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

1. **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.
2. **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. SURVEY OF TECHNOLOGIES

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

3. HTML 5 :

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.

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.

3. REQUIREMENTS & ANALYSIS

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.

3.2 Requirements Specification

3.2.1 Aim

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

3.2.2 Objective

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

3.2.3 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.

3.2.4 Purpose

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

3.2.5 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.

3.2.6 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.

3.3 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**				

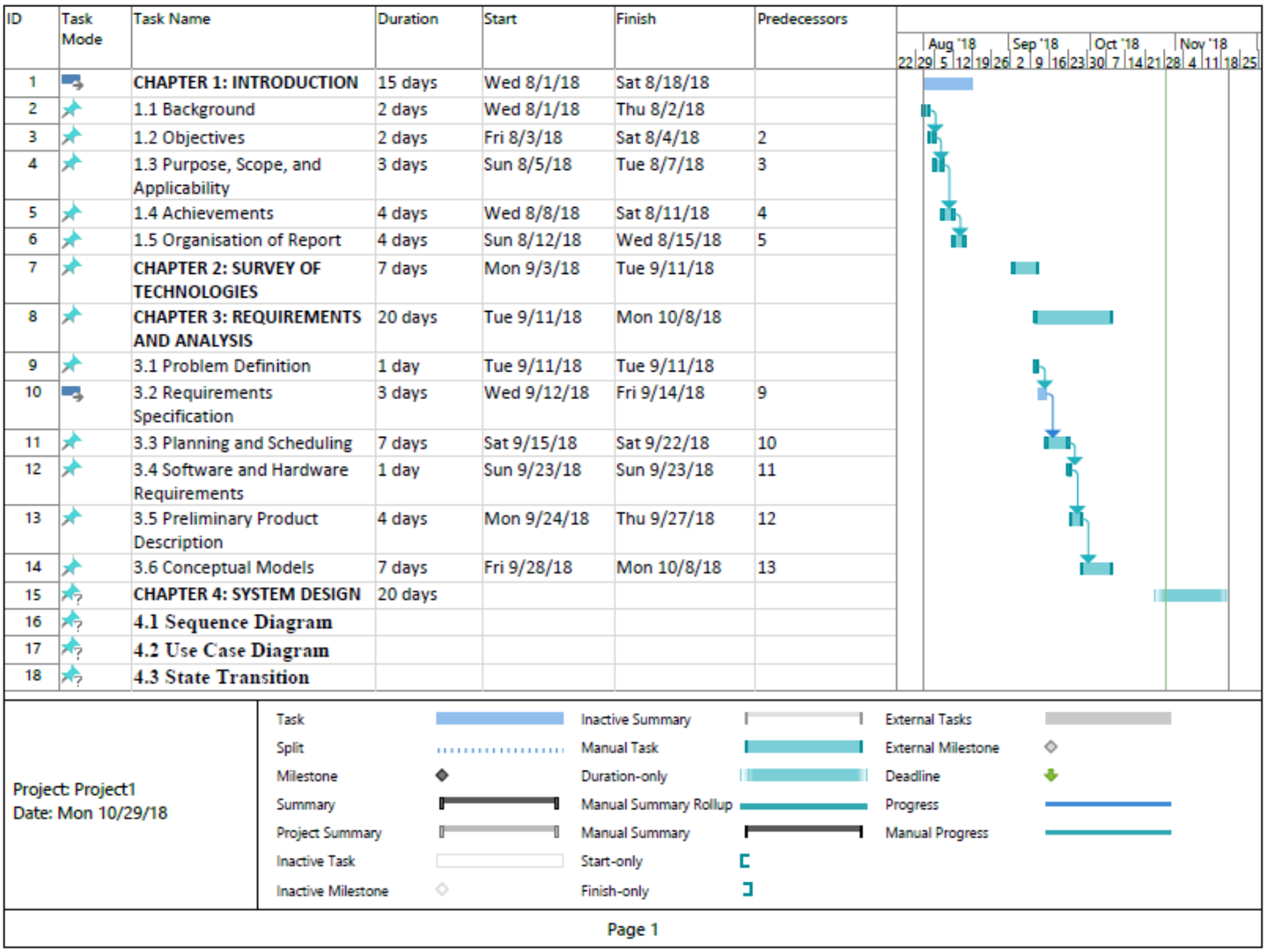


Fig.3.3.1 Gantt Chart

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.

