SPARE HUB

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 **"SPARE HUB"** is a bonafide record of the original work done by **JITHU K PAVITHRAN (Reg.No:C0GMCA2105)** during the **third semester** of the year **2020-2022** in partial fulfilment of the requirements for the award of Master's Degree in Computer Applications under the Kannur university.

Internal Guide **Date:**

External Examiners **Head of the Institution**

1.

2.

DECLARATION

I JITHU K PAVITHRAN, third semester MCA, student of Don Bosco College, Angadikadavu, under Kannur University do hereby declare that the project entitled "SPARE HUB" is the record of original work done by me under the supervision of Mr.KEVINSON KURIAN (HOD, Dept of MCA), 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

As we step forward into the modern era of technology, there exist online shopping systems for all the products but shops in our areas are not included in the system, they sell the products through offline. And now it is difficult to find the best servicing workshops in different places and access the services. By this project we will be able to implement a online shopping and service system for those who want to buy spare parts online and search the workshop services. In the current scenario, the system has a wide scope because increased usage of vehicles and their parts and services, can be got through the online mode. "SPARE HUB" will act as a online shopping and service system for many users.

Main actors of this project are the Admin, Spare parts shop, Workshop and Users. Admin can control Spare part shop and Workshops. Users' product ordering and payments are managed by spare parts shops. Spare part shop updates the product details and workshops also update the details for the users. Admin can control all the services by spare part shop and workshops and views all the sales report and service.

TABLE OF CONTENT

CHAPTER-1
INTRODUCTION

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Nowadays most of the spare parts items are available in the market, can be available through the online mode. There are many online shopping websites like Amazon, Flipkart, etc for these purposes and we can easily place our order through it. We know that these big online shopping websites includes a delivery system as part of it so that they can track their products. There are many small companies and shops who provides an online way of purchasing spare parts products, but it does not provide online workshop services. As the vehicles are within the end of the day, their internal and external parts require frequent replacements, consistent with the wear and tear and tear of the part in use.

This project aims to implement such an online ordering and delivery system called **'Spare Hub'** which offers online ordering of spare parts and searching of nearby workshops in different locations. Using this system, it provides users to search spare parts for any vehicles, compare products, order the products and online payment It also provides users to search workshops and access the services.

Main actors of this project are the Admin, Spare parts shop, Workshop and Users. Admin can control Spare part shop and Workshops. Users' product ordering and payments are managed by spare parts shops. Spare part shop updates the product details and workshops also update the details for the users. Admin can control all the services by spare part shop and workshops and views all the sales report and service

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

1.2 SCOPE OF THE PROJECT

Today’s huge amount of vehicle usage needs the availability of spare parts also. A system for this purpose is needed hence I propose “SPARE HUB” for ordering the parts online and access the workshop services.

As we step forward into the modern era of technology, there exist online shopping systems for all products but shops in our areas not included in that system, they sell the products now in offline. And now it is difficult to find the best servicing workshops in different places and access the services.

By this project we will be able to implement a online shopping and service system for those who want to buy spare parts online and search the workshop services. In the current scenario, the system has a wide scope because increased usage of vehicles and their parts and services, can be got through the online mode. SPARE HUB will act as a online shopping and service system for many users.

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 spare parts shopping Websites. These sites are given below:
    - ✓ www.Gomechanic.com
    - ✓ www.bootmo.com
    - ✓ www.safexbikes.com

- The interview method helped to collect more information from some shops

2.1 EXISTING SYSTEM

There are several online web portals as well as android-based applications for ordering products. We know that big online shopping companies provide all services. Nowadays, there are many small companies and shops who provides an online way of purchasing products but it does not exist a proper system for including all the spare parts shop and workshops with their products and services.

.

Disadvantages

- No proper system to manage the online spare parts shops and access the services from our locality
- Takes long time to deliver the products
- Unavailability of spare parts
- No prebooking options for parts

2.2 PROPOSED SYSTEM

The proposed system was developed in such a way that it should overcome the drawbacks of existing system. It plays an important role in spare part shopping and searching vehicle services. The project named “Spare Hub” act as a online spare parts ordering and workshop searching system. It includes all registered spare part shops and workshops for the users.
This system is very user friendly, the retrieval and storing of data is fast and data is maintained efficiently.

Advantages

- Developed mainly for online spare parts shopping
- Includes workshop services
- Pre booking system for the currently unavailable products

- Search nearby workshops
- Product service availability
- Category and brands-based products for all vehicles

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 Project Spare Hub can be used by anyone without knowing the language python or Django framework. Anyone can be accessing the system if they have simple browsing experience, so they can easily use this software for parts ordering and to find out the workshops nearby them. 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 "Spare Hub" 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 Spare Hub 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, Spare parts shop, Workshop and User.
2.	Who will lead support from the system and do their tasks?	Administrator, Spare parts shop, Workshop and 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, Spare parts shop, Workshop and User.

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

- Admin
- Spare parts shop
- Workshop
- 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 Approve/Reject Spare part shops and Workshops, Add category, Add brand, Add model, View spare parts shop and Workshop, View complaint, Write reply, View sales report, View category, View brand and View model.
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
- Approve/Reject Spare part shops
- Approve/Reject Work shops
- View Spare part shops
- View Workshops
- Add category
- Add brand
- Add model
- View complaints
- Write reply
- View sales report
- Receive share
- View category
- View brand
- View model
- Logout

Use case for the actor Spare part shop

1	Which functions does the Spare part shop require from the system? What does the Driver need to do?	Spare part shop requires the following functionalities from the system such as Add spare parts, Update and delete spare parts, View orders, Approve/Reject order, Add tracking details, View pre booking, View rating and review, View payments, View spare parts, View profile and Update profile
2	Does the Spare part shop need to read, create, destroy, modify or store some kind of information in the system?	Yes. Spare part shop 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
- Add spare parts
- View spare parts
- View orders
- Approve/Reject order
- Add tracking details
- View pre booking
- View rating and review
- View payments
- View profile
- Update profile
- Logout

Use case for the actor Workshop

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 Add service, View service, Update service, View rating and review, View profile and Update profile

2	Does the workshop need to read, create, destroy, modify or store some kind of information in the system?	Yes. Workshop 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 User.

- Login
- Add service
- View service
- Update service
- View rating and reviews
- View profile
- Update profile
- Logout

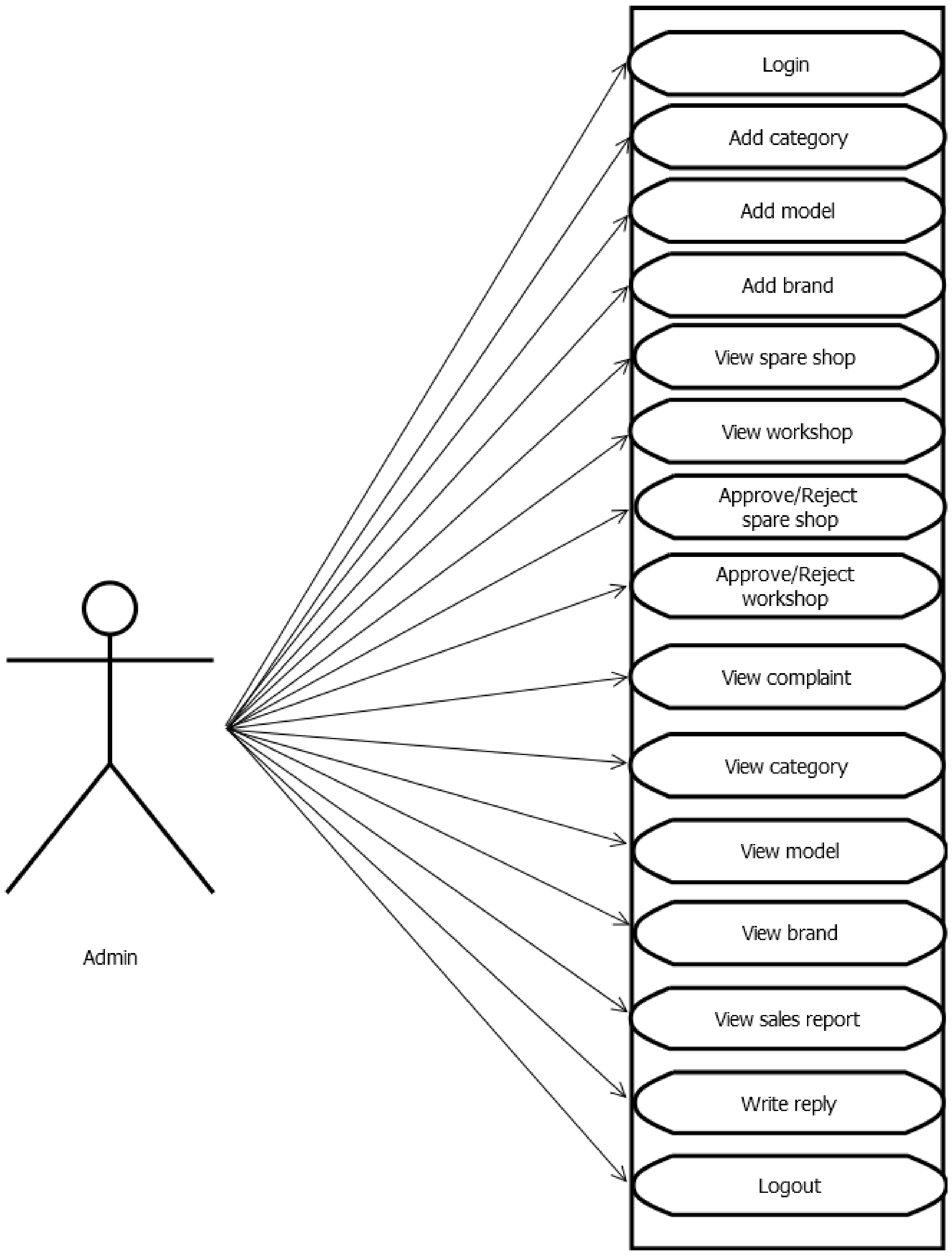
Use case for the actor User

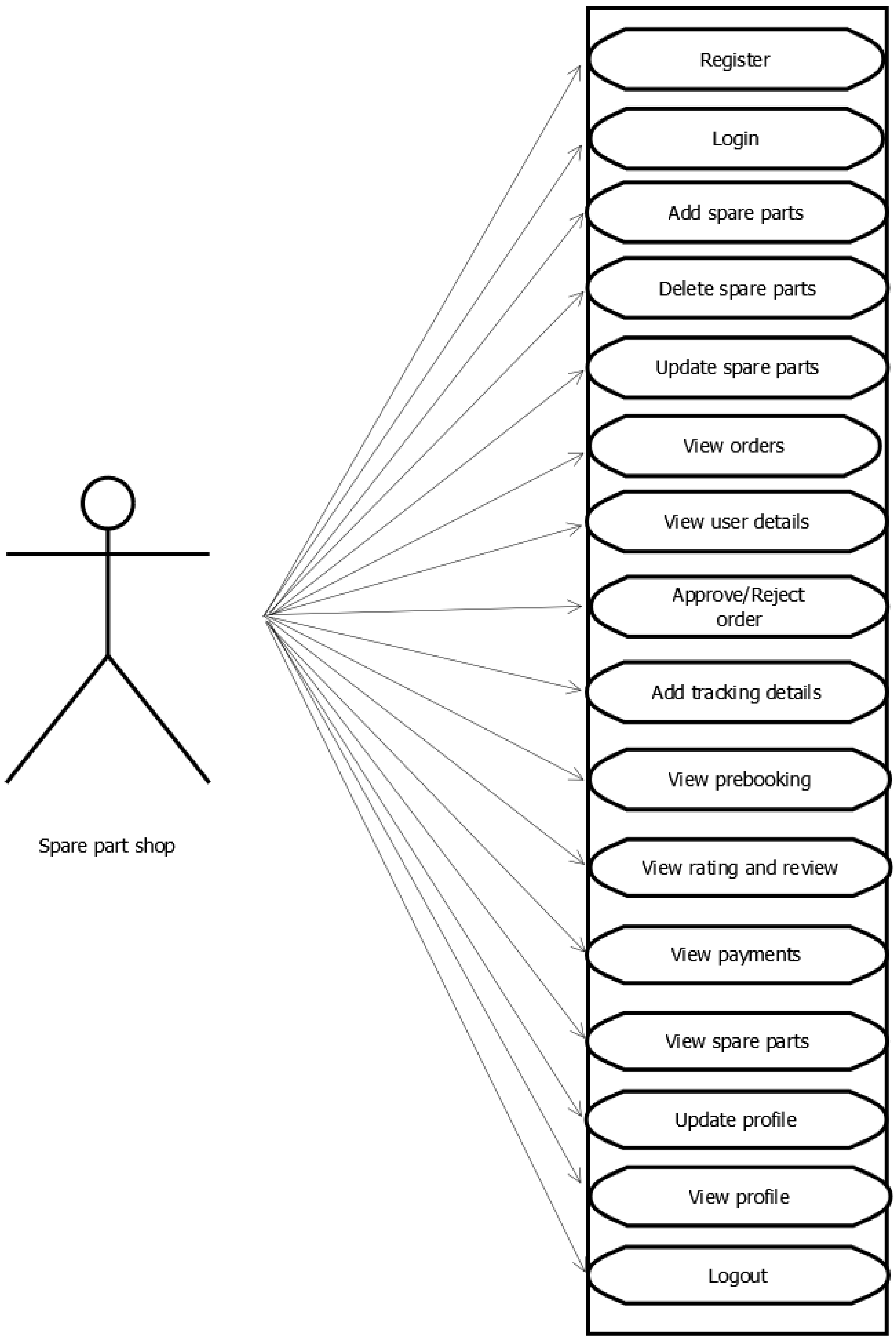
1	Which functions does the User require from the system? What does the workshop need to do?	User requires the following functionalities from the system such View spare parts, Order spare parts, Delete order, View reply, View order status, Pre booking, View pre booking status, Add payment, Rate and review, Search workshops, View service, Write complaint, View profile and Update profile
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.

