***PACKER AND MOVER***

***A project report submitted to Kannur University***

***In partial fulfilment of requirements for the award of***

**MASTER**

***Of***

**COMPUTER APPLICATION**

***By***

**JITHU K PAVITHRAN**

**REG. NO: C0GMCA2105**

**(2020-2022)**



**DEPARTMENT OF COMPUTER APPLICATIONS**

**DONBOSCO COLLEGE**

**ANGADIKADAVU, KANNUR**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**DONBOSCO COLLEGE**

**ANGADIKADAVU, KANNUR**

****

**CERTIFICATE**

This is to certify that the report of the project entitled **“PACKER AND MOVER”** is a bonafide record of the original work done by **JITHU K PAVITHRAN** (**Reg.No:C0GMCA2105)** during the **fourth semester** of the year **2020-2022** in partial fulfilment of the requirements for the award of Master of Computer Applications under the Kannur university.

**Internal Guide: Date:**

**1:**

**External Guide: Head of the institution**

**1:**

**2:**

**DECLARATION**

I, JITHU K PAVITHRAN, fourth semester MCA, student of Don Bosco College, Angadikadavu, under Kannur University do hereby declare that the project entitled “PACKER AND MOVER” is the record of original work done by me under the supervision of Mr. Kevinson Kurian, HoD, Department of MCA of Don Bosco College, Angadikadavu towards partial fulfilment of the requirement of Master’s degree in Computer Applications, and no part thereof has been presented for the award of any other degree.

Date: JITHU K PAVITHRAN

Reg No: C0GMCA2105,

Dept of MCA,

Don Bosco College,

Angadikadavu, Kannur

**ACKNOWLEDGEMENT**

First of all, I thank the lord Almighty for his immense grace and blessings showered on meat every stage of this work. I am greatly indebted to our Director Fr. Fidel Thachil SDB, Don Bosco College, Angadikadavu for providing the opportunity to take up this project as part of my curriculum.

I express my sincere thanks to Mr. Kevinson Kurian (HOD, Dept. of MCA, Don Bosco College), and other faculties, management and staff of Don Bosco College for the support throughout my MCA course.

Acknowledging the efforts of everyone, their chivalrous help in the course of the project preparation and their willingness to corroborate with the work, their magnanimity through lucid technical details lead to the successful completion of my project.

I would like to express my sincere thanks to all my friends, colleagues, parents and all those who have directly or indirectly assisted during this work.

JITHU K PAVITHRAN

**ABSTRACT**

Packer and Mover is a project which is developed to provide an interactive platform between clients and packer mover company. This project provides best and reliable services in relocating. Clients can book the services through mobile applications. This project provides useful information to clients in the process of relocating their house. Packer and Mover agency works according to the needs and requirement of the customer and provide them with the desirable results. The packer and Mover agencies us best and quality packing materials to pack our goods in such a way that all goods remain in safe condition during transit moving services assure the safe delivery of our goods at our destination. In a new place we cannot find a good freight worker at a short period of time. But in this freight worker can register to the app and all user can view the available worker and contact them for our works

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| Chapters | Contents | Page No |
| **1** | **INTRODUCTION** |  |
| 1.0 | Project Review |  |
| 1.1 | Scope of the Project |  |
| **2** | **SYSTEM ANALYSIS** |  |
| 2.0 | Requirement Analysis |  |
| 2.1 | Existing System |  |
| 2.2 | Proposed System |  |
| 2.3 | Feasibility Study |  |
| 2.3.1 | Economic Feasibility |  |
| 2.3.2 | Technical Feasibility |  |
| 2.3.3 | Behavioral Feasibility |  |
| 2.3.4 | Legal Feasibility |  |
| 2.4 | System Requirement Specification |  |
| 2.4.1 | Actor Identification |  |
| 2.4.2 | Use Case Identification |  |
| 2.4.2.1 | Use Case Diagrams |  |
| 2.4.3 | Activity Diagrams |  |
| 2.4.4 | Sequence Diagram |  |
| 2.5 | System Requirements |  |
| 2.5.1 | Hardware and Software Requirements |  |
| **3** | **SYSTEM DESIGN** |  |
| 3.1 | Database Design |  |
| 3.1.1 | Data Flow Diagrams |  |
| 3.1.2 | ER Diagrams |  |
| 3.1.3 | Table Design |  |
| 3.2 | Architecture Design |  |
| 3.2.2 | Hierarchical Diagram |  |
| 3.3 | Interface Design |  |
| 3.3.1 | Input Design |  |
| 3.3.2 | Output Design |  |
| 3.4 | Procedural Design |  |
| **4** | **CODING** |  |
| 4.1 | About Software Tools used |  |
| 4.1.1 | Python |  |
| 4.1.2 | SQLite |  |
| 4.2 | Coding Principle |  |
| 4.3 | Sample Code |  |
| **5** | **TESTING** |  |
| 5.0 | System Testing |  |
| 5.1 | Unit Testing |  |
| 5.2 | Integration Testing |  |
| 5.3 | System Testing |  |
| 5.4 | Validation Testing |  |
| **6** | **IMPLIMENTATION** |  |
| 6.1 | System Implementation |  |
| **7** | **CONCLUSION** |  |
| 7.1 | conclusion |  |
| **8** | **REFERENCES** |  |
| 8.1 | Websites |  |
| 8.2 | Reference Books |  |
| **9** | **APPENDIX** |  |

**CHAPTER-1**

**INTRODUCTION**

1.INTRODUCTION

**1.0 PROJECT OVERVIEW**

Packer and mover is software which provide an interactive platform between users and packer and mover agencies. User can book the services through this web portal. We do not have to maintain everything manually. Through this system if any enquiry occurs it is corresponding entries is done automatically because database management system gives of having relationship between the tables. In proposed system we do not have to maintain record manually. The Major Objectives of System are:

1. Man power required is less

2. Data can be stored for a long period

3. It is very much flexible to work

Main actors of this project are the Admin, User, Driver, Freight Worker. Admin can control Driver and Worker. Users can request services and workers. Driver can view assigned work and reschedule the work. Freight worker can view request and update status.

This project is developed by using Python (Flask) as the programming language, and MySQL as the database.

**1.1 SCOPE OF THE PROJECT**

Packers and Movers is an online platform for service seekers and service providers where all the companies are available at single and they do communicate directly with service seekers. Our packers and mover’s directory is having ultimate objective of providing information to its visitors about best packers movers and relocation companies offering its services in Indian destinations. The reliable and swift packers and movers’ services providers are the best for any kind of relocation and car transportation services. There are many packers and movers operating in India but all of them are not well established and experienced companies. It becomes difficult for you to choose best and reliable packers and movers in your city when you need to relocate your house, office and to provide packing moving services with full responsibility and reliability. The listed companies will provide door to door services within an appropriate time. Their professional management takes care of your every single need. Now Packers and Movers also provide storage and warehousing storage services providers at reasonable rate with full safety and security. Goods can be store in stock room for short and long duration. The warehouse has an absolute security. These packers and movers operate with full efficiency and accuracy

**CHAPTER-2**

**SYSTEM ANALYSIS**

**2. SYSTEM ANALYSIS**

System analysis is the process of collecting and interpreting facts, understanding problems, and using the information to suggest improvements on the system. This will help to understand the existing system and determine how computers make its operation more effective. The aim of this analysis is to collect the detailed information on the system and the feasibility study of the proposed system. This analysis focuses on the flow of the system module by module and the efficiency of each. To design the proposed system, we need the exact processing logic as well as the extended features of the existing system such as reliability, consistency, storage capacity etc. This report will discuss the advantages and drawbacks/ disadvantages of the existing system and the modifications and enhancements can be done. This analysis will concentrate on the information gathering for the efficient, user friendly and reliable system, which will carry forward the features of the existing system.

**2.0 REQUIREMENT ANALYSIS**

Requirements analysis results in the specification of software’s operational characteristics, indicates software’s interface with other system elements, and establishes constraints that software must meet. Requirements analysis allows you to elaborate on basic requirements established during the inception, elicitation, and negotiation tasks that are part of Requirements engineering.

**REQUIREMENT GATHERING**

The requirement gathering can be done by following ways.

* Interview
* Questionnaire
  + Website visit

For this project I used website visit and interview method. I visited the following resources.

* I visited a few Packer & Mover shopping Websites. These sites are given below:
* [www.nobroker.in](https://r.search.yahoo.com/_ylt=Awrxgvm1BbNjHAcATA_nHgx.;_ylu=Y29sbwMEcG9zAzMEdnRpZAMEc2VjA3Ny/RV=2/RE=1672705589/RO=10/RU=https%3a%2f%2fwww.nobroker.in%2fpackers-and-movers/RK=2/RS=fUoXvWE5dkIhHm0liP2dQ__Dgzo-)
* [www.agarwalpackers.com](http://www.agarwalpackers.com)
* The interview method helped to collect more information from some shops

**2.1 EXISTING SYSTEM**

In the existing system shifting goods and households is that either to take all the goods or to leave some of it or to sold them out. while relocating most of the goods get damaged and it takes lot of risk. It is seen that there are many agencies are working for this and all of these agencies has their own websites to give their service information and a user has to visit to individual sites, to overcome this problem we have designed a web portal so that all the companies register over it and user get the information on one single site only.

.

**2.2 PROPOSED SYSTEM**

Packers and movers is an online platform for service seekers and service providers where all the companies are available at single site as web portal and they do communicate directly with service seekers. In Packers and movers, we have listed excellent packing moving service providers of India, household shifting and relocation services providers, car transportation, office relocation, home, shop, industrial or commercial shifting service providers of India. Our packers and mover’s directory is having ultimate objective of providing information to its visitors about best packers movers and relocation companies offering its services in Indian destinations.

Advantages

* This project provides best and reliable services in relocating.
* Flexibility for users.
* Data can be stored for a longer period
* Man power required is very less

**2.3 Feasibility Study**

Feasibility analysis is the procedure for identifying the candidate system, evaluating and electing the most feasible system. When conducted feasibility study understand the need for change or improvement in the current system, which is manually. This is done by investigating the existing system. A feasibility study is conducted to check whether it is

* Possible (to build with the given technology and resources)
* Affordable (given time and cost constraints of the organizations)
* Acceptable

Basically, feasibility study tries to find out whether it is worth developing a new system before proceeding to developing it. Certain key considerations are involved in the feasibility analysis are:

* Economic Feasibility
* Technical Feasibility
* Behavioral Feasibility
* Legal Feasibility

**2.3.1 Economic Feasibility**

Economic Feasibility is cost-benefit analysis various benefits and costs involved are considered, calculated, and compared, if the benefits are more than the cost, the project is considered economically feasible. It also makes stresses whether the system can be built in the specified time interval. Today everyone using computers and familiar with all the functionalities. We only want to pay for the net connection. Now it is common for all. And the benefits will be more. So, the proposed system will replace the cost and man power involved in the existing system. thus, the system is economically feasible.

