A

Project Report

On

Foody: Your Organic Food Marketplace

Submitted by --

Soumya Das [2405103470003]

Mayank Patidar [2405103470005]

as

Partial fulfilment of Semester II

of Integrated Master of Computer Applications

for A.Y. 2024-2025

Under the Guidance of

Internal Guide name: Sohil Parmar

Submitted To

Parul Institute of Computer Application,

Faculty of IT & Computer Science

Parul University

PARUL INSTITUTE OF COMPUTER APPLICATION

CERTIFICATE

This is to certify that ***Soumya Das , Mayank Patidar*** the student(s) of Parul Institute of Computer Application, has/have satisfactorily completed the project entitled "***Foody: Your Organic Food Marketplace"*** as a part of course curriculum in IMCA Semester- II for the academic year 2024-2025 under guidance of ***Prof. Sohil Parmar*** .

Enrolment Number: 2405103470003

Enrolment Number: 2405103470005

Quality of work	**Grade**	**Sign of Internal guide**
Poor / Average / Good /Excellent	**B / B+ / A / A+**	

Date of submission:

HOD,

Prof. Abhishek Mehta

Principal,

Dr. Priya Swaminarayan

ACKNOWLEDGEMENT

The project presented as a part of our academic curriculum was our first experience of this kind. We approached this project not merely as a course requirement but as a meaningful opportunity to learn, explore, and develop skills in commercial software technologies.

We are pleased to state that we have successfully achieved our objectives in building and presenting this project. This accomplishment would not have been possible without the support and encouragement of several individuals.

First and foremost, we express our sincere gratitude to our internal guide and coordinator, and our respected principal, **Dr. Priya Swaminarayan**, for providing the necessary facilities and guidance throughout the project.

We also take this opportunity to thank our **internal guide, Prof. Sohil Parmar**, for his professional insights and support during the industrial development phase.

A heartfelt thanks goes to our **parents** for their constant encouragement and support, both emotionally and mentally, which motivated us to perform our best.

We are equally grateful to our **friends and classmates** whose direct and indirect contributions were invaluable and will always be remembered.

This project has given us valuable experience and practical exposure to software development, and it has significantly enriched our academic journey.

Soumya Das

ABSTRACT

Online Fresh Fruits and Vegetables Ordering System is a process in which one can order various fruits and vegetables from various vendor to local restaurant, hotels and even to super markets through the use of internet, just by sitting at home or any place. And the order is delivered to the told location.

The Online Fresh Fruits and Vegetables Ordering System is a simple project which is going to be created using PHP, JavaScript, and CSS. The project connects different restaurants with vendors. The project contains an admin (manager) and the user side. All the management like editing site contents, updating items, adding restaurants, vendors and checking order status can be managed from the admin side.

For the user section, the users can go through the homepage, about, and contact pages. In order to order the items, the user has to create an account and sign in or log in. The items comes with the cost as well. This project makes a convenient way for vendors to buy/purchase items online, without having to go to the market.

The design of this system is simple so that the user won't get any difficulties while working on it.

This is an Online ordering system written using PHP/MySQL.

TABLE OF CONTENT

Chapter 1: Introduction 1

1.1 Software Process Implementation 1
1.2 E-Commerce Overview 2
1.3 Business Models 2
1.4 Project Objectives 2
1.5 Project Scope 2

Chapter 2: Requirements and Analysis 3

2.1 Functional & Non-Functional Requirements 3
2.2 Software and Hardware Requirements 5
2.3 Project Plan & Gantt Chart 6

Chapter 3: Literature Review 6

3.1 Privacy and Security Issues 7
3.2 E-Commerce Processes 7
3.3 Security and Privacy Concerns 10
3.4 Waterfall Model 10
3.5 SDLC 11
3.6 Case Studies 12

Chapter 4: Software & Tool Requirement Analysis 13

4.1 Tools and Techniques 13
4.2 Database Details 13
4.3 Case Studies of Software Analysis 15

Chapter 5: Software Design 19

5.1 Use Case Diagram 19
5.2 System Architecture Diagram 20
5.3 Data Flow Diagrams (DFDs) 21
5.4 Class Diagram 23
5.5 Object Diagram 24
5.6 Deployment Diagram 25
5.7 Use Case Diagrams (Admin/User) 26
5.8 Sequence Diagrams 28
5.9 Activity Diagram 31
5.10 Flowchart 32
5.11 ER Diagram 33
5.12 Collaboration Diagram 34
5.13 Component Diagram 35
5.14 Association Model 35
5.15 State Chart Diagram 36
5.16 Communication Diagram 36

Chapter 6: Algorithm and Its Complexity 37

6.1 Algorithms and Pseudocode 37
6.2 Complexity Analysis 37
6.3 Dry Run and Examples 37
6.4 Search Algorithm & Analysis 38
6.5 Login Check Algorithm 39
6.6 Graphical Representations 40

Chapter 7: Development Phase 41
7.1 Layout and Form Design 41
7.2 Backend & Database Integration 42
7.3 Validation and Security Enhancements 46
Chapter 8: Testing 50
8.1 Testing Types and Methods 50
8.2 Performance, Stress & Load Testing 51
8.3 Functional Testing & Bug Reports 53
8.4 Tools Used & UAT 54
Chapter 9: Conclusion 56
9.1 Project Outcomes and Benefits 56
9.2 Future Scope and Impact 57
Chapter 10: Enhancements & Recommendations 58
References/Bibliography 59

CHAPTER 1

INTRODUCTION

Software Process Implementation – Foody: Your Organic Marketplace:

The **Foody** project follows a structured **software development process**, which involves key activities such as **planning**, **design**, **coding**, and **deployment/maintenance**. The project aims to develop an e-commerce platform for **organic products**, including features such as product catalogs, search functionality, shopping cart management, and secure online transactions. Users can browse products, make purchases, and receive email confirmations. The platform supports user registration/login, and the admin panel allows full site management.

Foody includes features such as:

- Categorized organic product listings
- Product search functionality
- Shopping cart management
- User registration/login
- Email confirmations after purchase
- Secure online transactions

E-Commerce Overview:

E-commerce refers to the buying and selling of products or services over the internet. It encompasses technologies like **mobile commerce**, **electronic funds transfer**, and **online transaction processing**. In the case of **Foody**, e-commerce features include an online storefront, product categories, payment gateways, and secure transactions, facilitating direct consumer purchases from organic product vendors.

E-Commerce Business Models:

There are four primary e-commerce models, and **Foody** primarily follows the **B2C (Business-to-Consumer)** model, where businesses sell directly to consumers:

1. **B2C (Business-to-Consumer)**: Foody allows consumers to purchase organic products directly from the platform.

2. **B2B (Business-to-Business)**: Foody may allow business transactions between organic suppliers and vendors.
3. **C2C (Consumer-to-Consumer)**: A future enhancement, allowing users to buy/sell organic products directly to one another.
4. **B2G (Business-to-Government)**: Potential for partnerships with government programs to promote organic agriculture.

Project Objectives:

- **User Registration/Login**: Users must register to make purchases.
- **Product Catalog**: Categorized listings and detailed product descriptions.
- **Search Functionality**: Allows customized product searches.
- **Cart Management**: Users can add, edit, and cancel orders.
- **Email Notifications**: Confirmation emails sent after a purchase.

Project Scope:

- **Admin Panel**: Full back-end management of users, products, and orders.
- **User Interaction**: Secure front-end for browsing products and making purchases.
- **Order Confirmation**: Real-time updates and email notifications for successful transactions.
- **Educational Use**: The system also serves as a case study for **software development courses**.

CHAPTER 2

REQUIREMENTS AND ANALYSIS

CHAPTER 2.1: FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

2.1.1 Problem Definition

The existing organic food trading system is manual and lacks digital integration, leading to inefficiencies.

Limitations in Existing System:

1. No centralized online platform for selling or purchasing organic food.
2. Difficulty for sellers to reach a wider audience.
3. Manual order management causing errors.
4. Limited product authenticity verification.
5. No real-time product availability updates.
6. Poor communication between buyers and sellers.
7. Offline promotion leading to extra costs.

Proposed System: Foody Organic Marketplace

- Centralized platform for both sellers and buyers.
- Easy product uploads and management.
- One-click order management and live tracking.
- Profile-based system for users and sellers.
- Shareable product URLs for social media.

2.1.2 Requirements Specification

2.1.2.1 User Requirements:

1. User registration with name, email, mobile, and password.
2. Email and mobile validation.
3. Secure login and password recovery.
4. Drag-and-drop product upload interface.
5. Seller dashboard for product management and order tracking.
6. Unique product URLs for sharing.
7. Inventory management for sellers.
8. Seamless browsing and purchasing for customers.

2.1.2.2 Functional Requirements:

1. Product upload and management for sellers.
2. Shareable product links.
3. Sorting/filtering by category, price, and location.
4. Order confirmation and delivery updates.
5. Seller notifications for purchases.
6. Password reset and profile updates.
7. Admin control for managing users and products.

2.1.2.3 Non-Functional Requirements:

1. **Availability**: MySQL for fast data access.
2. **Reliability**: Secure and reliable system performance.
3. **Security**: Encrypted data and HTTPS protection.
4. **Maintainability**: Structured data storage.
5. **Usability**: Easy-to-use interface for all users.

2.1.2.4 Problems in Existing Alternatives

1. Complex setup and testing of manual projects.
2. Lack of direct purchasing features.
3. Difficulty managing multiple projects.
4. Costly and time-consuming separate hosting.

2.2 Software and Hardware Requirements

2.2.1 Software Requirements

- **AptanaStudio:**
  Used as the Integrated Development Environment (IDE) for writing HTML, CSS, JavaScript, and PHP code efficiently.
- **XAMPP:**
  Provides a local server environment including Apache (web server), MySQL (database), and PHP for development and testing.
- **PHP:**
  Server-side scripting language used to create dynamic content and manage backend operations.
- **MySQL:**
  Relational database system used for storing and retrieving user and product data.

2.2.2 User Interface (UI) Requirements

- **User-Friendly Design:**
  Easy navigation and intuitive layout for both customers and admin users.
- **Responsive UI:**
  Works seamlessly across desktops, tablets, and mobile devices.
- **Optimized Fonts and Buttons:**
  Clearly visible and accessible controls to enhance usability.
- **Fast Loading:**
  Pages designed to load quickly with minimal delays to improve user experience.

2.2.3 Hardware Requirements

Operating Environment

- **Operating System:**
  Windows XP / 7 / 8 / 8.1 (32 or 64-bit)
- **Processor:**
  Minimum 1.5 GHz required for smooth performance.
- **RAM:**
  - Minimum: 512 MB
  - Recommended: 1 GB or more for better multitasking and performance.
- **Hard Disk Space:**
  - At least 2 GB free space for application files, images, and uploaded PDFs.

Browser Compatibility

- **Supported Browsers:**
  Google Chrome, Mozilla Firefox, Microsoft Edge, or similar modern browsers for fast and secure browsing.

2.3 Project Plan & Gantt Chart

This section outlines the project's timeline, key tasks, and milestones. A Gantt Chart is used to visually represent the schedule and task dependencies.