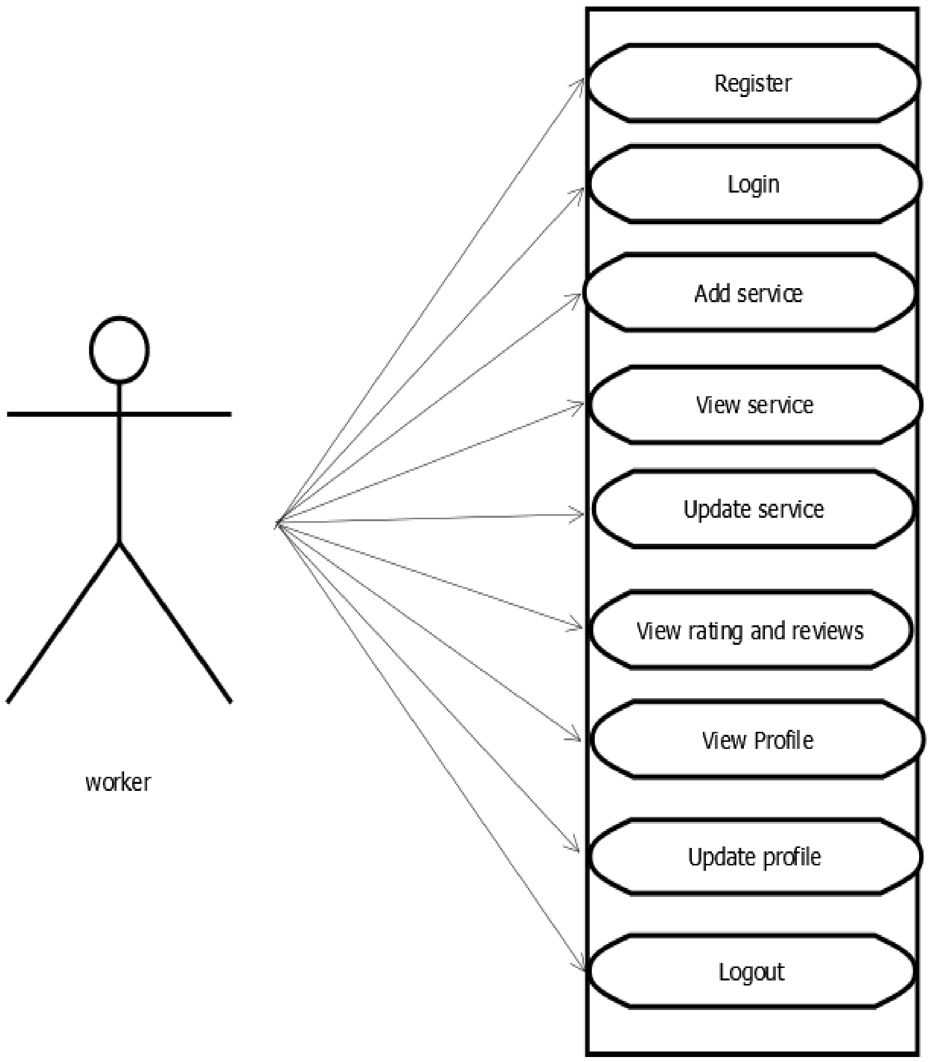
3	Could the User work be simplified by adding new functions to the system?	Yes, the system can reduce his/her work.

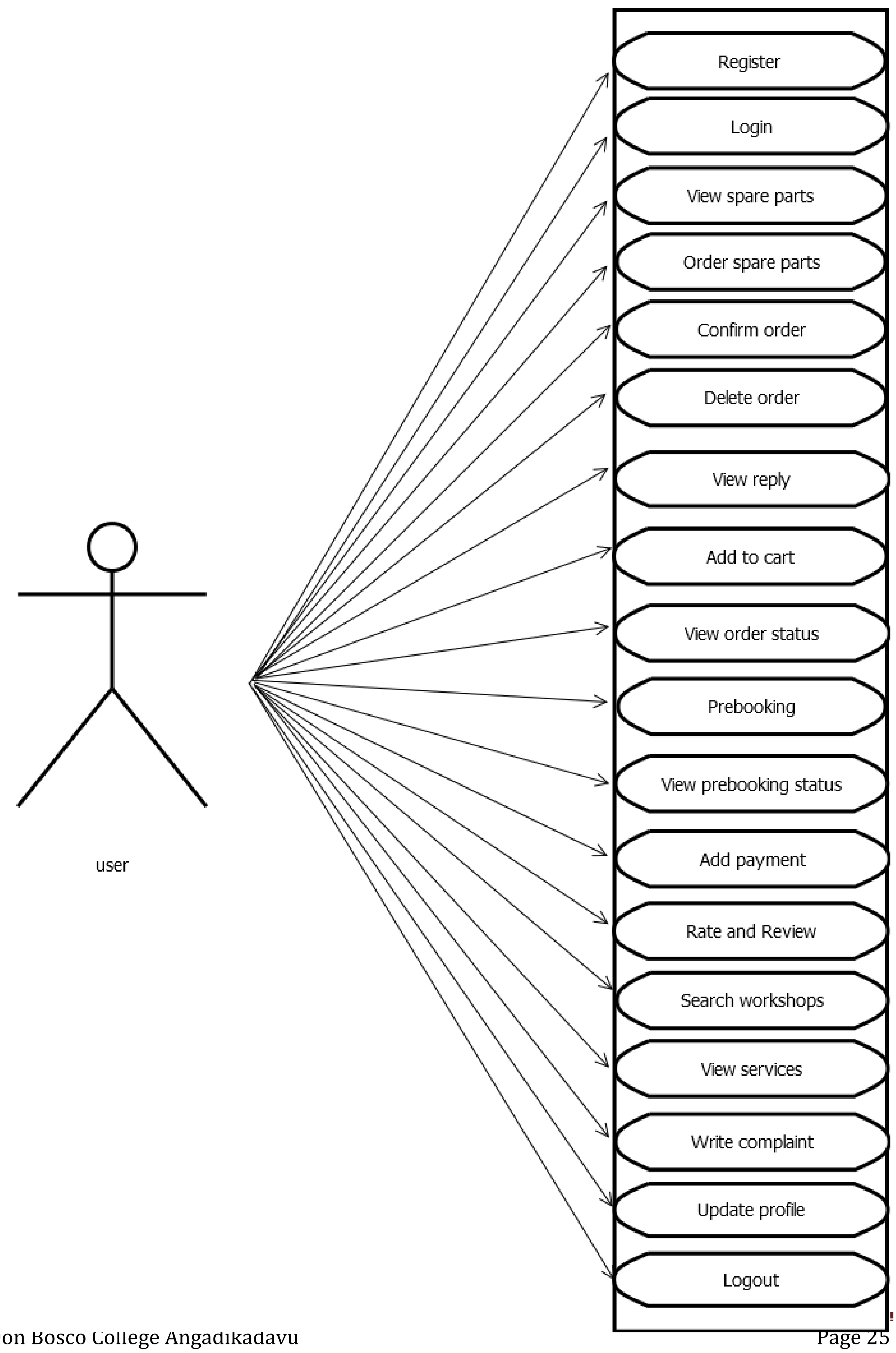
Above questions give the following use cases for the actor User.

- Login
- View spare parts
- Order spare parts
- Delete order
- View reply
- View order status
- Pre booking
- View pre booking status
- Add payment
- Rate and review
- Search workshops
- View services
- Write complaint
- View and update profile
- Logout


Login
Add category
Add model
Add brand
View spare shop
View workshop
Approve/Reject spare shop
Approve/Reject workshop
View complaint
View category
View model
View brand
View sales report
Write reply
Logout
Admin


Spare part shop
Register
Login
Add spare parts
Delete spare parts
Update spare parts
View orders
View user details
Approve/Reject order
Add tracking details
View prebooking
View rating and review
View payments
View spare parts
Update profile
View profile
Logout


Register
Login
Add service
View service
Update service
View rating and reviews
View Profile
Update profile
Logout
worker


Register
Login
View spare parts
Order spare parts
Confirm order
Delete order
View reply
Add to cart
View order status
Prebooking
View prebooking status
Add payment
Rate and Review
Search workshops
View services
Write complaint
Update profile
Logout
user

2.4.3. ACTIVITY DIAGRAM

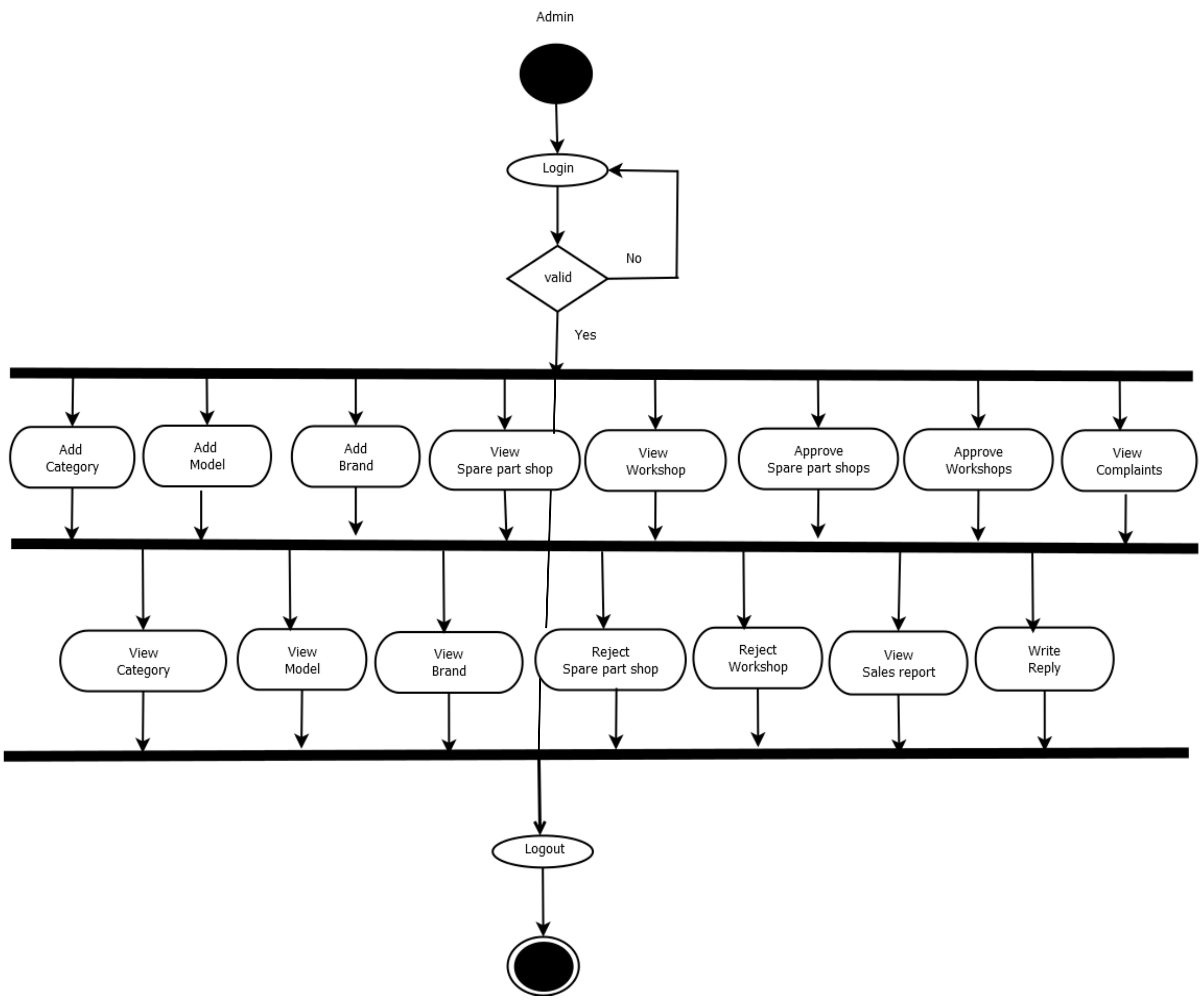
The activity diagram supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario. It uses rounded rectangles to imply a specific system function, arrows to represent flow through the system, decision diamonds to depict a branching decision, and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

