

# **RENTSHO MASTER- DRESS RENTAL SYSTEM**

*Project Report Submitted By*

**JEENA MATHEW**

**Reg. No.:AJC20MCA-2042**

*In Partial fulfillment for the Award of the Degree of*

**MASTER OF COMPUTER APPLICATIONS  
(REGULAR MCA)  
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING  
KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2021-2022**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**AMAL JYOTHI COLLEGE OF ENGINEERING**  
**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**RENTSHO MASTER – DRESS RENTAL SYSTEM**” is the bonafide work of **JEENA MATHEW (Reg.No:AJC20MCA-2042)** in partial fulfillment of the requirements for the award of the Graduation of Regular Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-2022.

**Dr. Bijimol T K**  
**Internal Guide**

**Ms. Nimmy Francis**  
**Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**  
**Head of the Department**

**External Examiner**

## **DECLARATION**

I hereby declare that the project report “**RENTSHO MASTER – DRESS RENTAL SYSTEM**” is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

**Date:**

**KANJIRAPPALLY**

**JEENA MATHEW**

**Reg.No:AJC20MCA-2042**

## ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Nimmy Francis** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide, **Dr. Bijimol T K** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions and encouragement to make this venture a success.

JEENA MATHEW

## **ABSTRACT**

Rentsho Master is an online clothing rental portal and web-based shopping application. A garment is rented for a short period of time, which can last from a few days to a week. This is a fabric rent shopping site and we can also rent special occasion dresses such as weddings, parties and formal events. The customer pays a one-time fee and can enjoy the garment without the commitment of owning a formal dress and rental dresses. It allows vendors to set up online shops, customers can rent special holiday dresses, as well as manage lists of shop categories and approve or reject requests for new lenders. The creation of an online shopping platform to manage in-store merchandise and enable customers to make purchases without going to the store in person is also on the agenda. The daily selling operations are made possible through an online shopping cart. Modules for managing the products and customers are included. Anyone wishing to rent products online will find this website helpful.

# CONTENT

Sl. No	Topic	Page No
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	3
2.1	INTRODUCTION	4
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	5
2.4	PROPOSED SYSTEM	5
2.5	ADVANTAGES OF PROPOSED SYSTEM	5
3	REQUIREMENT ANALYSIS	6
3.1	FEASIBILITY STUDY	7
3.1.1	ECONOMICAL FEASIBILITY	7
3.1.2	TECHNICAL FEASIBILITY	7
3.1.3	BEHAVIORAL FEASIBILITY	8
3.2	SYSTEM SPECIFICATION	8
3.2.1	HARDWARE SPECIFICATION	8
3.2.2	SOFTWARE SPECIFICATION	8
3.3	SOFTWARE DESCRIPTION	9
3.3.1	PHP	9
3.3.2	MYSQL	10
4	SYSTEM DESIGN	11
4.1	INTRODUCTION	12
4.2	UML DIAGRAM	13
4.2.1	USE CASE DIAGRAM	13
4.2.2	SEQUENCE DIAGRAM	15
4.2.3	STATE CHART DIAGRAM	20
4.2.4	ACTIVITY DIAGRAM	21
4.2.5	CLASS DIAGRAM	22
4.2.6	OBJECT DIAGRAM	23
4.2.7	COMPONENT DIAGRAM	24
4.2.8	DEPLOYMENT DIAGRAM	25
4.3	USER INTERFACE DESIGN	26

<b>4..4</b>	<b>DATA BASE DESIGN</b>	<b>31</b>
<b>4.5</b>	<b>TABLE DESIGN</b>	<b>34</b>
<b>5</b>	<b>SYSTEM TESTING</b>	<b>40</b>
<b>5.1</b>	<b>INTRODUCTION</b>	<b>41</b>
<b>5.2</b>	<b>TEST PLAN</b>	<b>42</b>
<b>5.2.1</b>	<b>UNIT TESTING</b>	<b>42</b>
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	<b>43</b>
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	<b>43</b>
<b>5.2.4</b>	<b>USER ACCEPTANCE TASTING</b>	<b>43</b>
<b>5.2.5</b>	<b>SELENIUM TASTING</b>	<b>44</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>48</b>
<b>6.1</b>	<b>INTRODUCTION</b>	<b>49</b>
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	<b>49</b>
<b>6.2.1</b>	<b>USER TRAINING</b>	<b>50</b>
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	<b>50</b>
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	<b>50</b>
<b>6.2.4</b>	<b>HOSTING</b>	<b>51</b>
<b>6.2.5</b>	<b>HOSTING METHOD STEPS</b>	<b>51</b>
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>52</b>
<b>7.1</b>	<b>CONCLUSION</b>	<b>53</b>
<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>54</b>
<b>9</b>	<b>APPENDIX</b>	<b>56</b>
<b>9.1</b>	<b>SAMPLE CODE</b>	<b>57</b>
<b>9.2</b>	<b>SREENSHOTS</b>	<b>67</b>

## List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
SQL	-	Structured Query Language
UML	-	Unified Modeling Language



## **CHAPTER 1**

### **INTRODUCTION**

## **1.1 PROJECT OVERVIEW**

Dress Rental system project main idea is to develop a clothing rental portal and Admin will add dresses according to occasions. And, customers will rent these dresses. Customers to book or rent dresses for some occasions. Here, there are multiple users like admin and customers and admin will manage the overall application by adding, and removing dresses from the system.

Rentsho Master is an online platform that helps the user to rent the dress in online. This website facilitates the customer can view the rent products, add rental product, add to cart, product details.

## **1.2 PROJECT SPECIFICATION**

This project aims at creating an Dress Rental system which can be used by customers to add and view dress and other facility online. Users can check availability of rental products and other facilities.

The system includes 4 users. They are:

### **1. Admin**

Admin must have a login into this system. He has the overall control of the system. Admin will add dresses according to occasions, manage user data, and add delivery boy etc. Admin can View all the registered users and also manage all his data.

### **2. Customers**

There are two types of customers

The first user is the customer, they can register or update their profile. And also they can check and view available of rent products and other facilities. And generate rent bill according to the return date.

Second user is the Lender, it is similar to the user and the additional functionality is they can add their own dresses according to occasions.

### **3. Delivery Boy**

The delivery boy is to delivery rent products and to collect return products with the help of user details.

## **CHAPTER 2**

### **SYSTEM STUDY**

---

## 2.1 INTRODUCTION

System analysis is the process of acquiring and analysing data, diagnosing issues, and using the data to suggest system changes. The system users and system developers must communicate extensively during this problem-solving process. Any system development process should start with a system analysis or research. The system is meticulously examined and assessed. The system analyst takes on the role of the interrogator and probes deeply into how the current system functions. The input to the system is identified, and the system is seen as a whole. The various procedures can be linked to the outputs from the organisations. Being aware of the issue, finding the pertinent and important variables, analysing, and synthesising are all parts of system analysis.

The process must be thoroughly studied using a variety of methodologies, including questionnaires and interviews. To reach a conclusion, the information gathered by these sources must be carefully examined. Understanding how the system works is the conclusion. The current system is the name of this system. Now that the issue with the current system has been thoroughly examined, trouble spots have been located. The designer now acts as a problem-solver and works to resolve the issues the business is having. Proposals are made in place of the solutions. The proposal is then analytically compared to the current system, and the best one is chosen. The user is given the opportunity to approve or reject the suggestion. The proposal is examined for appropriate adjustments based on user requests.

The process of acquiring and analysing data in order to use it for future system studies is known as preliminary study. Initial research is a problem-solving activity that necessitates close coordination between system users and developers. It conducts a number of feasibility studies. These investigations provide an approximate estimate of the system activities, which can be used to determine the tactics to be used for an efficient system research and analysis.

## **2.2 EXISTING SYSTEM**

The existing system is not fully automated system. The customer can only purchase products but they can't rent their own dresses. The proposed system rectify the drawbacks of the present system.

## **2.3 DRAWBACKS OF EXISTING SYSTEM**

- Customers who can't add their products.
- There is no commission of rent and security fee.

## **2.4 PROPOSED SYSTEM**

To overcome the problems of manual system. Customers can directly communicate with the shop. The central objective of dress rental system is to provide online facility for accessing all the facility of rental shop.

## **2.5 ADVANTAGES OF PROPOSED SYSTEM**

- Saves time and money of customers in quickly reserving all the facility of rent shop.
- The ability to get rent products, from anywhere with internet access.
- Provide the information about rent shop facilities.
- User friendly.

## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

### 3.1 FEASIBILITY STUDY

To ascertain if the project will, after completion, achieve the organization's goals in relation to the work, effort, and time put in it, a feasibility study is carried out. The developer can forecast the project's usefulness and potential future thanks to a feasibility study. The effectiveness of a system proposal is assessed by looking at its effects on the organisation, user needs, and resource utilisation. As a result, a feasibility evaluation is frequently performed before a new application is approved for development. The feasibility of the project is described in the paper, along with a list of the several aspects—such as technical, economic, and operational viability—that were carefully considered throughout the project's feasibility study. The following traits apply to it:

#### 3.1.1 Economical Feasibility

Analyses of costs and benefits are necessary to support the developing system. criteria to ensure that the project that would produce the best outcomes the quickest is given priority. One of the variables is the cost associated with establishing a new system.

Following are some important financial questions that were brought up during the original investigation:

- The costs look into the entire system.
- The price of the equipment and software.
- The advantages in terms of lower expenses or fewer expensive mistakes.

The suggested solution was established as part of a project, thus there are no manual costs associated with it. The system may also be put into place at a reasonable cost given the availability of the required resources. The project's cost for the employment exchange system was split based on the system used and its development costs. All estimates show that the project was created at a reasonable cost. As all of its development was done using open source software.

#### 3.1.2 Technical Feasibility

The system must first undergo a technical evaluation. The assessment of its viability must be built upon an overview design of the system's requirements in terms of input, output, programmes, and procedures. After identifying an outline system, the inquiry must next recommend the type of equipment, necessary steps for building the system, and methods of operating the system once it has been designed. The following technical issues came up during the investigation:

- 
- Is the technology now available sufficient for the suggested work?
  - Can the system expand if developed?

The project should be designed in such a way that the necessary functionality and performance are met within the constraints. The fact that a newer version of the same software still functions with an older version means that the system can still be used even though the technology may become outmoded with time. Consequently, there aren't many restrictions on this project. The system was developed with PHP for the front end and a MySQL server for the back end; the project can be finished technically. The secondhand system boasts a powerful Pentium processor, 6GB of RAM, and a 500GB hard disc.

### 3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Does the users' help meet their needs?
- Will the suggested system damage anyone?

Because it would accomplish the objectives after being developed and put into action, the project would be advantageous. The project is considered to be behaviorally feasible after carefully assessing all behavioural considerations.

## 3.2 SYSTEM SPECIFICATION

### 3.2.1 Hardware Specification

Processor - Intel core i5

RAM - 6 GB

Hard disk - 500 GB

### 3.2.2 Software Specification

Front End - HTML, CSS

Back end - MYSQL

Client on PC - Windows 10 and above.

Technologies used - JS, J Query, PHP



---

## 3.3 SOFTWARE DESCRIPTION

### 3.3.1 PHP

PHP is a server-side scripting language that was developed for web development but is now used for general-purpose programming, powering more than 244 million websites and 2.1 million web servers. The reference version of PHP, which Rasmus Ledorf created in 1995, is now created by the PHP group. What was originally known as personal Home page is now known as PHP: A web server understands the recursive acronym code and creates the final web page by using Hypertext Preprocessor commands that can be added directly into an HTML source document rather than accessing an external file to process data. It has also evolved to currently lack a command-line interface functionality and be incompatible with the GNU General Public License.

### 3.3.2 MySQL

The most well-known Open Source SQL database management system, MySQL, was developed, distributed, and supported by Oracle Corporation. On the MySQL website, you may find the most latest information regarding the MySQL programme.

- **MySQL is a database management system.**

A database is a structured collection of data. It may be anything from a simple shopping list to picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You setup rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well- designed database, your application never sees in consistent, duplicate, orphan, out-of-date, or missing data. The SQL part

of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL92” refers to the standard released in 1992,

“SQL: 1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- **MySQL software is Open Source.**

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything.

If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information.

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

You ought to give it a shot if that is what you're after. In addition to your other apps, web servers, and other software, MySQL Server can function smoothly on a desktop or laptop while requiring little to no maintenance. You can modify the settings to utilise all the RAM, CPU power, and I/O capacity if you dedicate an entire machine to MySQL.

- **MySQL Server works in client/server or embedded systems.**

The MySQL Database Software is a client/server system that includes a multi-threaded SQL server that supports several client programmes and libraries, administration tools, and a broad variety of application programming interfaces (APIs). Additionally, we provide MySQL Server as an integrated multi-threaded library that you can connect into your programme to create a separate project that is smaller, faster, and easier to administer

## **CHAPTER 4**

### **SYSTEM DESIGN**

---

## 4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be described as the process of using different ideas and approaches to specify a device, processor, or system in too little detail to allow for its physical implementation. Regardless of the development paradigm that is employed, software design forms the technical core of the software engineering process. The architectural detail needed to construct a system or product is developed through the system design. This programme has also through the best possible design phase, fine tuning all efficiency, performance, and accuracy levels, as in the case of any systematic technique. A user-oriented document is converted into a document for programmers or database staff throughout the design phase. The two stages of system design development are logical and physical.

## 4.2 UML DIAGRAM

A common language known as UML is used to specify, visualise, build, and document the software system artefacts. The Object Management Group (OMG) was responsible for developing UML, and a draught of the UML 1.0 definition was presented to the OMG in January 1997. **Unified Modeling Language**, also known as UML, is different from other well-known programming languages like C++, Java, COBOL, and others. UML is a pictorial language used to develop software blueprints and is a general-purpose visual modelling language for visualising, specifying, building, and recording software systems. Although representing software systems is the most popular use of UML, it is not the only one. Additionally, it is employed to model non-software-based systems. For instance, the manufacturing facility's process flow, etc. Despite the fact that UML is not a programming language, tools that use UML diagrams can generate code in a wide range of languages. The analysis and design of objects-oriented systems are directly related to UML. It has been so well standardised that OMG now recognises UML as a standard. All the components and relationships make to a complete UML diagram that represents a system. The visual impact of the UML diagram is the most important factor in the entire process. The remaining parts, including the following nine schematics, are used to complete it.

- 
- Class diagram
  - Object diagram
  - Use case diagram
  - Sequence diagram
  - Collaboration diagram
  - Activity diagram
  - State chart diagram
  - Deployment diagram
  - Component diagram

### 4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are created to depict a system's functional requirements. To create an effective use case diagram after identifying the aforementioned things, we must adhere to the following rules.

- A use case's naming is very significant. The name should be selected in a way that makes it clear what functions are being performed.
- Give the actors names that fit them.
- Clearly depict links and dependencies in the diagram.
- Keep in mind that the main goal of the diagram is to establish the needs, therefore avoid attempting to include all possible relationships.

