

# **CASA DECOR**

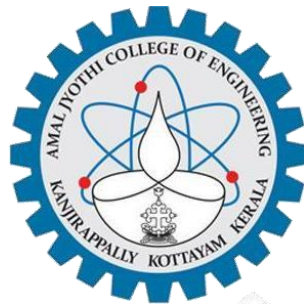
*Project Report Submitted By*

**MEKHA RAMLAL**

**Reg. No.: AJC17MCA-I038**

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

**INTEGRATED MASTER OF COMPUTER APPLICATIONS  
(INMCA)  
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, “**CASA DECOR**” is the bonafide work of **MEKHA RAMLAL (Reg.No: AJC17MCA-I038)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

**Dr. BIJIMOL TK**  
**Internal Guide**

**Ms. MEERA ROSE MATHEW**  
**Coordinator**

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

**External Examiner**

## **DECLARATION**

I hereby declare that the project report “**CASA DECOR**” 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 Integrated Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

**Date: 20/05/2022**

**KANJIRAPPALLY**

**MEKHA RAMLAL**

**Reg. No: AJC17MCA-I038**

## 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 (Head of Department)** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew (Assistant Professor)** for her 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 (Assistant Professor)** 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.

MEKHA RAMLAL

## **ABSTRACT**

**CASA DÉCOR** is a web application to purchase Home Decoration items from anywhere and anytime.

The Internet has become an essential part of our daily life, and companies realize that the Internet can be a shopping avenue to reach existing and potential consumers of any business.

Online shopping has been known as a rapidly growing business, and although online shopping has not followed these same growth patterns in the past, it is now being recognized for its potential. Now consumers approach and attitudes towards online shopping is rapidly increased. Now people prefer online shopping rather than offline shopping, because it is very easy, time consuming, will get brand products, have return opportunity and can order product from anywhere, anytime.

In this project, the main aim is to demonstrate that with better interaction features in decoration websites could improve sales over the internet in Casa Décor. The CASA DÉCOR presents an online display of an order and an associated delivery window for items selected by the customers. An online platform for buying decoration items & accessories for your interior, which is most important segment in setting the complete ambience of your home. The system consists of decoration items such as plants, tables, mirrors, lamps etc.”

In this contemporary society, bounded by the search of constant technological advances and innovations, consumers are becoming less and less loyal to any specific brand or retail format and increasingly focus on the satisfaction of immediate goals and needs . Consequently, online shopping has become a highly profitable retail format, achieving high sales values across much of the developed world.

# CONTENT

Sl. No	Topic	Page No
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>1.1</b>	<b>PROJECT OVERVIEW</b>	<b>2</b>
<b>1.2</b>	<b>PROJECT SPECIFICATION</b>	<b>2</b>
<b>2</b>	<b>SYSTEM STUDY</b>	<b>3</b>
<b>2.1</b>	<b>INTRODUCTION</b>	<b>4</b>
<b>2.2</b>	<b>EXISTING SYSTEM</b>	<b>5</b>
<b>2.3</b>	<b>DRAWBACKS OF EXISTING SYSTEM</b>	<b>5</b>
<b>2.4</b>	<b>PROPOSED SYSTEM</b>	<b>5</b>
<b>2.5</b>	<b>ADVANTAGES OF PROPOSED SYSTEM</b>	<b>6</b>
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>7</b>
<b>3.1</b>	<b>FEASIBILITY STUDY</b>	<b>8</b>
<b>3.1.1</b>	<b>ECONOMICAL FEASIBILITY</b>	<b>8</b>
<b>3.1.2</b>	<b>TECHNICAL FEASIBILITY</b>	<b>9</b>
<b>3.1.3</b>	<b>BEHAVIORAL FEASIBILITY</b>	<b>9</b>
<b>3.2</b>	<b>SYSTEM SPECIFICATION</b>	<b>10</b>
<b>3.2.1</b>	<b>HARDWARE SPECIFICATION</b>	<b>10</b>
<b>3.2.2</b>	<b>SOFTWARE SPECIFICATION</b>	<b>10</b>
<b>3.3</b>	<b>SOFTWARE DESCRIPTION</b>	<b>10</b>
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>13</b>
<b>4.1</b>	<b>INTRODUCTION</b>	<b>14</b>
<b>4.2</b>	<b>UML DIAGRAM</b>	<b>14</b>
<b>4.2.1</b>	<b>USE CASE DIAGRAM</b>	<b>15</b>
<b>4.2.2</b>	<b>SEQUENCE DIAGRAM</b>	<b>18</b>
<b>4.2.3</b>	<b>COLLABORATION DIAGRAM</b>	<b>20</b>
<b>4.2.4</b>	<b>STATE CHART DIAGRAM</b>	<b>22</b>
<b>4.2.5</b>	<b>ACTIVITY DIAGRAM</b>	<b>24</b>
<b>4.2.6</b>	<b>CLASS DIAGRAM</b>	<b>26</b>
<b>4.2.7</b>	<b>OBJECT DIAGRAM</b>	<b>29</b>
<b>4.2.8</b>	<b>COMPONENT DIAGRAM</b>	<b>30</b>
<b>4.2.9</b>	<b>DEPLOYMENT DIAGRAM</b>	<b>31</b>

<b>4.5</b>	<b>USER INTERFACE DESIGN USING FIGMA</b>	<b>33</b>
<b>4.5.1</b>	<b>INPUT DESIGN</b>	<b>33</b>
<b>4.5.2</b>	<b>OUTPUT DESIGN</b>	<b>35</b>
<b>4.6</b>	<b>DATA BASE DESIGN</b>	<b>37</b>
<b>4.6.1</b>	<b>RELATIONAL DATABASE MANAGEMENT SYSTEM</b>	<b>37</b>
<b>4.6.2</b>	<b>NORMALIZATION</b>	<b>38</b>
<b>4.6.3</b>	<b>TABLE DESIGN</b>	<b>40</b>
<b>5</b>	<b>SYSTEM TESTING</b>	<b>45</b>
<b>5.1</b>	<b>INTRODUCTION</b>	<b>46</b>
<b>5.2</b>	<b>TEST PLAN</b>	<b>47</b>
<b>5.2.1</b>	<b>UNIT TESTING</b>	<b>47</b>
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	<b>52</b>
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	<b>53</b>
<b>5.2.4</b>	<b>USER ACCEPTANCE TASTING</b>	<b>53</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>54</b>
<b>6.1</b>	<b>INTRODUCTION</b>	<b>55</b>
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	<b>55</b>
<b>6.2.1</b>	<b>USER TRAINING</b>	<b>56</b>
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	<b>56</b>
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	<b>57</b>
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>58</b>
<b>7.1</b>	<b>CONCLUSION</b>	<b>59</b>
<b>7.2</b>	<b>FUTURE SCOPE</b>	<b>60</b>
<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>61</b>
<b>9</b>	<b>APPENDIX</b>	<b>62</b>
<b>9.1</b>	<b>SAMPLE CODE</b>	<b>63</b>
<b>9.2</b>	<b>SCREEN SHOTS</b>	<b>85</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

**CASA DÉCOR** is a web application to purchase Home Decoration items from anywhere and anytime. This is meant to help the customers to make purchase home decoration items. The customer can also reduce the time, effort and cost in searching items by using this system. Customer can easily find decoration items which is suitable according to many factors. The proposed system includes two users they are administrator and customer. Registered customers can login to the site and can search the items he wants to purchase and can also make the payment through online.

## 1.2 PROJECT SPECIFICATION

The proposed system is a website in which user can book online for servicing. Also that the customers how come to service center also have access to the web where he/she can view all the service history of their vehicle.

The system includes 3 modules. They are:

### 1. Admin Module

Admin must have a login into this system. He has the overall control of the system. Admin can add or update service details, manage user data etc. Admin can View all the registered users, orders, shop details and also manage all his data. Admin can add shop and send login details to each shop where added.

### 2. Customer Module

Customer can register and they can book for service and also view also information about His/her purchase.

### 3. Shop Module

Shop can add product and if any customer purchase product from that shop, the shop Will get the amount of the product and deduct site charge from the customer.

## **CHAPTER 2**

### **SYSTEM STUDY**

## 2.1 INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

## **2.2 EXISTING SYSTEM**

Existing system is not a fully automated system. Customer can register and they can book for service. Each customer can create their own profile .The proposed system rectify the drawbacks of the present system.

It is necessary to modify the existing system in order to include additional information and make the system efficient, flexible and secure. Using the new system customers can view all information about his order, bill etc.

## **2.3 DRAWBACKS OF EXISTING SYSTEM**

- No proper online management of system
- Human effort is needed.
- It is difficult to maintain important information in books.
- More manual hours need to generate required reports.

## **2.4 PROPOSED SYSTEM**

The proposed system is defined to meets all the disadvantages of the existing system. It is necessary to have a system that is more user friendly and user attractive for growth of service center; on such consideration the system is proposed. In our proposed system there is admin who can view all the customers. It allows customers to make their service booking and do their transactions by using online payment method .Users of this proposed system are admin and customer. The software application which avoids more manual hours that needs to spend in record keeping and generating reports. This application keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the distributors to use this application. They can easily use the tool that decreases manual hours spending for normal things and hence increases the performance. It is very easy to record the information of online sales and purchases in the databases

## 2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources, and the system will work in almost all configurations. It has got following features:

➤ **Better security: -**

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.

➤ **Ensure data accuracy: -**

The proposed system eliminates the manual errors while entering the details of the users during the registration.

➤ **Better service: -**

The product will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

## **CHAPTER 3**

# **REQUIREMENT ANALYSIS**

### 3.1 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

#### 3.1.1 Economical Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

The cost of project, DREAMS was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software.



### 3.1.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

### 3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

---

## 3.2 SYSTEM SPECIFICATION

### 3.2.1 Hardware Specification

Processor - Intel core i3

RAM - 4 GB

Hard disk - 1 TB

### 3.2.2 Software Specification

Front End - HTML, CSS

Backend - MYSQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, PHP, CSS

## 3.3 SOFTWARE DESCRIPTION

### 3.3.1 PHP

PHP is a server side scripting language designed for web development but also used as a general purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. Originally created by Rasmus Ledorf in 1995, the reference implementation of PHP is now produced by the PHP group. While PHP originally stood for personal Home page, it now stands for PHP: Hypertext Preprocessor, a recursive acronym. PHP code is interpreted by a web server with a PHP processor module which generates the resulting web page. PHP commands can be embedded directly into a HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

### 3.3.2 MySQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. The MySQL Web site provides the latest information about MySQL software.

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

A database is a structured collection of data. It may be anything from a simple shopping list to a 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 set up 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 inconsistent, 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.**

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available.

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

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

## **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 defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

## 4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object

oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Statechart 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 the 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 drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.





Figure (a) – Use Case Diagram

### 4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

#### Sequence Diagram Notations –

- i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
- ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram
- iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

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** – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular 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

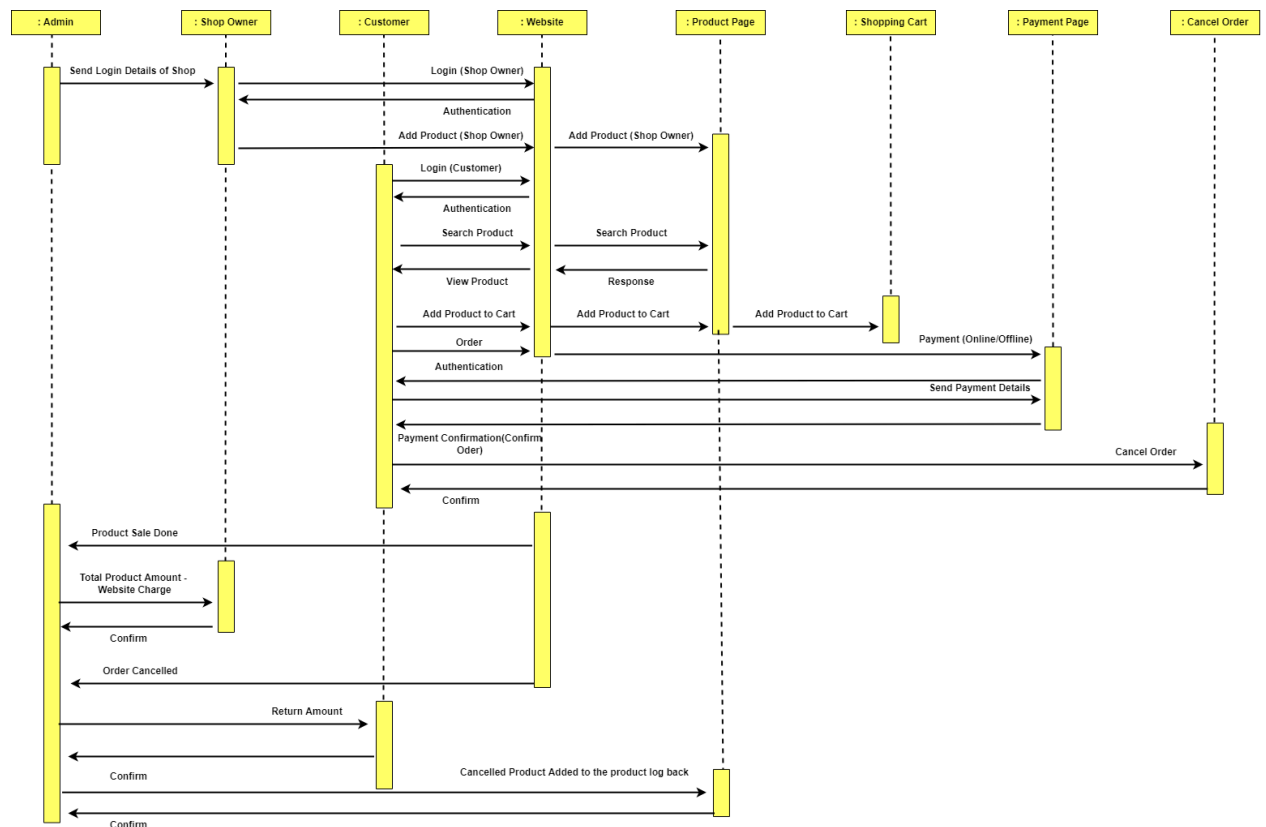


Figure (b) – Sequence Diagram

### 4.2.3 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

#### Notations of a Collaboration Diagram

Following are the components of a component diagram that are enlisted below:

1. **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.  
In the collaboration diagram, objects are utilized in the following ways:
  - The object is represented by specifying their name and class.
  - It is not mandatory for every class to appear.
  - A class may constitute more than one object.
  - In the collaboration diagram, firstly, the object is created, and then its class is specified.
  - To differentiate one object from another object, it is necessary to name them.
2. **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
3. **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.
4. **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

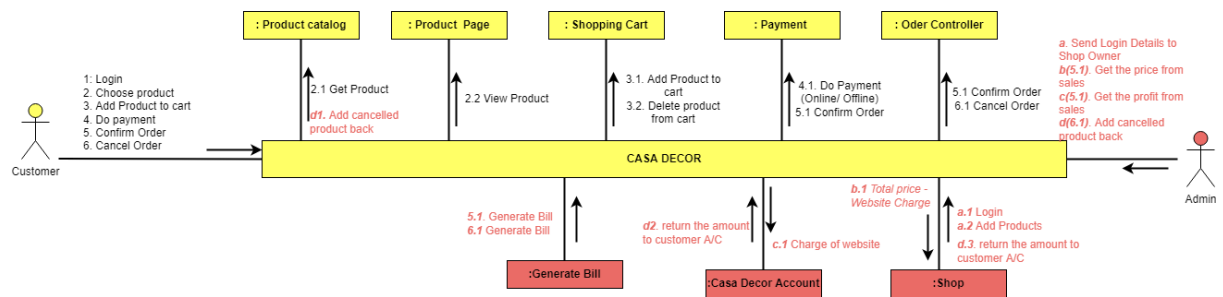


Figure (c) - Collaboration Diagram

#### 4.2.4 STATE CHART DIAGRAM

The state machine diagram is also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system. It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.