**2.3.2 Technical Feasibility**

The Technical Feasibility is the concept that deals with the hardware as well as software requirements. Through the technology may become obsolete after some period, because newer version of some software supports older versions, the system may still be used. So, there are only minimal constraints involved with this project. The Packer and Mover can be used by anyone without knowing the language python. Anyone can be access the system if they have simple browsing experience, so they can easily use this software. It does not have any operational barriers. People don’t want to know the framework used by this system. By seeing the interface, itself, any user can easily identify the functions and access the system. So, the system is technically feasible.

**2.3.3 Behavioral Feasibility**

The “Packer and Mover” is designed in a user-friendly manner and we need not to provide any special training for the persons using this software. Those who have simple browsing experience, they can easily use this software. It does not have any operational barriers, so there is no need to provide any special training for using this software. Hence it is behaviorally feasible.

**2.3.4 Legal Feasibility**

The system cannot create any violation of rules and regulation of government. It is not making any security issues to outside world. So, the system is legally feasible.

**2.4 SYSTEM REQUIREMENT SPECIFICATION**

System requirements are expressed in a software requirement document. The Software requirement specification (SRS) is the official statement of what is required for the system developers. This requirement document includes the requirements definition and the requirement specification. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent. The software specification document satisfies the following: -

* It specifies the external system behavior.
* It specifies constraints on the implementation.
* It serves as reference tool for system maintainers.
* It records forethought about the life cycle of the system.
* It characterizes acceptable response to undesired events.

**2.4.1. ACTOR IDENTIFICATION**

An actor is someone or something that interacts with the system. An actor is he /she what uses the system. An actor exchanges information with the system. Asking certain questions as detailed below can identify the actors of the system.

|  |  |  |
| --- | --- | --- |
| **1.** | Who will use the main functionality of the system? | Administrator, Driver, Freight Worker, User |
| **2.** | Who will lead support from the system and do their tasks? | Administrator, Driver, Freight Worker, User |
| **3.** | Who will maintain and administrate the system? | Administrator. |
| **4.** | With which other systems, does this system need to interact? | Database. |
| **5.** | Who was interest in the result produced by the system? | Administrator, Driver, Freight Worker, User |

As per the above answers we can conclude the actors. They are:

* Admin
* Driver
* Freight Worker
* User

**2.4.2. USE CASE IDENTIFICATION**

A use cases represents the functionality of an actor. It is defined as a set of actions performed by a system, which yields an observable result. An ellipse containing its name inside the ellipse or below it represents it. It is placed inside the system boundary and connected to an actor with an association. This shows how the use cases and the actor interact.

To find out the use cases, ask the following questions to each of the actors.

* Which functions does the actor require from the system? What does the actor need to do?
* Does the actor need to read, create, destroy, modify or store some kind of information in the system?
* Could the actor’s daily work be simplified or made more efficient by adding new functions to the system?

**2.4.2.1 USE CASES**

**Use case for the actor Administrator**

|  |  |  |
| --- | --- | --- |
| **1** | Which functions does the Administrator require from the system? What does the Administrator need to do? | Administrator requires the following functionalities from the system such as Add and manage driver, vehicle, Item, View request and update status, assign work and Vehicle to Driver, View bill information, View rescheduled request |
| **2** | Does the Administrator need to read, create, destroy, modify or store some kind of information in the system? | Yes. Administrator need to create, view and edit the data if require. |
| **3** | Could the Administrator work be simplified by adding new functions to the system? | Yes, the system can reduce his/her work. |

Above questions give the following use cases for the actor Administrator.

* Login
* Add and manage driver
* Approve freight worker
* Add and manage vehicle
* Add and manage item
* View request and update status
* Assign work and vehicle to driver
* View drivers rating and freight workers rating
* Block driver and freight workers
* View bill information
* View rescheduled request
* Logout

**Use case for the actor Driver**

|  |  |  |
| --- | --- | --- |
| **1** | Which functions does the Spare part shop require from the system? What does the Driver need to do? | Packer and mover requires the following functionalities from the system such as View assigned works and update status, Rescheduled request, view rescheduled work, Update location, Generate Bill |
| **2** | Does the Spare part shop need to read, create, destroy, modify or store some kind of information in the system? | Yes. Packer and Mover need to create, view and edit the data if require. |
| **3** | Could the Spare part shop work be simplified by adding new functions to the system? | Yes, the system can reduce his/her work. |

Above questions give the following use cases for the actor Driver.

* Login
* View assigned works and update status
* Rescheduled request
* View rescheduled work
* Update location
* Generate Bill
* Logout

**Use case for the actor Freight Worker**

|  |  |  |
| --- | --- | --- |
| **1** | Which functions does the workshop require from the system? What does the workshop need to do? | Workshop requires the following functionalities from the system such Register, Login, View request and update status, View Payment, and Logout |
| **2** | Does the workshop need to read, create, destroy, modify or store some kind of information in the system? | Yes. Freight Worker need to create, view and edit the data if require. |
| **3** | Could the workshop work be simplified by adding new functions to the system? | Yes, the system can reduce his/her work. |

Above questions give the following use cases for the actor freight worker.

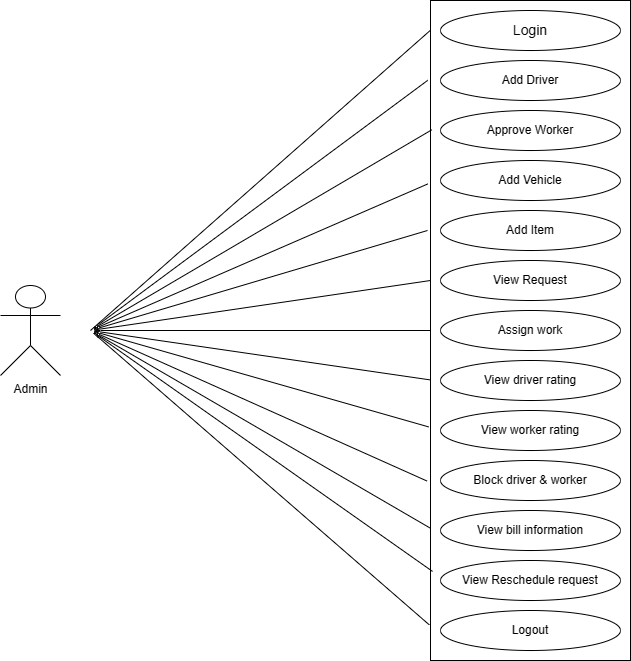
* Login
* View Request and Update status
* View payment
* Logout

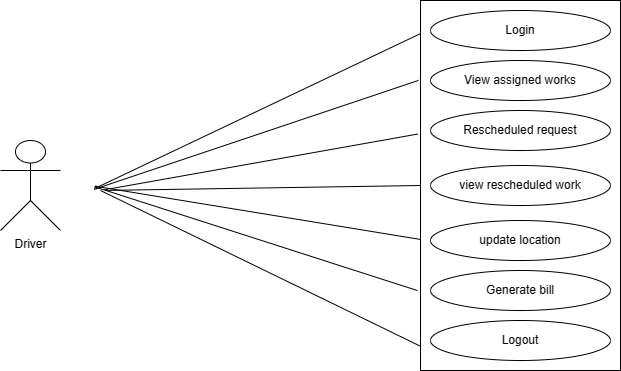
**Use case for the actor User**

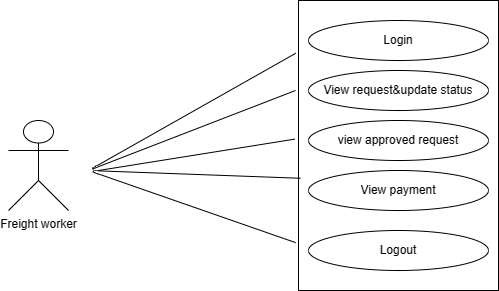
|  |  |  |
| --- | --- | --- |
| **1** | Which functions does the User require from the system? What does the workshop need to do? | User requires the following functionalities from the system such Send request, View request status, Track request, Add rate to driver, View Driver bill, Scan QR code, Request freight worker, View freight worker bill, Add rate to freight worker |
| **2** | Does the User need to read, create, destroy, modify or store some kind of information in the system? | Yes. User need to create, view and edit the data if require. |
| **3** | Could the User work be simplified by adding new functions to the system? | Yes, the system can reduce his/her work. |

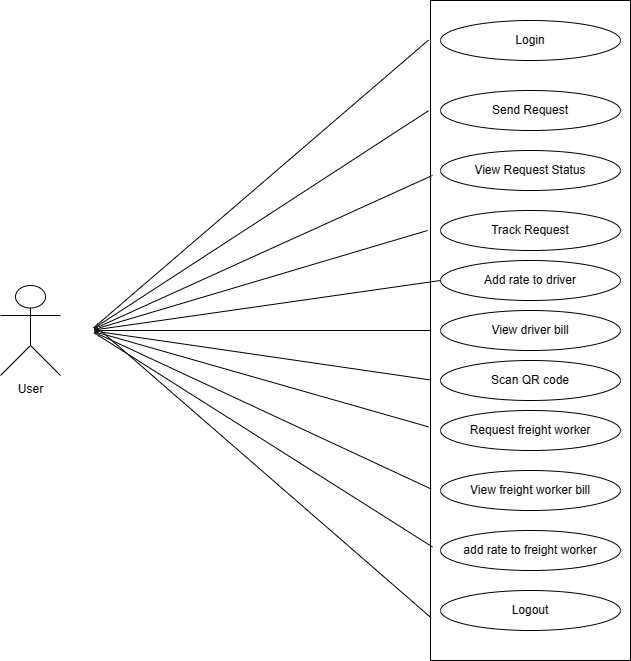
Above questions give the following use cases for the actor User.