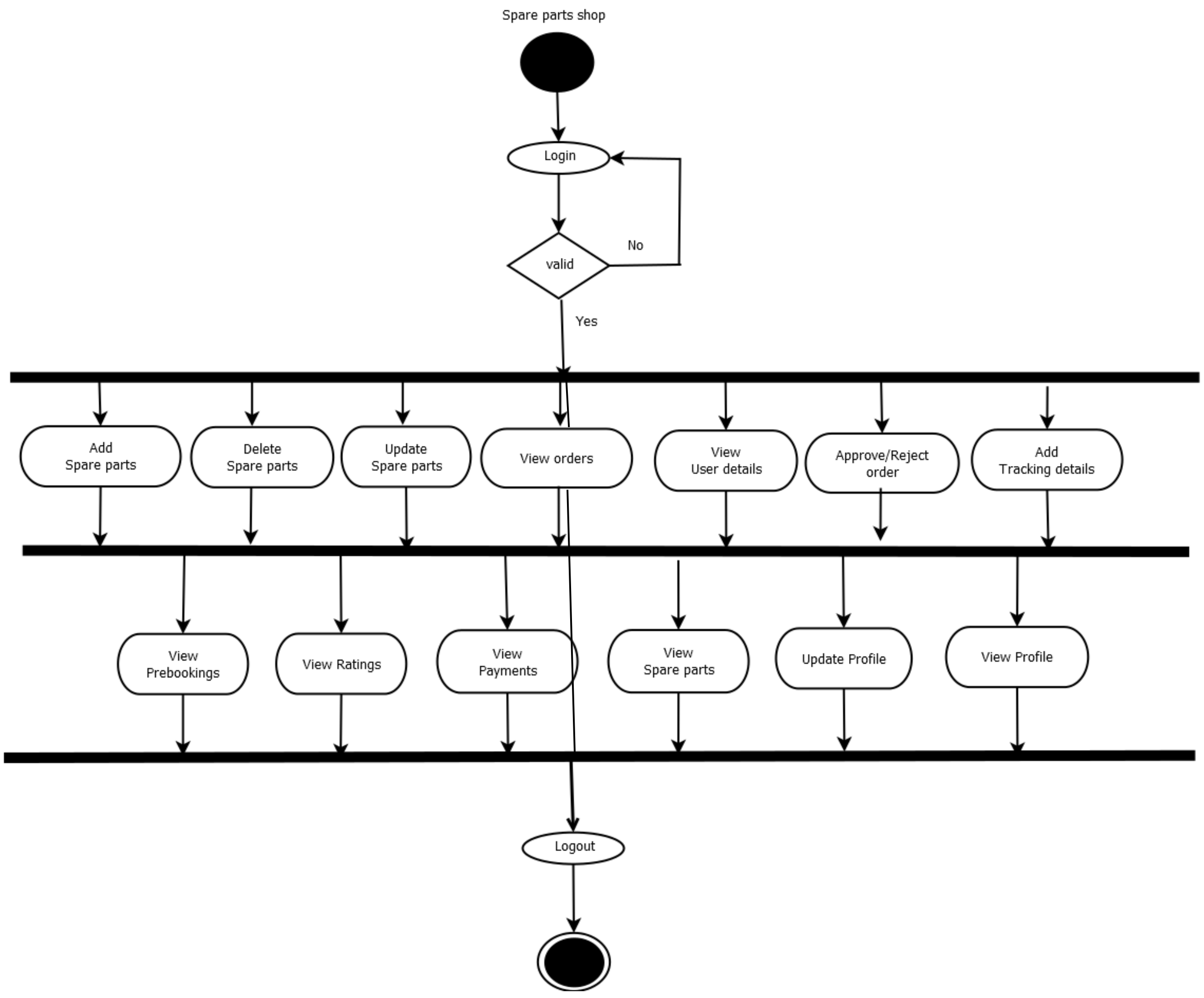
So the purposes can be described as:

- 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.

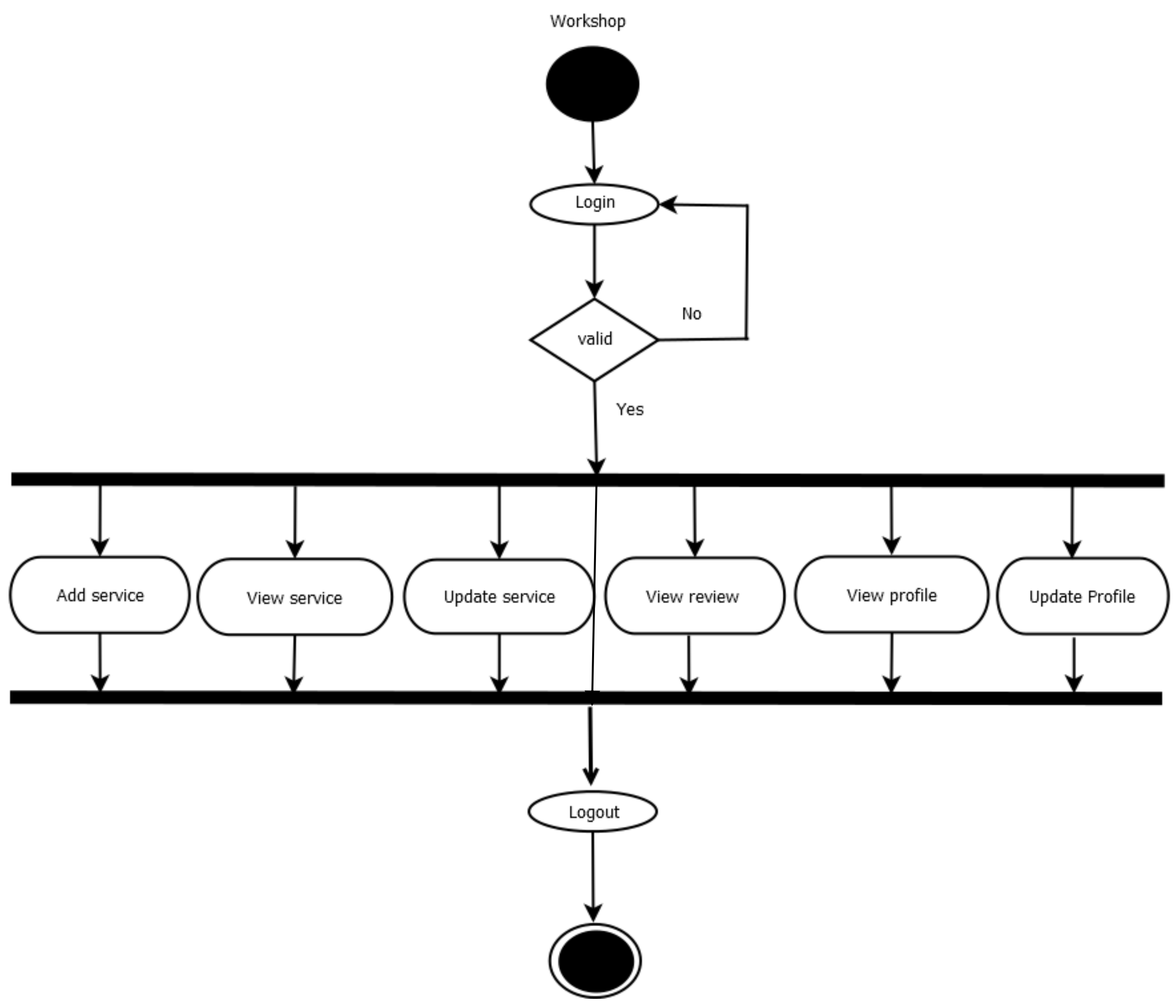
Activity Diagram for Administrator



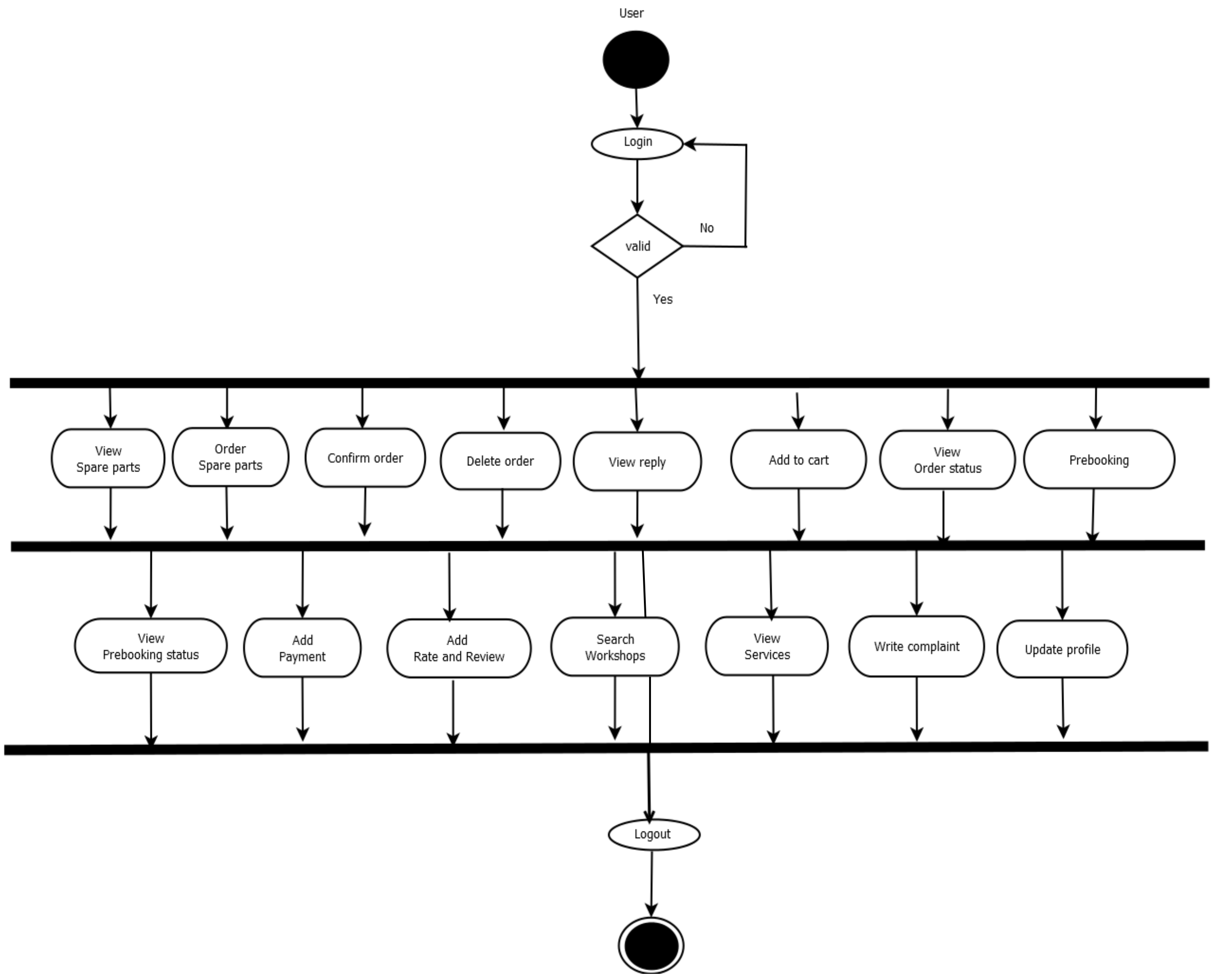
Activity Diagram for Spare part shop



Activity Diagram for Workshop



Activity Diagram for User



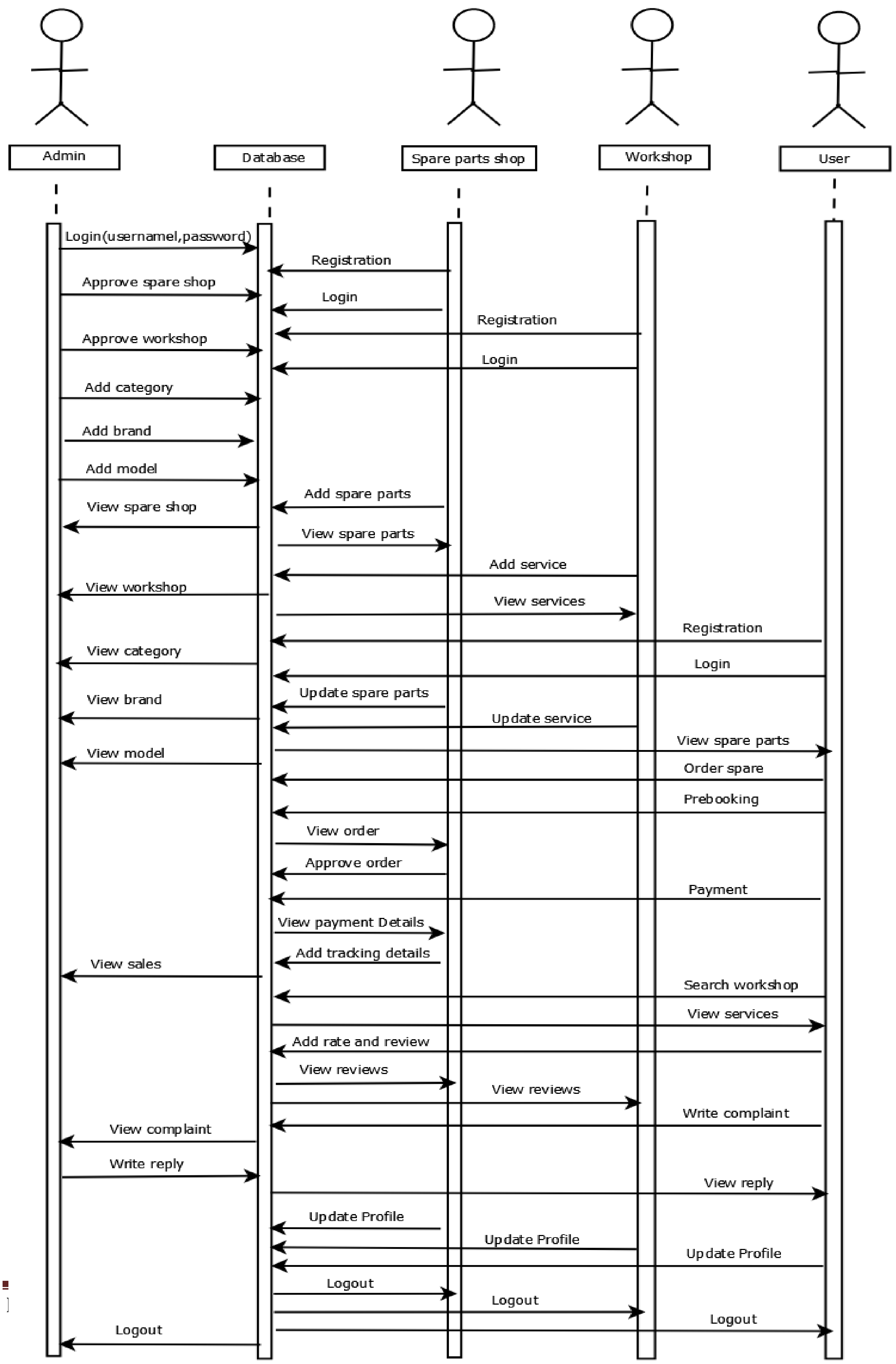
2.4.4. SEQUENCE DIAGRAM

Sequence diagrams are an easy and intuitive way of describing the behaviour 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.


