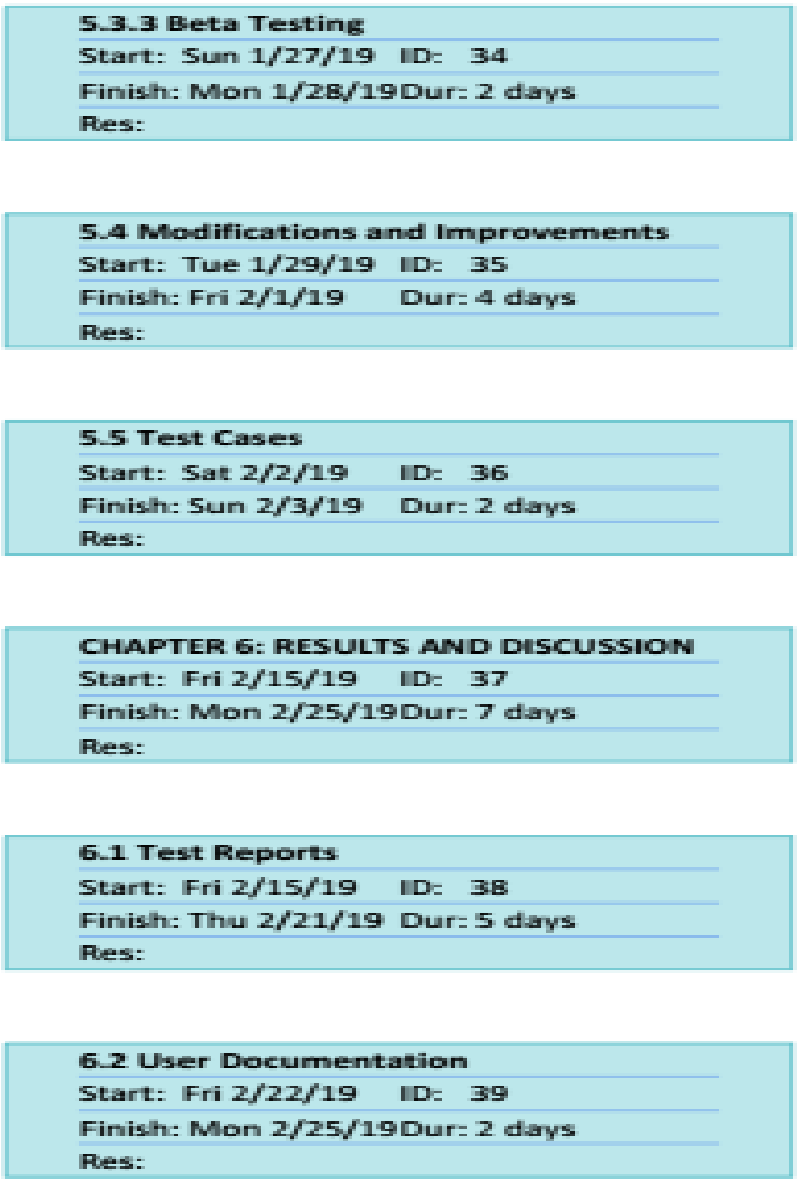
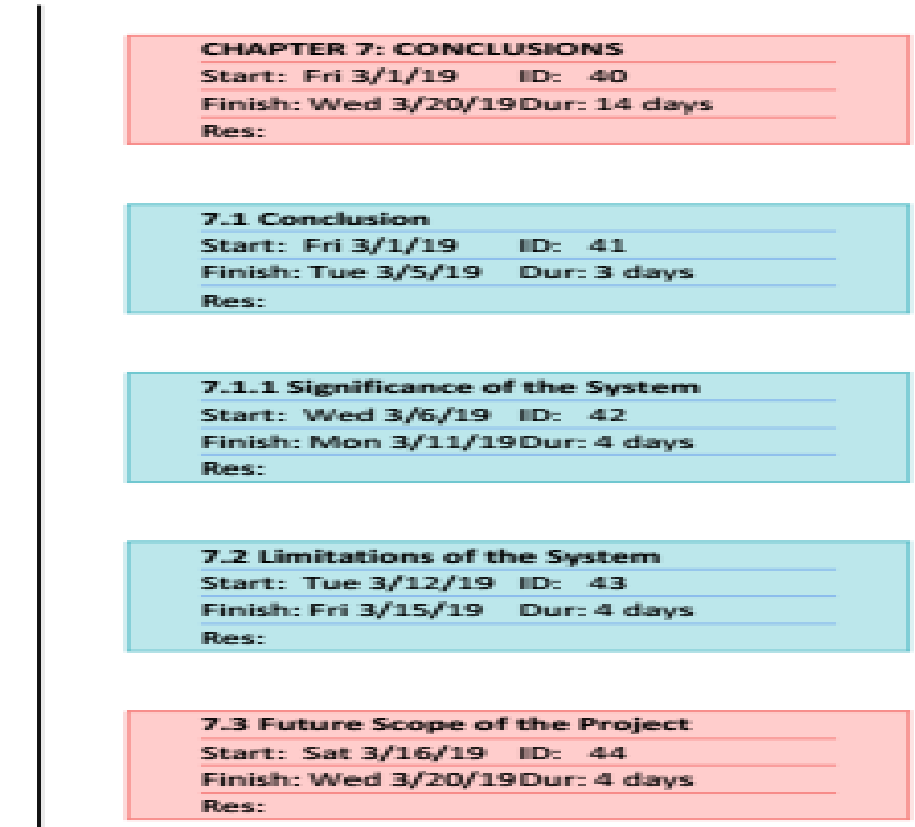