* Login
* Send Request
* View request status
* Track request
* Add rate to driver
* View driver bill
* Scan QR code
* Request freight worker
* View freight worker bill
* Add rate to freight worker
* Logout

****

****

****

****

**2.4.3. ACTIVITY DIAGRAM**

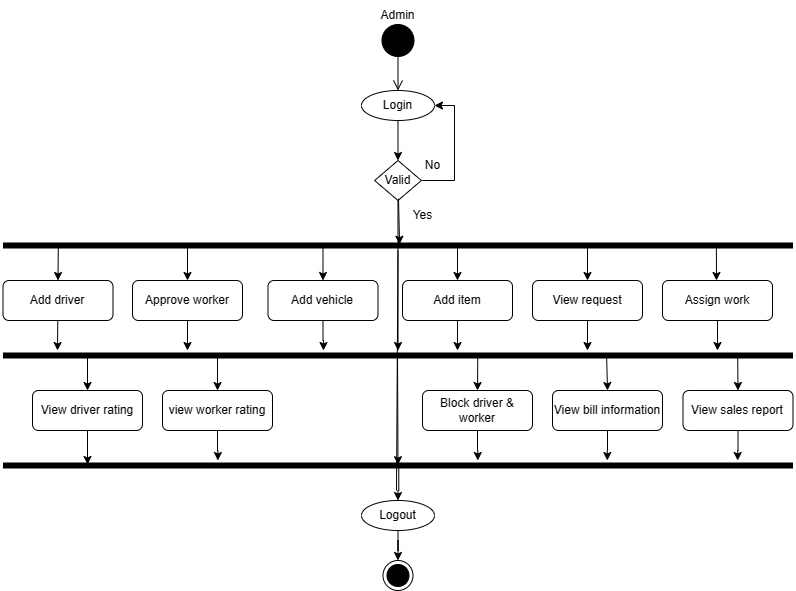
The activity diagram supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario. It uses rounded rectangles to imply a specific system function, arrows to represent flow through the system, decision diamonds to depict a branching decision, and solid horizontal lines to indicate that parallel activities are occurring.

The basic purposes of activity diagrams are similar to other diagrams. It captures the dynamic behavior of the system. Other diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

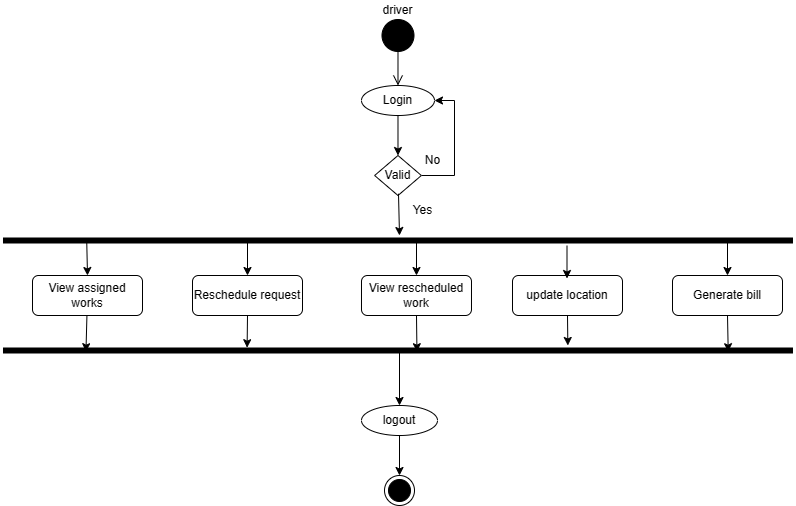
So, the purposes can be described as:

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched, and concurrent flow of the system.

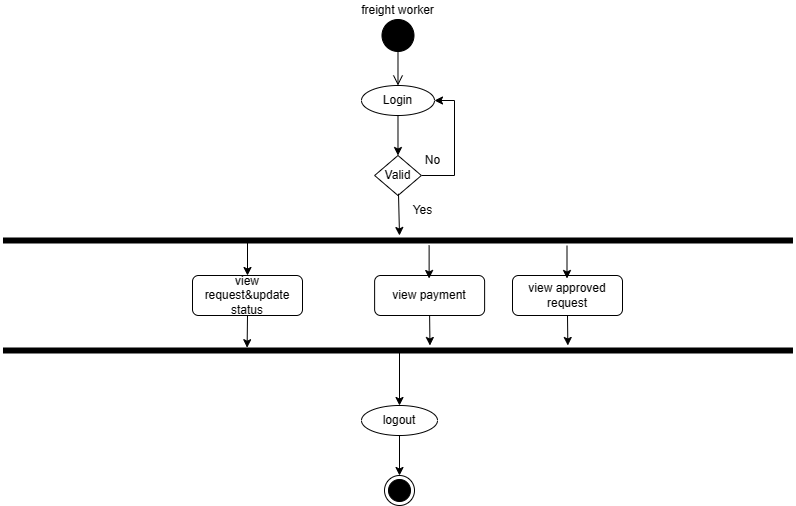
**Admin**

****

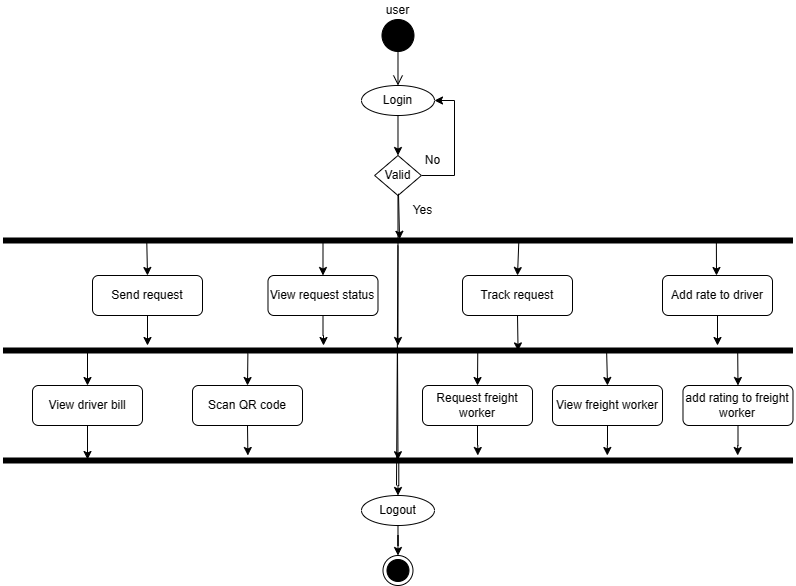
**Driver**

****

**Freight Worker**

****

**User**

****

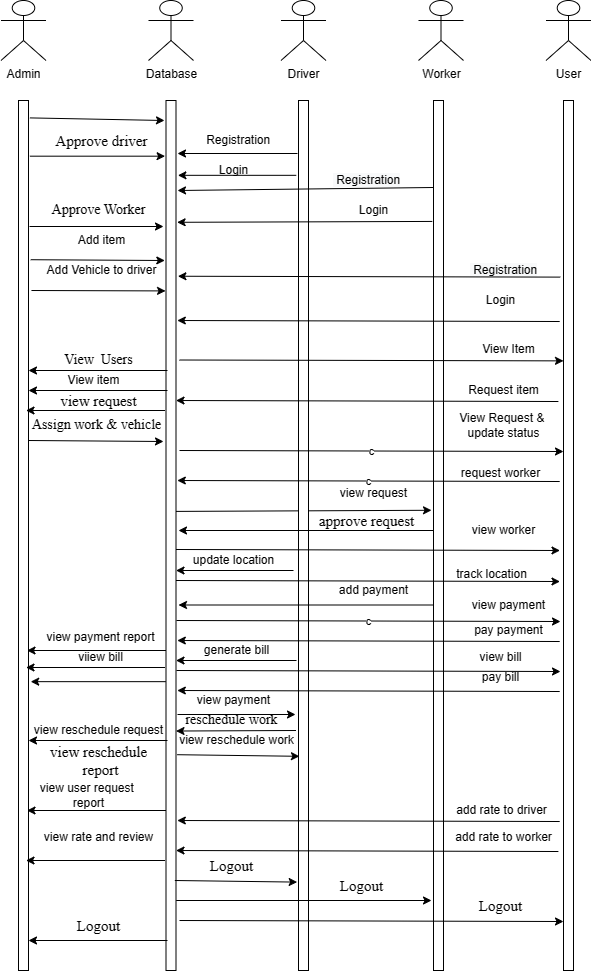
**2.4.4. SEQUENCE DIAGRAM**

Sequence diagrams are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram shows an interaction arranged in a time sequence. It shows the objects participating in the interaction by their life lines and the messages they exchange, arranged in a time sequence.

A sequence diagram has two dimensions: a vertical dimension represents time, horizontal dimension represents different objects. The vertical line is called the object’s lifeline. The lifeline represents the object’s existence during the interaction. This form was first popularized by Jacobson. An object is shown as a box at top of a dashed vertical line. A role is slot for an object within a collaboration that describes the type of object that may play the role and its relationships to other roles. However, a sequence diagram does not show the relationships among the roles or the association among the objects. An object role is shown as a vertical dashed line, the life line.

Each message is represented by an arrow between the life lines of two objects. The order in which these messages occur shown top to bottom on the page. Each message is labeled with the message name. The label also can include the argument and some control information and show self-delegation, a message that an object sends to itself, by sending the message arrow back to the same lifeline. The horizontal ordering of the lifelines is arbitrary. Often, all arrows are arranged to proceed in one direction across the page, but this is not always possible and the order conveys no information.

The sequence diagram is very simple and has immediate visual appeal- this is its greatest strength. A sequence diagram is an alternative way to understand the overall flow of the control of a program. Instead of looking at the code and trying to find out the overall sequence of behavior, we can use the sequence diagram to quickly understand that sequence.

****

**2.5. SYSTEM REQUIREMENTS**

**2.5.1 Hardware and Software Requirements**

Hardware and software requirements for the installation and smooth functioning of this product could be configured based on the requirements needed by the component of the operating environment that works as front-end system here we suggest minimum configuration for the both hardware and software components.

Working off with this application is requirements concrete on system environments. It includes two phases.

* Hardware Requirements
* Software Requirements

**External Interfaces Requirements**

This will use the standard input/output devices for a personal computer. This includes the following.

* Keyboard
* Mouse
* Monitor
* Mobile Phone/Tablet

1. **HARDWARE REQUIREMENTS**

* Processor : Dual Core
* Speed : 2 GHz
* RAM : 4 GB
* Hard Disk : 40 GB

1. **SOFTWARE REQUIREMENTS**

* Operating System : Windows 10
* Front End : HTML, CSS, JavaScript
* Back End : MySQL
* Programming Language : Python Flask
* Development Platform : PyCharm

**CHAPTER-3**

**SYSTEM DESIGN**

**3.0 SYSTEM DESIGN**

Design is a meaningful engineering representation of something that is to be built. It is an iterative process through which requirements are translated in to a blueprint for constructing the software. The goal of the design phase is to plan a solution of the problem specified by the requirements document.

Major activities during the design phase are:

* Data Base Design
* Architectural Design
* Interface Design
* Modular Design

**3.1 DATABASE DESIGN**

A database is collections of inter related data stored with minimum redundancy to serve many users quickly and efficiently. In database design data independence, accuracy, privacy, and security are given higher priority. Database design is an integrated approach to file design. This activity deals with the design of the physical database. All entries and attributes have been identified while creating the database. The database design deals with the grouping of data into number of tables so as to reduce the duplication of data, minimize storage space, and retrieve the data efficiently.