Task Name	Duration	Start	Finish	Predecessors
Project topic deciding	3 days	07/02/2025	11/02/2025	
Collecting data	8 days	12/02/2025	21/02/2025	1
Problem definition	19 days	24/02/2025	20/03/2025	2
System design	13 days	21/03/2025	08/04/2025	3
Problem evaluation	4 days	09/04/2025	14/04/2025	4
Define function and behavior	11 days	15/04/2025	28/04/2025	5
Requirement analysis	17 days	29/04/2025	21/05/2025	6
Implementation and coding	23 days	22/05/2025	24/06/2025	7
Unit testing	4 days	25/06/2025	30/06/2025	8
Integration and validation	7 days	01/07/2025	09/07/2025	9
System Testing	6 days	10/07/2025	17/07/2025	10

Table 1 Project Plan

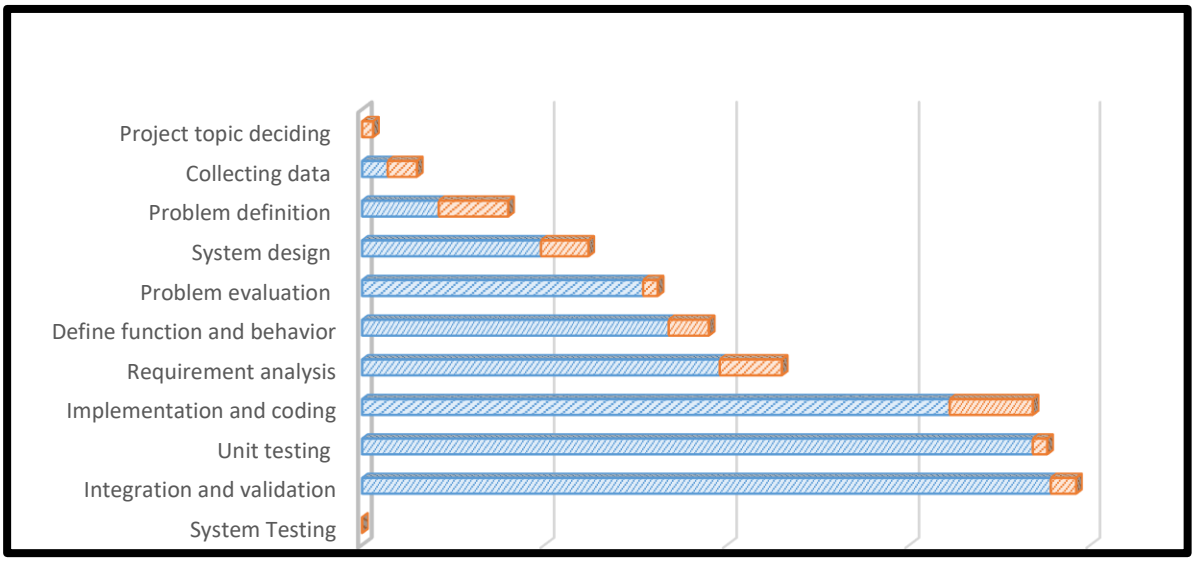


Fig2.2 Gantt chart

CHAPTER 3

LITERATURE REVIEWS

3.1 Privacy and Security Issues

Overview:

- The primary concern of online shoppers is privacy and security. Despite the growing popularity of e-commerce platforms like Amazon and eBay, security remains a significant challenge. Even reputable platforms still face frequent security threats. Customers' personal and payment information can be vulnerable to hackers, as many websites don't protect data stored in their databases or use inadequate encryption methods.

Customer Perception:

- A significant percentage of consumers are concerned about their personal information being compromised. Studies indicate that privacy and security risks account for over 65% of customers' concerns when shopping online.

Security Measures:

- While large e-commerce platforms may use encryption during data transfer (SSL, VeriSign), they are still vulnerable to attacks. Data stored on servers is at risk if it's not adequately protected. Thus, even when shopping from trusted retailers, it's essential for customers to ensure their data is protected.
- Consumers are advised to avoid sharing sensitive information, like credit card details, via email and always check for security protocols (SSL certificates) when making online payments.

3.2. Processes in the E-Commerce System:

3.2.1. Sign In Process

3.2.2. Sign Up Process

3.2.3. Shopping Cart Process

3.2.4. Purchase Process

3.2.5. Order Process

3.2.6. Checkout Process

3.2.7. Admin Process

3.2.8. Confirmation Process

3.2.1 Sign In Process: The user enters their username and password. If the login credentials are correct, the system grants access to the user's account. If the credentials are incorrect, an error message is displayed.

Figure 3.2.1 Sign in process

- *Security:* Password hashing and session management ensure secure handling.

3.2.2 Sign Up Process: Users need to provide necessary details (like name, email, address, etc.) to register an account. Once the information is submitted, the system saves it and provides a confirmation for successful sign-up.

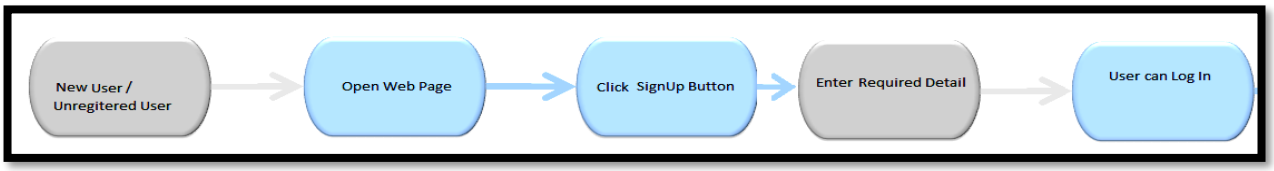


Figure 3.2.2 Sign up process

- *Security:* Secure data validation and storage prevent unauthorized access.

3.2.3 Shopping Cart Process: The shopping cart holds selected items. Users can add or remove products and adjust quantities. The cart is updated dynamically when changes are made.

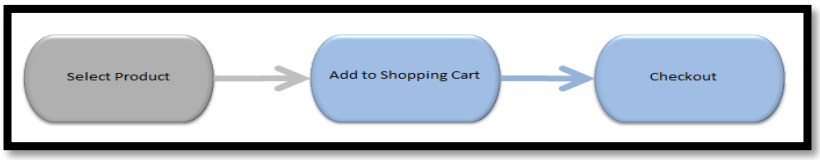


Figure 3.2.3 Shopping cart process

- *Security:* Cart data is stored session-wise, ensuring secure updates.

3.2.4 Checkout Process: After finalizing the cart, users enter shipping and payment info to complete the transaction, receiving a confirmation email.

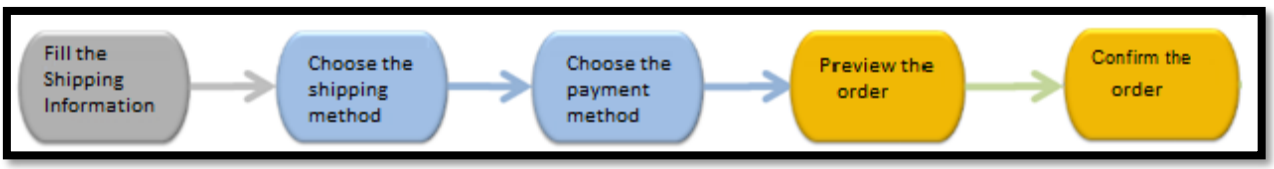


Figure 3.2.3 Checkout process

- *Security:* Payment details are encrypted.

3.2.5 Purchase Process: Users browse the product catalog and add desired products to their shopping cart. They can modify the quantity and proceed to checkout. Once everything is set, users complete the purchase.

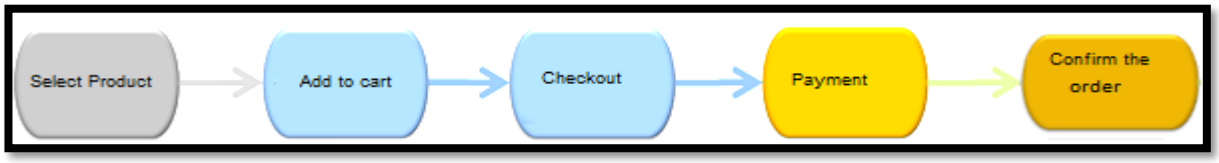


Figure 3.2.4 Purchase Process

- *Security:* All transactions are securely processed.

3.2.6 Order Process: After the user checks out, they provide the required shipping details, and the order is processed. The user will then receive a confirmation email, signaling that the order has been successfully placed.

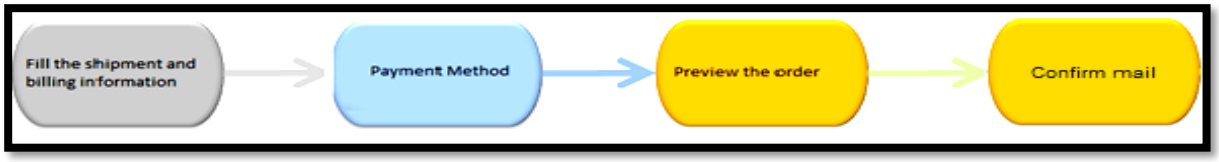


Figure 3.2.5 Order purchase

- *Security:* Order data is securely transmitted to the admin.

3.3.7 Admin Process: The admin has full access to the backend, allowing them to add, edit, and delete products, as well as manage user accounts (activate/deactivate users).

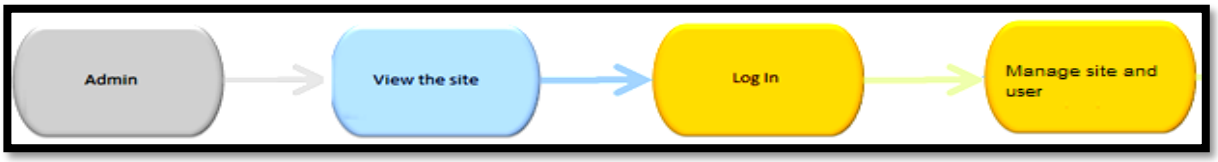


Figure 3.2.6 Admin process

- *Security:* Actions like deleting products require confirmation.

3.2.8 Confirmation Process: After completing a purchase, users receive a confirmation email detailing the order, shipping address, and expected delivery date.

Figure 3.2.7 Confirmation process

- *Security:* The email system ensures data integrity and confidentiality.

3.3 Security and Privacy Concerns:

- Customers are often wary of privacy and security issues while shopping online. Studies have shown that privacy and security concerns are one of the main reasons people hesitate to shop online.
- Security measures like SSL encryption, protecting sensitive data during transmission, are essential, but even large companies face security threats. Therefore, customers should also be proactive in safeguarding their personal information.

3.4. Waterfall Model:

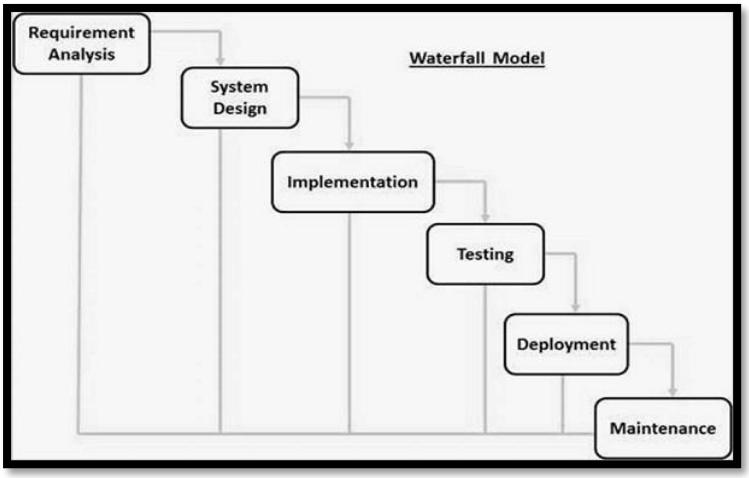


Figure 3.3.3 Waterfall Model

Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

- **Advantages:** Simple to use, clearly defined phases, works well for small projects with well-understood requirements.
- **Disadvantages:** Limited flexibility, difficult to accommodate changes during the development process, not suitable for complex or long-term projects.

3.5. System Development Life Cycle (SDLC):

Figure 3.4 SDLC diagram

3.5.1 Planning:

- Detailed planning is carried out, including defining the project scope, timeline, and deliverables. This phase also includes creating mockups, UML diagrams, and technical documentation.

3.5.2 Data Requirement & Analysis:

- In this phase, the project team gathers data and performs detailed analysis to ensure the system meets business needs and user expectations.