Fig.3.2.2 PERT Chart



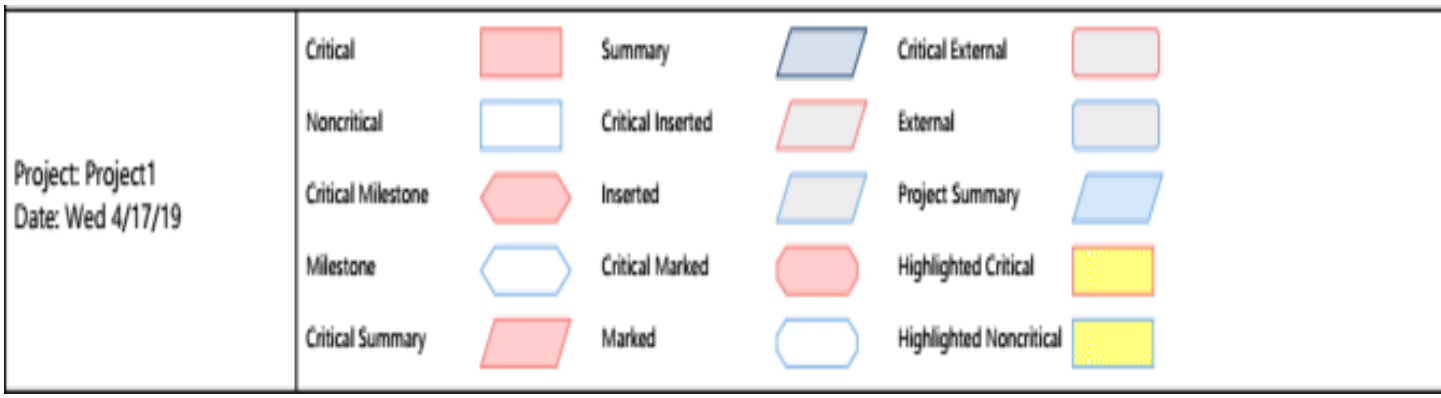


Fig.3.2.2 PERT Chart

3.3 DataFlow Diagrams / UML

3.3.1 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

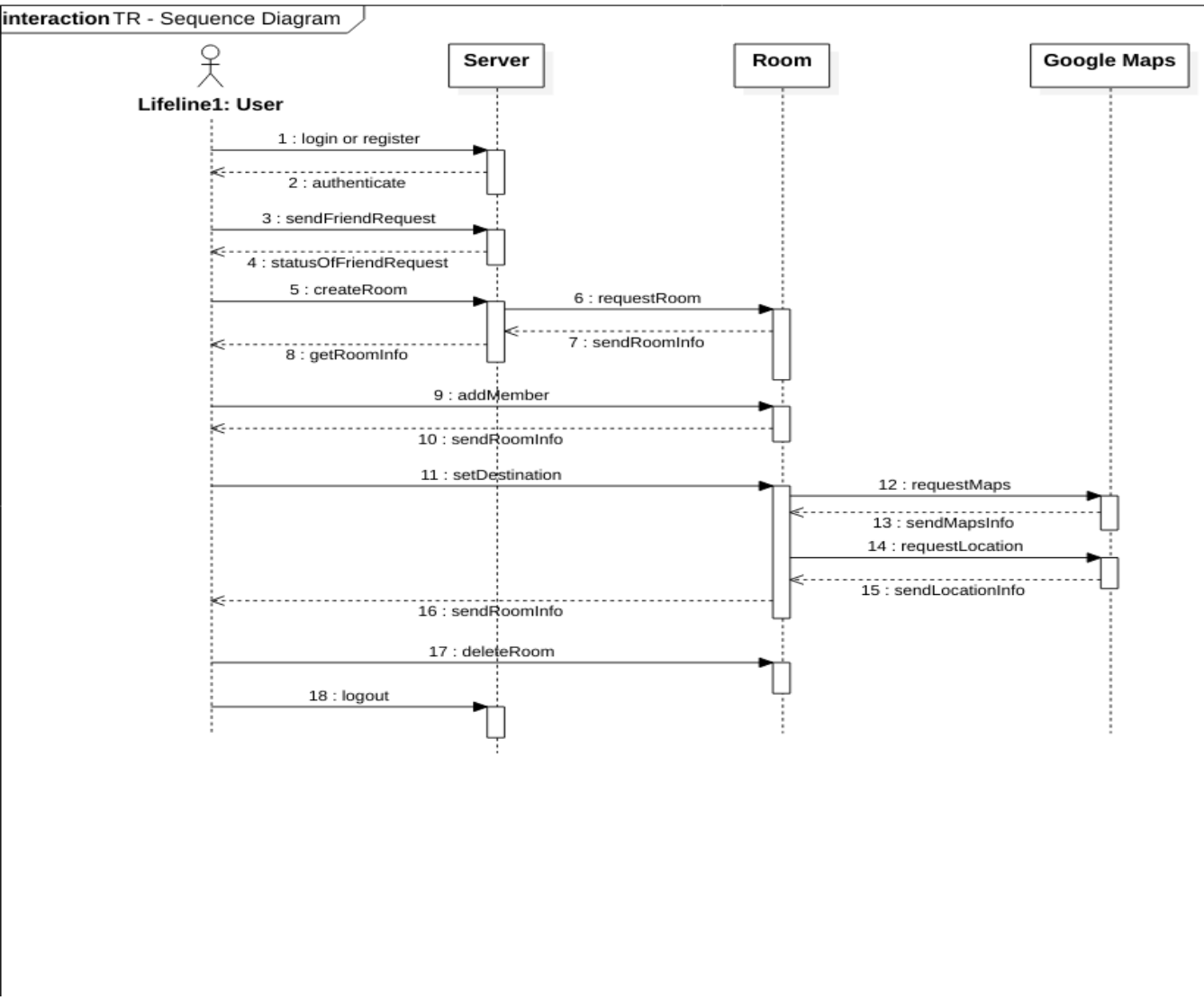


Fig. 3.3.1 Sequence Diagram

3.3.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed. Siau and Lee conducted research to determine if there was a valid situation for use case diagrams at all or if they were unnecessary. What was found was that the use case diagrams conveyed the intent of the system in a more simplified manner to stakeholders and that they were "interpreted more completely than class diagrams".

The purpose of the use case diagrams is simply to provide the high level view of the system and convey the requirements in laypeople's terms for the stakeholders. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.

Fig. 3.3.2 Use Case Diagram

3.3.3 State Transition Diagram

A State Transition Diagram also known as State Diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states, sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

State diagrams are used to give an abstract description of the behaviour of a system. This behavior is analyzed and represented as a series of events that can occur in one or more possible states. Hereby "each diagram usually represents objects of a single class and track the different states of its objects through the system".

State diagrams can be used to graphically represent finite state machines.

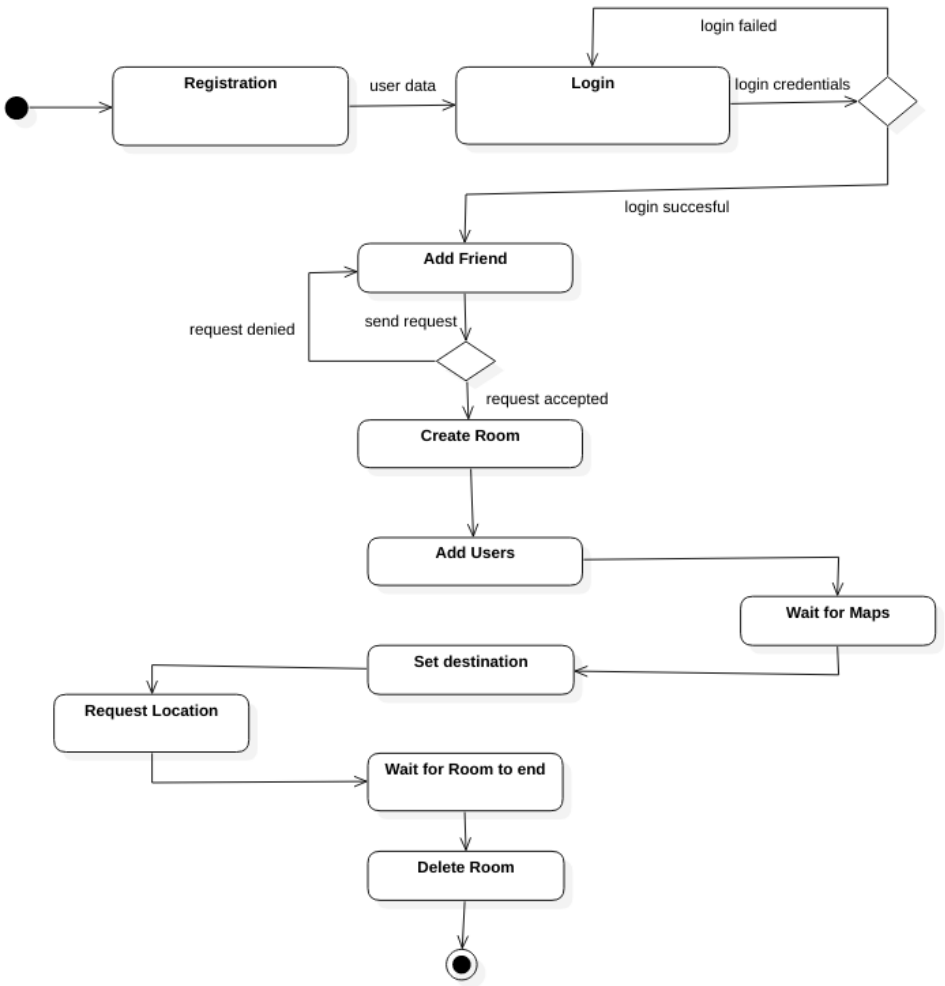


Fig. 3.3.3 State Transition Diagram

3.3.4 Component Diagram

In Unified Modeling Language (UML), a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

A component is something required to execute a stereotype function. Examples of stereotypes in components include executables, documents, database tables, files, and library files.

Components are wired together by using an *assembly connector* to connect the required interface of one component with the provided interface of another component. This illustrates the *service consumer - service provider* relationship between the two components.

An *assembly connector* is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port.”

A *delegation connector* is a "connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component’s parts."

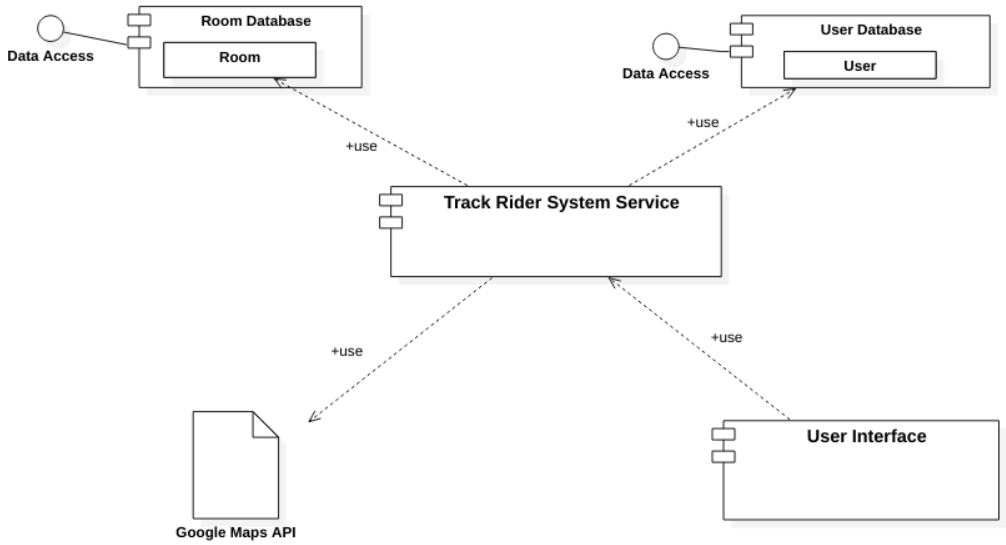


Fig. 3.3.4 Component Diagram

3.3.5 Collaboration Diagram

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagrams.

