**TRACK RIDER**

**A Project Report**

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)**

**By**

Pankaj Makhija (25) &

Indra Chawaria (65)

**Under the esteemed guidance of**

**Mr. Haresh Pawar**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**V.E.S COLLEGE OF ARTS, SCIENCE AND COMMERCE**

***(Affiliated to University of Mumbai)***

**CHEMBUR, 400071**

**MAHARASHTRA**

**2018-2019**

**V.E.S COLLEGE OF ARTS, SCIENCE AND COMMERCE**

***(Affiliated to University of Mumbai)***

**CHEMBUR – MAHARASHTRA – 400071**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is to certify that the project entitled, **“Track Rider”**, is bonafied work of **Pankaj Makhija** bearing Seat No: 25 submitted in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY** from University of Mumbai.

**Internal Guide Coordinator**

**External Examiner**

**Date: College Seal**

**PROFORMA FOR THE APPROVAL PROJECT PROPOSAL**

PNR No.: …………………… Roll No. : 25

1. Name of the Student: Pankaj Makhija

2. Title of the Project: Track Rider

3. Name of the Guide: Mr. Haresh Pawar

4. Teaching experience of the Guide:

5. Is this your first submission? Yes No

Signature of the Student Signature of the Guide

Date: ………………… Date: …………………….

Signature of the Coordinator

Date: …………………

**ABSTRACT**

Back in the day, Riding & Touring wasn't much popular with owning a vehicle just being a luxury that some can afford. Now a days, vehicles are becoming very common & owning a vehicle which can make our life easier in the city & gives equal the fun while riding long distances is not a luxury anymore but a necessity. Due to this, Riding Community is growing at a pace that is never seen before. Riding isn’t easy & when done wrong can be fatal. Making products for riders which ease their riding are accepted widely in the riding community. Various products have already been made for such purpose. Such a product is this Android Application which helps them keeping a track on their partners on the go. This paper is a review of Track Rider mobile application which is made for Android Devices.

**ACKNOWLEGEMENT**

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank Dr. (Mrs.) J.K. Phadnis, Principal of V.E.S College of Arts, Science and Commerce for providing me an opportunity to do the project work in the college premises and giving us all support and guidance which made me complete the project duly.

I owe my deep gratitude to our Co-ordinator Prof. (Mrs.) Jayalakshmi Srinivasan, who took keen interest on our project work and guided us all along, till the completion of our project report by providing all the necessary information for developing a good report.

I would not forget to remember Prof. (Mr.) Digvijay Parab and Prof. (Mr.) Ganesh Anandraj, for their encouragement and more over for their timely support and guidance till the completion of our project work.

I heartily thank our internal project guide, Prof. (Mr.) Haresh Pawar, for his guidance and suggestions during this project report.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of Information Technology which helped us in successfully completing our project work.

**DECLARATION**

I here by declare that the project entitled, “Track Rider” done at V.E.S College of Arts, Science and Commerce, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY) to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