Guidelines for designing a database:

* Design a relational schema so that it is easy to explain its meaning. Do not combine attributed from multiple entity and relationship types into a single relation.
* Design the database schema so that no insertion, deletion or modification anomalies are present in the relation.
* As far as possible, avoid placing attributes in a base relation whose values may frequently be null.

**Advantages**

* Easy to use.
* Data independence.
* Accuracy and integrity.
* Avoiding inordinate delays.
* Recovery from failures.

**3.1.1 DATA FLOW DIAGRAM**

A data flow diagram is a graphical technique that depicts data flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A Level 0 DFD also called a fundamental system model or context model represents the entire software elements as a single bubble with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in next level i.e., Level 1 DFD. Each of the processes represented at level 1 are sub functions of overall system depicted in the context model.

**Data flow diagram Symbols:**

Source/Destination of Data

Data Flow

Process

**S**torage

**Level 0**

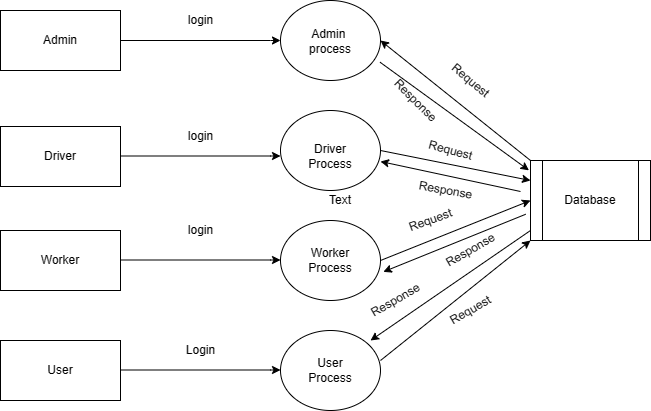
user Database

Response Request

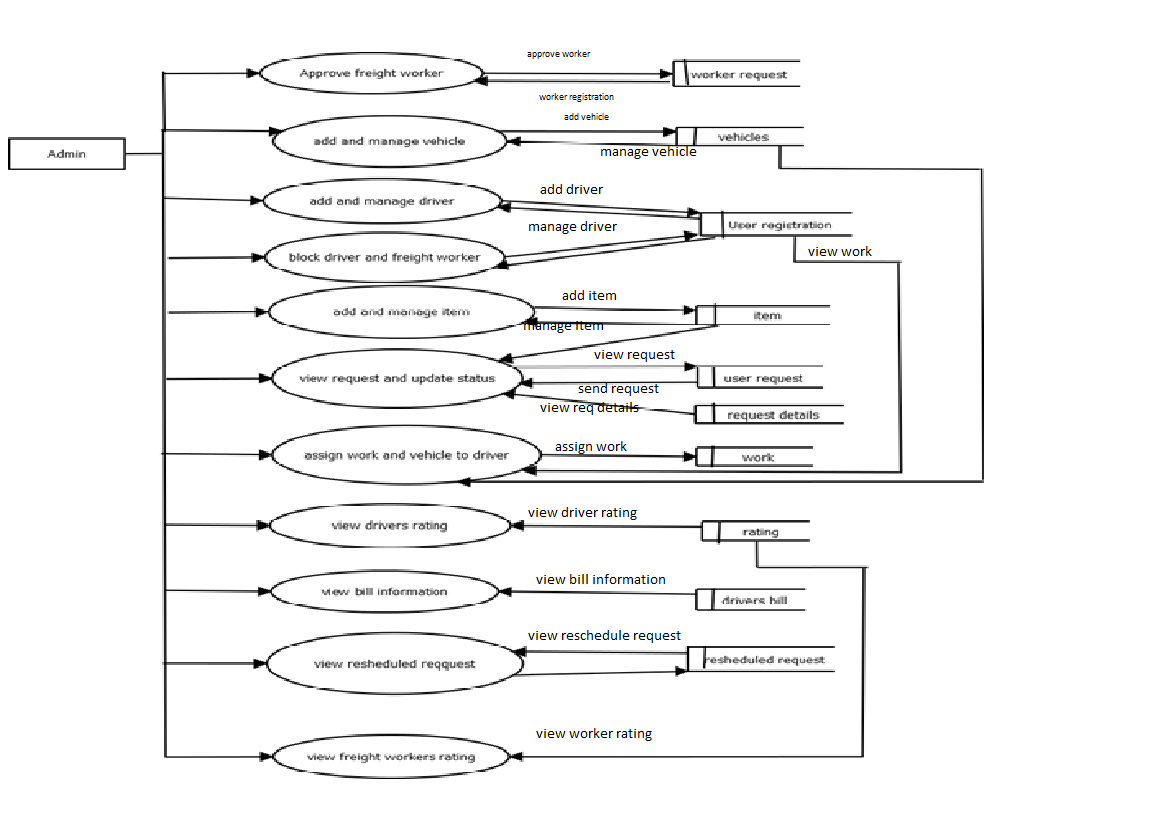
DD

Request Response

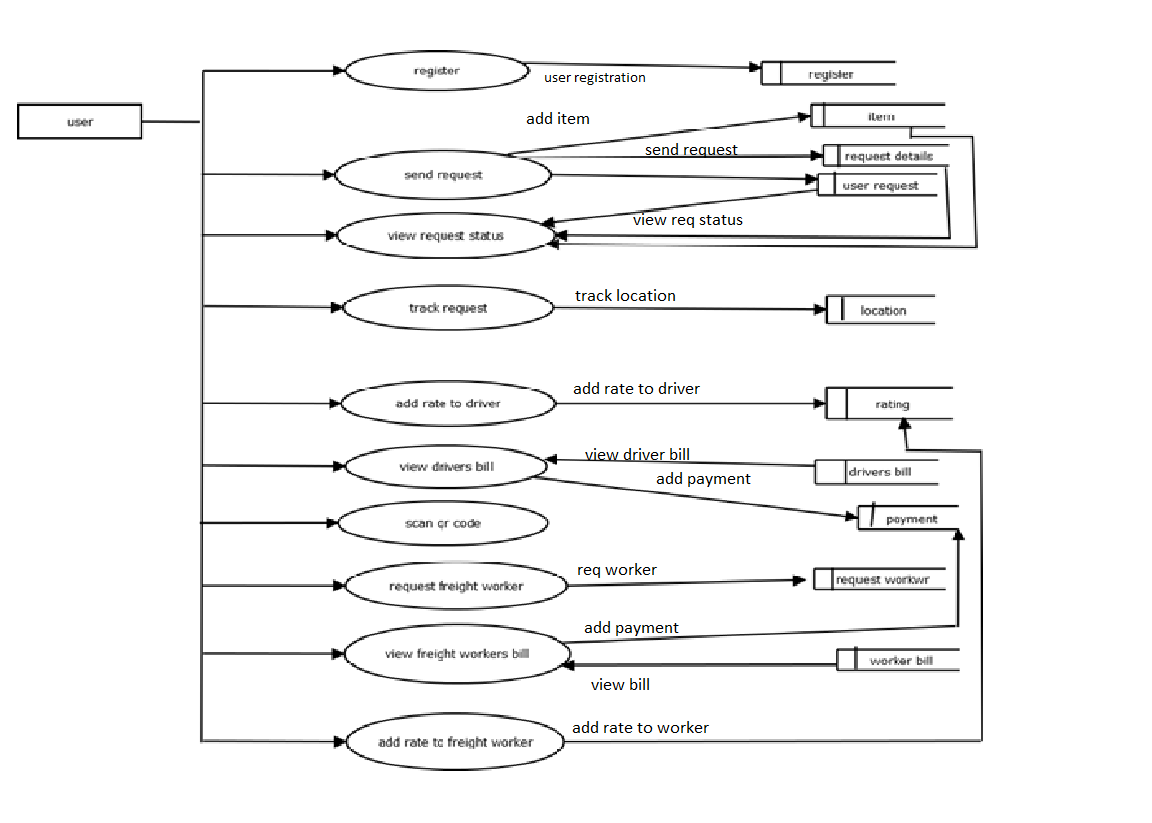
**Level 1**

****

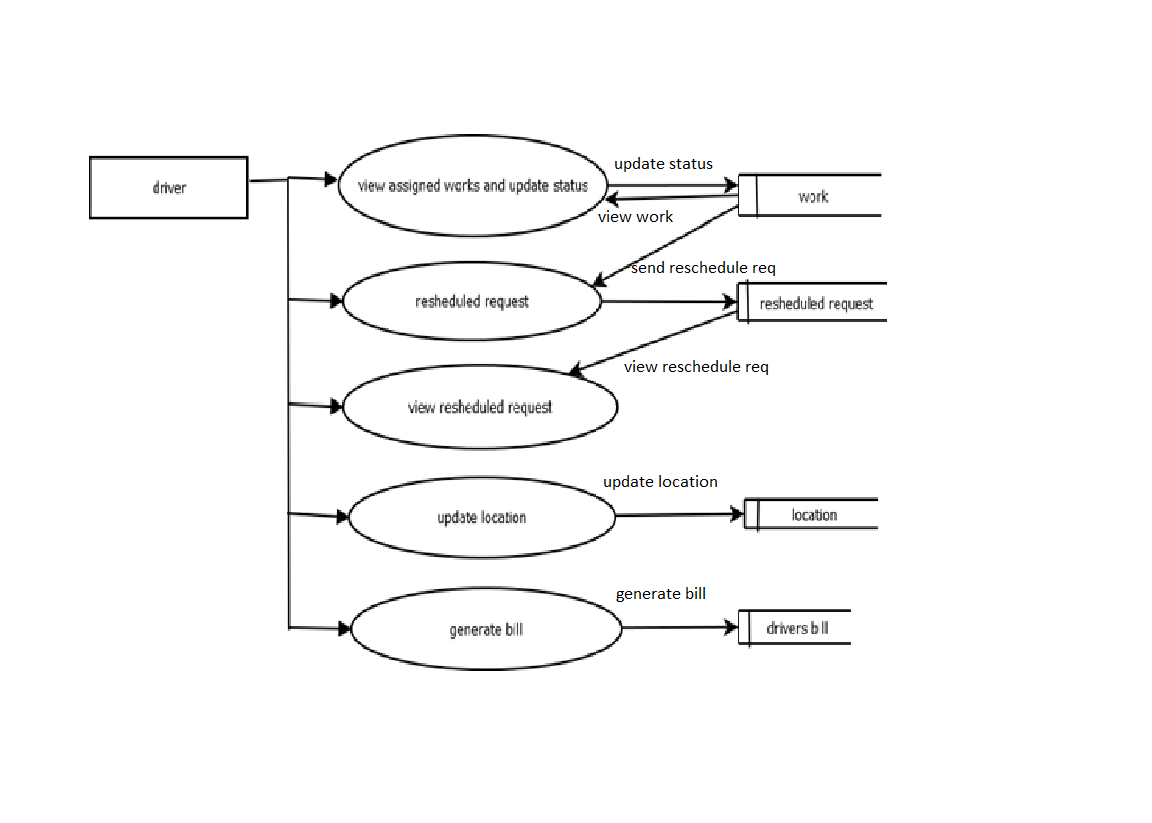
**Level 2: Admin**

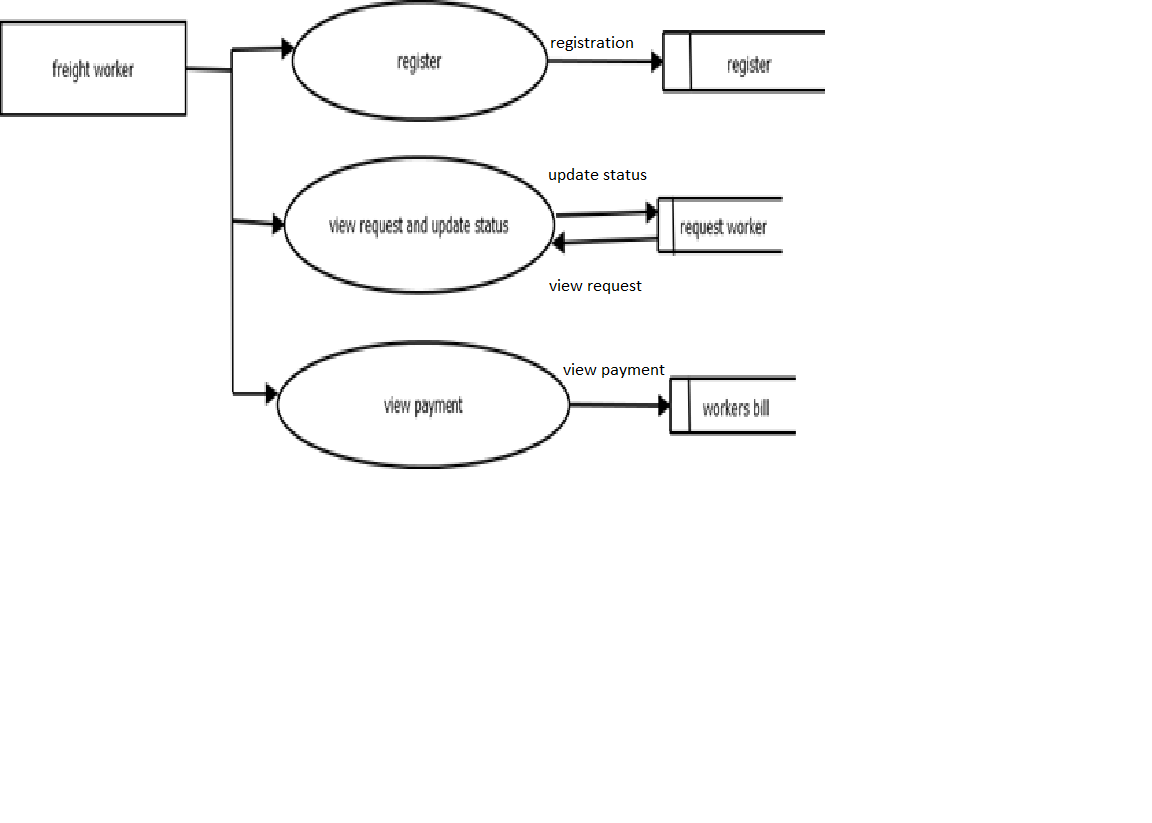
****

**Level 2: User**

****

**Level 2: Driver**

**Level 2: Freight Worker**

****

**3.1.2 E-R DIAGRAM**

An entity-relationship diagram is a data modeling technique that creates a graphical representation of the entities, and relationship between entities, within an information system.

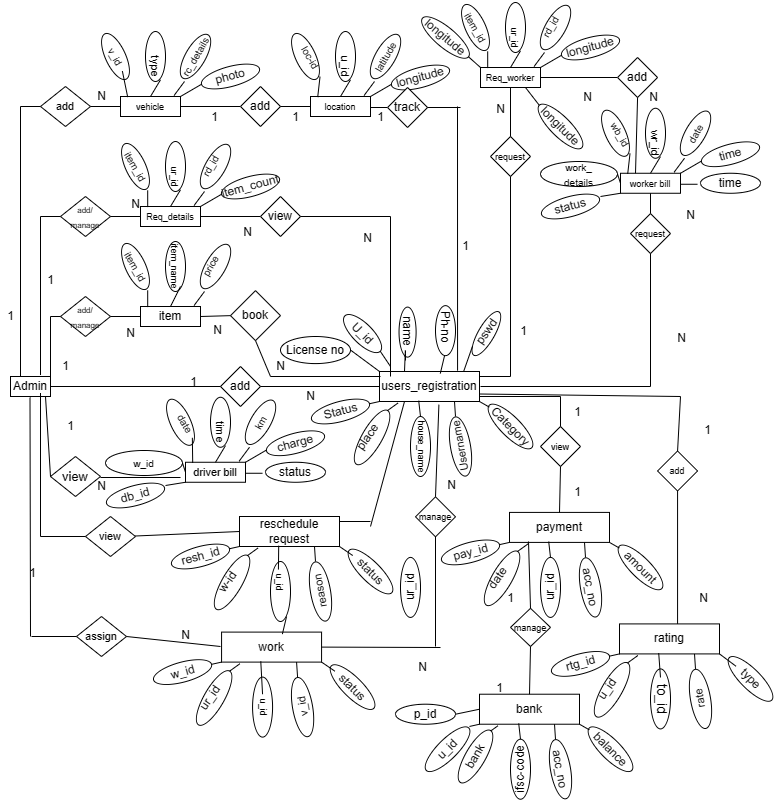
**There are three basic elements in ER models:**

* **Entities** are the “things” about which we seek information
* **Attributes** are the data we collect about entities.
* **Relationships** provided the structure needed to draw information from multiple entities.

Entity

Attributes

Relation

****

**3.1.3 TABLE DESIGN**

In the database all the information is stored in the form of tables. A table is simply a way storing data in rows and columns. In the system data is stored in many tables.

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| User\_tb | Store user details. |
| User\_request\_tb | Store request details. |
| Work\_tb | Store work details |
| Worker\_bill\_tb | Store work bill details |
| Driver\_bill\_tb | Store driver bill details |
| Vehicle\_tb | Store vehicle details |
| Item\_tb | Store items details |
| Rating\_tb | Store user reviews |
| Payment\_tb | Store payment details |

1. Table Name : Work\_tb

**Primary Key(\*) : W\_id**

**Foreign Key(\*) :U\_id, Ur\_id, V\_id**

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| W\_id | Integer | Id of the work |
| Ur\_id | Integer | Foreign key from table work |
| U\_id | Integer | Foreign key from table user |
| V\_id | Integer | Foreign key from table vehicle |
| Status | Varchar(20) | Status |

2. Table Name : User\_tb

**Primary Key(\*) :userid**



|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| userid(\*) | Integer | User Id |
| First\_name | Varchar(50) | First name |
| Last\_name | Varchar(20) | Last name |
| Phone\_no | Bigint(20) | Phone no |