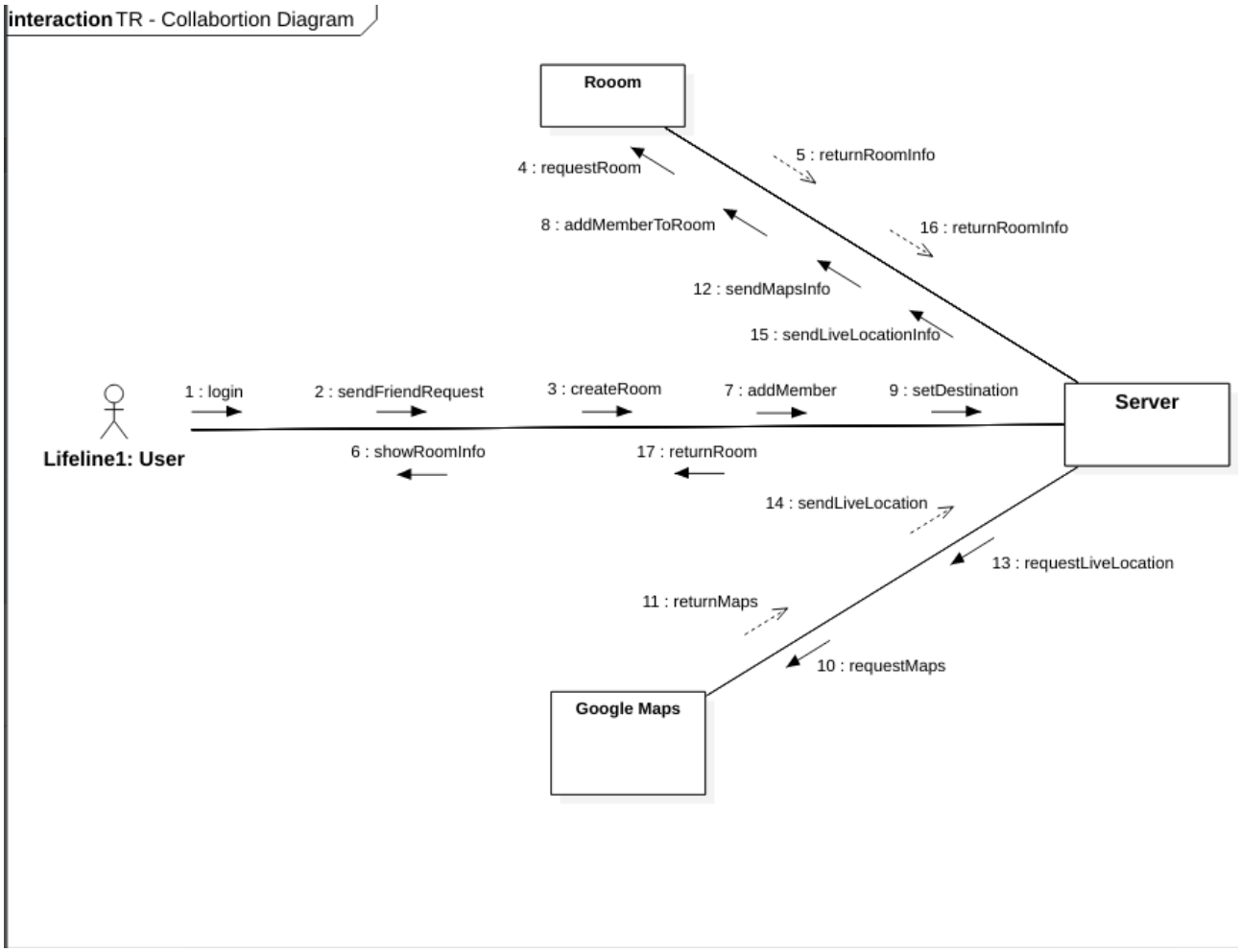


Fig. 3.3.5 Colloboration Diagram

3.3.6 Activity Diagram

Activity diagram is used to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The basic purposes of activity diagrams is to capture the dynamic behavior of the system. Activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

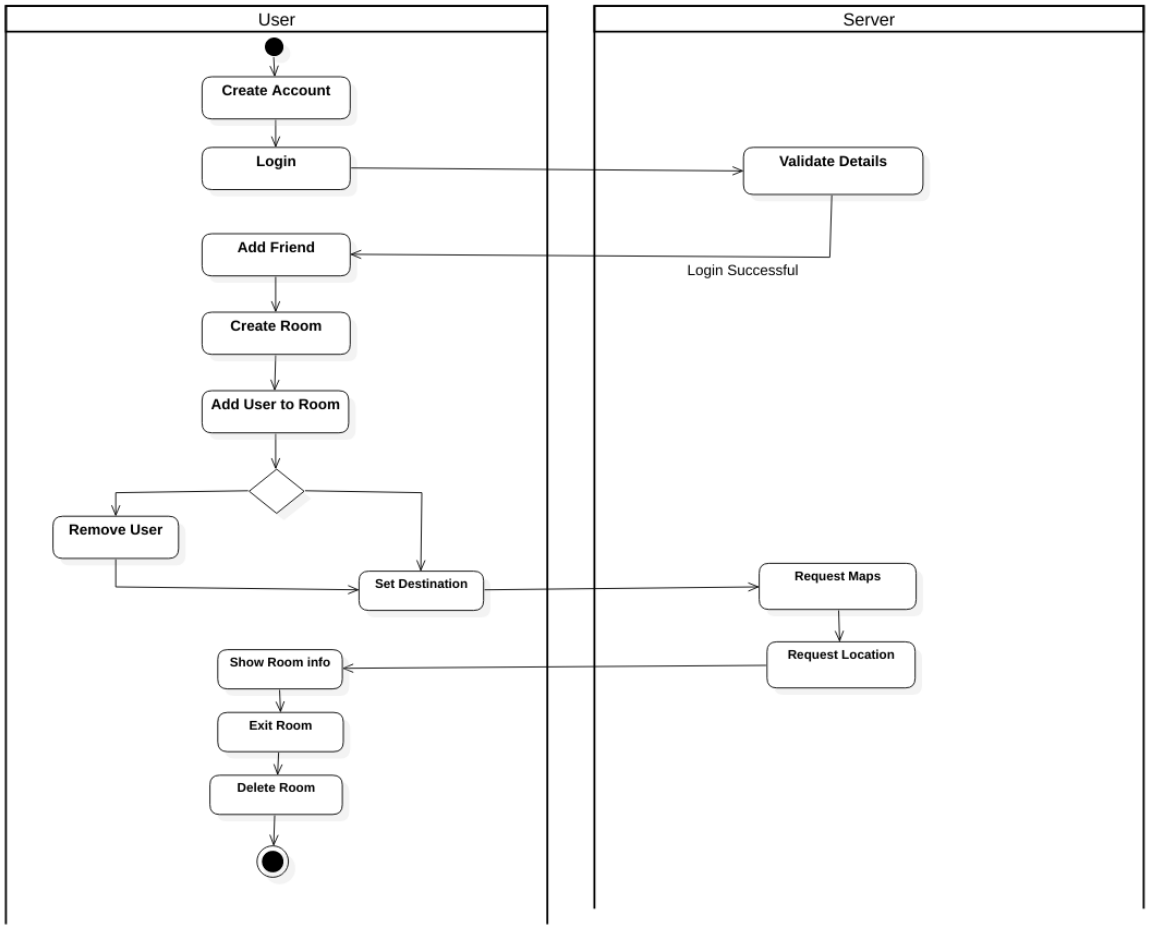


Fig. 3.3.6 Activity Diagram

3.3.7 ER Diagram

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

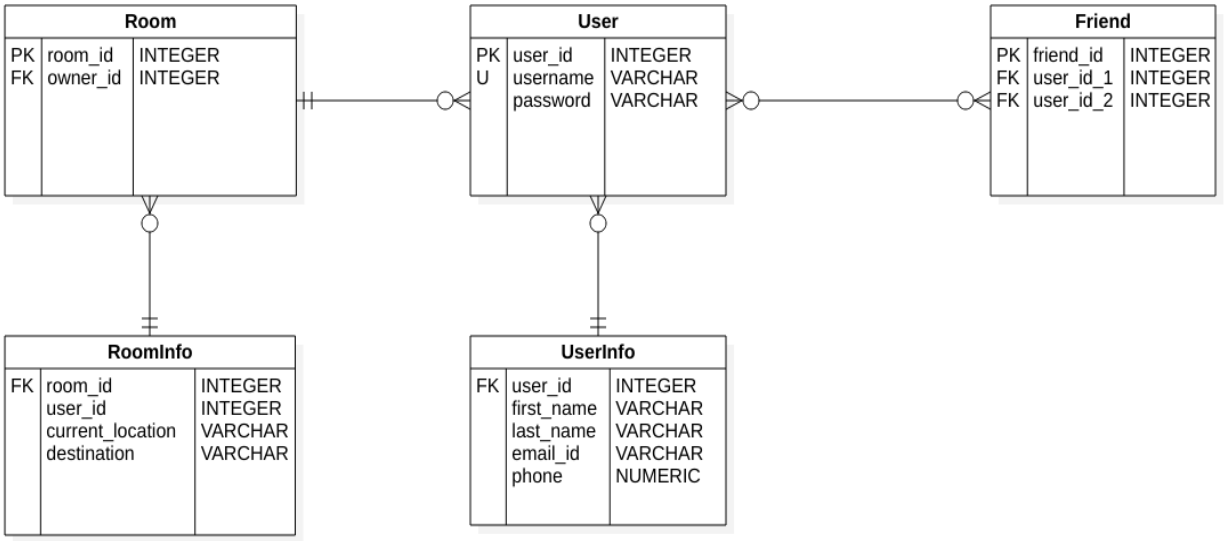
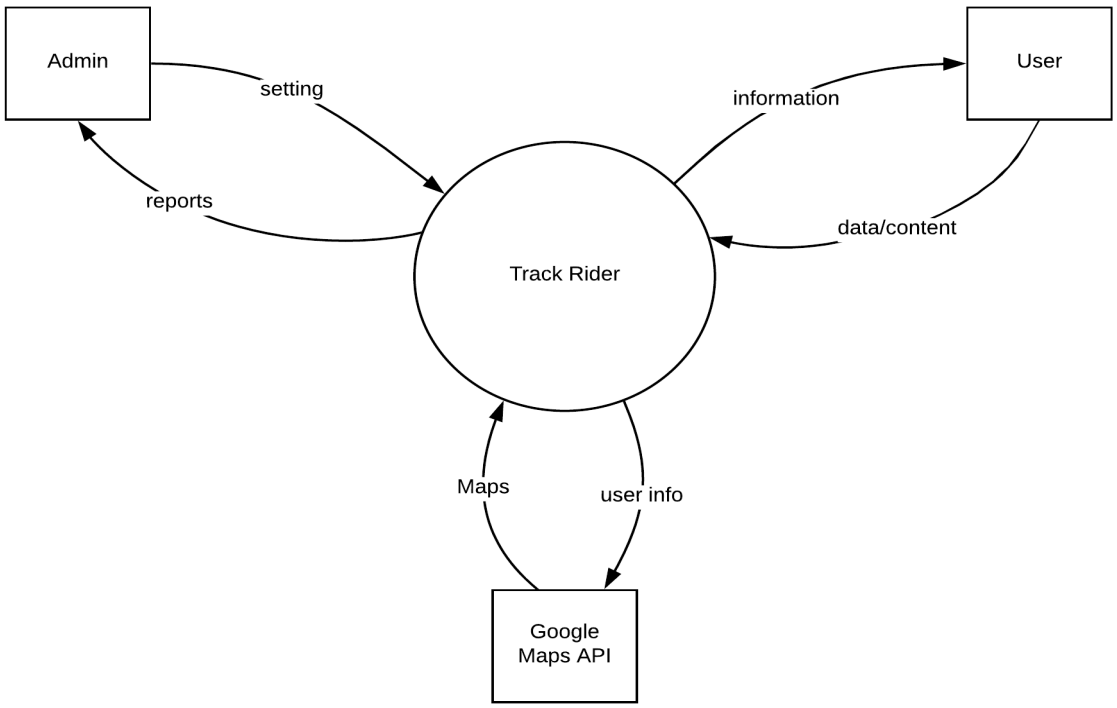