**TABLE OF CONTENTS**

**CHAPTER 1: INTRODUCTION**

**1.1 Background**

**1.2 Objectives**

**1.3 Purpose, Scope, and Applicability**

**1.3.1 Purpose**

**1.3.2 Scope**

**1.3.3 Applicability**

**1.4 Achievements**

**1.5 Organization of Report**

**CHAPTER 2: SURVEY OF TECHNOLOGIES**

**CHAPTER 3: REQUIREMENTS AND ANALYSIS**

**3.1 Problem Definition**

**3.2 Requirements Specification**

**3.3 Planning and Scheduling**

**3.4 Software and Hardware Requirements**

**3.5 Preliminary Product Description**

**CHAPTER 4: SYSTEM DESIGN**

**4.1 Basic Modules**

**4.2 Data Design**

**4.2.1 Schema Design**

**4.2.2 Data Integrity and Constraints**

**4.3 Procedural Design**

**4.3.1 Logic Diagrams**

**4.3.2 Data Structures**

**4.3.3 Algorithms Design**

**4.4 User interface design**

**4.5 Secur****ity Issues**

**4.6 Test Cases Design**

**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

1. **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**

**3.3 Planning & Scheduling**

**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**

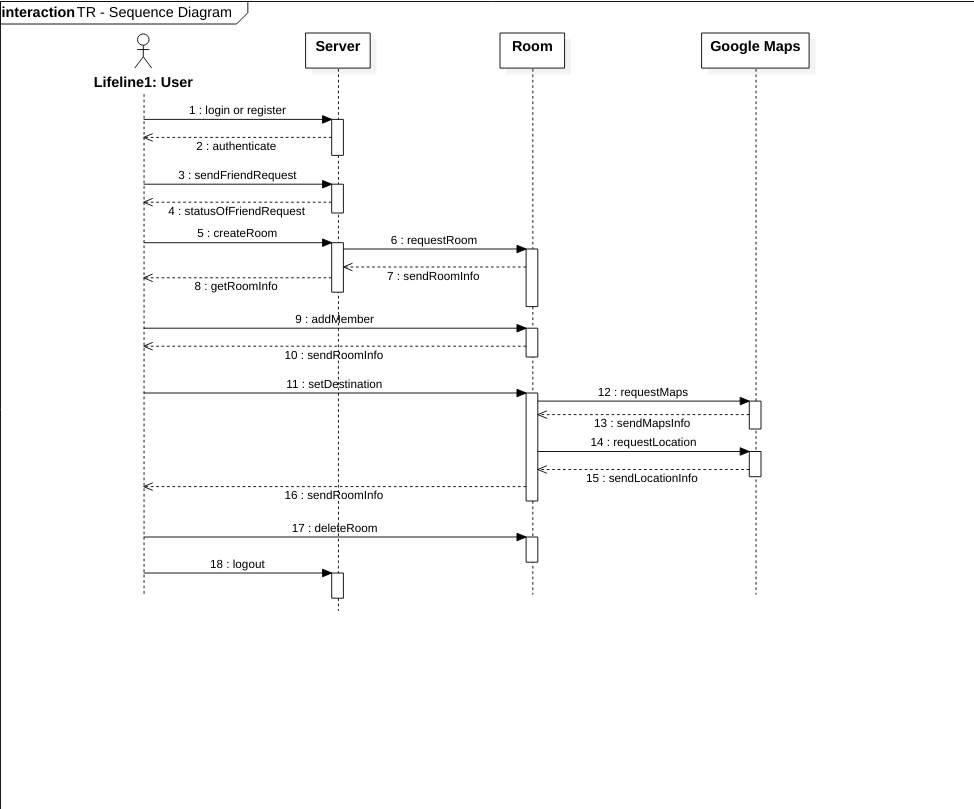
| Software Requirements | Hardware Requirements |
| --- | --- |
| Android v5.0 or higher | 1GB RAM or higher  Qualcomm Snapdragon 450 or higher  Atleast 100MB free space |