3.5.3 Design:

- The system's architecture is designed, keeping in mind usability, scalability, and security. Both high-level and low-level designs are documented for implementation.

3.5.4 Development:

- The system is developed using appropriate technologies (PHP, MySQL, and frameworks like CodeIgniter). HTML, CSS, and JavaScript are used to create the frontend.

3.5.5 Integrate & Testing:

- After development, the system is tested for bugs, performance, and functionality. Integration tests ensure that all components of the system work as expected.

3.5.6 Implementation:

- The final system is implemented into the production environment. The system undergoes user acceptance testing to ensure that the e-commerce platform is functional and reliable.

3.6. Case Studies:

- **Case Study 1:** Addresses the issue of users being unable to update the quantity of products in their shopping cart without pressing the update button. The solution was to replace the update button with a dropdown for quantity selection that automatically updates in the database.
- **Case Study 2:** Details how the shopping cart's quantity updating process was improved to allow users to modify quantities directly and instantly without the need to click an update button.

CHAPTER 4

SOFTWARE & TOOL REQUIREMENT ANALYSIS

4.1 Tools and Techniques

The development of this e-commerce project utilized a variety of tools and technologies, each contributing uniquely to the system's frontend, backend, and database management functionalities.

1. PHP

PHP (Hypertext Preprocessor) is a widely-used open-source server-side scripting language especially suited for web development. Created by Rasmus Lerdorf in 1994, it allows embedding code within HTML or in combination with various web frameworks and content management systems. PHP scripts are executed on the server, generating dynamic page content that can include text, images, and interactive forms. It can also be used from the command line for automation and backend logic.

2. XAMPP

XAMPP is a free, open-source cross-platform web server solution developed by Apache Friends. It includes:

- **Apache HTTP Server**
- **MariaDB** (a fork of MySQL)
- **Interpreters for PHP and Perl**

XAMPP simplifies the process of creating a local server for testing and development, making it a crucial tool for PHP developers. Its cross-platform nature ensures compatibility across Windows, Linux, and macOS.

3. MySQL Workbench (MySQL Yog)

MySQL Workbench is a visual database design tool that provides:

- Data modeling
- SQL development
- Server configuration
- User and security management
  It simplifies database operations and supports cross-platform usage (Windows, Linux, macOS), making it ideal for managing MySQL-based applications.

4. HTML (HyperText Markup Language)

HTML is the standard markup language used to create and structure web pages. It forms the skeleton of a web application, allowing for the inclusion of:

- Headings
- Paragraphs
- Lists
- Forms
- Embedded media
  It works in conjunction with CSS and JavaScript to deliver responsive and interactive web experiences.

5. Bootstrap

Bootstrap is a powerful front-end framework that includes ready-made HTML, CSS, and JavaScript components. It enables developers to quickly design responsive, mobile-first web interfaces, offering pre-designed:

- Buttons
- Forms
- Navigation bars
- Modals
- Grids
  It enhances development speed and ensures design consistency.

6. Sublime Text

Sublime Text is a popular cross-platform code editor known for its performance and simplicity. It supports numerous programming and markup languages out of the box, and additional functionality can be added via community plugins. Key features include:

- Syntax highlighting
- Auto-completion
- Split editing
- Distraction-free mode

7. GitHub

GitHub is a cloud-based platform for hosting and version control using **Git**. It provides:

- Source code management (SCM)
- Collaboration tools
- Issue tracking
- Project wikis
  GitHub is widely used for both private and open-source projects, enabling team collaboration and code version history.

8. JavaScript

JavaScript (JS) is a dynamic, high-level scripting language that allows developers to implement interactive features on web pages. It is one of the core technologies of the web, alongside HTML and CSS. JavaScript enables:

- Form validation
- Dynamic content updates
- Interactive UI components
  All modern web browsers have built-in JavaScript engines for executing code.

9. CSS (Cascading Style Sheets)

CSS is used for defining the visual presentation of HTML elements. It allows for:

- Styling fonts, colors, and layouts
- Creating responsive designs
- Separating content from design
  CSS enhances maintainability and enables consistent styling across multiple pages by using reusable stylesheets.

4.3 CASE STUDIES OF SOFTWARE ANALYSIS

4.3.1 Case Study: General Shopping Process

- A customer visits the shopping portal and browses by category/brand.
- They add items to the cart.
- Final cart can be reviewed and edited.
- To proceed with payment, users must log in or register.

- Once logged in, the user confirms details and completes the payment.

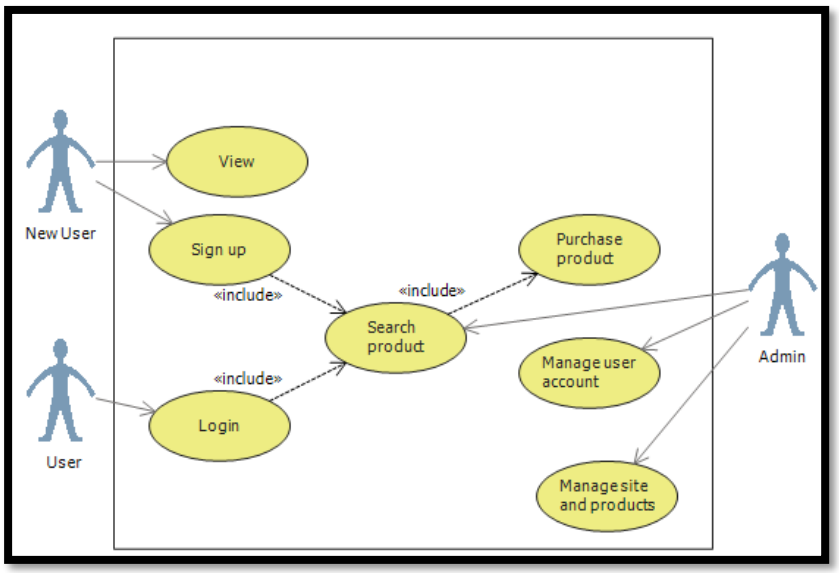


Figure 4.3.1 – Shopping Portal

4.3.2 Case Study: User

- Users explore products, add items to their cart, and proceed to checkout.
- Users are authenticated before final payment.
- Users can modify the cart at any time.

4.3.3 Case Study: Admin

- Admin oversees website functionality and user management.
- Admin can update product details and categories.

- Can block/unblock users and manage their activity.

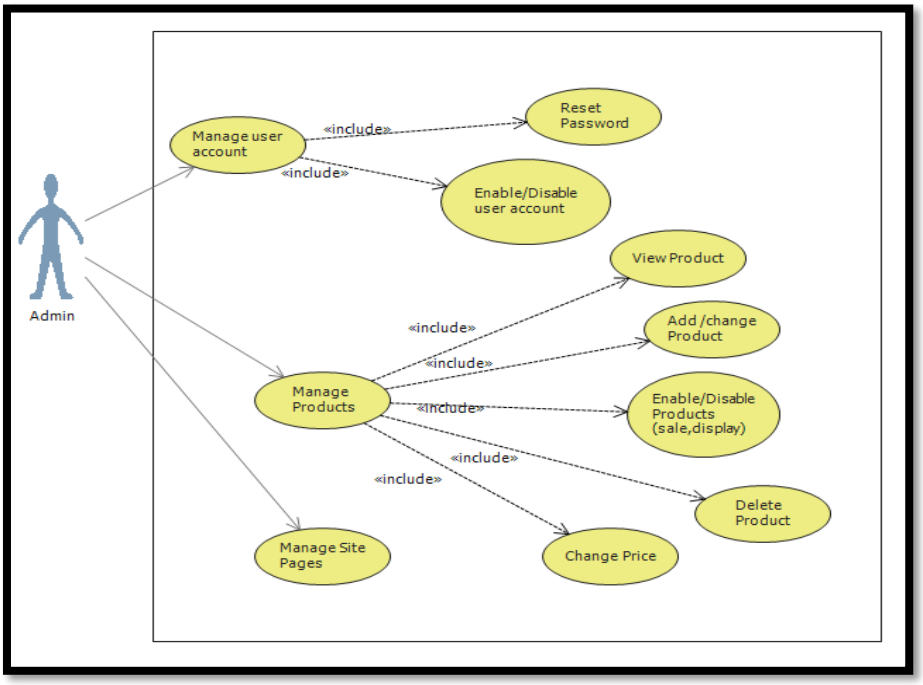


Figure 4.3.3 – Admin Panel

4.3.4 Case Study: Shopping Cart

- Users add products to the cart as they shop.
- Before checkout, cart items and quantities can be adjusted.

4.3.4.1 Problem During Development

- Quantity was not updating automatically, which caused incorrect totals in the cart.

4.3.4.2 Working of Shopping Cart

- Shopping cart maintains session-wise selected items.
- Dropdown for quantity auto-updates database without pressing an update button.

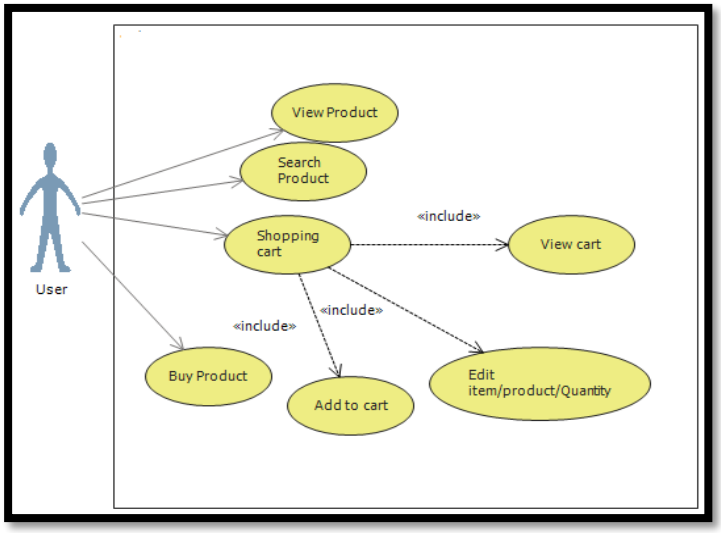


Figure 4.3.4 – Shopping Cart

4.3.5 Case Study: Sign In and Sign Up

- Users enter credentials on the login screen.
- System validates credentials and grants access.
- New users fill a registration form and get registered.

4.3.5.1 Problem During Development

- After signup, users had to log in manually again.
- Bug was fixed so users are auto-logged in post-registration.

4.3.5.2 Working of Sign In & Sign Up

- Secure login mechanism using session control.
- Auto-login after successful signup.

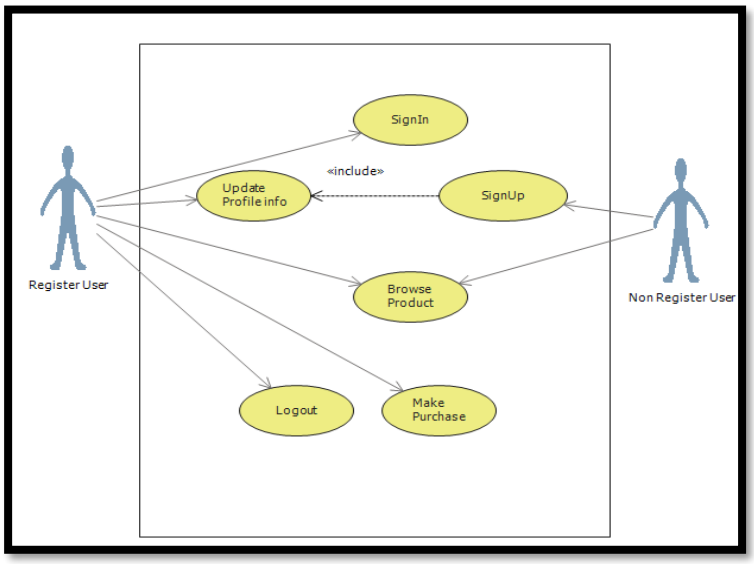


Figure 4.3.5 – Sign In and Sign Up

4.3.7 Case Study: Product Catalogue & Details

- Users can browse products via categories or directly from the homepage.
- Product detail pages are accessible from any page.

