Table of Contents

[Chapter-1: Overview 4](#_Toc6358184)

[Introduction 4](#_Toc6358185)

[Justification of System/ Background of the Project 4](#_Toc6358186)

[Problem Statement 4](#_Toc6358187)

[Description of System 5](#_Toc6358188)

[Features of the project 5](#_Toc6358189)

[Chapter-2: Scope 6](#_Toc6358190)

[Aims 6](#_Toc6358191)

[Objectives 6](#_Toc6358192)

[Overview of the System scope 7](#_Toc6358193)

[Limitations 7](#_Toc6358194)

[Chapter-3: Development Methodology 8](#_Toc6358195)

[Waterfall model 8](#_Toc6358196)

[MVC 9](#_Toc6358197)

[Architecture 10](#_Toc6358198)

[Chapter-4: Work Breakdown Structure (WBS)/ Project Planning 11](#_Toc6358199)

[Time estimation 12](#_Toc6358200)

[Milestone 13](#_Toc6358201)

[Scheduling/ Gantt Chart 13](#_Toc6358202)

[Chapter-5: Other Project Activities 15](#_Toc6358203)

[Risk Management 15](#_Toc6358204)

[Configuration management 16](#_Toc6358205)

[Chapter 5:Analysis 19](#_Toc6358206)

[Requirement 19](#_Toc6358207)

[MoSCow Prioritization 19](#_Toc6358208)

[Use-Case diagram 20](#_Toc6358209)

[Introduction 20](#_Toc6358210)

[Justification 21](#_Toc6358211)

[Advantages 21](#_Toc6358212)

[Disadvantages 21](#_Toc6358213)

[Chapter 6: Design 22](#_Toc6358214)

[Structural Model 22](#_Toc6358215)

[Class Diagram 22](#_Toc6358216)

[Justification 23](#_Toc6358217)

[Advantages 23](#_Toc6358218)

[Disadvantages 23](#_Toc6358219)

[ER Diagram 24](#_Toc6358220)

[Justification 25](#_Toc6358221)

[Advantages 25](#_Toc6358222)

[Disadvantages 25](#_Toc6358223)

[Behavioral model 26](#_Toc6358224)

[Activity Diagram 26](#_Toc6358225)

[Vehicle Rental 26](#_Toc6358226)

[Driver Hire 27](#_Toc6358227)

[Justification 28](#_Toc6358228)

[Advantages 28](#_Toc6358229)

[Disadvantages 28](#_Toc6358230)

[UI design 29](#_Toc6358231)

[Chapter-8: Testing 33](#_Toc6358232)

[Unit Testing 33](#_Toc6358233)

[Chapter-9: Conclusion 35](#_Toc6358234)

[Chapter-10: References and Bibliography 35](#_Toc6358235)

Table of Figures

[Figure 1: Waterfall model 9](#_Toc6358238)

[Figure 2: MVC pattern 10](#_Toc6358239)

[Figure 3: 3-tier-architecture 11](#_Toc6358240)

[Figure 4: WBS 12](#_Toc6358241)

[Figure 5: Time allocation 13](#_Toc6358242)

[Figure 6: Milestone track 14](#_Toc6358243)

[Figure 7: Table for Gantt chart 15](#_Toc6358244)

[Figure 8: Gantt chart 15](#_Toc6358245)

[Figure 9: Risks 16](#_Toc6358246)

[Figure 10: Consequences 16](#_Toc6358247)

[Figure 11: Impacts and Action 17](#_Toc6358248)

[Figure 12: Configuration 18](#_Toc6358249)

[Figure 13: GitHub 18](#_Toc6358250)

[Figure 14: Use Case diagram 21](#_Toc6358251)

[Figure 15:Class Diagram 23](#_Toc6358252)

[Figure 16:Final Class Diagram 24](#_Toc6358253)

[Figure 17: ER diagram 25](#_Toc6358254)

[Figure 18:Activity Diagram for Rental 27](#_Toc6358255)

[Figure 19: Hire driver with vehicle 28](#_Toc6358256)

[Figure 20: View location 28](#_Toc6358257)

[Figure 21: Login form 30](#_Toc6358258)

[Figure 22: User dashboard 31](#_Toc6358259)

[Figure 23: Staff dashboard 32](#_Toc6358260)

[Figure 24: Owner dashboard 33](#_Toc6358261)

[Figure 25: Model number test 34](#_Toc6358262)

[Figure 26: Email format test 34](#_Toc6358263)

[Figure 27:Test when accurate result is returned 35](#_Toc6358264)

# Chapter-1: Overview

## Introduction

Vehicle monitoring with location tracking involves driver or vehicle tracking using Global Positioning System (GPS) which is a satellite-based navigation for marking the position of someone/something. This tracking is done via storing API key (which stores information about latitude and longitude) of users or vehicles and using it to visualize the appropriate location in the globe using Google Maps. This system targets users or companies which face the loss of vehicles and huge amount of finances as they cannot trust everyone completely. Also, login and registration is essential before handing the users the corresponding vehicles and admin monitor it accordingly. Also, this system provides reminders about vehicle insurance, tax, fuel and servicing date managing the information that users find difficult to remember. This application acts an interface of communicating with our vehicle.

## Justification of System/ Background of the Project

We cannot trust even our family when it comes to financial affairs and trusting someone to return our vehicle (which we bought through a big struggle) is somehow foolish. Also, with many things going on in our life, we might not be able to remember about insurance and tax payment, servicing. We might have to leave our car somewhere and walk home someday only because we didn’t notice that our car is out of gas. People cannot always have time to manage information about their car and at some event of time they might have to face issues legally about tax payment just because they couldn’t get notified about tax payment which they forgot to record. We might not always be able to access our paper log we keep at our home to update our information anywhere at any time. Demographic studies suggest that people would at least look at their phone once if they get a simple notification because mobile is a part of their life now. (MobileStudies,2016)

## Problem Statement

Also, company cannot manage bulk of information manually as they have to manage numerous vehicles and monitoring is somewhat difficult in the scenario. This system is solution to vehicle security, monitoring and condition. People would get notified about information about their payments, servicing, fuel amount right in their phones which supports the condition of the car and eliminates issues relating to tax payments. Also, their cars are secure as they can easily monitor the exact location of their cars using GPS through API key of individual or the car itself. This would reduce the chances of vehicle loss vastly and people could be assured about vehicle return accordingly. Internet disconnection and false API key might lead to complications in the future and hence internet available globally (Wi-Max, Satellite) and correct information must be used.

## Description of System

The system is developed via Android using framework of Android Studio and Firebase as the database for data management. The system would enable admin to monitor their vehicle location on the google map using Google Maps API. The parameters are obtained via user table and vehicle table. The API key of users (to which vehicles are allocated) is entered during their registration and admin verifies the users accordingly and hence they are eligible for taking the vehicles. Also, we can enter our vehicle information with the API key within the vehicle table in the database. This table would also comprise information about payment dates and fuel quantity. Admin would be the owner themselves be it individual or the company. They can validate a user after their registration and only approve the request after studying documents if essential. The application comprises User mode and Company mode. User mode is limited in vehicle tracking and monitoring.

Company mode enables us to allocate salaries to drivers hired by the company itself and users can hire drivers accordingly. The management of their information is also made easier by this system and communication between the admin and driver can be done accordingly. Copy of License of the drivers, citizenship or passport is essential for driver registration though and it’s in the hand of the admin to hire them. Also, contact information of reconditioning house, service center can be entered and retrieved accordingly (in the user mode as well).

## Features of the project

* Owners can track vehicle and driver location using GPS.
* API key for location mark is entered during registration of vehicles and users.
* The accurate location is determined using satellite navigation and internet access.
* Users can enter payment date, deadlines, servicing date manually.
* They can update payment information and date.
* Company can register new drivers and keep hold of their vital information.
* Company can edit hired driver information accordingly.
* Company can allocate salary to their drivers.
* Users get notified about any key dates and if fuel is getting low.
* Owners can communicate with their drivers.
* The contact information about recondition house, service center and others can be stored.
* For handing vehicle to anyone, owner can ask people to register their information and API key.
* Vehicle information can be entered or edited accordingly.

# Chapter-2: Scope

Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, features, functions, tasks, deadlines, and ultimately costs. It is important to pin down the scope early in a project’s life cycle as it can greatly impact the schedule or cost (or both) of the project down the track.

## Aims

The major aims of the project are:

* To provide an interface to the owner to track location of their vehicles and be addressed about any suspicious activity of the driver.
* To provide an interface to enter, retrieve and edit information about their vehicle payments, other driver information, payment schedule, deadlines, fuel quantity and address owners accordingly.
* Overcome the problem of vehicle loss causing major economic impact to the people.
* Eliminate the cost of extra manpower required to manage user data manually and automate it by directly entering into database.

## Objectives

The above aims of Vehicle Monitoring System are met as:

* Payment record to be automated which eliminates the paper-based log about vehicle information of payments.
* Provide an interface for driver registration along with their API key.
* Location tracking by using the entered API key during registration.
* The interface of Google Map provided to admin(owners) to monitor their vehicles.
* Provide an interface to enter deadlines for payments and address owners through registration.
* Record information about vehicle condition and fuel in it.
* The system should address and provide approval privilege to owners after drivers register their information.
* Non-duplication of records shall be guaranteed.
* System should notify owners about any suspicious activity.
* The system should allocate salaries to employed drivers based on their work and record the information accordingly.
* Provide walkthrough to users who are new to the application.

## Overview of the System scope

* Users can track and monitor their vehicles from anywhere and contact the drivers accordingly.
* They do not have to maintain a manual paper record about their vehicle payments like insurance, servicing and tax.
* Users are notified about their payment deadlines and fuel quantity in their phone which helps them become aware if they had forgotten.
* Admin can always reject any registration if they don’t trust the users asking for approval.
* Company can automate the driver and salary records eliminating the manual account and calculate the salary based on it.