**4. System Design**

**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.

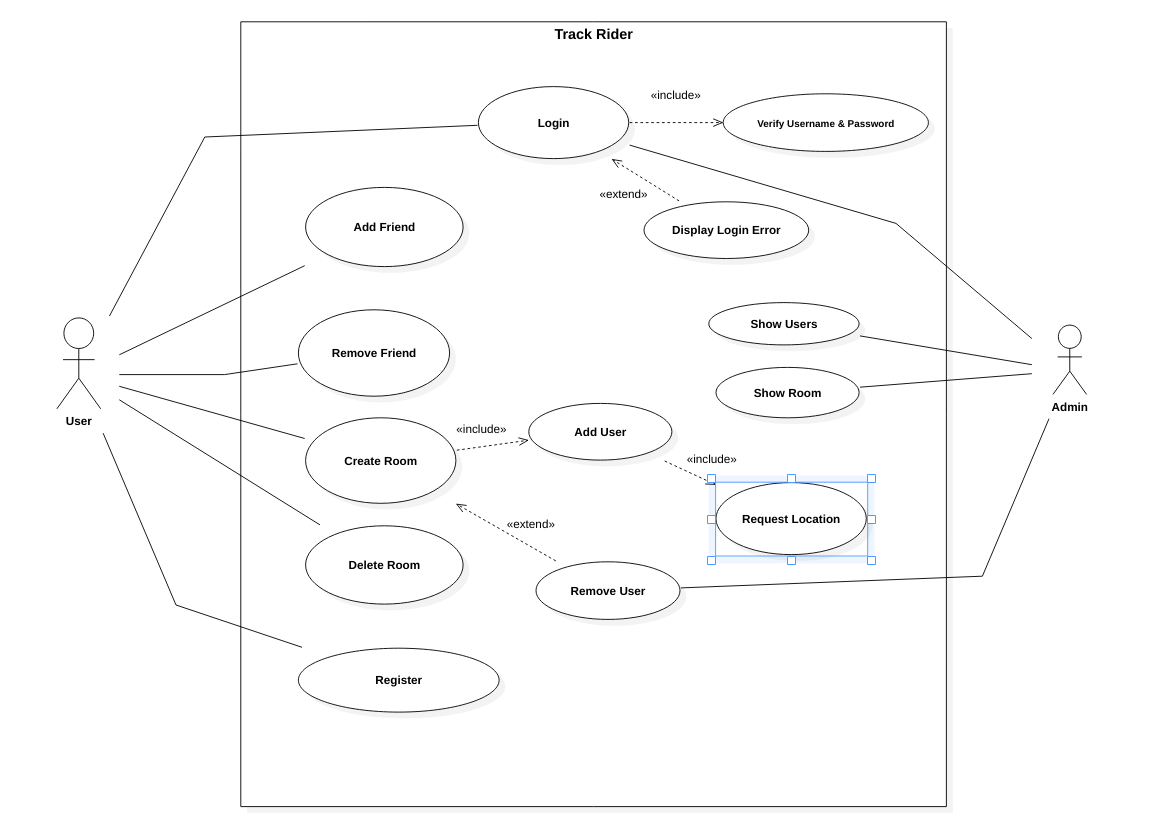
****

**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.

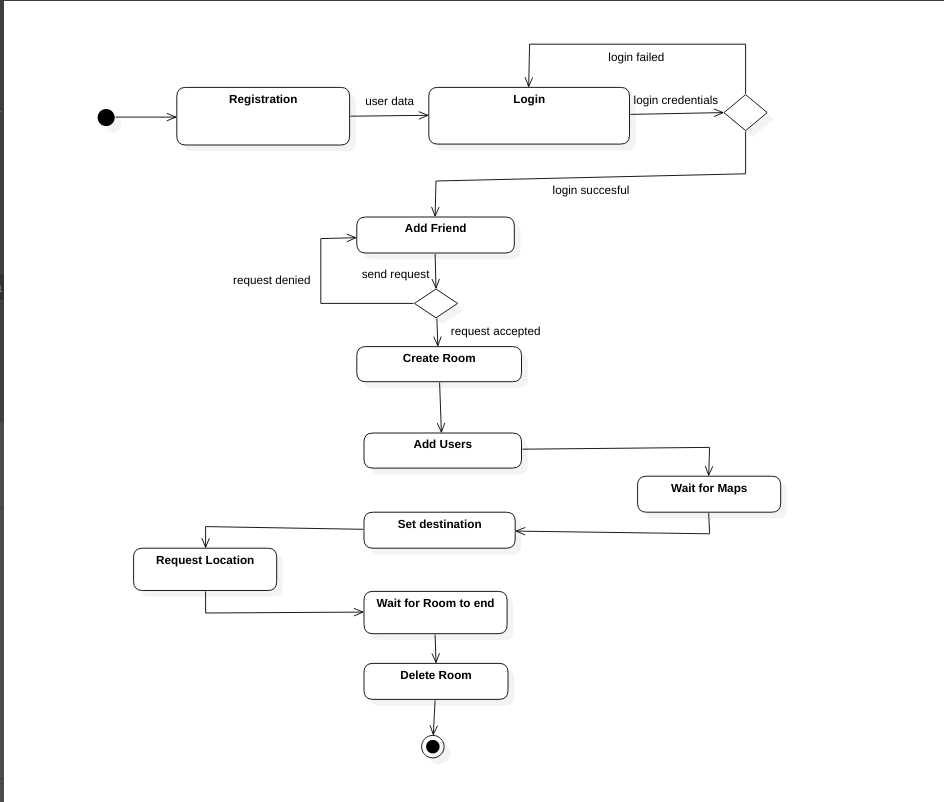


**State Transition Diagram**

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 [a](https://en.wikipedia.org/wiki/Abstraction_(computer_science))bstraction. Many forms of state diagrams exist, which differ slightly and have different [s](https://en.wikipedia.org/wiki/Semantics" \l "Computer_science)emantics.

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.