Fig. 3.3.7 ER Diagram

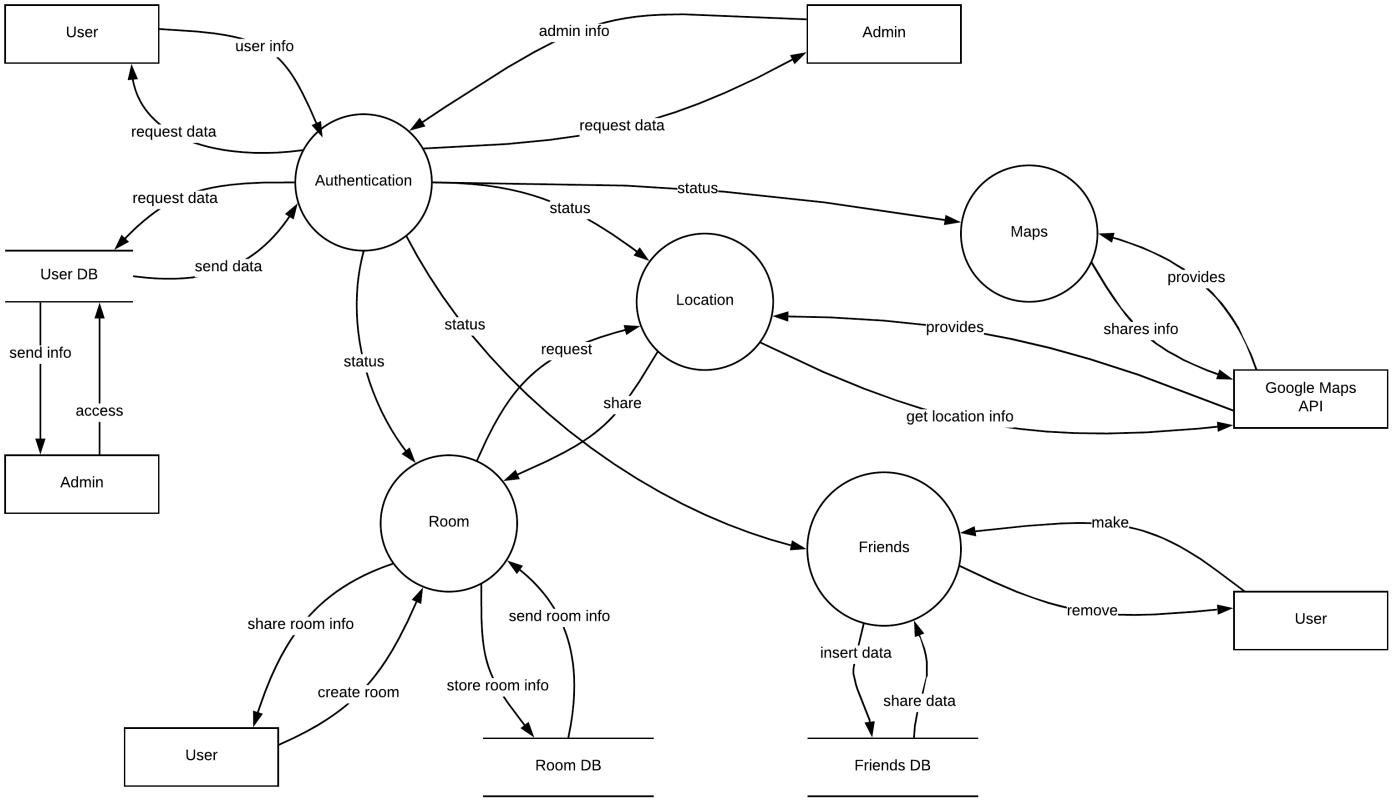
3.3.8 DFD Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.



Level 0

Fig. 3.3.8 DFD Diagram (Level 0)



Level 1

Fig. 3.3.8 DFD Diagram (Level 1)

4. IMPLEMENTATION & TESTING

4.1 Code

Once we start working on software development, Coding or Development is the third phase of SDLC. This phase is the third step of software development life cycle and comes after Requirement Analysis and Designing.

In designing phase we have already taken all the major decision regarding the system, now it’s time to develop the physical system. We will consider designing phase output as input for coding phase. The basic role of this phase is to convert designing into code using the programming language decided in designing phase. The well-developed code in this phase can help to reduce the efforts required in testing and maintenance. But even we make any silly mistake; it may lead us to put extra efforts in testing and maintenance.

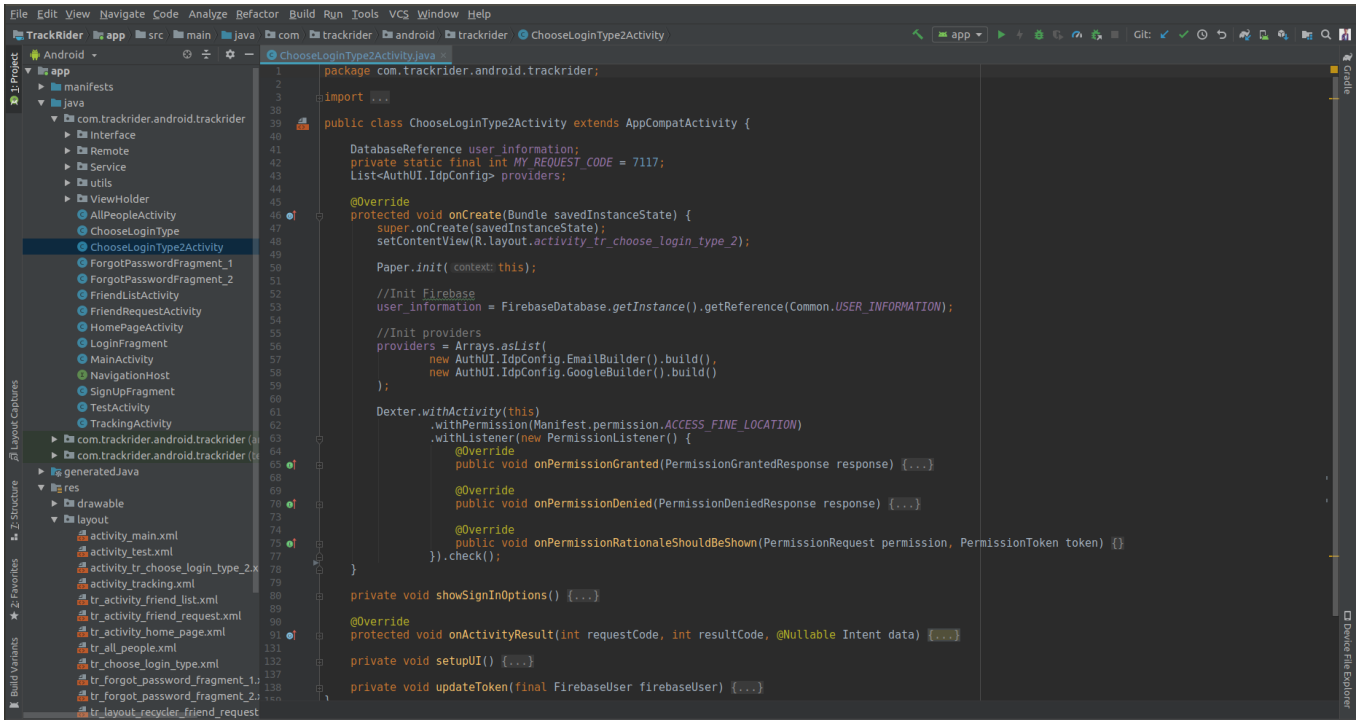
If we see it in a business perspective, the cost for testing efforts and maintenance is much higher than coding. So it always makes sense to spend time on coding phase. Here all developers write their own code and merged with other developer’s code to make sure that all modules developed by different developers interact with each other as per expectations. This is one of the longest phases in software development life cycle.

