

ORGANI-SHOP

Project Report Submitted By

PARVATHY R

Reg. No: AJC19MCA015

In Partial fulfilment for the Award of the Degree Of

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



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPALLY**

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

2019-2022

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**ORGANI-SHOP**” is the bonafide work of **PARVATHY R (Reg No: AJC19MCA015)** in partial fulfilment of the requirements for the award of the Degree of Regular Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2019-22.

Ms. Grace Joseph
Internal Guide

Mr. Binumon Joseph
Coordinator

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

External Examiner

DECLARATION

I hereby declare that the project report “**ORGANI SHOP**” is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2019-2022.

Date:

KANJIRAPALLY

PARVATHY R

Reg. No: AJC19MCA015

ACKNOWLEDGEMENT

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

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinators **Rev. Fr. Dr. Rubin Thottupurathu Jose** and **Mr. Binumon Joseph** for their 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, **Ms. Grace Joseph** 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.

PARVATHY R

ABSTRACT

Organi Shop is a web application deals with organic vegetables delivery. The purpose of Organic Vegetable Shop is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirement, so that their valuable data or information can be stored for a longer period with easy accessing and manipulation of the same. The necessary software and hardware are readily available and simple to use. As previously said, an online vegetable shop can lead to an error-free, secure, dependable, and rapid management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. As a result, the company will be able to make better use of its resources. That implies you won't be side-tracked by irrelevant information while still getting to the information you need. The aim is to automate its existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data or information can be stored for a longer period with easy accessing and manipulation of the same. The system is a web-based application that helps to provide their clients an online platform to explore their services and order products they wanted. The system is easy to use and has a pleasant user interface.

CONTENT

Sl. No	Topic	Page No
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	3
2.1	INTRODUCTION	4
2.2	EXISTING SYSTEM	5
2.3	PROPOSED SYSTEM	5
2.4	ADVANTAGES OF PROPOSED SYSTEM	6
3	REQUIREMENT ANALYSIS	7
3.1	FEASIBILITY STUDY	8
3.1.1	ECONOMICAL FEASIBILITY	8
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	9
3.2	SYSTEM SPECIFICATION	10
3.2.1	HARDWARE SPECIFICATION	10
3.2.2	SOFTWARE SPECIFICATION	10
3.3	SOFTWARE DESCRIPTION	10
3.3.1	LARAVEL	10
3.3.2	MYSQL	11
4	SYSTEM DESIGN	13
4.1	INTRODUCTION	14
4.2	UML DIAGRAM	14
4.2.1	USE CASE DIAGRAM	15
4.2.2	SEQUENCE DIAGRAM	18
4.2.3	COLLABORATION DIAGRAM	20
4.2.4	STATE CHART DIADRAM	21

4.2.5	ACTIVITY DIAGRAM	21
4.2.6	OBJECT DIAGRAM	22
4.2.7	DEPLOYMENT DIAGRAM	23
4.3	USER INTERFACE DESIGN	25
4.4	DATA BASE DESIGN	27
5	SYSTEM TESTING	33
5.1	INTRODUCTION	34
5.2	TEST PLAN	35
5.2.1	UNIT TESTING	35
5.2.2	INTEGRATION TESTING	39
5.2.3	VALIDATION TESTING	39
5.2.4	USER ACCEPTANCE TESTING	40
6	IMPLEMENTATION	41
6.1	INTRODUCTION	42
6.2	IMPLEMENTATION PROCEDURE	42
6.2.1	USER TRAINING	43
6.2.2	TRAINING ON APPLICATION SOFTWARE	43
6.2.3	SYSTEM MAINTENANCE	43
7	CONCLUSION & FUTURE SCOPE	44
7.1	CONCLUSION	45
7.2	FUTURE SCOPE	45
8	BIBLIOGRAPHY	46
9	APPENDIX	48
9.1	SAMPLE CODE	49
9.2	SCREENSHOT	71

List of Abbreviations

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

“**Organi Shop**” is a web application which is meant to help the customers to make their vegetables fresh and organic. Customer gets the freedom to buy organic and hygienic fruits and vegetables at their doorsteps as it becomes time saving for customers. The main objective of the project is to manage the details of vegetables, stocks, order, customer, payment. It manages all the information about vegetables, bill, payment. The project is totally built on administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the vegetables, stocks, bill, order. It tracks all the details about the customer, order, payment.

1.2 PROJECT SPECIFICATION

The system includes 3 modules. They are:

1. Admin Module

Admin must have a login into this system. Admin can handle all over the system like edit, update and delete the data.

2. Customer Module

Customer can register and they can order for vegetables and do payment. Customers can add feedbacks and complaints to them.

3. Visitor Module

Visitor can see all the basic pages.

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

The Online Shopping System which is available requires a customer to visit the shop manual and to choose the items from the shop. As the system requires a manual visit of a person, it consumes time and is not a user-friendly process. The customer who went shopping must know the details of the product which he/she want to buy. Otherwise, it will be difficult to identify the product as there are various similar looking products available. The shop will be out of reach from the distant customers and hence loose the customers.

2.3 PROPOSED SYSTEM

The Online Shopping System will manage the items on the web and provide products to the customer online without a physically appearance of that customer. The system will contain all the items in one place, so users need not to go the different shop. As the Online Shopping system is web based so the distant user from the shop can also get the item and the vendor can cover the wide place to sell. A user can choose from the different products, can make online payments and will get the products at the doorstep. This will reduce the time consumption in the purchase of different of different items, make the system efficient and user-friendly.

2.4 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.

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 system has been developed using Laravel in front end and MySQL in server in back end, the project is technically feasible for development. The system has been developed using Laravel 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 - Laravel

Backend - MySQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, J Query, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Laravel

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like Code Igniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development. If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks. Laravel is a powerful MVC PHP framework, designed for developers who need a simple and elegant toolkit to create full-featured web applications. Laravel was created by Taylor Otwell. This is a brief tutorial that explains the basics of Laravel framework.

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
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

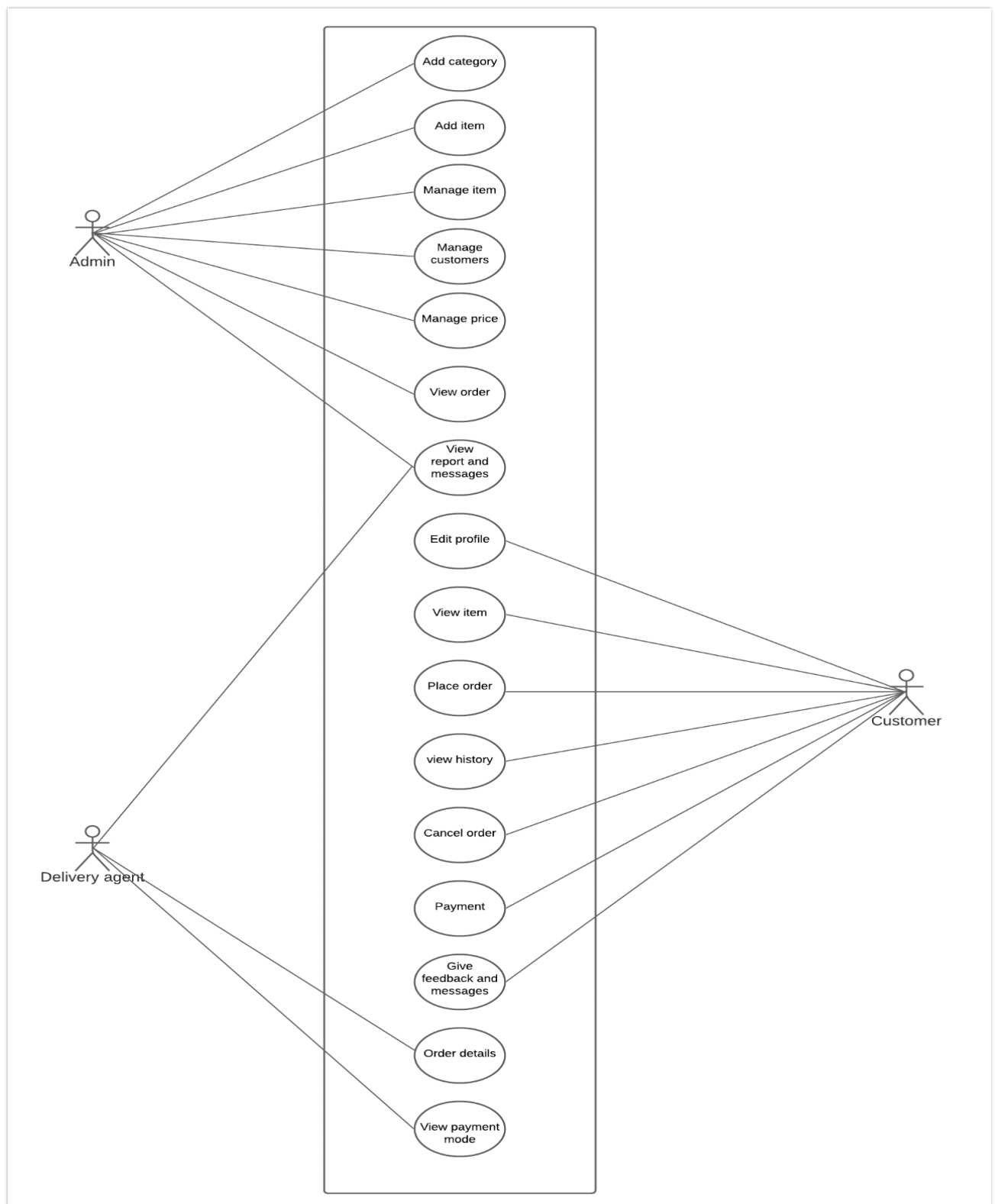
A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order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



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.