Admin
Database
Spare parts shop
Workshop
User
Login(usernamel,password)
Registration
Approve spare shop
Login
Registration
Approve workshop
Login
Add category
Add brand
Add model
Add spare parts
View spare shop
View spare parts
Add service
View workshop
View services
Registration
View category
Login
Update spare parts
View brand
Update service
View spare parts
View model
Order spare
Prebooking
View order
Approve order
Payment
View payment Details
Add tracking details
View sales
Search workshop
View services
Add rate and review
View reviews
View reviews
Write complaint
View complaint
Write reply
View reply
Update Profile
Update Profile
Update Profile
Logout
Logout
Logout
Logout

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. **<u>HARDWARE REQUIREMENTS</u>**

➢ Processor	: Dual Core
➢ Speed	: 2 GHz
➢ RAM	: 2 GB
➢ Hard Disk	: 40 GB or Above

2. SOFTWARE REQUIREMENTS

➢ Operating System	: Windows 10
➢ Front End	: HTML, CSS, JavaScript
➢ Back End	: SQLite
➢ Programming Language	: Python Django
➢ Development Platform	: Visual Studio Code

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

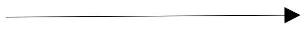
- Easy to use.
- Data independence.
- Accuracy and integrity.
- Avoiding inordinate delays.
- Recovery from failures.

3.1.1 DATA FLOW DIAGRAM

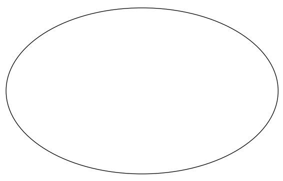
A data flow diagram is a graphical technique that depicts data flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A Level 0 DFD also called a fundamental system model or context model represents the 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:

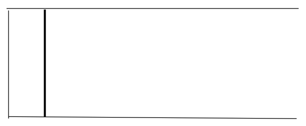
Source/Destination of Data



Data Flow

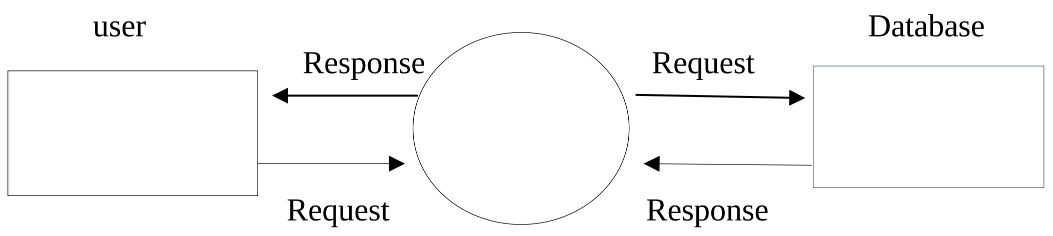


Process

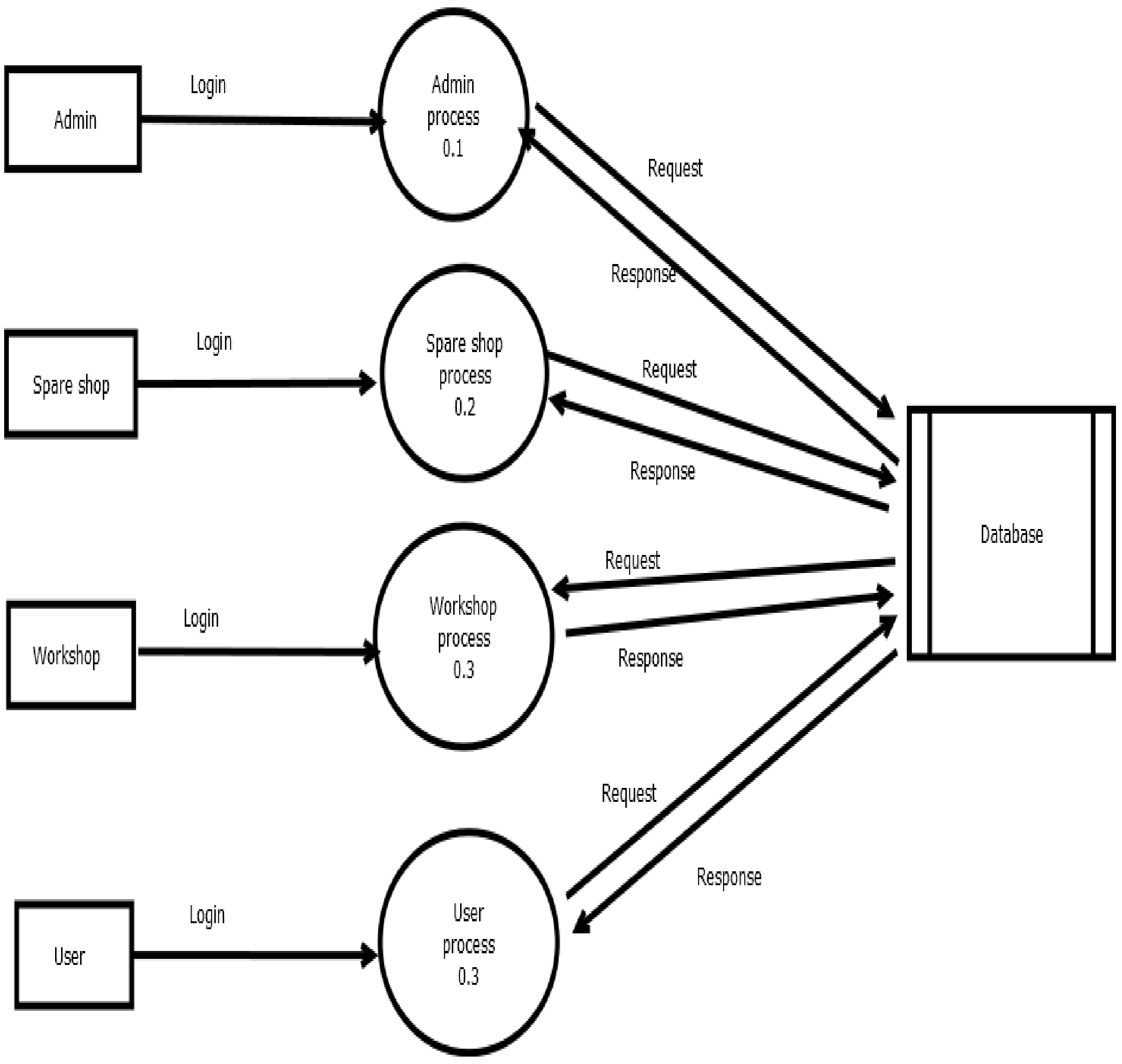


Storage

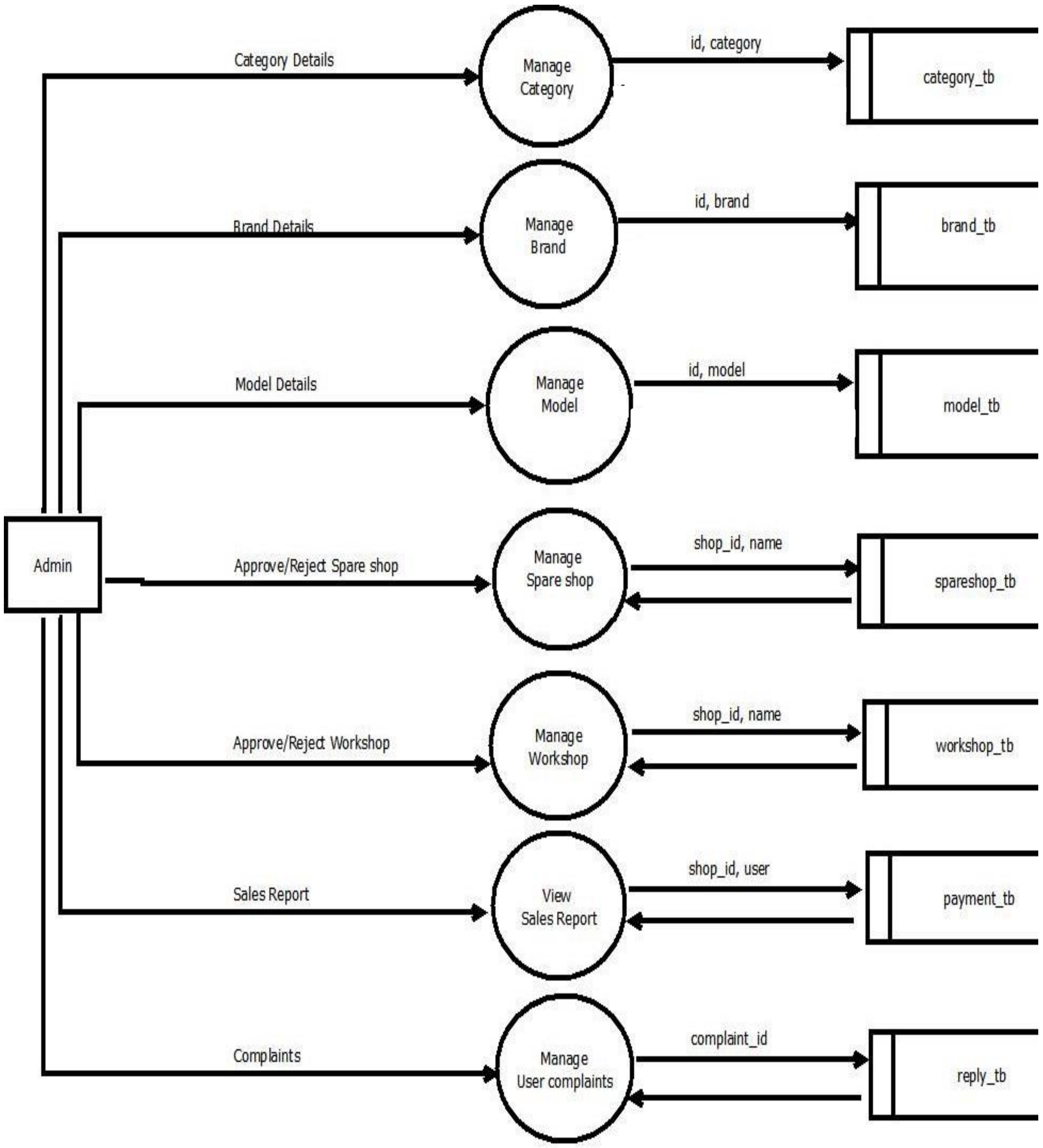
Level 0



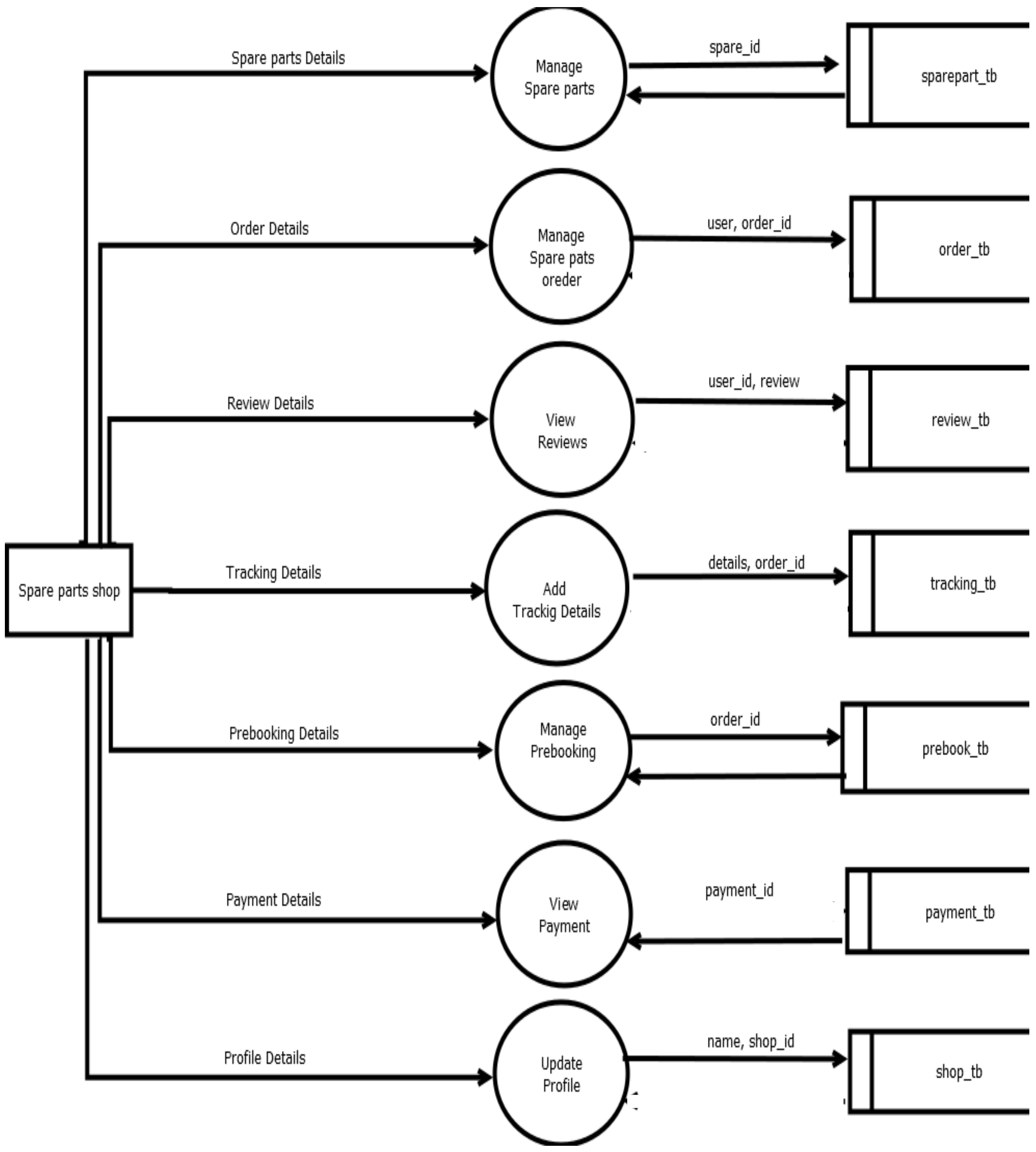
Level 1:


Admin
Login
Admin process 0.1
Request
Response
Spare shop
Login
Spare shop process 0.2
Request
Response
Database
Workshop
Login
Request
Workshop process 0.3
Response
Request
Response
User
Login
User process 0.3

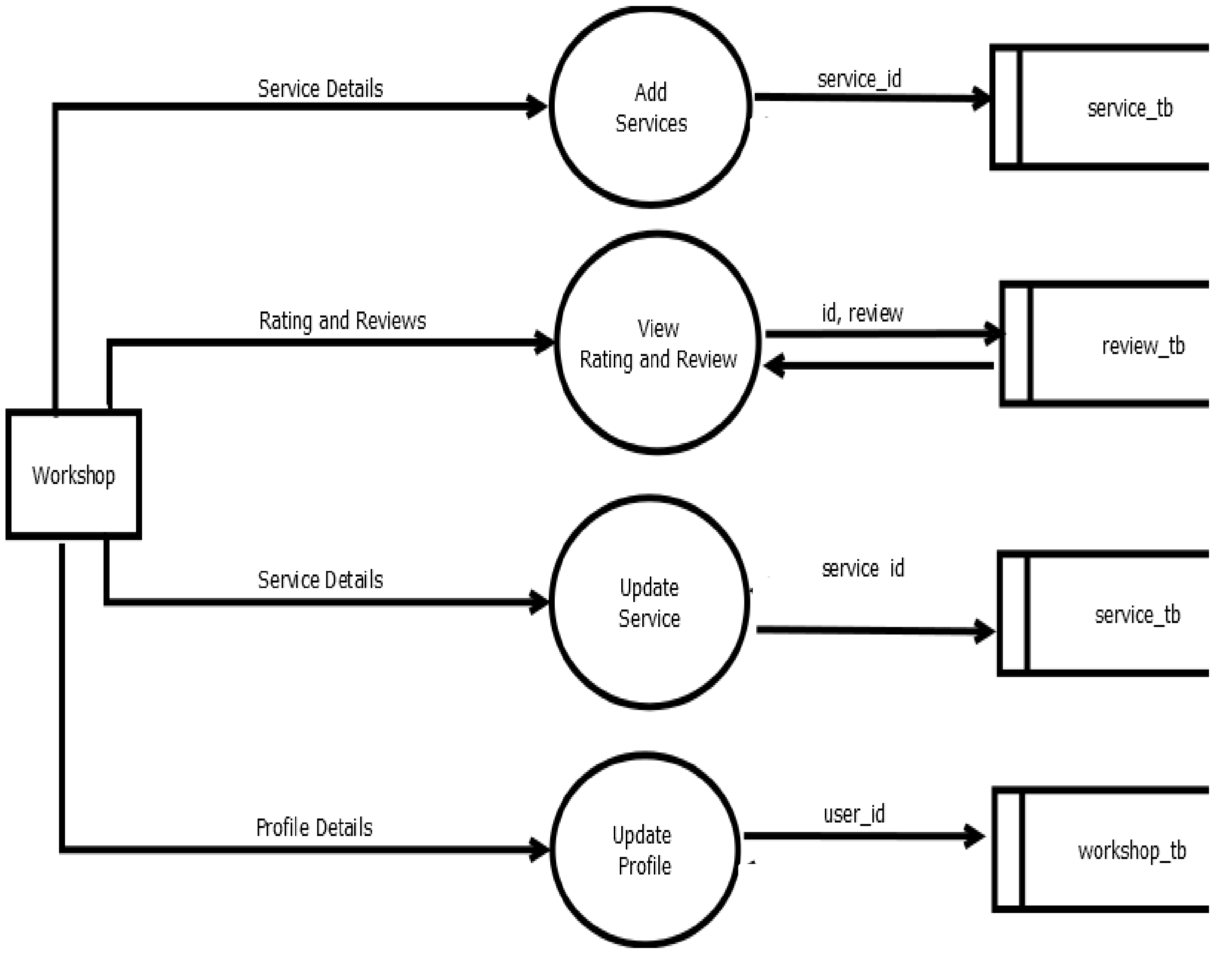
Level 2: Admin



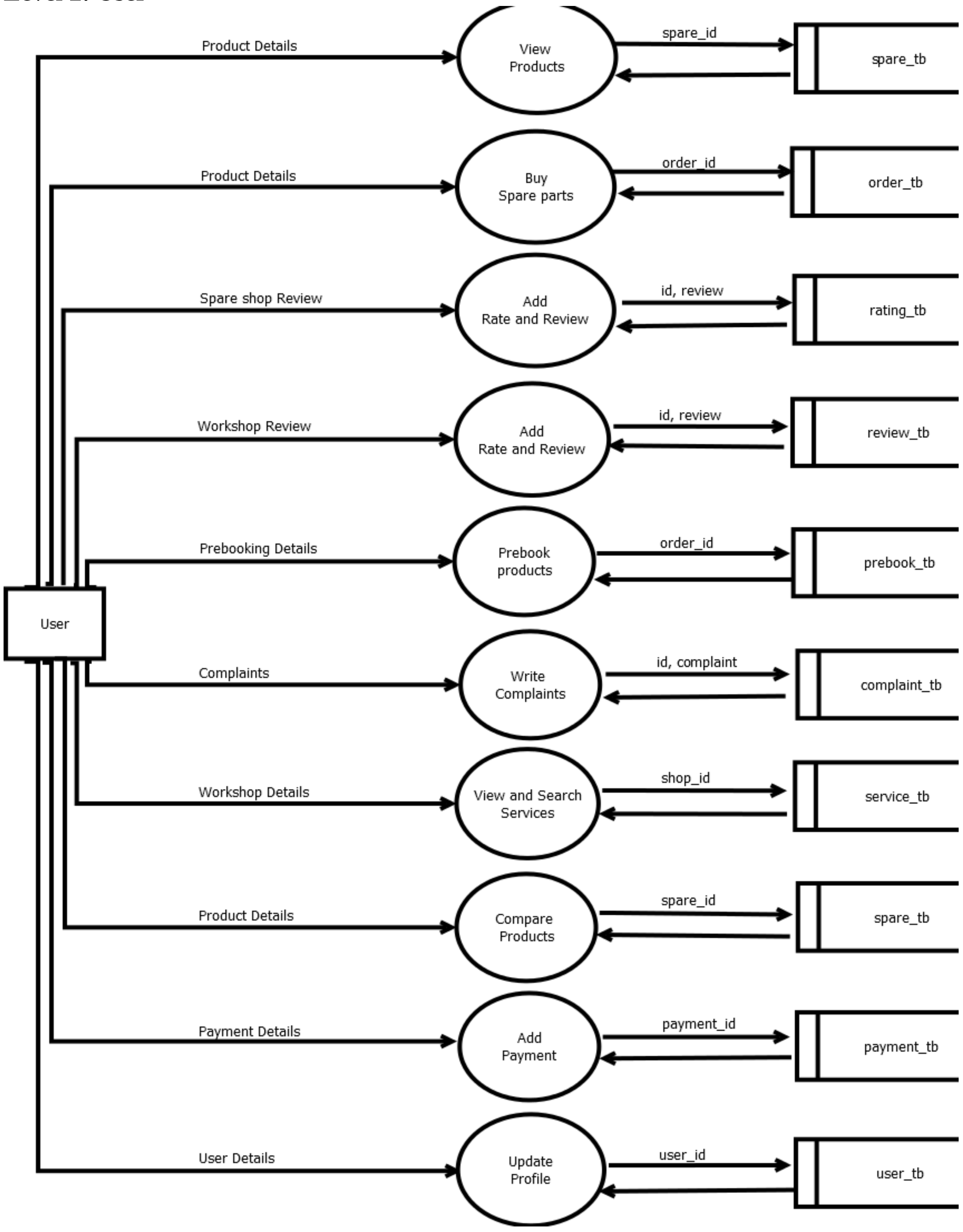
Level 2: Spare Parts Shop



Level 2: Workshop



Level 2: User

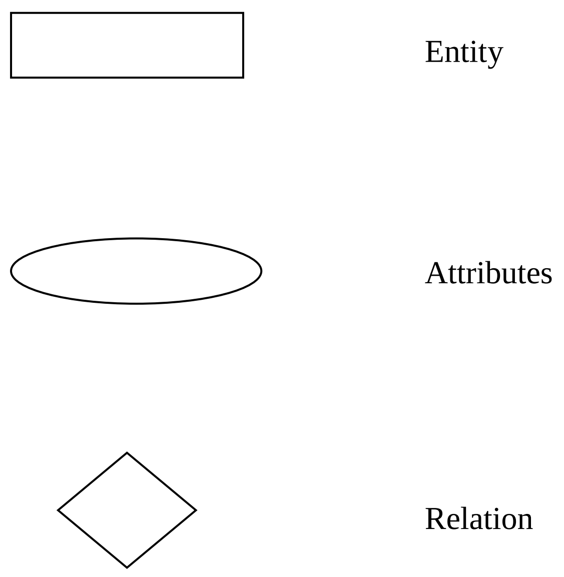

User
Product Details
View Products
spare_id
spare_tb
Product Details
Buy Spare parts
order_id
order_tb
Spare shop Review
Add Rate and Review
id, review
rating_tb
Workshop Review
Add Rate and Review
id, review
review_tb
Prebooking Details
Prebook products
order_id
prebook_tb
Complaints
Write Complaints
id, complaint
complaint_tb
Workshop Details
View and Search Services
shop_id
service_tb
Product Details
Compare Products
spare_id
spare_tb
Payment Details
Add Payment
payment_id
payment_tb
User Details
Update Profile
user_id
user_tb

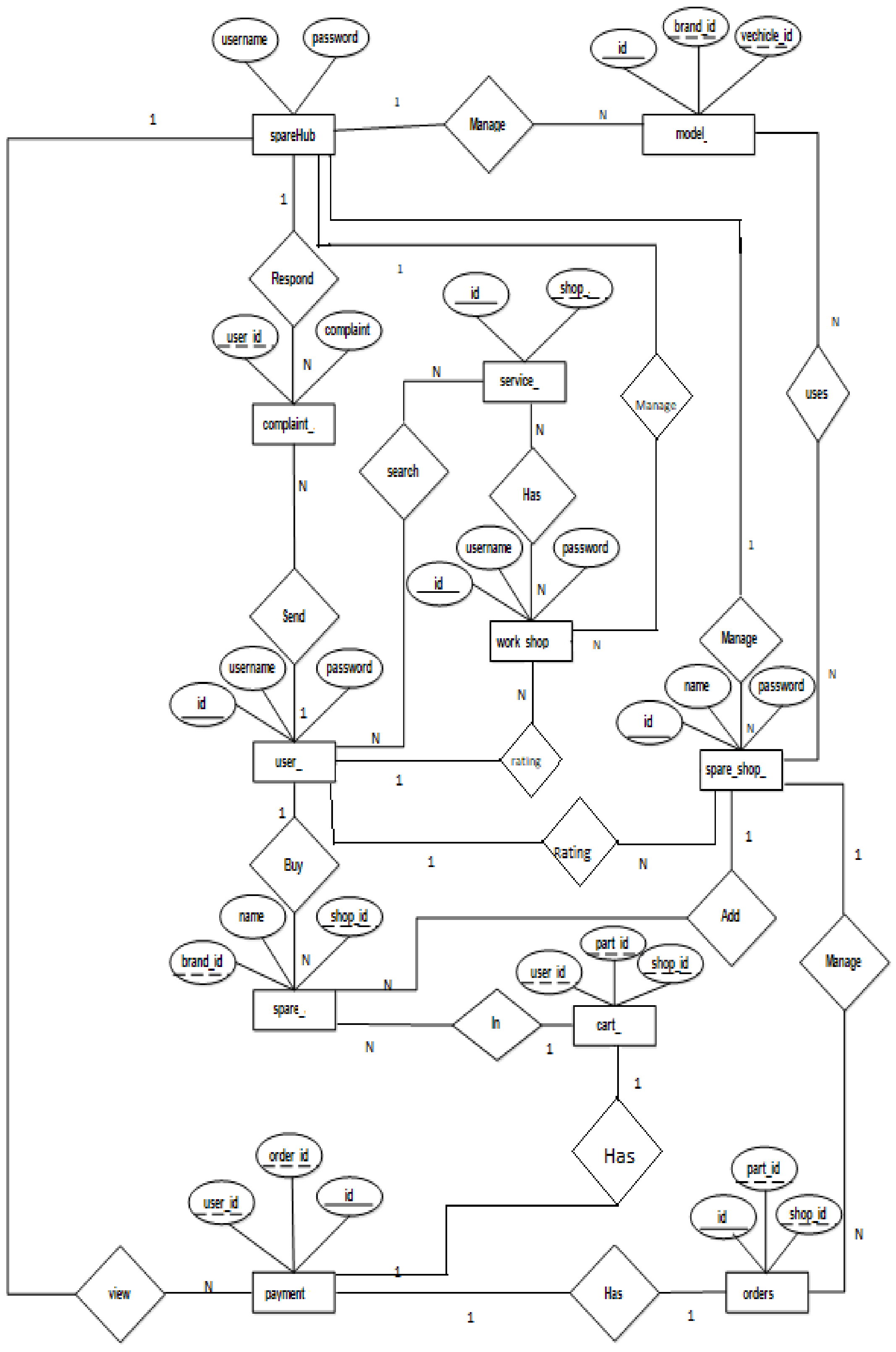
3.1.2 E-R DIAGRAM

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.