Few points which we need to take care in this phase :

- Version control application required in this phase.
- Before begin the actual coding, you should spend some time on selecting development tool, which will be suitable for your debugging, coding, modification and designing needs.
- Before actual writing code, some standard should be defined, as multiple developers going to use the same file for coding.
- During development developer should write appropriate comments so that other developers will come to know the logic behind the code.
- Last but most important point. There should be a regular review meeting need to conduct in this stage. It helps to identify the prospective defects in an early stage. Helps to improve product and coding quality.

Some of the Code snippets are as follows :

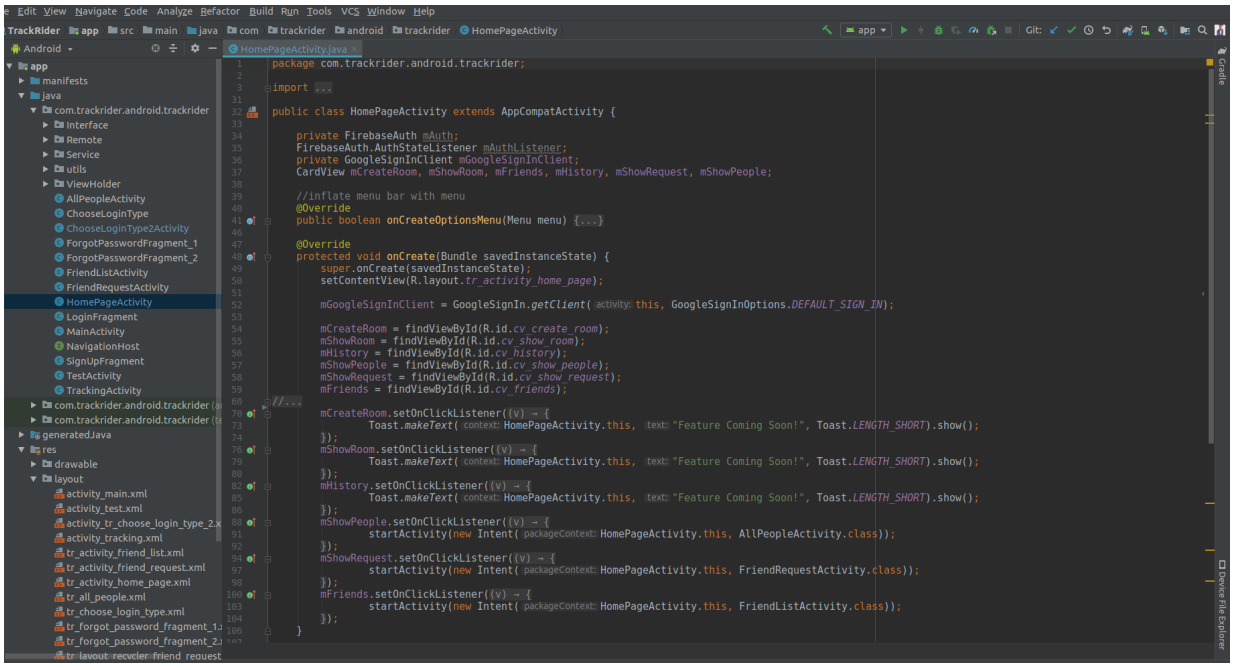
ChooseLoginType2Activity.java



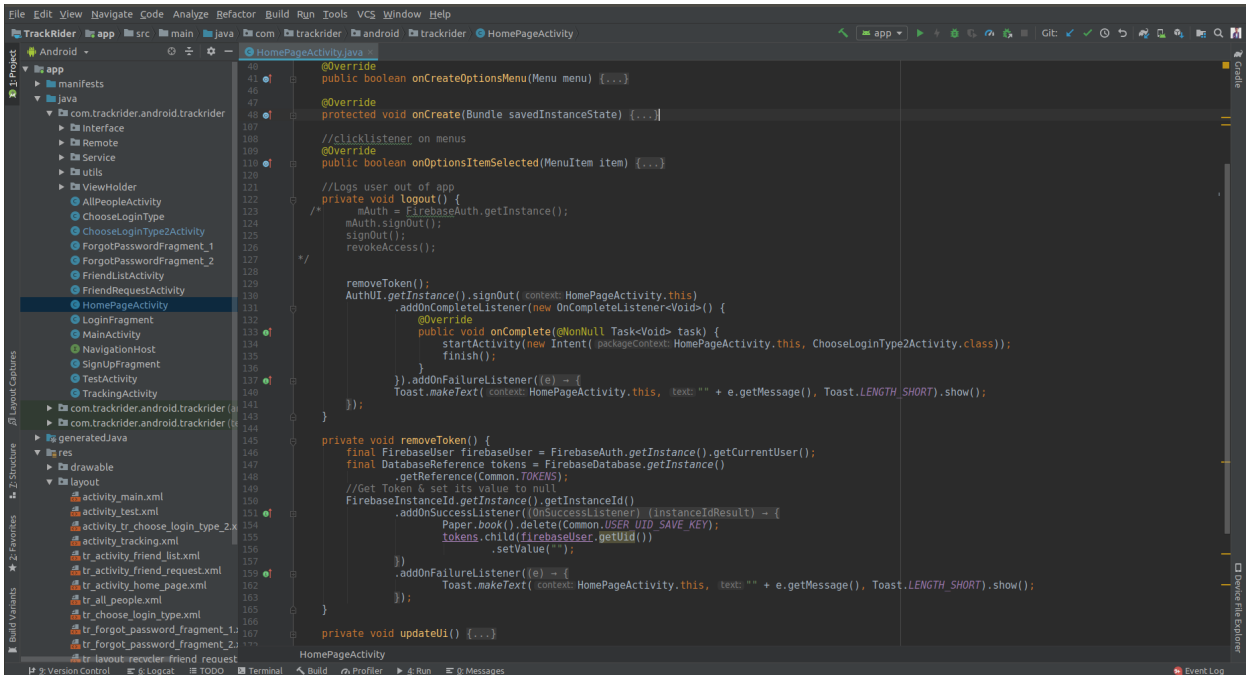
```
package com.trackrider.android.trackrider;

import ...

public class ChooseLoginType2Activity extends AppCompatActivity {

    DatabaseReference user_information;
    private static final int MY_REQUEST_CODE = 7117;
    List<AuthUI.IdpConfig> providers;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tr_choose_login_type_2);

        Paper.init( context: this);

        //Init Firebase
        user_information = FirebaseDatabase.getInstance().getReference(Common.USER_INFORMATION);

        //Init providers
        providers = Arrays.asList(
                new AuthUI.IdpConfig.EmailBuilder().build(),
                new AuthUI.IdpConfig.GoogleBuilder().build()
        );

        Dexter.withActivity(this)
                .withPermission(Manifest.permission.ACCESS_FINE_LOCATION)
                .withListener(new PermissionListener() {
                    @Override
                    public void onPermissionGranted(PermissionGrantedResponse response) {...}

                    @Override
                    public void onPermissionDenied(PermissionDeniedResponse response) {...}

                    @Override
                    public void onPermissionRationaleShouldBeShown(PermissionRequest permission, PermissionToken token) {}
                }).check();
    }

    private void showSignInOptions() {...}

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {...}

    private void setupUI() {...}

    private void updateToken(final FirebaseUser firebaseUser) {...}
```

This file loads the login providers and inflates the login activity. The onCreate() loads the layout file, initializes Firebase instance and asks for user permission to access GPS Location. When granted permission it shows sign in options to the user. After user logs in, the updateToken() updates the database with a firebase token which is unique to every user.