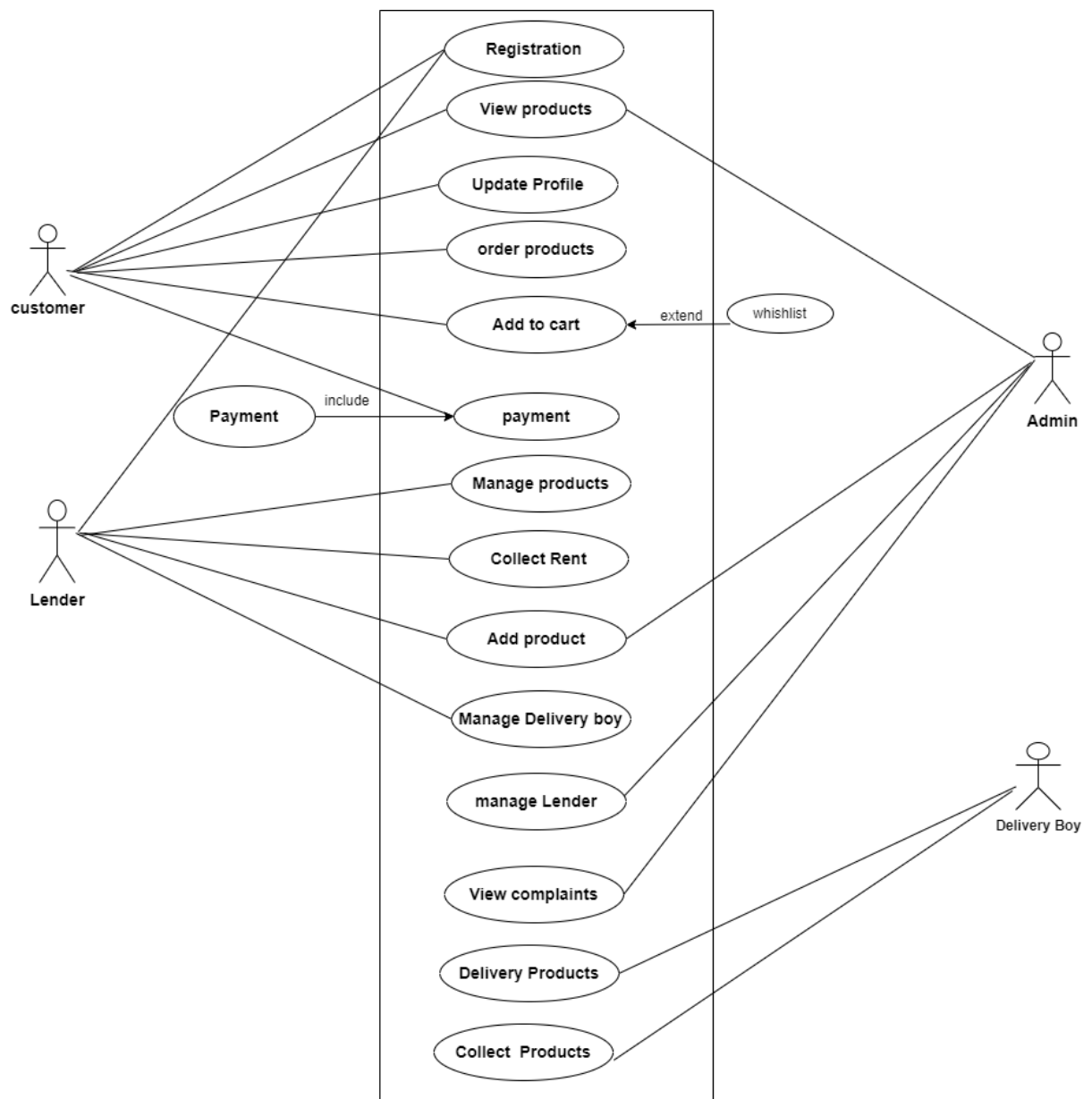


Figure 4.1 Use case diagram

## 4.2.2 SEQUENCE DIAGRAM

A sequence diagram fundamentally depicts the sequential order in which events occur or how they interact with one another. Event diagrams and event scenarios are other names for sequence diagrams. Sequence diagrams display the actions performed by a system's parts in a time-based manner. Business professionals and software engineers frequently use these diagrams to describe and document the requirements for new and current systems.

### Sequence Diagram Notations –

- i. **Actors** – – In a UML diagram, an actor represents a particular kind of role in which it communicates with the system's objects. An actor is always beyond the purview of the system that we want to use the UML diagram to represent. We employ actors to portray a variety of roles, including those of human users and other outside subjects. In a UML diagram, an actor is represented using a stick person notation. In a sequence diagram, there might be several actors.
- ii. **Lifelines** – A named piece that shows a specific participant in a sequence diagram is called a lifeline. In a sequence diagram, each episode is essentially represented by a lifeline. The lifeline components in a sequence diagram are at the top.
- iii. **Messages** – Using messages, communication between objects is demonstrated. The messages are displayed on the lifeline in chronological sequence. Arrows are how messages are represented. A sequence diagram's main components are lifelines and messages.

Messages can be broadly classified into the following categories:

- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message
- Found Message
- Lost Message

**iv. Guards** –In the UML, we utilize guards to model circumstances. When we need to limit the flow of messages under the guise of a condition being met, we use them. Software engineers rely on guards to inform them of the limitations imposed by a system or specific process.

#### Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.

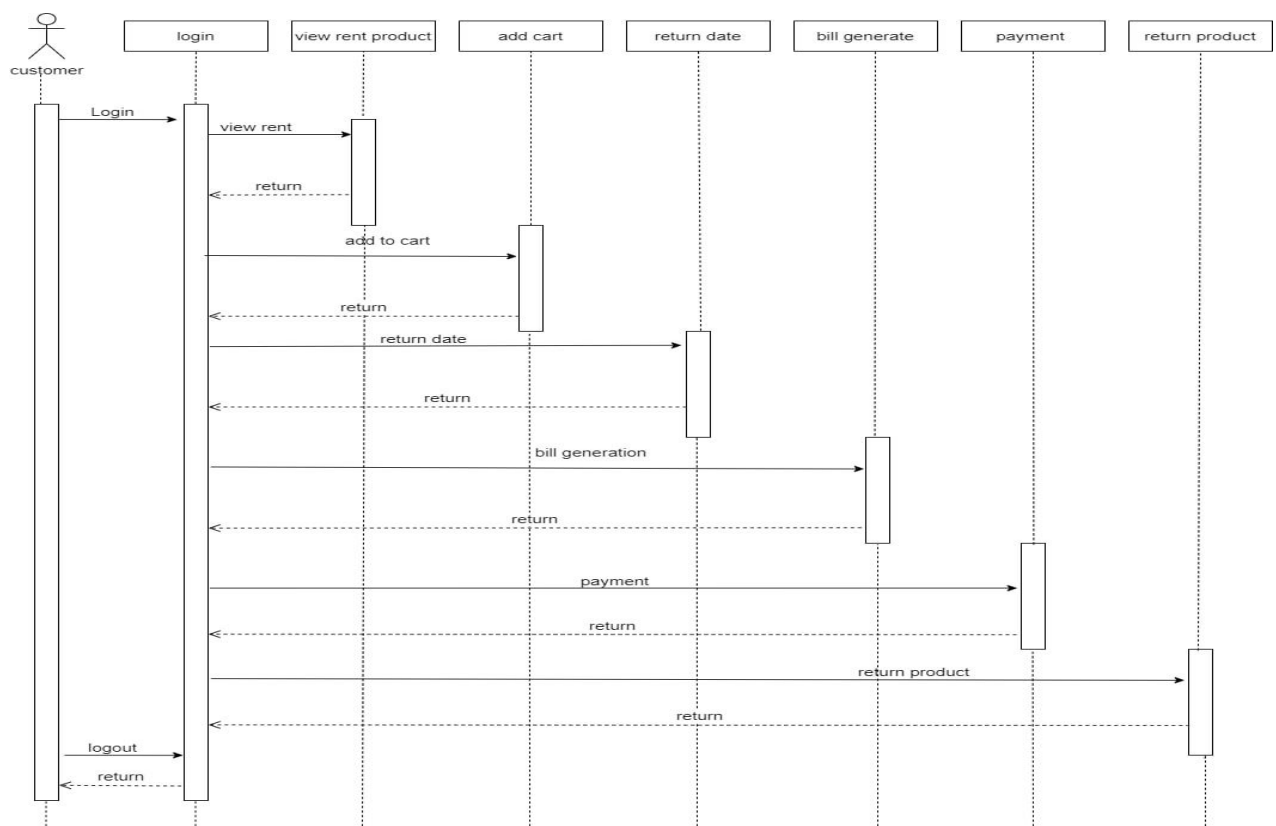


Figure 4.2.2.1 Sequence diagram-Customer



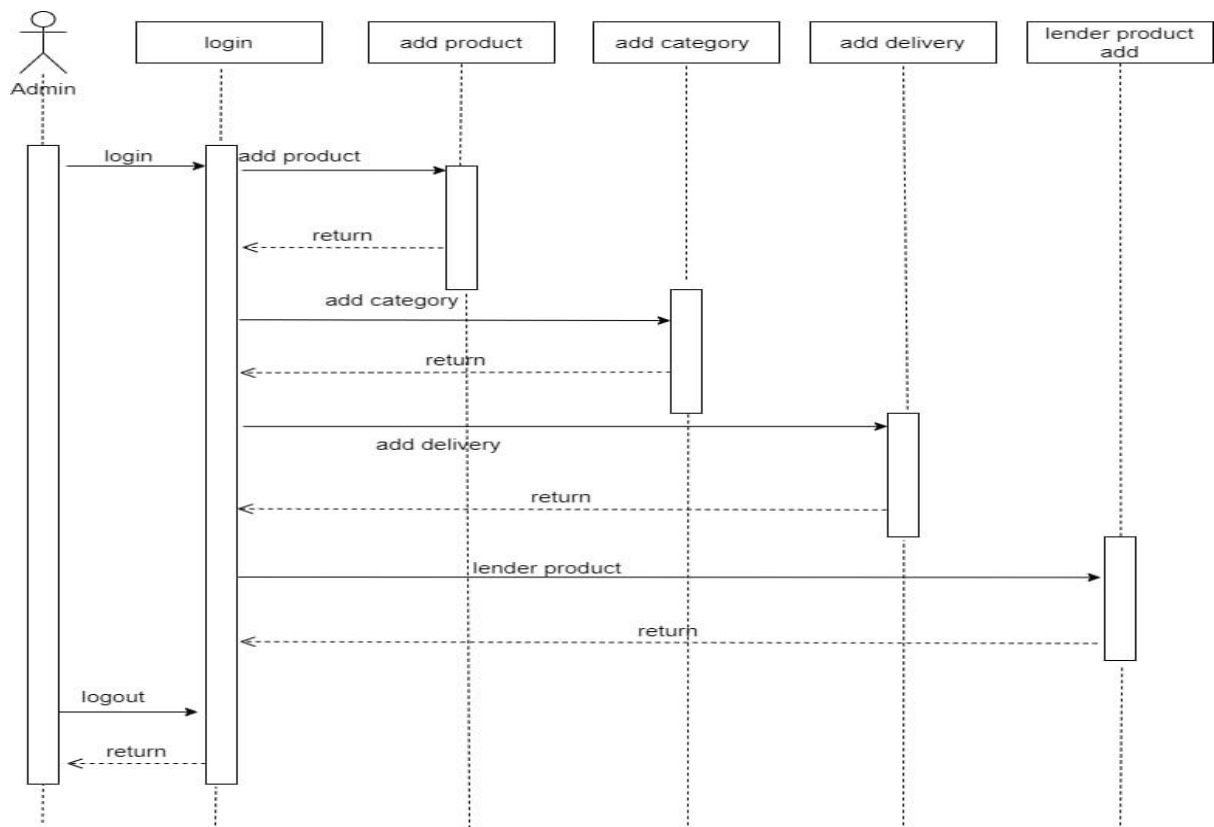


Figure 4.2.2.2 Sequence diagram- Admin

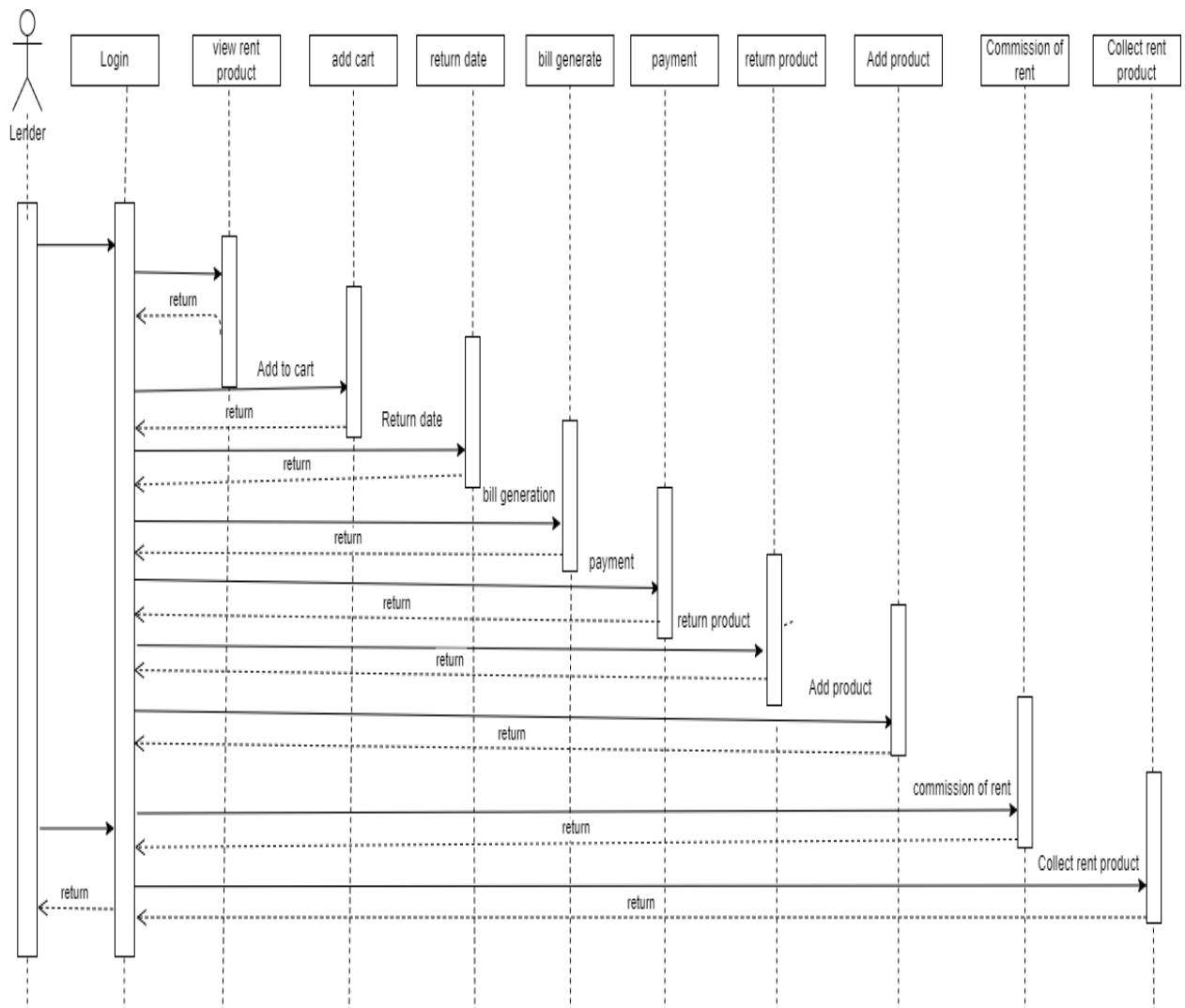


Figure 4.2.2.3 Sequence diagram- Lender

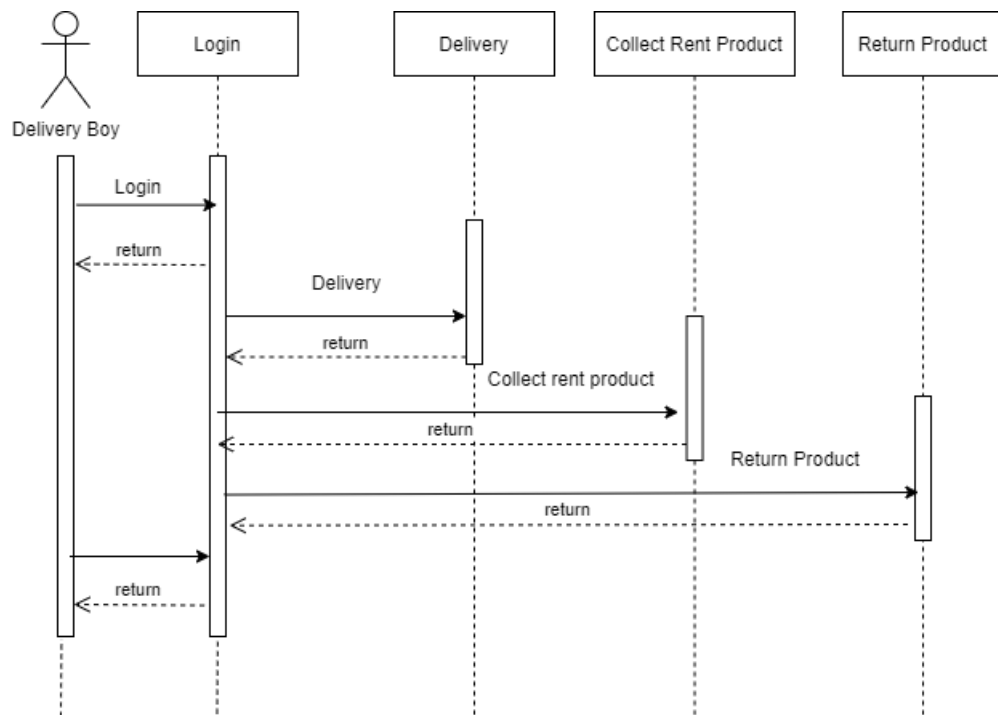


Figure 4.2.2.4 Sequence diagram- Delivery Boy

### 4.2.3 State Chart Diagram

State diagrams are a common tool for displaying a software system's behaviour. A state machine diagram in a UML model can represent the behaviour of a class, a subsystem, a package, or even an entire system. Other names for it include state charts and state transition diagrams. We can effectively describe the communications or interactions that occur between external entities and a system using state chart diagrams. These diagrams are used to model the event-based system. An event can be used to control an object's state. State chart diagrams are used to show the various states that an entity can be in within the application system.