username
password
spareHub
Manage
model_
id
brand_id
vechicle_id
Respond
user_id
complaint
complaint_
Send
username
password
id
user_
service_
id
shop_
search
Has
username
password
id
work shop
Manage
uses
Manage
name
password
id
spare_shop_
rating
Rating
Buy
name
shop_id
brand_id
spare_
Add
part_id
user_id
shop_id
cart_
In
Has
order_id
user_id
id
payment
view
Has
part_id
id
shop_id
orders
Manage
1
N

3.1.3 TABLE DESIGN

In the database all the information are 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.
Sparepart_tb	Store spare parts details.
Order_tb	Store spare parts ordering details
Complaint_tb	Store user complaints
Servive_tb	Store workshop service details
Payment_tb	Store payment details
Reply_tb	Store reply messages
Prebook_tb	Store pre booking details
Review_tb	Store user reviews

1. Table Name : User_tb

Primary Key(*) :userid

Field	DataType	Description
userid(*)	Integer	User Id
Path	Varchar(50)	Photo
Name	Varchar(20)	Name
Gender	Varchar(20)	Gender
Address	Varchar(20)	Address
Email	Varchar(20)	Mail ID
User name	Varchar(20)	User name
Password	Varchar(20)	Password

2. Table Name : Sparepart_tb

Promary Key(*) : Part Id

Foreign Key (**) : **Shop Id**

Field	DataType	Description
Part id	Integer	Part ID
Shop id	Integer	Shop ID
Part name	Varchar(20)	Part name
Price	Varchar(20)	Price details
Stock	Varchar(20)	Stock details
Details	Varchar(20)	Product details

3. Table Name : Order_tb

Primary Key(*) : Order Id

Field	DataType	Description
Id	Integer	Order ID
Phone	Varchar(20)	Phone number
Address	Varchar(20)	Address
Count	Varchar(20)	Total number
Part id	Integer	Part ID
Shop id	Integer	Shop ID
Time	Varchar(20)	Ordering time
Prebook id	Integer	Pre booking ID

4.Table Name : Complaint_tb

Field	DataType	Description
Id	Integer	User Id
Subject	Varchar(20)	Subject
Complaint	Varchar(50)	Complaint details
Date	Varchar(20)	Date
User id	Integer	User ID

5.Table Name : Service_tb

Field	DataType	Description
id	Integer	Service ID
Service	Varchar(20)	Service details
Discription	Varchar(200)	Service discription
Status	Varchar(20)	Service availability
Shop id	Integer	Shop ID
Image	Varchar(100)	Images

6.Table Name : Payment_tb

Field	DataType	Description
Id	Integer	Payment ID
Amount	Varchar(20)	Amount details
Transaction key	Varchar(20)	Bank details
Order id	Integer	Order ID
Date	Varchar(20)	Date
User id	Integer	User ID
Shop id	Integer	Shop ID
Status	Varchar(20)	Payment status

7.Table Name : Review_tb

Field	DataType	Description
Id	Integer	Review ID
Rating	Varchar(20)	Rating
Review	Varchar(30)	Review details
Shop id	Integer	SHop ID
User id	Integer	User ID
Date	Varchar(20)	Date

8.Table Name : Reply_tb

Field	DataType	Description
Id	Integer	Reply ID
Subject	Varchar(30)	Subject details
Reply	Varchar(30)	Reply details
Date	Varchar(20)	Date
Complaint id	Integer	Complaint ID
Status	Varchar(10)	Status
User id	Integer	User ID

+

3. Table Name : Prebook_tb

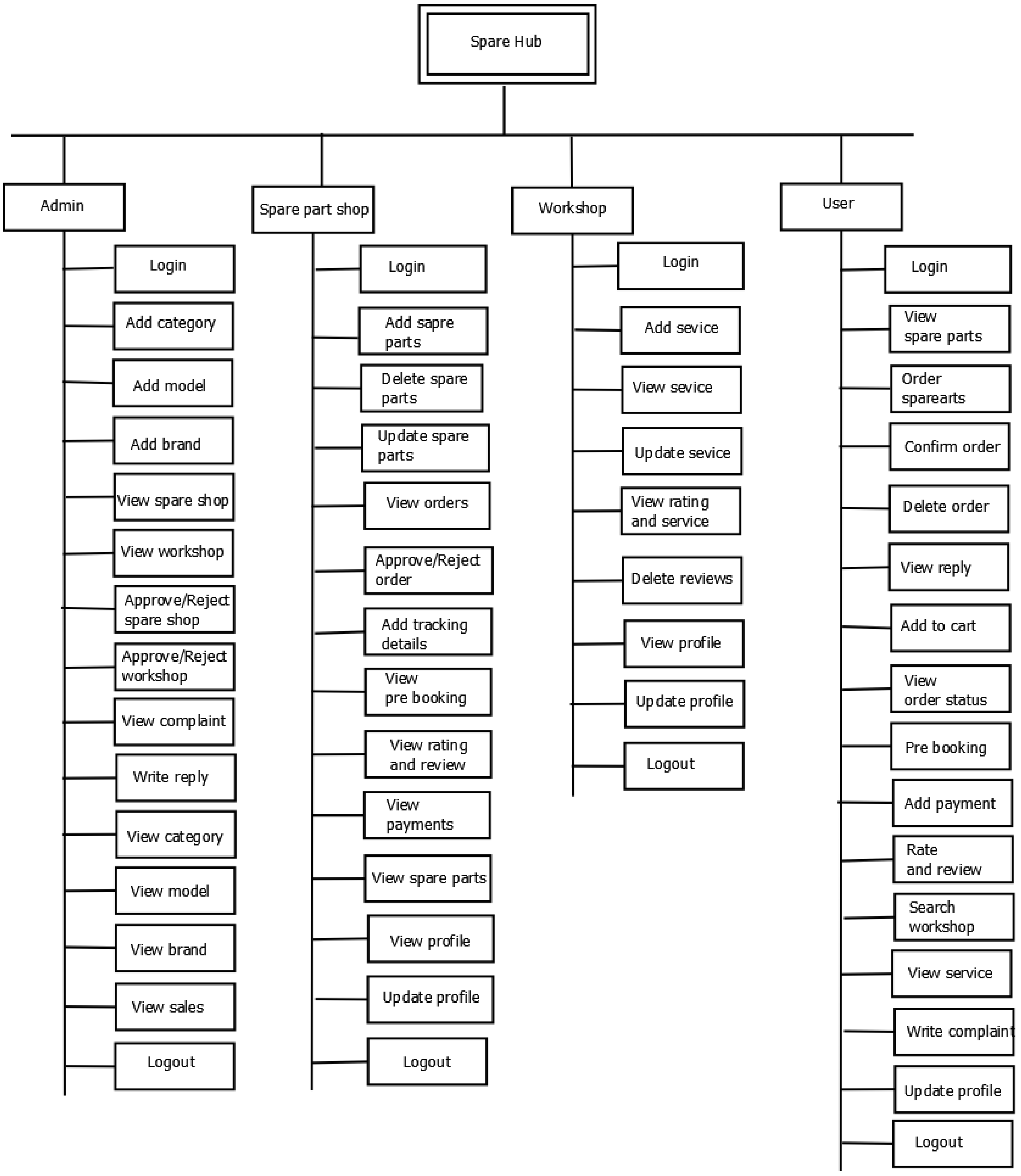
Field	DataType	Description
Id	Integer	Pre booking ID
Part id	Integer	Spare part ID
Shop id	Integer	Spare part shop ID
User id	Integer	User ID
Count	Varchar(20)	Total number
Date	Varchar(20)	Date
Status	Varchar(20)	Order status
Time	Varchar(20)	Time of booking

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 ***"Spare Hub",*** 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, Spare part shop and Workshop registration form
- Write complaints
- Add category, brand and model
- Add spare parts
- Add service
- 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 spare shop and work shop details
- Form for viewing spare parts
- Form for viewing workshop services
- Form for viewing user details
- Form for viewing rate and reviews
- Form for viewing order details
- Form for viewing sales details
- Form for viewing notifications

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
- Spare part shop
- Workshop
- User

1. Admin

Admin has the overall control of the system. Admin can manage spare part shops and workshops, manage category, analyzing report, view payment details, manage complaints and manage overall control.

2. Spare part shop

They can login, manage spare parts, order management, view rating and reviews and updating spare part details and profile.

3. Workshop

They can login, manage workshop and services, view ratings and reviews and updates services and profile.

4. Workshop

They can login, and view spare parts and workshop services, order spare parts, search services, add rating and review for spare parts and workshop services, write complaints and profile updation.

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.

<u>Characteristics of Python</u>

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

DJANGO

Django is a Python-based free and open-source web framework that follows the model–template–views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "plug ability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

4.1.2 SQLite

SQLite is a relational database management system contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite generally follows PostgreSQL syntax. SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions and will store the data as-is if such a conversion is not possible.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

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.
- Don’t 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. shop_signup.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
                    <script src="/static/jquery.min.js"></script>
                    <script>
                              $(document).ready(function(){
                                        var password=document.getElementById("password");
                                        var
confirm_password=document.getElementById("confirm_password");
                                        function validatePassword(){
                                                  if(password.value != confirm_password.value){

                    confirm_password.setCustomValidity("Passwords Don't Match");
                                                  }
                                                  else{
                                                            confirm_password.setCustomValidity('');
                                                  }
                                        }
                                        password.onchange=validatePassword;
                                        confirm_password.onkeyup=validatePassword;


                                        flag=0;
                                        $("#formsignup").submit(function(e){

                                                  if(flag == 0){

                                                            e.preventDefault();
                                        $.ajax({
                                             url:"{% url 'checkUsername' %}",
                                             data:{'username':$("#username").val()},
                                             success:function(data){
                                                  if(data.status == "exists"){
                                                       alert('Username    Already
Exists');

                                                  }
                                                  else{
                                                       flag=1;

          $("#formsignup").submit();
                                                  }

                                             }
                                        });
                                   }

                              });

                         });

          </script>
          <div class="span9">
          <ul class="breadcrumb">
               <li>Home<span class="divider">/</span></li>
               <li class="active">Spare Part Shop Registration</li>
          </ul>
          <h3>Spare Part Shop Registration</h3>
          <hr class="soft"/>
          <div class="well"  style="background:#202020;color:white">
          <br/>
          <form    class="form-horizontal"    method="post"    action="{%    url
'shopSignupAction' %}" id="formsignup" enctype="multipart/form-data">
          {% csrf_token %}
               <div class="control-group">
                    <label                                  class="control-label"
for="logo">Logo<sup>*</sup></label>
                    <div class="controls">
                      <input type="file" name="logo" id="logo" required/>
```

```
                        </div>
                </div>

                <div class="control-group">
                        <label class="control-label" for="shopname">Shop Name<sup>*</sup></label>
                        <div class="controls">
                          <input type="text" name="shopname" id="shopname" placeholder="Shop Name" required/>
                        </div>
                </div>

                <div class="control-group">
                        <label class="control-label" for="address">Address<sup>*</sup></label>
                        <div class="controls">
                          <textarea name="address" id="address" rows="4" placeholder="Address" required></textarea>
                        </div>
                </div>
                <div class="control-group">
                        <label class="control-label" for="address">State<sup>*</sup></label>
                        <div class="controls">
                          <input type="text" value="Kerala" readonly="true"/>
                        </div>
                </div>
                <div class="control-group">
                        <label class="control-label" for="district">District<sup>*</sup></label>
                        <div class="controls">
                         <select name="district" id="district" required>
                                <option value="">--select--</option>
                                {% for d in districts %}
                                        <option value="{{d.id}}">{{d.district_name}}</option>
                                {% endfor %}
                          </select>
                        </div>
                </div>
                <div class="control-group">
```

```
                                        <label                                          class="control-label"
for="place">Place<sup>*</sup></label>
                                        <div class="controls">
                                          <input        type="text"        name="place"        id="place"
placeholder="Place" required/>
                                        </div>
                              </div>
                              <div class="control-group">
                                        <label                                          class="control-label"
for="proof">Proof<sup>*</sup></label>
                                        <div class="controls">
                                          <input type="file" name="proof" id="proof" required/>
                                        </div>
                              </div>
                              <div class="control-group">
                                        <label                                          class="control-label"
for="phone">Phone<sup>*</sup></label>
                                        <div class="controls">
                                          <input        type="text"        name="phone"        id="phone"
pattern="^\d{10}$" placeholder="Phone" required/>
                                        </div>
                              </div>
                              <div class="control-group">
                                        <label                                          class="control-label"
for="email">Email<sup>*</sup></label>
                                        <div class="controls">
                                          <input        type="email"        name="email"        id="email"
placeholder="Email" required/>
                                        </div>
                              </div>
                              <div class="control-group">
                                        <label                                          class="control-label"
for="username">Username<sup>*</sup></label>
                                        <div class="controls">
                                          <input     type="text"     name="username"     id="username"
placeholder="Username" required/>
                                        </div>
                              </div>
                              <div class="control-group">
                                        <label                                          class="control-label"
for="password">Password<sup>*</sup></label>
```

```
                                        <div class="controls">
                                          <input type="password" name="password" id="password"
pattern=".{8,}" title="Eight or more Characters" placeholder="Password" required/>
                                        </div>
                               </div>
                              <div class="control-group">
                                        <label class="control-label" for="confirm_password">Confirm
Password<sup>*</sup></label>
                                        <div class="controls">
                                          <input type="password" name="confirm_password"
id="confirm_password" placeholder="Confirm Password" required/>
                                        </div>
                               </div>
                              <div class="control-group">
                                        <div class="controls">

                                                  <input class="btn btn-large btn-success" id="signup"
type="submit" value="Register" />
                                        </div>
                              </div>
                    </form>
</div>
</div>
                              {% if messages %}
                                        {% for message in messages %}
                                        <script>
                                                  alert('{{message}}');
                                        </script>
                                        {% endfor %}
                              {% endif %}
{% endblock %}
```

views.py