It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system. It models event-based systems to handle the state of an object. It also defines several distinct states of a component within the system. Each object/component has a specific state.

#### Notation of a State Machine Diagram

Following are the notations of a state machine diagram enlisted below:

- a) **Initial state:** It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
- b) **Final state:** It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
- c) **Decision box:** It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.
- d) **Transition:** A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued.
- e) **State box:** It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.

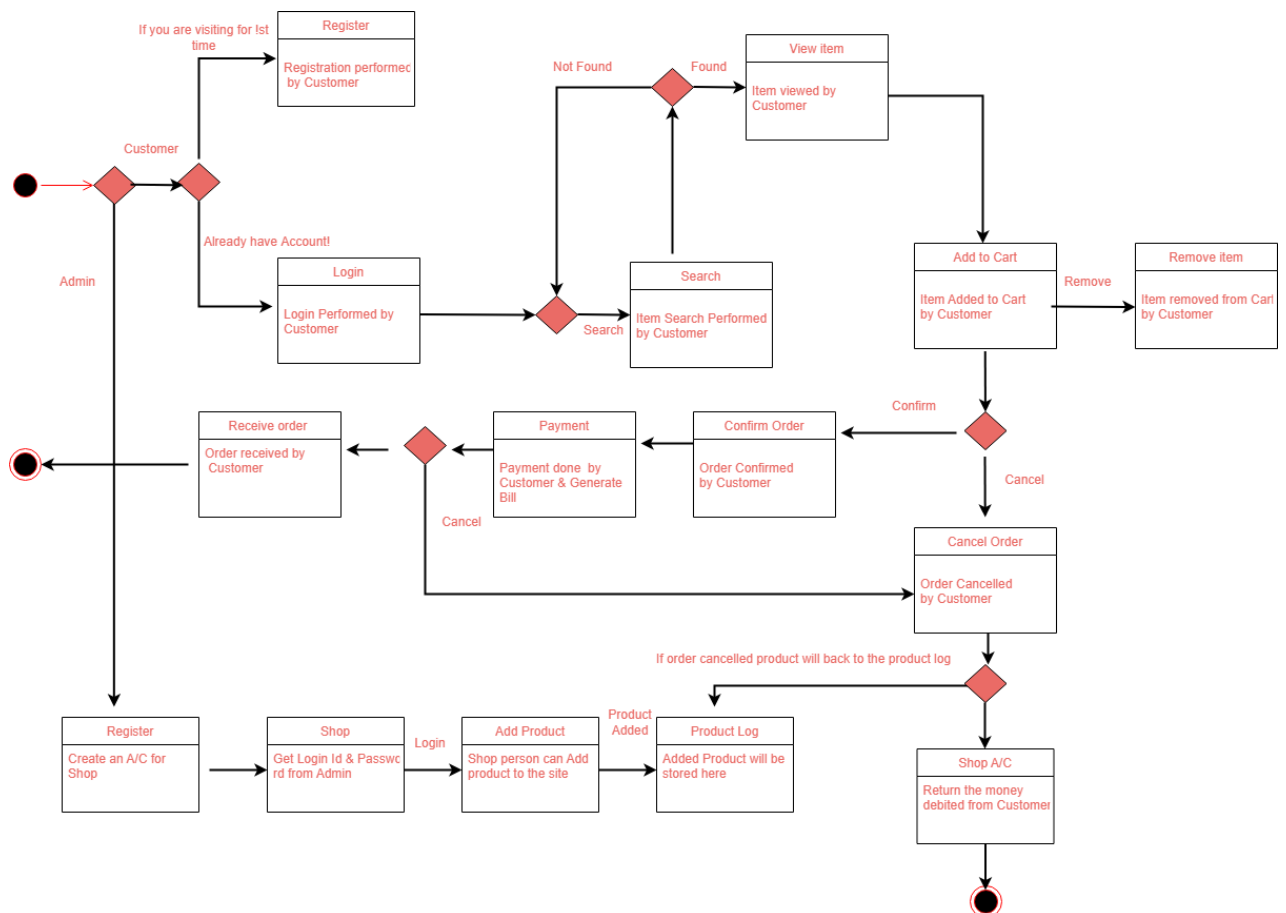


Figure (d) – State Chart Diagram

### 4.2.5 ACTIVITY DIAGRAM

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc. It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

#### Components of an Activity Diagram

Following are the component of an activity diagram:

##### a) Activities

The categorization of behavior into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.

The control flow of activity is represented by control nodes and object nodes that illustrates the objects used within an activity. The activities are initiated at the initial node and are terminated at the final node.

##### b) Activity partition /swimlane

The swimlane is used to cluster all the related activities in one column or one row. It can be either vertical or horizontal. It used to add modularity to the activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.

##### c) Forks



Forks and join nodes generate the concurrent flow inside the activity. A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.

#### **d) Join Nodes**

Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.

#### **Notation of an Activity diagram**

Activity diagram constitutes following notations:

**Initial State:** It depicts the initial stage or beginning of the set of actions.

**Final State:** It is the stage where all the control flows and object flows end.

**Decision Box:** It makes sure that the control flow or object flow will follow only one path.

**Action Box:** It represents the set of actions that are to be performed.

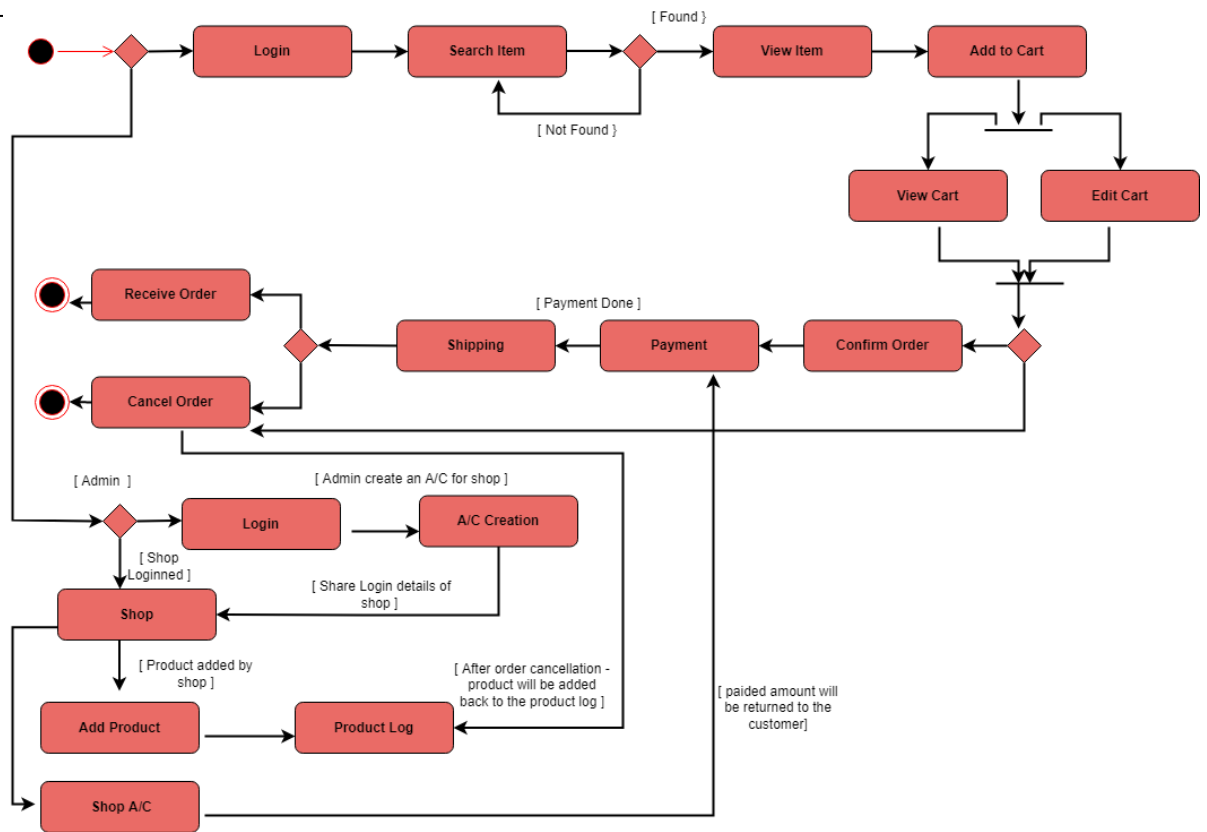


Figure (e) - Activity Diagram

#### 4.2.6 CLASS DIAGRAM

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

##### Components of a Class Diagram

The class diagram is made up of three sections:

**Upper Section:** The upper section encompasses the name of the class. A class is a representation of similar objects that shares the same relationships,

attributes, operations, and semantics. Some of the following rules that should be taken into account while representing a class are given below:

- a. Capitalize the initial letter of the class name.
- b. Place the class name in the center of the upper section.
- c. A class name must be written in bold format.
- d. The name of the abstract class should be written in italics format.

**Middle Section:** The middle section constitutes the attributes, which describe the quality of class.

The attributes have the following characteristics:

- The attributes are written along with its visibility factors, which are public (+), private (-), protected (#), and package (~).
- The accessibility of an attribute class is illustrated by the visibility factors.
- A meaningful name should be assigned to the attribute, which will explain its usage inside the class.

**Lower Section:** The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.

## Relationships

In UML, relationships are of three types:

- **Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class cause changes in another class. It forms a weaker relationship.
- **Generalization:** A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class.
- **Association:** It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship.

**Multiplicity:** It defines a specific range of allowable instances of attributes. In case if a range is not specified, one is considered as a default multiplicity.

**Aggregation:** An aggregation is a subset of association, which represents has a relationship. It is more specific than association. It defines a part-whole or part-of relationship. In this kind of relationship, the child class can exist independently of its parent class.

**Composition:** The composition is a subset of aggregation. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.

### **Abstract Classes**

In the abstract class, no objects can be a direct entity of the abstract class. The abstract class can neither be declared nor be instantiated. It is used to find the functionalities across the classes. The notation of the abstract class is similar to that of class; the only difference is that the name of the class is written in italics. Since it does not involve any implementation for a given function, it is best to use the abstract class with multiple objects.

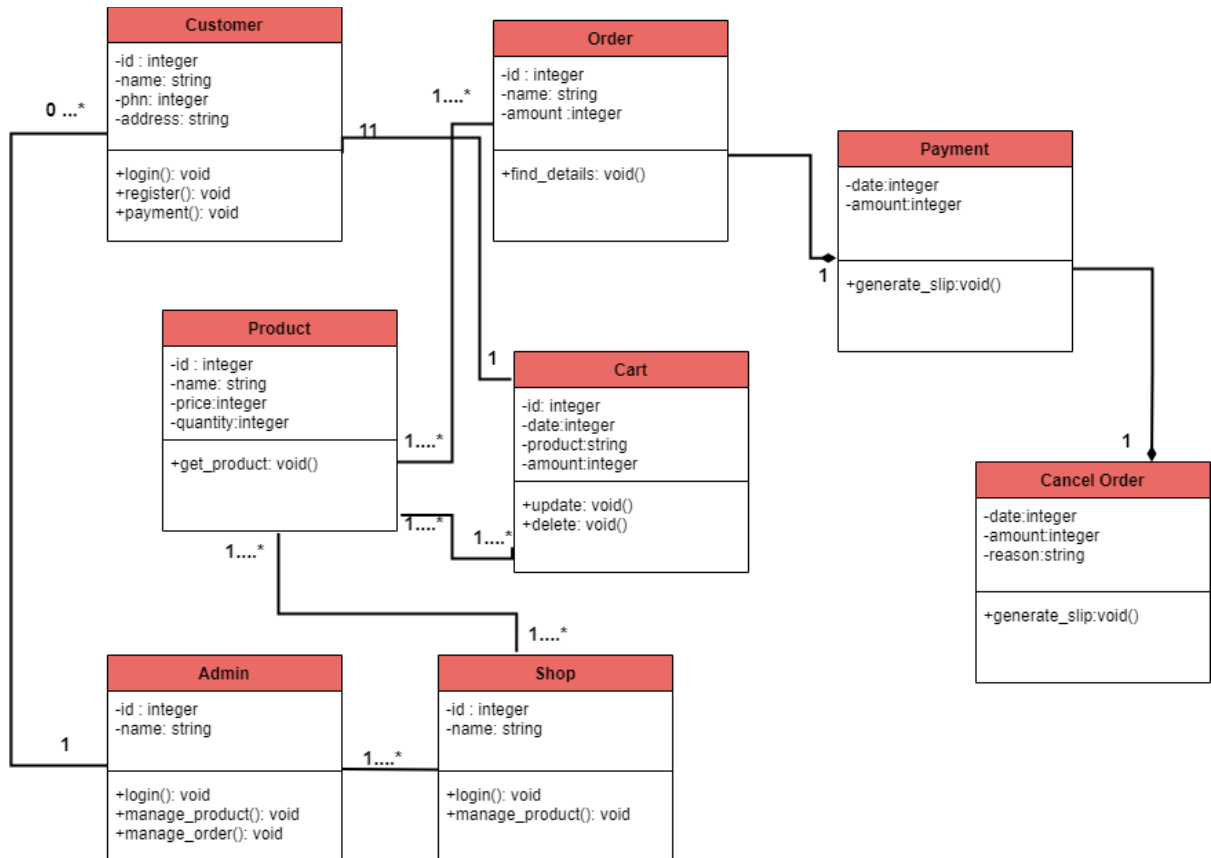


Figure (f) – Class Diagram

#### 4.2.7 OBJECT DIAGRAM

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant.

Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

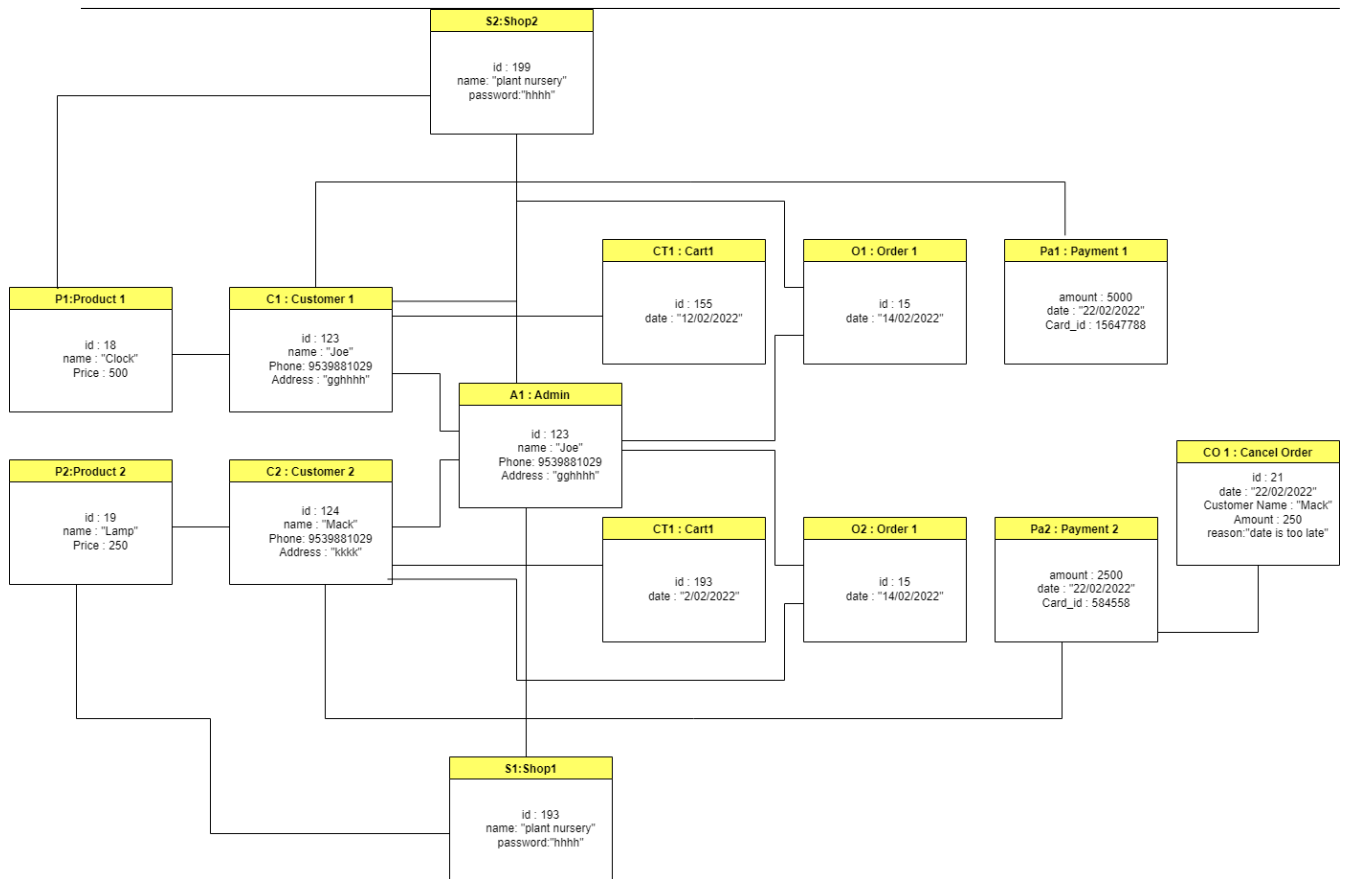


Figure (g) – Object Diagram

## 4.2.8 COMPONENT DIAGRAM

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

### Notation of a Component Diagram

a) A component

b) A node

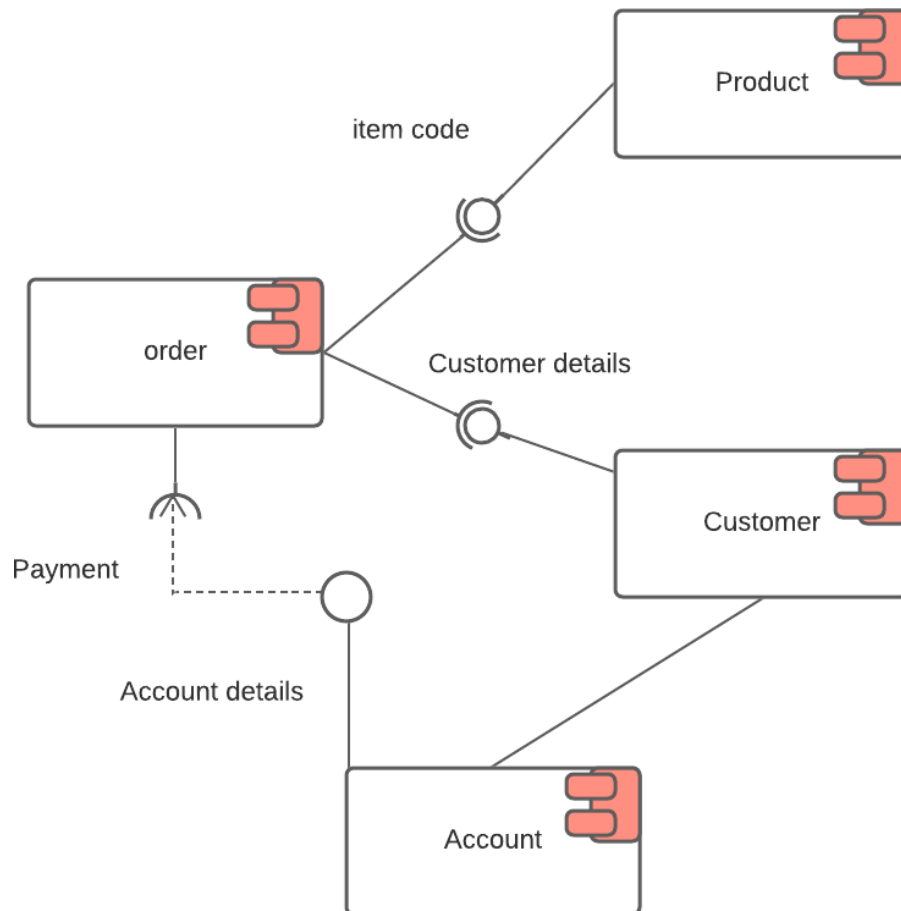


Figure (h) – Component Diagram

#### 4.2.9 DEPLOYMENT DIAGRAM

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

##### Notation of Deployment diagram

The deployment diagram consists of the following notations:

1. A component
2. An artifact
3. An interface
4. A node

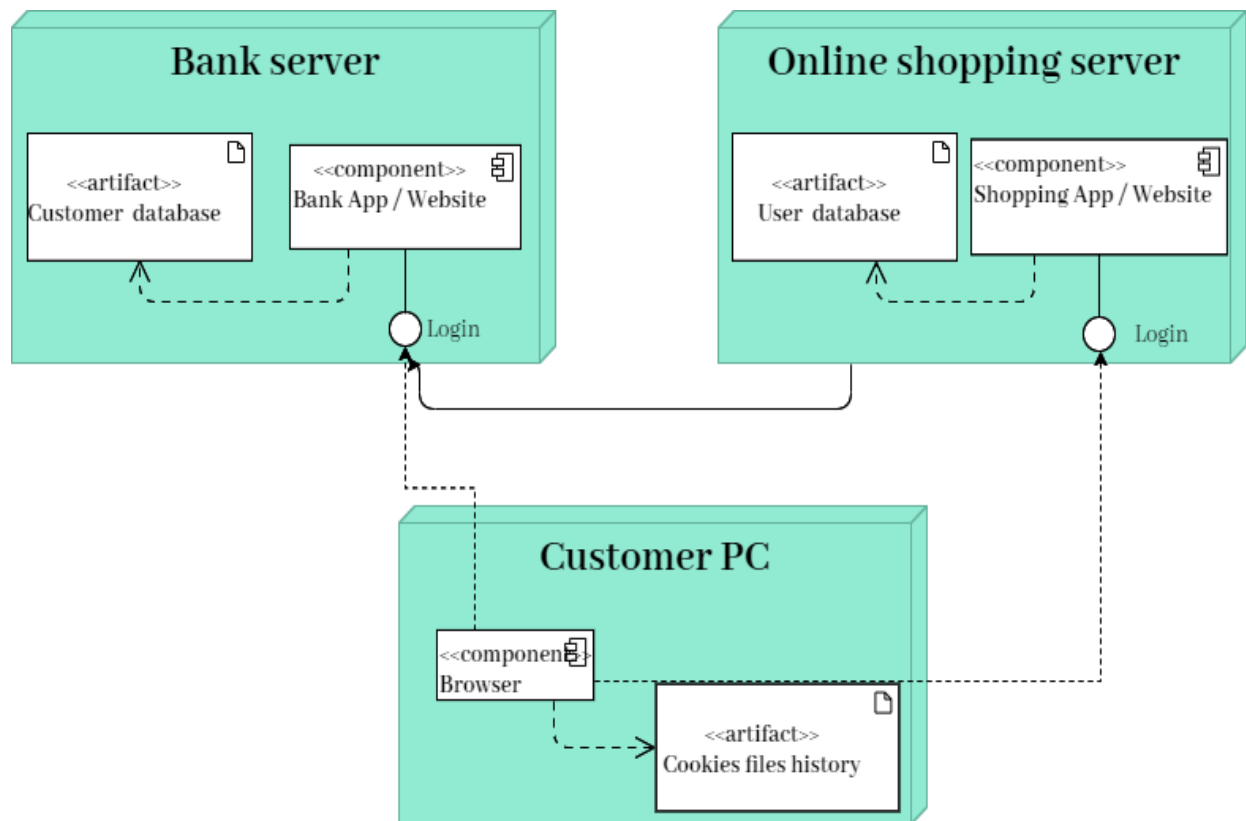


Figure (i) – Deployment Diagram

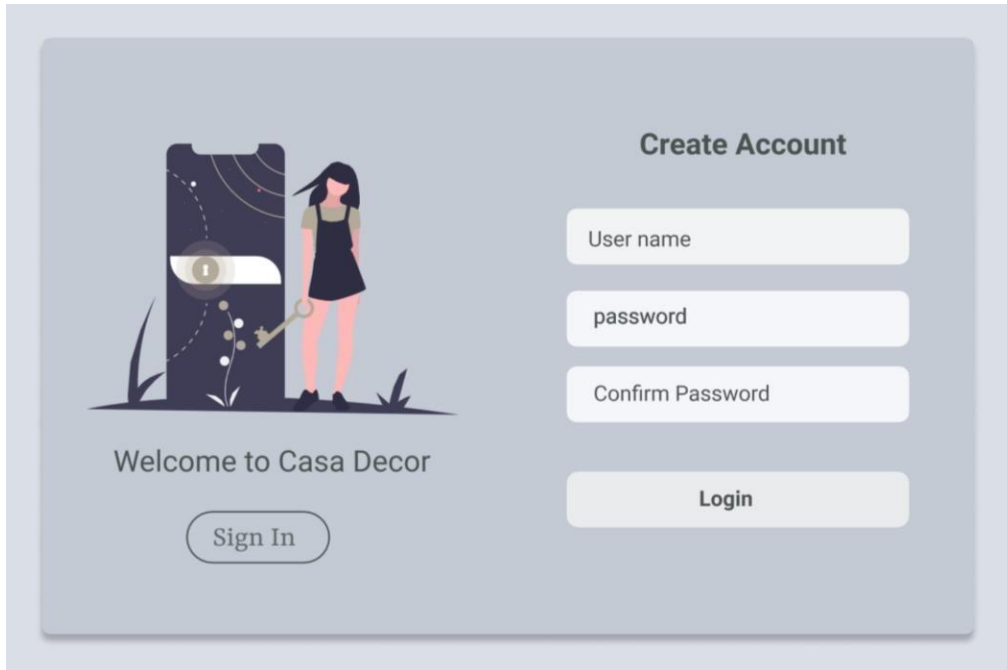


## USER INTERFACE DESIGN USING FIGMA

### 4.5

#### 4.5.1-INPUT DESIGN

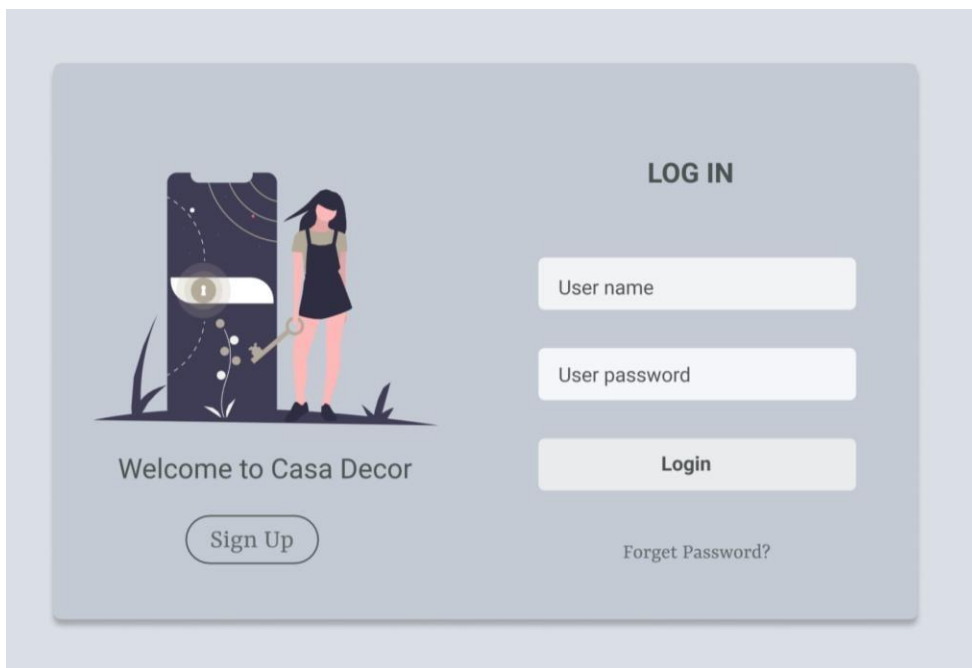
- Form Name : User Registration



The image shows a user interface prototype for account creation. On the left, there is an illustration of a woman in a black dress holding a key, standing next to a large smartphone displaying a login screen. Below the illustration, the text "Welcome to Casa Decor" is displayed, followed by a "Sign In" button. On the right, the heading "Create Account" is centered. Below it, there are three input fields labeled "User name", "password", and "Confirm Password". At the bottom of these fields is a "Login" button.

Figure (j) - Prototype of Account creation page.

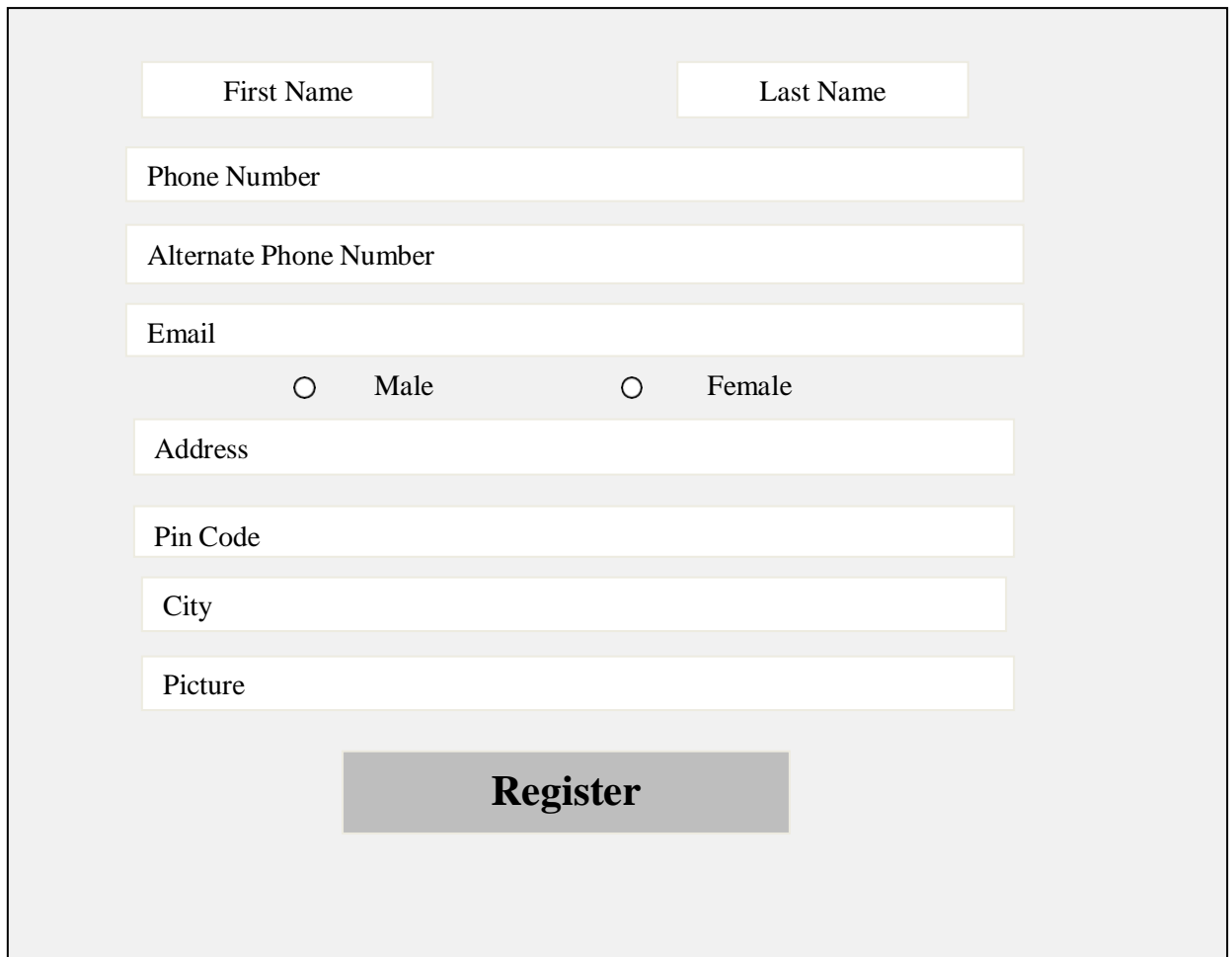
- Form Name : User Login



The image shows a user interface prototype for the login page. On the left, there is an illustration of a woman in a black dress holding a key, standing next to a large smartphone displaying a login screen. Below the illustration, the text "Welcome to Casa Decor" is displayed, followed by a "Sign Up" button. On the right, the heading "LOG IN" is centered. Below it, there are two input fields labeled "User name" and "User password". At the bottom of these fields is a "Login" button. Below the "Login" button is a link that says "Forget Password?".