| Date of Birth | Date | Date of birth |
| License\_no | Varchar(20) | License number |
| House\_name | Varchar(20) | Address |
| Post | Varchar(20) | Post |
| City | Varchar(20) | City |
| District | Varchar(20) | District |
| Place | Varchar(20) | Place |
| State | Varchar(20) | State |
| Pin | integer | Pin code |
| User name | Varchar(20) | User name |
| Password | Varchar(20) | Password |
| Category | Varchar(20) | Category |
| Status | Varchar(20) | Status |

3. Table Name : User\_request\_tb

**Primary Key(\*) : Ur\_Id**

**Foreign Key(\*) :U\_id**



|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| Ur\_id | Integer | Id of user request |
| U\_id | Integer | foreign key from table user |
| date | Varchar(20) | Order date |
| Latitude | Varchar(20) | Latitude |
| Longitude | Varchar(20) | Longitude |
| Status | Varchar(20) | Status |
| Amount | Varchar(20) | Amount |
|  |  |  |

4. Table Name : Work\_tb

**Primary Key(\*) : W\_id**

**Foreign Key(\*) :U\_id, Ur\_id, V\_id**

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| W\_id | Integer | Id of the work |
| Ur\_id | Integer | Foreign key from table work |
| U\_id | Integer | Foreign key from table user |
| V\_id | Integer | Foreign key from table vehicle |
| Status | Varchar(20) | Status |

5.Table Name : Worker\_bill

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| Wb\_Id | Integer | Id of worker bill |
| Wr\_id | Integer | Foreign key from Request\_worker |
| Date | date | date |
| Time | time | time |
| Charge | Varchar(50) | charge |
| Work\_Details | Varchar(20) | Work details |
| Status | Varchar(20) | Status |

6.Table Name : Driver\_bill\_tb

**Primary Key(\*) : db\_id**

**Foreign Key(\*) :W\_id**

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| Db\_id | Integer | Id of user driver bill |
| W\_id | Integer | F\_key from table work |
| Date | Date | Date |
| Time | time | time |
| Km | Varchar(20) | kilometer |
| Charge | Varchar(100) | Charge |
| Status | Varchar(100) | Status |

7.Table Name : Vehicle\_tb

**Primary Key(\*) : V\_id**

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| V\_id | Integer | ID of vehicle |
| Vehicle Type | Varchar(20) | Type of vehicle |
| RC\_Details | Varchar(20) | RC details |
| photo | Varchar(20) | Photo of vehicle |

8.Table Name : Item\_tb

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| Item\_Id | Integer | ID of the item |
| Price | Integer | Item price |
| Itemname | Varchar(30) | Item name |

9.Table Name : Rating\_tb

**Primary Key(\*) : db\_id**

**Foreign Key(\*) :W\_id**

|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| rtg | Integer | ID of rating |
| U\_id | Integer | Foreign key from table user |
| To\_id | Integer | To\_id |
| Rate | Varchar(20) | Rating |
| Type | Varchar(20) | Which type |

**10.Table Name : Payment\_tb**

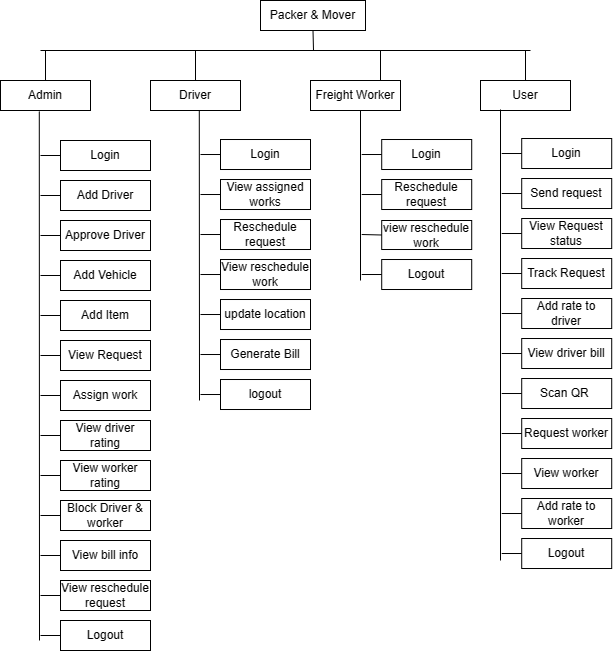
|  |  |  |
| --- | --- | --- |
| **Field** | **DataType** | **Description** |
| Payment\_Id | Integer | Id of payment |
| Date | date | Date |
| Ur\_id | Integer | F\_key from table user\_request |
| Acc no | Big int | Account number |
| Amount | Big int | Amount |

**3.2 ARCHITECTURAL DESIGN**

The architectural design develops a modular program structure and represents the control relationships between modules. It also defines interfaces that enable data to flow throughout the program.