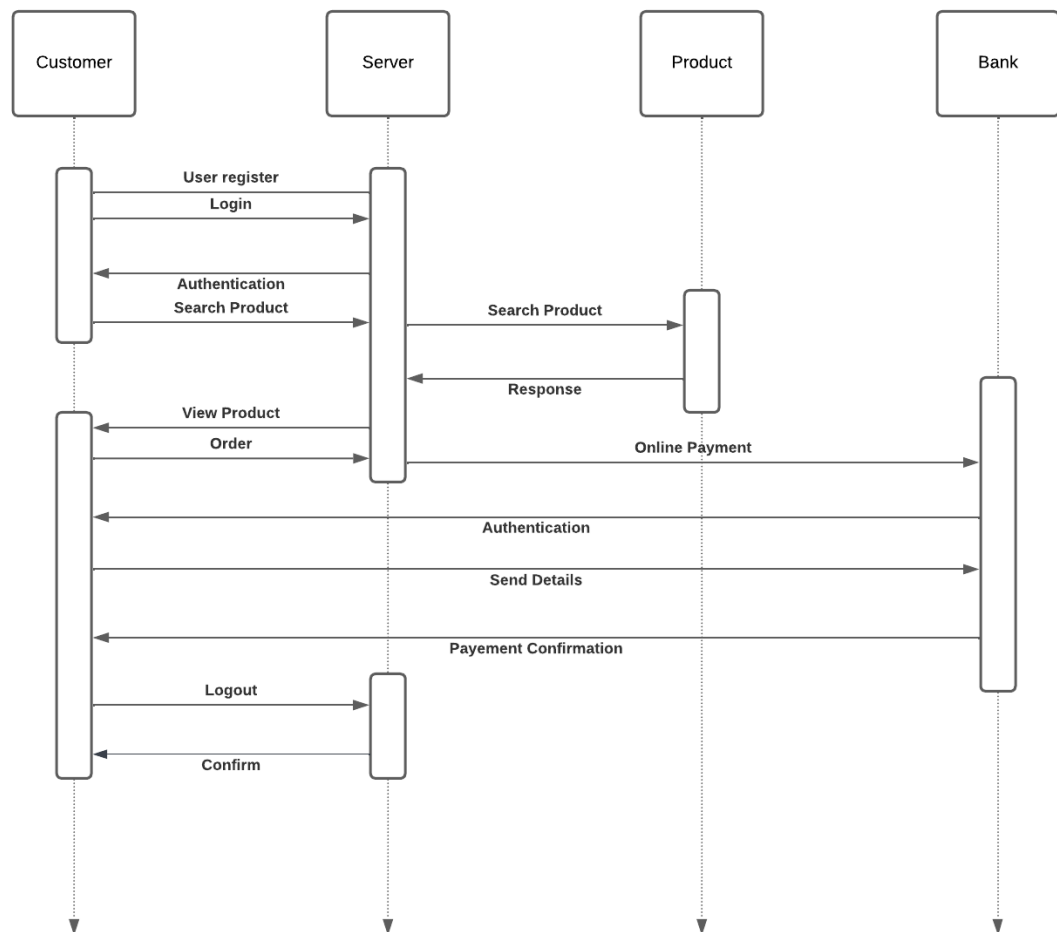
Messages can be broadly classified into the following categories:

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

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 –

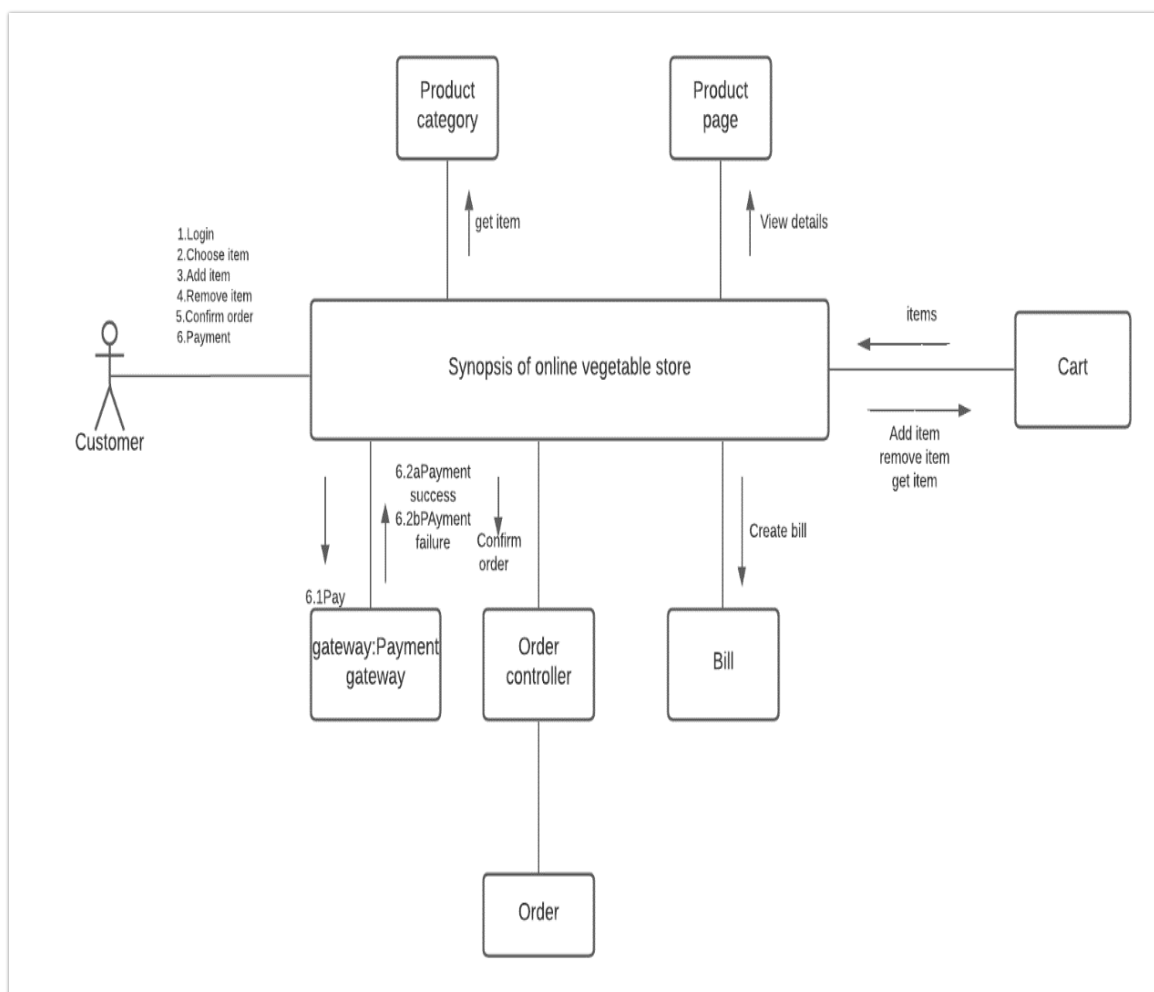
- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components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



4.2.3 Collaboration Diagram

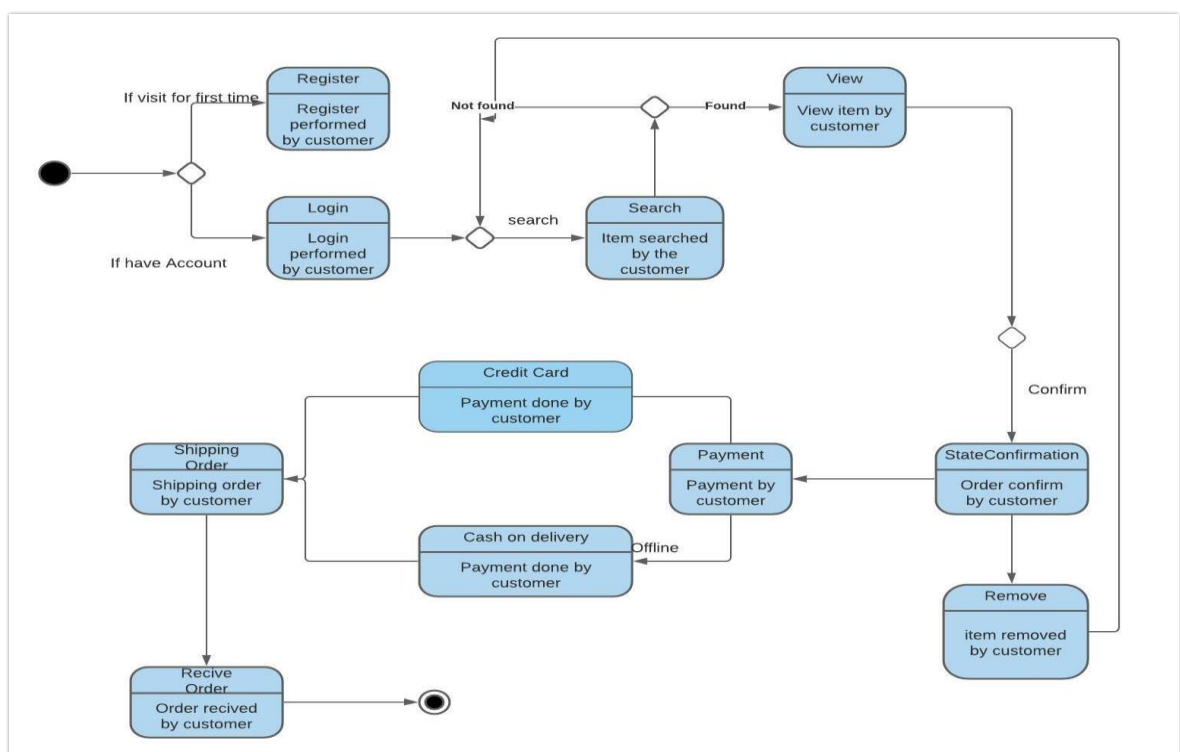
A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.