Figure (k) - Prototype of Login page.

- Form Name : Profile



The image shows a prototype of a profile creation page. It features a light gray background with a white border. The form consists of several input fields and a radio button group. The fields are labeled: First Name, Last Name, Phone Number, Alternate Phone Number, Email, Address, Pin Code, City, and Picture. The radio button group is labeled Male and Female. A large gray button labeled Register is positioned at the bottom center of the form.

First Name	Last Name
Phone Number	
Alternate Phone Number	
Email	
<input type="radio"/> Male	<input type="radio"/> Female
Address	
Pin Code	
City	
Picture	
<b>Register</b>	

Figure (l) - Prototype of Profile creation page.

#### 4.5.2 OUTPUT DESIGN

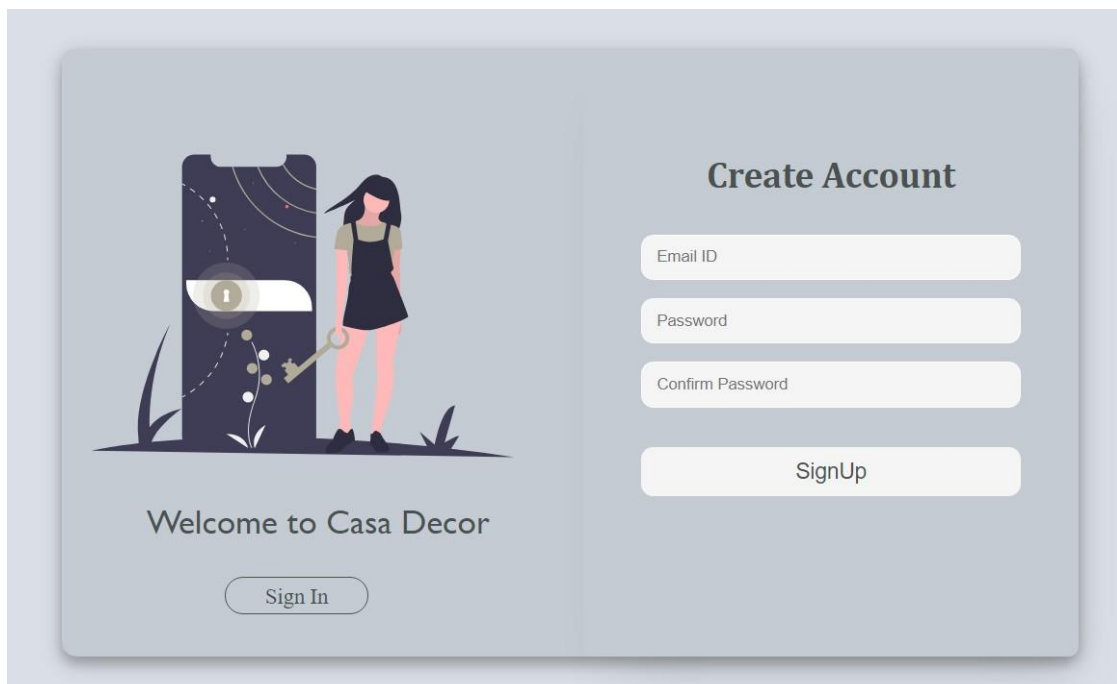
- User Login



The login page features a light gray background with a central white rounded rectangle. On the left side of this rectangle is an illustration of a woman in a black dress holding a key, standing next to a large smartphone displaying a login form. Below the illustration, the text "Welcome to Casa Decor" is displayed, followed by a "Sign Up" button. On the right side, the heading "LOG IN" is centered. Below it are two input fields labeled "Email ID" and "user Password", followed by a "Login" button and a "Forgot Password?" link.

Figure (m) - Login Page, Where users are login.

- User Registration



The registration page features a light gray background with a central white rounded rectangle. On the left side of this rectangle is an illustration of a woman in a black dress holding a key, standing next to a large smartphone displaying a registration form. Below the illustration, the text "Welcome to Casa Decor" is displayed, followed by a "Sign In" button. On the right side, the heading "Create Account" is centered. Below it are three input fields labeled "Email ID", "Password", and "Confirm Password", followed by a "SignUp" button.

Figure (n) - Account creation page, where user can create their account.

- Profile

**Details**

First Name Last Name

Phone Number

Alternate Phone Number

divya@gmail.com

☐ Male ☐ Female

Address

Pin Code

City

Choose File No file chosen

Register

Activate Windows  
Go to Settings to activate Windows.

Figure (o) - Profile creation form, where the user can fill their information in the form and it will show on their own profile.

## 4.6. DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

### 4.6.1 Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a table represents a set of related values.

#### Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of  $n$  elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain  $D$  is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in

interpreting its values.

Every value in a relation is atomic, that is not decomposable.

### **Relationships**

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.

#### **4.6.2 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.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

- ✓ Normalize the data.
- ✓ Choose proper names for the tables and columns.
- ✓ Choose the proper name for the data.

**First Normal Form**

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

**Second Normal Form**

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

**Third Normal Form**

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

### 4.6.3 TABLE DESIGN

#### 1. Table No 01 : tbl\_registration

Primary key: Customer\_id

Foreign Key:

Table 1 - Registration

Column name	Data type	Length	Description
Customer_Id	Int	50	Primary key of Registration table
Email	varchar	25	To store Username
Password	varchar	25	To store password

#### 2. Table No 02: tbl\_Customer\_details

Primary key: Customer\_detail\_id

Foreign Key: Customer\_id

Table 2 – Customer Details

Column name	Data type	Length	Description
Customer_detail_Id	Int	50	Primary key of Customer_details table
Customer_id	Int	25	Foreign Key of registration table
First Name	Varchar	25	To store Customer First Name
Last name	Varchar	25	To store Customer Last Name
Phone Number	Int	25	To store Customer Phone Number
Alternate phone Number	Int	25	To store Customer Alternate Phone Number
Permanent Address	Varchar	25	To store Customer Permanent Address
Pin Code	Int	25	To store pin number
City	Varchar	25	To store the city of customer



**3. Table No 03 : tbl\_Category**

Primary key: Category\_id

Foreign Key:

Table 3 – To add Category of Product

Column name	Data type	Length	Description
Category_Id	Int	50	Primary key of category table
Category_name	Varchar	25	To store Category name
Category_description	Varchar	25	To store Category description

**4. Table No 04 : tbl\_cart**

Primary key: cart\_id

Foreign Key:

product\_Id, customer\_id

Table 4 – To add Subcategory of Product

Column name	Data type	Length	Description
cart_id	Int	50	Primary key of cart table
Product_id	Int	25	Foreign Key of product table
Customer_id	Int	25	Foreign Key of Customer details table
Order_date	Timestamp	25	To store the date when product is ordered
Quantity	Int	25	To store the quantity of product

**5. Table No 05: tbl\_Product**

Primary key: Product\_id

Foreign Key: Category\_Id

Table 5 – To add Product

Column name	Data type	Length	Description
Product_id	Int	50	Primary key of Product table
Category_Id	Int	50	Foreign Key of Category table
Product name	Varchar	25	To store Product name
Product_Price	Int	25	To store Price of the product
Product_quantity	Int	25	To store quantity of the product
Product_Image	Varchar	200	To store the Image of the product
Shop	Varchar	50	To store the ship id
Product_code	Varchar	200	To store the code of the product

#### 6 .Table No 06: tbl\_cart

Primary key: cart\_id

Foreign Key: product\_Id,customer\_id

Table 6 – Cart

Column name	Data type	Length	Description
cart_id	Int	50	Primary key of cart table
Product_id	Int	25	Foreign Key of product table
Customer_id	Int	25	Foreign Key of Customer details table
Order_date	Timestamp	25	To store the date when product is ordered
Quantity	Int	25	To store the quantity of product

#### 7. Table No 07: tbl\_order\_items

Primary key: order\_id

Foreign Key: product\_id,

customer\_id, cart\_id

Table 7 – ordered\_items

Column name	Data type	Length	Description
Order_id	Int	50	Primary key of order table
Product_id	Int	25	Foreign Key of product table
Customer_id	Int	25	Foreign Key of register table
Cart_id	Int	25	Foreign Key of cart table
Product_name	Varchar	25	store the Product name of the product
Quantity	Int	25	To store the Quantity of the product
Amount	Int	25	store the Amount of the product
Tax	Int	25	to store the Tax of the product
Tracking Id	Int	25	store the tracking id of the product
Purchase_date	Varchar	25	To store the Purchased date of the product
Customer_name	Varchar	25	To store the customer name
Phone_number	Varchar	25	to store the phone number of the customer
Address	Varchar	25	store the Address of the customer to deliver the product
Status	Varchar	25	to store the status of the order whether it is purchased or cancelled
Reason	Varchar	200	reason if the order is cancelled.

**8. Table No 08: tbl\_payment**

Primary key: payment\_id

Foreign Key: order\_id

Table 8 – Payment

Column name	Data type	Length	Description
Payment_id	Int	50	Primary key of payment table
order_id	Int	25	Foreign Key of order items table
payment_date	Timestamp	25	To store the date of payment
Payment_type	Varchar	25	To store which mode of payment
Payable_amount	Decimal	25	To store the Total amount to be Payable

**9. Table No 09: tbl\_payment\_type**

Primary key: paymenttype\_id

Foreign Key: payment\_id

Table 9 –.Payment Type

Column name	Data type	Length	Description
Paymenttype_id	Int	50	Primary key of paymenttype table
payment_id	Int	25	Foreign Key of payment table
Type_name	Varchar	25	To store the type of payment

## **CHAPTER 5**

### **SYSTEM TESTING**

## 5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

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:

Testing is a process of executing a program with the intent of finding an error.

- 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 are 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 implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

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

### 5.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling

paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

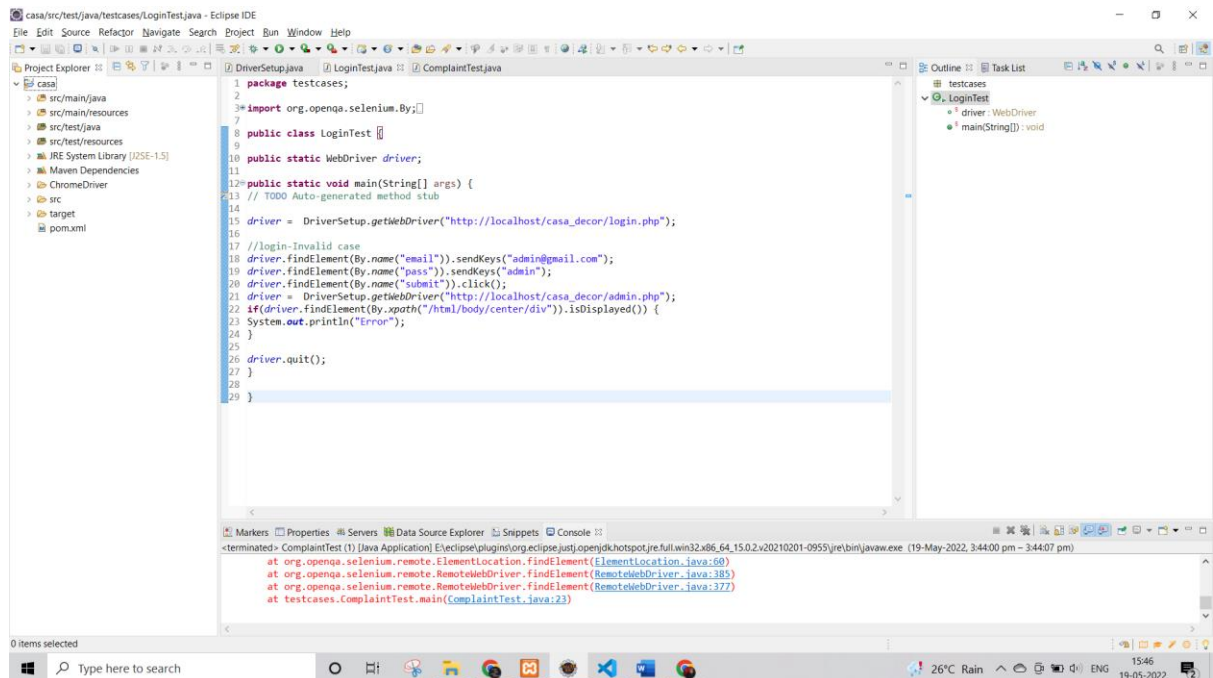
Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code were removed and ensured that all modules are working, and gives the expected result.