**3.2.2 HIERARCHICAL DIAGRAM**

The hierarchical diagram is a technique for representing the modules of a system as a hierarchy and for documenting each module. It was used to develop requirements, construct the design, and support implementation of an expert system to demonstrate and verify the system. Structure charts can be used to display several types of information.



**3.3 INTERFACE DESIGN**

An interface design element for the software tell how information flows into and out of the system and how it is communicated among the components as part of the architecture.

**3.3.1 INPUT DESIGN**

Input design is the link between the information system and users and those steps that are necessary to put transaction data into a usable form for processing data entry. Instructing the computer to read data from a written printed document can active the activity of putting data into the computer for processing or it can occur by keying data directly into the system. The design of input focusing on controlling the errors, avoid delay, and keeping the process simple. System analyst decides the following input design details.

* What data to input?
* What medium to use?
* How the data is arranged and coded?

In my project named **“*Packer and Mover”,*** I tried to include the following design constrains provided in the software engineering.

**1: Avoid scattering of fields in the forms**

In all forms of the software the textboxes (which provided to input some data), label (which label the text boxes), combo box (list a set of values) etc all are arranged in a neat and well format. It provides a simple look to the pages. The buttons are placed at the bottom of the page and easily accessible to the user. The menus are arranged below the heading and at a minimum level of menus are arranged with pages. Menu provides the continuity to the pages.

**2: User only needs to enter a minimum amount of data**

All forms contain a minimum amount data, but most essentials. No page provides or wanted bulky of data. It provides more easiness to the user. It creates more the software to the end user. Also, the operation continues by single click.

**3: Avoid confusion in the forms**

All forms have a well-defined menu and each menu name indicate its purpose. So the user can easily access various forms without confusion. Each form and its sub forms are well labelled. So the user can easily identify the forms and work on that.

**The following are the input forms present in this project:**

* Login form
* User, Freight worker, Driver registration form
* Write complaints
* Add and Manage Item
* Reschedule Request
* Add messages, reviews and reply

**3.3.2 OUTPUT DESIGN**

Designing computer should proceed in well thought out manner. The term output means any information produced by the information system weather printed or displayed. Output design is a process that involves designing necessary output that have to be used by various users according to requirement. The efficient intelligent output design should remove the system relationship with the users and help in decision making.

When designing the output, system analyst must accomplish the following:

* Determine the information present
* Decide whether to print, display the information and select output medium
* Arrange information in acceptable format.

In my project, the outputs are in the form of reports. They are well format and it provides the output in a correct and neat format.

**The following are the output forms present in this project:**

* Form for viewing Assigned works
* Form for viewing request and status
* Form for viewing user details
* Form for viewing rate and reviews
* Form for viewing bill information
* Form for viewing payment details

**3.4 PROCEDURAL DESIGN**

The procedural design determines the modules included in the whole project which help us to identify the major functions.

**MODULE SPECIFICATIONS**

The “Spare Hub” consist of four actors:

* Admin
* Driver
* Freight Worker
* User

**1. Admin**

Admin has the overall control of the system. Admin can manage Driver and Freight worker, view payment details, manage complaints and manage overall control.

**2. Driver**

They can login, view assigned works and status, reschedule work request, update location and can generate bill

**3. Freight Worker**

They can login, and reschedule the request and can view rescheduled works

**4. User**

They can login, and view Items, request freight worker, add rating and review, view driver bill, write complaints.

**CHAPTER 4**

**CODING**

**4. CODING**

**4.1. ABOUT SOFTWARE TOOLS USED**

**4.1.1 PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

**4.1.2 FLASK**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

**4.1.3 MySQL**

**MySQL** is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation.

* The data in a MySQL database are stored in tables which consists of columns and rows.
* MySQL is a database system that runs on a server
* MySQL uses a standard form of the well-known SQL data language
* MySQL works very quickly and work well even with large date sets
* MySQL is ideal for both small and large applications.
* MySQL is very fast, reliable, and easy to use database system. It uses standard SQL
* MySQL compiles on several platforms.

**4.1.4 ANDROID**

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android’s user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars and android Wear for wrist watches, each with a specialized user interface.

**4.2 CODING PRINCIPLE**

The input to the coding phase is the design document. During coding phase, modules identified in the design document are coded according to the module specification. Objectives of coding phase are, to transform design into code and unit test the code.

**CODING GUIDELINES**

• Code should be easy to understand.

• Do not take pride in cryptic code.

• Code should be well documented.

• Comments should be present.

• Functions should be small.

• Do not use Go-to statement.

**4.3. SAMPLE CODE**

**1. Login.html**

<**body**>  
<**style**>  
  
 .**table**{  
  
 **width**: 50%;  
  
 }  
  
</**style**>  
<**form id="form1" name="form1" method="post" action="LOGIN#about"**>  
 <**div align="center"**>  
 <**table width="50" border="1" class="table table-striped table-hover"**>  
 <**tr**>  
 <**th scope="row"**>username</**th**>  
 <**td**><**label**>  
 <**input type="text" name="textfield" required**/>  
 </**label**></**td**>  
 </**tr**>  
 <**tr**>  
 <**th scope="row"**>password</**th**>  
 <**td**><**label**>  
 <**input type="password" name="textfield2" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters" required** />  
 </**label**></**td**>  
 </**tr**>  
 <**tr align="center"**>  
 <**th colspan="2" scope="row"**><**input type="submit" name="Submit" value="login"** />  
 *<!--{% if val=="no" %}-->  
 <!--<a href="adminrg#about">admin registration</a>-->  
 <!--{% endif %}-->  
 <!--<a href="workerrg#about">freight worker registration</a></th>-->* </**tr**>  
 </**table**>  
 </**div**>  
 <**label**></**label**>  
</**form**>  
</**body**>  
{% endblock %}

**2. Assign\_Vehicle.html**

<**body**>  
<**style**>  
  
 .**table**{  
  
 **width**: 50%;  
  
 }  
</**style**>  
<**form id="form1" name="form1" method="post" action="ASSIGN\_VEHICLE"**>  
 <**div align="center"**>  
 <**table width="200" border="1" class="table table-striped table-hover"**>  
 <**tr**>  
 <**th scope="row"**>VEHICLE</**th**>  
 <**td**><**label**>  
 <**select name="select"**>  
 {% for i in val2 %}  
 <**option value="{{i[0]}}"**>{{i[1]}}</**option**>  
 {% endfor %}  
 </**select**>  
 </**label**></**td**>  
 </**tr**>  
 <**tr**>  
 <**th scope="row"**>DRIVER</**th**>  
 <**td**><**label**>  
 <**select name="select2"**>  
 {% for i in val %}  
 <**option value="{{i[0]}}"**>{{i[1]}}{{i[2]}}</**option**>  
 {% endfor %}  
 </**select**>  
 </**label**></**td**>  
 </**tr**>  
 <**tr**>  
 <**th colspan="2" scope="row"**><**label**>  
 <**input type="submit" name="Submit" value="ASSIGN"** />  
 </**label**></**th**>  
 </**tr**>  
 </**table**>  
 </**div**>  
</**form**>  
</**body**>  
{% endblock %}

**3. Registration.py**

**def** WORKER\_REGISTRATION():  
 **try**:  
 fname=request.form[**'textfield14'**]  
 lname = request.form[**'textfield2'**]  
 phone = request.form[**'textfield3'**]  
 dob = request.form[**'textfield4'**]  
 licence\_no = request.form[**'textfield'**]  
 house\_name = request.form[**'textfield6'**]  
 place = request.form[**'textfield7'**]  
 post = request.form[**'textfield8'**]  
 city = request.form[**'textfield9'**]  
 district = request.form[**'textfield10'**]  
 state = request.form[**'textfield11'**]  
 pin = request.form[**'textfield12'**]  
 username = request.form[**'textfield5'**]  
 password = request.form[**'textfield13'**]  
  
 qry=**"insert into user\_registration values(NULL,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,'freight worker','pending')"** val=(fname,lname,phone,dob,licence\_no,house\_name,place,post,city,district,state,pin,username,password)  
 iud(qry,val)  
 **return '''<script>alert("successfully registered");window.location="/"</script>'''  
 except** Exception **as** e:  
 **return '''<script>alert("Already Exist");window.location="/workerrg"</script>'''**

**4.Rating.html**

<**body**>  
<**style**>  
  
 .**table**{  
  
 **width**: 50%;  
  
 }  
  
</**style**>  
<**form id="form1" name="form1" method="post" action=""**>  
 <**div align="center"**>  
 <**table width="200" border="1" class="table table-striped table-hover"**>  
 <**tr**>  
 <**th scope="col"**>name</**th**>  
 <**th scope="col"**>rate</**th**>  
 </**tr**>  
 {% for i in val %}  
 <**tr**>  
 <**td**>{{i[0]}}-{{i[1]}}</**td**>  
 <**td**>{{i[2]}}</**td**>  
 </**tr**>  
 {% endfor %}  
 </**table**>  
 </**div**>  
</**form**>  
</**body**>  
{% endblock %}

**CHAPTER 5**

**TESTING**

**5.0 SYSTEM TESTING**

For software that is newly developed, primary importance is given to testing the system. It is the last opportunity for the developer to detect the possible errors in the software before handing over it to the customer. Testing is the processes by which the developer will generate a set of data, which gives the maximum probability of finding all types of errors that can occur in the software. The various steps in testing the system can be listed as below:

**1.** Running the program to identify any errors that might have occurred while feeding the program into the system.

**2.** Applying the screen formats to regulate users to extend, so that the screens are comprehensible to the user.

**3.** Presenting the formats to the administration for the purpose of obtaining approval and checking if any modification has to be done. Obtaining feedbacks from users and analyzing the scope for improvement.