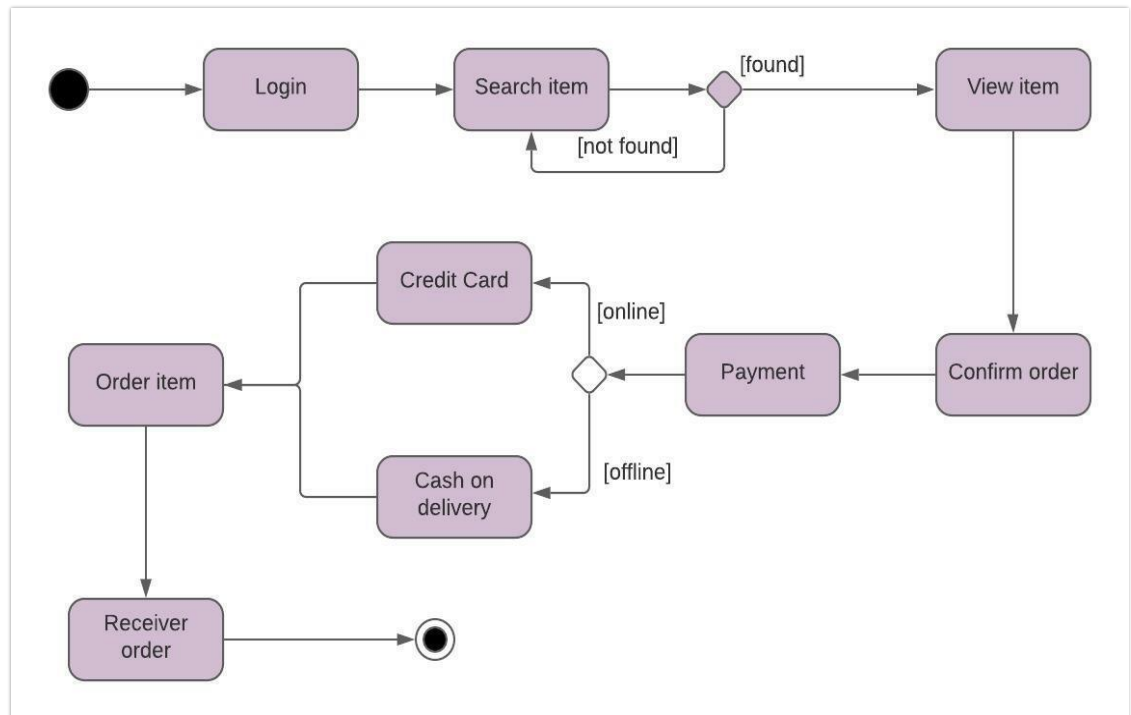
4.2.4 State Chart Diagram

State Diagram are used to capture the behaviour of a software system. UML State machine diagrams can be used to model the behaviour of a class, a subsystem, a package, or even an entire system. It is also called a State chart or State Transition diagram. Statechart diagrams provide us an efficient way to model the interactions or communication that occur within the external entities and a system. These diagrams are used to model the event-based system. A state of an object is controlled with the help of an event. Statechart diagrams are used to describe various states of an entity within the application system.



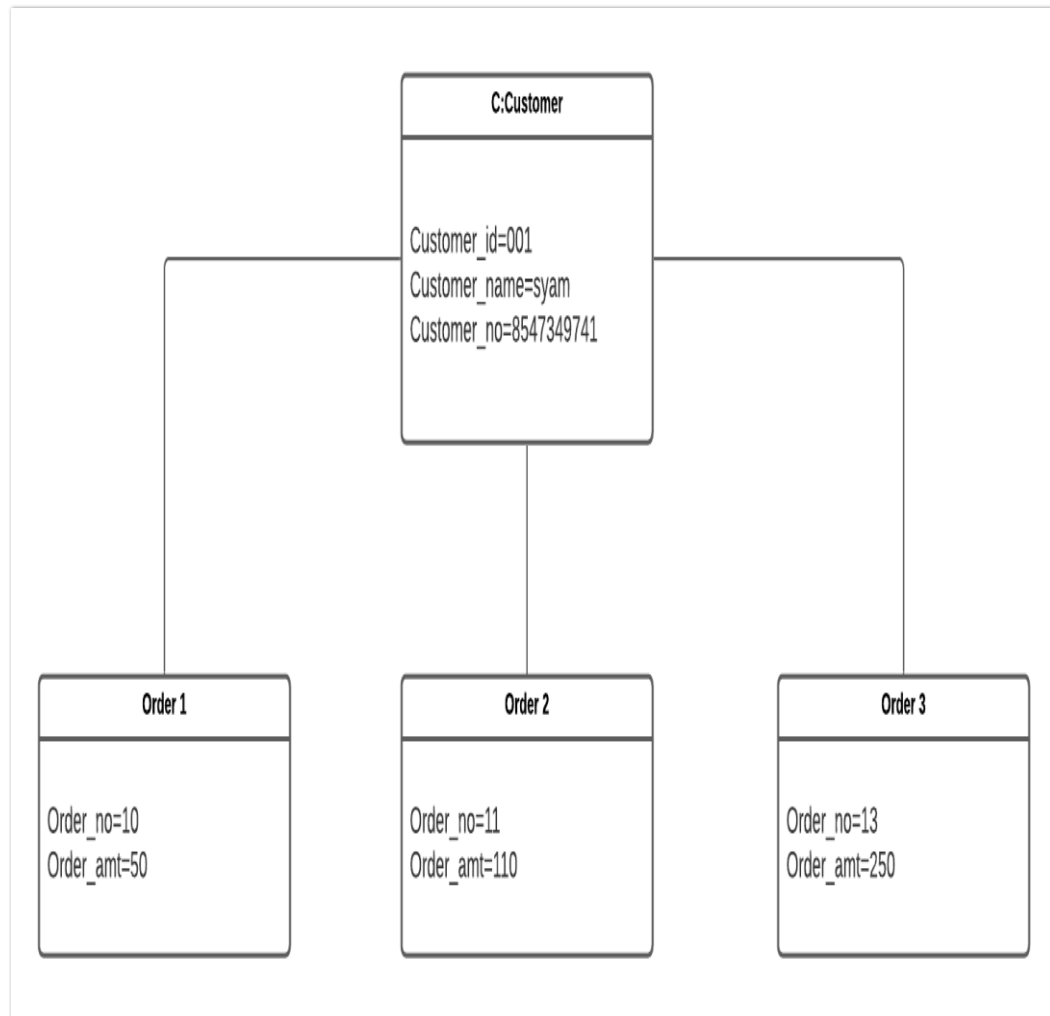
4.2.5 Activity Diagram

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.



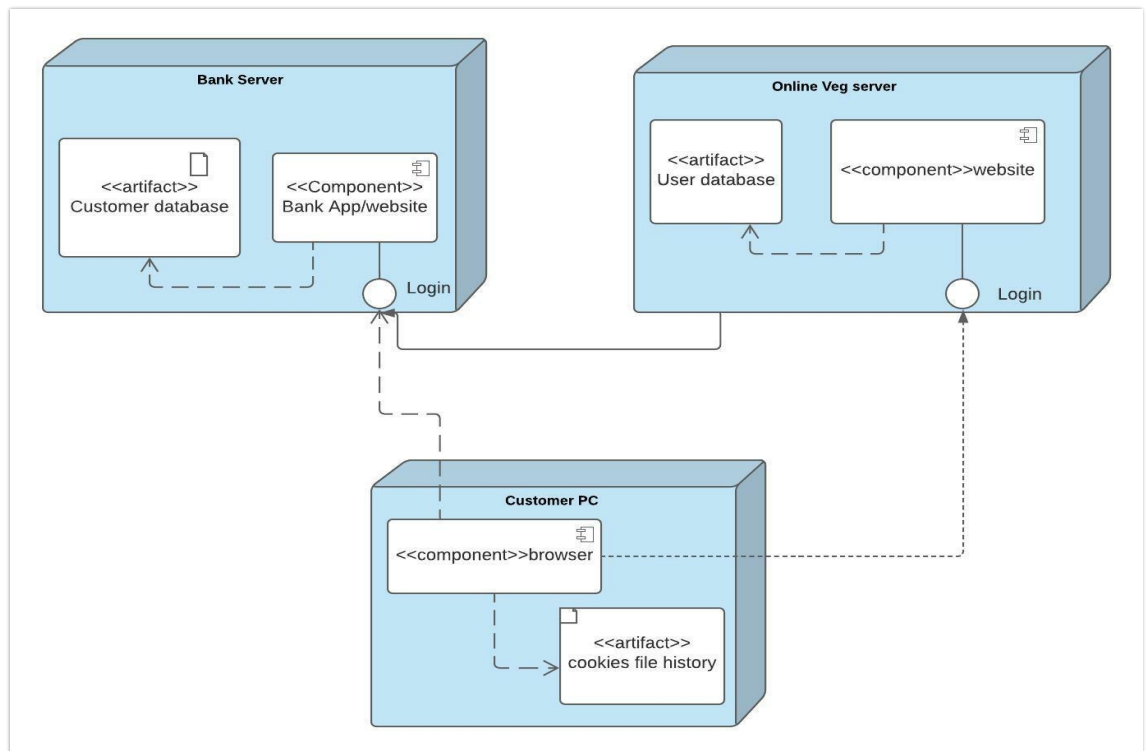
4.2.6 Object Diagram

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.