### Test Case 1

Project Name : Casa Decor					
Login Test Case					
Test Case ID: Fun_1			Test Designed By: Mekha Ramlal		
Test Priority (Low/Medium/High): High			Test Designed Date: 19-05-2022		
Module Name: Login Screen			Test Executed By: Dr.Bijimol T K		
Test Title: Verify login with valid email and password			Test Execution Date: 19-05-2022		
Description: Test the Login Page					
Pre-Condition: User has valid email id and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login page displayed	Pass
2	Provide Valid User Name	User Name: admin@g mail.com	User should be able to Login	User Logged in and navigated to the dashboard with records	Pass
3	Provide Valid Password	Password: admin			
4	Click on Sign In button				



5	Provide Invalid user name or password	Email Id: test@gmail.com Password: 1234	User shouldnot be able to Login	Message for enter valid user name or password displayed	Pass
6	Provide Null Username Id or Password	Email Id: null Password: null			
7	Click on Sign In button				
Post-Condition: User is validated with database and successfully login into account.The Account session details are logged in database					



### Code

```
package testcases;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class LoginTest {
    public static WebDriver driver;

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}
```

```

driver = DriverSetup.getWebDriver("http://localhost/casa_decor/login.php");
//login-Invalid case
driver.findElement(By.name("email")).sendKeys("admin@gmail.com");
driver.findElement(By.name("pass")).sendKeys("admin");
driver.findElement(By.name("submit")).click();
driver = DriverSetup.getWebDriver("http://localhost/casa_decor/admin.php");
if(driver.findElement(By.xpath("/html/body/center/div")).isDisplayed()) {
System.out.println("Error");
}
driver.quit();
}
}

```

## Test Case 2

<b>Test Priority(Low/Medium/High):</b> High			<b>Test Designed Date:</b> 19-05-2022		
<b>Module Name:</b> Shopping screen(purchasing product)			<b>Test Executed By :</b> Dr.Bijimol T K		
<b>Test Title :</b> Verify login with validemail and password			<b>Test Execution Date:</b> 19-05-2022		
<b>Description:</b> Test the Complaint Registration					
<b>Pre-Condition :</b> Must be valid user					
<b>Step</b>	<b>Test Step</b>	<b>Test Data</b>	<b>Expected Result</b>	<b>Actual Result</b>	<b>Status(Pass/Fail)</b>
1	Navigation toLogin Page		Login Page should be displayed	Login page displayed	Pass
2	Provide Valid username	User Name: athira@gmail.com	User should be able to Login	User Logged in and navigated to index page	Pass
3	Provide Valid Password	Password: athira			
4	Click on Sign In button				
5	Click on add to cart button		User shouldt be able	User successfully add product	Pass

			to add product to cart	to cart and navigated to cart page.	
6	Click on chekout button		User will navigated to checkout_end_page	User successfully navigated to checkout_end_page	Pass
7	Click on confirm button		User will be able to navigate payment page.	User navigated to payment page and done payment.	Pass.
<b>Post-Condition:</b> User is validated with database and successfully login into account. The Account session details are logged in database. And the user can successfully add product to cart, confirm order and finally did payment through payment gateway.					

**Code**

```

package testcases;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import chromedriver.DriverSetup;

public class ComplaintTest {
    public static WebDriver driver;

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        driver = DriverSetup.getWebDriver("http://localhost/casa_decor/login.php");
        //login-Invalid case

        driver.findElement(By.name("email")).sendKeys("athira@gmail.com");
        driver.findElement(By.name("pass")).sendKeys("athira");
        driver.findElement(By.name("submit")).click();

        driver = DriverSetup.getWebDriver("http://localhost/casa_decor/index.php");
        driver.findElement(By.name("btncart")).click();
    }
}

```

```

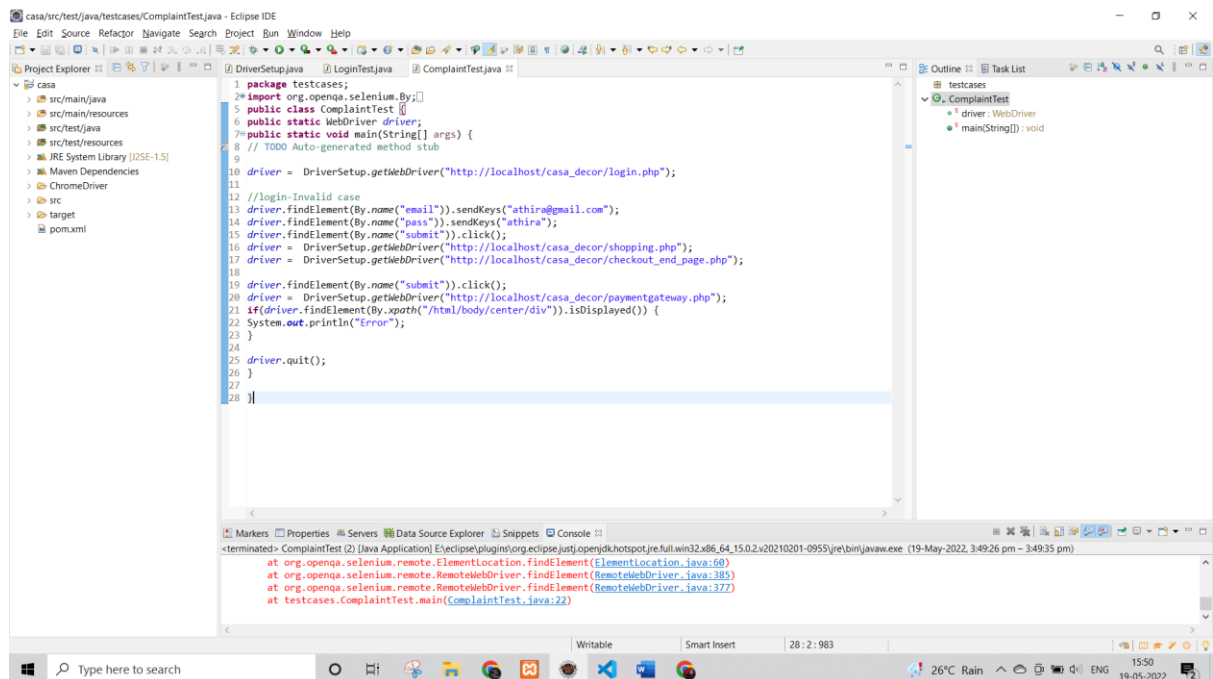
driver = DriverSetup.getWebDriver("http://localhost/casa_decor/shopping.php");
driver.findElement(By.name("checkout")).click();

driver = DriverSetup.getWebDriver("http://localhost/casa_decor/checkout_end_page.php");
driver.findElement(By.name("submit")).click();

driver = DriverSetup.getWebDriver("http://localhost/casa_decor/paymentgateway.php");
if(driver.findElement(By.xpath("/html/body/center/div")).isDisplayed()) {
System.out.println("Error");
}

driver.quit();
}
}

```



### 5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover

---

differences in program structures were removed and a unique program structure was evolved.

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

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

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

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

## **CHAPTER 6**

### **IMPLEMENTATION**

## 6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

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

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system.

In many organizations someone who will not be operating it, will commission the

software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system. □ Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

### **6.2.1 User Training**

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

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

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy

### **6.2.3 System Maintenance**

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the



---

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

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## **7.1 CONCLUSION**

The current system working technology is old fashioned and there is no usage of commonly used technologies like internet, digital money. The proposed system introduces facility for customer to book service online and view all information.

## **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.
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

**WEBSITES:**

- [www.w3schools.com](http://www.w3schools.com)
- <http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf>
- [www.agilemodeling.com/artifacts/useCaseDiagram.html](http://www.agilemodeling.com/artifacts/useCaseDiagram.html)

## **CHAPTER 9**

## **APPENDIX**

## 9.1 Sample Code

### ➤ Login.php

Code for Login user, admin and shop

```
<!DOCTYPE html>
<html>

<head>
  <title>Login Page </title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="css/style.css">
  <style>
    #login_shop {
      text-decoration: none;
      color: blueviolet;
      font-size: large;
      margin-left: 31%;
    }

    #login_shop:hover {
      color: black;
    }
  </style>
</head>
<body id="lg">
  <div class="login-form">
    <div class="main">
      <div class="form-img">
        
        <span class="txt1">Welcome to Casa Decor</span><br>
        <a href="register.php" class="button1" style="vertical-align:middle"><span
          class="span1">Sign Up </span></a>
      </div>
      <div class="content"><br><br>
        <h2>LOG IN</h2>
        <br>
        <form action="#" method="POST">
          <input type="email" name="email" placeholder="Email ID" required>
          <input type="password" name="pass" placeholder="user Password"
            required><br><br>
          <button class="btn" type="submit" name="submit"> Login </button>
        </form>
        <p class="account"> <a href="forget.php">Forget Password?</a></p><br>
        <!--<a href="shop_login.php" id="login_shop">LOGIN AS SHOP</a> <br><br>
        -->
      </div>
    </div>
  </div>
</body>
```

```
</body>

</html>
<?php
session_start();
if (isset($_POST["submit"])) {
    include "db.php";
    $name = $_POST['email'];
    $pass = $_POST['pass'];
    $sql = "SELECT `email`,`pass`,`status` FROM `register` WHERE `email`='$name' AND
`pass`='$pass' AND `status` IN ('shop')";
    $result = mysqli_query($con, $sql);

    $sql_user = "SELECT `email`,`pass`,`status` FROM `register` WHERE `email`='$name'
AND `pass`='$pass' AND `status` IN ('user')";
    $result_user = mysqli_query($con, $sql_user);

    $_SESSION["email"] = $name;

    if ($result->num_rows > 0) {
        if ($result) {
            header("Location: shopindex.php");
        }
    } else if (($name == "admin@gmail.com") && ($pass == "admin")) {
        header("Location: admin.php");
    } else if ($result_user) {
        header("Location: index.php");
    } else {
        echo "<font color=red><h3>Username/password is incorrect.</font></h3>";
    }
}

?>
```



➤ **Register.php**

Code for Register new account

```
<!DOCTYPE html>
<html>

<head>
  <title>Register </title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="css/style.css">
  <script>
    function validate_pass() {
      var password = document.myform.pass.value;
      var pname = document.forms["reg_form"]["pass"];
      if (password.length < 6) {
        var error = "Password must be at least 6 characters long.";
        document.getElementById("pass_err").innerHTML = error;
        pname.focus();
        return false;
      } else {
        document.getElementById("pass_err").innerHTML = "";
        document.myform.cpass.focus();
        return true;
      }
    }
  </script>
</head>

<body id="lg">
  <div class="login-form">
    <div class="main">
      <div class="form-img">
        
        <span class="txt1">Welcome to Casa Decor </span><br>
        <a href="login.php" class="button1" style="vertical-align:middle"><span
class="span1">Sign In </span></a>
      </div>
      <div class="content"><br><br>
      <h2>Create Account</h2>
      <form action="register.php" method="POST" name="reg_form">
        <input type="email" name="email" placeholder="Email ID" required>
        <input type="password" name="pass" placeholder="Password "
onblur="validate_pass()" required>
        <h5 style="color:red" ;>
          <p id="pass_err"></p>
        </h5>
        <input type="password" name="cpass" placeholder="Confirm Password"
required><br><br>
```

```

        <button class="btn" type="submit" name="submit"> SignUp
    </button><br><br><br><br>
</form>

</div>
</div>
</div>
</body>

</html>

<?php
session_start();
include "db.php";
if (isset($_POST["submit"])) {
    $name = $_POST["email"];
    $p = $_POST["pass"];
    $c = $_POST["cpass"];
    $st = "user";
    $query = "SELECT `email` FROM `register` WHERE email = '$name' ";
    $result = mysqli_query($con, $query);
    if (mysqli_num_rows($result) > 0) {
        $_SESSION["email"] = $name;
        echo "<script> alert('Oops Email already exists.. ')</script>";
    } else {
        if ($p == $c) {
            $sql = "INSERT INTO `register`(`email`, `pass`, `cpass`, `status`)
VALUES ('$name','$p','$c','$st')";
            $res = mysqli_query($con, $sql);
            if ($res) {
                echo "<script> alert('Wow! User Registration Completed. ')</script>";
                header("Location: login.php");
            } else {
                echo "<script> alert('OOPS... Something Wrong. ')</script>";
            }
        } else {
            echo "<script> alert('Password not Mached. ')</script>";
        }
    }
}
?>

```

### ➤ Profile.php

Code for creating a profile of user

```

<DOCTYPE html>
<html>
<head>
    <title>index Page </title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="css/style.css">

```

```
</head>

<body id="index-body">
  <div class="profile-container">
    <div class="navbar">
      <ul>
        <li class="list ">
          <a href="index.php">
            <span class="icon">
              <ion-icon name="home-outline"></ion-icon>
            </span>
            <span class="title">Home</span>
          </a>
        </li>
        <li class="list">
          <a href="abot.php">
            <span class="icon">
              <ion-icon name="information-circle-outline"></ion-icon>
            </span>
            <span class="title">About</span>
          </a>
        </li>
        <li class="list">
          <a href="product_gallery.php">
            <span class="icon">
              <ion-icon name="grid-outline"></ion-icon>
            </span>
            <span class="title">Collection</span>
          </a>
        </li>
        <li class="list active">
          <a href="profile.php">
            <span class="icon">
              <ion-icon name="person-outline"></ion-icon>
            </span>
            <span class="title">profile</span>
          </a>
        </li>

        <li class="list">
          <a href="shopping.php">
            <span class="icon">
              <ion-icon name="cart-outline"></ion-icon>
            </span>
            <span class="title">Cart</span>
          </a>
        </li>
        <li class="list">
          <a href="order.php">
            <span class="icon">
```

```

        <ion-icon name="cart-outline"></ion-icon>
    </span>
    <span class="title">Order</span>
</a>
</li>
<li class="list">
    <a href="logout.php">
        <span class="icon">
            <ion-icon name="log-in-outline"></ion-icon>
        </span>
        <span class="title">Logout</span>
    </a>
</li>
</ul>
</div>
<br><br>

<?php
session_start();
if ($_SESSION["email"]) {
    include "db.php";
    $self = $_SESSION["email"];
    echo "<p class=welcome_profile> Welcome " . $_SESSION["email"] . "</p>";
    $check = mysqli_query($con, "SELECT * FROM `user_detail` WHERE `email` =
'$self' ");
    $row = mysqli_fetch_array($check);
    if ($row) {
        echo "<a href=update_user_details.php id=prof>Update profile</a>";
    } else {
        echo "<a href=user_details.php id=prof>complete profile</a>";
    }
}

?>
<div class="toggle">
    <ion-icon name="menu-outline" class="open"> </ion-icon>
    <ion-icon name="close-outline" class="close"></ion-icon>
</div>

</div>

<div class="profile-div">
    <div class="userdetailtable1">
        <?php
        $result = mysqli_query($con, "SELECT * FROM `user_detail` WHERE
`email`='$self'");
        if ($result) {
            while ($row = mysqli_fetch_array($result)) {
                ?>
                <div class="pfl">
                    <div class="pone">

```

---

```

        " class="pf-
img"></img><br>
        <p id="pfname"><?php echo "" ?><?php echo $row["first_name"] . " " .
$row["last_name"]; ?></p><br>
        <p id="pem"><?php echo " " ?><?php echo $row["email"]; ?></p><br>
        <p id="pphn"><?php echo " +91 " ?><?php echo
$row["phone_number"]; ?></p><br>
    </div>
    <div class="ptwo">
        <p id="pgender"><?php echo "Gender : " ?><?php echo $row["gender"];
?></p><br>
        <p id="palphn"><?php echo "Alternate phone Number : " ?><?php echo
$row["alternate_phone_number"]; ?></p><br>
        <p id="padd"><?php echo "Address : " ?><?php echo $row["address"];
?></p><br>
        <p id="ppin"><?php echo "Pin Number : " ?><?php echo
$row["pin_number"]; ?></p><br>
        <p id="pcity"><?php echo "City : " ?><?php echo $row["city"];
?></p><br>
    </div>

</div>

<?php
}
} else {
    echo "<marquee direction = right style=width:50%;margin-left:27%
scrollamount=12><font color=red><h3>Profile not added yet!</font></h3></marquee>";
}
?>

</div>

<script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
<script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>

<script>
    let menuToggle = document.querySelector('.toggle');
    let navbar = document.querySelector('.navbar');
    menuToggle.onclick = function() {
        menuToggle.classList.toggle('active');
        navbar.classList.toggle('active');
    }

    //add active class in selected list item.