HomePageActivity.java

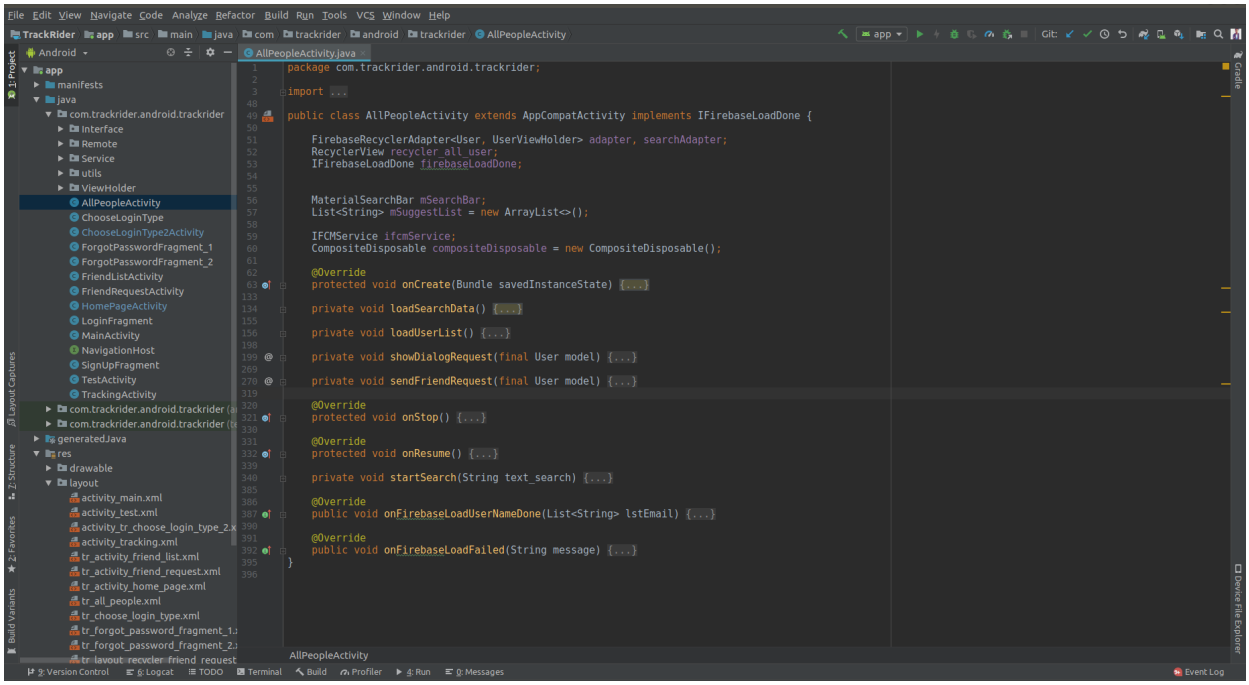


```
package com.trackrider.android.trackrider;

import ...

public class HomePageActivity extends AppCompatActivity {

    private FirebaseAuth mAuth;
    FirebaseAuth.AuthStateListener mAuthListener;
    private GoogleSignInClient mGoogleSignInClient;
    CardView mCreateRoom, mShowRoom, mFriends, mHistory, mShowRequest, mShowPeople;

    //inflate menu bar with menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

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

        mGoogleSignInClient = GoogleSignIn.getClient( activity: this, GoogleSignInOptions.DEFAULT_SIGN_IN);

        mCreateRoom = findViewById(R.id.cv_create_room);
        mShowRoom = findViewById(R.id.cv_show_room);
        mHistory = findViewById(R.id.cv_history);
        mShowPeople = findViewById(R.id.cv_show_people);
        mShowRequest = findViewById(R.id.cv_show_request);
        mFriends = findViewById(R.id.cv_friends);
//...
        mCreateRoom.setOnClickListener((v) -> {
                Toast.makeText( context: HomePageActivity.this, text: "Feature Coming Soon!", Toast.LENGTH_SHORT).show();
        });
        mShowRoom.setOnClickListener((v) -> {
                Toast.makeText( context: HomePageActivity.this, text: "Feature Coming Soon!", Toast.LENGTH_SHORT).show();
        });
        mHistory.setOnClickListener((v) -> {
                Toast.makeText( context: HomePageActivity.this, text: "Feature Coming Soon!", Toast.LENGTH_SHORT).show();
        });
        mShowPeople.setOnClickListener((v) -> {
                startActivity(new Intent( packageContext: HomePageActivity.this, AllPeopleActivity.class));
        });
        mShowRequest.setOnClickListener((v) -> {
                startActivity(new Intent( packageContext: HomePageActivity.this, FriendRequestActivity.class));
        });
        mFriends.setOnClickListener((v) -> {
                startActivity(new Intent( packageContext: HomePageActivity.this, FriendListActivity.class));
        });
    }
```



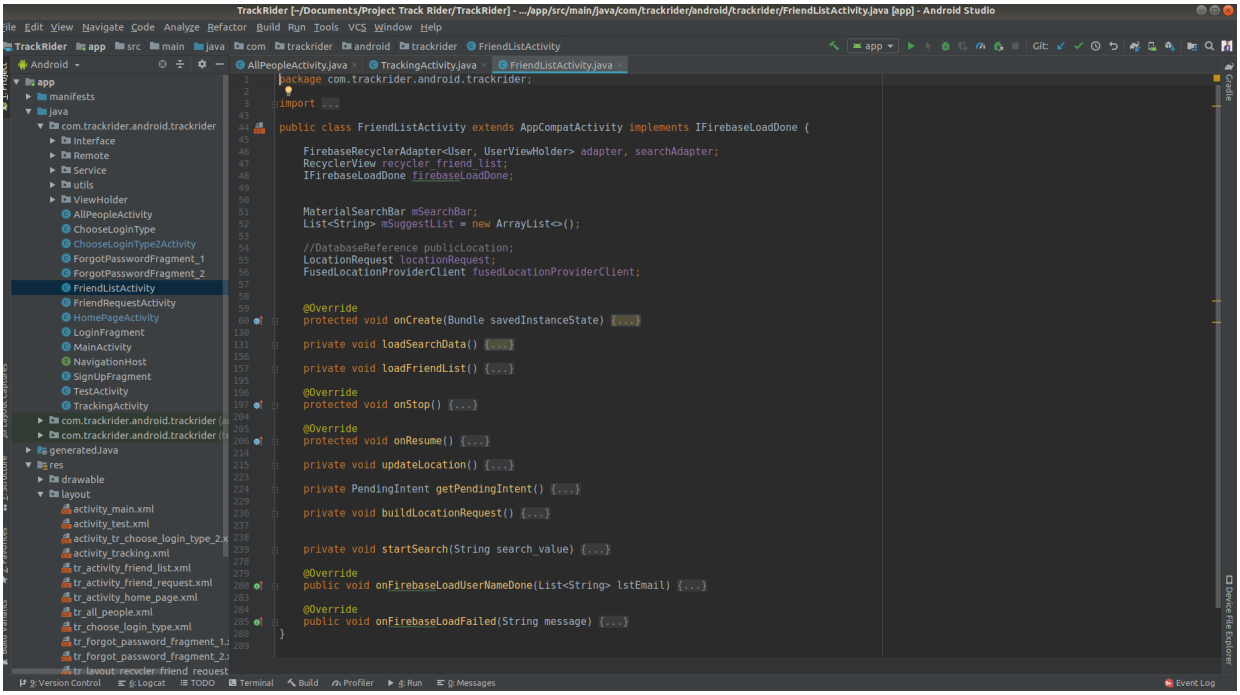
This file loads the Home page activity and provides functionality to go to any particular screen. This screen also provides the logout functionality.

AllPeopleActivity.java



This file loads & shows all of the people from the database using recycler view. This file also allows us to send friend request to a particular user.

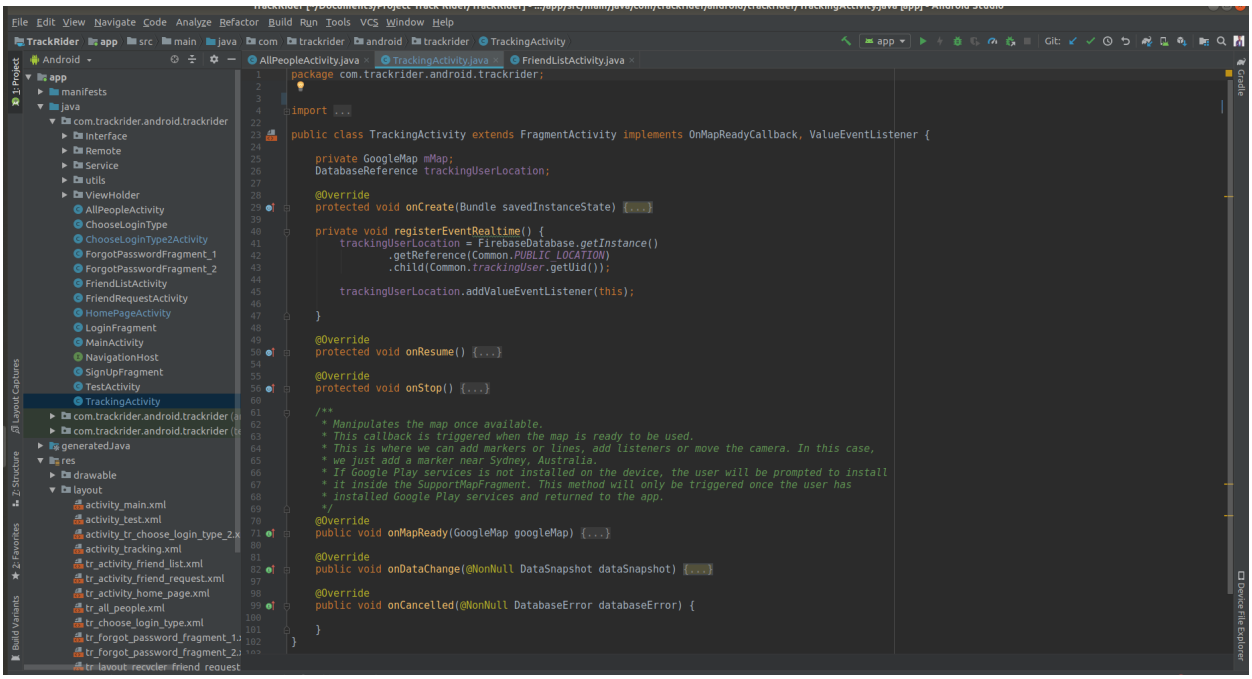
FriendListActivity.java



```
package com.trackrider.android.trackrider;

import ...

public class FriendListActivity extends AppCompatActivity implements IFirebaseLoadDone {

    FirebaseRecyclerAdapter<User, UserViewHolder> adapter, searchAdapter;
    RecyclerView recycler_friend_list;
    IFirebaseLoadDone firebaseLoadDone;


    MaterialSearchBar mSearchBar;
    List<String> mSuggestList = new ArrayList<>();

    //DatabaseReference publicLocation;
    LocationRequest locationRequest;
    FusedLocationProviderClient fusedLocationProviderClient;


    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    private void loadSearchData() {...}

    private void loadFriendList() {...}

    @Override
    protected void onStop() {...}

    @Override
    protected void onResume() {...}

    private void updateLocation() {...}

    private PendingIntent getPendingIntent() {...}

    private void buildLocationRequest() {...}


    private void startSearch(String search_value) {...}

    @Override
    public void onFirebaseLoadUserNameDone(List<String> lstEmail) {...}

    @Override
    public void onFirebaseLoadFailed(String message) {...}
}
```

This activity loads & shows all of the user's friend. It also creates an intent, on click upon a user's friend which start the maps activity to show that friend's real time location.