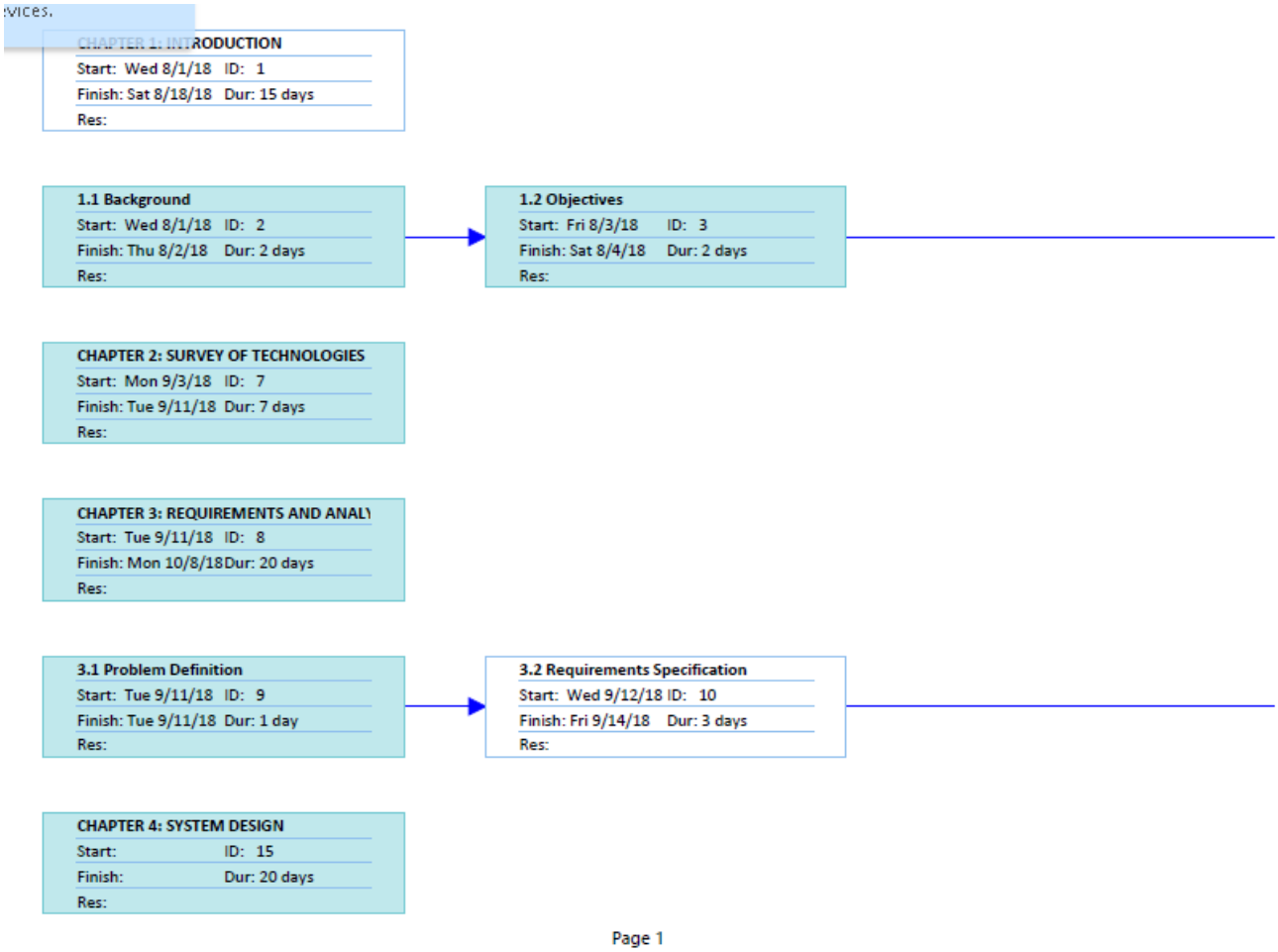


Fig.3.3.2 PERT Chart

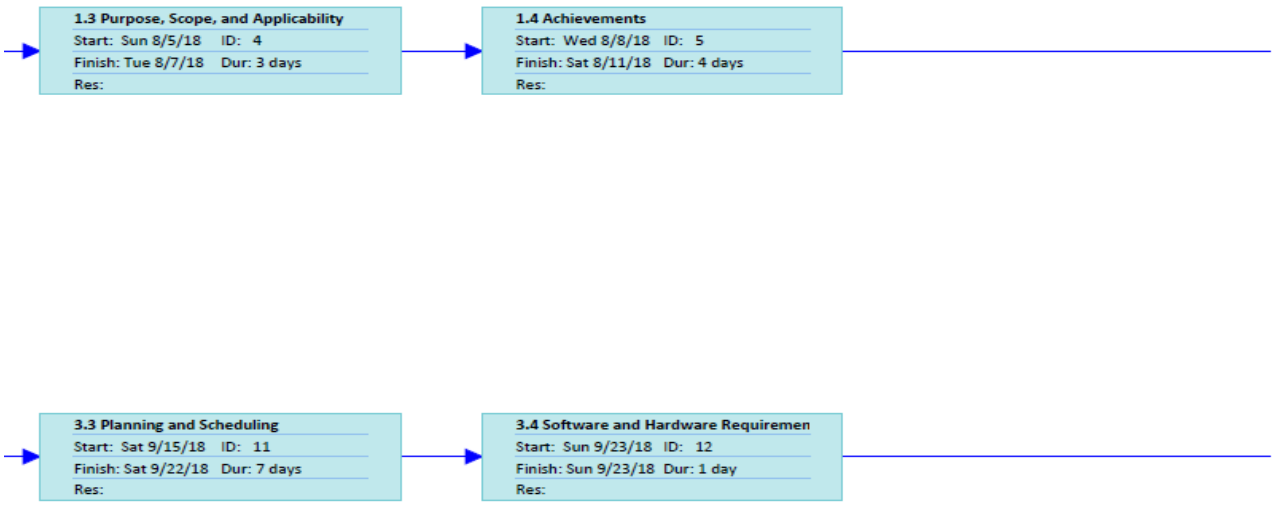


Fig.3.3.2 PERT Chart

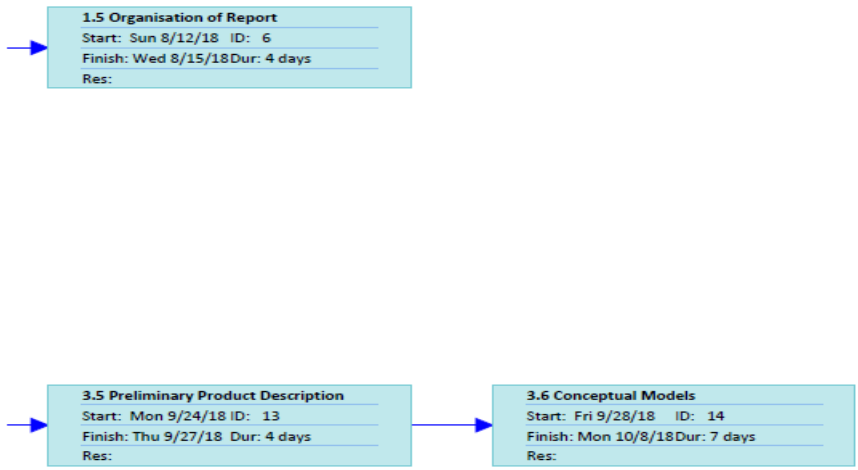


Fig.3.3.2 PERT Chart

3.4 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

4. System Design

4.1 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.