```

---

```

    let list = document.querySelectorAll('.list');
    for (let i = 0; i < list.length; i++) {
        list[i].onclick = function() {
            let j = 0;
            while (j < list.length) {
                list[j++].className = 'list';
            }
            list[i].className = 'list active';
        }
    }
}
</script>
<br><br><br><br><br><br><br><br>
</center>

```

```

</body>

```

```

</html>

```

#### ➤ AddProduct.php

Code for Add new Product

```

<DOCTYPE html>
<html>

<head>
    <title>index Page </title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="css/style.css">
</head>

<body>
    <div class="admin_body">
        <div class="admin_nav">
            <ul>
                <li class="list">
                    <a href="admin.php">
                        <span class="icon">
                            <ion-icon name="grid-outline"></ion-icon>
                        </span>
                        <span class="title">Dashboard</span>
                    </a>
                </li>
                <li class="list">
                    <a href="userdetails.php">
                        <span class="icon">
                            <ion-icon name="people-outline"></ion-icon>
                        </span>
                        <span class="title">User Details</span>
                    </a>

```

---

```

        </li>
        <li class="list">
            <a href="#">
                <span class="icon">
                    <ion-icon name="cart-outline"></ion-icon>
                </span>
                <span class="title">Order Details</span>
            </a>
        </li>
        <li class="list active">
            <a href="category_view.php">
                <span class="icon">
                    <ion-icon name="pricetags-outline"></ion-icon>
                </span>
                <span class="title">Product</span>
            </a>
        </li>
        <li class="list">
            <a href="#">
                <span class="icon">
                    <ion-icon name="server-outline"></ion-icon>
                </span>
                <span class="title">Stock</span>
            </a>
        </li>
        <li class="list">
            <a href="logout.php">
                <span class="icon">
                    <ion-icon name="log-in-outline"></ion-icon>
                </span>
                <span class="title">Logout</span>
            </a>
        </li>
    </ul>

</div>
<?php
session_start();
if ($_SESSION["email"]) {
    echo "<p class=welcomoadmin> Welcome " . $_SESSION["email"] . "</p>";

    //echo"<a href=user_details.php class=prof>complete profile</a>";
} else {
    header("location:login.php");
}

?>
<h4 class="overview">ADD Product</h4>
<p id="dash_date"></p>
<script>
    var today = new Date();

```

---

---

```

        var date = today.getDate() + '-' + (today.getMonth() + 1) + '-' +
today.getFullYear();
        document.getElementById("dash_date").innerHTML = date;
    </script>
    <div class="productdiv">

        <form action="product.php" method="POST" enctype="multipart/form-data">
            <select name="pcid" required id="pname">
                <span class="fa fa-user"></span>
                <option>--Select User Type--</option>
                <?php
                    include "db.php";

                    $records = mysqli_query($con, "SELECT * From category");

                    while ($row = mysqli_fetch_array($records)) {
                        echo "<option value='" . $row['category_id'] . "'> " . $row['category_name'] .
    "</option>";
                    }
                ?>
            </select>
            <input type="text" name="pname" placeholder="Product Name" required
id="pname">
            <input type="text" name="pprice" placeholder="Product Price" required
id="pname">
            <input type="text" name="pqu" placeholder="Product Quantity" required
id="pname">
            <input type="text" name="shp" placeholder="Shop" value=<?php echo
$_SESSION["email"]; ?> required id="pname">
            <input type="file" name="img" style="width:428px;">
            <input type="text" name="pc" placeholder="Product Code" required
id="pname">
            <br><button class="prdbtn" type="submit" name="submit"> Add Product
    </button>
        </form>
    </div>
</div>
<script>
    //add active class in selected list item.
    let list = document.querySelectorAll('.list');
    for (let i = 0; i < list.length; i++) {
        list[i].onclick = function() {
            let j = 0;
            while (j < list.length) {
                list[j++].className = 'list';
            }
            list[i].className = 'list active';
        }
    }
</script>

```

---



---

```

        <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
        <script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>

        <?php
include "db.php";
if (isset($_POST["submit"])) {

    $ps = $_POST["pcid"];
    $pn = $_POST["pname"];
    $pp = $_POST["pprice"];
    $pq = $_POST["pqu"];
    $sh = $_POST["shp"];
    $pcc = $_POST["pc"];
    $img = $_FILES["img"]["name"];
    move_uploaded_file($_FILES["img"]["tmp_name"], "product/" .
$_FILES["img"]["name"]);

    $sql = "INSERT INTO `product1`(`cat_id`, `product_name`, `product_price`,
`product_quantity`, `product_image`, `shop`, `product_code`)
VALUES('$ps','$pn','$pp','$pq','$img','$sh','$pcc')";
    echo $ps, $pn, $pp, $pq, $img;
    $res = mysqli_query($con, $sql);
    if ($res) {
        echo "<script> alert('Wow! product Added. ')</script>";
    } else {
        echo "<script> alert('OOPS... Something Wrong. ')</script>";
    }

    header("location:product_view.php");
}
?>

</body>

</html>

```

#### ➤ Shopping.php

Code for add product to the cart

```

<?php
//session_start();
include 'db.php';
//include "side.php";
require_once "ShoppingCart.php";

if (isset($_SESSION['email'])) {
    include("db.php");
    $lmail = $_SESSION['email'];

```

---

---

```

    $sqli = "select email from register where email = '$lmail' and status = 'user'";
    $resulti = mysqli_query($con, $sqli);
    $row = mysqli_fetch_array($resulti);
}

$sql = "select * from register where email = '$lmail' and status = 'user' ";
$resulti = mysqli_query($con, $sql);
$row = mysqli_fetch_array($resulti);
$reg_id = $row['regid'];

// you can your integrate authentication module here to get logged in member

$shoppingCart = new ShoppingCart();
if (!empty($_GET["action"])) {
    switch ($_GET["action"]) {
        case "add":
            if (!empty($_POST["quantity"])) {

                $productResult = $shoppingCart->getProductByCode($_GET["code"]);

                $cartResult = $shoppingCart->getCartItemByProduct($productResult[0]["id"],
                $reg_id);

                if (!empty($cartResult)) {
                    // Update cart item quantity in database
                    $newQuantity = $cartResult[0]["quantity"] + $_POST["quantity"];
                    $shoppingCart->updateCartQuantity($newQuantity, $cartResult[0]["id"]);
                } else {
                    // Add to cart table
                    $shoppingCart->addToCart($productResult[0]["id"], $_POST["quantity"],
                $reg_id);
                }
            }
            break;
        case "remove":
            // Delete single entry from the cart
            $shoppingCart->deleteCartItem($_GET["id"]);
            break;
        case "empty":
            // Empty cart
            $shoppingCart->emptyCart($reg_id);
            break;
    }
}
?>
<!DOCTYPE HTML>

<HEAD>
    <TITLE> CASA DECOR Shopping Cart</TITLE>
    <meta name="viewport" content="width=device-width, initial-scale=1">

```

---

```
<link href="style1.css" type="text/css" rel="stylesheet" />
<script src="jquery-3.2.1.min.js"></script>
<script>
    function increment_quantity(cart_id, price) {
        var inputQuantityElement = $("#input-quantity-" + cart_id);
        var newQuantity = parseInt($(inputQuantityElement).val()) + 1;
        var newPrice = newQuantity * price;
        save_to_db(cart_id, newQuantity, newPrice);
    }

    function decrement_quantity(cart_id, price) {
        var inputQuantityElement = $("#input-quantity-" + cart_id);
        if ($(inputQuantityElement).val() > 1) {
            var newQuantity = parseInt($(inputQuantityElement).val()) - 1;
            var newPrice = newQuantity * price;
            save_to_db(cart_id, newQuantity, newPrice);
        }
    }

    function save_to_db(cart_id, new_quantity, newPrice) {
        var inputQuantityElement = $("#input-quantity-" + cart_id);
        var priceElement = $("#cart-price-" + cart_id);
        $.ajax({
            url: "update_cart_quantity.php",
            data: "cart_id=" + cart_id + "&new_quantity=" + new_quantity,
            type: 'post',
            success: function(response) {
                $(inputQuantityElement).val(new_quantity);
                $(priceElement).text("$" + newPrice);
                var totalQuantity = 0;
                $("input[id*='input-quantity-']").each(function() {
                    var cart_quantity = $(this).val();
                    totalQuantity = parseInt(totalQuantity) + parseInt(cart_quantity);
                });
                $("#total-quantity").text(totalQuantity);
                var totalItemPrice = 0;
                $("div[id*='cart-price-']").each(function() {
                    var cart_price = $(this).text().replace("$", "");
                    totalItemPrice = parseInt(totalItemPrice) + parseInt(cart_price);
                });
                $("#total-price").text(totalItemPrice);
            }
        });
    }
</script>
<link rel="stylesheet" type="text/css" href="css/style.css">
</HEAD>

<body id="indx-bdy">
```

```
<div class="navbar">
  <ul>
    <li class="list ">
      <a href="index.php">
        <span class="icon">
          <ion-icon name="home-outline"></ion-icon>
        </span>
        <span class="title">Home</span>
      </a>
    </li>
    <li class="list">
      <a href="about.php">
        <span class="icon">
          <ion-icon name="information-circle-outline"></ion-icon>
        </span>
        <span class="title">About</span>
      </a>
    </li>
    <li class="list ">
      <a href="product_gallery.php">
        <span class="icon">
          <ion-icon name="grid-outline"></ion-icon>
        </span>
        <span class="title">Collection</span>
      </a>
    </li>
    <li class="list">
      <a href="profile.php">
        <span class="icon">
          <ion-icon name="person-outline"></ion-icon>
        </span>
        <span class="title">profile</span>
      </a>
    </li>
    <li class="list active">
      <a href="shopping.php">
        <span class="icon">
          <ion-icon name="cart-outline"></ion-icon>
        </span>
        <span class="title">Cart</span>
      </a>
    </li>
    <li class="list">
      <a href="order.php">
        <span class="icon">
          <ion-icon name="person-outline"></ion-icon>
        </span>
        <span class="title">Order</span>
      </a>
    </li>
  </ul>
</div>
```

```

        <li class="list">
            <a href="logout.php">
                <span class="icon">
                    <ion-icon name="log-in-outline"></ion-icon>
                </span>
                <span class="title">Logout</span>
            </a>
        </li>
    </ul>
</div>
<br><br>

<div class="toggle">
    <ion-icon name="menu-outline" class="open"> </ion-icon>
    <ion-icon name="close-outline" class="close"></ion-icon>
</div>
<br><br><br><br><br><br>
<script>
    let menuToggle = document.querySelector('.toggle');
    let navbar = document.querySelector('.navbar');
    menuToggle.onclick = function() {
        menuToggle.classList.toggle('active');
        navbar.classList.toggle('active');
    }

    //add active class in selected list item.
    let list = document.querySelectorAll('.list');
    for (let i = 0; i < list.length; i++) {
        list[i].onclick = function() {
            let j = 0;
            while (j < list.length) {
                list[j++].className = 'list';
            }
            list[i].className = 'list active';
        }
    }
</script>
<script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
<script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>

<?php
$cartItem = $shoppingCart->getMemberCartItem($reg_id);
if (!empty($cartItem)) {
    $item_quantity = 0;
    $item_price = 0;
    $count = 0;
    if (!empty($cartItem)) {
        foreach ($cartItem as $item) {
            $item_quantity = $item_quantity + $item["quantity"];

```

```

        $item_price = $item_price + ($item["product_price"] * $item["quantity"]);
        $count += count((array)$item['product_price']) * 10;
    }

    ?>
    <?php
    $_SESSION['tp'] = $item_price;
    $_SESSION['tp2'] = $count;
    //echo $_SESSION['tp'];
    ?>

    <div class="txt-heading-label">Shopping Cart</div>

    <div id="shopping-cart">
        <div class="txt-heading">

            <a id="btnEmpty" href="shopping.php?action=empty" style="margin-
            left:200px;margin-top:20px;"></a>
            <div class="cart-status">
                <div>Total Quantity: <span id="total-quantity"><?php echo $item_quantity;
            ?></span></div>
                <div>Total Price: <span id="total-price"><?php echo $item_price;
            ?></span></div>

            </div>
        </div>
    <?php
    }
}

if (!empty($cartItem)) {
    ?>
    <div class="shopping-cart-table">
        <div class="cart-item-container header">
            <div class="cart-info title" style="margin-left: 2%;">Product</div>
            <div class="cart-info" style="margin-left: -20%;">Quantity</div>
            <div class="cart-info price" style="margin-left: 2%;">Unit Price</div>
            <div class="cart-info tprice" style="margin-left:-20px; margin-top:-3%;">Total
            Price</div>
        </div>
        <?php
        foreach ($cartItem as $item) {
            ?>
            <div class="cart-item-container">

                <div class="cart-info title">
                    <td>"
                    class="cart-item-image" />
                    <?php echo $item["product_name"]; ?>
                </div>

```

```

        <div class="cart-info quantity" style="margin-left: -16%;">
            <div class="btn-increment-decrement"
onClick="decrement_quantity(<?php echo $item["cart_id"]; ?>, '<?php echo
$item["product_price"]; ?>')"></div><input class="input-quantity" id="input-quantity-
<?php echo $item["cart_id"]; ?>" value="<?php echo $item["quantity"]; ?>">
            <div class="btn-increment-decrement" onClick="increment_quantity(<?php
echo $item["cart_id"]; ?>, '<?php echo $item["product_price"]; ?>')">+</div>
        </div>

        <div class="cart-info price" style="margin-left: 15%;"> <?php echo
$item["product_price"]; ?>

    </div>

    <div class="cart-info tprice" style="margin-left: -20px;">
        <?php echo ($item["product_price"] * $item["quantity"]); ?>
    </div>

    <div class="cart-info action">
        <a href="shopping.php?action=remove&id=<?php echo $item["cart_id"];
?>" class="btnRemoveAction">
            </a>
        </div>
    </div> <br>

    <?php
    }
    ?>
    <div><a href="product_gallery.php" class="btn btn-block btn-light"
style="margin-left: 1050px"><i class="fa-fas shopping-cart"></i> Continue Shopping </a>
    &nbsp;

        <a type="button" class="btn btn-primary" href="checkout_end_page.php">
Checkout </a>
    </div>
</div>
<?php
}
?>
</div>
<?php //require_once "product-list.php";
?>

<?php

?>

</BODY>

```

</HTML>

➤ **paymentgateway.php**

Code for add product to the cart

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width,  
initial-scale=1.0" />

<style>

```
.row {  
    display: -ms-flexbox;  
    /* IE10 */  
    display: flex;  
    -ms-flex-wrap: wrap;  
    /* IE10 */  
    flex-wrap: wrap;  
    margin-left: 25%;  
    width: 50%;  
    margin-top: -1%;  
}
```

```
.col-25 {  
    -ms-flex: 25%;  
    /* IE10 */  
    flex: 25%;  
}
```

```
.col-50 {  
    -ms-flex: 50%;  
    /* IE10 */  
    flex: 50%;  
}
```

```
.col-75 {  
    -ms-flex: 75%;  
    /* IE10 */  
    flex: 75%;  
}
```

```
.col-25,  
.col-50,  
.col-75 {  
    padding: 0 16px;  
}
```



```
.container {  
    background-color: #f2f2f2;  
    padding: 5px 20px 15px 20px;  
    border: 1px solid lightgrey;  
    border-radius: 3px;  
}  
  
input[type=text] {  
    width: 100%;  
    margin-bottom: 20px;  
    padding: 12px;  
    border: 1px solid #ccc;  
    border-radius: 3px;  
}  
  
label {  
    margin-bottom: 10px;  
    display: block;  
}  
  
.icon-container {  
    margin-bottom: 20px;  
    padding: 7px 0;  
    font-size: 24px;  
}  
  
.btn {  
    background-color: #04AA6D;  
    color: white;  
    padding: 12px;  
    margin: 10px 0;  
    border: none;  
    width: 100%;  
    border-radius: 3px;  
    cursor: pointer;  
    font-size: 17px;  
}  
  
.btn:hover {  
    background-color: #45a049;  
}  
  
span.price {  
    float: right;  
    color: grey;  
}  
  
.buttonpur {  
    display: inline-block;  
    border-radius: 20px;  
    border: 1px solid #4B5251;
```

```
color: #4B5251;
text-align: center;
font-size: 18px;
padding: 5px;
width: 10%;
height: 4.5%;
transition: all 0.5s;
cursor: pointer;
margin-left: 10%;
margin-top: 0.5%;
```

```
}
```

```
.spanpur {
  cursor: pointer;
  display: inline-block;
  position: relative;
  color: #4B5251;
  transition: 0.5s;
}
```

```
.buttonpur .spanpur:after {
  content: '\00bb';
  position: absolute;
  opacity: 0;
  top: 0;
  right: -20px;
  transition: 0.5s;
}
```

```
.buttonpur:hover .spanpur {
  padding-right: 20px;
}
```

```
.buttonpur:hover .spanpur:after {
  opacity: 1;
  right: 0;
}
```