```
def shopSignupAction(request):
    logo_img=""
    proof=""
    if(len(request.FILES)>0):
        logo_img=request.FILES['logo']
        proof=request.FILES['proof']
    else:
        logo_img="no logo"
        proof="no proof"
    did=district_tb.objects.get(id=request.POST['district'])

shop=shop_tb(logo=logo_img,shop_name=request.POST['shopname'],phone=request.POST['
phone'],email=request.POST['email'],address=request.POST['address'],

district_id=did,place=request.POST['place'],proof=proof,username=request.POST['username'
],password=request.POST['password'],
                    status='pending')
    shop.save()

shop=User.objects.create_user(username=request.POST['username'],password=request.POST
['password'])
    shop.save()
    messages.add_message(request,messages.INFO,"Signup successful")
    return redirect('shopSignup')
```

2. order_details.html

```
{% extends 'base.html' %}
{% block content %}


   <div class="span9">
    <ul class="breadcrumb">
    <li>Home<span class="divider">/</span></li>
    <li>Spare Parts<span class="divider">/</span></li>
    <li class="active">product Details</li>
    </ul>
                    <h3> Spare Part Details </h3>
                    <hr class="soft"/>
                    <div class="tab-content">
                    <div class="row" style="background:#202020;color:white;padding-left:5em">
                            <br/><br/>
                        {% for v in data %}
                                <div id="gallery" class="span3">
```

```
                                        <a href="{{v.path.url}}"><img class="img-
resize-details" src="{{v.path.url}}" style="width:100%" alt="no pic"/></a>

                                </div>
                                <div class="span6">
                                        <h3>{{v.part_name}}</h3>
                                        <hr class="soft"/>
                                          <div class="control-group">
                                                <label class="control-label"><span style="font-
size:20px">{{v.price}} .RS</span></label>
                                                <div class="controls">
                                                {% if 'user_id' in request.session %}
                                                  <a class="btn btn-large btn-primary pull-right"
href="{% url 'addToCart' v.id %}"> Add to cart <i class=" icon-shopping-cart"></i></a>
                                                {% else %}
                                                  <a class="btn btn-large btn-primary pull-right"
href="{% url 'login' %}"> Add to cart <i class=" icon-shopping-cart"></i></a>
                                                {% endif %}
                                                </div>
                                          </div>




                                        <br class="clr"/>
                                </div>
                                <br/>
                                <br/>

                                <div class="span9">
          <div id="myTabContent" class="tab-content">
            <div class="tab-pane fade active in" id="home" style="padding-right:130px">
                         <br/>
              <table class="table table-bordered">
                                        <tbody>
                                        <tr class="techSpecRow"><th colspan="2">Spare Part
Details</th></tr>
                                        <tr class="techSpecRow"><td
class="techSpecTD1">Brand: </td><td
class="techSpecTD2">{{v.brand_id.brand_name}}</td></tr>
                                        <tr class="techSpecRow"><td
class="techSpecTD1">Model:</td><td class="techSpecTD2">{% for p in models
%}{{p.model_id.model_name}}{% if forloop.counter != models.count %},{% endif %}{%
endfor %}</td></tr>
                                        {% if v.stock == '0' %}
                                        <tr                             class="techSpecRow"><td
class="techSpecTD1">Stock:</td><td class="techSpecTD2">{{v.stock}}</td></tr>
                                        {% else %}
                                        <tr                             class="techSpecRow"><td
class="techSpecTD1">Stock:</td><td class="techSpecTD2">{{v.stock}}</td></tr>
                                        {% endif %}
                                        <tr                             class="techSpecRow"><td
class="techSpecTD1">Seller:</td><td
class="techSpecTD2">{{v.shop_id.shop_name}}</td></tr>
                                        <tr                             class="techSpecRow"><td
class="techSpecTD1">Seller                                                Address:</td><td
class="techSpecTD2">{{v.shop_id.address}}</td></tr>
                                        </tbody>
                                        </table>




         </div>
                        <h5>Features</h5>
                <p class="p-restrict">

                        {{ v.details }}
                </p>
                <br/><br/>
                </div>

</div>
{% endfor %}
</div>
</div>
</div>
{% endblock %}
```

views.py

```
def selectOrder(request,cid):
   cart_obj=cart_tb.objects.filter(id=cid)
   part_model=part_model_tb.objects.filter(part_id=cart_obj[0].part_id)
```

```
    all_cart=cart_tb.objects.filter(user_id=request.session['user_id'])
    return
render(request,'view_cart.html',{'data':all_cart,'cart':cart_obj,'part_model':part_model})
```

3. write_complaint.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
                <div class="span9">
    <ul class="breadcrumb">
                        <li>Home<span class="divider">/</span></li>
                        <li class="active">Complaint</li>
    </ul>
                <h3> Complaint</h3>
                <hr class="soft"/>
                <div class="well" style="background:#202020;color:white;">
                <br/><br/>
                <form class="form-horizontal" method="post" action="{% url
'writeComplaintAction' %}">
                {% csrf_token %}
                        <div class="control-group">
                                <label class="control-label"
for="subject">Subject<sup>*</sup></label>
                                <div class="controls">
                                  <textarea name="subject" id="subject" rows="2"
placeholder="Subject" required></textarea>
                                </div>
                         </div>


                        <div class="control-group">
                                <label class="control-label"
for="complaint">Complaint<sup>*</sup></label>
                                <div class="controls">
                                  <textarea name="complaint" id="complaint" rows="4"
placeholder="Complaint" required></textarea>
                                </div>
                         </div>
                <div class="control-group">
                    <div class="controls">

                        <input class="btn btn-large btn-success" type="submit"
value="Submit" />
                    </div>
                </div>
            </form>
</div>

</div>



                <script src="/static/jquery.min.js"></script>
                {% if messages %}
                    {% for message in messages %}
                    <script>
                        alert('{{message}}');
                    </script>
                    {% endfor %}
                {% endif %}

{% endblock %}
```

Views.py

```
def writeComplaintAction(request):
    uid=user_tb.objects.get(id=request.session['user_id'])

complaint=complaint_tb(user_id=uid,subject=request.POST['subject'],complaint=request.PO
ST['complaint'],date=datetime.date.today())
    complaint.save()
    messages.add_message(request,messages.INFO,"Submitted successfully")
    return redirect('writeComplaint')
```

4. view_services.html

```
{% extends 'base.html' %}
{% load static %}

{% block content %}
```

```
<div class="span9">
   <ul class="breadcrumb">
                         <li>Home<span class="divider">/</span></li>
                         <li>Find Workshops<span class="divider">/</span></li>
                         <li>Workshops<span class="divider">/</span></li>
                         <li class="active">Services</li>
   </ul>
                 <h3> Services </h3>
                 <hr class="soft"/>

                         {% if msg %}
                                 <label>{{msg}}</label>
                         {% else %}
                         <div class="tab-content">
                           <div id="listView" style="background:#202020;color:white;">
                                 <br/>
                                 {% for v in data %}
                                         <div class="row">
                                                 <div class="span2" style="padding-left:5em">
                                                         <a    href="{{v.service_img.url}}"><img
class="img-resize-details" src="{{v.service_img.url}}" alt=""/></a>
                                                 </div>
                                                 <div class="span4" style="padding-left:2em">
                                                         <h3                              style="font-
size:20px">{{v.service}}</h3>
                                                         <hr class="soft"/>
                                                         <p>
                                                                 Description                           :
{{v.description}}<br/><br/>
                                                                 Status   : {{v.status}}

                                                         </p>
                                                 </div>

                                         </div>
                                 <hr class="soft"/>
                                 {% endfor %}
                           </div>

                           <hr class="soft"/>
```

```
                              <div>
                            {% endif %}


                                                {% else %}

<li class="disabled"><span>&raquo;</span></li>
                                                {% endif %}
                                        </ul>
                            {% endif %}
                            </div>
                            <br/>
</div>
</div>
</div>
{% endblock %}
```

Views.py

```
def viewServices(request,wid):
    services=service_tb.objects.filter(shop_id=wid)
    if(services.count()>0):
        page=request.GET.get('page',1)
        paginator=Paginator(services,4)
        try:
            all_services=paginator.page(page)
        except PageNotAnInteger:
            all_services=paginator.page(1)
        except EmptyPage:
            all_services=paginator.page(paginator.num_pages)
        return render(request,'view_services.html',{'data':all_services})
    else:
        return render(request,'view_services.html',{'msg':'No services'})
```

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.
- Creating the tables of the system with actual data.
- 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.
- Evaluation of the changeover method

Technologies used in the development of the software are:

- Development tool: Visual Studio Code 1.72.0
- Language: Python 3.10
- Framework: Django 3.0.5
- Database: SQLite

- Web Server: Chrome
- Scripting: HTML, JavaScript, CSS, bootstrap

CHAPTER-7

MAINTENANCE AND REVIEW

7. 1 MAINTENANCE AND REVIEW

Maintenance is making adaptation of the software for external changes (requirements changes or enhancements) and internal changes (fixing bugs). When changes are made during the maintenance phase all preceding steps of the model must be revisited. There are three types of maintenance:

1. Corrective (Fixing Bugs/errors)
2. Adaptive (Updates due to environment changes)
3. Perfective (Enhancements, requirements changes)

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

Spare Hub has been developed, tested, documented, and implemented successfully. The main objective of the system was brought in to effect. The system is developed in Python Django as front-end tool and SQLite as backend tool.

This application is currently an open one, which promises any number of modules to be integrated along with it. Considering the current trends and the developments the future might offer, this is considered as an excellent system with a promising bright future in the coming Years.

The important benefits that have been found out through the implemented system are:

1. User friendly.
2. Simplified operation.
3. Reduced processing time.
4. Accurate result providing.
5. Increases accuracy.

Any system that has been used for several years gradually decades and becomes less effective because of the changes in environment to which it has to adopt. For a time, it is possible to overcome problems by appending the need of fundamental changes.

CHAPTER 9

REFERENCE

9. REFERENCES

9.1 WEBSITES

- http://www.w3schools.com
- http://www.stackoverflow.com
- http://www.codeproject.com
- https://youtube.com