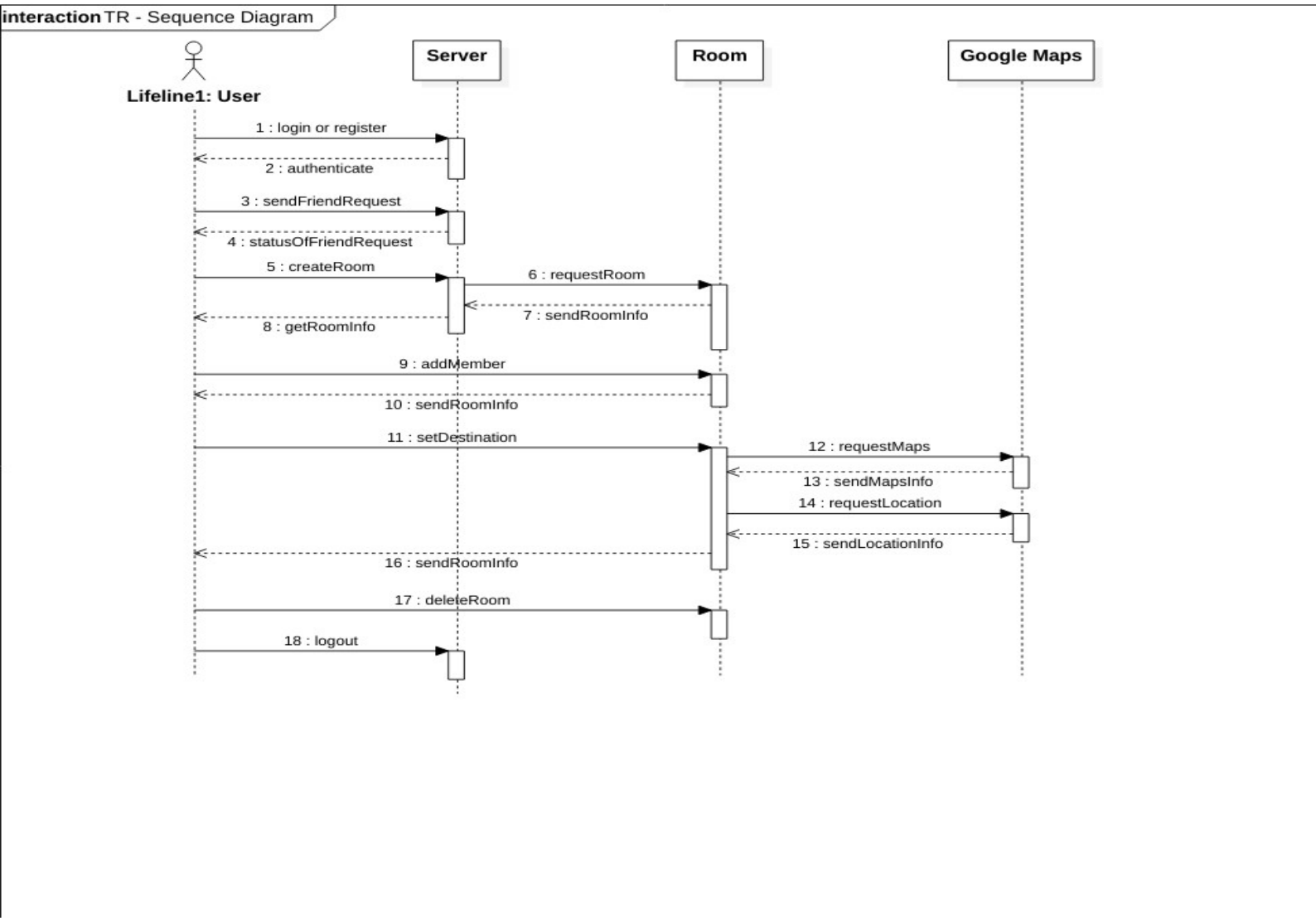


Fig. 4.1 Sequence Diagram

4.2 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. 4.2 Use Case Diagram

4.3 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.

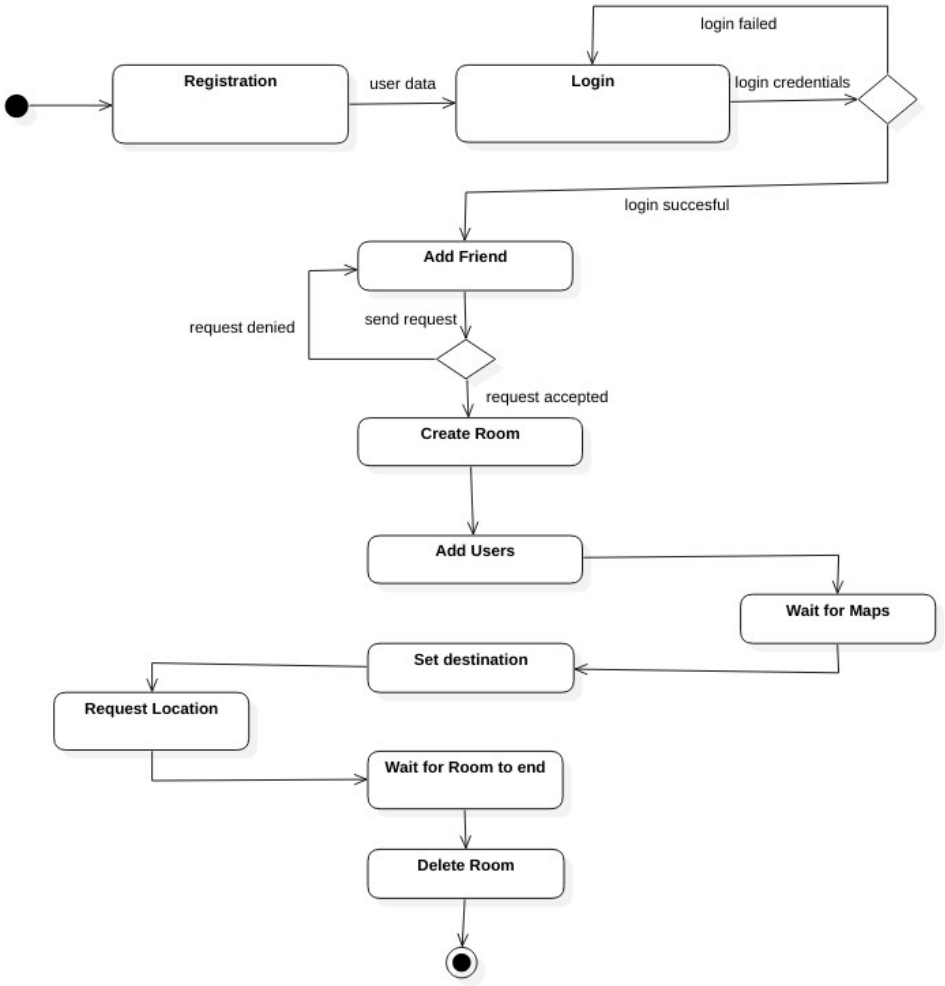


Fig. 4.3 State Transition Diagram

4.4 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."