### Limitations

* Internet connectivity is a must for this project and on some hazards, correct information might not be provided to owners.

# Chapter-3: Development Methodology

## Waterfall model

We approach the system development through Waterfall model which is a linear-sequential life cycle model which is divided into phases where each phase must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. The phases are:

* Requirement Analysis
* System design
* Implementation
* Integration and testing
* Deployment of the system
* Maintenance

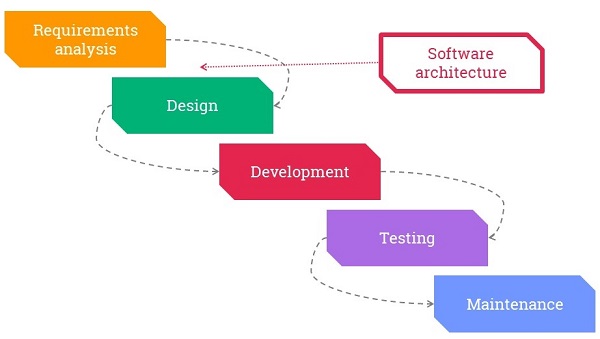


Figure 1: Waterfall model

## MVC

MVC design pattern would be used for system design and development. The **Model View Controller** (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects. MVC is more of an architectural pattern, but not for complete application. MVC mostly relates to the UI / interaction layer of an application.



Figure 2: MVC pattern

* The **Model** contains only the pure application data, it contains no logic describing how to present the data to a user.
* The **View** presents the model’s data to the user. The view knows how to access the model’s data, but it does not know what this data means or what the user can do to manipulate it.
* The **Controller** exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events.

## Architecture

Different parts of systems (architecture) communicates with each other to handle a request and provide output as per incoming request and 3-tier-architecture is used here. 3-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. It composes:

1. A **Presentation Layer** that sends content to browsers in the form of HTML/JS/CSS. This might leverage frameworks like React, Angular, Ember, Aurora, etc.
2. An **Application Layer** that uses an application server and processes the business logic for the application. This might be written in C#, Java, C++, Python, Ruby, etc.
3. A **Data Layer** which is a database management system that provides access to application data. This could be MSSQL, MySQL, Oracle, or PostgreSQL, Mongo, etc.

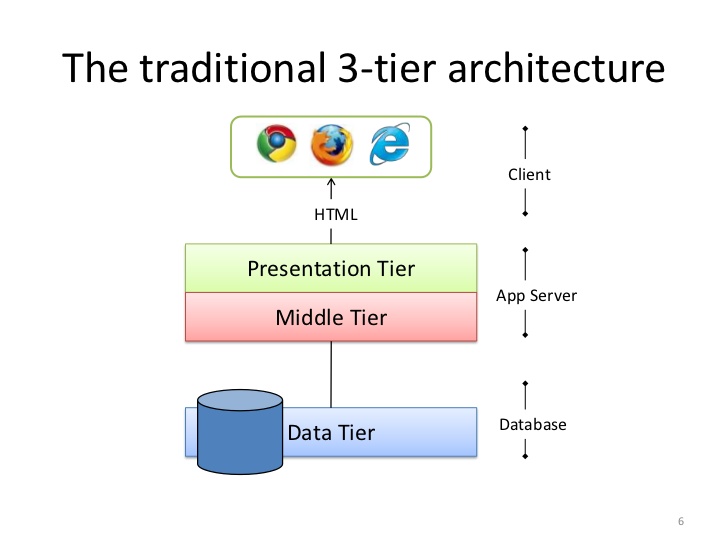


Figure 3: 3-tier-architecture

# Chapter-4: Work Breakdown Structure (WBS)/ Project Planning

A [work breakdown structure](https://www.matchware.com/wbs-software) (WBS) is a key project deliverable that organizes the team's work into manageable sections. The Project Management Body of Knowledge ([PMBOK](https://www.workbreakdownstructure.com/work-breakdown-structure-according-to-pmbok.php)) defines the work breakdown structure as a "deliverable oriented hierarchical decomposition of the work to be executed by the project team. The work breakdown structure visually defines the scope into manageable chunks that a project team can understand, as each level of the work breakdown structure provides further definition and detail. (WBS.com,2014)

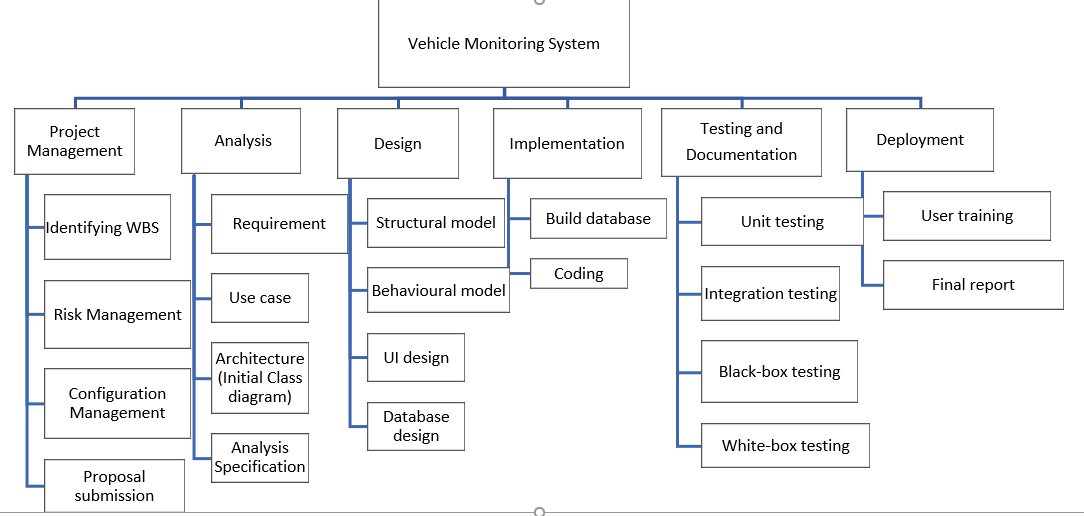


Figure 4: WBS

## Time estimation

The time estimation of each of the tasks given in WBS is given as:

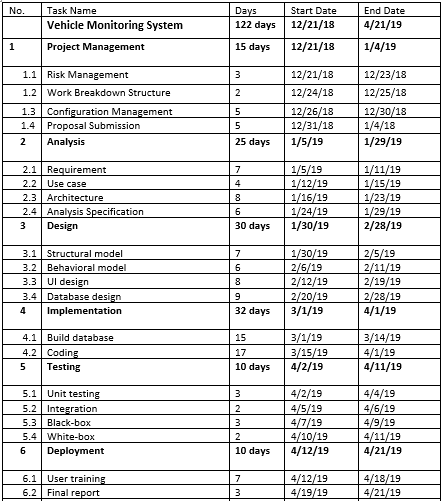
****

Figure 5: Time allocation

## Milestone

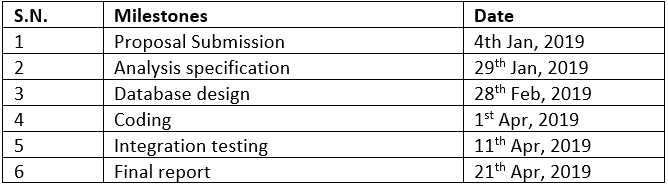
A project milestone is a management tool that is used to delineate a point in a project schedule. These points can note the start and finish of a project, and mark the completion of a major phase of work. Milestones can be used to symbolize anything that has started or finished, though it’s primarily used as a scheduling tool. ****

Figure 6: Milestone track

The proposal submission is to be submitted on 4th January completing every stages of project management. Analysis specification is to be concluded on 29th January and by 28th of February Design phase would end after completing database design. Coding would end on 1st April and integration testing which marks the end of testing is estimated to be on 11th April. The final report of the project is to be submitted on 21st April as per estimation.

## Scheduling/ Gantt Chart

**Project scheduling** is a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what timeframe. A **project schedule** is a document collecting all the work needed to deliver the **project** on time. A Gantt chart is a type of bar chart that illustrates a project schedule, named after its inventor, Henry Gantt, who designed such a chart around the years 1910–1915. (Wikipedia,2013) This scheduling is visualized using a Gantt chart given as:

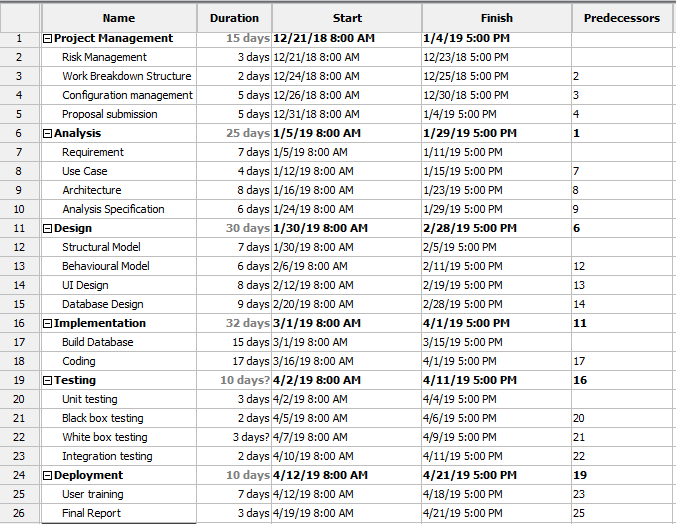


Figure 7: Table for Gantt chart

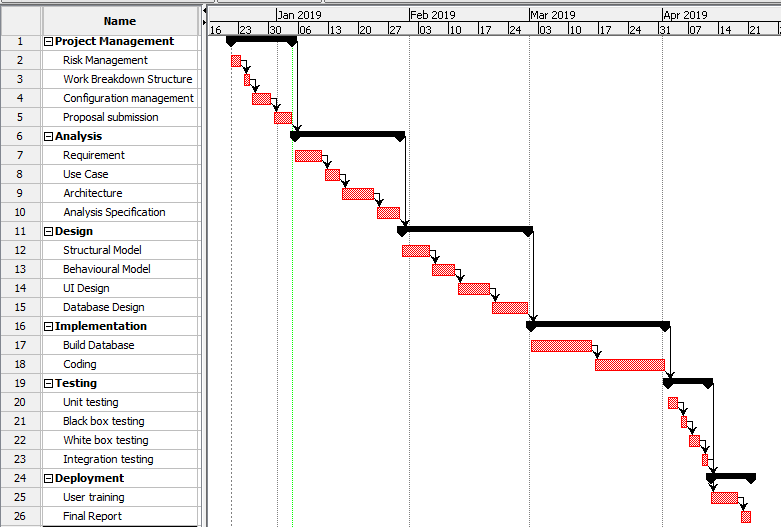


Figure 8: Gantt chart

# Chapter-5: Other Project Activities

## Risk Management

Managing risks on projects is a process that includes risk assessment and a mitigation strategy for those risks. Risk assessment includes both the identification of potential risk and the evaluation of the potential impact of the risk. A risk mitigation plan is designed to eliminate or minimize the impact of the risk events—occurrences that have a negative impact on the project. Identifying risk is both a creative and a disciplined process. The creative process includes brainstorming sessions where the team is asked to create a list of everything that could go wrong. All ideas are welcome at this stage with the evaluation of the ideas coming later. (ProjectManager,2013)

The learning objectives of risk management include:

1. Identify the major elements in managing project risk.
2. Describe the processes for identifying project risk.
3. Describe the processes for evaluating risk.
4. Describe the processes for mitigating risk.

The risks have likelihood based on scenario with different consequences, impact and acts need to be take accordingly. Risks are categorized into 3 categories as:

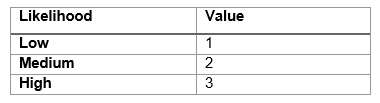


Figure 9: Risks

Also, the consequences are sub-divided as:

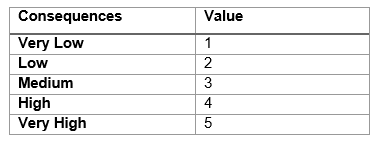


Figure 10: Consequences

Impact of the risk is calculated based on consequences using the formula:

**Impact= Likelihood\* Consequence**

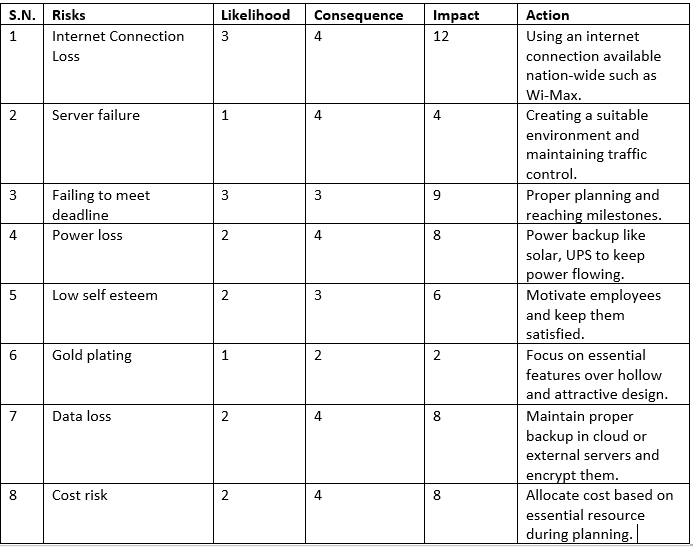


Figure 11: Impacts and Action

## Configuration management

Configuration management (CM) is a [governance](https://searchcompliance.techtarget.com/definition/information-governance) and [systems engineering](https://whatis.techtarget.com/definition/systems-engineering-SE) process for ensuring consistency among physical and logical assets in an operational environment. The configuration management process seeks to identify and track individual configuration items (CIs), documenting functional capabilities and interdependencies.  Administrators, technicians and software developers can use configuration management tools to verify the effect a change to one configuration item has on other systems. Developers and others involved in the project can use SCM to keep track of artifacts, including [source code](https://searchmicroservices.techtarget.com/definition/source-code), documentation, problems, [changes requested](https://searchcio.techtarget.com/definition/change-request) and changes made. (TechTarget,2015)

The process of configuration management involves:

1. Configuration identification
2. Configuration control
3. Configuration status accounting
4. Configuration audits

The following directories are made for configuration of the project and changes are applied accordingly:

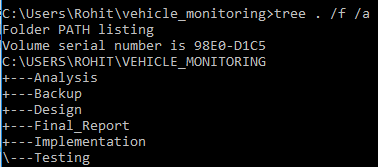


Figure 12: Configuration

The following directories comprise files related to project and they are organized based on sub-folders accordingly for easier access. Also, essential backup is stored in backup folder as well as pushed to Git-Hub as:

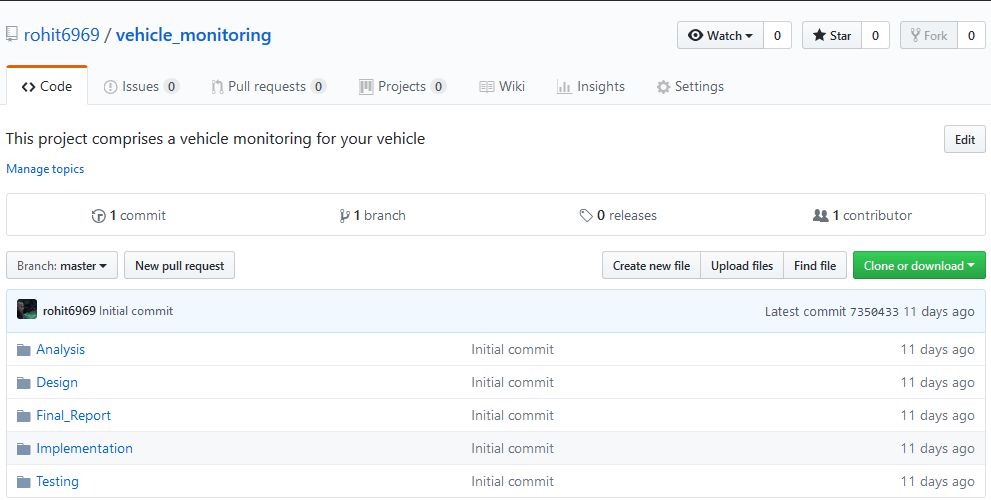


Figure 13: GitHub

Changes are committed here and it’s automatically updated in GitHub through commit. Only files that are committed face changes while others remain exactly the same. This acts as backup for the project as well.

# Chapter 5:Analysis

# Requirement

## MoSCow Prioritization

**Functional requirement:** The requirements of a system that are implementation within the system and we can actually use the corresponding functionalities is termed as functional requirement.

**Non-functional requirement:** The requirements of a system that are implemented as the behavior of the system over system functionalities is termed as non-functional requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Requirements** | **Functional**  **/Non-functional** | **MoSCow** |
| 1. | Login and Registration | Functional | Must have |
| 2. | View location | Functional | Must have |
| 3. | Hire vehicle | Functional | Should have |
| 4. | Rent vehicle | Functional | Should have |
| 5. | View fuel status | Functional | Should have |
| 6. | Add staff | Functional | Must have |
| 7. | Update staff | Functional | Must have |
| 8. | Delete staff | Functional | Must have |
| 9. | View users | Functional | Must have |
| 10. | Delete users | Functional | Must have |
| 11. | Efficiency | Non-functional | Must have |
| 12. | Security | Non-functional | Must have |
| 13. | Smooth performance | Non-functional | Could have |
| 14. | Reliable | Non-functional | Should have |
| 15. | Portability | Non-functional | Should have |

Table-1: Moscow prioritization

# Use-Case diagram

## Introduction

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated. System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task.

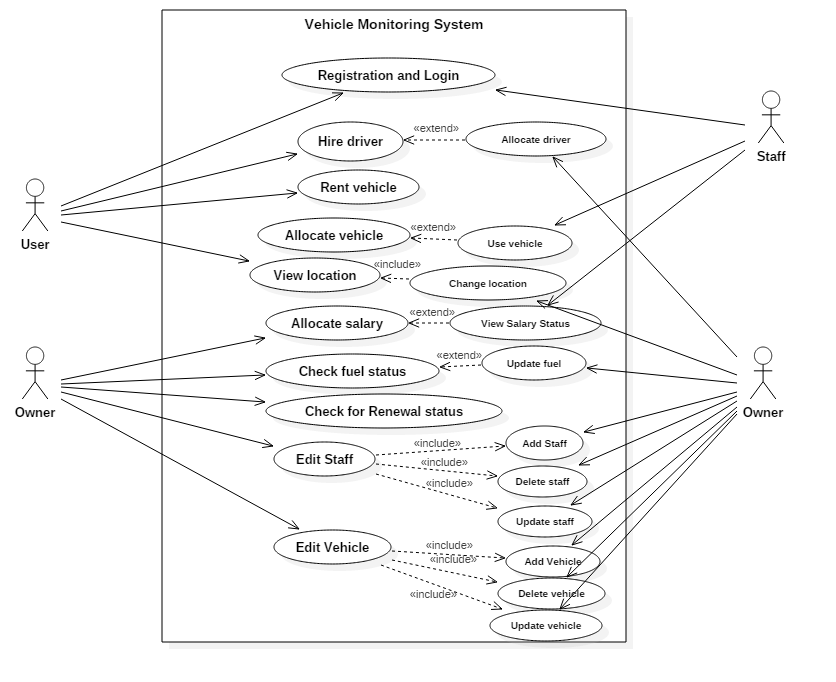


Figure 14: Use Case diagram

## Justification

### Advantages

For the scenarios below, use cases are ideal:

* Helps in detailing user interaction goals with a product.
* Relevant outlining and ensuring the requirements of a system.
* Determining the specific needs of a project.
* Modeling the basic flow of events in a use case.