**4.** Checking the data accessibility from the data server and whether any improvement is needed or not.

Testing is a methodology for evaluating the project. The good test has a high probability of finding an error. Testing is generally two types- Black box testing and White box testing.

• Unit Testing

• Integration Testing

• System Testing

• Validation Testing

**5.1 UNIT TESTING**

Unit testing is carried out to screen wise, each screen being identified as an object. Attention is diverted to individual modules, independently to one another to locate in coding and logic.

In unit testing,

 Module interface is tested to ensure that information properly flows into and out of the program under test.

 Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in algorithm execution.

 Boundary condition is tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

 All independent paths through the control structures are executed to ensure that all statements in the module have been executed at least once.

 Error handling paths are also tested.

**TEST CASES**

**Login form**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1. | Enter wrong user name and pass word. | Display login form again with a warning message. | Message displayed. | Pass |
| 2. | Enter correct user name and wrong password. | Display login form again with a warning message. | Message displayed. | Pass |
| 3. | Enter correct user name and password. | Users can login into the system. | Appropriate home page is displayed. | Pass |
| 4. | Press login button without filling the user name and password. | Display a warning message to fill the fields. | Warning message is displayed. | Pass |

**Registration form**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Form displayed. | Display the registration form. | Form loaded | Pass |
| 2 | Enter the name in integers. | Display an invalid message. | Invalid message displayed | Pass |
| 3 | Enter the mobile number in characters. | Display an invalid message. | Invalid message displayed. | Pass |
| 4 | Enter the mobile number more than and less than 10 integers. | Display an invalid message. | Invalid message displayed. | Pass |
| 5 | Click the save button without filling the details | Display a warning message to fill the details. | Warning message displayed. | Pass |
| 6 | Click on save button with filled fields. | Accept the details. | Registration successfully done. | Pass |
| 7 | Click cancel button | Clear all fields to blank | All fields cleared. | Pass |

**5.2 INTEGRATION TESTING**

Integration testing is a symmetric technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. Unit tested module were taken and a single program structure was built that has been dictated by and tested in small segments, where errors were easy to locate and rectify. Each database or table manipulation operation was written as single program was tested again with numerous test data to check for its functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Input/procedure** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Check the value pass between different forms are appropriate format | Appropriate operations of different forms. | Same as expected. | Pass |

**5.3 SYSTEM TESTING**

System testing is used test the entire system (Integration of the all modules). It also tests to find the discrepancies between the system and the original objective, current specification and system documentation. The entire system is checked to correct deviation to achieve correctness.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Input/procedure** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Check whether indented output is obtained. | All operations are carried out properly. | Same as expected. | Pass |

**5.4 VALIDATION TESTING**

At the conclusion of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins validation test has been conducted one of the two possible conditions exists. One is the function or performance characteristics confirm to specification and are accepted and the other is deviation from specification is uncovered and a deficiency list is created.

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 SYSTEM IMPLEMENTATION**

System implementation is the stage where the theoretical design is turned into a working system. The system can be implemented only after through testing is done and if it if found to work according to specifications. The following methods were undergone.

• Testing developed programs with updating.

• Correction of errors identified. .

• Making necessary changes with actual data.

• Doing a parallel run of the system to find out any errors identified and to correct them.

• Training of user personnel.

The implementation method used to implement Queue Management is Parallel Run. That is, the new system will work parallel to the existing system. The new system will replace the existing system completely. The implementation stage involves following tasks.

• Careful planning.

• Investigation of the current system and constraints.

• Design of methods to achieve the changeover.

• Training of the staff in the changeover phase.

Technologies used in the development of the software are:

• Development tool: PyCharm 2017

• Language: Python 3.10

• Framework: Flask 1.1.5

• Database: MySQL

• Web Server: Chrome

• Scripting: HTML, JavaScript, CSS, bootstrap

**CHAPTER 7**

**CONCLUSION**

**7.1 CONCLUSION**

As a result, this application software will be easy to modify the information. It was done under the guidance of the experienced project guide. All the current requirements and possibilities have been taken care during the project time. Packer and mover is a project which is developed to provide an interactive platform between clients and packer and mover agencies. Many efforts have been taken in a way to implement this project in many areas. And it will increase the use of the packer and mover. To manage all this requirement, they need software which will work and take care of it.

**CHAPTER 8**

**REFERENCE**

**8. REFERENCES**

**8.1 WEBSITES**

• http://www.w3schools.com

•<http://www.stackoverflow.com>

•<http://www.codeproject.com>

•<https://youtube.com>

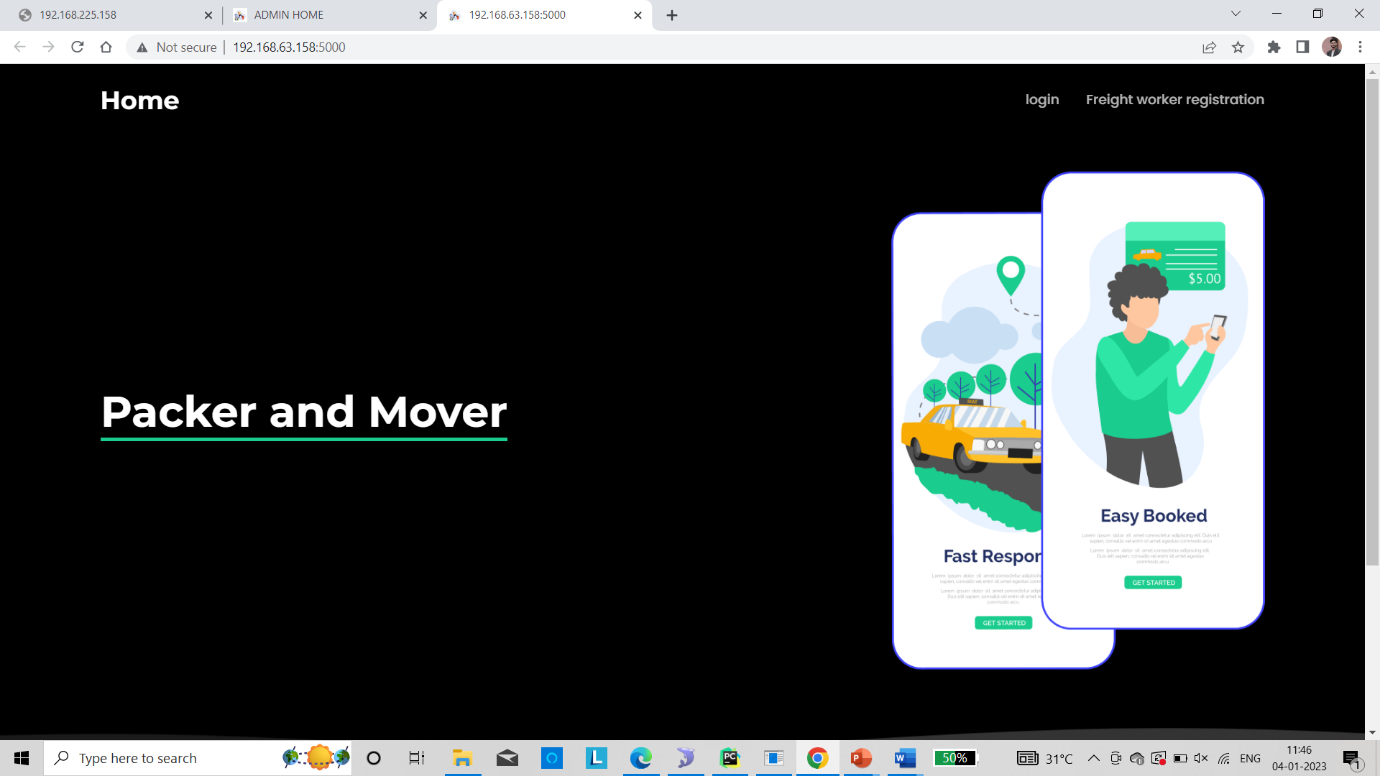
**8.2 REFERENCE BOOKS**

* Software Engineering, ROGER. S. PRESSMAN, Tata McGraw Hill, Fifth Edition, Year 2004.
* Database System Concepts, Abraham Silberschatz, Henry F Korth, S Sudarshan, Sixth edition, Year 2011