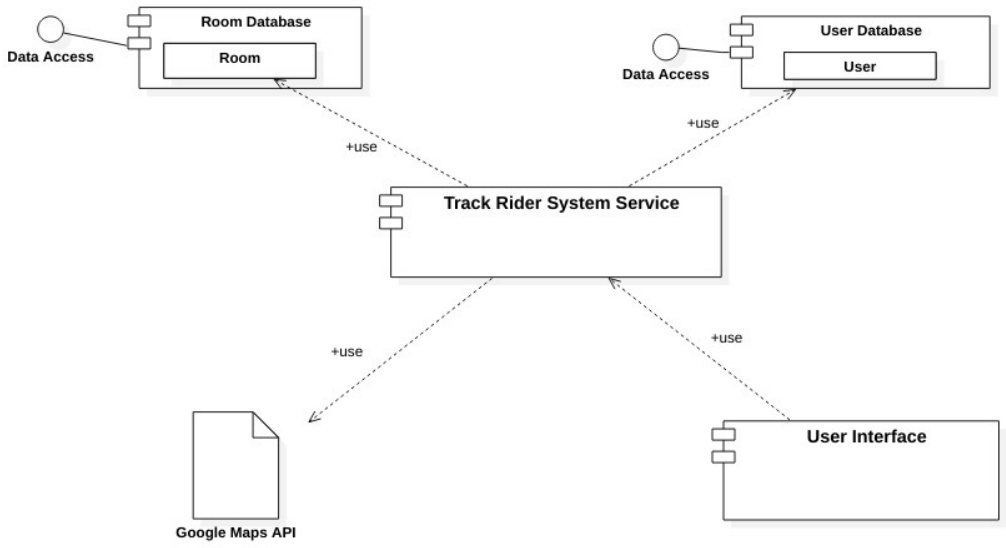


Fig. 4.4 Component Diagram

4.5 Colloboration 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.

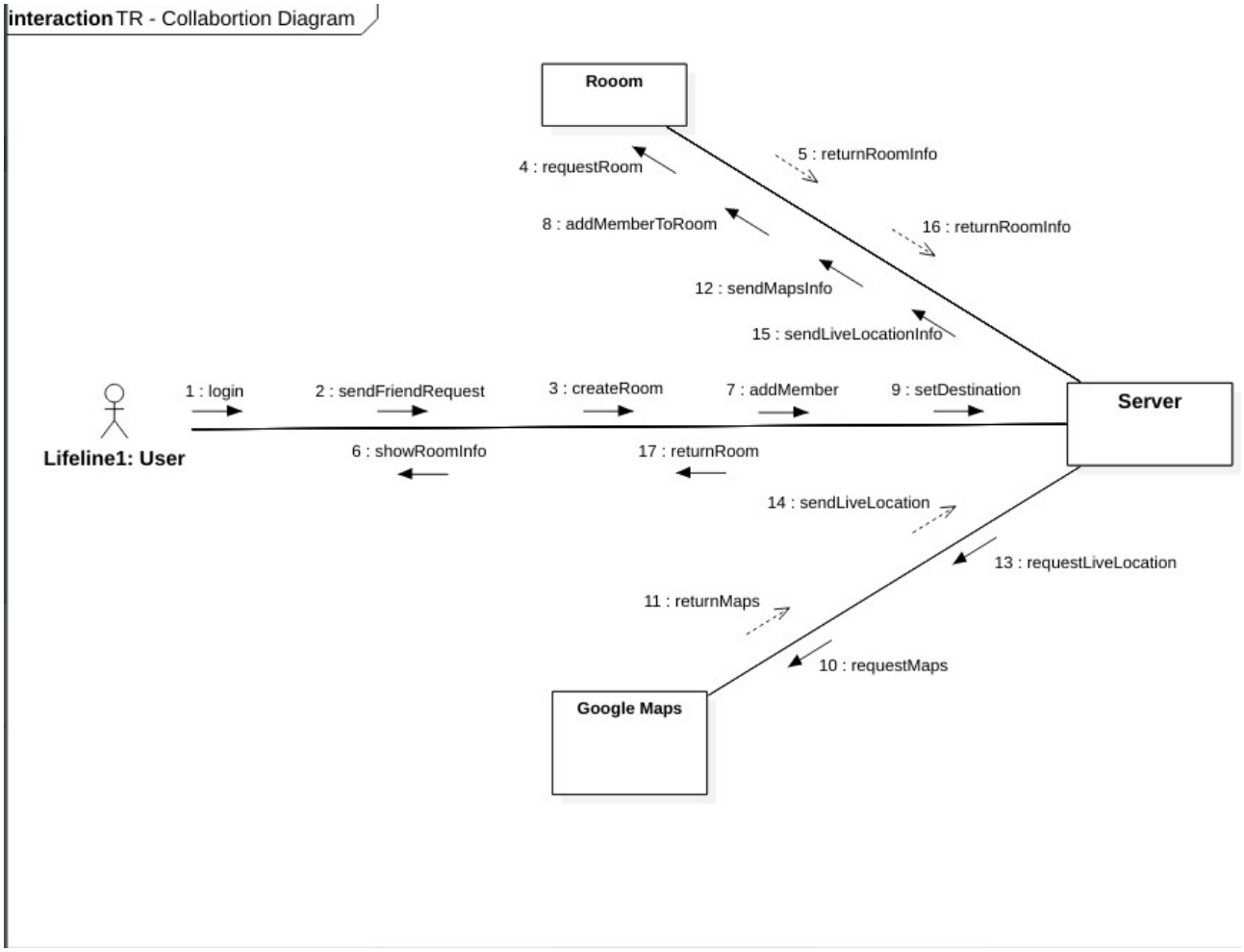


Fig. 4.5 Colloboration Diagram