### Disadvantages

* They do not capture the non-functional requirements easily.
* There might be a learning curve for the developer and/or specially, the client in using these use cases.
* Use case diagram is not suitable if we want to lay out step-by-step details. These diagrams summarize interactions, not explain them.

# Chapter 6: Design

# Structural Model

# Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP).

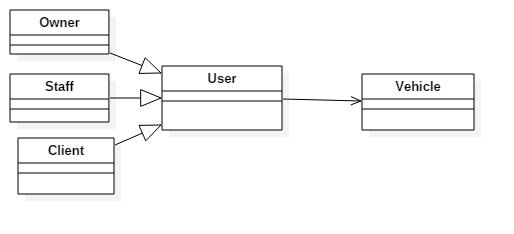


Figure 15:Class Diagram

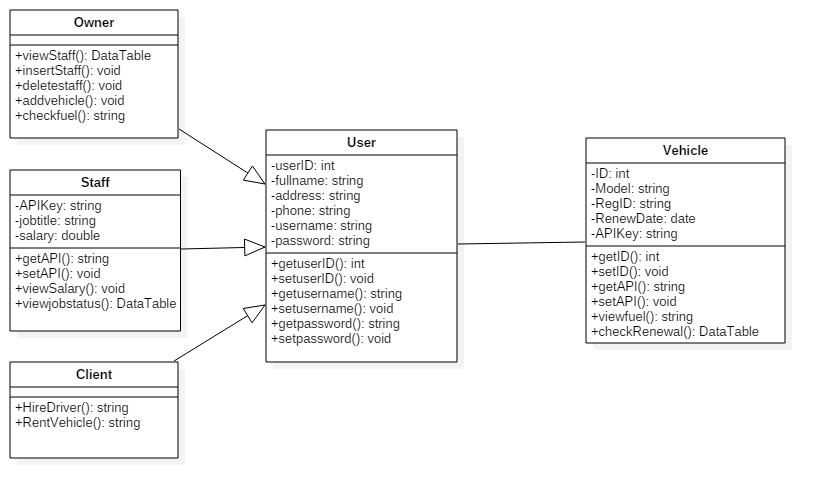


Figure :Final Class Diagram

## Justification

### Advantages

* It forces the programmer to think out the structure of his/her classes and how they will interact with each other before actually writing any code. This may lead to a more robust application.
* It provides a blueprint for maintenance programmers to get an overview of how the application is structured before examining the actual code. This may reduce maintenance time.

### Disadvantages

* The programmer may need to learn UML to build the class diagram in the first place.
* The time spent building the class diagram may add to overall development time.
* If the class diagram is overcomplicated, then it may be difficult to correlate with the actual code.

# ER Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.ER diagrams are used to sketch out the design of a database.

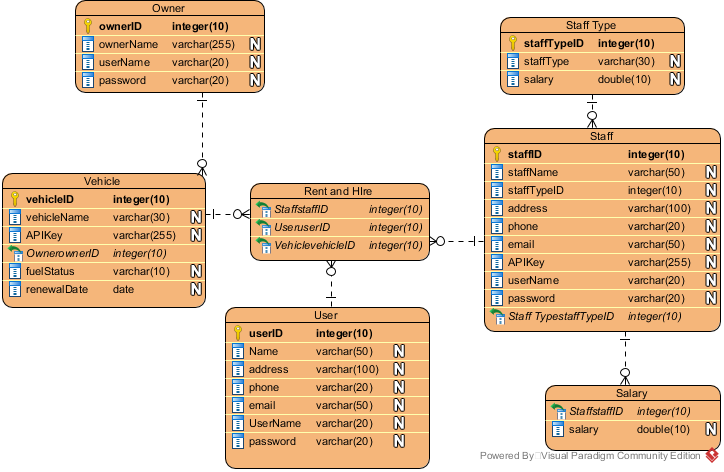


Figure 17: ER diagram

## Justification

### Advantages

* Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.
* Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.
* Effective communication tool: It is an effective communication tool for database designer.
* Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.
* Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

### Disadvantages

* Limited constraints and specification
* Loss of information content: Some information be lost or hidden in ER model
* Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.
* No representation of data manipulation: It is difficult to show data manipulation in ER model.
* Popular for high level design: ER model is very popular for designing high level design
* No industry standard for notation

# Behavioral model

# Activity Diagram

Activity diagram is another important diagram in UML 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.