4.2.7 Deployment Diagram

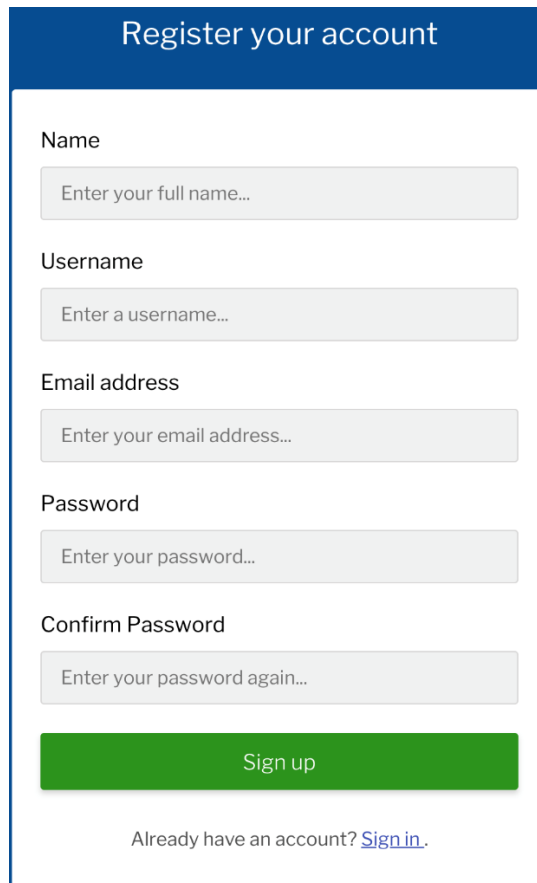
Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.



4.3 USER INTERFACE DESIGN

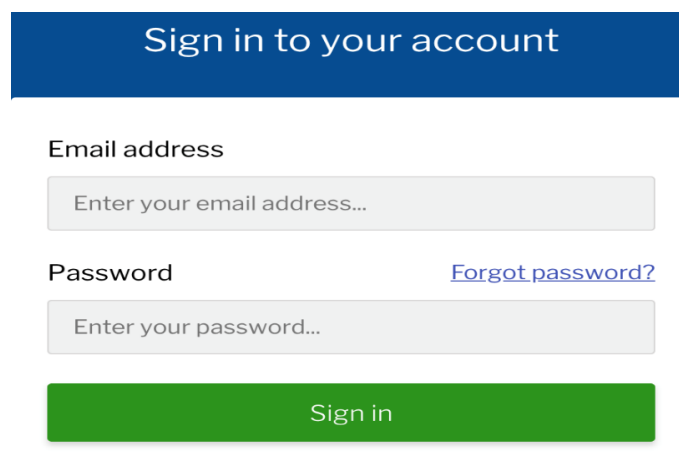
4.3.1-INPUT DESIGN

Form Name: User Registration



The registration form is titled "Register your account" in a blue header. It contains five input fields: "Name" (placeholder: "Enter your full name..."), "Username" (placeholder: "Enter a username..."), "Email address" (placeholder: "Enter your email address..."), "Password" (placeholder: "Enter your password..."), and "Confirm Password" (placeholder: "Enter your password again..."). Below the fields is a green "Sign up" button. At the bottom, there is a link: "Already have an account? [Sign in.](#)"

Form Name: User Login



The login form is titled "Sign in to your account" in a blue header. It contains two input fields: "Email address" (placeholder: "Enter your email address...") and "Password" (placeholder: "Enter your password..."). To the right of the password field is a link: "[Forgot password?](#)". Below the fields is a green "Sign in" button.

4.3.2 OUTPUT DESIGN

User Login



The User Login page features a top navigation bar with links for Home, About us, Blog, and Pricing. The VegiFruit SuperMarket logo, with the tagline 'online groceries', is centered. A large banner on the left shows a woman feeding a child, with the text 'Something for our loved ones' and a description of the service. On the right, there are input fields for email and password, a 'Forgot password' link, a green 'Login' button, a 'Sign Up' link for new users, and a 'Continue with Google' button.

Home About us Blog Pricing

VegiFruit SuperMarket
online groceries

Something for our loved ones

VegiFruit Supermarket offer Customers the best service and sell only the freshest, safest products. Moreover, we always look for great products for today and tomorrow. We make sure to give Customers low prices with the best value.

Enter your email

Enter your password

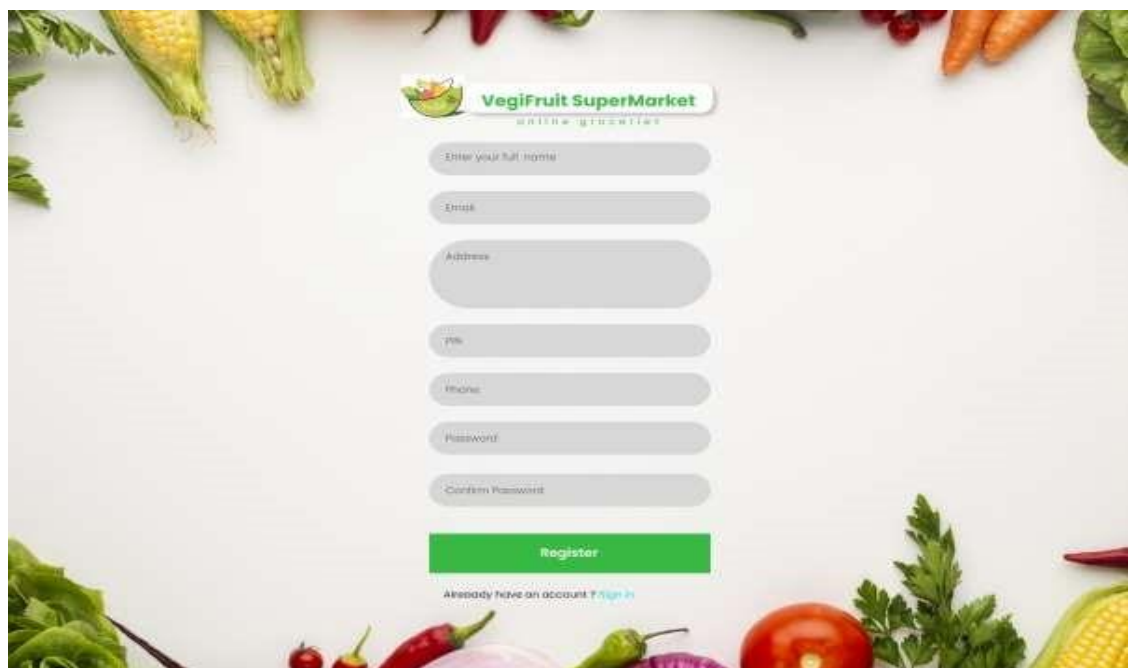
[Forgot password](#)

Login

Don't have an account? [Sign Up](#)

 [Continue with Google](#)

User Registration



The User Registration page features the VegiFruit SuperMarket logo and a series of input fields for user information: full name, email, address, pin, phone, password, and confirm password. A green 'Register' button is at the bottom, with a link to 'Sign In' for existing users.

VegiFruit SuperMarket
online groceries

Enter your full name

Email

Address

Pin

Phone

Password

Confirm Password

Register

Already have an account? [Sign In](#)

4.4 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.

4.4.1 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.

4.4.2 Normalization

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

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.