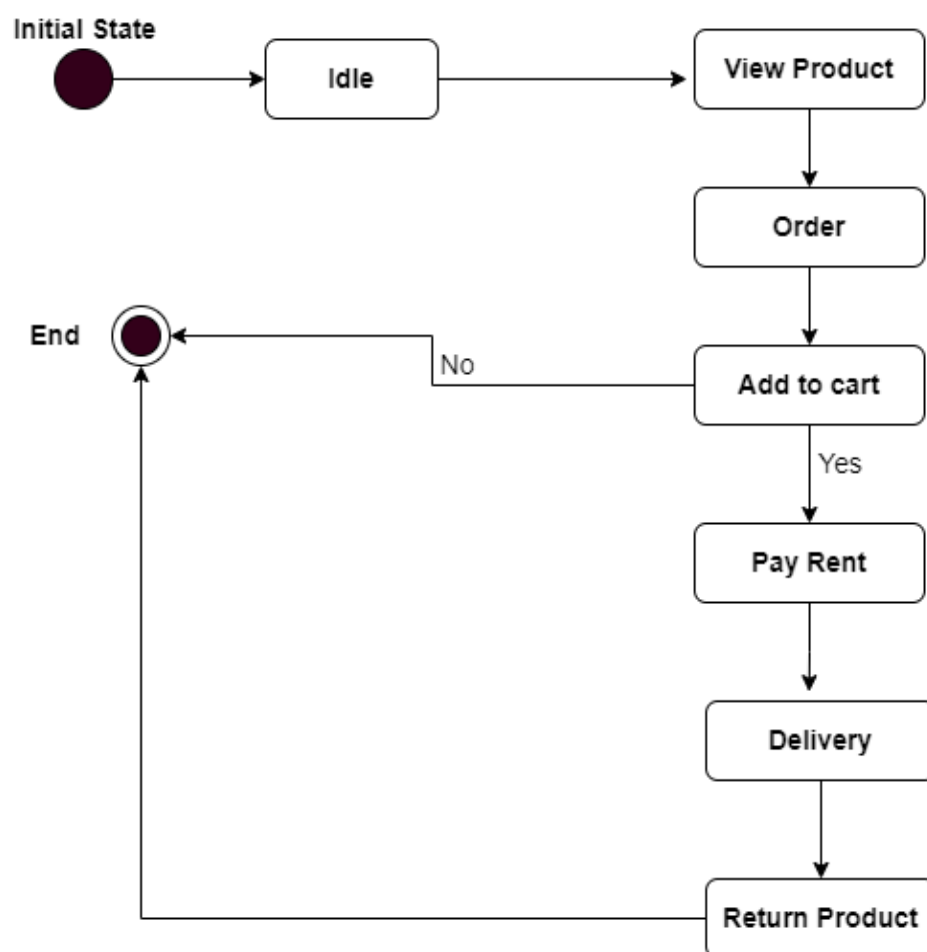


Figure 4.2.3.1 State diagram

### 4.2.4 Activity Diagram

Activity diagrams show how multiple levels of abstraction of activities are coordinated to produce a service. Typically, an event must be accomplished by some operations, especially when the operation is meant to accomplish several different things that call for coordination. Another common requirement is how the events in a single use case relate to one another, especially in use cases where activities may overlap and require coordination. It is also appropriate for modelling the coordination of a set of use cases to reflect business workflows.

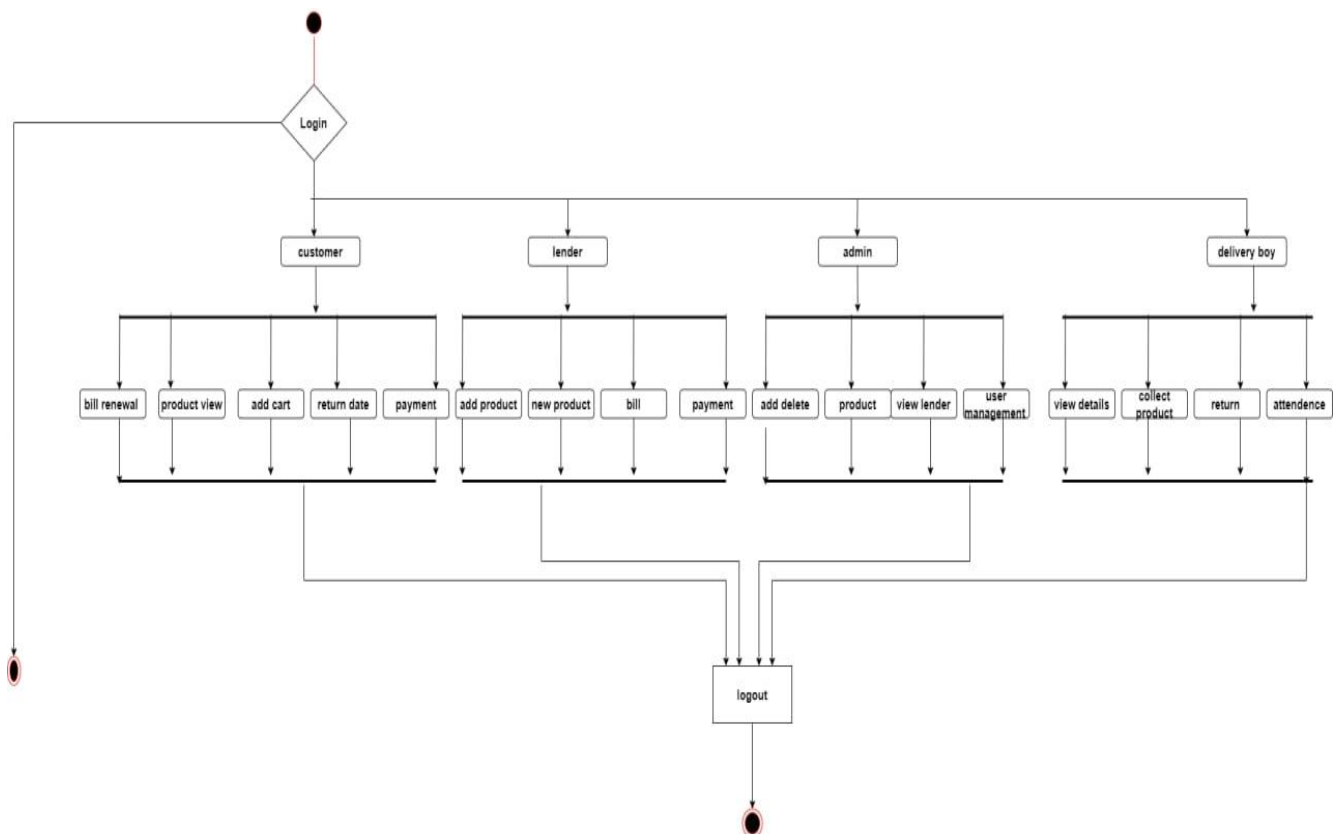


Figure 4.2.4 Activity diagram

### 4.2.5 Class Diagram

Static diagrams include class diagrams. It represents the application's static view. Class diagrams are helpful for developing executable code for software programmes as well as for visualising, explaining, and documenting various system components. The properties and operations of a class are described in a class diagram together with the constraints placed on the system. Since class diagrams are the only UML diagrams that can be immediately translated using object-oriented languages, they are frequently employed in the modelling of object-oriented systems. An assortment of classes, interfaces, affiliations, collaborations, and limitations are displayed in a class diagram. Additionally known as a "structural diagram".

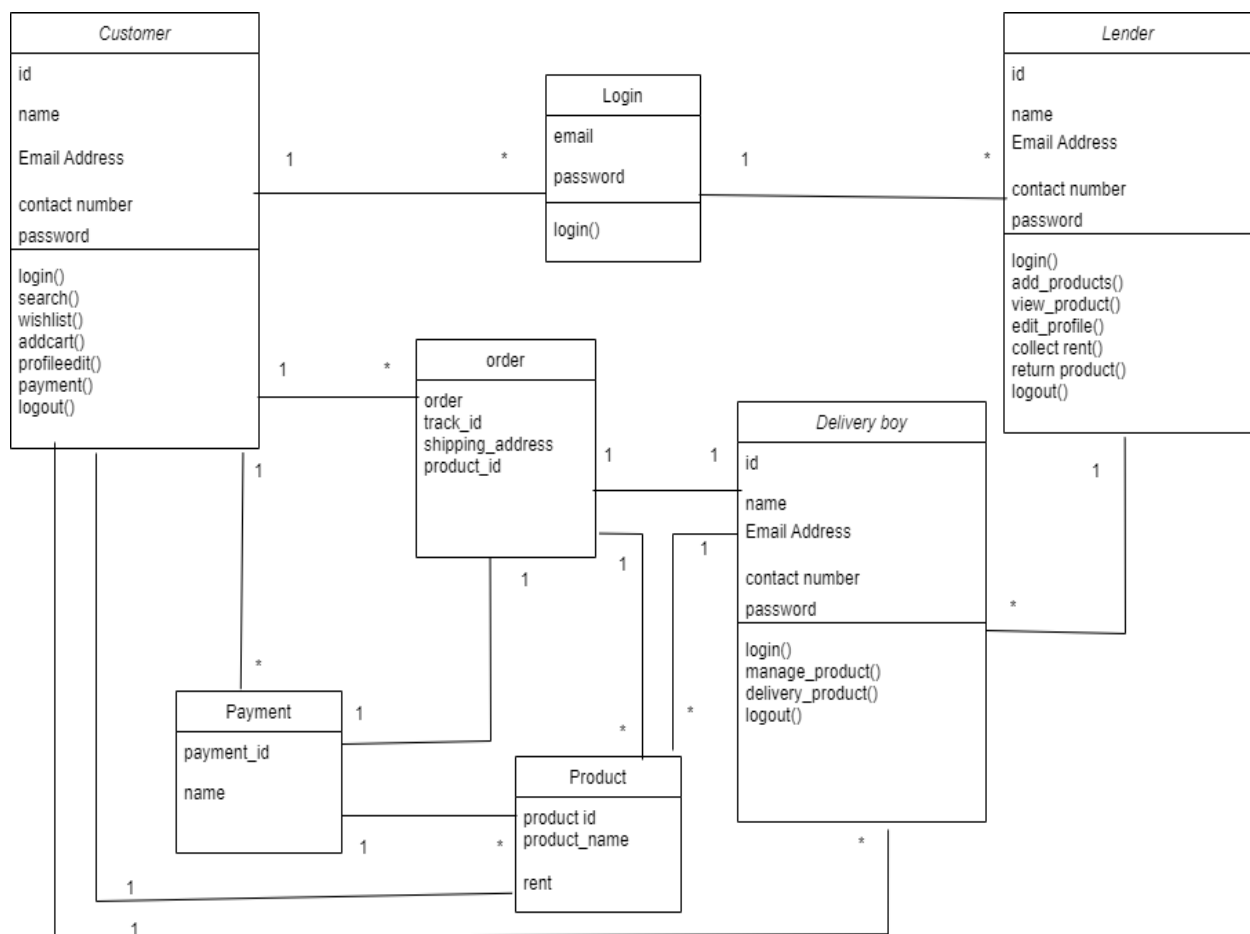


Figure 4.2.5 Class diagram

### 4.2.6 Object Diagram

Class diagrams are a requirement for object diagrams because they are the source of class diagrams. An object diagram illustrates a specific instance of a class diagram. Diagrams of classes and objects share the same underlying concepts. Object diagrams can also be used to describe a system's static view, which is a brief snapshot of the system. Object diagrams are used to illustrate a collection of items and their connections as an example.

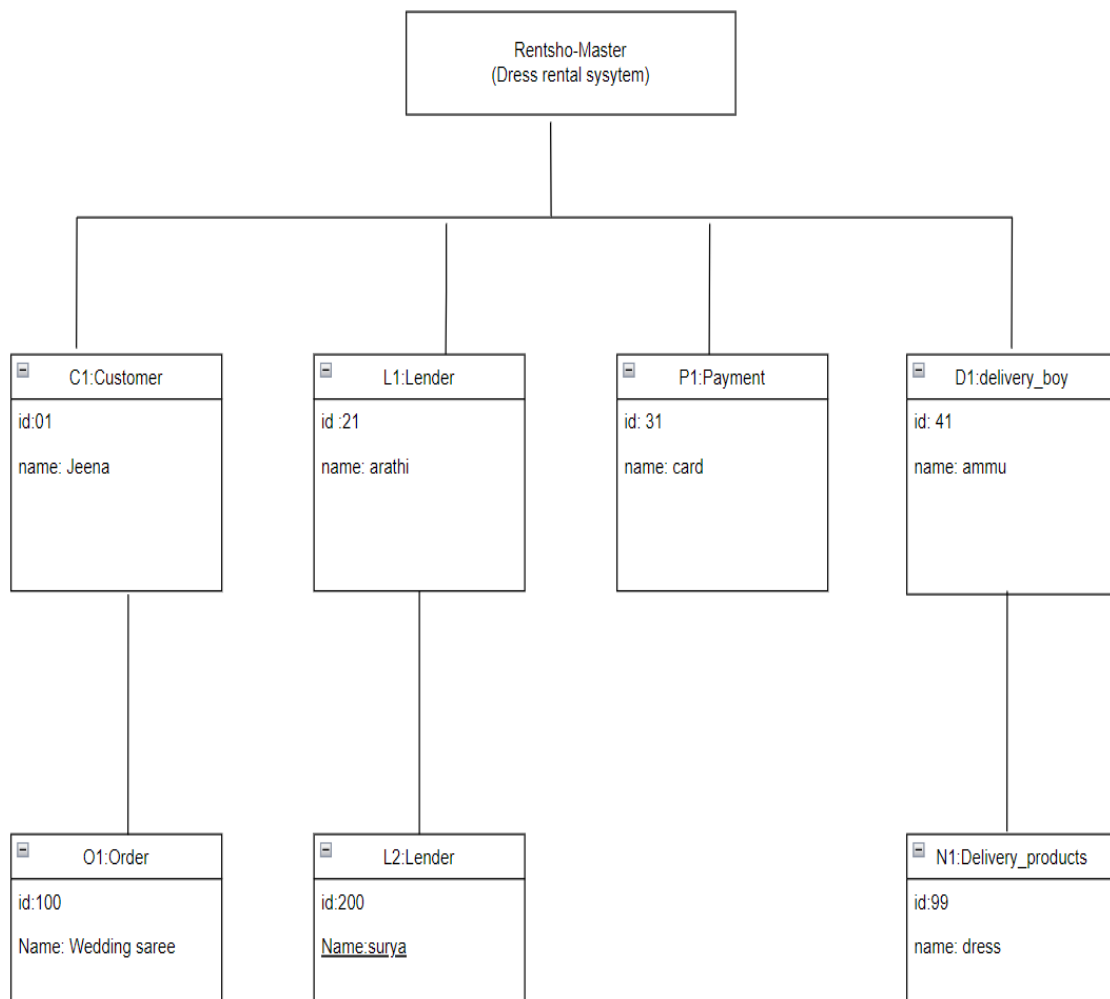


Figure 4.2.6 Object diagram

### 4.2.7 Component Diagram

Component diagrams have different behaviours and personalities. The physical parts of the system are represented using component diagrams. Executables, libraries, files, documents, and other items that are physically present in a node are just a few examples. Component diagrams are used to show how the components of a system are connected and arranged. These diagrams can also be used to construct systems that can be run.

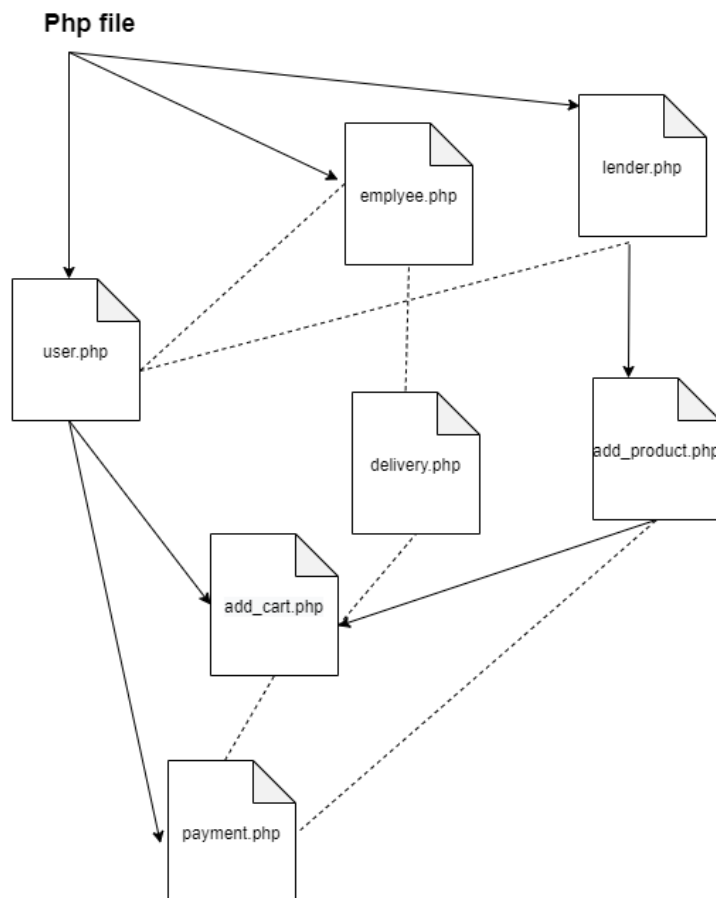


Figure 4.2.7 Component diagram



## 4.2.8 DEPLOYMENT DIAGRAM

An execution architecture of a system, containing nodes like hardware or software execution environments, and the middleware linking them, is shown in a deployment diagram, a form of UML diagram. Typically, deployment diagrams are used to represent the actual hardware and software of a system.