## Vehicle Rental

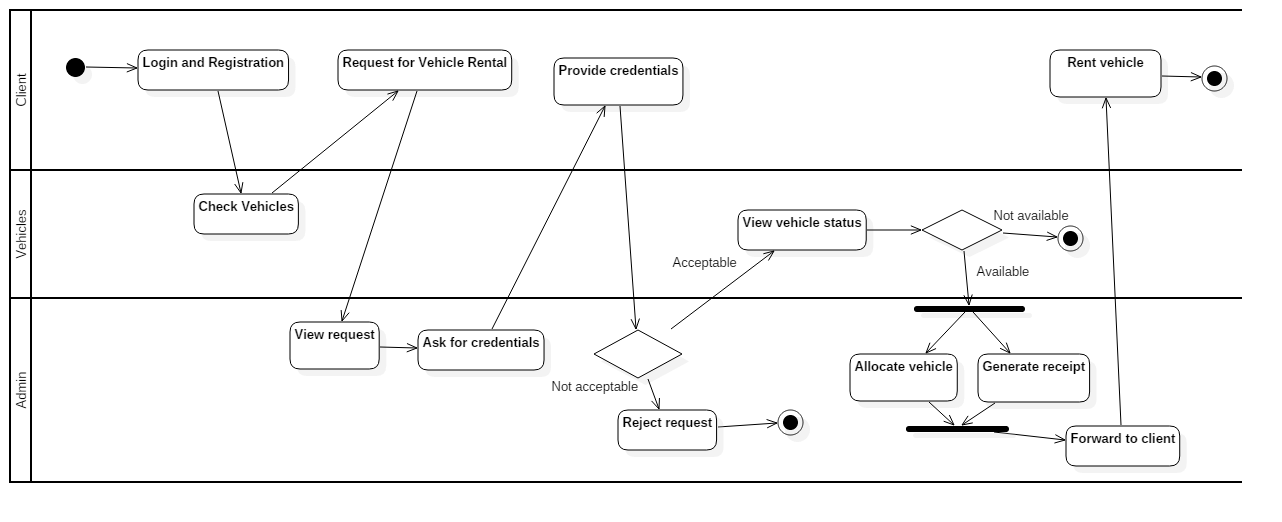


Figure :Activity Diagram for Rental

## Driver Hire

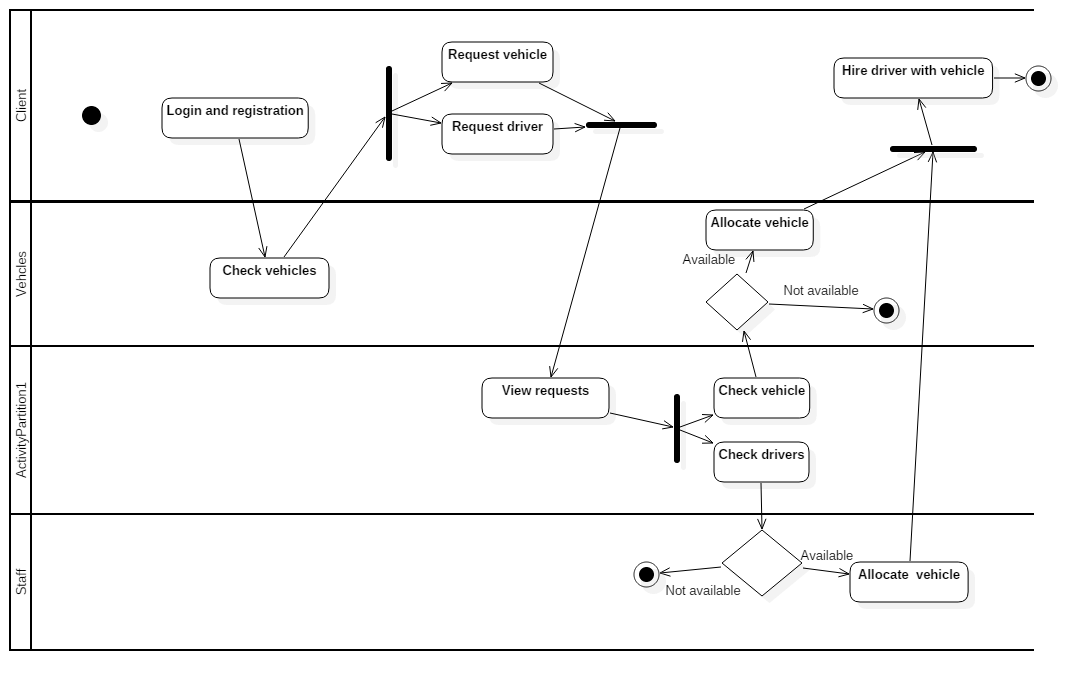


Figure : Hire driver with vehicle

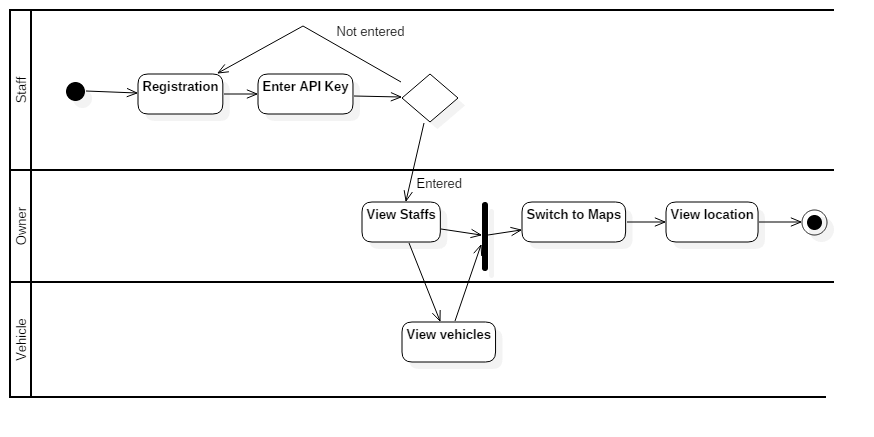


Figure : View location

## Justification

### Advantages

* UML modeling language included that these diagrams are normally easily comprehensible for both analysts and stakeholders.
* Since they are among the most user-friendly diagrams available, they are generally regarded as an essential tool in an analyst’s repertoire.
* Additionally, as stated above, activity diagrams allow an analyst to display multiple conditions and actors within a work flow through the use of swim-lanes. Swim-lanes, however, are optional as a single condition or actor is normally displayed without them.

### Disadvantages

* UML modeling language include that activity diagrams have the potential to become overly complex because their user-friendly nature may lend itself to an all-inclusive description.
* Another aspect of activity diagram is that they may not be used in lieu of a state diagram or sequence diagram because activity diagrams do not give detail about how objects behave or how objects collaborate.

# Sequence Diagram

The object interaction and the explanation of message forwarding and interaction is explained in a sequence termed as sequence diagram. The sequence for vehicle rental is explained as:

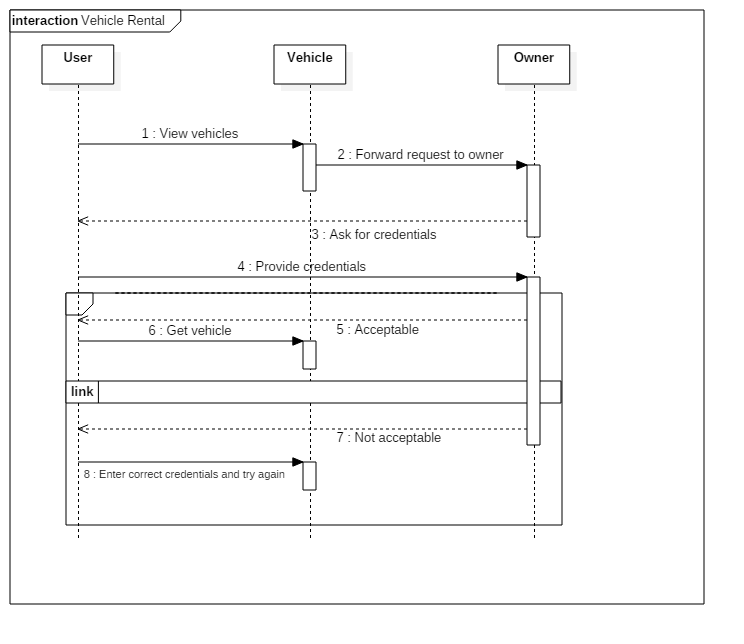


Figure : Rental sequence

If the client wishes for driver hiring, the sequence is explained as:

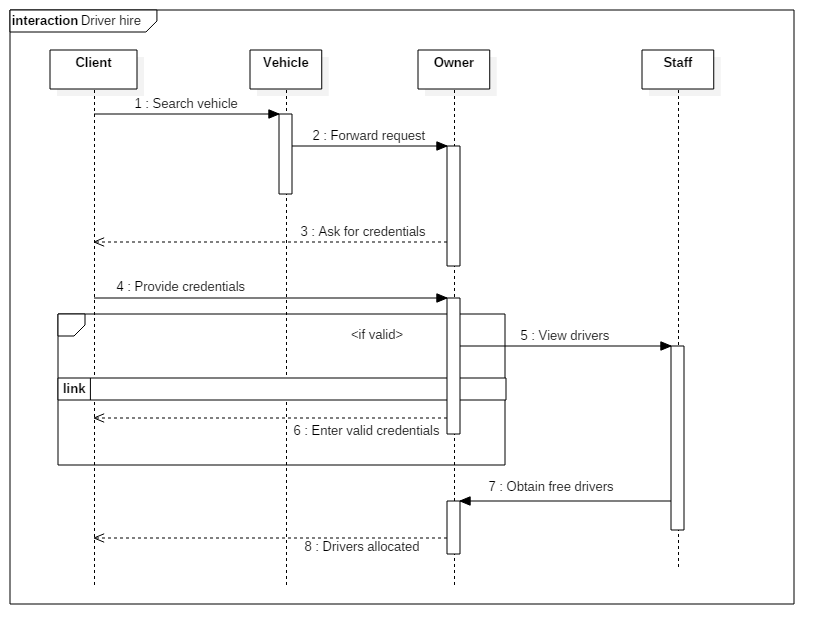


Figure : Drive hire

The owner can view their vehicle upon adding API key and the location derive sequence is explained as:

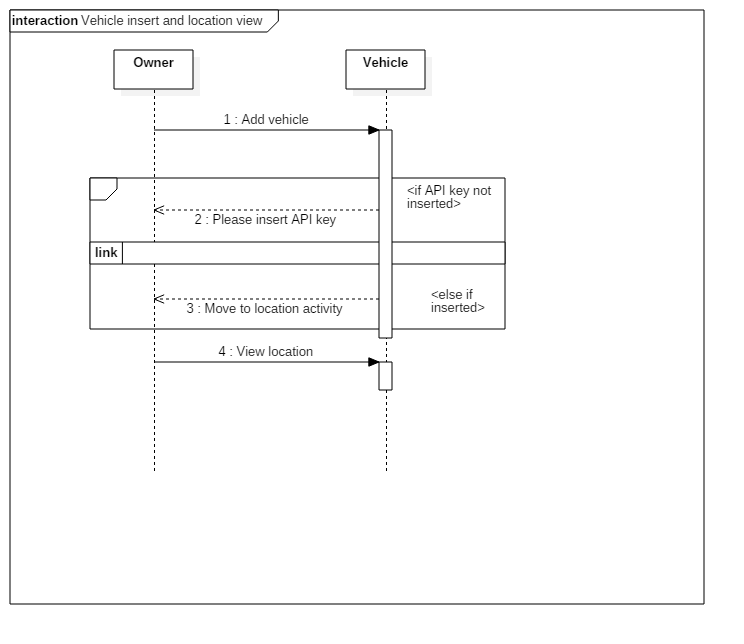


Figure : Viewing vehicle location

# 

# UI design

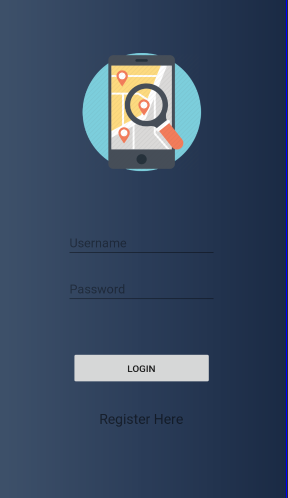


Figure : Login form

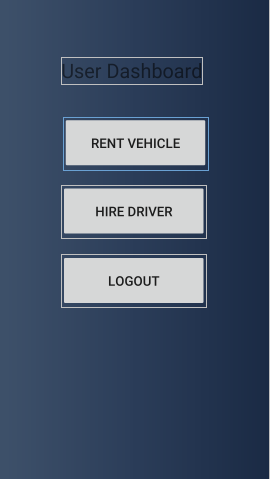


Figure : User dashboard

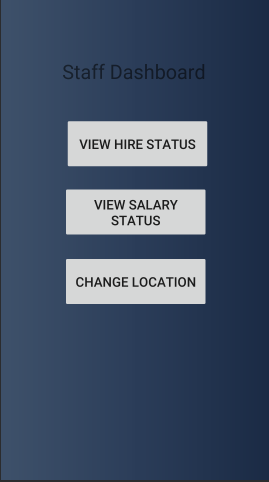


Figure : Staff dashboard

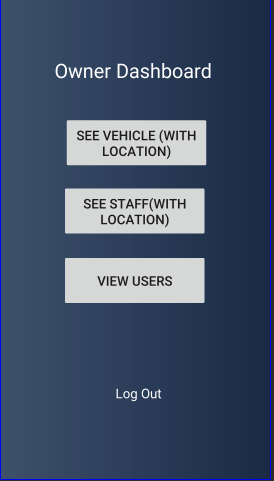


Figure : Owner dashboard

# Chapter-8: Testing

## Unit Testing

Testing of software functionality and its accuracy in a code level is termed as unit testing. It is essential to check whether the code (component of software) gives us a desired output or not. We view the expected and actual output for the test.