****

**Component Diagram**

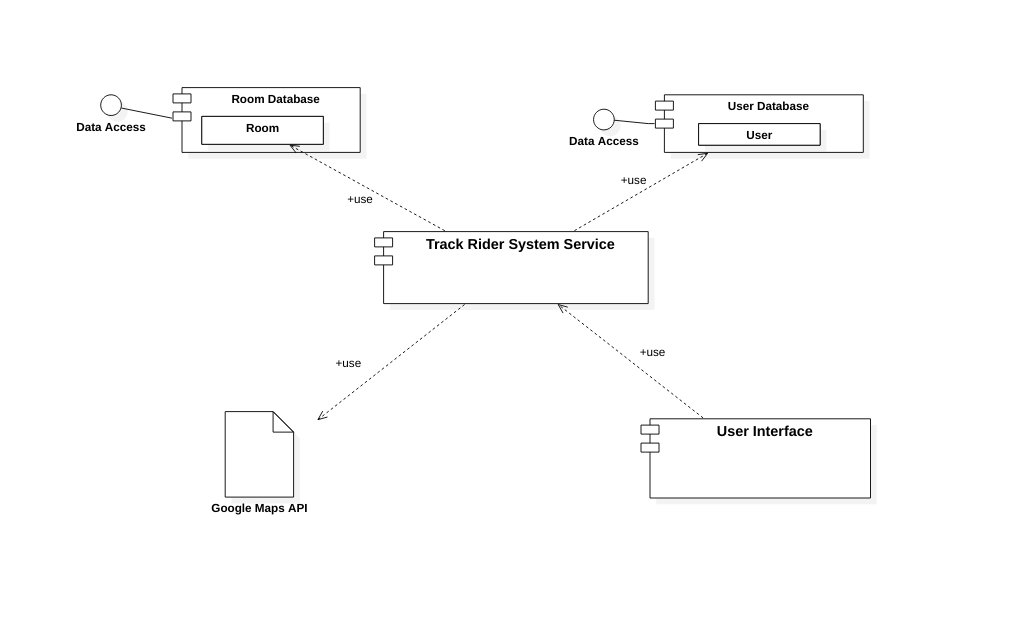
In Unified Modeling Language (UML), a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

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."