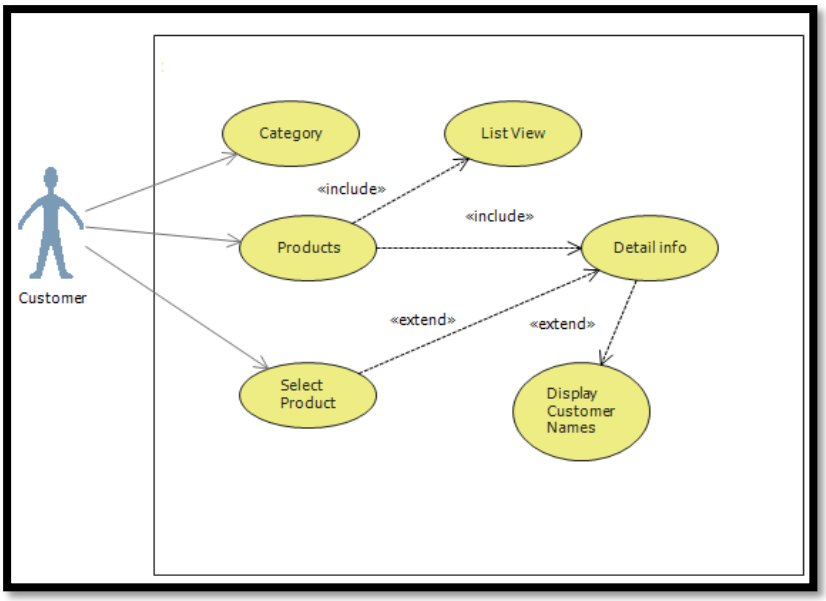


Figure 4.3.7 – Product Catalogue & Details

CHAPTER 5

SOFTWARE DESIGN

5.1 Use Case Diagram

Illustrates the interactions between users and the system, covering product browsing, order placement, payment processing, and order tracking.

- Actors: **Customers, Farmers, Admin**.

Use Cases: **Browse Products, Place Order, Make Payment, Track Order, Manage Inventory**

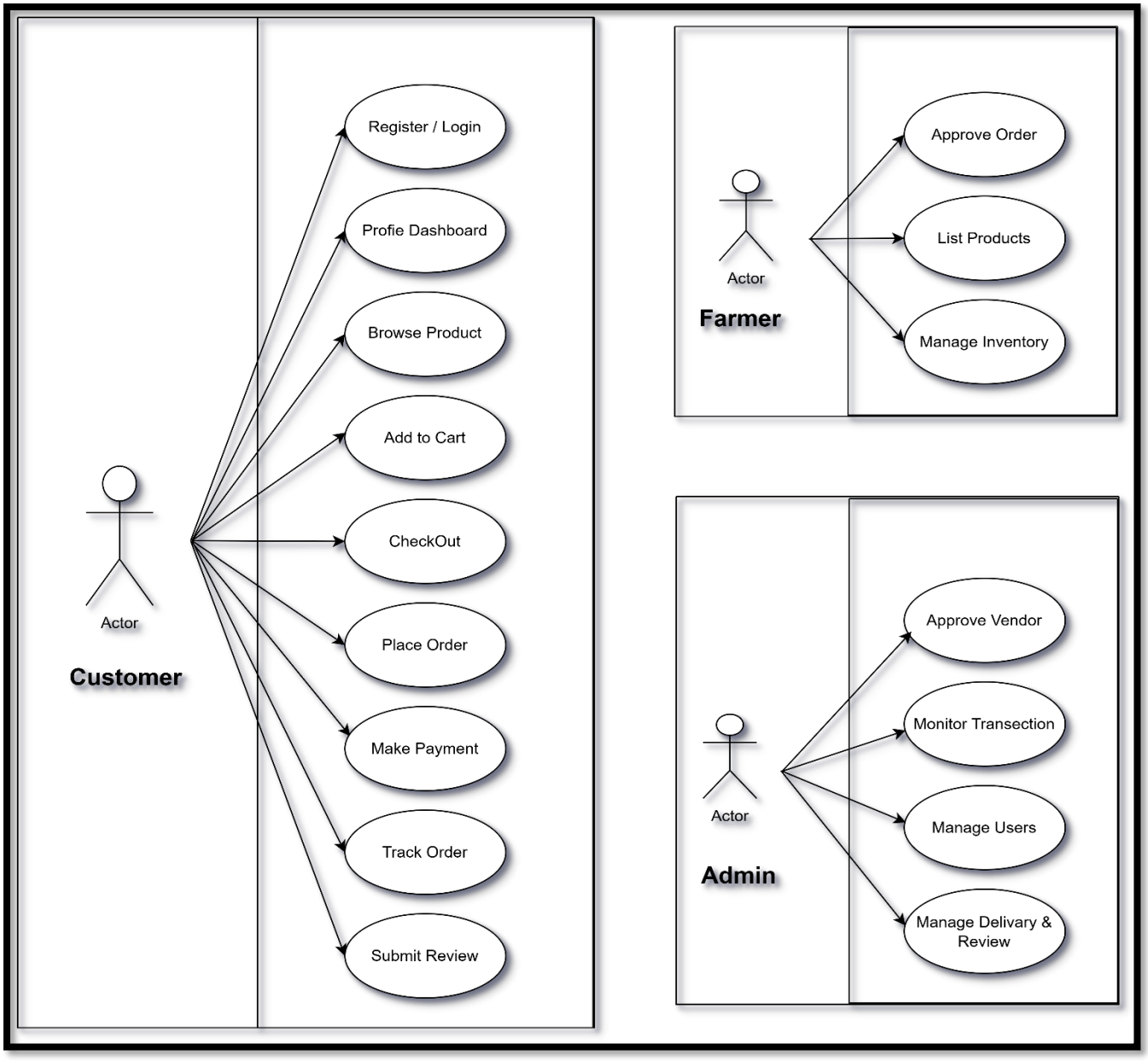


Fig. 5.1 Use Case Diagram

5.2 System Architecture Diagram

A system architecture diagram is an alternate way to show s scenario. This type of diagram shows object interactions organized around the objects and their links to each other.

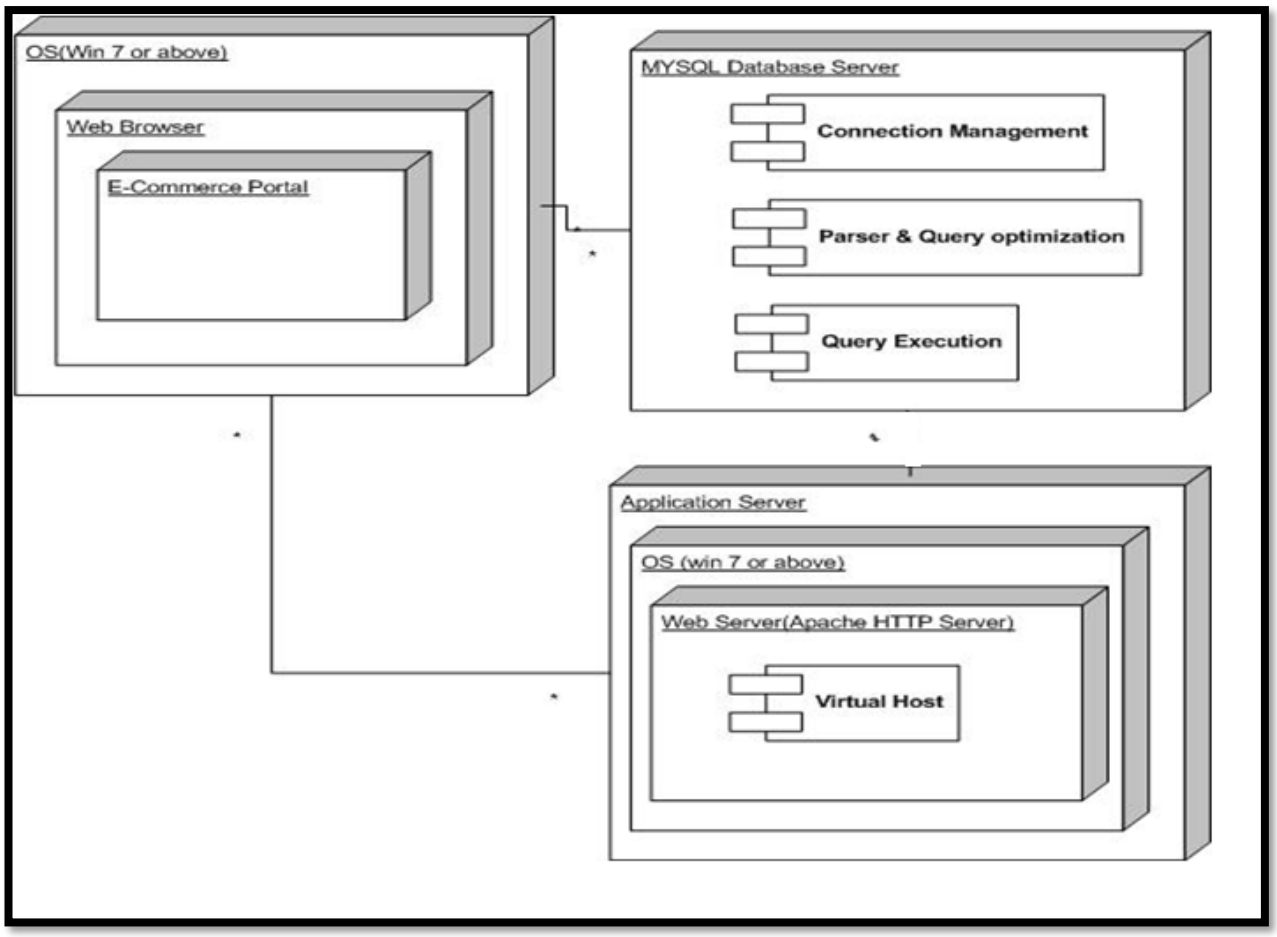


Fig. 5.2 System Architecture Diagram

In Fig. 5.1, it shows the object interaction in how they linked to each other manner, first user provides its username and password then its verified by the server and after that user can able to do shopping

5.3 DATA FLOW DIAGRAM (DFD)

Data Flow Diagram (DFD) is a graphical representation of a data flow in a system.