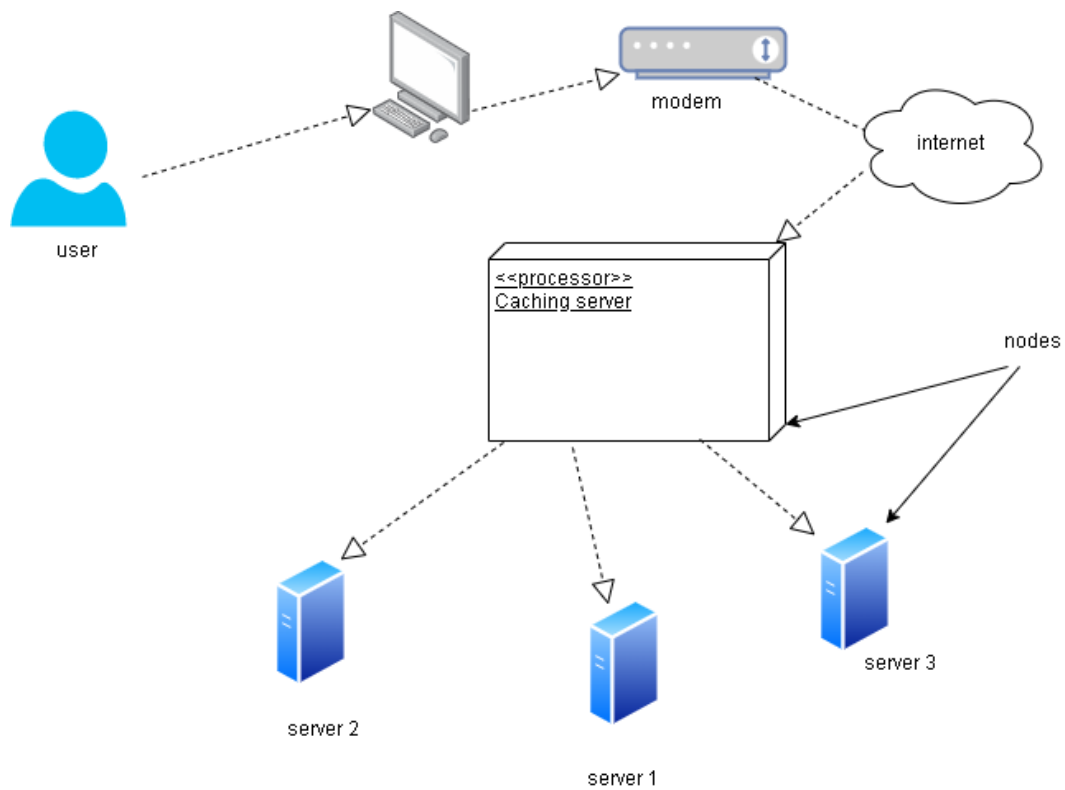


Figure 4.8 Deployment diagram

## 4.3 USER INTERFACE DESIGN

### 4.3.1-INPUT DESIGN

Form Name : User Registration

The registration form is titled "Registration" in a bold, black font. It features four input fields: "Uname", "Phone No", "Password", and "Confirm Password". Below these fields is a "Sign up" button. To the left of the form is a decorative graphic of a woman in a colorful, flowing dress with the word "welcome" in a cursive font and "Rentsho-Master" in a bold, sans-serif font at the bottom.

Figure 4.3.1.1 User registration

Form Name : User Login

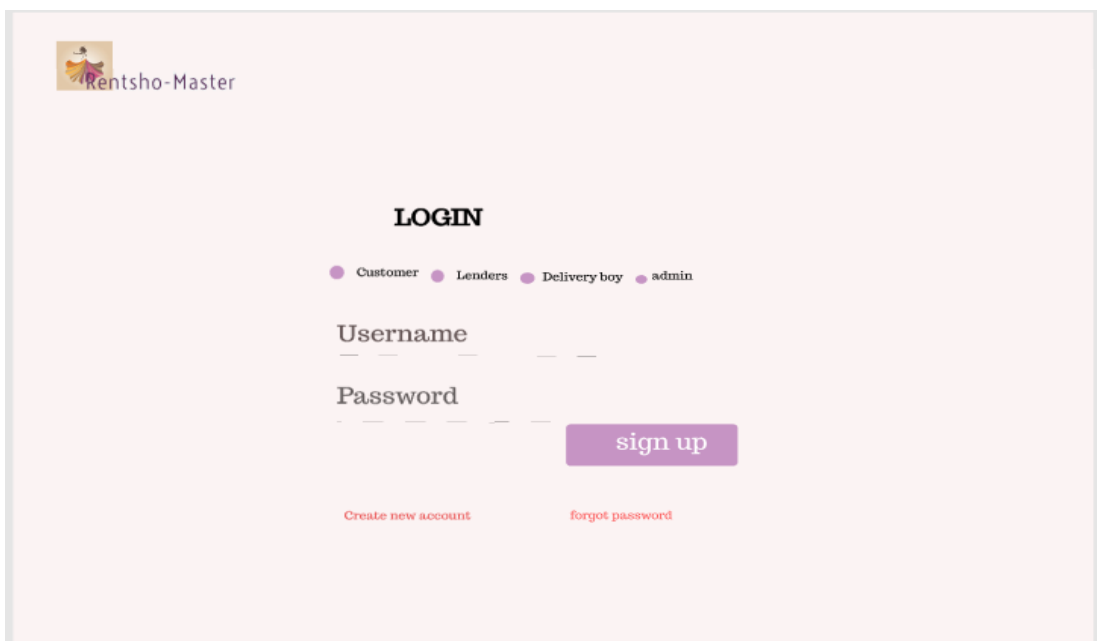
The login form is titled "LOGIN" in a bold, black font. It features two input fields: "Username" and "Password". Below these fields is a "sign up" button. Above the input fields are four radio buttons labeled "Customer", "Lenders", "Delivery boy", and "admin". At the bottom of the form are two links: "Create new account" and "forgot password". The form is set against a light pink background with a small "Rentsho-Master" logo in the top left corner.

Figure 4.3.1.2 User login

Form Name : Home Page

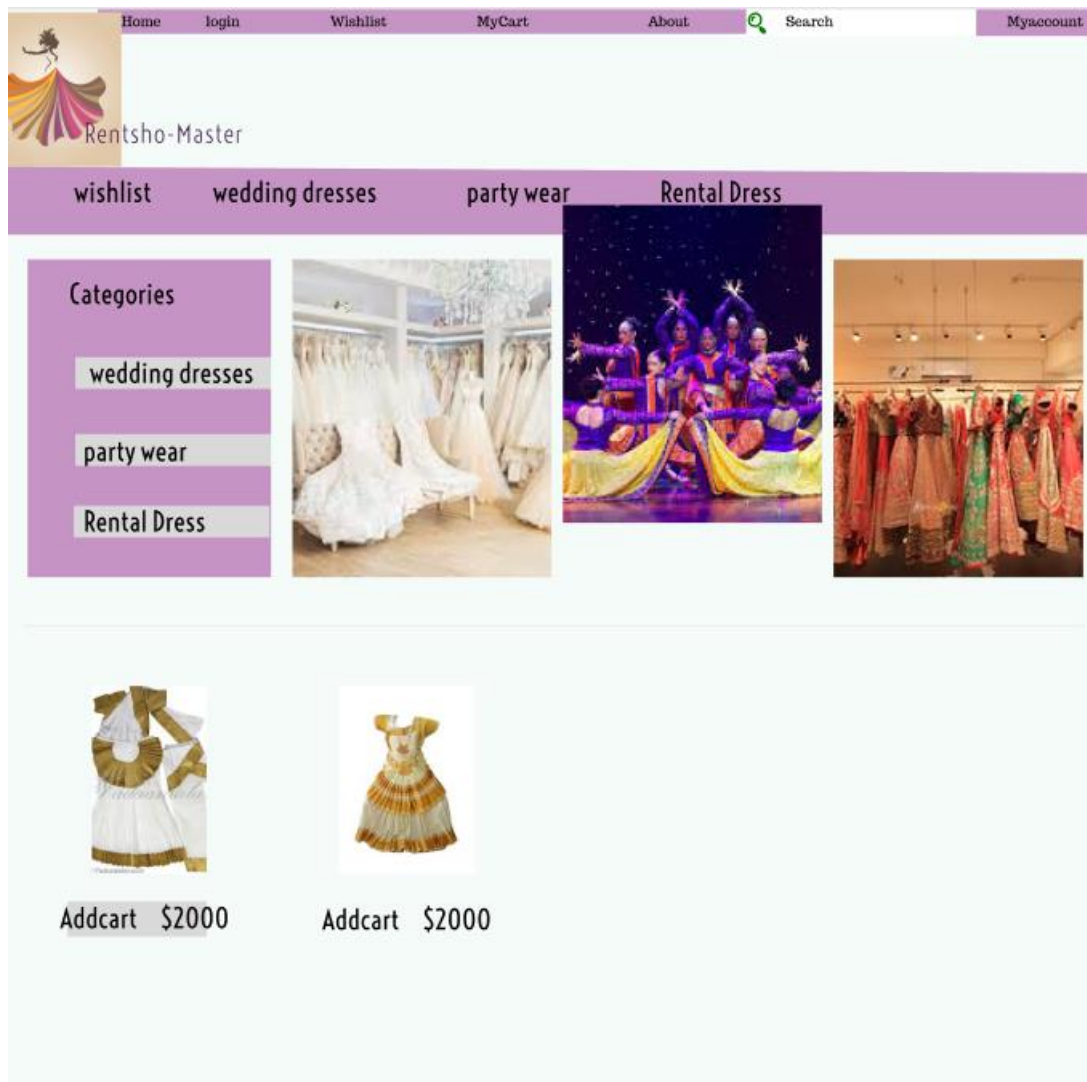


Figure 4.3.1.3 Home page

Form Name : Admin Dashboard

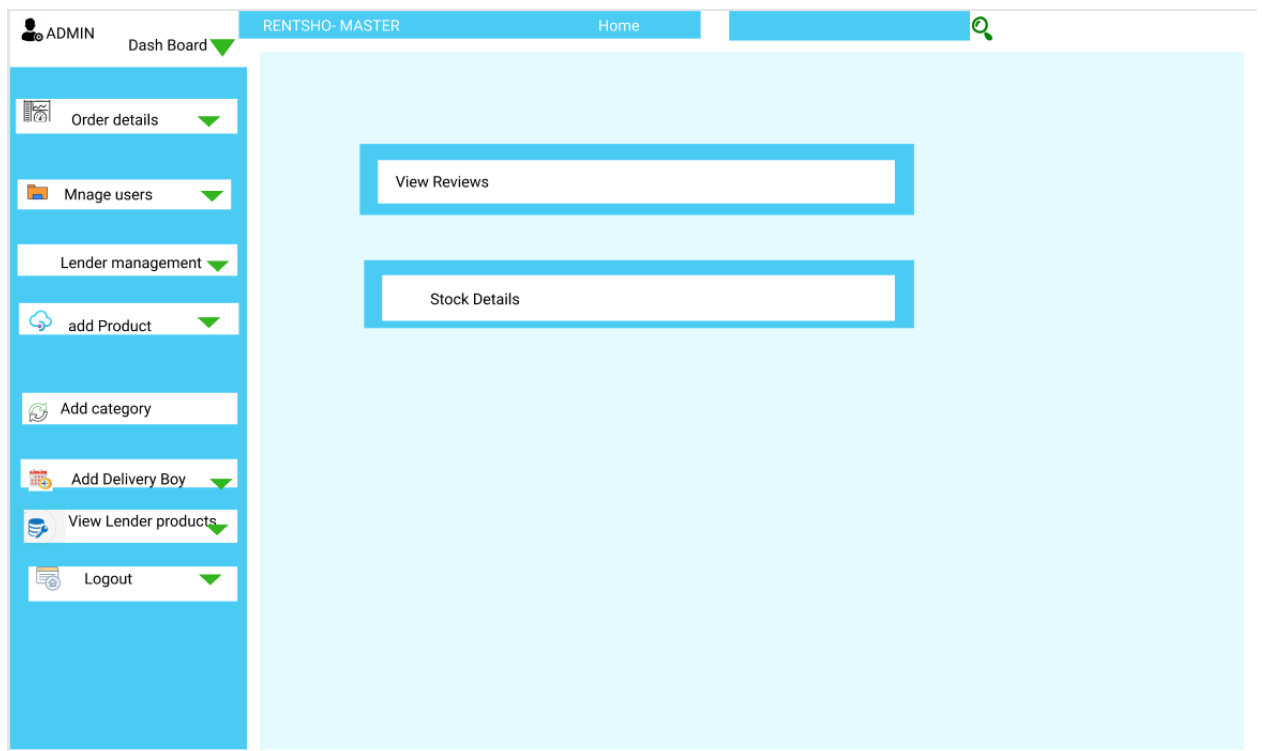
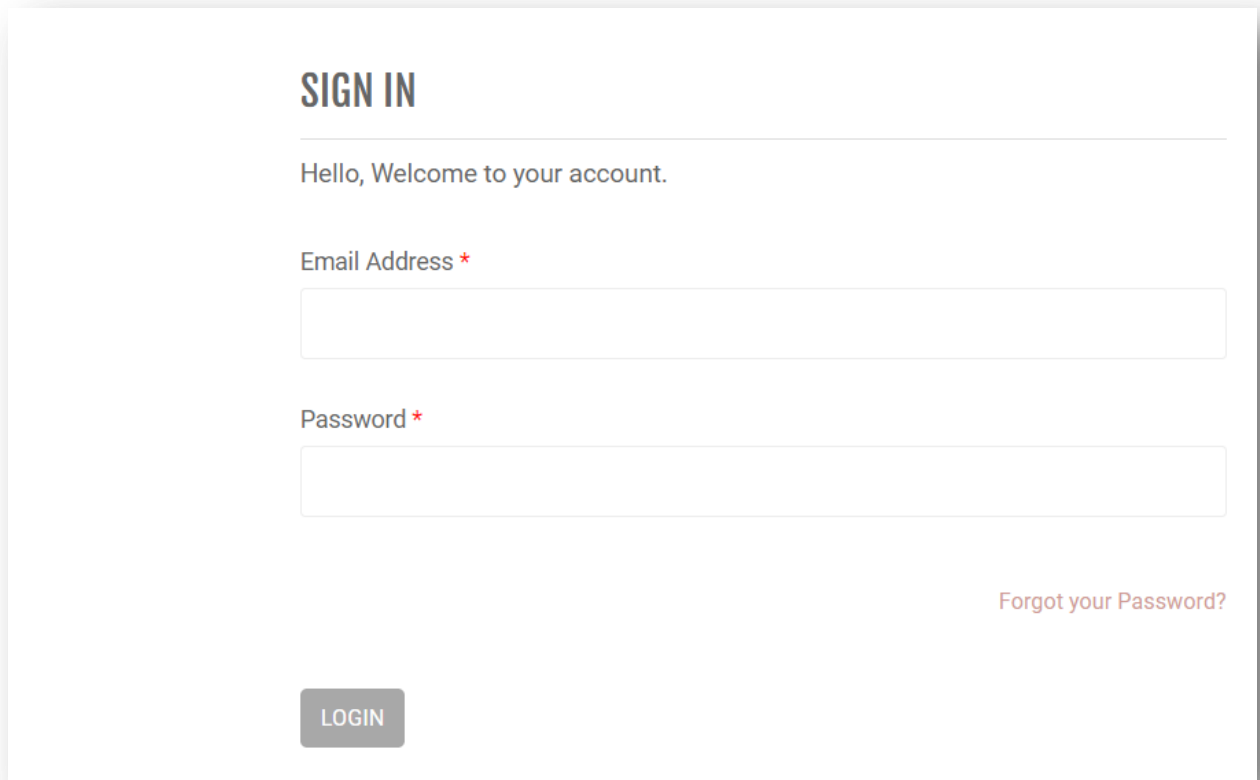


Figure 4.3.1.4 Admin dashboard

### 4.3.2 OUTPUT DESIGN

#### User Login



The image shows a user login form titled "SIGN IN". Below the title is a horizontal line, followed by the text "Hello, Welcome to your account." The form contains two input fields: "Email Address" and "Password", both marked with a red asterisk. Below the "Email Address" field is a text input box. Below the "Password" field is a text input box. To the right of the "Password" field is a link that says "Forgot your Password?". At the bottom left of the form is a button labeled "LOGIN".