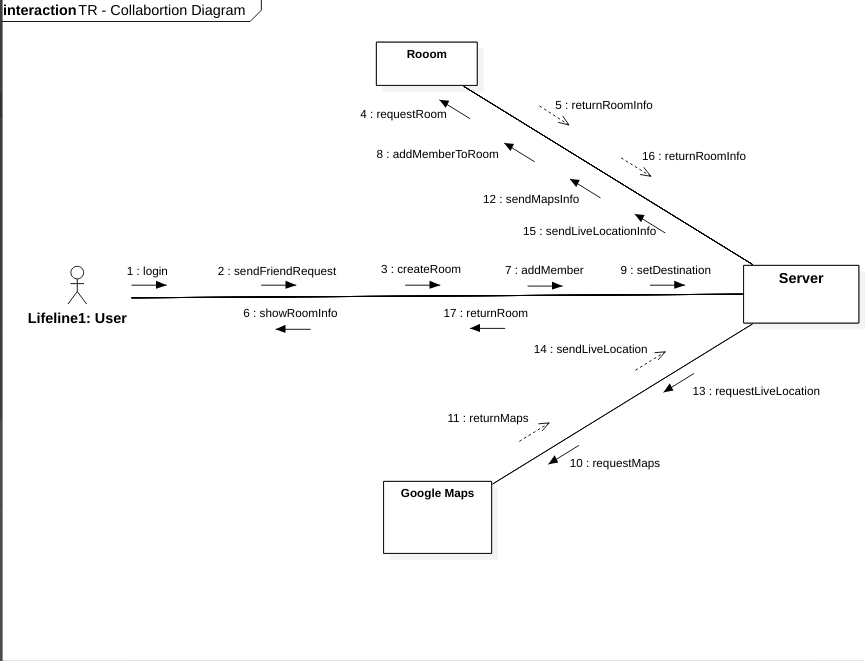
****

**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.

****

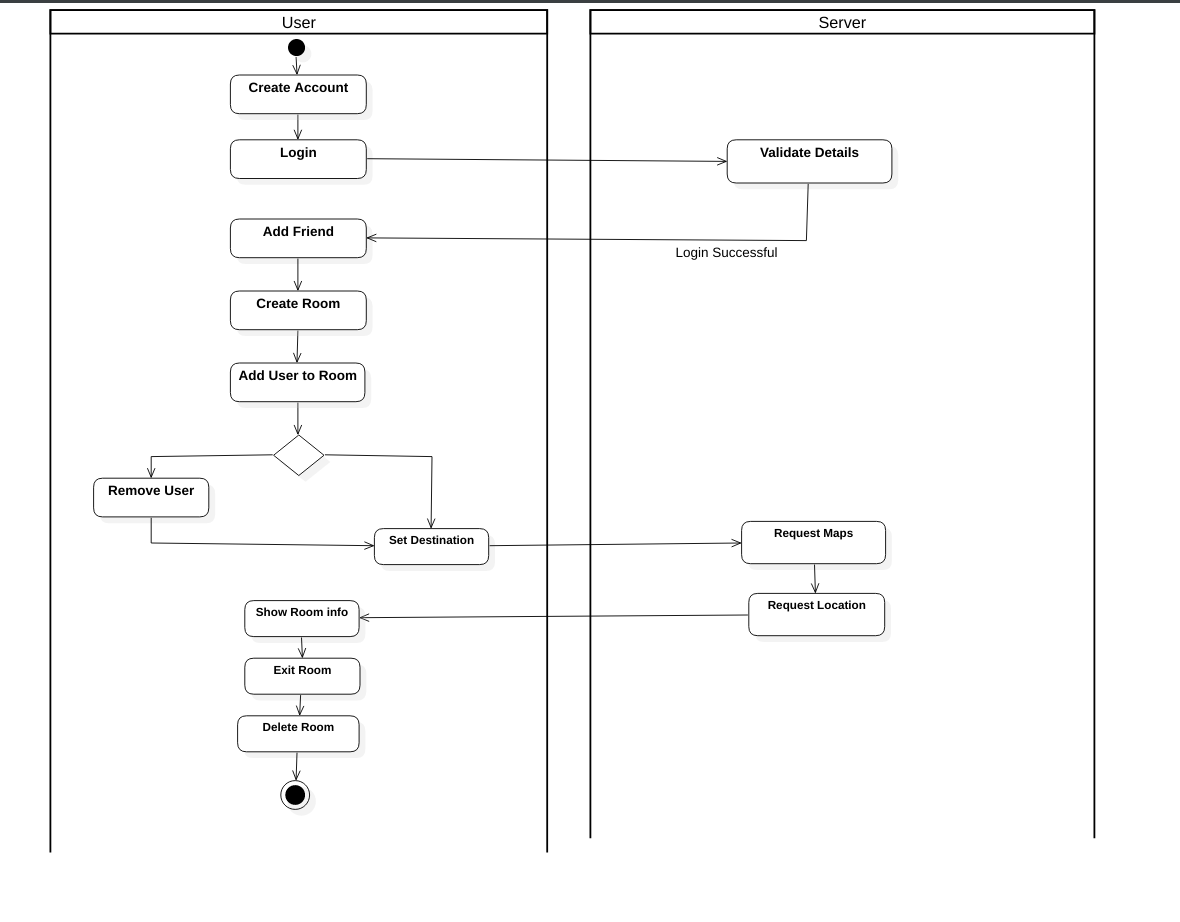
**Activity Diagram**

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.

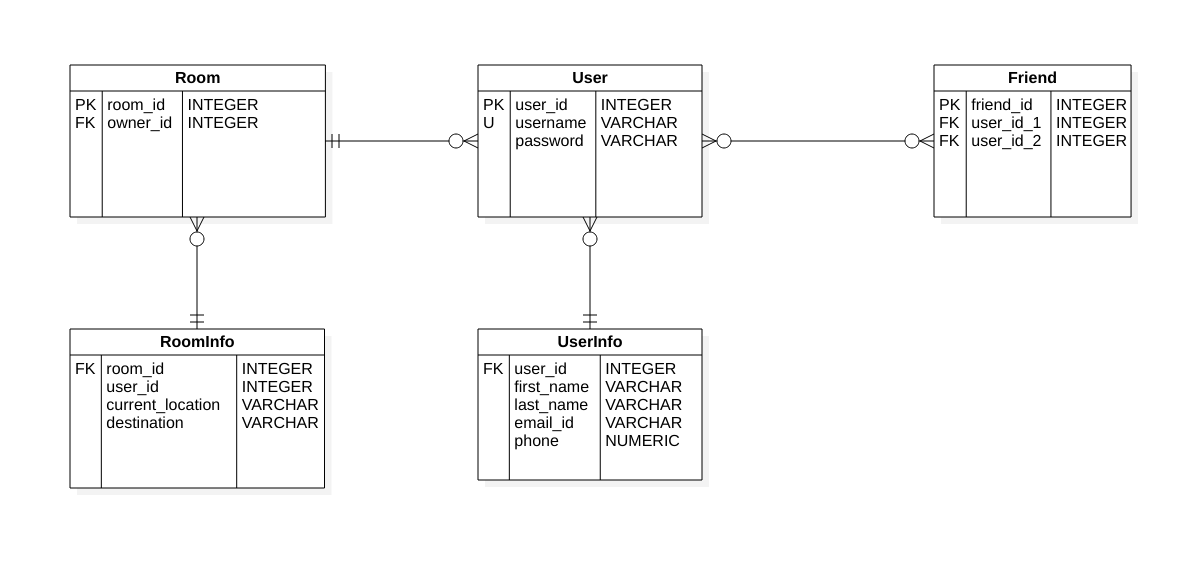
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message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



**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.



**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.