5.3.1 '0' level DFD

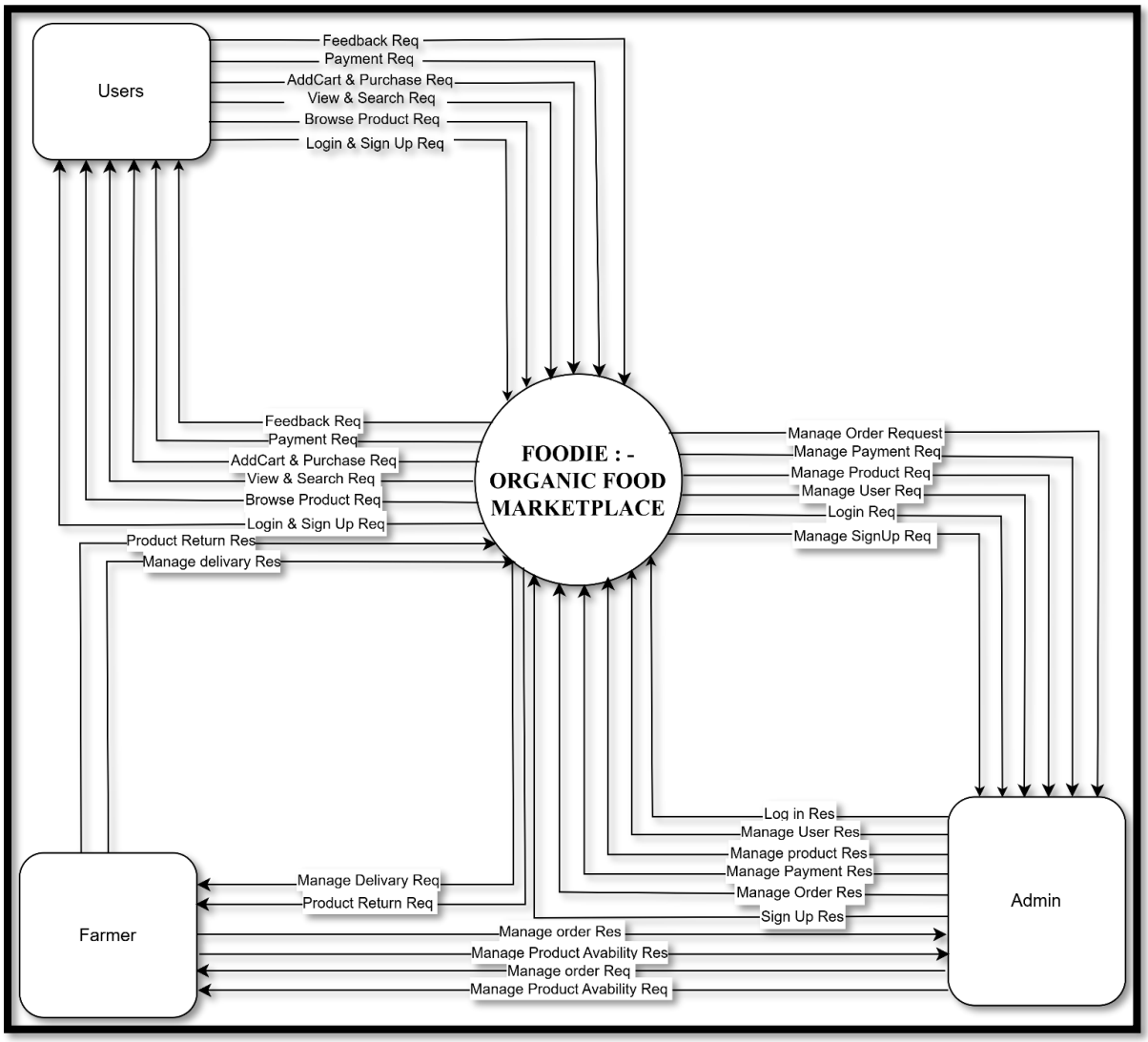


Fig. 5.3.1 '0' level Data Flow Diagram

In Fig. 5.3.1, Depicts the flow of data from user inputs to processing and output, ensuring transparency in transactions.

User inputs → System processes order → Farmer receives order → Delivery tracking.

5.3.2.1 '1' level DFD

5.3.2.1 '1' level DFD Admin Panel

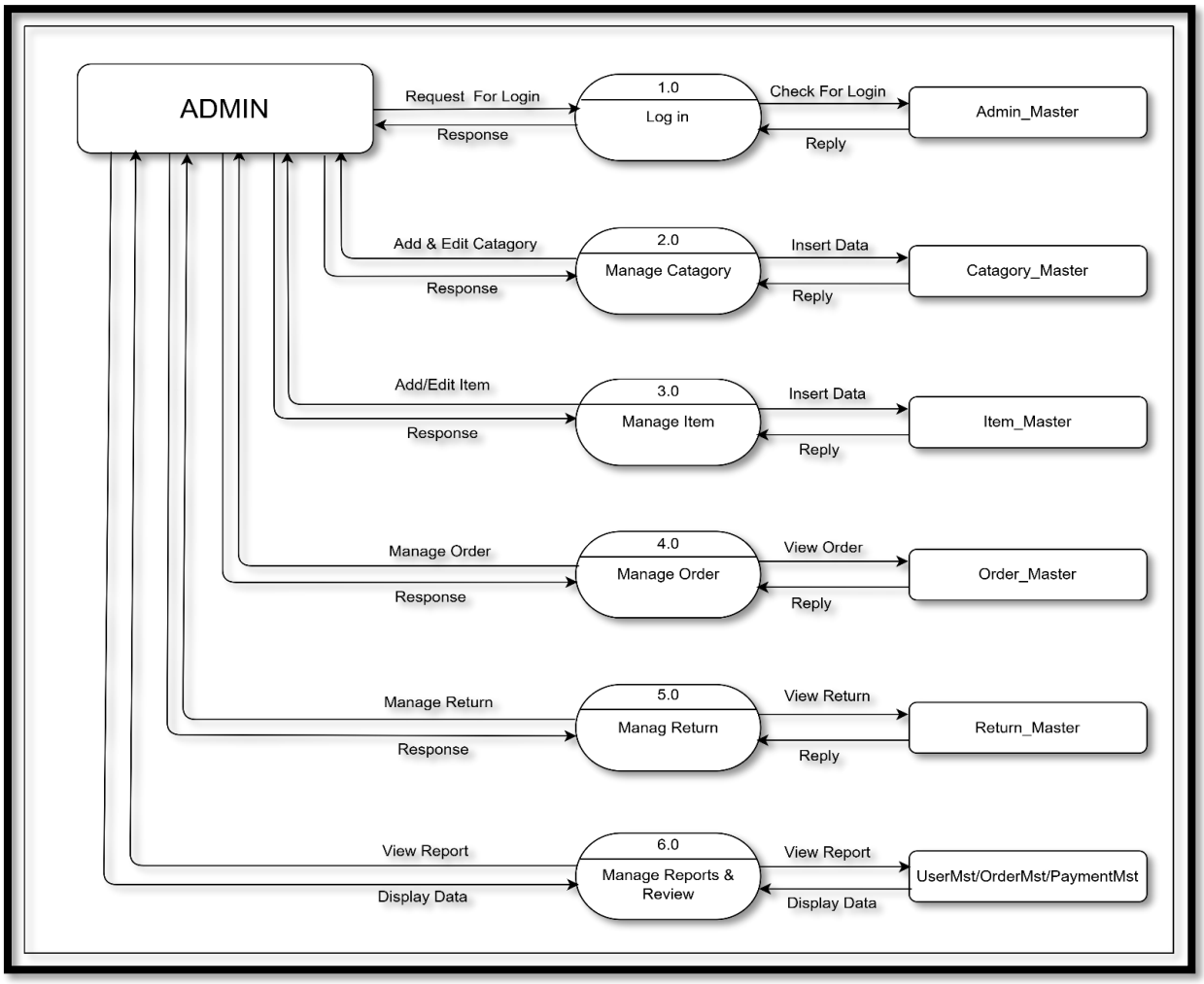


Fig. 5.3.1 '1' Data Flow Diagram Admin Panel

In Fig. 5.3.2.1, it shows the data flow in system, first user login into the system after that he/she has two options (i)Admin can manage site

5.3.2.2 '1' level DFD User Panel

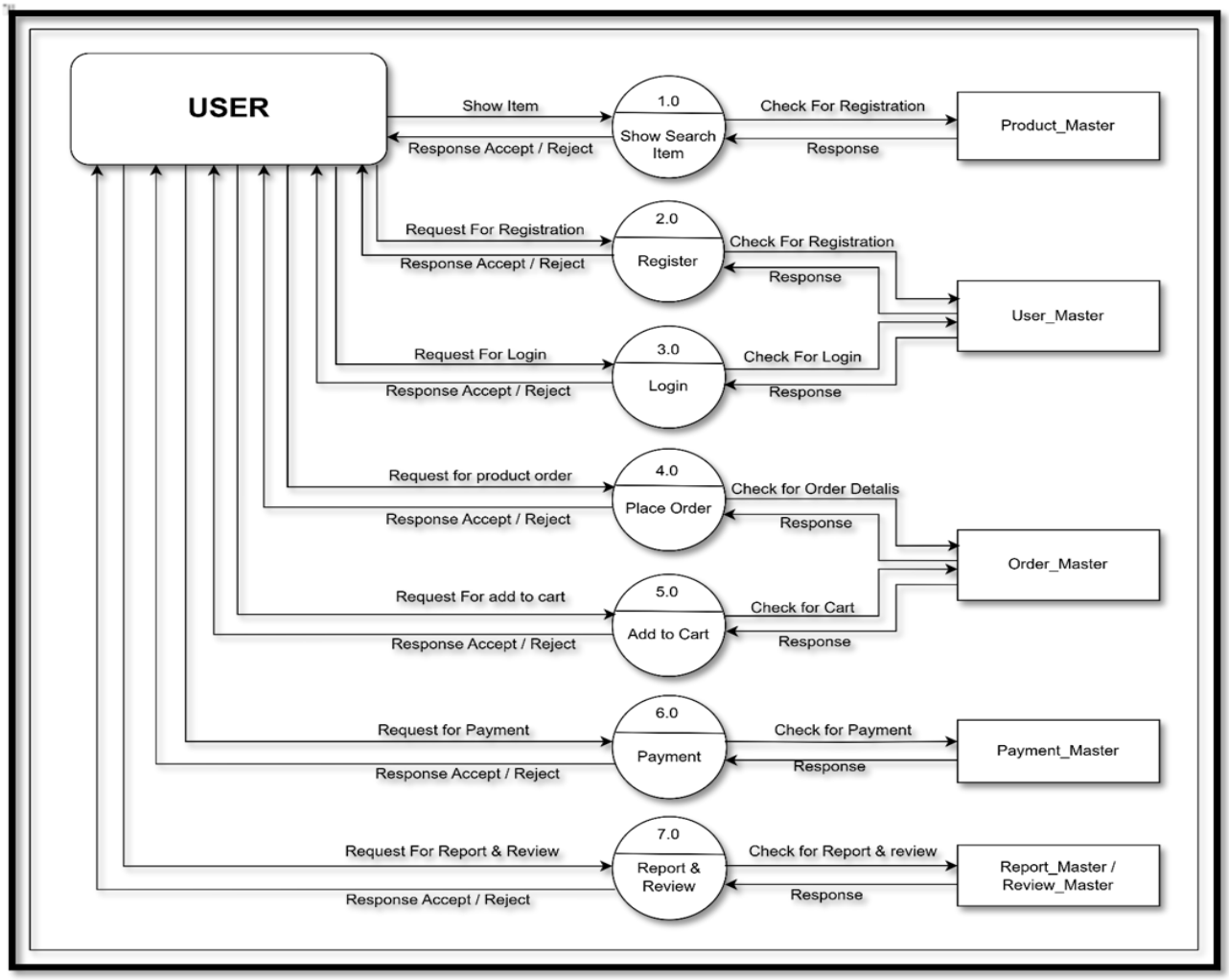


Fig. 5.3.2 '1' Data Flow Diagram User Panel

In Fig. 5.3.2.2, it shows the data flow in system, first user login into the system after that he/she has two options (i)User has to login or signup to do shopping