Test Case-1:

|  |  |
| --- | --- |
| **Test number:** | 1 |
| **Purpose of Test:** | Viewing if accurate model number is returned or not. |
| **Test Data:** | FORD-56940 |
| **Test Name:** | modelNo( ) |
| **Expected Result:** | FORD-56940 |
| **Actual Result:** | FORD-56490 |
| **Test Status:** | Pass |

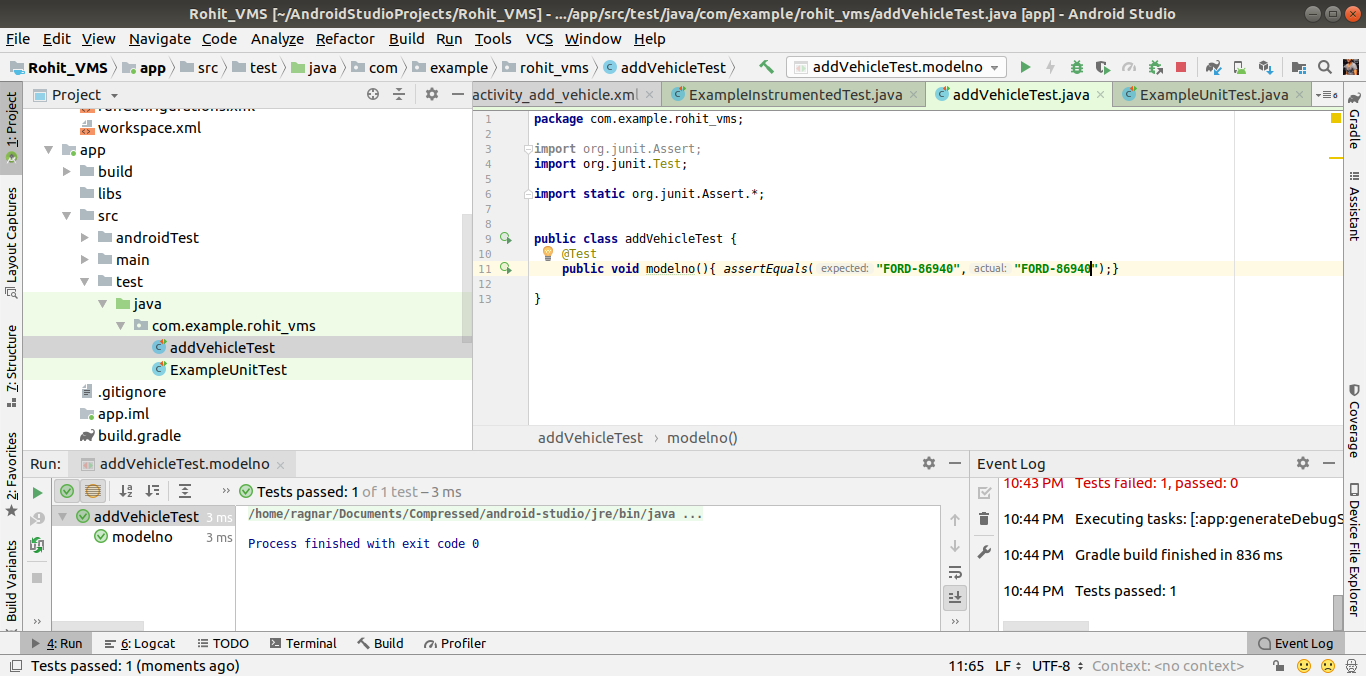


Figure : Model number test

Test Case-2:

|  |  |
| --- | --- |
| **Test number:** | 2 |
| **Purpose of Test:** | Viewing if accurate email in the expected format is returned or not. |
| **Test Data:** | [cool.thedevilrohit@gmail.com](mailto:cool.thedevilrohit@gmail.com) |
| **Test Name:** | emailFormat( ) |
| **Expected Result:** | [cool.thedevilrohit@gmail.com](mailto:cool.thedevilrohit@gmail.com) |
| **Actual Result:** | [cool.thedevilrohit@gmail.com](mailto:cool.thedevilrohit@gmail.com) |
| **Test Status:** | Pass |

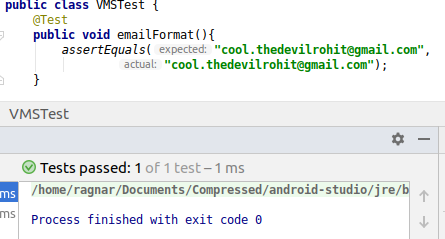


Figure : Email format test

Test Case-3:

|  |  |
| --- | --- |
| **Test number:** | 3 |
| **Purpose of Test:** | When accurate email format is not returned |
| **Test Data:** | [cool.thedevilrohit@gmailcom](mailto:cool.thedevilrohit@gmailcom) |
| **Test Name:** | emailFormat( ) |
| **Expected Result:** | [cool.thedevilrohit@gmail.com](mailto:cool.thedevilrohit@gmail.com) |
| **Actual Result:** | [cool.thedevilrohit@gmailcom](mailto:cool.thedevilrohit@gmailcom) |
| **Test Status:** | Fail |



Figure :Test when accurate result isn’t returned

Test Case-4:

|  |  |
| --- | --- |
| **Test number:** | 4 |
| **Purpose of Test:** | Checking firebase connection |
| **Test Data:** | DatabaseReference |
| **Test Name:** | CheckFirebase( ) |
| **Expected Result:** | Connection fail, exception thrown |
| **Actual Result:** | Connection fail, exception thrown |
| **Test Status:** | Pass |

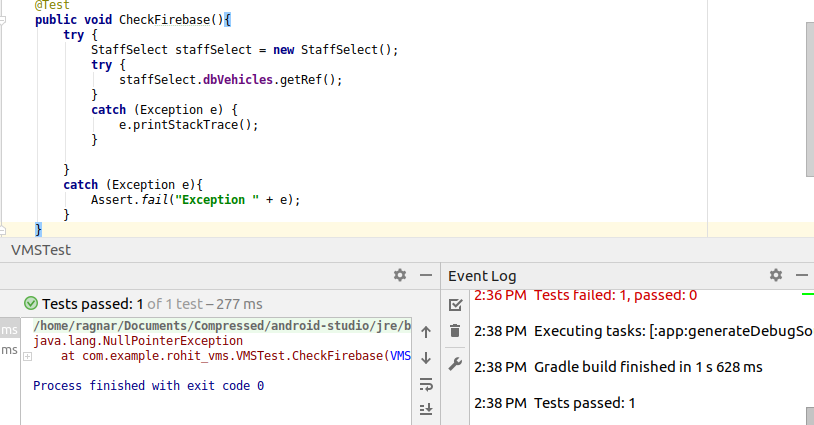


Figure : Firebase connection failed

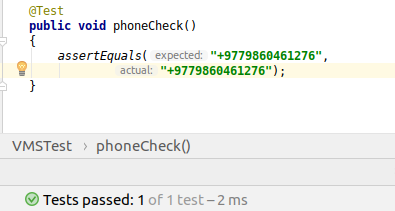
Test Case-5:

|  |  |
| --- | --- |
| **Test number:** | 5 |
| **Purpose of Test:** | Phone number check |
| **Test Data:** | 9860461276 |
| **Test Name:** | phoneCheck ( ) |
| **Expected Result:** | +9779860461276 |
| **Actual Result:** | 9860461276 |
| **Test Status:** | Fail |



Test Case-6:

|  |  |
| --- | --- |
| **Test number:** | 6 |
| **Purpose of Test:** | Phone number check |
| **Test Data:** | +9779860461276 |
| **Test Name:** | phoneCheck ( ) |
| **Expected Result:** | +9779860461276 |
| **Actual Result:** | +9779860461276 |
| **Test Status:** | Pass |



## Black Box testing

Without actually viewing the specific code level implementation of the application, the tester performs a test of the functionality of an application termed as black box testing. (SoftwareTestingFundamentals,2013)

Test Case-11:

|  |  |
| --- | --- |
| **Test number:** | 11 |
| **Purpose of Test:** | Login with incorrect email |
| **Test Data:** | [suveksha01@gmail.co](mailto:suveksha01@gmail.co) |
| **Test Name:** | Test Email |
| **Expected Result:** | Success and move to home page |
| **Actual Result:** | Thrown at same page with Toast message |
| **Test Status:** | Fail |

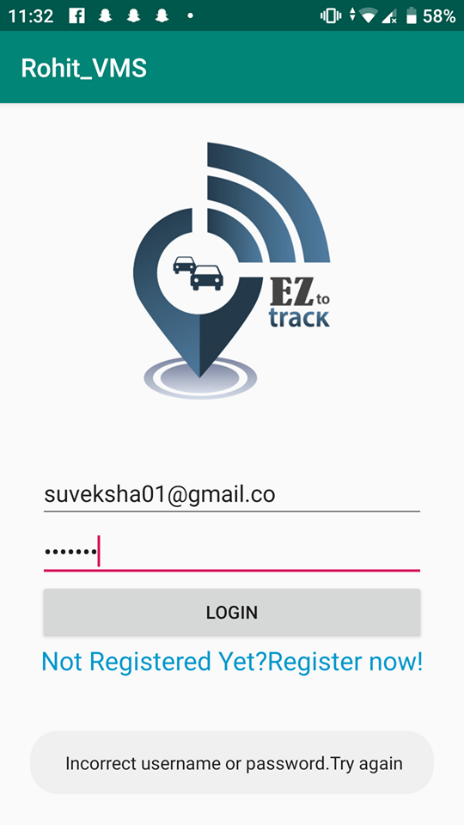


Figure : Email password validation

Test Case-12:

|  |  |
| --- | --- |
| **Test number:** | 12 |
| **Purpose of Test:** | Valid credential enter and login |
| **Test Data:** | [suveksha01@gmail.com](mailto:suveksha01@gmail.com) |
| **Test Name:** | Test Email |
| **Expected Result:** | Success and move to home page |
| **Actual Result:** | Success and move to home page |
| **Test Status:** | Pass |