4.6 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.

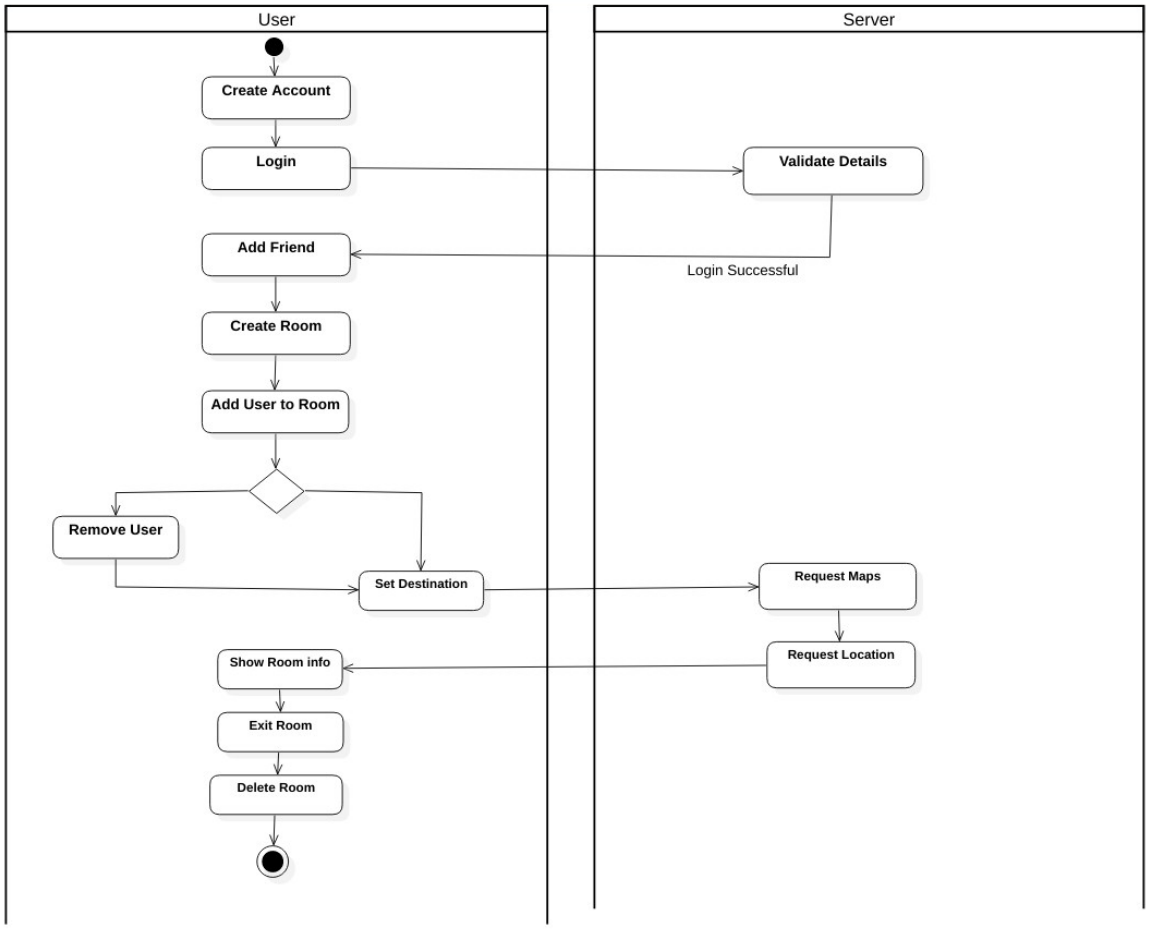


Fig. 4.6 Activity Diagram

4.7 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.