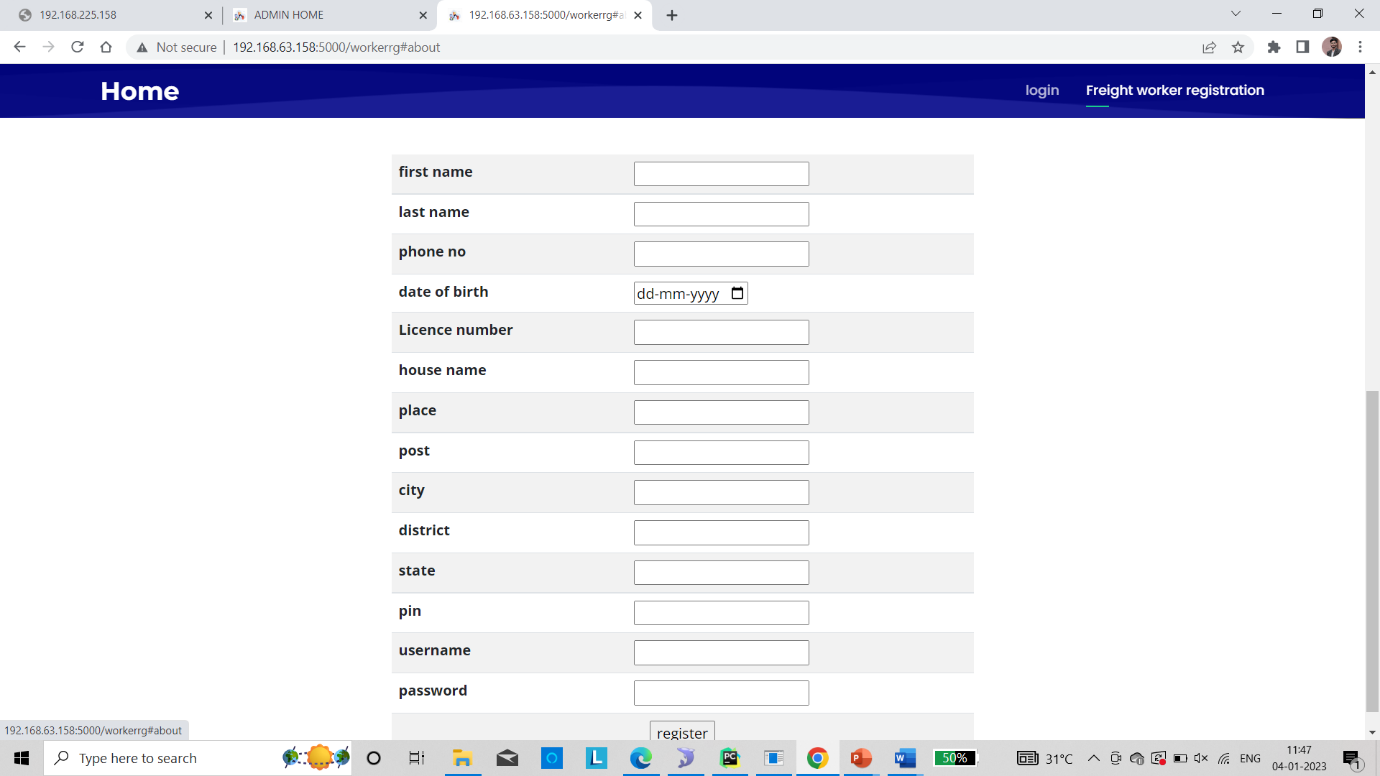
**CHAPTER 9**

**APPENDIX**

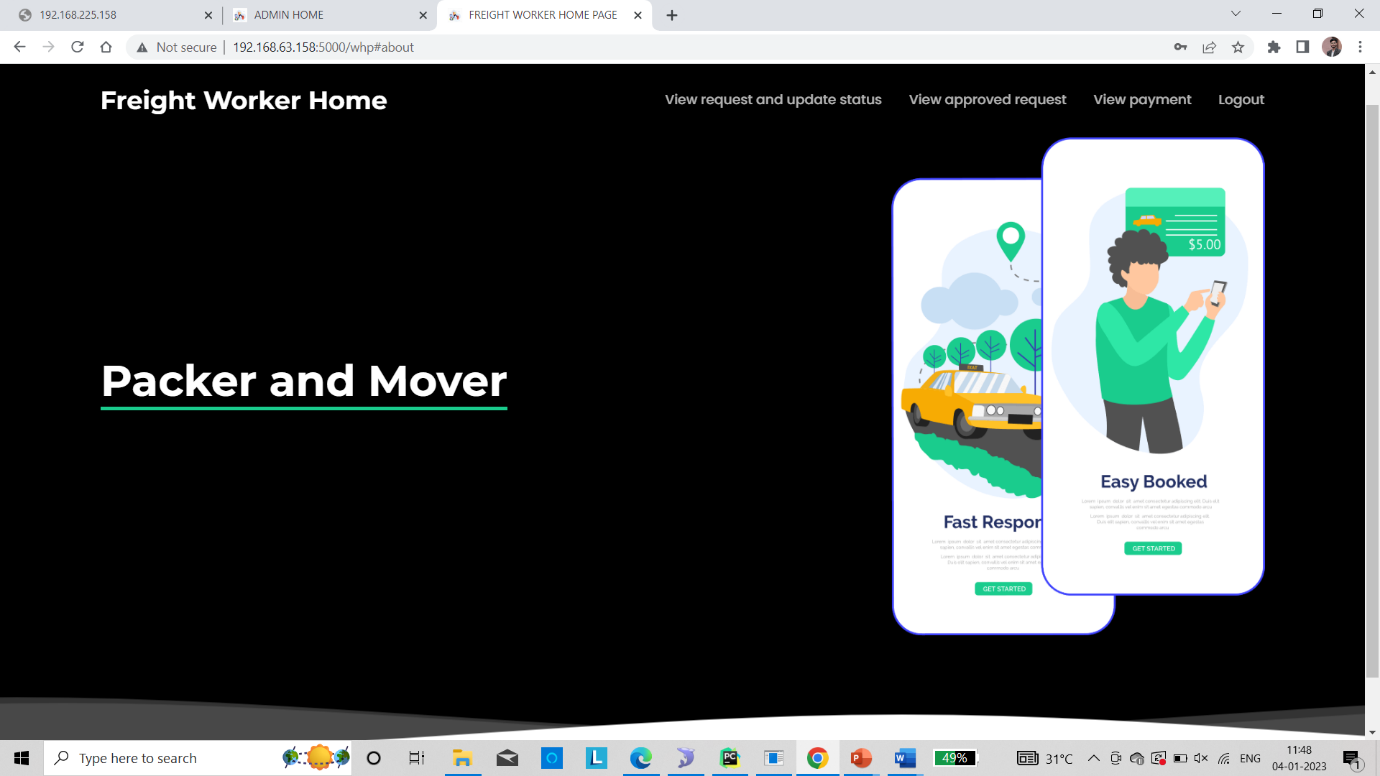
**Home**

****

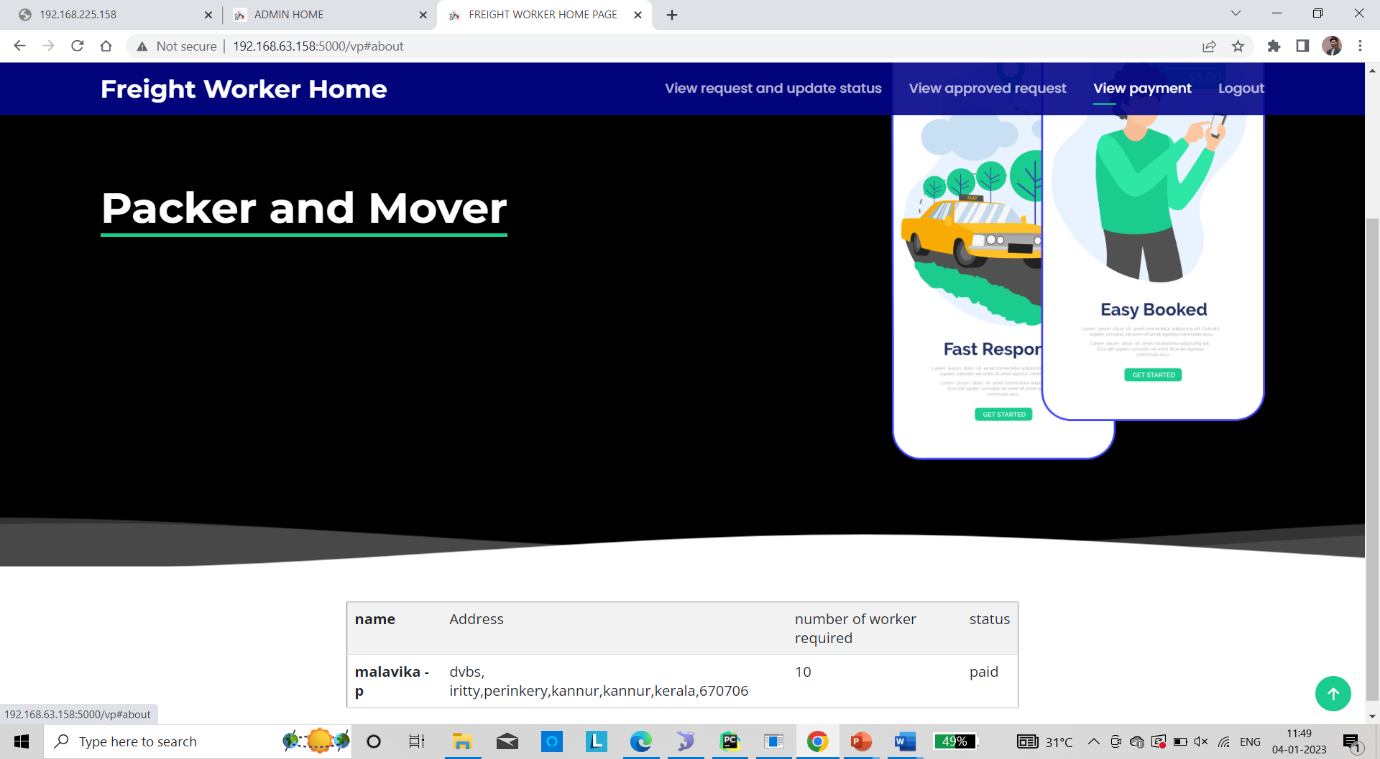
**Freight Worker Registration**

****

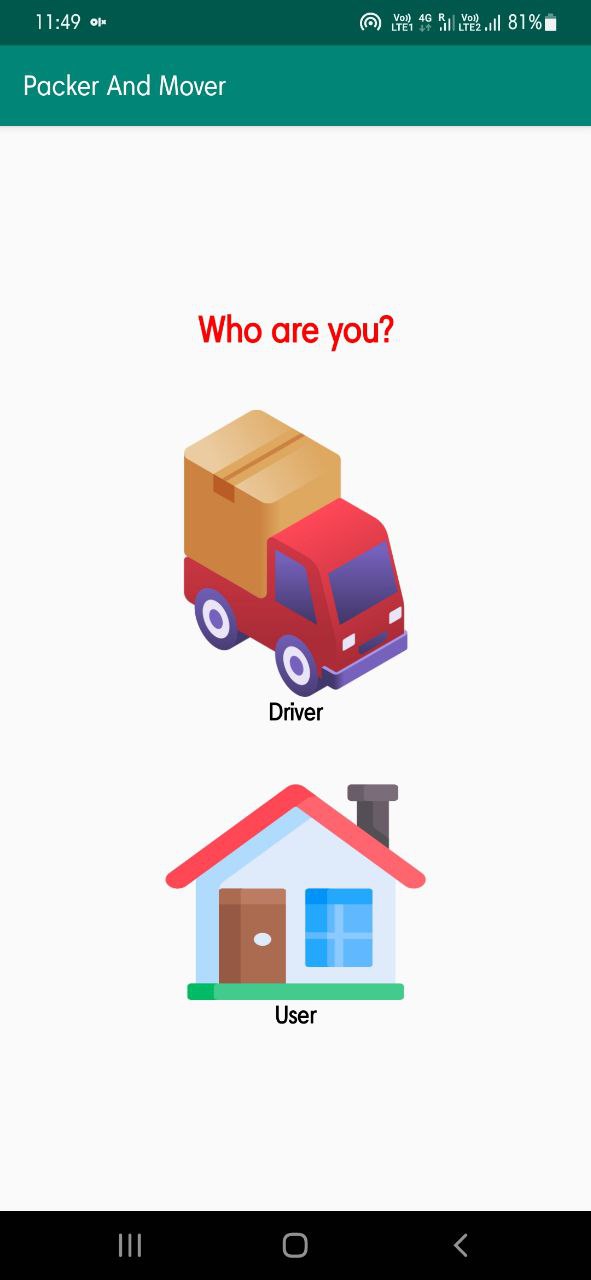
**Freight Worker Home**

****

**Payment**

****

**Android Home**

****