/\* Responsive layout - when the screen is less than 800px wide, make the two columns stack on top of each other instead of next to each other (and change the direction - make the "cart" column go on top) \*/

```
@media (max-width: 800px) {
  .row {
    flex-direction: column-reverse;
  }

  .col-25 {
    margin-bottom: 20px;
  }
}
```

```

</style>
</head>

<body>
  <a href="cart1.php" class="buttonpur" style="vertical-align:middle"><span
class="spanpur">Back to cart </span></a>
  <div class="row">
    <div class="col-75">
      <div class="container">
        <form action="/action_page.php">

          <div class="row">
            <div class="col-50">
              <h3>Billing Address</h3>
              <label for="fname"><i class="fa fa-user"></i> Full Name</label>
              <input type="text" id="fname" name="firstname">
              <label for="email"><i class="fa fa-envelope"></i> Email</label>
              <input type="text" id="email" name="email">
              <label for="adr"><i class="fa fa-address-card-o"></i> Address</label>
              <input type="text" id="adr" name="address">
              <label for="city"><i class="fa fa-institution"></i> City</label>
              <input type="text" id="city" name="city">

            </div>
            <div class="row">
              <div class="col-50">
                <label for="state">State</label>
                <input type="text" id="state" name="state">
              </div>
              <div class="col-50">
                <label for="zip">Zip</label>
                <input type="text" id="zip" name="zip">
              </div>
            </div>
          </div>

          <div class="col-50">
            <h3>Payment</h3>
            <label for="fname">Accepted Cards</label>
            <div class="icon-container">
              <i class="fa fa-cc-visa" style="color:navy;"></i>
              <i class="fa fa-cc-amex" style="color:blue;"></i>
              <i class="fa fa-cc-mastercard" style="color:red;"></i>
              <i class="fa fa-cc-discover" style="color:orange;"></i>
            </div>
            <label for="cname">Name on Card</label>
            <input type="text" id="cname" name="cardname">
            <label for="ccnum">Credit card number</label>
            <input type="text" id="ccnum" name="cardnumber">
            <label for="expmonth">Exp Month</label>
            <input type="text" id="expmonth" name="expmonth">

```

```
<div class="row">
  <div class="col-50">
    <label for="expyear">Exp Year</label>
    <input type="text" id="expyear" name="expyear">
  </div>
  <div class="col-50">
    <label for="cvv">CVV</label>
    <input type="text" id="cvv" name="cvv">
  </div>
</div>
<div>
  <label>
    <input type="checkbox" checked="checked" name="sameadr"> Shipping address
    same as billing
  </label>
  <input type="submit" value="Continue to checkout" class="btn">
</form>
</div>
</div>
</body>

</html>
```

## 9.2 Screen Shots

### Home page

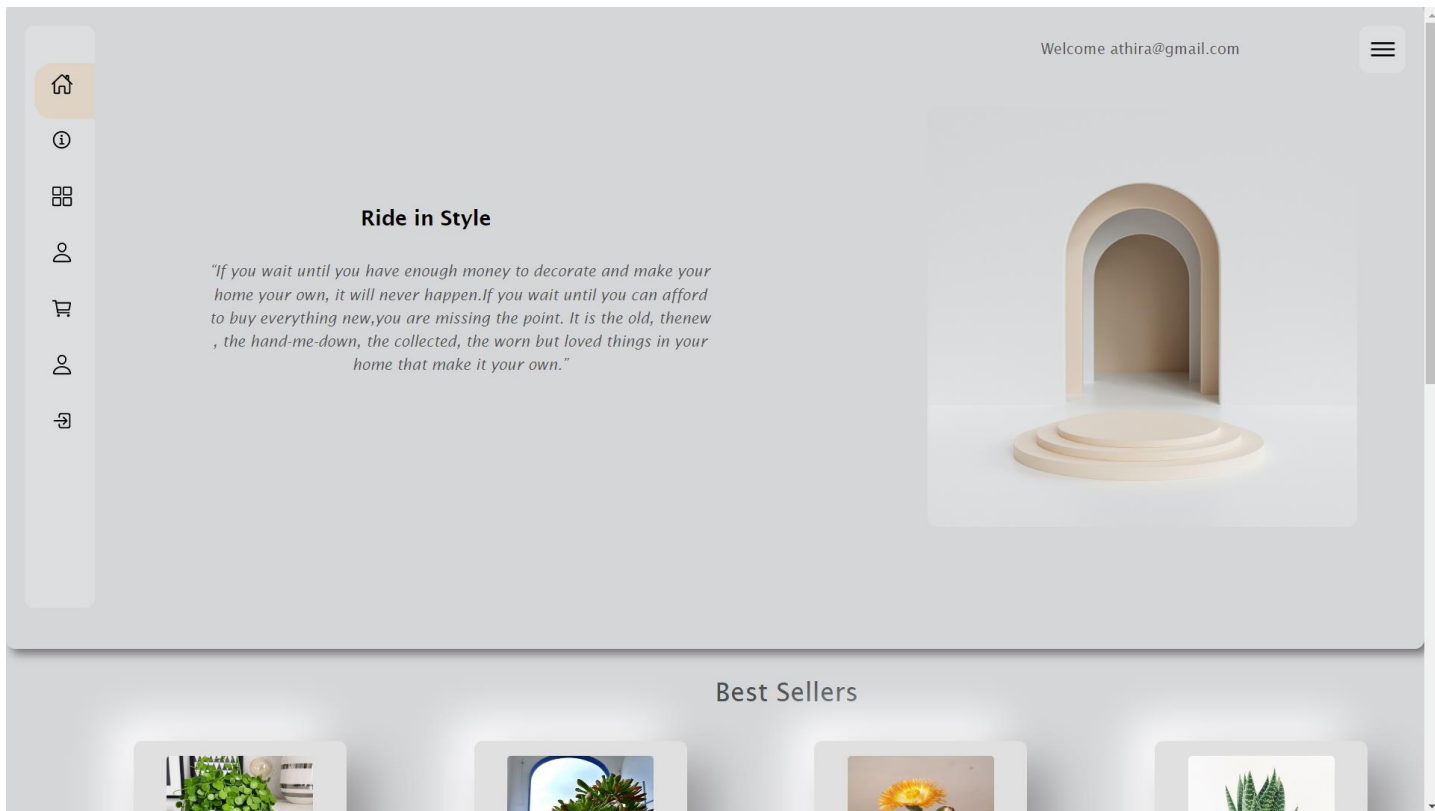


Figure (p) - Home page of Casa Decor

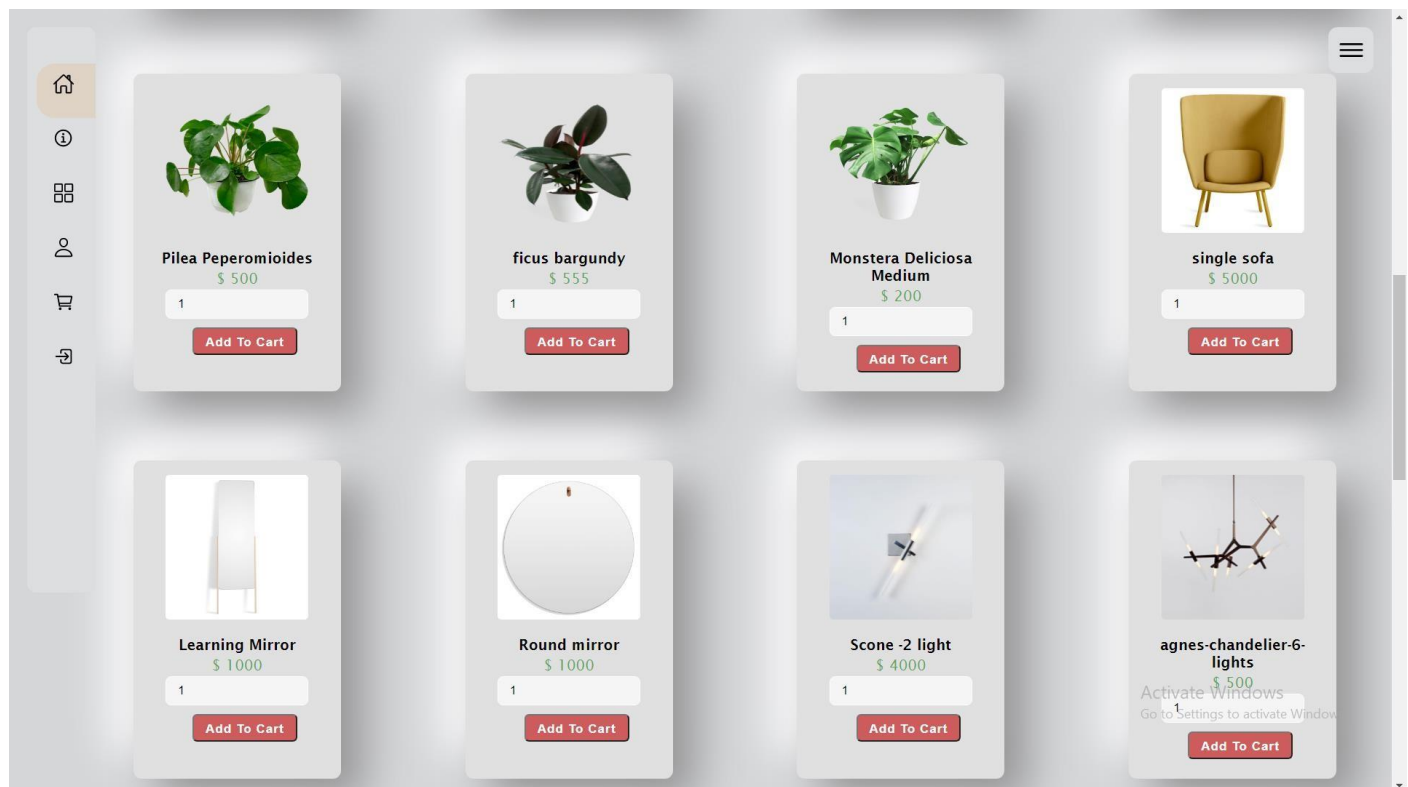


Figure (q) - Home page of Casa Decor

## Product Gallery

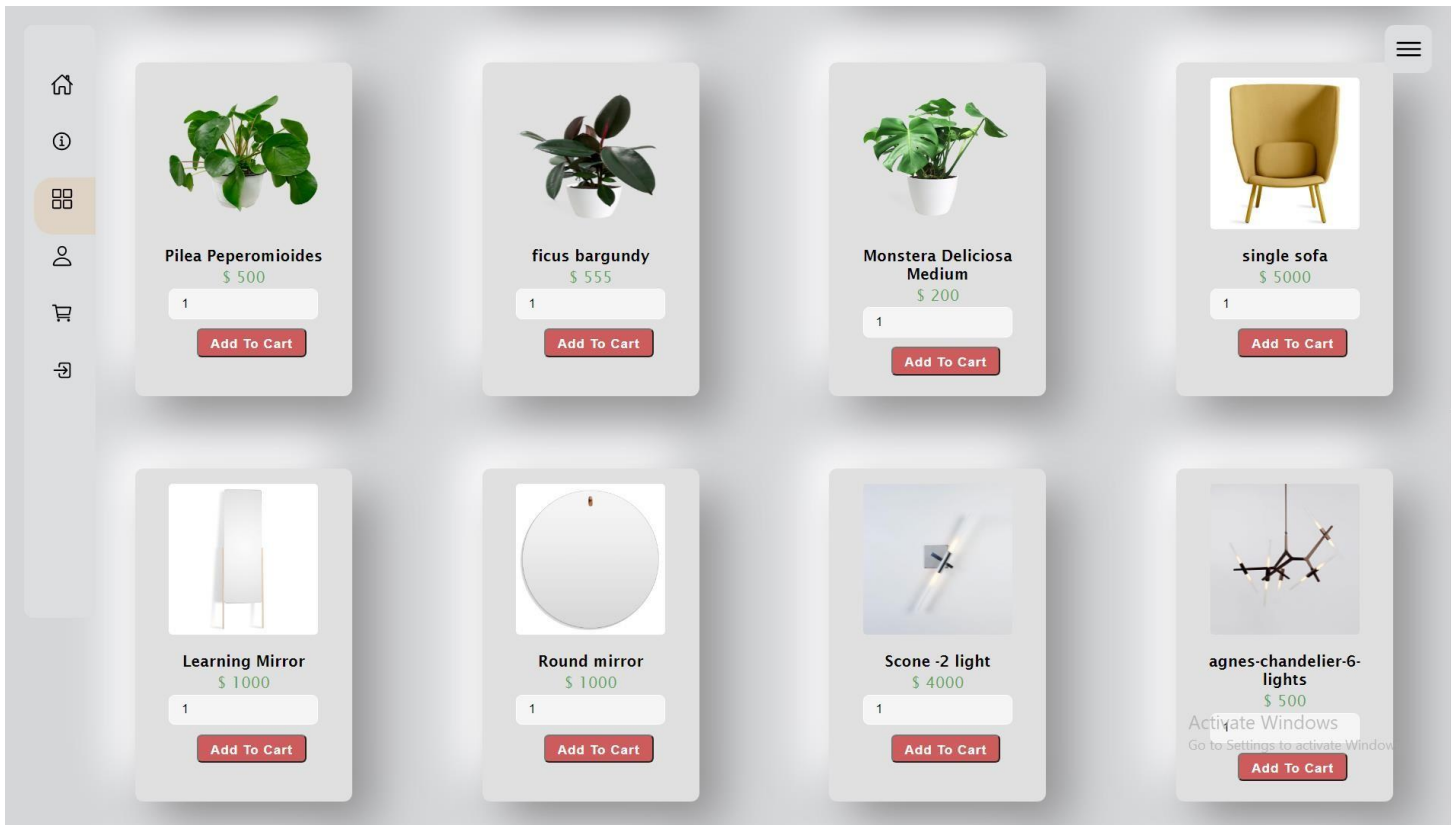


Figure (r) - Product Gallery of Casa Décor.