5.4 CLASS DIAGRAM

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed

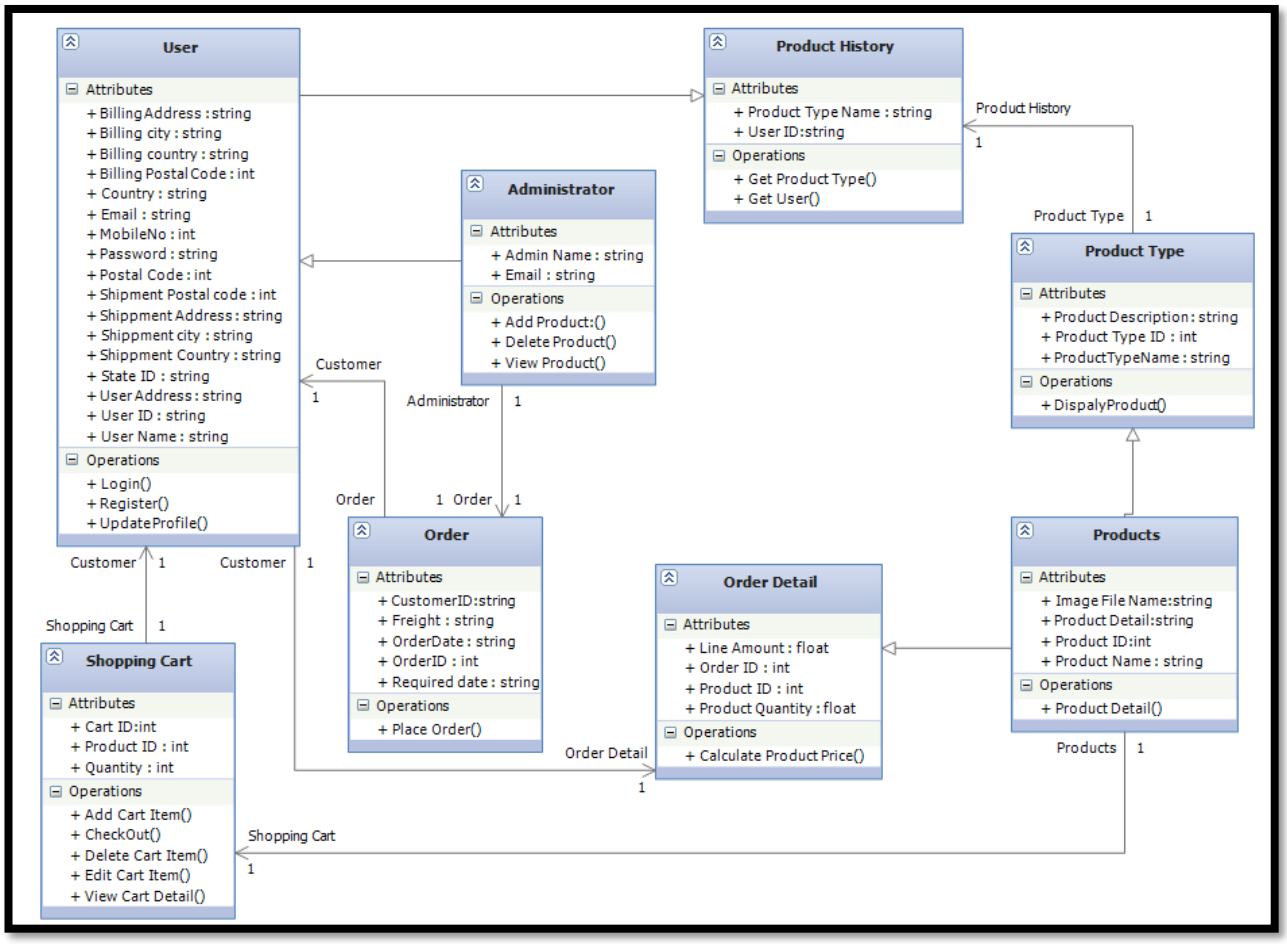


Fig. 5.4 Class Diagram

In Fig. 5.4, it shows six classes of the system

(i) User class which is responsible to add user, update user record, delete user record and change user password.
(ii) Administrator class which is responsible to deactivate user account and reset user password.
(iii) Product history which is responsible for user's product history.
(iv) Shopping cart class which is responsible to add and delete the product.
(v) Order class is used place order of customer
(vi) Order detail class is responsible for detail of products

5.5 OBJECT DIAGRAM

Object diagram shows a snapshot of instances of things in class diagrams. Similar to class diagrams, object diagrams show the static design of system but from the real or prototypical perspective.

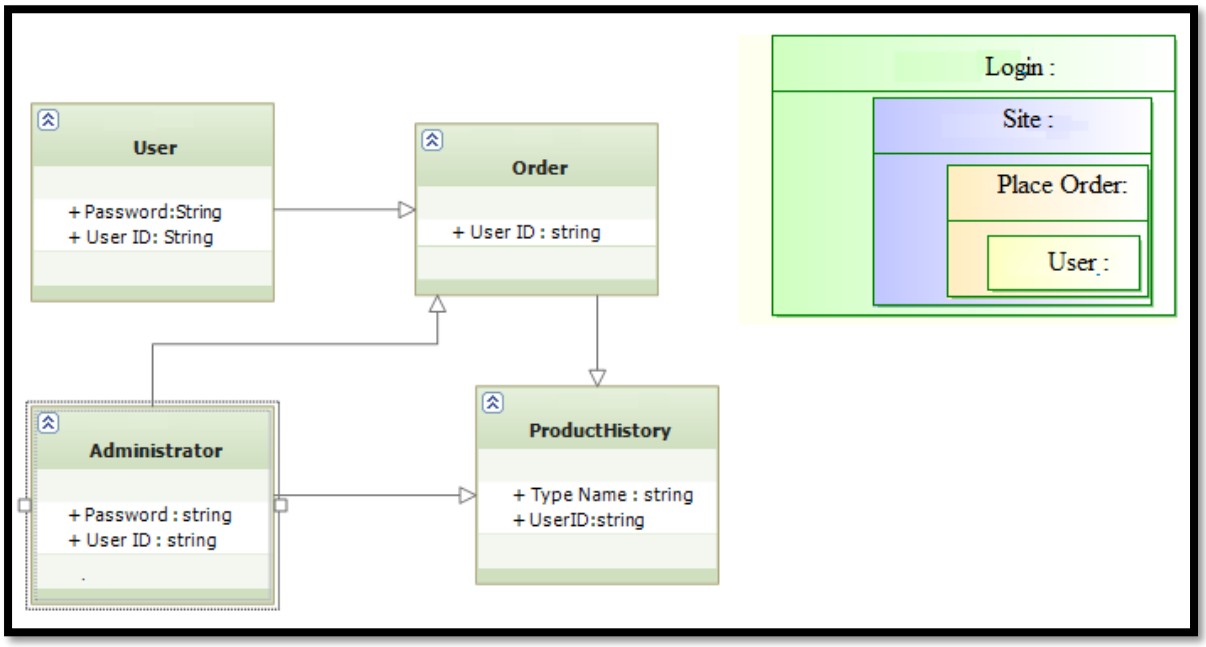


Fig. 5.5 Object Diagram

Fig 5.5 shows the static design of system but from the real or prototypical perspectives

5.6 DEPLOYMENT DIAGRAM

The Deployment Diagram also helps to model the physical aspect of an Object-Oriented software system. It models the run-time configuration in a static view and visualizes the distribution of components in an application

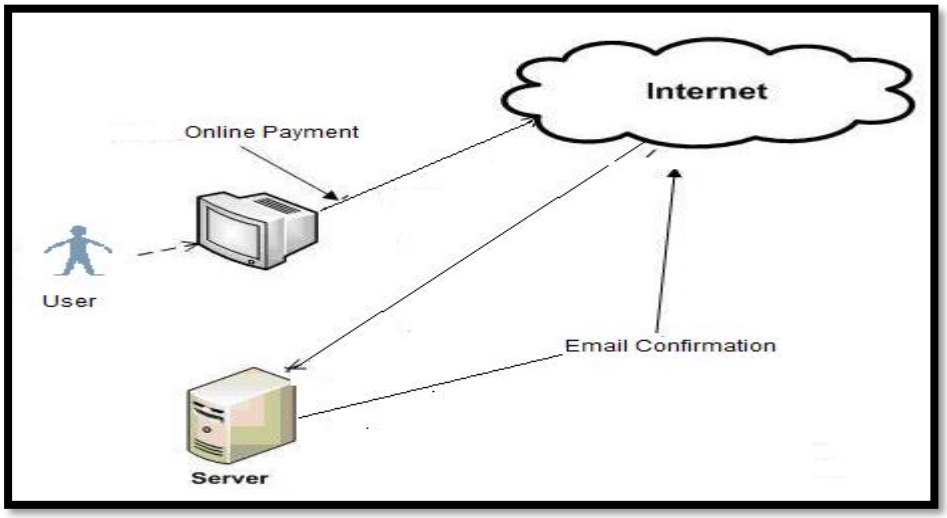


Fig. 5.6 Deployment Diagram

5.7 USE CASE

A Use-case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. Use-case diagram contains the actor and the use case symbols, along with connection lines. Actors are similar to external entities; they exist outside the system. The term actor refers to a particular role of a user of the system. The main purpose of a use-case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

The purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors

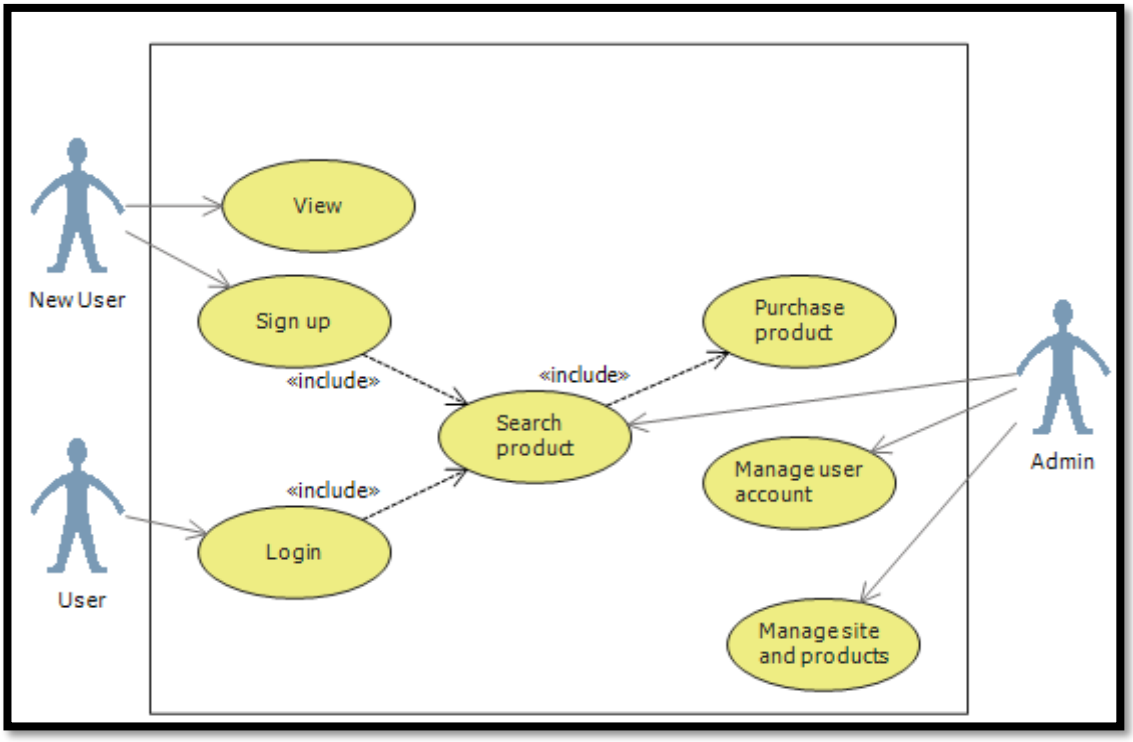


Fig.5.7 Use-Case Diagram (High Level Diagram)

In fig 5.7 shows (i) New customer can search products, register account then purchase product (ii) Normal customer can simply login then purchase product (iii) Admin can manage site and manage user account

5.7.1 User Management Use Case Diagram

User management use case shows that only registered user can buy products, update profile. Non registered user can only browse the products but cannot buy anything