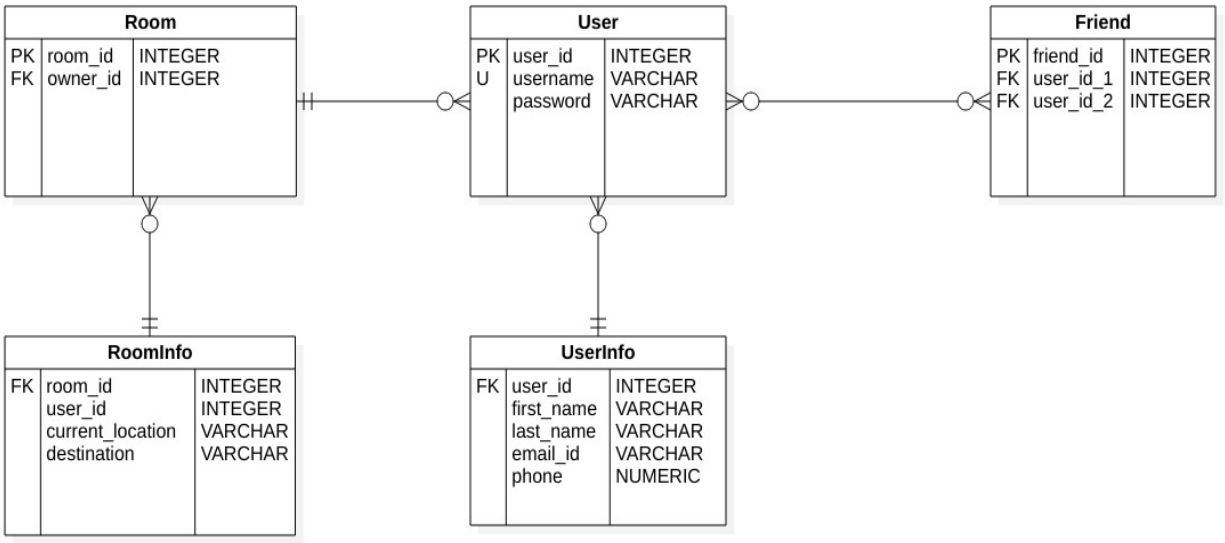


Fig. 4.7 ER Diagram

4.8 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.