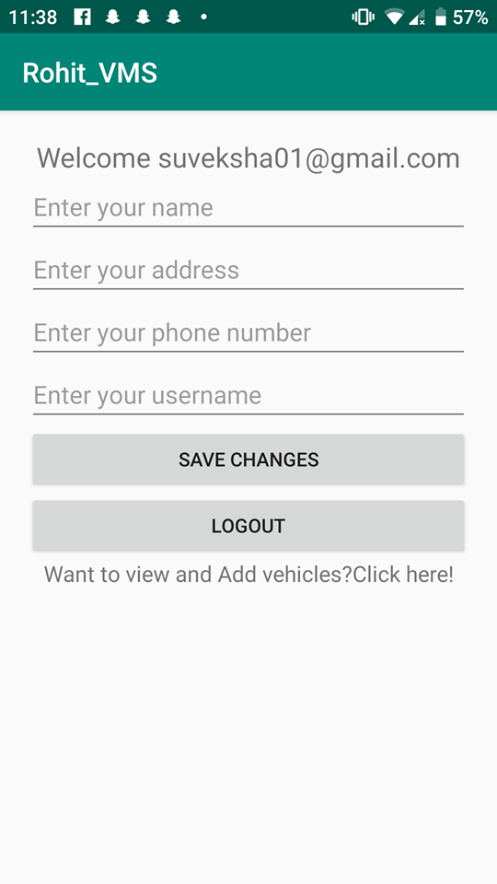


Figure : Login success

Test Case-13:

|  |  |
| --- | --- |
| **Test number:** | 13 |
| **Purpose of Test:** | Display message on leaving fields empty |
| **Test Data:** |  |
| **Test Name:** | Test Empty field |
| **Expected Result:** | Message display |
| **Actual Result:** | Message displayed |
| **Test Status:** | Pass |

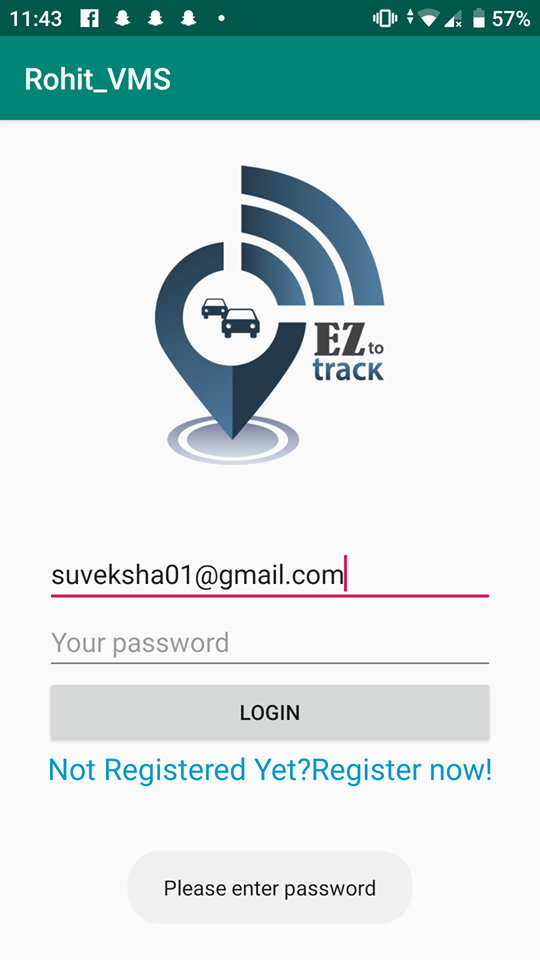


Figure : Displaying message on empty field

Test Case-14:

|  |  |
| --- | --- |
| **Test number:** | 14 |
| **Purpose of Test:** | Update values |
| **Test Data:** | “Suveksha Khanal”, “Sinamangal”,  “9803562398”, “suveksha1” |
| **Test Name:** | Update test |
| **Expected Result:** | Values updated |
| **Actual Result:** | Values updated |
| **Test Status:** | Pass |

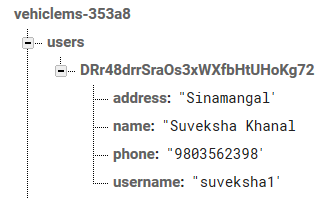
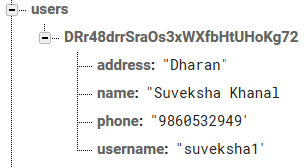
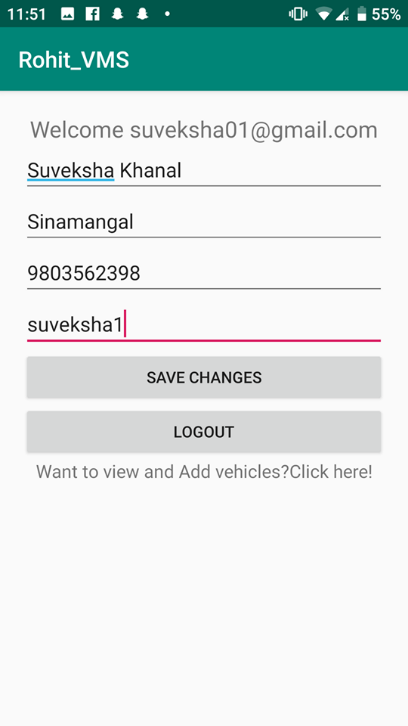


Figure : Update test

Test Case-15:

|  |  |
| --- | --- |
| **Test number:** | 15 |
| **Purpose of Test:** | Add vehicles |
| **Test Data:** | “TATA-98”, “Tata”, “42”, “32000”, …. |
| **Test Name:** | Add vehicles test |
| **Expected Result:** | Vehicle added |
| **Actual Result:** | Vehicle added |
| **Test Status:** | Pass |

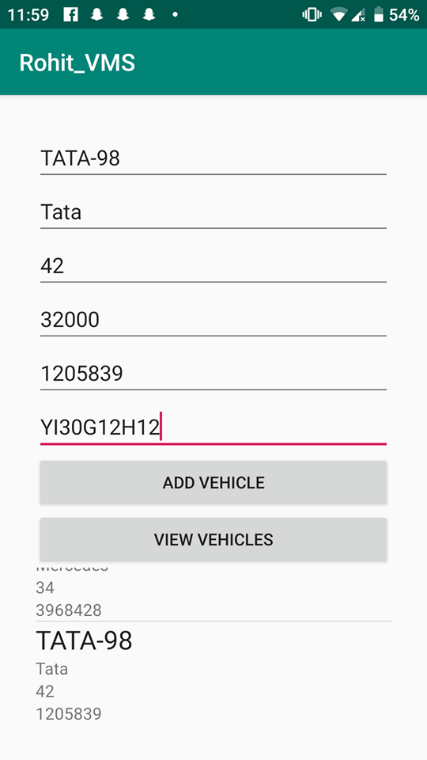
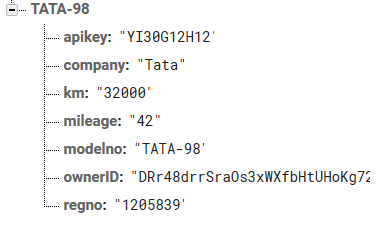
 

Figure : Add test

Test Case-16:

|  |  |
| --- | --- |
| **Test number:** | 16 |
| **Purpose of Test:** | Registration without credentials |
| **Test Data:** | “Rohit” |
| **Test Name:** | Breach without credentials |
| **Expected Result:** | Successful login |
| **Actual Result:** | Thrown back to registration |
| **Test Status:** | Fail |

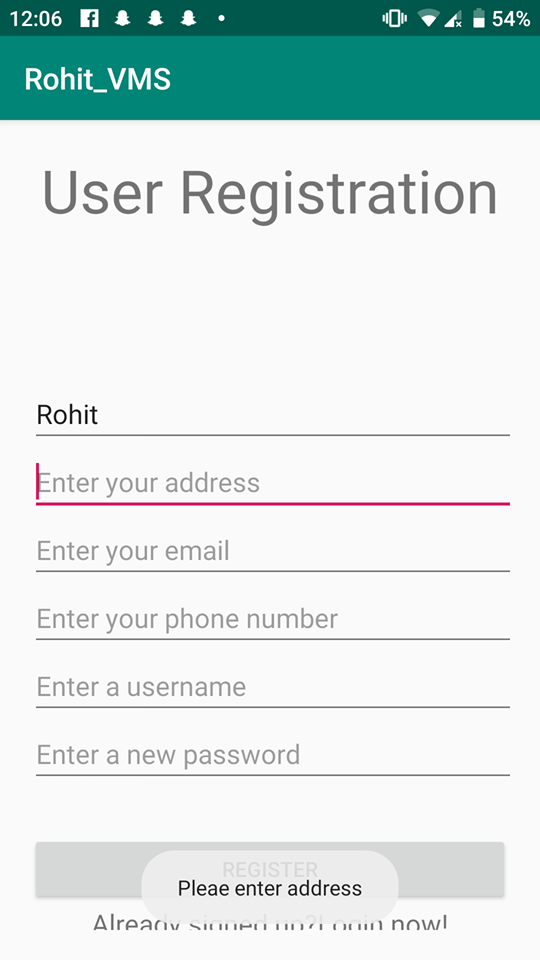


Figure : Login without credentials test

Test Case-17:

|  |  |
| --- | --- |
| **Test number:** | 17 |
| **Purpose of Test:** | Registration without credentials |
| **Test Data:** | “Sajan”, ”Jorpati”, [sajandangal1@gmail.com](mailto:sajandangal1@gmail.com),  “sajan1”, “\*\*\*\*\*\*” |
| **Test Name:** | Signup with credential |
| **Expected Result:** | Successful login and registration |
| **Actual Result:** | Successful login and registration |
| **Test Status:** | Pass |