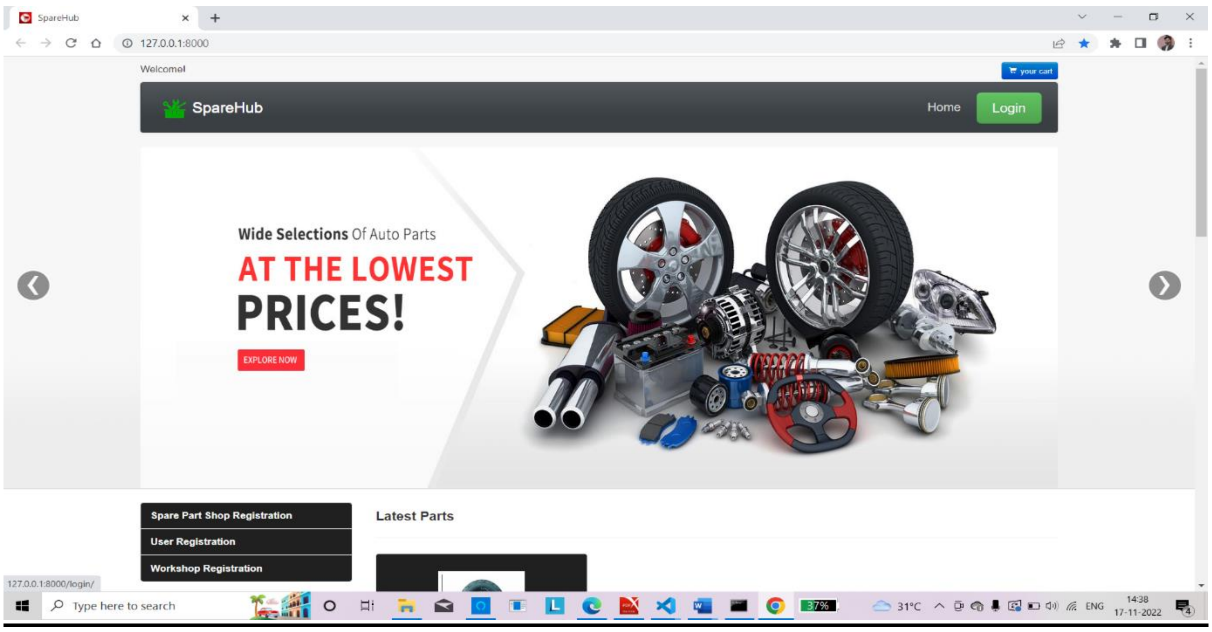
9.2 REFERENCE BOOKS

- 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

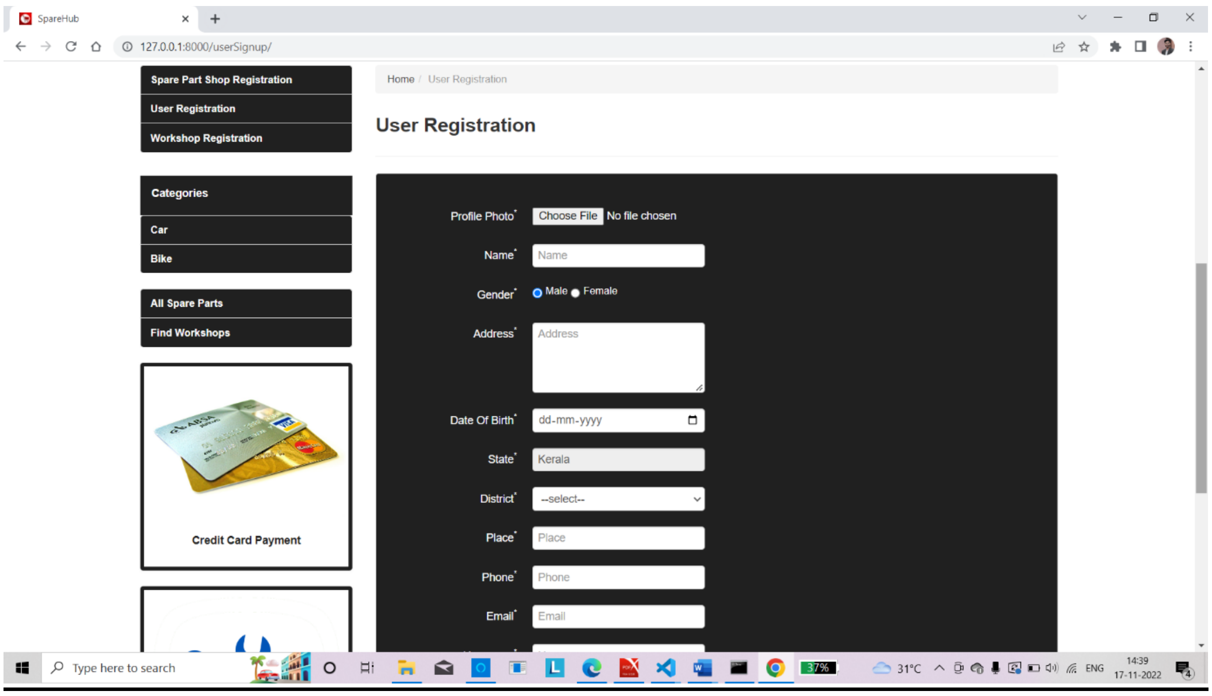
CHAPTER 10
APPENDIX

SAMPLE SCREENSHOTS

Home



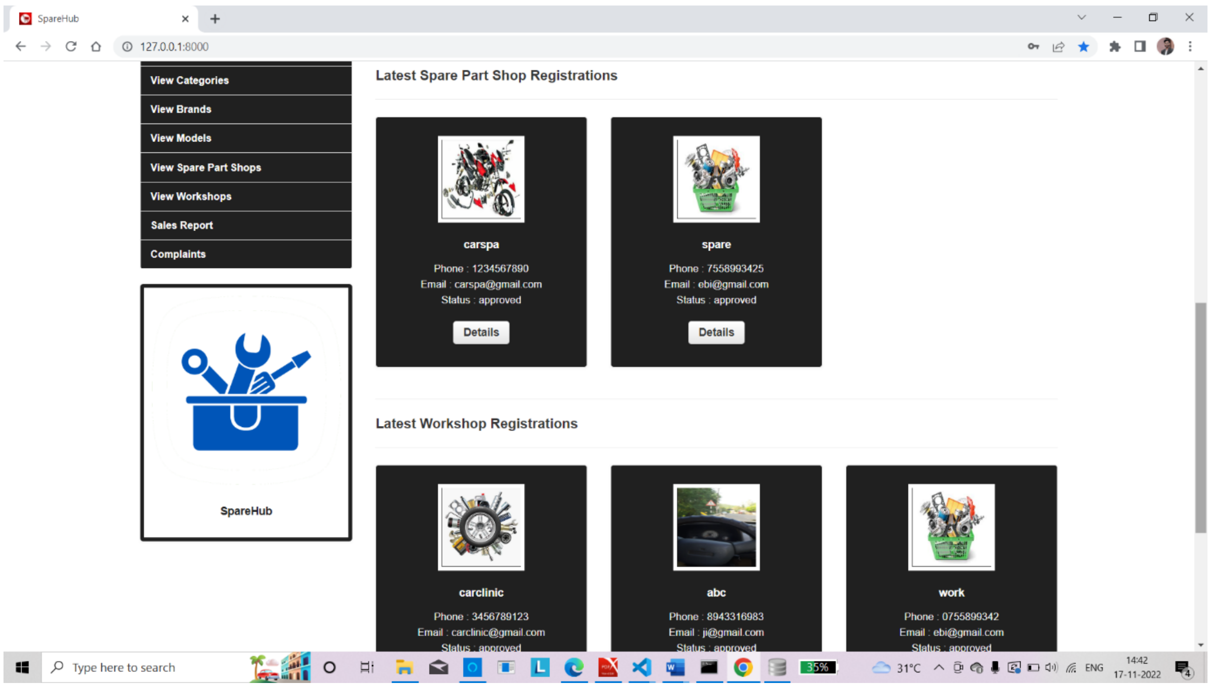
User Registration



User Registration

Profile Photo*	Choose File No file chosen
Name*	Name
Gender*	● Male ● Female
Address*	Address
Date Of Birth*	dd-mm-yyyy
State*	Kerala
District*	--select--
Place*	Place
Phone*	Phone
Email*	Email

Approve/ Reject shops


SpareHub

127.0.0.1:8000

View Categories
View Brands
View Models
View Spare Part Shops
View Workshops
Sales Report
Complaints

SpareHub

Latest Spare Part Shop Registrations

carspa
Phone : 1234567890
Email : carspa@gmail.com
Status : approved
Details

spare
Phone : 7558993425
Email : ebi@gmail.com
Status : approved
Details

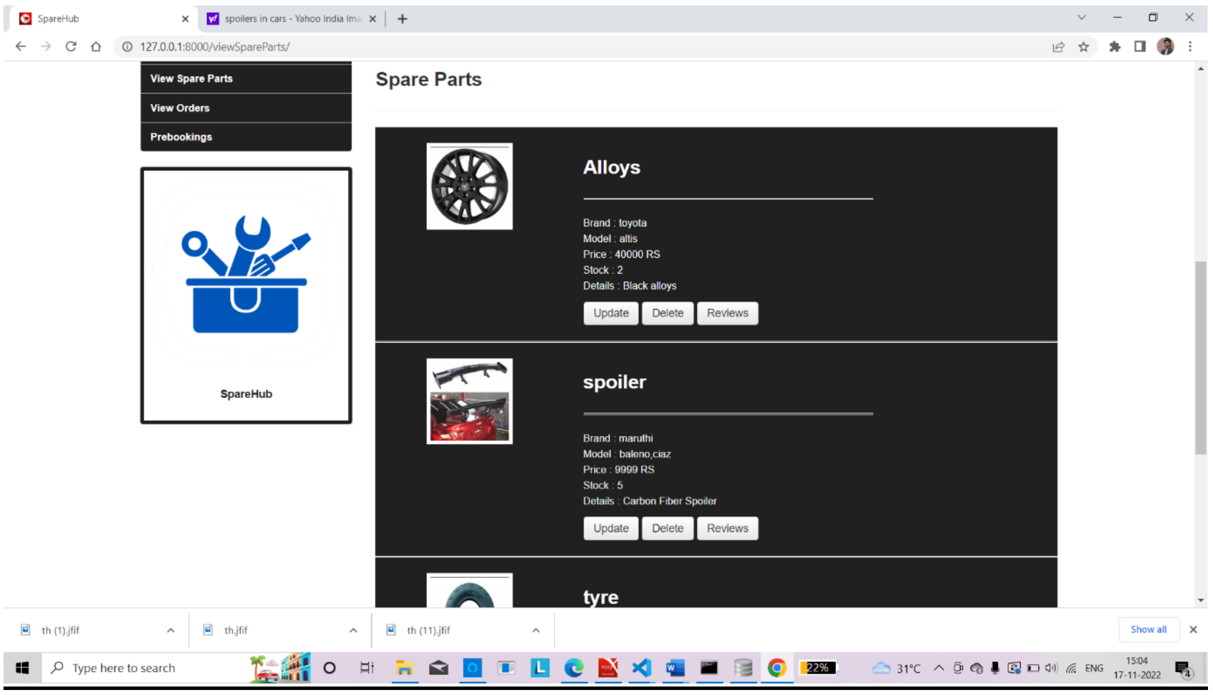
Latest Workshop Registrations

carclinic
Phone : 3456789123
Email : carclinic@gmail.com
Status : approved

abc
Phone : 8943316983
Email : ji@gmail.com
Status : approved

work
Phone : 0755899342
Email : ebi@gmail.com
Status : approved

Order products


SpareHub

spoilers in cars - Yahoo India Ima

127.0.0.1:8000/viewSpareParts/

View Spare Parts
View Orders
Prebookings

SpareHub

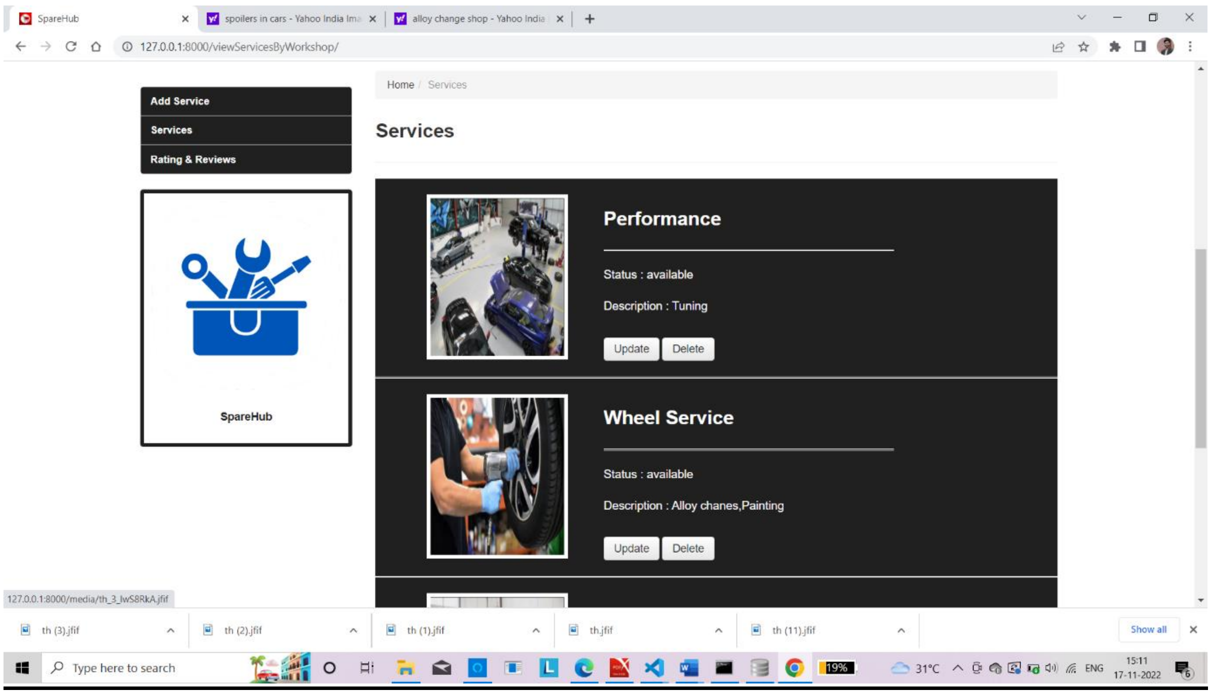
Spare Parts

Alloys
Brand : toyota
Model : altis
Price : 40000 RS
Stock : 2
Details : Black alloys
Update Delete Reviews

spoiler
Brand : maruthi
Model : baleno,ciaz
Price : 9999 RS
Stock : 5
Details : Carbon Fiber Spoiler
Update Delete Reviews

tyre

View services



Services

Performance

Status : available

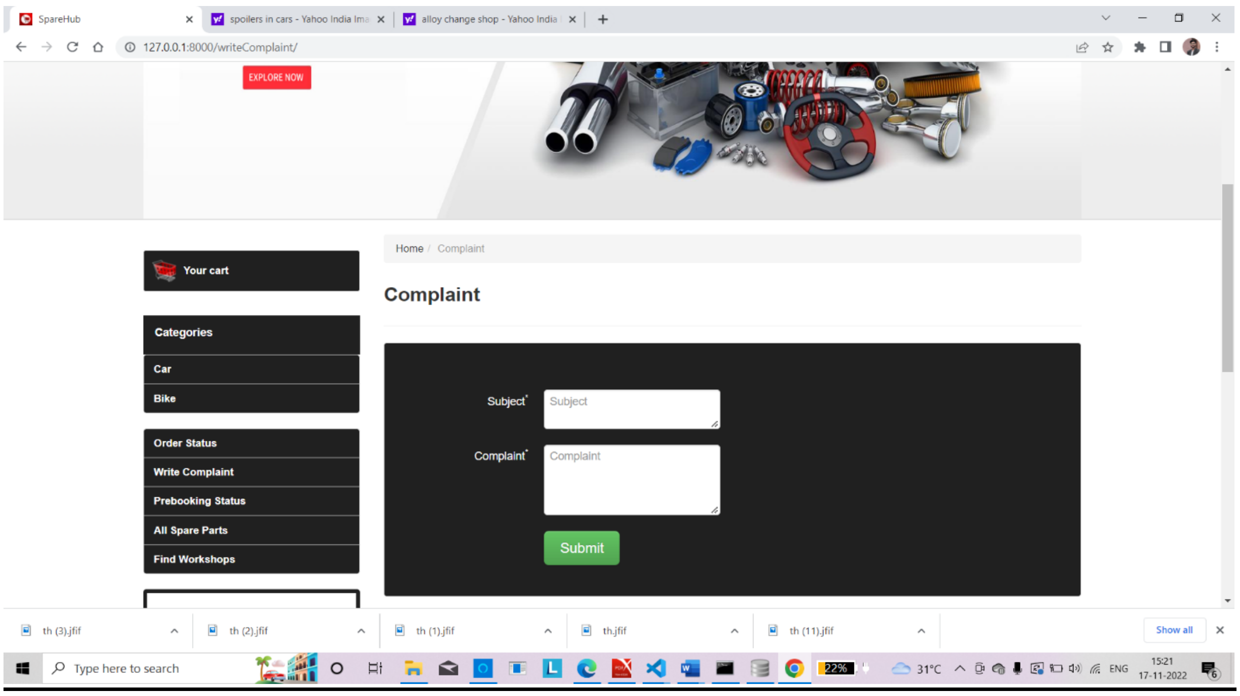
Description : Tuning

Wheel Service

Status : available

Description : Alloy chanes,Painting

Write complaint



Complaint

Subject* Subject

Complaint* Complaint