TABLE DESIGN**tb_login****Primary Key:Login_id****Foreign Key:**

Column	Datatype	Key	Description
Login_id	Int(20)	Primary Key	Login id for the user
Login_Email	Varchar(50)	Not null	Email of the user
Login_Password	Varchar(50)	Not null	Password of the user
Login_Name	Varchar(50)	Not null	Name of the user
Login_status	Int(10)	Not null	

tb_product**Primary Key: Product_id****Foreign Key:Category_id**

Column	Datatype	Key	Description
Product_id	Int(20)	Primary Key	Id for the product
Category_id	Int(20)	Foreign Key	Category_id
Product_Price	Float(50)	Not null	Price of the product
Product_Quantity	Float(50)	Not null	Quantity of the product
Product_Photo	Varchar(50)	Not null	Photo
Product_Description	Varchar(50)	Not null	Description
status	Int(10)	Not null	

tb_order**Primary Key: Order_id****Foreign Key: Login_id, Product_id**

Column	Datatype	Key	Description
Order_id	Int(20)	Primary Key	Id for sales
Login_id	Int(20)	Foreign Key	Login id of user
Product_id	int(20)	Foreign key	Id of the product
Order_Quantity	Float(50)	Not null	
Order_Purchasedate	date	Not null	Purchase date
Order_Tprice	float(50)	Not null	Total price
status	Int(10)	Not null	

tb_category**Primary Key: Category_id****Foreign Key:**

<u>column</u>	<u>Datatype</u>	<u>key</u>	<u>descriptions</u>
Category_id	Int(20)	Primary key	Id for the different category
Category_name	Varchar(50)	Not null	Category name
status	Int(10)	Not null	

tb_cart**Primary Key:Cart_id****Foreign Product_id**

<u>column</u>	<u>Datatype</u>	<u>key</u>	<u>descriptions</u>
Cart_id	Int(20)	Primary key	Id for cart
Product_id	Int(20)	Foreign key	Id of the product
Login_id	Int(20)	Foreign Key	Login id
status	Int(10)	Not Null	

tb_payment**Primary Key: Payment_id****Foreign Key:Login_id, Order_id**

<u>column</u>	<u>Datatype</u>	<u>key</u>	<u>descriptions</u>
Payment_id	Int(20)	Primary key	Payment_id
Login_id	Int(20)	Foreign key	Login id
Order_id	Int(20)	Foreign Key	Sales id
Payment_Date	Date	Not null	Date of payment
Payment_Tprice	Float(50)	Not null	Total price

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.

5.2.1.1 Test Case

Test Case 1					
Project Name: Organi Shop					
Login Test Case					
Test Case ID: Fun_1			Test Designed By: Parvathy R		
Test Priority(Low/Medium/High):High			Test Designed Date: 19-05-2022		
Module Name: Login Screen			Test Executed By : Mr Grace Joseph		
Test Title : Verify login with validemail 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 toLogin Page		Login Page should be displayed	Login page displayed	Pass
2	Provide Valid Email Id	User Name: paruchinnu1932@gmail.com	User should beable to Login	User Logged inand navigated to Subadmin Dashboard with records	Pass
3	Provide Valid Password	Password: chinnu98			
4	Click on Sign In button				
5	Provide Invalid Email Id or password	Email Id: user@gmail.l.Com Password: User1234	User shouldnot be able to Login	Message for enter valid email id or password displayed	Pass
6	Provide Null Email 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 newtest;

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

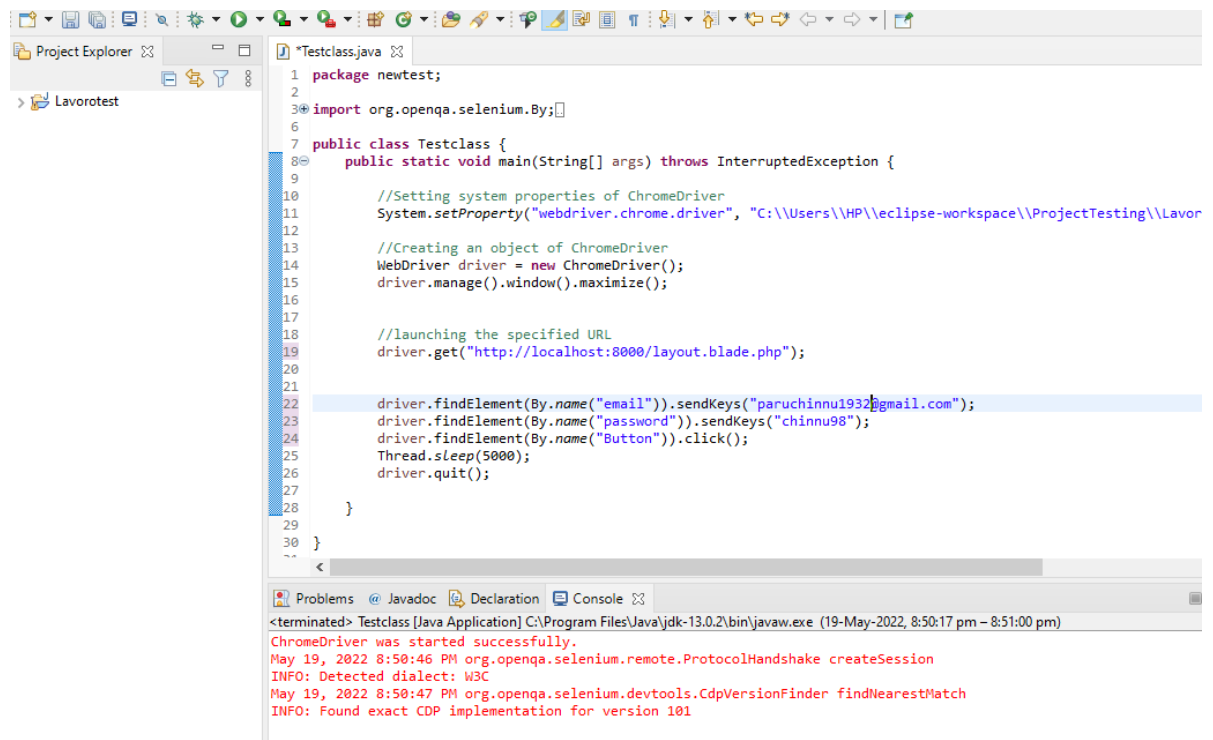
public class Testclass {
    public static void main(String[] args) throws InterruptedException {

        //Setting system properties of ChromeDriver
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\eclipse-workspace\\ProjectTesting\\Lavorotest\\chromedriver.exe");

        //Creating an object of ChromeDriver
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        //launching the specified URL
        driver.get("http://localhost:8000/layout.blade.php");

        driver.findElement(By.name("email")).sendKeys("paruchinnu1932@gmail.com");
        driver.findElement(By.name("password")).sendKeys("chinnu98");
        driver.findElement(By.name("Button")).click();
        Thread.sleep(5000);
        driver.quit();
    }
}
```



Test Case-2

Project Name: Organi Shop					
Cart Test Case					
Test Case ID: Fun_2			Test Designed By: Parvathy R		
Test Priority(Low/Medium/High):High			Test Designed Date: 21-05-2022		
Module Name: Payment Screen			Test Executed By : Ms Grace Joseph		
Test Title : Adding product to cart and payment			Test Execution Date: 21-05-2022		
Description: Test the payment Page					
Pre-Condition :Must be a valid customer					
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 Email Id	User Name:paruchinnu1932@gmail.com	User should be able to Login	User Logged in and navigated to Sub admin Dashboard with records	Pass
3	Provide Valid Password	Password: 123456			
4	Click on Sign In button				
5	Choose products	Products: Carrot			Pass
6	Add to cart				
7	Buy now				
8	Payment	Online Payment	Payment Successful	Payment Successful	Pass
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 chromedriver.DriverSetup;

public class Cart {
    public static WebDriver driver;

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

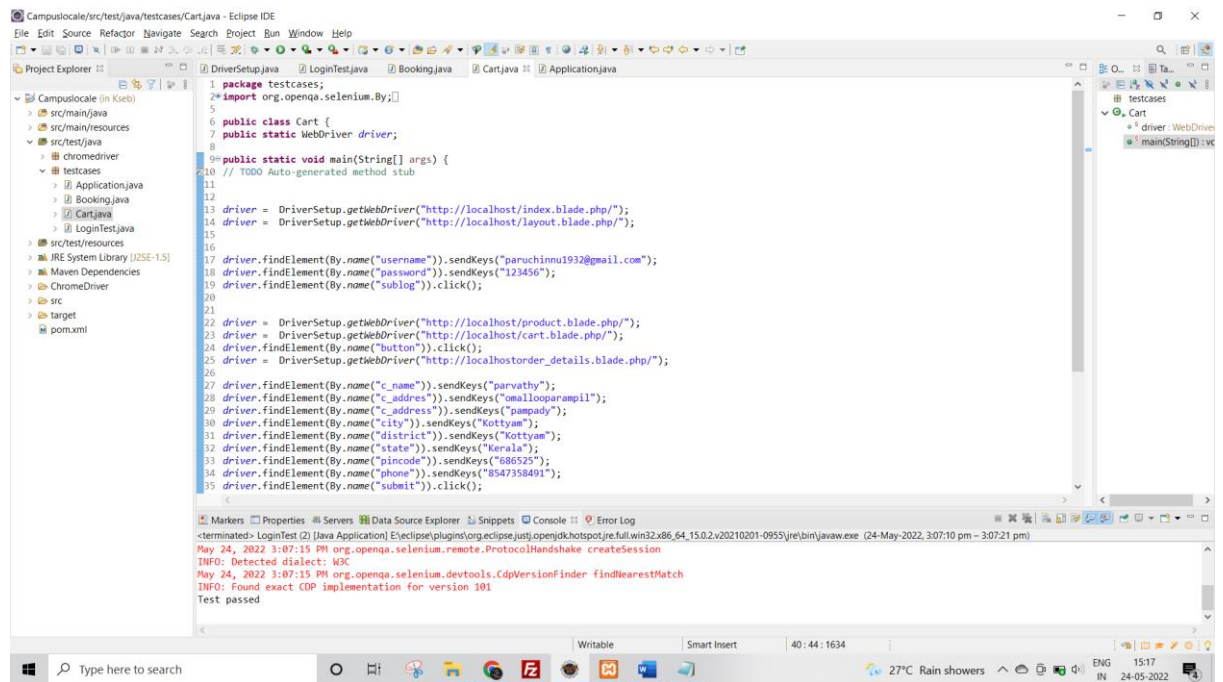
        driver = DriverSetup.getWebDriver("http://localhost/ index.blade.php/");
        driver = DriverSetup.getWebDriver("http://localhost/ layout.blade.php/");

        driver.findElement(By.name("username")).sendKeys("paruchinnu1932@gmail.com");
        driver.findElement(By.name("password")).sendKeys("123456");
        driver.findElement(By.name("sublog")).click();

        driver = DriverSetup.getWebDriver("http://localhost/ product.blade.php /");
        driver = DriverSetup.getWebDriver("http://localhost/ cart.blade.php /");
        driver.findElement(By.name("b1")).click();
        driver = DriverSetup.getWebDriver("http://localhost/order_details.blade.php /");

        driver.findElement(By.name("name")).sendKeys("parvathy");
        driver.findElement(By.name("addres")).sendKeys("omallooparampil(h)");
        driver.findElement(By.name("district")).sendKeys("Kottayam");
        driver.findElement(By.name("state")).sendKeys("Kerala");
        driver.findElement(By.name("pincode")).sendKeys("686525");
        driver.findElement(By.name("phone")).sendKeys("85473584911");
        driver.findElement(By.name("submit")).click();

        String actualUrl = "http://localhost/payment.blade.php /";
        String expectedUrl= driver.getCurrentUrl();
        if(actualUrl.equalsIgnoreCase(expectedUrl)) {
            System.out.println("Test passed"); }
        else {
            System.out.println("Test failed"); }
        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.

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.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The project entitled Organi Shop completed successfully. The project developed using Laravel and MySQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement. The expanded functionality of today's software requires an appropriate approach towards software development. The project has all the required essential features. This project has a user side where he/she can view product category and add products to cart and proceed for checkout whereas from admin side he/she can view sales, number of product, users, add product and categories. The user can also leave comments on each product if he/she wants.

7.2 FUTURE SCOPE

There is a scope for further development in our project to a great extent. A number of features can be added to this system in future like providing. The feature we wished to implement was providing classes for customer so that different offers can be given to each class. System may keep track of history of purchases provide suggestions based on their history using Machine Learning Algorithm. These features could have been implemented if time and skills did not limit me.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- <https://www.tutorialspoint.com/mysql/>
- <https://www.JSP.net/>
- <https://www.xampp.com/en/>
- <https://www.tutorialspoint.com/java/>
- Database programming with JDBC and Java by O'Reilly
- <https://www.javatpoint.com/javatutorial/>

WEBSITES:

- www.w3schools.com
- www.jQuery.com
- <https://app.diagrams> laravel

CHAPTER 9

APPENDIX

9.1 Sample Code

Layout.blade.php