# 

Figure : Registration with credentials

Test Case-18:

|  |  |
| --- | --- |
| **Test number:** | 18 |
| **Purpose of Test:** | Logout |
| **Test Data:** |  |
| **Test Name:** | Logout |
| **Expected Result:** | End session, logout |
| **Actual Result:** | End session, logged out |
| **Test Status:** | Pass |

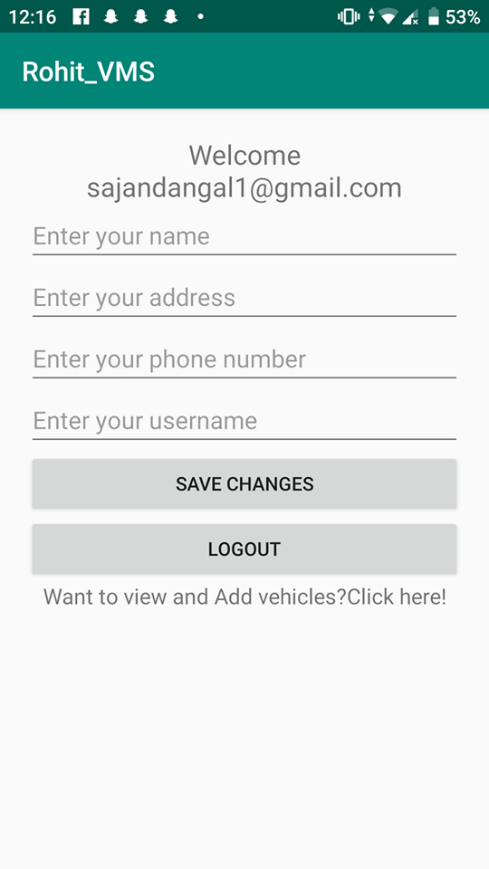
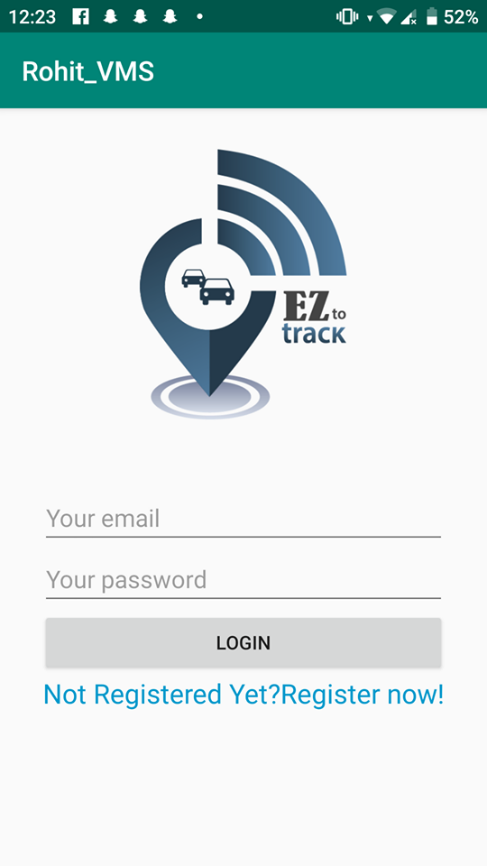
 

Figure : Logout test

# Chapter-9: Conclusion

Hence, we would be able to develop a project for monitoring and tracking our vehicle/s and create a documentation based on it as per above steps. Our application would somewhat assure security of our vehicle and handover to only trusted individuals. Also, necessary information about payments, reconditioning, servicing and deadlines are notified to owners along with suspicious activities. Data management is made easier and database handles it efficiently. Also, the final report would address the overall project such that any user can understand it in details.

# Chapter-10: References and Bibliography

* Scope of Project: https://totally.tech/how-to-define-the-scope-of-a-project/
* Waterfall model: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm
* Design pattern: https://www.geeksforgeeks.org/mvc-design-pattern/
* Scheduling: https://www.projectmanager.com/blog/what-is-project-scheduling
* Risk Management: <https://pm4id.org/chapter/11-2-risk-management-process/>

# Chapter-11: Appendix

## Login:

Firebase real-time database has been used for authentication and storage purposes. A reference is essential for identification of specific field here and due to the tree structure, we deal with nodes here. The specific nodes and values are called accordingly and cross verified with the user entered data. Since the verification takes a couple of seconds, a progress dialog is played accordingly.

# 

Figure : Creating instance of Auth



Figure : Specifying view fields



Figure :Logic for email/password verification

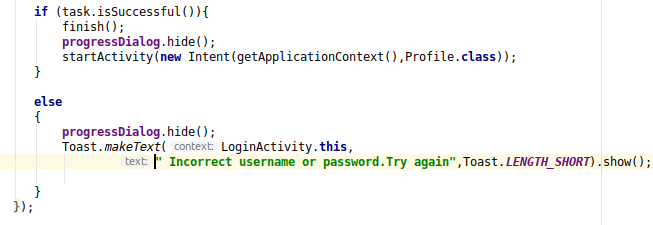


Figure : Success and failure

## Registration

Authentication is provided separately by firebase involving email and password registration. The user can view and proceed for hire or rental activities and only during confirmation, their credentials are asked. Hence for registration, firebase authentication supports us and if fields are left empty, toast is displayed as per the fields.

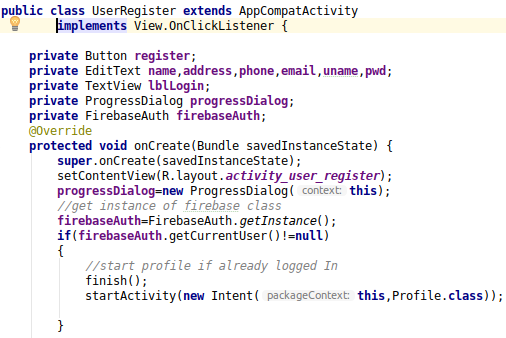


Figure : Calling firebaseauth instance



Figure : Associating fields and button events

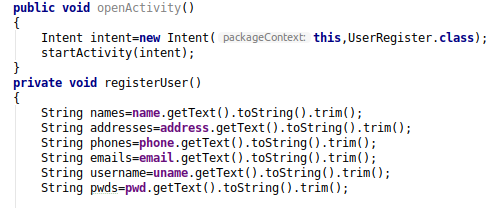
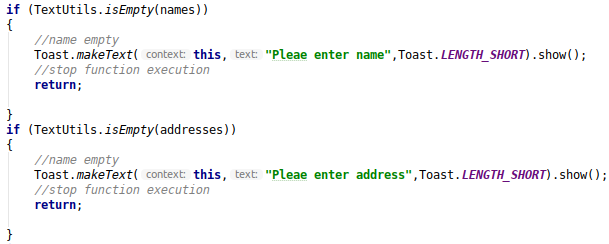
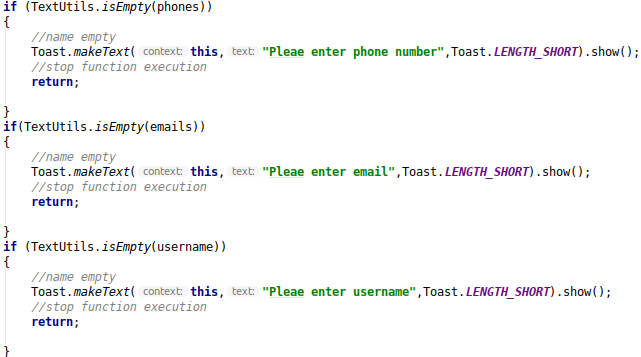


Figure : Initializing model attributes





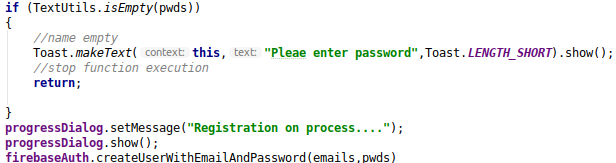


Figure : Verifying values and registration

## Add vehicles

As an owner, we can add vehicles and the vehicles are displayed to users accordingly. The Vehicle node is added with the associated vehicle. Also the owner can view the vehicle obtained through DataAdapter containing list of vehicles while he adds them.

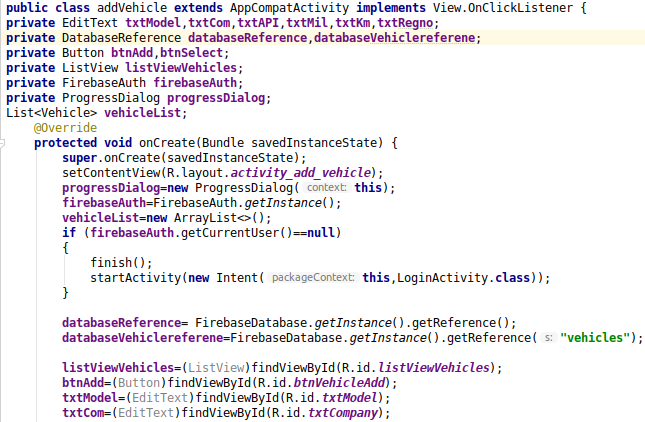


Figure : Creating reference of database



Figure : Retrieval of list



Figure : Adding vehicles

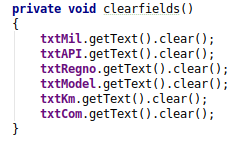


Figure : Clearing fields after add

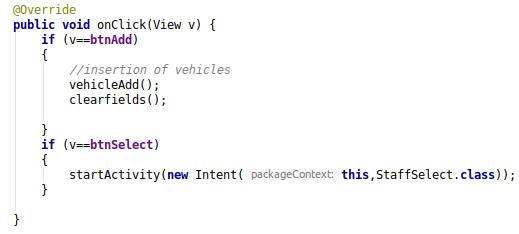


Figure :Associating button with events