**SIGN IN**

---

Hello, Welcome to your account.

Email Address \*

Password \*

[Forgot your Password?](#)

**LOGIN**

Figure 4.3.2.1 User login

## User Registration

### CREATE A NEW ACCOUNT

Create your own Rent Shopping account.

Full Name \*

Email Address \*

Contact No. \*

Password. \*

Confirm Password. \*

Type:

Figure 4.3.2.2 User registration

## 4.4 DATABASE DESIGN

A database is a structured system with the capacity to store information and allows users to retrieve stored information quickly and effectively. Any database's primary goal is its data, which demands protection. There are two stages to the database design process. The user needs are obtained in the first step, and a database is created to as clearly as possible meet these criteria. This process, known as information level design, is carried out independently of all DBMSs.

The design for the specific DBMS that will be used to construct the system in issue is converted from an information level design to a design in the second stage. Physical Level Design is the stage where the characteristics of the particular DBMS that will be used are discussed. Parallel to the system design is a database design. The database's data arrangement aims to accomplish the two main goals listed below.

- Data Integrity
- Data independence

### 4.4.1 Relational Database Management System (RDBMS)

In a relational model, the database is shown as a set of relations. Each relation resembles a file or table of records with values. A row is referred to as a tuple, a column heading is referred to as an attribute, and the table is referred to as a relation in formal relational model language. A relational data base is made up of a number of tables, each with its own name. In a story, each row represents a group of associated values.

#### Relations, Domains & Attributes

A relation is a table. Tuples are the units of a table's rows. An ordered group of  $n$  elements is a tuple. Attributes are referred to as columns. Every table in the database has relationships already established between them. This guarantees the integrity of both referential and entity relationships. A group of atomic values make up a domain  $D$ . Choosing a data type from which the domain's data values are derived is a typical way to define a domain. To make it easier to understand the values of the domain, it is also helpful to give it a name. Each value in a relation is atomic and cannot be broken down.

---

**Relationships**

- Key is used to create table relationships. Primary Key and Foreign Key are the two principal keys that are most crucial. With the use of these keys, relationships for entity integrity and referential integrity can be created.
- Entity Integrity forbids the use of null values for any Primary Key.
- No Primary Key may contain null values, according to Referential Integrity.
- Referential Integrity: A Primary Key value in the same domain must correspond to each unique Foreign Key value. Super Key and Candidate Keys are additional keys.

**4.4.2 Normalization**

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

It means placing things in their natural form, as the name suggests. By using normalisation, the application developer aims to establish a coherent arrangement of the data into appropriate tables and columns, where names may be quickly related to the data by the user. By removing recurring groups from the data, normalisation prevents data redundancy, which puts a heavy strain on the computer's resources. The data should be normalised.

- Give the tables and columns proper names.
- Give the data the right name.



**First Normal Form**

According to the First Normal Form, each attribute's domain must only include atomic values, and each attribute's value in a tuple must be a single value from that domain. In other words, 1NF forbids using relationships as attribute values within tuples or relations within relations. Single atomic or indivisible values are the only attribute values that are permitted under 1NF. The data must first be entered into First Normal Form. This can be accomplished by separating data into tables of a similar type in each table. Depending on the needs of the project, a Primary Key or Foreign Key is assigned to each table. For each nested attribute or non-atomic attribute in this, we create new relations.

**Second Normal Form**

No non-key attribute should be functionally dependent on a portion of the main key for relations where the primary key has several attributes, according to Second Normal Form. This involves breaking down each partial key into its dependent characteristics and setting up a new relation for each one. Keep the original primary key and any properties that are entirely dependent on it in your database. This procedure aids in removing data that depends only on a small portion of the key. If a connection meets all the requirements for first normal form for the main key and every non-primary key feature of the relation is completely dependent on its primary key alone, then and only then is the relation said to be in second normal form.

**Third Normal Form**

Relation should not have a non-key attribute that is functionally determined by another non-key attribute or by a collection of non-key attributes, according to the Third Normal Form. The primary key should not be transitively dependent, in other words. The non-key attributes that functionally determine other non-key attributes are decomposed in this way put up in relation. This procedure is used to eliminate anything not wholly dependent on the Primary Key. Only when a relation is in second normal form and, more importantly, when its non-key qualities do not depend on any other non-key attributes, is it considered to be in third normal form.

## 4.5 TABLE DESIGN

### 1. tbl\_userdetails

Table Description: Table to store user details

Primary Key:U\_id

Foreign Key:Null

Table 4.5.1: Users

Field	Data type	Constraints
U_id	Int(10)	Primary Key
U_type	Varchar(50)	Not Null
U_Name	Varchar(50)	Not Null
contactno	Varchar(50)	Not Null
Password	Varchar(50)	Not Null
billingAddress	Varchar(50)	Not Null
billingState	Varchar(50)	Not Null
billingCity	varchar(255)	Not Null
billingPincode	int(11)	Not Null
regDate	Text(10)	Not Null
updataDate	varchar(255)	Not Null

---

**2. tbl\_userlog**

Table Description: Table to store user login details

Primary Key:id

Foreign Key:Useremail

Table 4.5.2 : user Login

Field	Data Type	Constraints
id	Int(11)	Primary key
userEmail	varchar(25)	Foreign key
loginTime	timestamp	Not null
logout	Varchar(255)	Not null
status	int(11)	Not null

**3. tbl\_admin**

Table Description: Table to store admin login details

Primary Key:aid

Foreign Key:Null

Table 4.5.3 : Admin

Field	Data Type	Constraints
aid	Int(11)	Primary key
username	varchar(25)	Not null
password	varchar(25)	Not null
creationDate	timestamp	Not null
updationDate	varchar(25)	Not null

**4 . tbl\_products**

Table Description: Table to store products details

Primary Key:pt\_id

Foreign Key:category

Table 4.5.4: Products

Field	Data Type	Constraints
pt_id	Int(11)	Primary key
category	int(11)	Foreign Key
productName	varchar(25)	Not null
productCompany	varchar(25)	Not null
productPrice	varchar(11)	Not null
productPriceBeforeDiscount	varchar(11)	Not null
securitycharge	varchar(20)	Not null
productDescription	varchar(20)	Not null
productImage1	varchar(25)	Not null
productImage2	varchar(25)	Not null
productImage3	varchar(25)	Not null
quantity	varchar(11)	Not null
productAvailability	varchar(25)	Not null
postingDate	timestamp	Not null
updaionDate	varchar(25)	Not null

**5. tbl\_l\_product**

Table Description: Table to store lender products details

Primary Key:lp\_id

Foreign Key: category

Table 4.5.5: l\_ Products

Field	Data Type	Constraints
lp_id	int(100)	Primary key
category	varchar(10)	Foreign key
productname	varchar(20)	Not null
productrent	int(10)	Not null
productdescription	varchar(100)	Not null
productimage1	varchar(100)	Not null
Productimage2	varchar(100)	Not null
commission	int(10)	Not null
returndate	varchar(100)	Not null
status	varchar(100)	Not null

**6. tbl\_category**

Table Description: Table to store products category details

Primary Key:c\_id

Foreign Key: Null

Table 4.5.6:category

Field	Data Type	Constraints
c_id	int(11)	Primary key
categoryName	varchar(100)	Not null
categoryDescription	longtext	Not null
creationDate	timestamp	Not null
updatetime	varchar(100)	Not null

**7. tbl\_adddeliveryboy**

Table Description: Table to store deliveryboy details

Primary Key:d\_id

Foreign Key: Null

Table 4.5.7:adddeliveryboy

Field	Data Type	Constraints
d_id	int(100)	Primary key
lname	varchar(10)	Not null
fname	varchar(10)	Not null
gender	varchar(10)	Not null
email	varchar(10)	Not null
contact	int(10)	Not null
username	varchar(20)	Not null
password	varchar(20)	Not null

**8. tbl\_orders**

Table Description: Table to store order details

Primary Key: id

Foreign Key: userId, productId, quantity, securitycharge, productPrice.

Table 4.5.8: orders

Field	Data Type	Constraints
id	int(11)	Primary key
userId	int(11)	Foreign key
productId	varchar(25)	Foreign key
quantity	int(11)	Foreign key
orderDate	date	Not null
paymentMethod	varchar(50)	Not null
orderStatus	varchar(55)	Not null
returndate	varchar(20)	Not null
productPrice	int(20)	Foreign key
securitycharge	varchar(10)	Foreign key
total	varchar(100)	Not null

**9. tbl\_productreviews**

Table Description: Table to store productreview details

Primary Key: pid

Foreign Key: Null

Table 4.5.9: productreviews

Field	Data Type	Constraints
pid	int(100)	Primary key
quality	varchar(10)	Not null
rent	varchar(10)	Not null
value	varchar(20)	Not null
securityfee	varchar(10)	Not null
name	varchar(10)	Not null
suggestions	varchar(200)	Not null
review	varchar(100)	Not null

---

**10. tbl\_payment**

Table Description: Table to store payment details

Primary Key:pay\_id

Foreign Key: Null

Table 4.5.10:payment

Field	Data Type	Constraints
pay_id	int(100)	Primary key
pay_type	varchar(10)	Not null
pay_date	varchar(10)	Not null
pay_date	varchar(10)	Not null

**11. tbl\_wishlist**

Table Description: Table to store products wish list details

Primary Key:lp\_id

Foreign Key: userId,productId

Table 4.5.5: wishlist

Field	Data Type	Constraints
id	int(11)	Primary key
userId	int(11)	Foreign key
productId	int(11)	Foreign key
postingDate	timestamp	Not null

## **CHAPTER 5**

### **SYSTEM TESTING**



## 5.1 INTRODUCTION

Software testing is the practise of carefully controlling the execution of software to determine whether it behaves as intended. The terms "software testing" and "verification and validation" are frequently used together. Validation is the process of examining or evaluating a product, including software, to determine whether it complies with all relevant specifications. One type of verification, software testing, uses methods including reviews, analyses, inspections, and walkthroughs as well. Verifying that what has been specified matches what the user truly desired is the process of validation.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness is supposed to verify that a program does exactly what it was designed to do.

This is much more difficult than it may at first appear, especially for large programs.

## 5.2 TEST PLAN

A test plan suggests a number of required steps that need be taken in order to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer programme, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfils the purpose for which it was intended. In order to solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). The particular objective soft testing should do the best in numerical terms. Therefore, the expense to uncover and remedy flaws, the theme of failure at any time.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

### 5.2.1 Unit Testing

Unit testing concentrates verification work on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. the level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and it may be done simultaneously for several components. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. To make sure that each statement in a module has been executed, boundary conditions are evaluated.

processing when a mistake does happen. The final step of unit testing is boundary testing. Software frequently fails at its limits. In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

### **5.2.2 Integration Testing**

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a programme structure that has been determined by design using unit tested components. The programme as a whole is tested. Correction is challenging since the size of the overall programme makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All modules were incorporated into the system after unit testing was completed in order to check for any interface consistency issues. A distinctive programme structure also developed when discrepancies in programme structures were eliminated.

### **5.2.3 Validation Testing or System Testing**

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing.

The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every programme requirement.

The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

### **5.2.4 Output Testing or User Acceptance Testing**

The system under consideration is tested for user acceptance; in this case, it must satisfy the business' requirements. The programme should consult the user and the perspective system as it is being developed in order to make any necessary adjustments. This is carried out in relation to the following things:

5.2.4.1 Input Screen Designs,

5.2.4.2 Output Screen Designs,

---

A variety of test data are used to conduct the aforementioned testing. The process of system testing requires the preparation of test data. After the test data preparation, the system under investigation is tested using the test data. The system's flaws are once more discovered during testing, repaired with the help of the aforementioned techniques, and recorded for future use.

### **5.2.5 Selenium Testing**

Selenium is an open-source application that automates web browsers. You can write test scripts using a single interface in a variety of computer languages, including Ruby, Java, NodeJS , PHP, Perl, Python, and C#. The Selenium testing tool automates the cross-browser compatibility testing of web applications. It is used to ensure that web apps are of a high level, whether they are responsive, progressive, or standard. Selenium is a free piece of software.

## Test cases for a Login Page

Project Name: Rentsho Master -Dress Rental System					
Login Test Case					
Test Case ID: login			Test Designed By: Jeena Mathew		
Test Priority (Low/Medium/High): High			Test Designed Date:18-07-2022		
Module Name: Login Screen			Test Executed By: Dr. Bijimol T K		
Test Title: Verify login with valid username and password			Test Execution Date: 18-07-2022		
Description: Test the Login Page					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be display ed	Login page displayed	Pass
2	Provide Valid username	Username :jeena@gmail.com	User should d be able to Login	User Logged in and navigated to User Dashboard	Pass
3	Provide Valid Password	Password:Jeena12			
4	Click on Sign In button				
5	Provide Invalid username or password	Username:jeena@gm ail.com Password: User12345	User should not be able to Login	Message for enter valid email id or password displayed	Pass
6	Provide Null username or Password	Username : null Password: null			
7	Click on Sign In button				

Table 5.2.5.1 Testing

**Post-Condition:** User is validated with database and successfully login into account. The Account session details are logged in database.

### Code package

```
package testing1;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class login {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","C:\\\\Users\\albin\\Downloads\\chromedriver.exe"
    );
        WebDriver driver=new ChromeDriver();

        driver.get("http://localhost/Main%20Project/rentsho-master/login.php");
        driver.findElement(By.id("exampleInputEmail1")).sendKeys("jeena@gmail.com");
        driver.findElement(By.id("exampleInputPassword1")).sendKeys("Jeena12");
        driver.findElement(By.id("login")).click();
        String actualUrl="http://localhost/Main%20Project/rentsho-master/my-cart.php";
        String expectedUrl= driver.getCurrentUrl();
        if(actualUrl.equalsIgnoreCase(expectedUrl)) {
            System.out.println("Test passed");
        } else {
            System.out.println("Test failed");
        }
    }
}
```