```
<!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">
    <title>@yield('page_title')</title>
    <link href="{{ asset('front_assets/css/font-awesome.css') }}" rel="stylesheet">
    <link href="{{ asset('front_assets/css/bootstrap.css') }}" rel="stylesheet">
    <link href="{{ asset('front_assets/css/jquery.smartmenus.bootstrap.css') }}" rel="stylesheet">
    <link
        rel="stylesheet"
        href="{{ asset('front_assets/css/jquery.simpleLens.css') }}"
        type="text/css">
    <link rel="stylesheet" type="text/css" href="{{ asset('front_assets/css/slick.css') }}">
    <link rel="stylesheet" type="text/css" href="{{ asset('front_assets/css/nouislider.css') }}">
    <link id="switcher" href="{{ asset('front_assets/css/theme-color/default-theme.css') }}"
    rel="stylesheet">
    <link href="{{ asset('front_assets/css/sequence-theme.modern-slide-in.css') }}" rel="stylesheet"
    media="all">
    <link href="{{ asset('front_assets/css/style.css') }}" rel="stylesheet">

    <!-- Google Font -->
    <link href='https://fonts.googleapis.com/css?family=Lato' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Raleway' rel='stylesheet' type='text/css'>

    <script>
        var PRODUCT_IMAGE="{{ asset('storage/media/') }}";
    </script>
```

```
</head>

<body>

<!-- wpf loader Two -->

<div id="wpf-loader-two">

  <div class="wpf-loader-two-inner">

    <span>Loading</span>

  </div>

</div>

<!-- / wpf loader Two -->

<!-- SCROLL TOP BUTTON -->

  <a class="scrollToTop" href="#"><i class="fa fa-chevron-up"></i></a>

<!-- END SCROLL TOP BUTTON -->


<!-- Start header section -->

<header id="aa-header">

  <!-- start header top -->

  <div class="aa-header-top">

    <div class="container">

      <div class="row">

        <div class="col-md-12">

          <div class="aa-header-top-area">

            <!-- start header top left -->


            <div class="aa-header-top-left">


              <!-- start cellphone -->

              <div class="cellphone hidden-xs">

                <p><span class="fa fa-phone"></span>9400257158</p>

              </div>

              <!-- / cellphone -->
```

```
</div>

<!-- / header top left -->

<div class="aa-header-top-right">

    <ul class="aa-head-top-nav-right">

        <li class="hidden-xs"><a href="{{ url('/cart') }}">My Cart</a></li>

        @if(session()->has('FRONT_USER_LOGIN')!=null)

            <li><a href="{{ url('/order') }}">My Order</a></li>

            <li><a href="{{ url('/logout') }}">Logout</a></li>

            @else

                <li><a href="" data-toggle="modal" data-target="#login-modal">Login</a></li>

            @endif

        </ul>

    </div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- start header bottom -->

<div class="aa-header-bottom">

    <div class="container">
```

```

<div class="row">
  <div class="col-md-12">
    <div class="aa-header-bottom-area">
      <!-- logo -->
      <div class="aa-logo">
        <!-- Text based logo -->
        <a href="{ {url('/') }}">
          <span class="fa fa-shopping-cart"></span>
          <p>ORGANI<strong>SHOP</strong></p>
        </a>
        <!-- img based logo -->
        <!-- <a href="javascript:void(0)"></a> -->
      </div>
      <!-- / logo -->
      <!-- cart box -->
      @php
$addToCartTotalItem=getAddToCartTotalItem();
$totalCartItem=count($addToCartTotalItem);
$totalPrice=0;
      @endphp

<div class="aa-cartbox">
  <a class="aa-cart-link" href="#">
    <span class="fa fa-shopping-basket"></span>
    <span class="aa-cart-title">CART</span>
    <span class="aa-cart-notify">{ { $totalCartItem } }</span>
  </a>
  <div class="aa-cartbox-summary">
    @if($totalCartItem>0)

```

```

<ul>
    @foreach($getAddToCartTotalItem as $cartItem)
    <?php
        @$totalPrice=$totalPrice+($cartItem->qty*$cartItem->price)
    ?>
    <li>
        <a class="aa-cartbox-img" href="#"></a>
        <div class="aa-cartbox-info">
            <h4><a href="#">{{ $cartItem->name }}</a></h4>
            <p>{{ $cartItem->qty }} * Rs {{ $cartItem->price }}</p>
        </div>
        <a class="aa-remove-product" href="#"><span class="fa fa-times"></span></a>
    </li>
    @endforeach
    <li>
        <span class="aa-cartbox-total-title">
            Total
        </span>
        <span class="aa-cartbox-total-price">
            Rs {{ $totalPrice }}
        </span>
    </li>
</ul>
<a class="aa-cartbox-checkout aa-primary-btn" href="{{ url('/cart') }}">Cart</a>

@endif
</div>
</div>

```



```

        <!-- / cart box -->

        <!-- search box -->

        <div class="aa-search-box">

            <form action="">

                <input type="text" id="search_str" placeholder="Search here">

                <button type="button" onclick="funSearch()"><span class="fa fa-
search"></span></button>

            </form>

        </div>

        <!-- / search box -->

    </div>

</div>

</div>

</div>

</div>

</div>

<!-- / header bottom -->

</header>

<!-- / header section -->

<!-- menu -->

<section id="menu">

<!-- Start slider -->

```

```
@section('container')
```

```
@show
```

```

<!-- footer -->

<footer id="aa-footer">

    <!-- footer bottom -->

    <div class="aa-footer-top">

        <div class="container">

            <div class="row">

                <div class="col-md-12">

```

```
<div class="aa-footer-top-area">
  <div class="row">
    <div class="col-md-3 col-sm-6">
      <div class="aa-footer-widget">
        <h3>Main Menu</h3>
        <ul class="aa-footer-nav">
          <li><a href="#">Home</a></li>
          <li><a href="#">Our Services</a></li>
          <li><a href="#">Our Products</a></li>
          <li><a href="#">Our Farmers</a></li>
          <li><a href="/about">About Us</a></li>
          <li><a href="/contact">Contact Us</a></li>
        </ul>
      </div>
    </div>
    <div class="col-md-3 col-sm-6">
      <div class="aa-footer-widget">
        <h3>Useful Links</h3>
        <ul class="aa-footer-nav">
          <li><a href="#">Site Map</a></li>
          <li><a href="#">Search</a></li>
          <li><a href="#">Advanced Search</a></li>
          <li><a href="#">Suppliers</a></li>
          <li><a href="#">FAQ</a></li>
        </ul>
      </div>
    </div>
  </div>
```