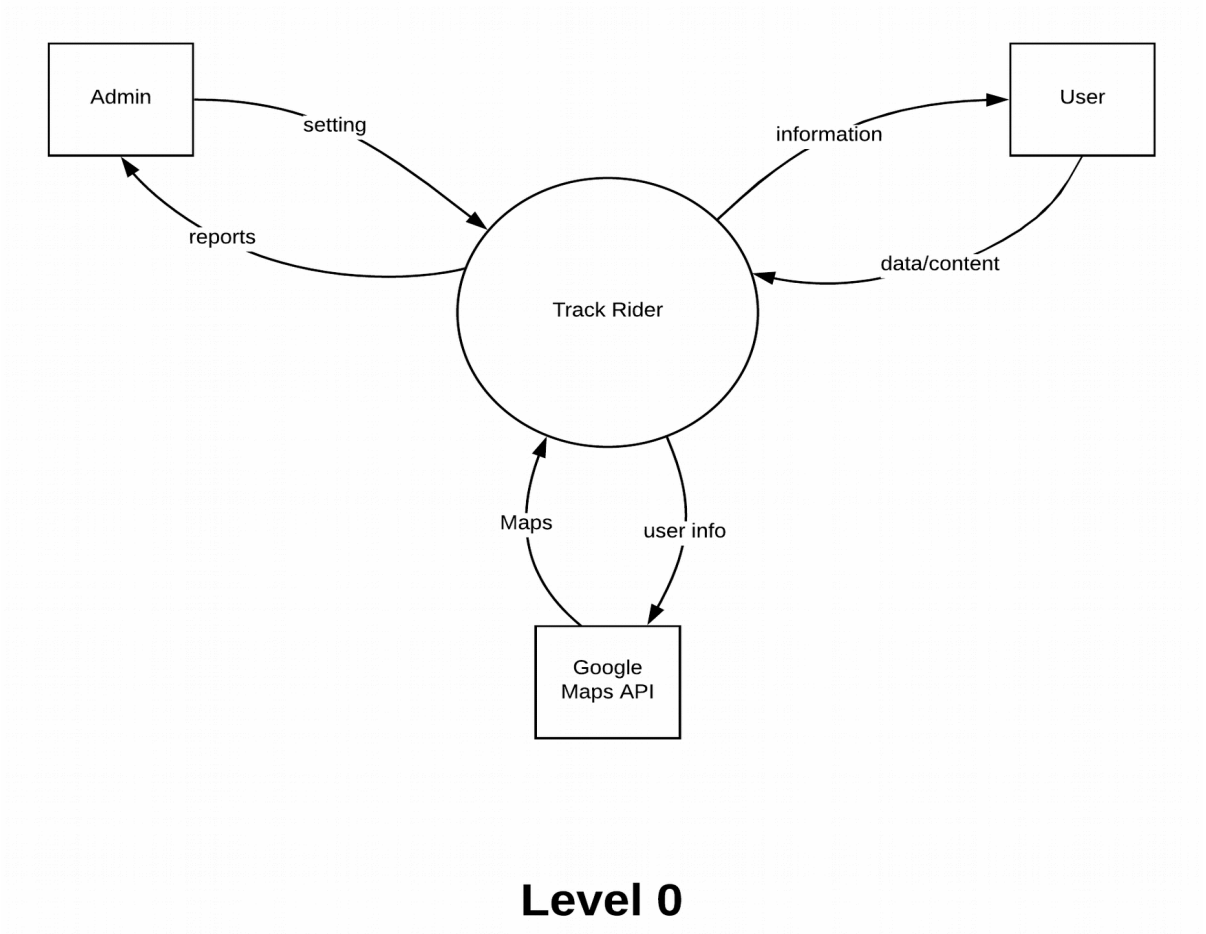
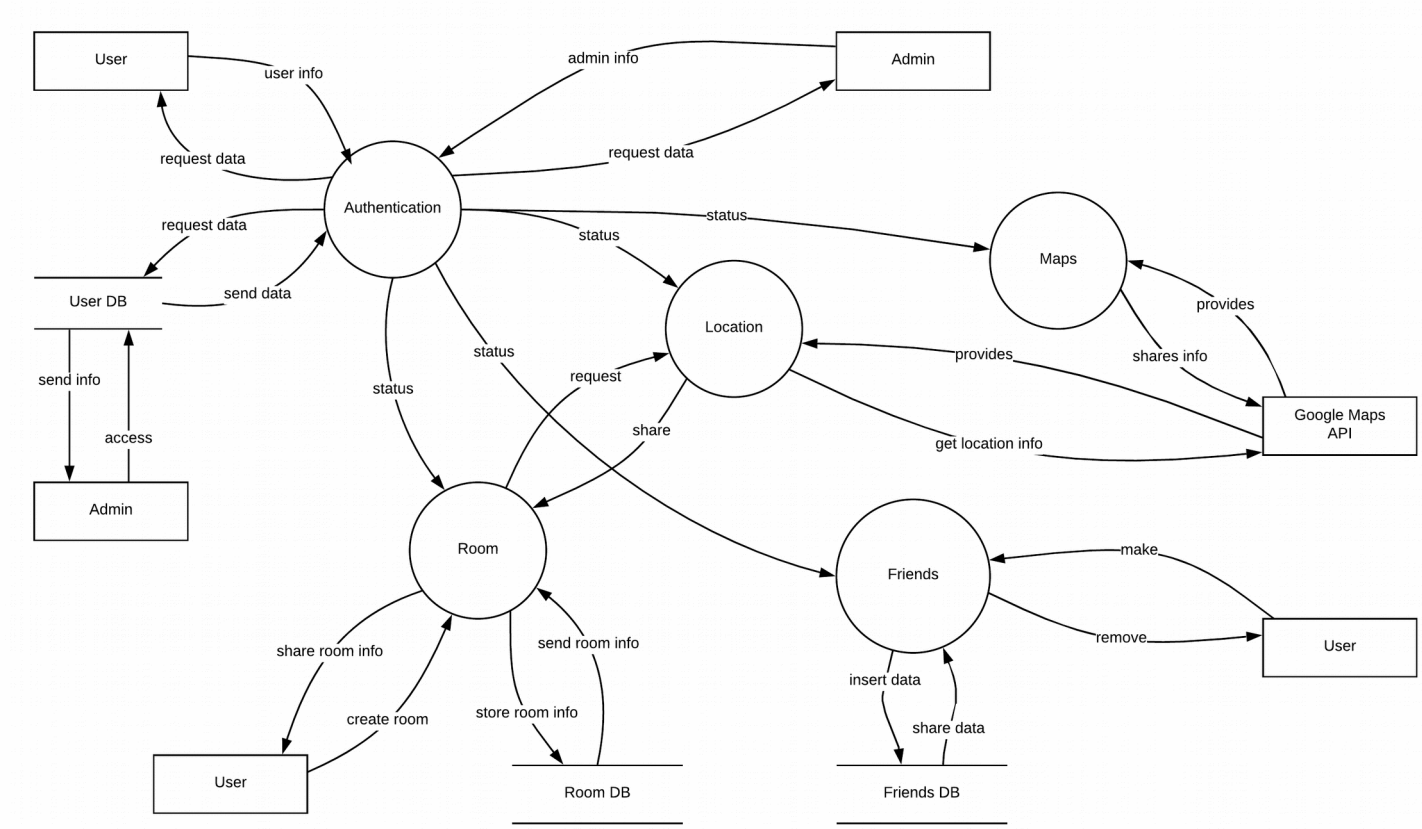


Fig. 4.8.1 DFD Diagram (Level 0)



Level 1

Fig. 4.8.1 DFD Diagram (Level 1)