## User Profile

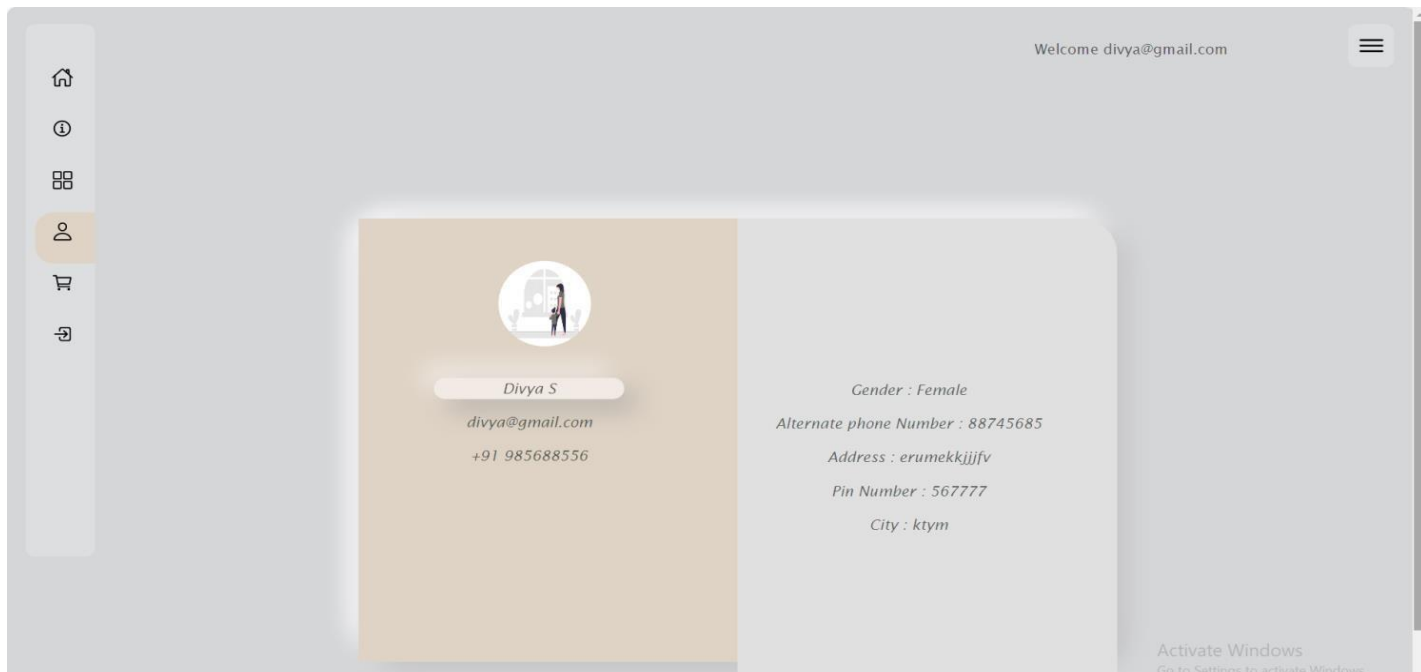


Figure (s) - Customer profile page of Casa Décor.

## Cart

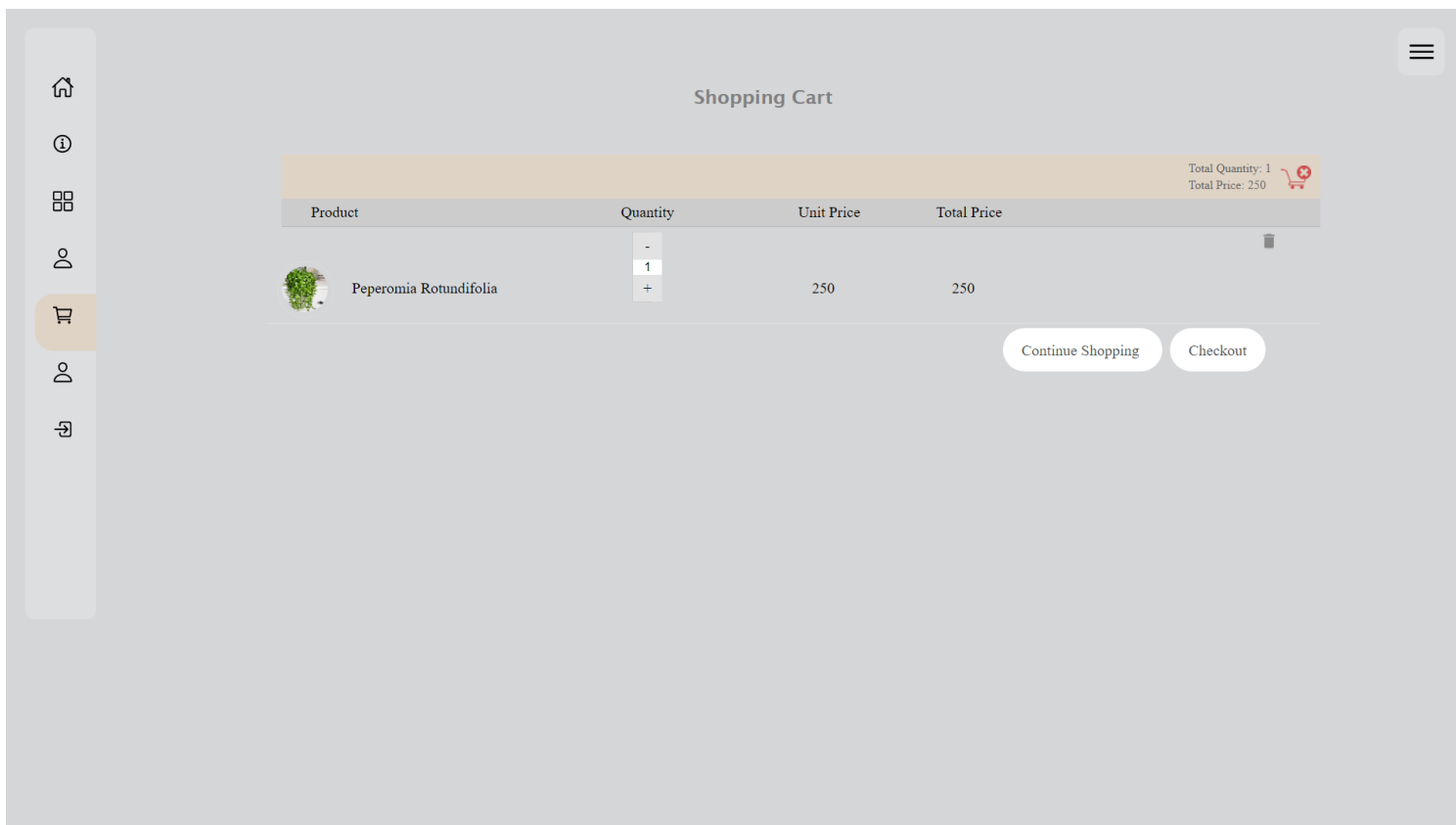


Figure (s) - Cart page of Casa Décor.

## Checkout confirmation

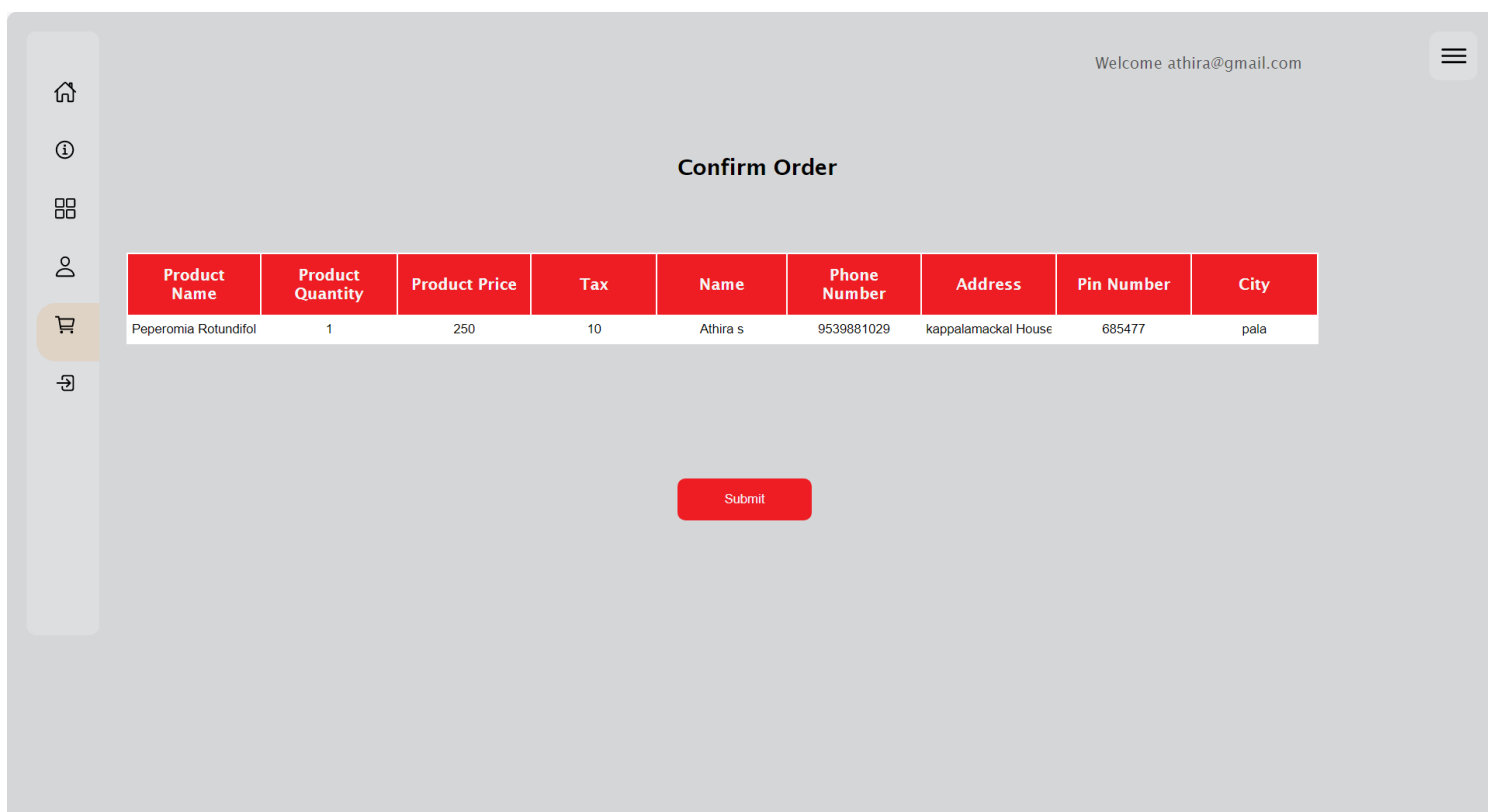


Figure (t) – Checkout confirmation page of Casa Décor.

## Admin Dashboard



Figure (u) - Admin Dashboard of Casa Decor

## Category



Figure (v) - Product Category page of Casa Décor.



## Add Category

Welcome admin@gmail.com

**ADD Category**  
19-5-2022

View category

Category Name

Category description

Add Category

Dashboard

User Details

Order Details

Category

Product

Generate Report

Logout

Figure (w) - where admin can add new category of product to Casa Décor.

## Product view in admin

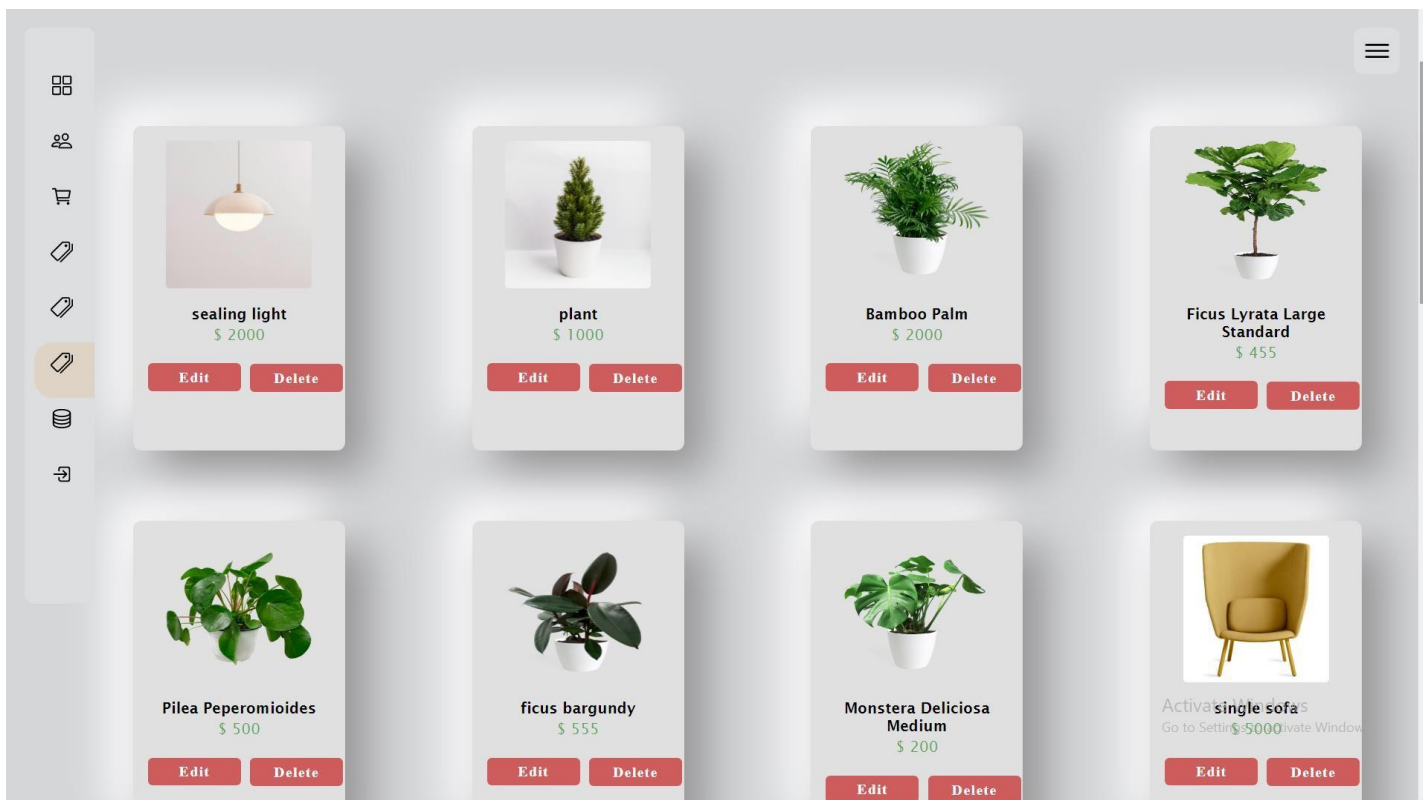


Figure (x) - Product View page of Casa Décor, where the admin can edit and delete product.

## Shop

Welcome admin@gmail.com

**Shop**  
19-5-2022

Add Shop

SHOP ID	SHOP USER NAME	SHOP PASSWORD	SEND LOGIN DETAILS
1	canva@gmail.com	canva	<input type="button" value="canva@gmail.com"/>
2	timenew@gmail.com	time	<input type="button" value="timenew@gmail.com"/>
22	plant@gmail.com	plant	<input type="button" value="plant@gmail.com"/>
23	niya@gmail.com	niya	<input type="button" value="niya@gmail.com"/>
27	mirrorworld@gmail.com	mirror	<input type="button" value="mirrorworld@gmail.com"/>
30	shop@gmail.com	shop1	<input type="button" value="shop@gmail.com"/>


Dashboard  
User Details  
Order Details  
Category  
product  
Order Details  
Shop  
Logout

Figure (y) – Shop added by admin.

## Order details

Welcome admin@gmail.com

**Order Details**  
19-5-2022

User Name	Product Image	Product Name	Price	Quantity	Amount	Address	Shop	Status
Athira s		Alove	500	1	500	kappalamackal House nala	plant@gmail.com	purchased

Dashboard  
User Details  
Order Details  
Category  
Generate Report  
Shop  
Logout

Figure (z1) – order details.

Welcome plant@gmail.com

**Sales**  
19-5-2022



Product Image	Product Name	Price	Quantity	Amount	Purchased User	Status
	pearls plant	500	1	500	athira@gmail.com	canceled
	Alove	500	1	500	athira@gmail.com	purchased

Figure (z2) – order details according to shop.