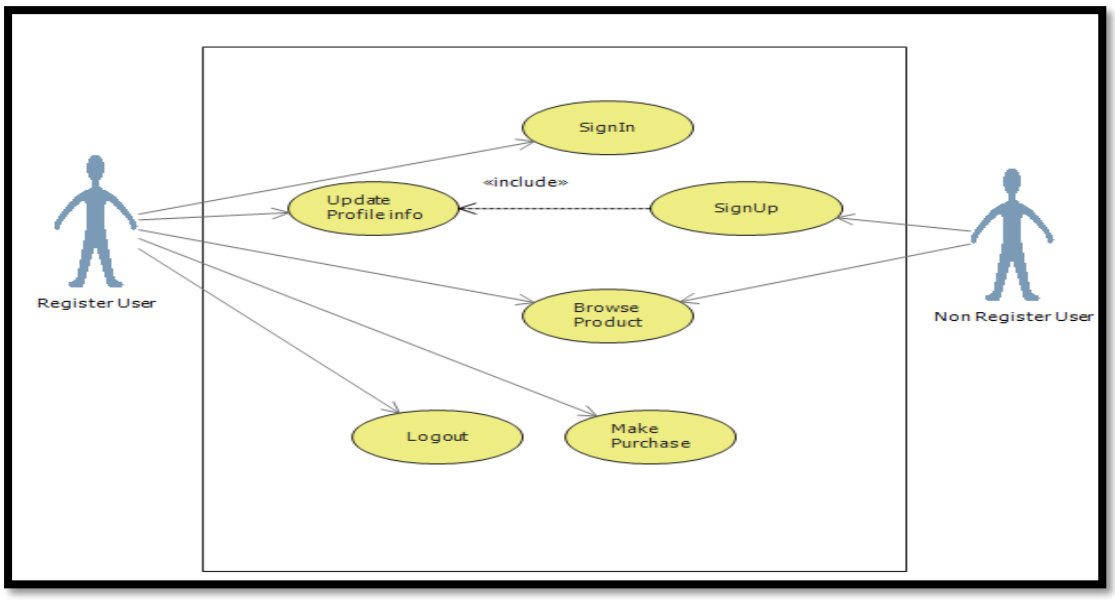


Fig.5.7.1 Use-Case Diagram (User Management Use Case Diagram)

In fig 5.7.1 shows (i) Register user can sign in, update profile and purchase product (ii) Non Register user has to sign up to purchase product.

5.7.2 Admin Use Case Diagram

Admin use case diagram shows that admin can manage user account, manage products and manage site

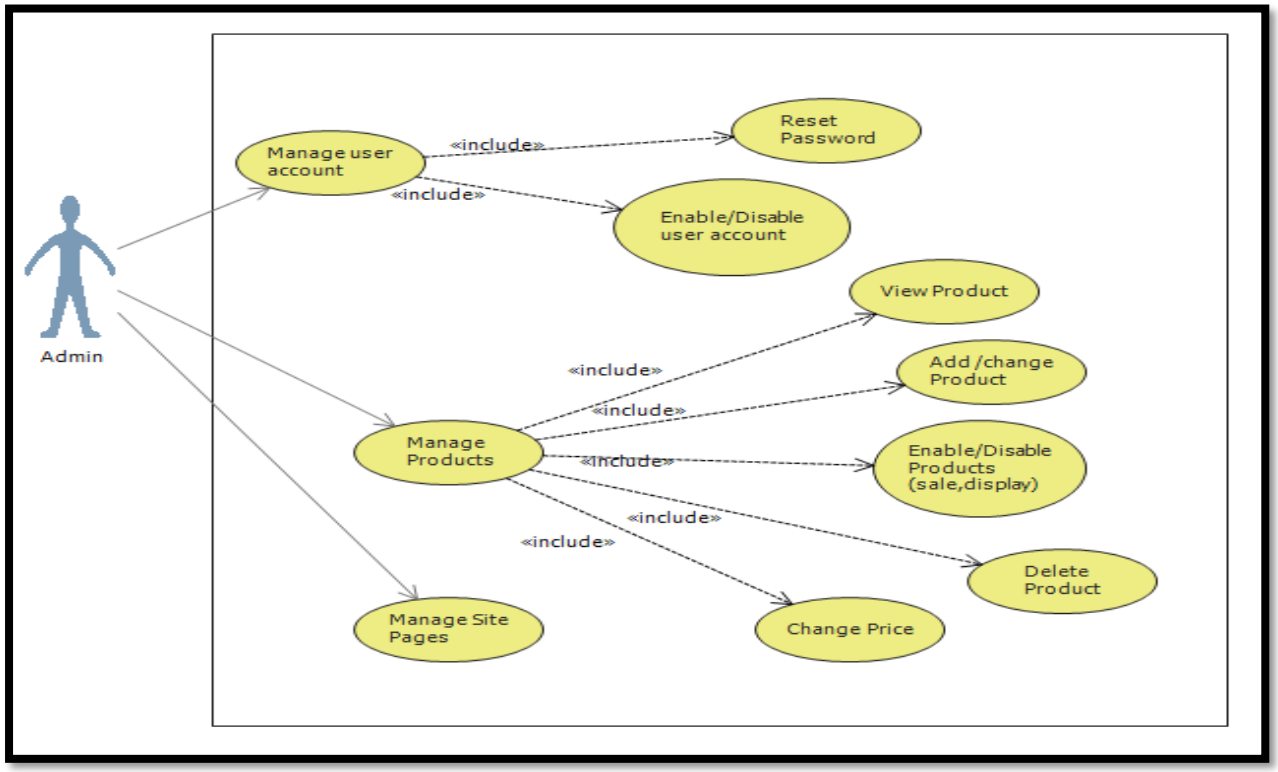


Fig.5.7.2 Use-Case Diagram (Admin Diagram)

5.8 Sequence Diagram

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. It shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

5.7.1 Non-Register user sequence diagram

In non-register user sequence diagram user has to first sign up for the site than user can purchase product

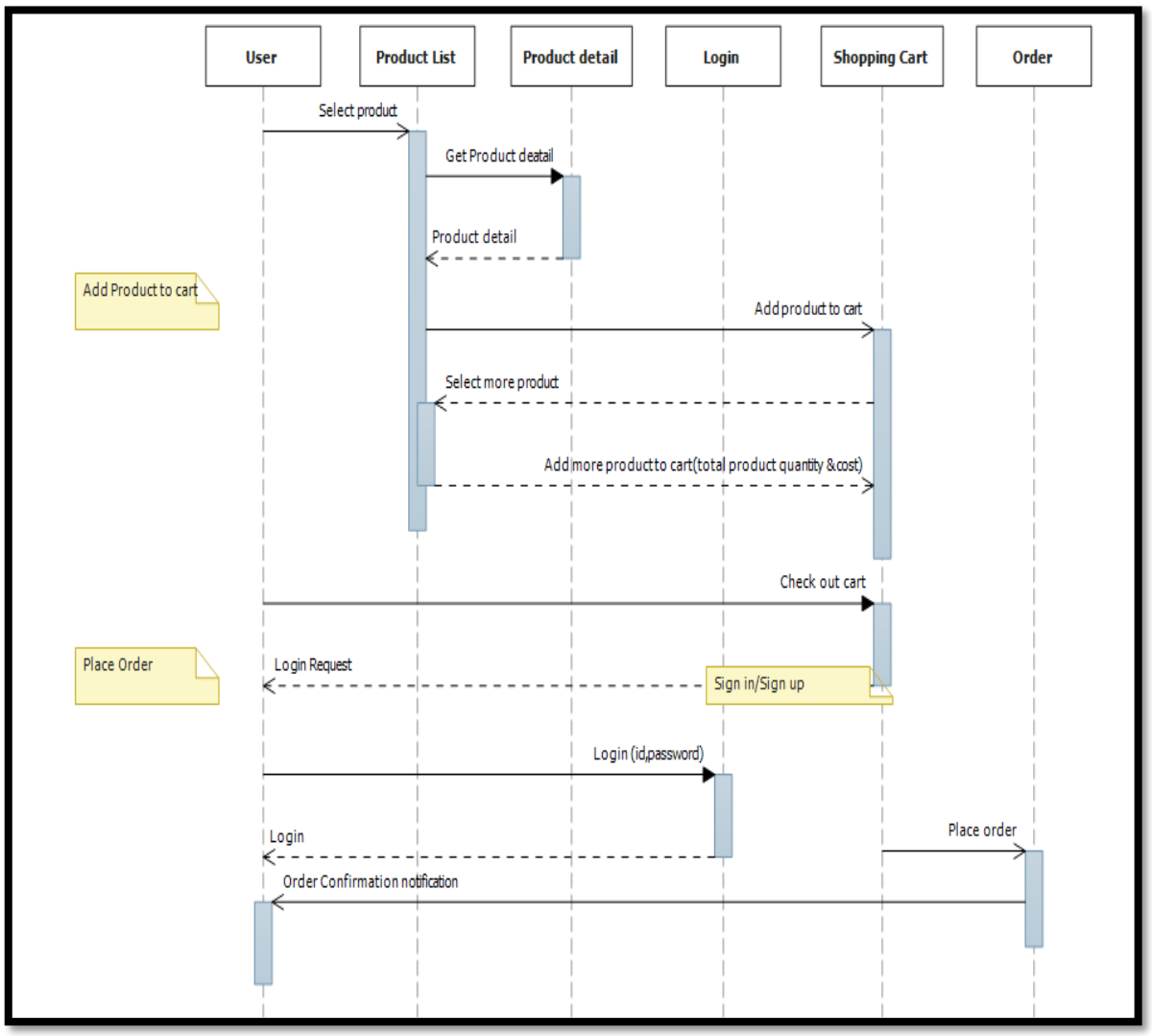


*Fig. 5.8.1 Sequence Diagram (***Non-Register user sequence diagram***)*

In Fig. 5.8.1, it shows object interactions of sequence, first new user have to sign up into the system then he/she will able to buy the product

5.8.2 Register user sequence diagram

In register user sequence diagram that user has simply log in for purchasing the product

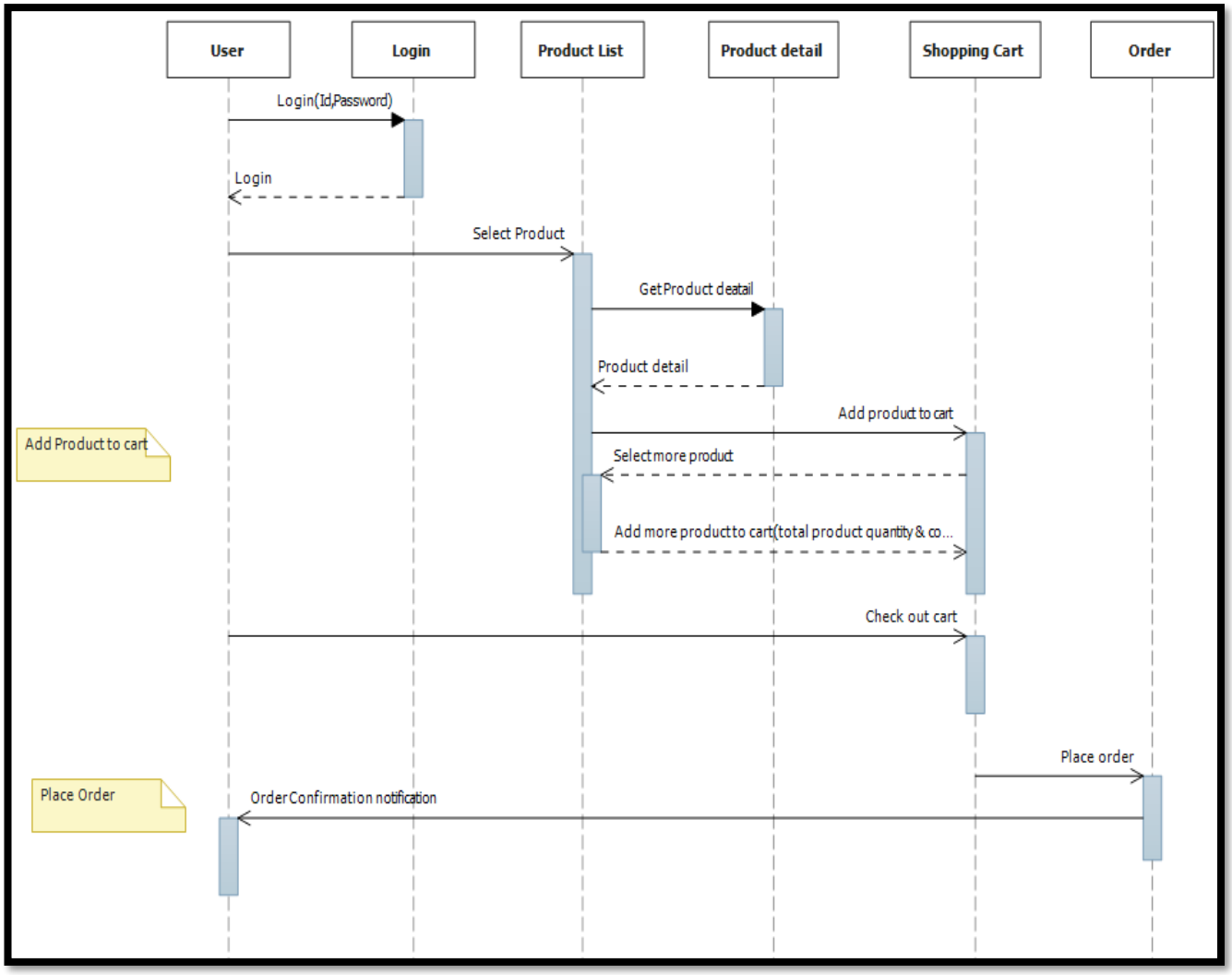


Fig. 5.8.2 Sequence Diagram (*Register user sequence diagram*)