```
</div>
<div class="col-md-3 col-sm-6">
  <div class="aa-footer-widget">
    <div class="aa-footer-widget">
      <h3>Contact Us</h3>
      <address>
        <p>Organishop,Kakkanad, Eranakulam</p>
        <p><span class="fa fa-phone"></span>9400257158</p>
        <p><span class="fa fa-envelope"></span>organishop@gmail.com</p>
      </address>
      <div class="aa-footer-social">
        <a href="#"><span class="fa fa-facebook"></span></a>
        <a href="#"><span class="fa fa-twitter"></span></a>
        <a href="#"><span class="fa fa-google-plus"></span></a>
        <a href="#"><span class="fa fa-youtube"></span></a>
      </div>
    </div>
  </div>
</div>
</div>

</div>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- footer-bottom -->
<div class="aa-footer-bottom">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
```

```

<div class="modal-dialog">
  <div class="modal-content">
    <div class="modal-body">
      <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
      <div id="popup_login">
        <h4>LOGIN OR REGISTER</h4>
        <form class="aa-login-form" id="frmLogin">
          <label for="">Email address<span>*</span></label>
          <input type="email" placeholder="Email" name="str_login_email" required
value="{{ $login_email }}">
          <label for="">Password<span>*</span></label>
          <input type="password" placeholder="Password" name="str_login_password" required
value="{{ $login_pwd }}">
          <button class="aa-browse-btn" type="submit" id="btnLogin">Login</button>
          <label for="rememberme" class="rememberme"><input type="checkbox"
id="rememberme" name="rememberme" {{ $is_remember }}> Remember me </label>
        </form>
        <div id="login_msg"></div>
        <p class="aa-lost-password"><a href="javascript:void(0)"
onclick="forgot_password()">Forgot password</a></p>
        <div class="aa-register-now">
          Don't have an account?<a href="{{ url('registration') }}">Register now!</a>
        </div>
        @csrf
      </form>
    </div>
    <div id="popup_forgot" style="display:none;">
      <h4>FORGOT PASSWORD</h4>
      <form class="aa-login-form" id="frmForgot">

```

```

<label for="">Email address<span>*</span></label>

<input type="email" placeholder="Email" name="str_forgot_email" required>

<button class="aa-browse-btn" type="submit" id="btnForgot">Submit</button>

<br><br>

<div id="forgot_msg"></div>

<div class="aa-register-now">

    Login Form?<a href="javascript:void(0)" onclick="show_login_popup()">Login
now!</a>

</div>

@csrf

</form>

</div>

</div>

</div><!-- /.modal-content -->

</div><!-- /.modal-dialog -->

</div>

<!-- jQuery library -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

<script src="{ { asset('front_assets/js/bootstrap.js') } }"></script>

<script type="text/javascript" src="{ { asset('front_assets/js/jquery.smartmenus.js') } }"></script>

<script src="{ { asset('front_assets/js/jquery.smartmenus.bootstrap.js') } }" type="text/javascript"></script>

<script src="{ { asset('front_assets/js/sequence.js') } }"></script>

<script src="{ { asset('front_assets/js/sequence-theme.modern-slide-in.js') } }"></script>

<script src="{ { asset('front_assets/js/jquery.simpleGallery.js') } }" type="text/javascript"></script>

<script type="text/javascript" src="{ { asset('front_assets/js/jquery.simpleLens.js') } }"></script>

<script type="text/javascript" src="{ { asset('front_assets/js/slick.js') } }"></script>

<script type="text/javascript" src="{ { asset('front_assets/js/nouislider.js') } }"></script>

<script src="{ { asset('front_assets/js/custom.js') } }"></script>

```

```
</body>
```

```
</html>
```

Checkout.blade.php

```
@extends('front/layout')
```

```
@section('page_title','Checkout Page')
```

```
@section('container')
```

```
<!-- catg header banner section -->
```

```
<section id="aa-catg-head-banner">
```

```
<div class="aa-catg-head-banner-area">
```

```
<div class="container">
```

```
</div>
```

```
</div>
```

```
</section>
```

```
<!-- / catg header banner section -->
```

```
<section id="checkout">
```

```
<div class="container">
```

```
<div class="row">
```

```
<div class="col-md-12">
```

```
<div class="checkout-area">
```

```
<form id="frmPlaceOrder">
```

```
<div class="row">
```

```
<div class="col-md-8">
```

```
<div class="checkout-left">
```

```
<div class="panel-group" id="accordion">
```

```
@if(session()->has('FRONT_USER_LOGIN')==null)
```

```
<input type="button" value="Login" class="aa-browse-btn" data-toggle="modal" data-target="#login-modal">
```

```

<br/><br/>
OR
<br/><br/>
@endif
<!-- Shipping Address -->
<div class="panel panel-default aa-checkout-billaddress">
  <div class="panel-heading">
    <h4 class="panel-title">
      <a data-toggle="collapse" data-parent="#accordion">
        User Details Address
      </a>
    </h4>
  </div>
  <div id="collapseFour" class="panel-collapse collapse in">
    <div class="panel-body">
      <div class="row">
        <div class="col-md-4">
          <div class="aa-checkout-single-bill">
            <input type="text" placeholder=" Name*" value="{{ $customers['name'] }}"
name="name" required>
          </div>
        </div>
        <div class="col-md-4">
          <div class="aa-checkout-single-bill">
            <input type="email" placeholder="Email Address*"
value="{{ $customers['email'] }}" name="email" required>
          </div>
        </div>
        <div class="col-md-4">
          <div class="aa-checkout-single-bill">
            <input type="tel" placeholder="Phone*" value="{{ $customers['mobile'] }}"
name="mobile" required>
          </div>
        </div>
      </div>
    </div>
  </div>

```



```

    </div>

    </div>

    <div class="row">
        <div class="col-md-12">
            <div class="aa-checkout-single-bill">
                <textarea cols="8" rows="3" name="address" required placeholder="Enter
Address*">{{ $customers['address'] }}</textarea>
            </div>
        </div>
    </div>

    <div class="row">
        <div class="col-md-4">
            <div class="aa-checkout-single-bill">
                <input type="text" placeholder="City / Town*"
value="{{ $customers['city'] }}" name="city" required>
            </div>
        </div>
    </div>

    </div>

    </div>

    </div>

    </div>

    </div>

    <div class="col-md-4">
        <div class="checkout-right">
            <h4>Order Summary</h4>
            <div class="aa-order-summary-area">
                <table class="table table-responsive">

```

```

<thead>

<tr>

<th>Product</th>

<th>Total</th>

</tr>

</thead>

<tbody>

@php
$totalPrice=0;

@endphp

@foreach($cart_data as $list)

@php
$totalPrice=$totalPrice+($list->price*$list->qty)
@endphp

<tr>

<td>{{ $list->name }} <strong> * {{ $list->qty }}</strong>

<br/>

</td>

<td>{{ $list->price*$list->qty }}</td>

</tr>

@endforeach

</tbody>

<tfoot>

<tr class="hide show_coupon_box">

<th>Coupon Code <a href="javascript:void(0)"
onclick="remove_coupon_code()" class="remove_coupon_code_link">Remove</a></th>

<td id="coupon_code_str"></td>

</tr>

<tr>

```

```

        <th>Total</th>

        <td id="total_price">INR { { $totalPrice } }</td>

    </tr>

</tfoot>

</table>

</div>

<h4>Coupon Code</h4>

<div class="aa-payment-method coupon_code">

    <input type="text" placeholder="Coupon Code" class="aa-coupon-code
apply_coupon_code_box" name="coupon_code" id="coupon_code">

    <input type="button" value="Apply Coupon" class="aa-browse-btn
apply_coupon_code_box" onclick="applyCouponCode()">

    <div id="coupon_code_msg"></div>

</div>

<br/>

<h4>Payment Method</h4>

<div class="aa-payment-method">

    <label for="cod"><input type="radio" id="cod" name="payment_type" value="COD"
checked> Cash on Delivery </label>

    <label for="instamojo">

        <input type="radio" id="instamojo" name="payment_type" value="Gateway"> Via
Instamojo </label>

        <input type="submit" value="Place Order" class="aa-browse-btn"
id="btnPlaceOrder">

    </div>

    <div id="order_place_msg"></div>

</div>

</div>

</div>

</div>

@csrf

</form>

</div>

```

```
</div>
```

```
</div>
```

```
</div>
```

```
</section>
```

```
@endsection
```

Order.blade.php

```
@extends('front/layout')
```

```
@section('page_title','Order Page')
```

```
@section('container')
```

```
<!-- catg header banner section -->
```

```
<section id="aa-catg-head-banner">
```

```
<div class="aa-catg-head-banner-area">
```

```
<div class="container">
```

```
</div>
```

```
</div>
```

```
</section>
```

```
<!-- / catg header banner section -->
```

```
<section id="cart-view">
```

```
<div class="container">
```

```
<div class="row">
```

```
<div class="col-md-12">
```

```
<div class="cart-view-area">
```

```
<div class="cart-view-table">
```

```
<form action="">
```

```
<div class="table-responsive">
```

```
<table class="table">
```

```
<thead>
```

```
<tr>
```

```
<th>Order Status</th>
<th>Payment Status</th>
<th>Total Amt</th>
<th>Payment ID</th>
<th>Placed At</th>