TrackingActivity.java



```
package com.trackrider.android.trackrider;

import ...

public class TrackingActivity extends FragmentActivity implements OnMapReadyCallback, ValueEventListener {

    private GoogleMap mMap;
    DatabaseReference trackingUserLocation;

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    private void registerEventRealtime() {
        trackingUserLocation = FirebaseDatabase.getInstance()
                .getReference(Common.PUBLIC_LOCATION)
                .child(Common.trackingUser.getUid());

        trackingUserLocation.addValueEventListener(this);

    }

    @Override
    protected void onResume() {...}

    @Override
    protected void onStop() {...}

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {...}

    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {...}

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
}
```

This activity loads up the friend's location and shows it to user.

4.2 Testing Approach

What is software testing?

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- Meets the requirements that guided its design and development
- Responds correctly to all kinds of inputs
- Performs its functions within an acceptable time
- It is sufficiently usable
- Can be installed and run in its intended environments, &
- Achieves the general result its stakeholders desire.

Software testing depending upon the testing method employed, can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test-driven approach and place an increased portion of the testing in the hands of the developer, before it reaches the formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

There are two basic types of testing: black box testing and white box testing.

Black Box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and executed of the system. It is also called functional testing.

White Box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is called structural testing and glass box testing.

Black Box testing is often used for validation & White Box testing is used often for verification.

Types of Testing

There are many types of testing like :

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

METHODOLOGIES ADOPTED FOR TESTING

Software Testing Methodology is defined as strategies and testing types used to certify that the Application Under Test meets client expectations. Test methodologies include functional and non-functional testing to validate the AUT. Examples of Testing Methodologies are Unit Testing, Integration Testing, System Testing, Performance Testing etc. Each testing methodology has a defined test objective, test strategy and deliverables.

4.2.1 Unit Testing

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc.

In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock / fake objects are used to assist in unit testing.

Test Scenario ID	Login-1	**Test Case ID**	Login-1A
Test Case Descrption	Login – Positive Test Case	**Test Priority**	High
Pre-Requisite	A valid User Account	**Post-Requisite**	NA

Sr. No.	Action	Inputs	Expected O/p	Actual O/p	Test Result
1	Launch Application	Select the app	Login Screen	Login Screen	PASS
2	(Email Signin) Enter correct Email & Password	Email id : abcd@gmail.com Password : 12345	Opens Home Activity	Opens Home Activity	PASS
3	(Google Signin)	Choose an existing account	Opens Home Activity	Opens Home Activity	PASS
4	(Google Signin)	Click on ‘ use another account ’ & add google account	Opens Home Activity	Opens Home Activity	PASS
5	(Google Signin)	Click on ‘ use another account ’ & create google account	Opens Home Activity	Opens Home Activity	PASS
6	(Email Signin)	Enter / Select Email of an existing google account	Shows 'You already have An account’	Shows 'You already have An account’	PASS

Test Scenario ID	Login-1	Test Case ID	Login-1B
Test Case Descrption	Login – Negative Test Case	Test Priority	High
Pre-Requisite	NA	Post-Requisite	NA

Sr. No.	Action	Inputs	Expected O/p	Actual O/p	Test Result
1	Launch Application	Select the app	Login Screen	Login Screen	PASS
2	(Email Sign in) Enter valid Email & Invalid Password	Email id : abcd@gmail.com Password : invalid	Incorrect Password	Incorrect Password	PASS
3	(Email Sign in) Enter invalid Email	Enter invalid / new Email id	Shows registration Screen	Shows registration Screen	PASS

Test Scenario ID	People	Test Case ID	People-1
Test Case Descrption	Send Request to New People	Test Priority	High
Pre-Requisite	User logged in	Post-Requisite	NA

Sr. No.	Action	Inputs	Expected O/p	Actual O/p	Test Result
1	Open People Activity	Click on People button	Shows all people in Present in the app	Shows all people in Present in the app	PASS
2	Send Request to New People (both user online)	Click on a Email id & send request	Request Sent!	Request Sent!	PASS
3	Send Request to same person again	Click on a Email id & send request	Request already Sent!	Request Sent!	FAIL
4	Send Request to A Friend	Click on a Friend's Name	You are already friends With John Doe	You are already friends With John Doe	PASS
5	Send Request to New People (Receiver offline)	Click on a Email id & send request	Request Sent!	Request Sent!	PASS
6	Send Request to Myself	Click on your mail id	User can't be Selected	User can't be Selected	PASS

Test Scenario ID	Notification	**Test Case ID**	Notification-1
Test Case Descrption	Receive Notificatiion	**Test Priority**	High
Pre-Requisite	User logged in	**Post-Requisite**	NA

Sr. No.	Action	Inputs	Expected O/p	Actual O/p	Test Result
1	Open People Activity	Click on People button	Shows all people in Present in the app	Shows all people in Present in the app	PASS
2	Send Request to New People (both user online)	Click on a Email id & send request	Request Sent! & Notification received On receiver end	Request Sent! & Notification received On receiver end	PASS
3	Send Request to same Person again	Click on a Email id & send request	Request already Sent!	Request already Sent!	PASS
4	Send Request to A Friend	Click on a Friend's Name	You are already friends With John Doe	You are already friends With John Doe	PASS
5	Send Request to New People (Receiver offline)	Click on a Email id & send request	Request Sent!	Request Sent! & Notification received On receiver end	PASS
6	Send Request to Myself	Click on your mail id	User can't be Selected	User can't be Selected	PASS

Test Scenario ID	Friend Request	**Test Case ID**	FriendRequest-1
Test Case Descrption	Check Friend Request	**Test Priority**	High
Pre-Requisite	A valid User Account	**Post-Requisite**	NA

Sr. No.	Action	Inputs	Expected O/p	Actual O/p	Test Result
1	Open Requests Activity	Click on Requests Button	Opens Requests Activity	Opens Requests Activity	PASS
2	Accept Request	Click on accept Request Button	User added in Friends	User added in Friends	PASS
3	Decline Request	Click on decline Request Button	Request Deleted	Request Deleted	PASS

4.2.2 Integration Testing

Integration Testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing. In Integration Testing, individual software modules are integrated logically and tested as a group.

A typical software project consists of multiple software modules, coded by different programmers.

Integration Testing focuses on checking data communication amongst these modules.

Hence, it is also termed as ‘I & T’(Integration & Testing), ‘String Testing’ and sometimes ‘Thread Testing’.

System Testing

System testing is the testing to ensure that by putting the software in different environment (e.g Operating system) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

System Testing is the testing of a complete and fully integrated software product. Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware system. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer based system.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating result within a specified time as in performance requirement. It falls under the class of black box testing.

Performance testing is a type of testing to ensure software applications will perform well under their expected workload.

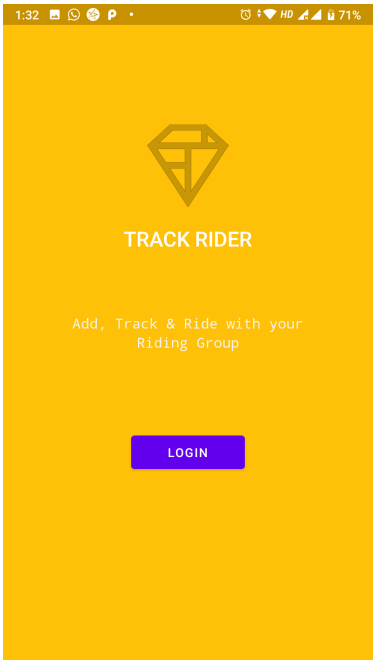
The focus of performance testing is checking a software program’s :

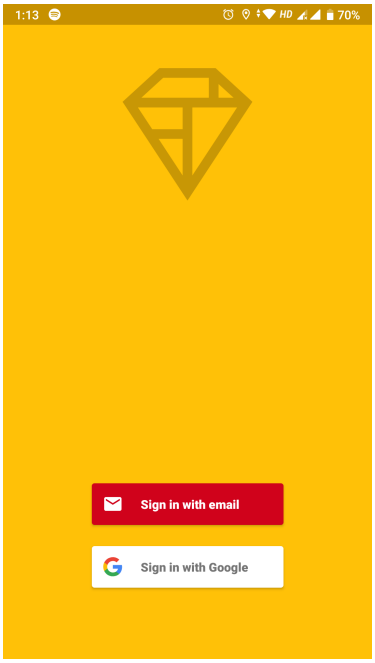
- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability- Determines if the application is stable under varying loads.

5. RESULTS & DISCUSSIONS

This is how our application is going to look.