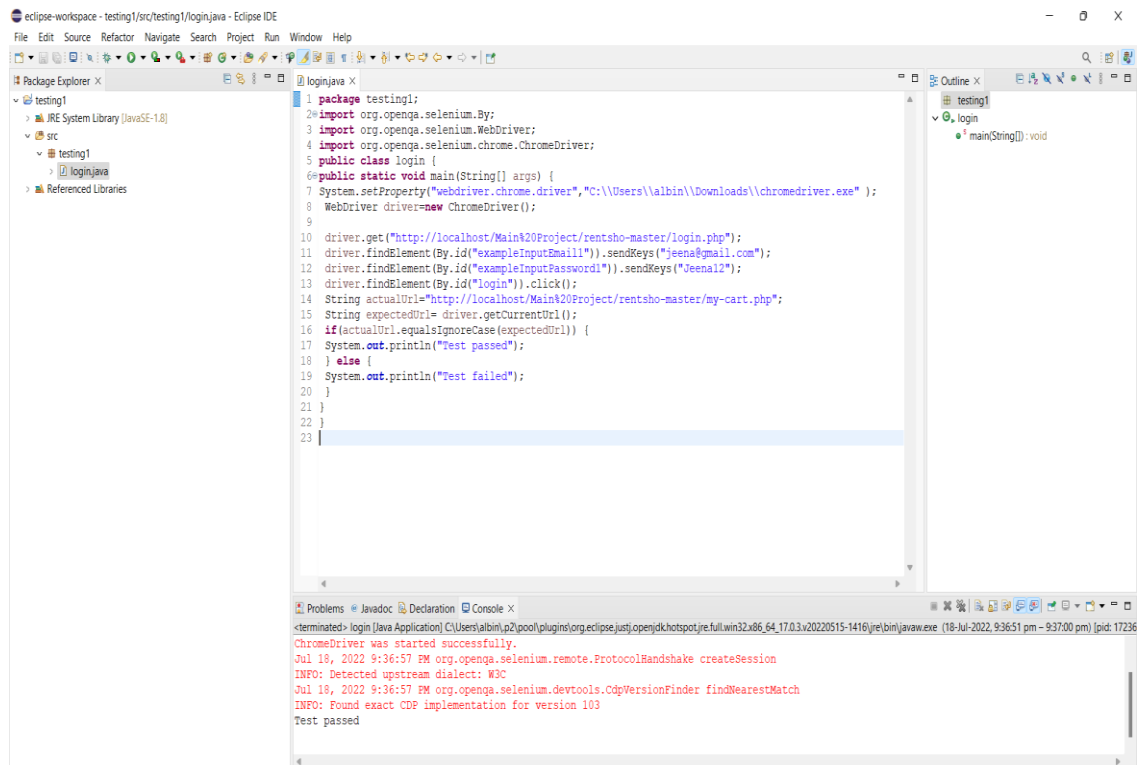


Figure 5.2.51 Testing

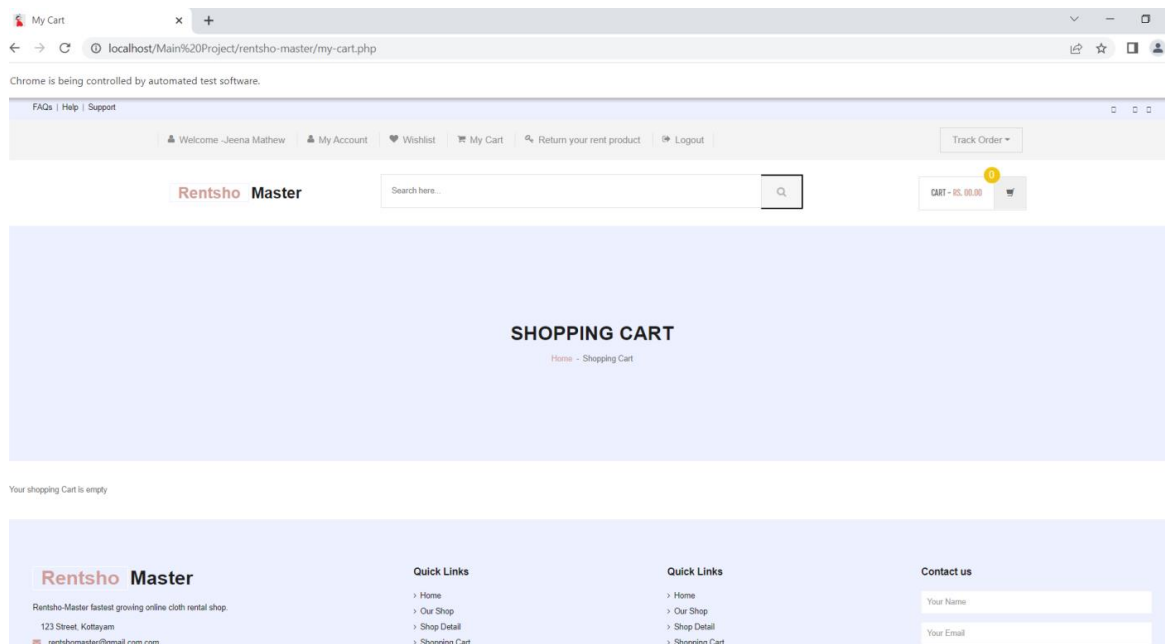


Figure 5.2.2 Testing result

## **CHAPTER 6**

### **IMPLEMENTATION**



## 6.1 INTRODUCTION

The project's implementation phase is where the conceptual design is transformed into a functional system. It can be regarded as the most important stage in creating a successful new system since it gives users assurance that the system will operate as intended and be reliable and accurate. User documentation and training are its main concerns. Usually, conversion happens either during or after the user's training. Implementation is the process of turning a new, revised system design into a standalone operation, and it simply refers to convening a new system design into operation. The user department now has the majority of the burden and the most impact on the current system, according to Haviland. Should the implementation fail. The new system could be entirely different, take the place of an existing manual or automated system, or it could be modified to work better. A reliable system must be implemented properly in order to satisfy organisational needs. System implementation refers to the process of actually using the built system. This comprises all the processes involved in switching from the old to the new system. Only after extensive testing and if it is determined that the system is operating in accordance with the standards can it be put into use. The system personnel assess the system's viability. The system analysis and design effort needed to implement the three principles will increase in complexity as a result of the system being developed.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

## 6.2 IMPLEMENTATION PROCEDURES

Software implementation describes the whole setup of the package in the intended environment, as well as the system's usability and fulfilment of the applications it was designed to support. It's common for someone who won't use the programme to commission the development effort. Early scepticism toward the software is common, but we must be careful to prevent further scepticism by observing the following:

The new system's advantages must be known to the active user.

- Their faith in the software is increased.

- 
- The user receives the appropriate instruction so that he feels at ease using the application.

Before examining the system, the user must be aware that the server software needs to be running on the server in order to access the results. The actual process won't happen if the server object is not active and functioning on the server.

### **6.2.1 User Training**

To prepare the user to test and alter the system is the goal of user training. The participants must have confidence in their capacity to advance the objective and reap the rewards of the computer-based system. As systems get more complex, training becomes increasingly important. Through user training, the user learns how to enter data, respond to error messages, query the database, call up routines that will generate reports, and carry out other crucial tasks.

### **6.2.2 Training on the Application Software**

The user must first obtain the fundamental training in computer literacy, after which they must be taught how to operate the new application software. In addition to how the screens function, what sort of help is shown on them, what kinds of errors are created while entering data, how each entry is validated, and how to update the date that was entered, this will explain the core concepts of how to use the new system. The information needed by the specific user or group to operate the system or a particular component of the system should therefore be covered during the program's training on the application. The user group and hierarchy level may have an impact on how this training is delivered.

### **6.2.3 System Maintenance**

The mystery of system development is maintenance. When a software product is in the maintenance stage of its lifecycle, it is actively working. A system should be properly maintained after it has been effectively implemented. An essential part of the software development life cycle is system maintenance. In order for a system to be flexible to changes in the system environment, maintenance is required.

Software maintenance is of course, far more than "Finding Mistakes".

### 6.2.4 Hosting

A website builder, WordPress support, and no adverts are just a few of the useful features offered by 000webhost, a free website hosting service. More important than the computer space that is provided for Web site files is the fast connection to the Internet. Most hosting services offer connections on T-carrier system lines. Typically, an individual business hosting its own site would require a similar connection and it would be expensive.

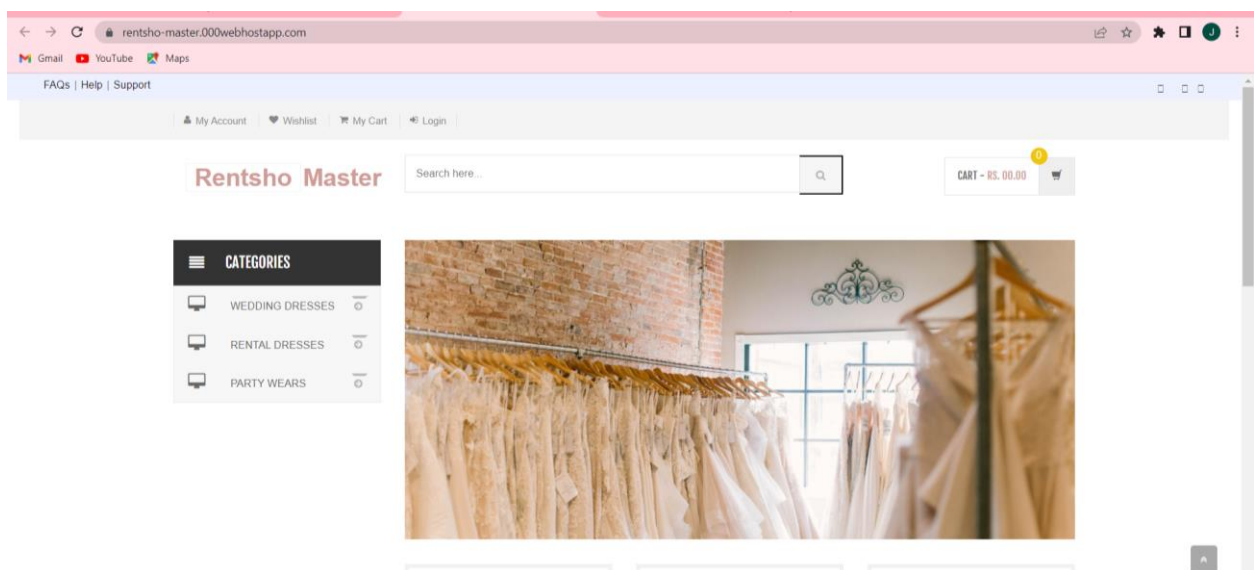


Figure6.2.4.1.Hosting Result

### 6.2.5. Hosting Method Steps

- Go to the 000webhost homepage
- Sign up to make an account.
- Make sure you verify your email.
- Click + button to create a new website.
- Enter your details and click create.
- Click Manage Website to start building website.
- Choose a platform for your website.
- In website Builder, Drag and Drop the project files.
- Insert database in Database Management.
- Get the custom Domain name.
- Publish the website.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## 7.1 CONCLUSION

In comparison to previous experiences, No modern technologies exist in such situation rather giving proper dress rental system. And when every activity related to the dress rental system was restricted to a physical place alone, the dress rental industry has emerged with new delicacies. To overcome the above problem, this web application was found in ease to rent dresses. Even if the physical location has not been completely eliminated, the internet's power has altered the nature of functions and how these tasks are accomplished. Customers may now rent dress in online, and they can be add their own dresses for rent and have the rent product delivered to their home if they are a registered member, or they can travel to the shop to pick up the rental dress.

## **CHAPTER 8**

## **BIBLIOGRAPHY**

**REFERENCES:**

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”,2009.
- Roger S Pressman, “*Software Engineering*”,1994.
- PankajJalote, “*Software engineering: a precise approach*”,2006.
- James lee and Brent ware Addison, “Open source web development with LAMP”, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

**WEBSITES:**

- [www.w3schools.com](http://www.w3schools.com)
- [www.jquery.com](http://www.jquery.com)
- <https://www.fatbit.com/fab/launch-designer-dress-rental-portal-with-top-website-features>
- <http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf>
- <https://www.bing.com/search?q=php+tutorial&cvid=ecf6e80c06744639a6bf650f62a97f0d&aqs=e dge.3.69i57j0l5j69i60l3.4955j0j1&pgt=43&FORM=ANNTA1&PC=U531#>

## **CHAPTER 9**

## **APPENDIX**



## 9.1 Sample Code

### Login. Php

```
<?php
session_start();
error_reporting(0);
include('includes/config.php');
// Code user Registration
if(isset($_POST['submit']))
{
$name=$_POST['fullname'];
$email=$_POST['emailid'];
$contactno=$_POST['contactno'];
$password=$_POST['password'];
$u_type=$_POST['u_type'];
$query=mysqli_query($con,"insert into users(name,email,contactno,password,u_type)
values('$name','$email','$contactno','$password','$u_type')");
if($query)
{
    echo "<script>alert('You are successfully register');</script>";
}
else{
echo "<script>alert('Not register something went worng');</script>";
}
}
// Code for User login

if(isset($_POST['login']))
{
    $email=$_POST['email'];
    $password=(($_POST['password']));
    $password2=$_POST['password'];
```

---

```
$query2=mysqli_query($con,"SELECT * FROM admin WHERE username='$email' and
password='$password2'");
if (mysqli_num_rows($query2) > 0){
    header("Location:Admin_new/home.php");
}
else{
    $query=mysqli_query($con,"SELECT * FROM users WHERE email='$email' and
password='$password'");
    $num=mysqli_fetch_array($query);
    if($num>0)
    {
        $e1="my-cart.php";
        $_SESSION['login']=$_POST['email'];
        $_SESSION['id']=$num['id'];
        $_SESSION['username']=$num['name'];

        $status=1;
        $utype=$num['u_type'];
        echo '<script>alert($utype)</script>';
        if($utype=='lender')
        {
            $e2="Lender/home.php";

            $log=mysqli_query($con,"insert into userlog(userEmail,status)
values('".$_SESSION['login']. "','$status')");

            header("location:Lender/home.php");
            exit();
        }
    }
    else{

        $log=mysqli_query($con,"insert into userlog(userEmail,status)
values('".$_SESSION['login']. "','$status')");
```

---

---

```
header("location:my-cart.php");
exit();
}
}
else{
    $query=mysqli_query($con,"SELECT * FROM adddelivery WHERE email='$email' and
password='$password'");
    $num=mysqli_fetch_array($query);
    if($num>0)
    {
        $e1="my-cart.php";
        $_SESSION['login']=$_POST['email'];
        $_SESSION['id']=$num['id'];
        $_SESSION['username']=$num['name'];

        $status=1;
        $utype=$num['u_type'];
        echo '<script>alert($utype)</script>';
        if($utype=='lender')
        {
            $e2="Lender/home.php";

            $log=mysqli_query($con,"insert into userlog(userEmail,status)
values('".$_SESSION['login']. "','$status')");

            header("location:Lender/home.php");
            exit();
        }
        else{

            $log=mysqli_query($con,"insert into userlog(userEmail,status)
values('".$_SESSION['login']. "','$status')");

            header("location:my-cart.php");
```

---

---

```
exit();
}
else
{
$e1="login.php";
$email=$_POST['email'];

$status=0;
$log=mysqli_query($con,"insert into userlog(userEmail,status) values('$email','$status')");

$_SESSION['errmsg']="Invalid email id or Password";

}
}}
?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Meta -->
    <meta charset="utf-8">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-
scalable=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <meta name="keywords" content="MediaCenter, Template, eCommerce">
    <meta name="robots" content="all">

    <title>Rentsho-Master | Signi-in | Signup</title>

    <!-- Bootstrap Core CSS -->
    <link rel="stylesheet" href="assets/css/bootstrap.min.css">

    <!-- Customizable CSS -->
```

---

---

```
<link rel="stylesheet" href="assets/css/main.css">
<link rel="stylesheet" href="assets/css/green.css">
<link rel="stylesheet" href="assets/css/owl.carousel.css">
<link rel="stylesheet" href="assets/css/owl.transitions.css">
<!--<link rel="stylesheet" href="assets/css/owl.theme.css">-->
<link href="assets/css/lightbox.css" rel="stylesheet">
<link rel="stylesheet" href="assets/css/animate.min.css">
<link rel="stylesheet" href="assets/css/rateit.css">
<link rel="stylesheet" href="assets/css/bootstrap-select.min.css">

<!-- Demo Purpose Only. Should be removed in production -->
<link rel="stylesheet" href="assets/css/config.css">

<link href="assets/css/green.css" rel="alternate stylesheet" title="Green color">
<link href="assets/css/blue.css" rel="alternate stylesheet" title="Blue color">
<link href="assets/css/red.css" rel="alternate stylesheet" title="Red color">
<link href="assets/css/orange.css" rel="alternate stylesheet" title="Orange color">
<link href="assets/css/dark-green.css" rel="alternate stylesheet" title="Darkgreen color"><!--
- Icons/Glyphs -->
<link rel="stylesheet" href="assets/css/font-awesome.min.css"> <!-- Fonts -->
<link href='http://fonts.googleapis.com/css?family=Roboto:300,400,500,700' rel='stylesheet'
type='text/css'>
<!-- Favicon -->
<link rel="shortcut icon" href="assets/rent_logo1.png">
<script type="text/javascript">
function valid()
{
if(document.register.password.value!= document.register.confirmpassword.value)
{
alert("Password and Confirm Password Field do not match !!");
document.register.confirmpassword.focus();
return false;
}
return true;
```

---

---

```

}
</script>
    <script>
function userAvailability() {
$("#loaderIcon").show();
jQuery.ajax({
url: "check_availability.php",
data:'email='+$("#email").val(),
type: "POST",
success:function(data){
$("#user-availability-status1").html(data);
$("#loaderIcon").hide();
},
error:function (){}
});
}
</script></head>
    <body class="cnt-home"><header class="header-style-1">
<?php include('includes/top-header.php');?>
<?php include('includes/main-header.php');?>
<?php include('includes/menu-bar.php');?>
</header><div class="breadcrumb">
    <div class="container">
        <div class="breadcrumb-inner">
            <ul class="list-inline list-unstyled">
                <li><a href="index.php"></a></li>
                <li class='active'>Authentication</li>
            </ul></div><!-- /.breadcrumb-inner -->
        </div><!-- /.container -->
    </div><!-- /.breadcrumb -->
    <div class="body-content outer-top-bd">
        <div class="container">
            <div class="sign-in-page inner-bottom-sm">
                <div class="row">

```

---

---

```

<!-- Sign-in -->
<div class="col-md-6 col-sm-6 sign-in">
<h4 class="">sign in</h4>
<p class="">Hello, Welcome to your account.</p>
<form class="register-form outer-top-xs" method="post">
<span style="color:red;" >
<?php
echo htmlentities($_SESSION['errmsg']);
?>
<?php
echo htmlentities($_SESSION['errmsg']="");
?></span>
<div class="form-group">
<label class="info-title" for="exampleInputEmail1">Email Address <span>*</span></label>
<input type="email" name="email" class="form-control unicast-form-control text-input"
id="exampleInputEmail1" >
</div><div class="form-group">
<label class="info-title" for="exampleInputPassword1">Password <span>*</span></label>
<input type="password" name="password" class="form-control unicast-form-control text-input"
id="exampleInputPassword1" >
</div>
<div class="radio outer-xs">
<a href="forgot-password.php" class="forgot-password pull-right">Forgot your Password?</a>
</div>
<button type="submit" class="btn-upper btn btn-primary checkout-page-button"
name="login">Login</button>
</form>
</div>
<!-- Sign-in -->
<!-- create a new account -->
<div class="col-md-6 col-sm-6 create-new-account">
<h4 class="checkout-subtitle">create a new account</h4>
<p class="text title-tag-line">Create your own Shopping account.</p>
<form class="register-form outer-top-xs" role="form" method="post" name="register"