</tr>

</thead>

<tbody>

  @foreach($orders as $list)

    <tr>

      <td class="order_id_btn"><a href="{ {url('order_detail')}}/{ { $list->id } }">{ { $list->orders_status } }</a></td>

      <td>{ { $list->payment_status } }</td>

      <td>{ { $list->total_amt } }</td>

      <td>{ { $list->payment_id } }</td>

      <td>{ { $list->added_on } }</td>

    </tr>

  @endforeach

</tbody>

</table>

</div>

</form>

<!-- Cart Total view -->

</div>

</div>

</div>

</div>

</section>

@endsection
```

Order.blade.php

```

@extends('admin/layout')

@section('page_title','Order')

@section('order_select','active')

@section('container')

    <h1 class="mb10">Order</h1>

    <div class="row m-t-30">

        <div class="col-md-12">

            <!-- DATA TABLE-->

            <div class="table-responsive m-b-40">

                <table class="table table-borderless table-data3">

                    <thead>

                        <tr>

                            <th>Order ID</th>

                            <th>Customer Details</th>

                            <th>Amt</th>

                            <th>Order Status</th>

                            <th>Payment Status</th>

                            <th>Payment Type</th>

                            <th>Order Date</th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach($orders as $list)

                            <tr>

                                <td><a href="{{url('/admin/order_detail')}}/{{$list->id}}">{{$list->id}}</a></td>

                                <td>

                                    {{$list->name}}<br/>

                                    {{$list->email}}<br/>

                                    {{$list->mobile}}<br/>

                                    {{$list->address}},{{$list->city}}

                                </td>

                            </tr>

                        @endforeach

                    </tbody>

                </table>

            </div>

        </div>

    </div>

```

```

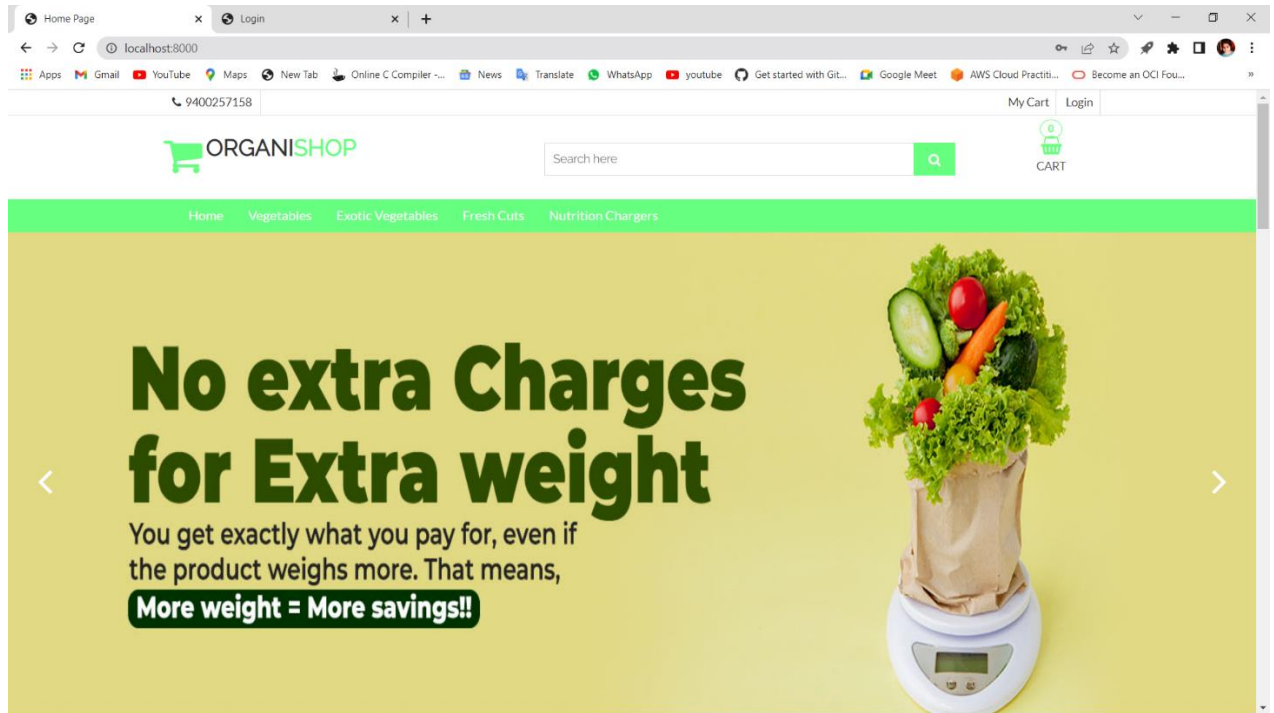
        </td>
        <td>{{ $list->total_amt }}</td>
        <td>{{ $list->orders_status }}</td>
        <td>{{ $list->payment_status }}</td>
        <td>{{ $list->payment_type }}</td>
        <td>{{ $list->added_on }}</td>
    </tr>
    @endforeach
</tbody>
</table>

</div>
    <button onclick="window.print()" class="btn btn-primary">print </button>
    <!-- END DATA TABLE-->
</div>
</div>

@endsection
```

9.2 Screen Shots

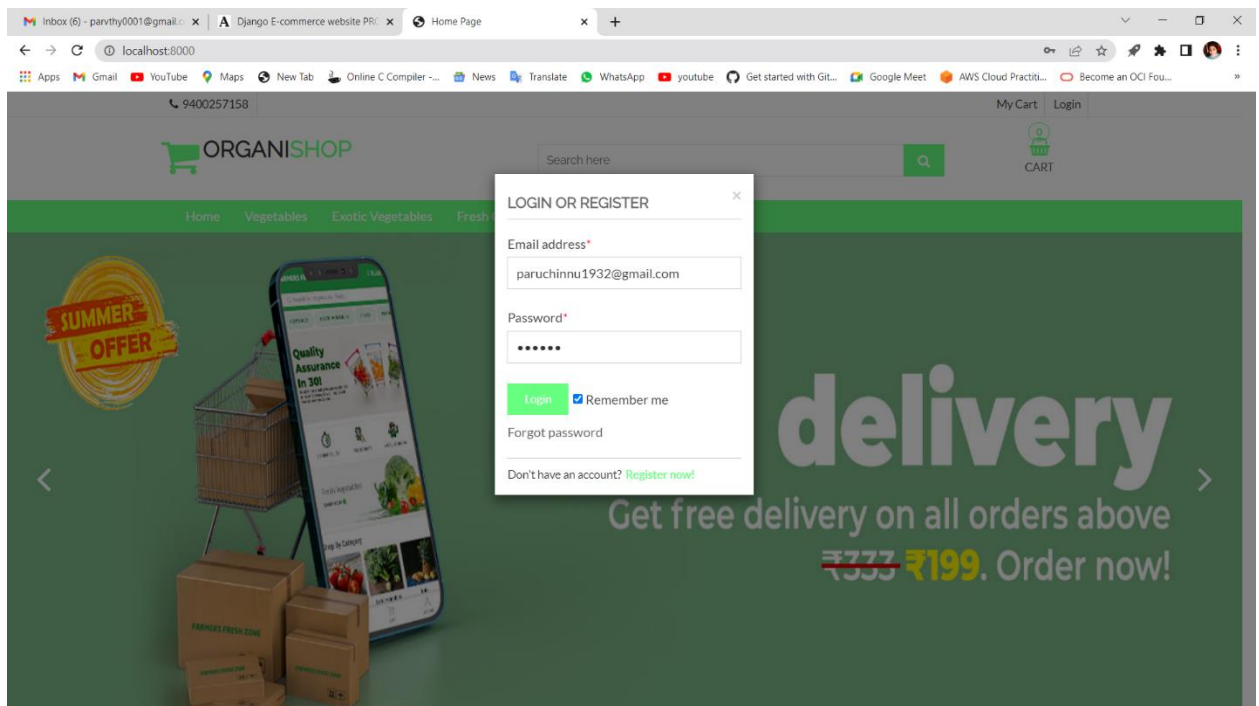
Home page



Signup Page

A screenshot of a web browser displaying the Organi Shop registration page. The browser's address bar shows 'localhost:8000/registration'. The page features a green navigation bar with links: Home, Vegetables, Exotic Vegetables, Fresh Cuts, and Nutrition Chargers. Below the navigation bar is a registration form titled 'Register'. The form contains four input fields: 'Name*', 'Email*', 'Password*', and 'Mobile Number'. Each field has a placeholder text: 'name', 'email', 'password', and 'mobile' respectively. Below the input fields is a green 'Register' button. The browser's tab bar shows 'Inbox (5) - parvthy001@gmail.com', 'Django E-commerce website PR...', and 'Registration Page'. The browser's toolbar includes various icons for search, home, and other functions.

Login Page



ADMIN PAGES

Login Page