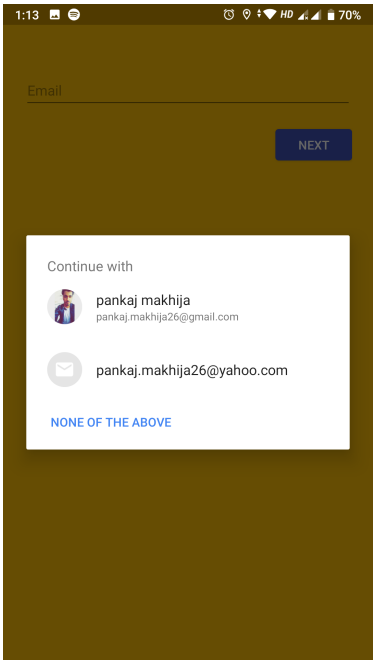
Login Module :

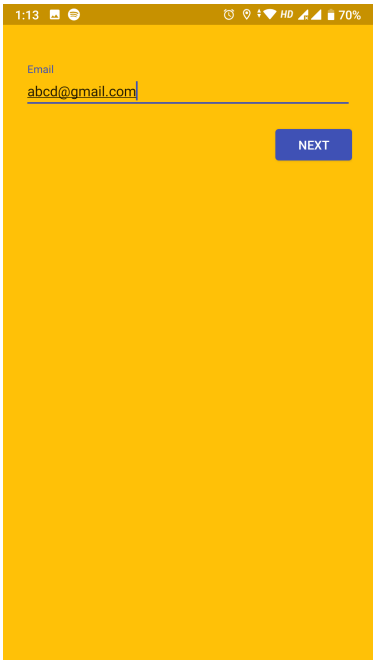




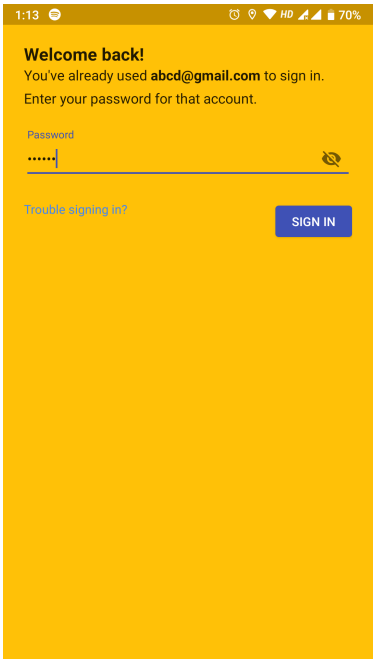
This is the Starting activity of the Application. Upon launching the application the activity (left) will open up. On click of the Login button the activity (right) will open up.

The activity on the right Shows two login options viz. By Email ID & by Google account

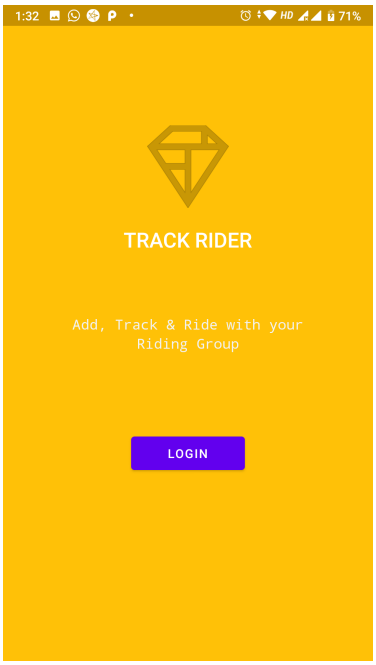




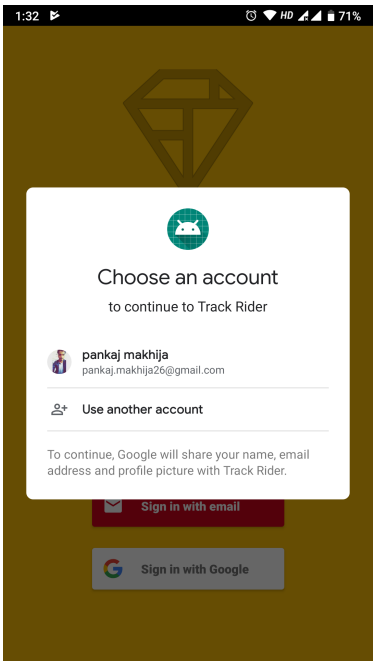
On selecting ‘Sign in with email’. The activity (right) will launch. It will first ask user whether they want to login via a registered email or a new email. When clicked on ‘none of the above’ the activity will look something like the screeshot (right).



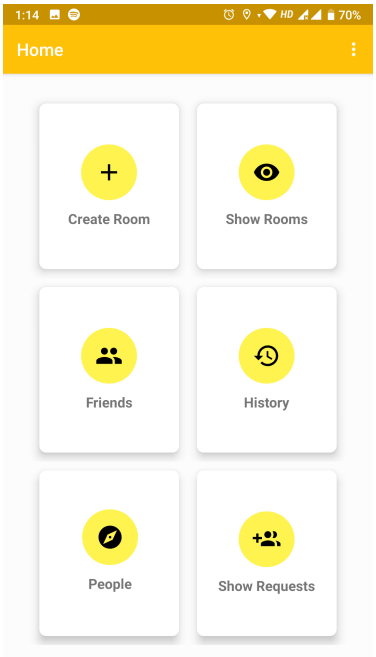
If the email provided is right or a registered email id is selected then it will ask for password.



If the email provided is not previously registered then the registration page will pop up.

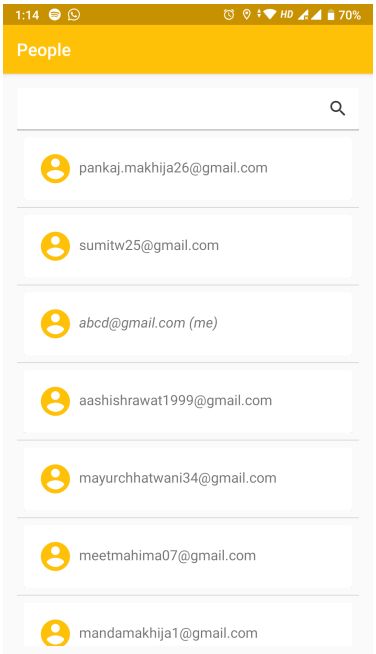


If ‘Sign in with Google’ is selected, it will show the registered google ids of the device. User can select one of the registered id or create a new google account to proceed.



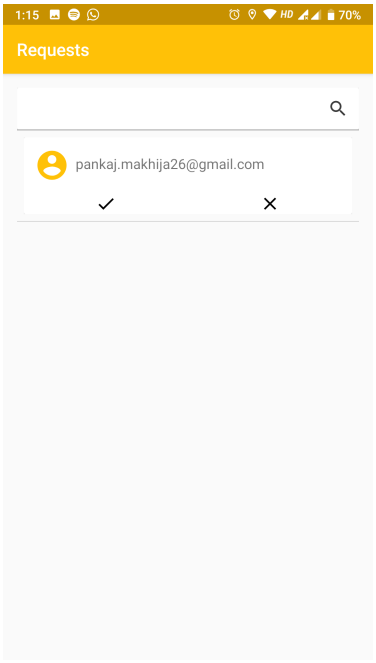
On successful login, the Home Actvity will load up. This activity is used to access different parts of the application.

People Module :



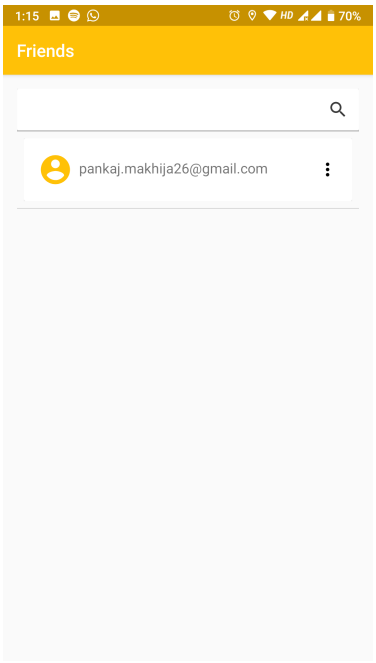
The People activity shows all the people registered in the Application. Users can send friend request to other users from this activity.

Friend Request Module :



This activity shows requests that

Friends Module :



This activity shows users friends. This activity can be used to see a friends location.

Location Module :

On clicking on a Friends name, you

6. CONCLUSION & FUTURE WORK

6.1 Conclusion

This thesis presents an Android Applicaion which can be used by riders to track their fellow riders exact location. This can help a riding group track all of their riders in real time. So if someone's vehicle breaks down in the middle of nowhere, other riders will know where to find them. This application can also be used in various other scenarios like if your friends/family are coming at your place & you just like to know how far away they are from your place.

6.2 Limitations

- This project is limited to Android Phones only.
- Areas with bad Network Connection can be a Problem.
- Mobile Data Pack should be activated on users end.

6.3 Future Work

In Future, we will make this application as light weight as possible. This application is being made in such a way that it provides end user the best User Experience (UX) possible & will continue to do so. This application will implement Material Design Standards more & in the best way possible. This application will also implement additional features that will ease Riders problems. Some of the features will be Distance between each rider, instant chat messaging, predefined messages, in app calling feature, etc.

7. REFERENCES

1. Google Developers website

   https://www.developers.google.com/

2. Android Developers Youtube Channel

   https://www.youtube.com/androiddevelopers

3. Google Design Youtube Channel

   https://www.youtube.com/googledesign

4. Firebase Youtube Channel

   https://www.youtube.com/firebase

5. EDMT Dev Youtube Channel

   https://www.youtube.com/eddydn71/

6. Stackoverflow Website

   https://www.stackoverflow.com/

7. Hitesh Choudhary Youtube Channel

   https://www.youtube.com/hiteshitube