```

---

---

```

onSubmit="return valid();">
<div class="form-group">
<label class="info-title" for="fullname">Full Name <span>*</span></label>
<input type="text" class="form-control unicast-form-control text-input" id="fullname"
name="fullname" required="required">
</div><div class="form-group">
<label class="info-title" for="exampleInputEmail2">Email Address <span>*</span></label>
<input type="email" class="form-control unicast-form-control text-input" id="email"
onBlur="userAvailability()" name="emailid" required >
<span id="user-availability-status1" style="font-size:12px;"></span>
</div>
<div class="form-group">
<label class="info-title" for="contactno">Contact No. <span>*</span></label>
<input type="text" class="form-control unicast-form-control text-input" id="contactno"
name="contactno" maxlength="10" pattern="[7-9]{1}[0-9]{9}" >
</div>
<div class="form-group">
<label class="info-title" for="password">Password. <span>*</span></label>
<input type="password" class="form-control unicast-form-control text-input" id="password"
name="password" required >
</div>
<div class="form-group">
<label class="info-title" for="confirmpassword">Confirm Password. <span>*</span></label>
<input type="password" class="form-control unicast-form-control text-input"
id="confirmpassword" name="confirmpassword" required >
</div><label for="u_type">Type:</label>
<select name="u_type" id="u_type">
<option value="customer">Customer</option>
<option value="lender">Lender</option> </select>
<p alignment="left">
<button type="submit" name="submit" class="btn-upper btn btn-primary checkout-page-button"
id="submit">Sign Up</button></p>
</form>
<span class="checkout-subtitle outer-top-xs">Sign Up Today And You'll Be Able To : </span>

```

---



---

```

<div class="checkbox">
<label class="checkbox">
Speed your way through the checkout.
</label>
<label class="checkbox">
    Track your orders easily.
</label>
<label class="checkbox">
    Keep a record of all your purchases.
</label>
</div>
</div>
</div>
<?php include('includes/brands-slider.php');?>
</div>
</div>
<?php include('includes/footer.php');?>
<script src="assets/js/jquery-1.11.1.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/bootstrap-hover-dropdown.min.js"></script>
<script src="assets/js/owl.carousel.min.js"></script>
<script src="assets/js/echo.min.js"></script>
<script src="assets/js/jquery.easing-1.3.min.js"></script>
<script src="assets/js/bootstrap-slider.min.js"></script>
<script src="assets/js/jquery.rateit.min.js"></script>
<script type="text/javascript" src="assets/js/lightbox.min.js"></script>
<script src="assets/js/bootstrap-select.min.js"></script>
<script src="assets/js/wow.min.js"></script>
<script src="assets/js/scripts.js"></script>
<!-- For demo purposes – can be removed on production -->
<script src="switchstylesheet/switchstylesheet.js"></script>
<script>$(document).ready(function(){
    $(".change-color").switchstylesheet( { separator:"color" } );
    $('.show-theme-options').click(function(){

```

---

---

```
        $(this).parent().toggleClass('open');
        return false;
    });
});

$(window).bind("load", function() {
    $('.show-theme-options').delay(2000).trigger('click');
});
</script>
</body>
</html>
```

## 9.2 ScreenShots

### Home page

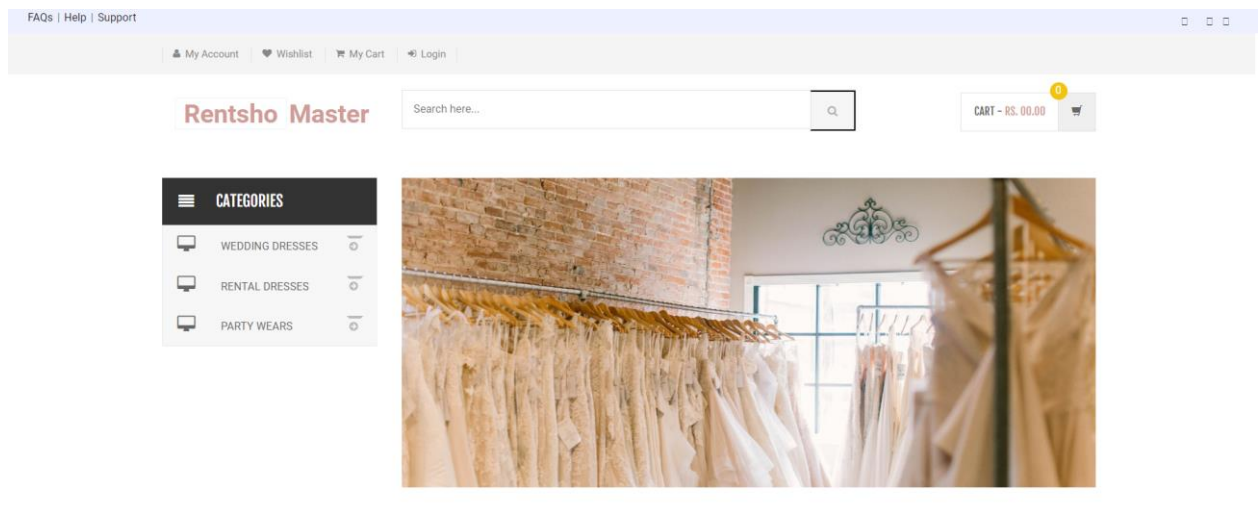


Figure 9.2.1 Home page

### Add cart

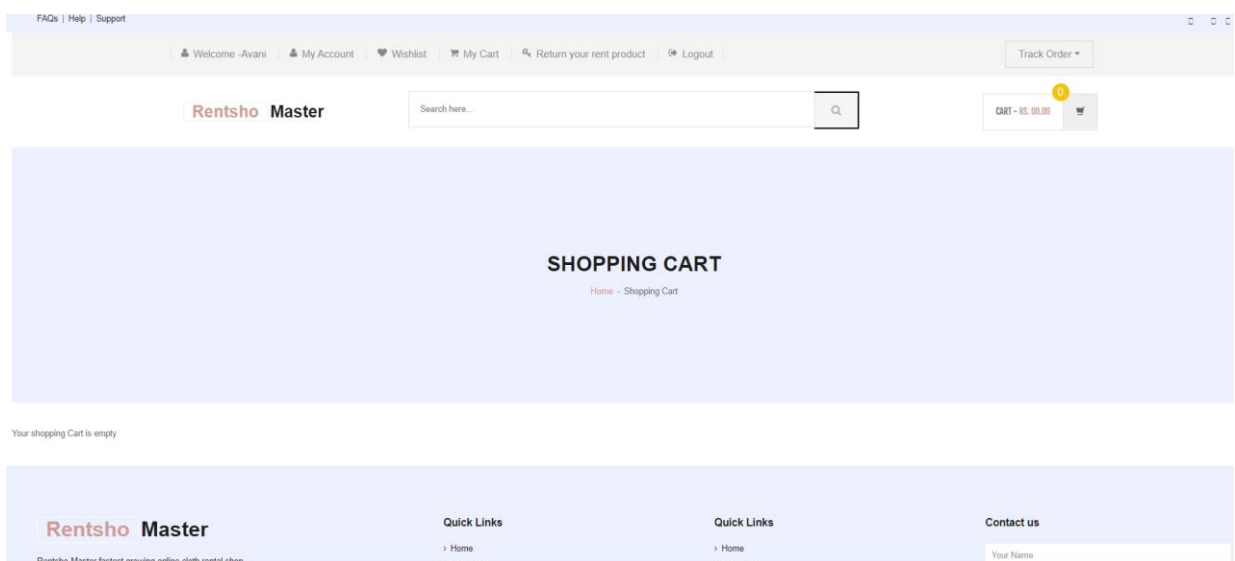


Figure 9.2 .2 shopping cart

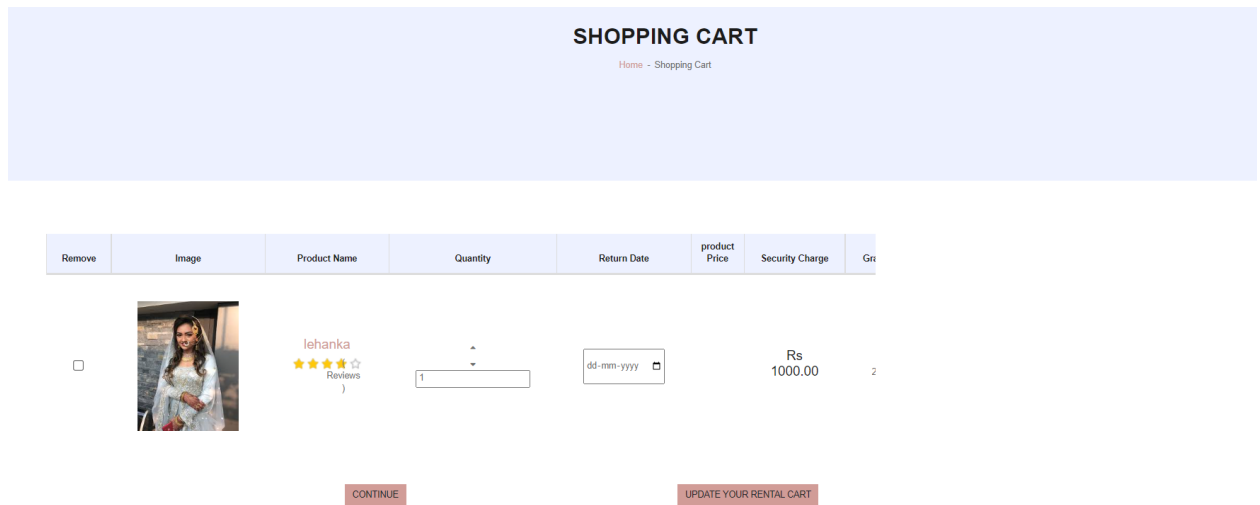


Figure 9.2 .2.1 shopping cart with bill

## Product Details

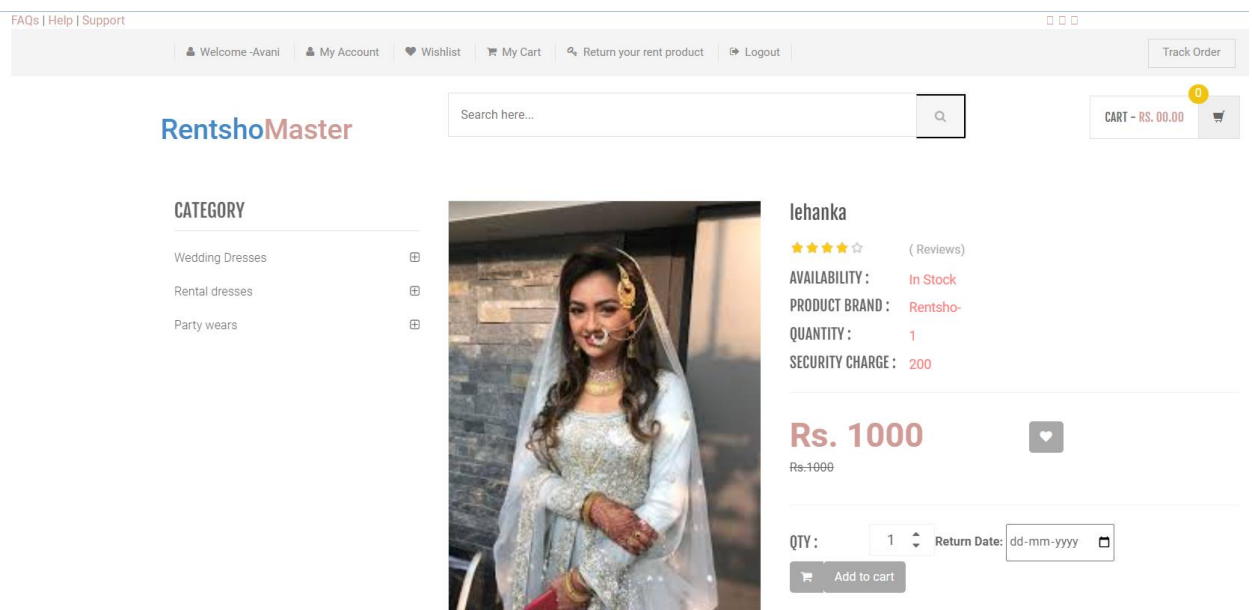


Figure 9.2 .3 Product details

## User Profile page

[Home](#) / [Profile](#)

**MY PROFILE**

**Personal info**

Name\*

Email Address\*

Contact No.\*

**YOUR DETAILS**

- My Account
- Shipping / Billing Address
- Order History
- Payment Pending Order

Figure 9.2.4 User profile

## User Checkout Bill

**SHOPPING CART**  
[Home](#) - [Shopping Cart](#)

**Cart Summary**

<b>Total</b>		<b>2200.00</b>
<input type="button" value="BACK"/>		<input type="button" value="PROCEED TO PAY"/>

Order Date	2022-07-13	Return date	2022-07-14
------------	------------	-------------	------------

Order Date	2022-07-13	Return date	2022-07-22
------------	------------	-------------	------------

Figure 9.2.5 Total Bill

## User Payment

[Welcome -Avani](#) | [My Account](#) | [Wishlist](#) | [My Cart](#) | [Return your rent product](#) | [Logout](#)

Home /

### CHOOSE PAYMENT METHOD

SELECT YOUR PAYMENT METHOD

☒ Pay Online

submit

Quick Links

Quick Links

Con

Figure 9.2.6 Total Bill

## Lender- Add products

### WELCOME

### ADD YOUR RENT PRODUCT

Minimum Commission Require 20%.

Product Name \*

Category Select Category

Product Rent \*

Product Description(size,featur)s \*

Image1 Choose File No file chosen Image2 Choose File No file chosen Return da

ADD

Figure 9.2.7 Lender page to add Product

## Admin Page

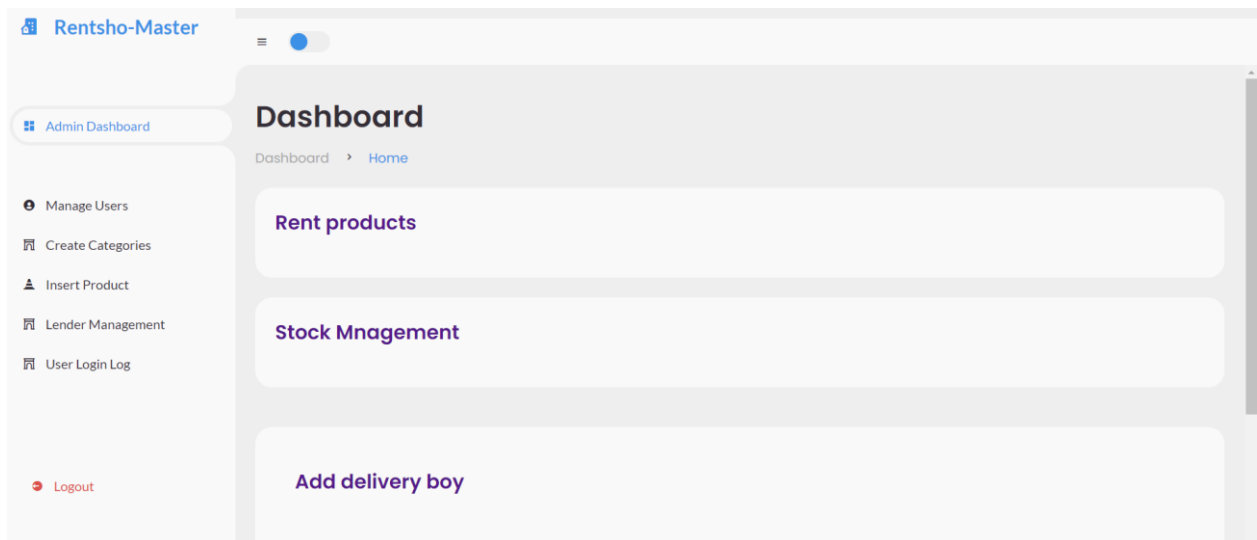


Figure 9.2.8 Admin page

## Admin Create Category

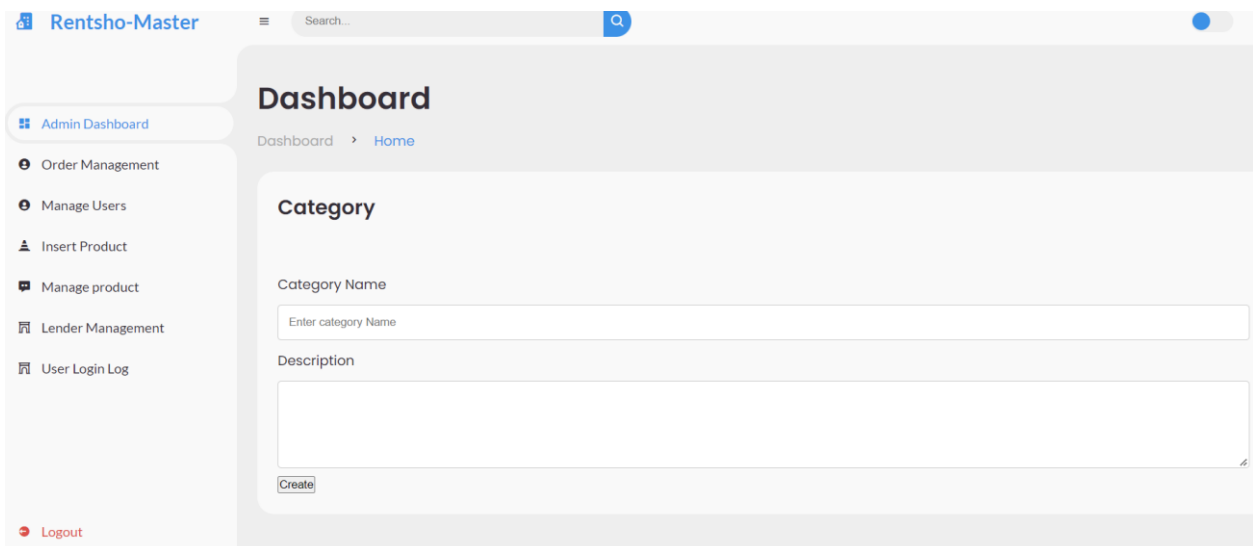
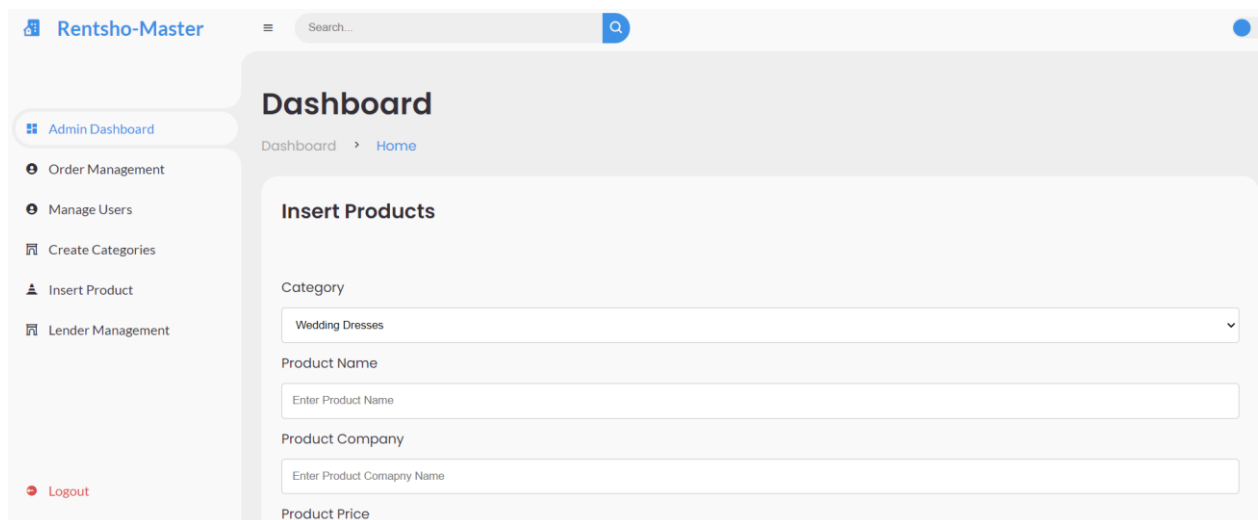


Figure 9.2.9 Add categories

## Admin Add Product



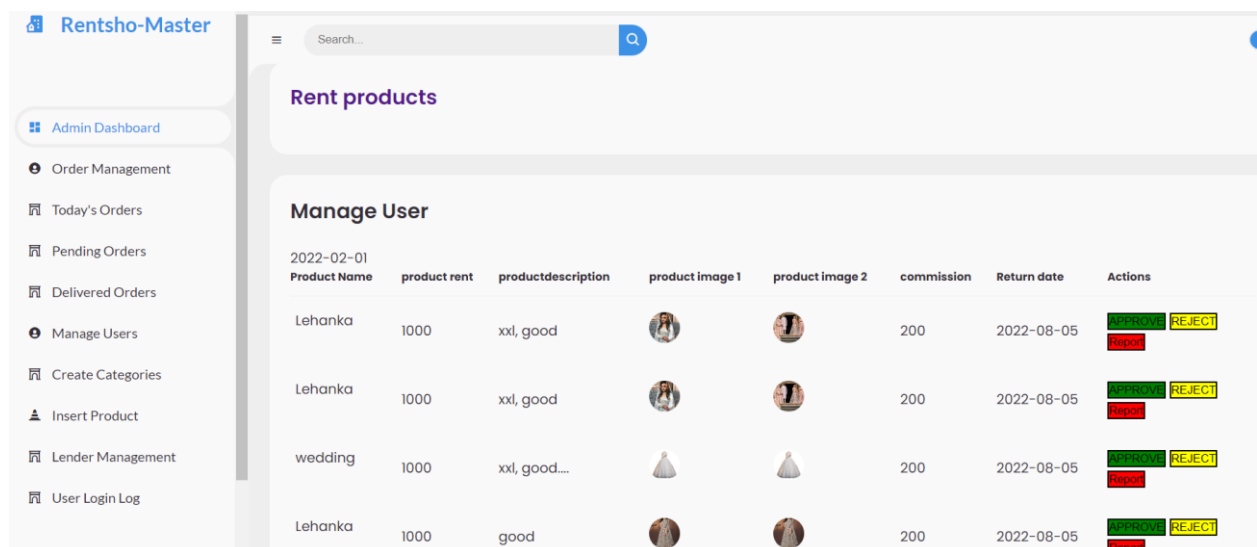
The screenshot shows the 'Admin Dashboard' of the 'Rentsho-Master' system. The left sidebar contains navigation links: Admin Dashboard, Order Management, Manage Users, Create Categories, Insert Product, and Lender Management. The main content area is titled 'Dashboard' and 'Home'. The 'Insert Products' form includes the following fields:

- Category:** A dropdown menu with 'Wedding Dresses' selected.
- Product Name:** A text input field with the placeholder 'Enter Product Name'.
- Product Company:** A text input field with the placeholder 'Enter Product Company Name'.
- Product Price:** A text input field.

A 'Logout' button is located in the bottom left of the sidebar.

Figure 9.2.10 Add product

## Admin Manage Rent product



The screenshot shows the 'Rent products' section of the 'Rentsho-Master' system. The left sidebar contains navigation links: Admin Dashboard, Order Management, Today's Orders, Pending Orders, Delivered Orders, Manage Users, Create Categories, Insert Product, Lender Management, and User Login Log. The main content area is titled 'Manage User' and displays a table of rent products for the date '2022-02-01'.









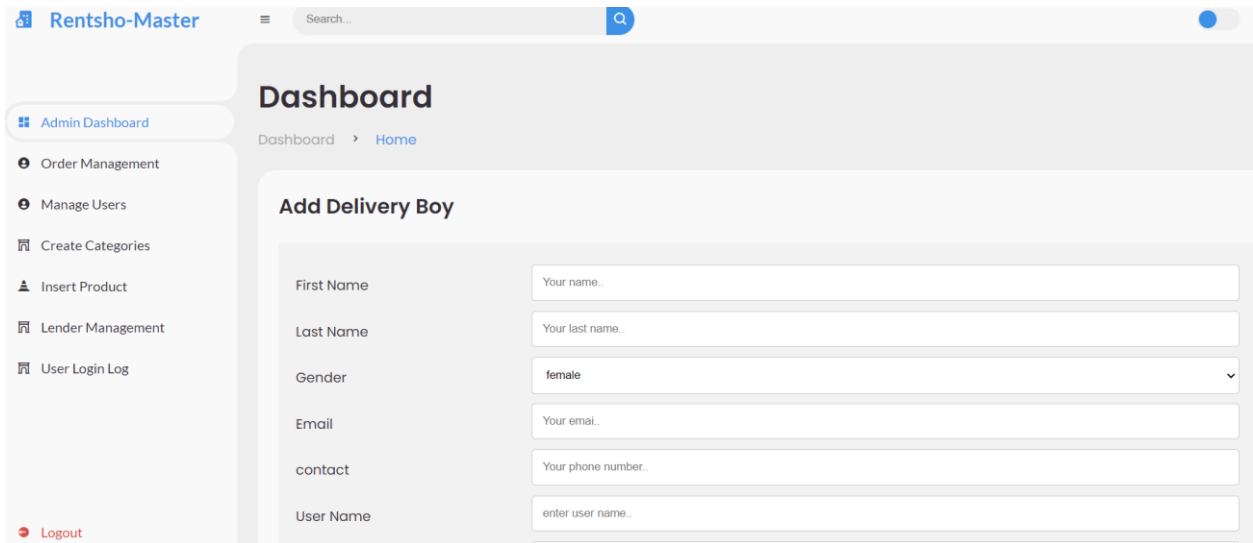
Product Name	product rent	productdescription	product image 1	product image 2	commission	Return date	Actions
Lehanka	1000	xxl, good			200	2022-08-05	<a href="#">APPROVE</a> <a href="#">REJECT</a> <a href="#">Report</a>
Lehanka	1000	xxl, good			200	2022-08-05	<a href="#">APPROVE</a> <a href="#">REJECT</a> <a href="#">Report</a>
wedding	1000	xxl, good....			200	2022-08-05	<a href="#">APPROVE</a> <a href="#">REJECT</a> <a href="#">Report</a>
Lehanka	1000	good			200	2022-08-05	<a href="#">APPROVE</a> <a href="#">REJECT</a> <a href="#">Report</a>

Figure 9.2.11 Lender Product Details



## Admin Add Delivery boy

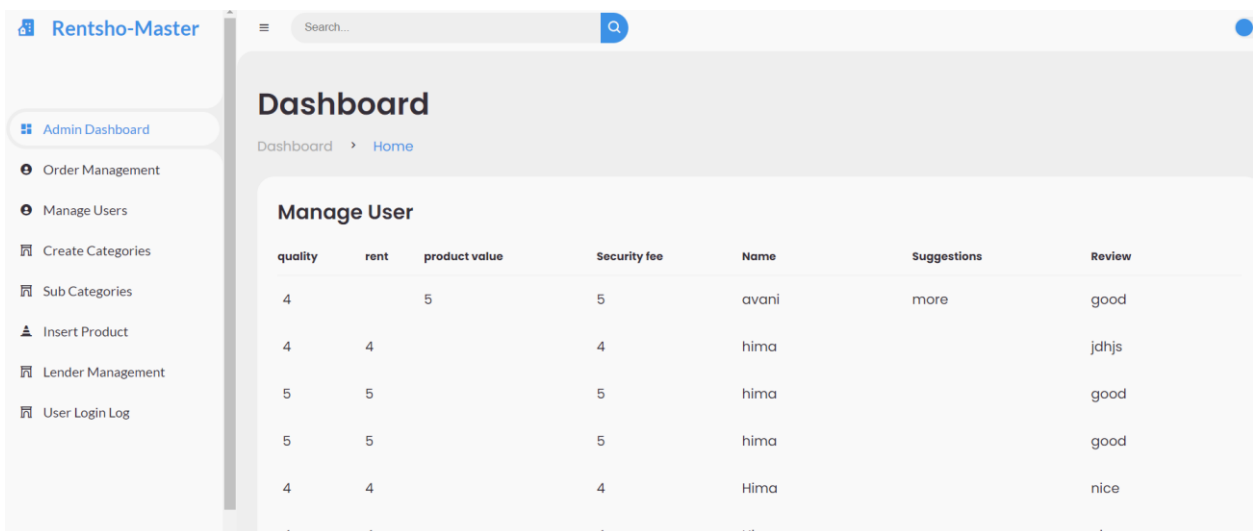


The screenshot shows the 'Rentsho-Master' Admin Dashboard. The left sidebar contains navigation links: Admin Dashboard, Order Management, Manage Users, Create Categories, Insert Product, Lender Management, and User Login Log. The main content area is titled 'Dashboard' and 'Add Delivery Boy'. The form includes the following fields:

- First Name:
- Last Name:
- Gender:
- Email:
- contact:
- User Name:

Figure 9.2.12 Add delivery boy

## Admin View user reviews



The screenshot shows the 'Rentsho-Master' Admin Dashboard with the 'Manage User' table. The table has the following columns: quality, rent, product value, Security fee, Name, Suggestions, and Review. The data rows are as follows:

quality	rent	product value	Security fee	Name	Suggestions	Review
4		5	5	avani	more	good
4	4		4	hima		jdhjs
5	5		5	hima		good
5	5		5	hima		good
4	4		4	Hima		nice
4	4		4	Hima		nice

Figure 9.2.13 user reviews

## Delivery boy login page

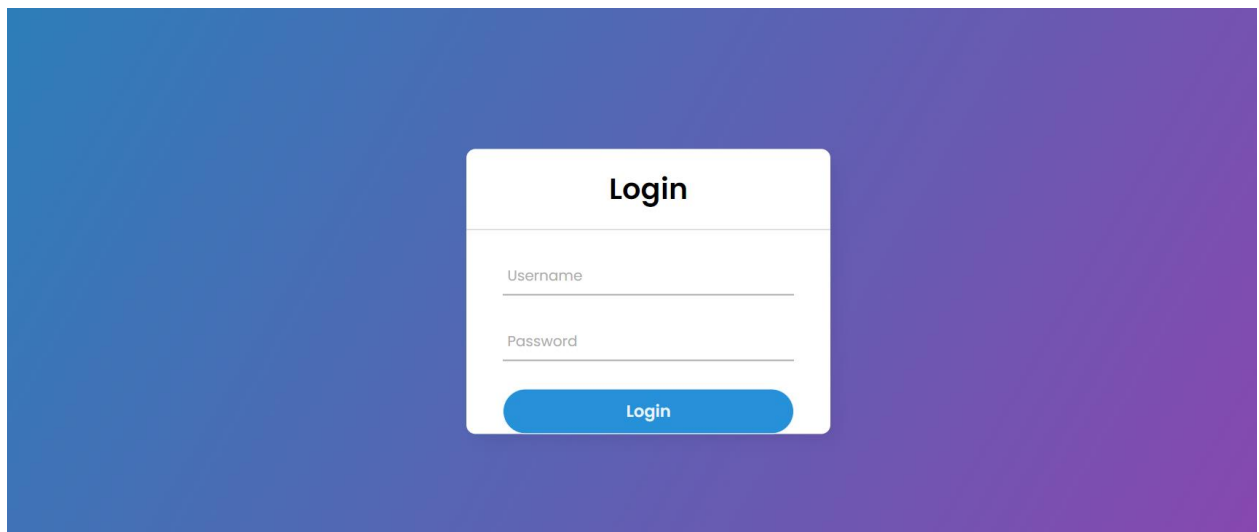


Figure 9.2.14 Delivery boy login

## Delivery boy page

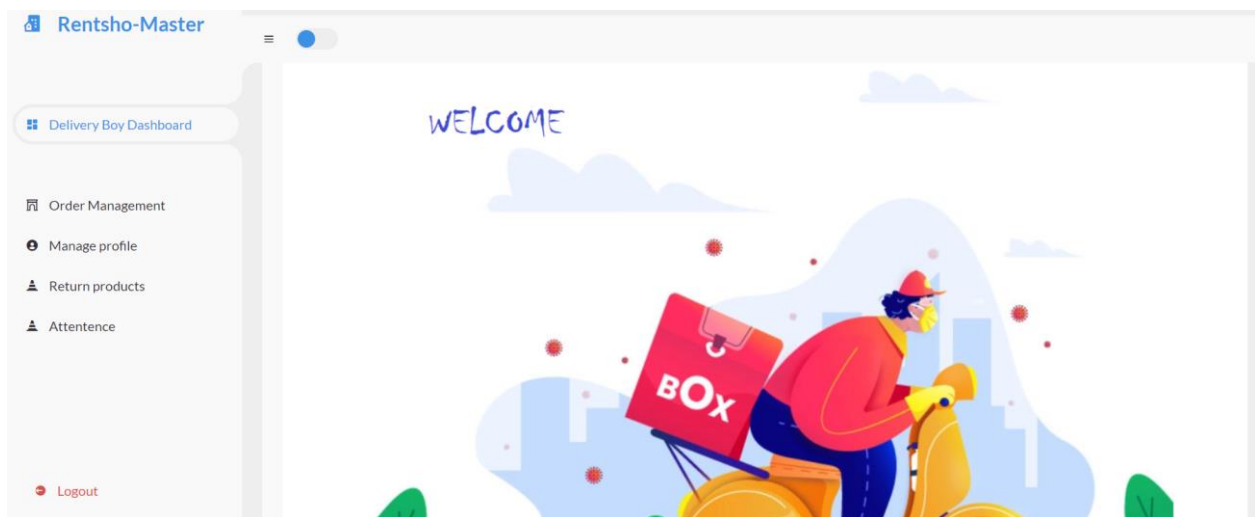


Figure 9.2.15 Home Page