In Fig. 5.8.2, it shows object interactions of sequence, first registered user don't have to sign up into the system user can only login then he/she will able to buy the product

5.8.3 Admin sequence diagram

In admin sequence diagram that admin can manage products, manage user account also enable or disable account.

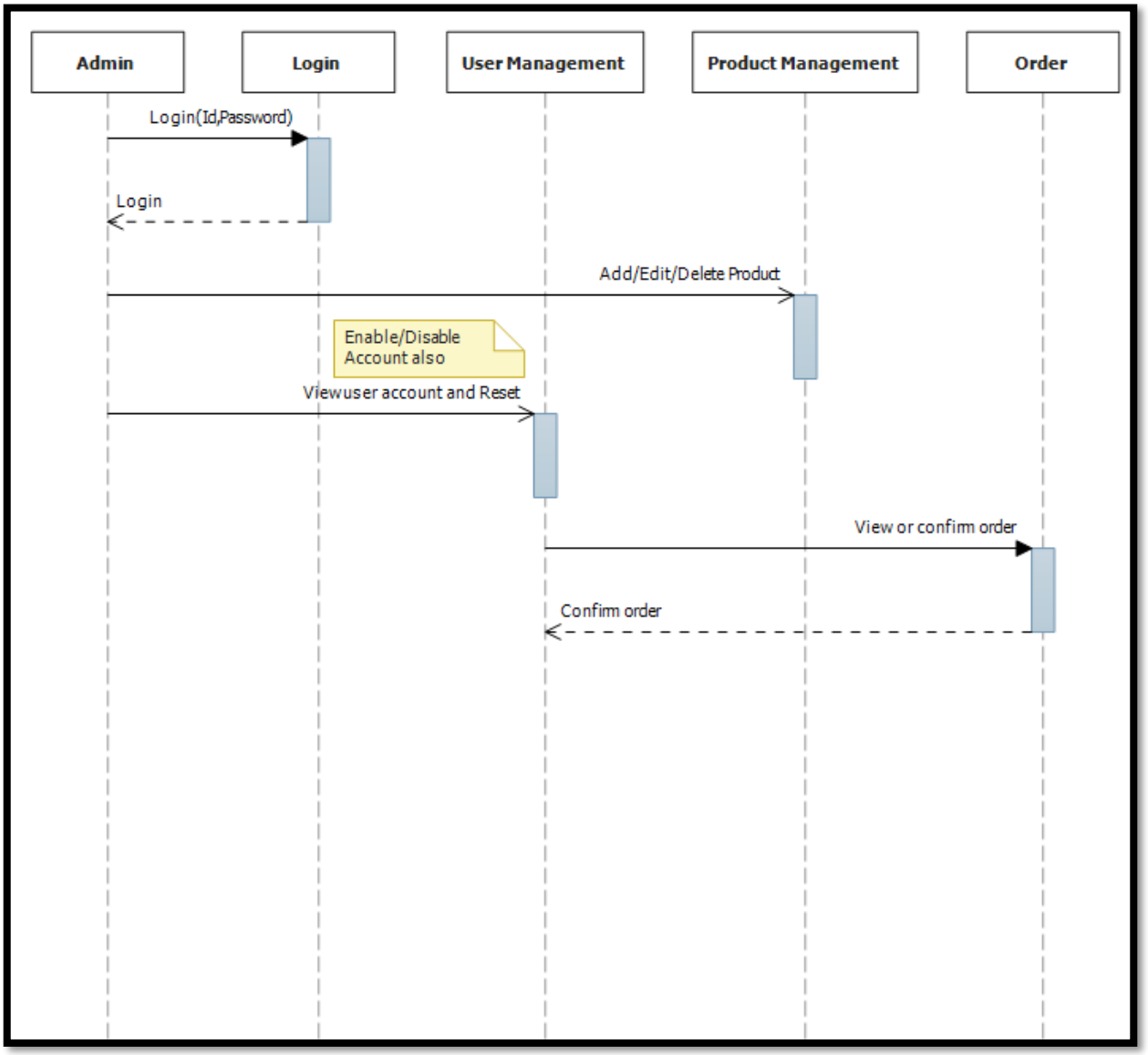


Fig. 5.8.3 Sequence Diagram (Register user sequence diagram)

In Fig. 5.7.3, it shows admin can add/edit/delete product and reset user password and confirm order of product

5.9 Activity Diagram

Activity diagrams represent workflows in a graphical way. They can be used to describe business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- Rounded rectangles represent actions;
- Diamonds represent decisions;
- Bars represent the start (split) or end (join) of concurrent activities;
- A black circle represents the start (initial state) of the workflow;
- An encircled black circle represents the end (final state).
- Arrows run from the start towards the end and represent the order in which activities happen.

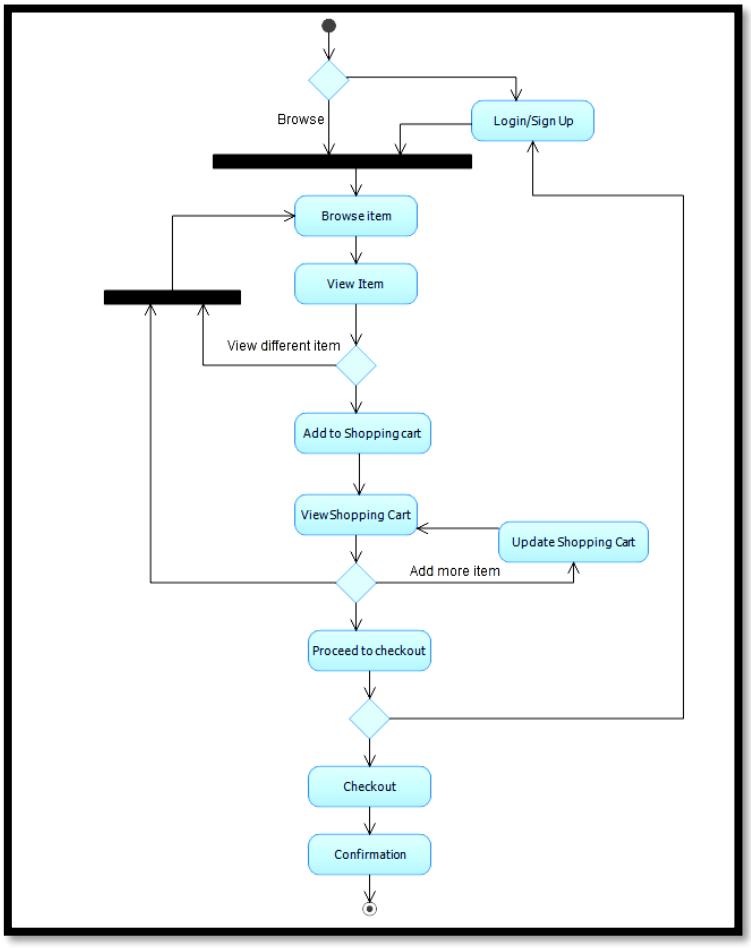


Fig. 5.9 Activity Diagram

In Fig. 5.8 This activity diagram is used to identify the user and then granting him the selected privileges**.**

5.10 FLOWCHART

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.

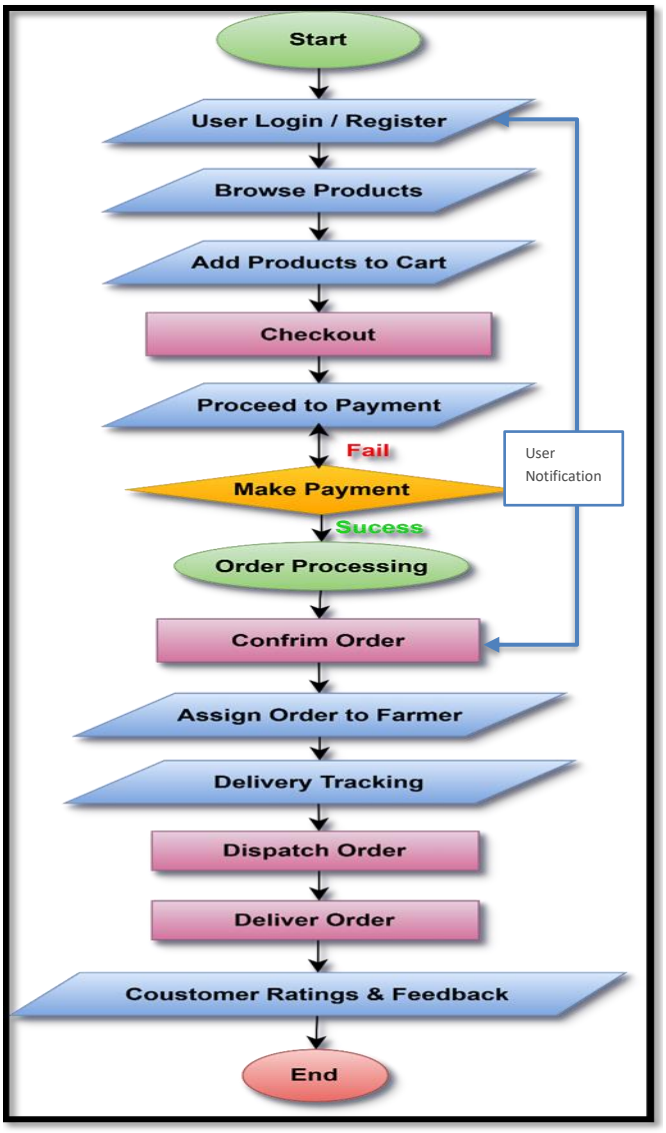


Fig. 5.10 Flowchart

In this Fig 5.9, shows the user has to sign in or sign up for the site to do the shopping

5.11 ER DIAGRAM

An entity-relationship diagram is a data modeling technique that creates a graphical representation of the entities in a system, and the relationships between entities. The main components of ER models are entities (things) and the relationships that can exist among them, and databases.

The purposes of entity relationship (E.R) diagrams can be as follows:

- Designs are pictures called entity-relationship diagram.
- Fairly mechanical ways to convert E.R diagrams to real implementations like relational databases exist
- To create an accurate reflection of the real world in a database.
- It gives us an intermediate step from which it is easy to define a database

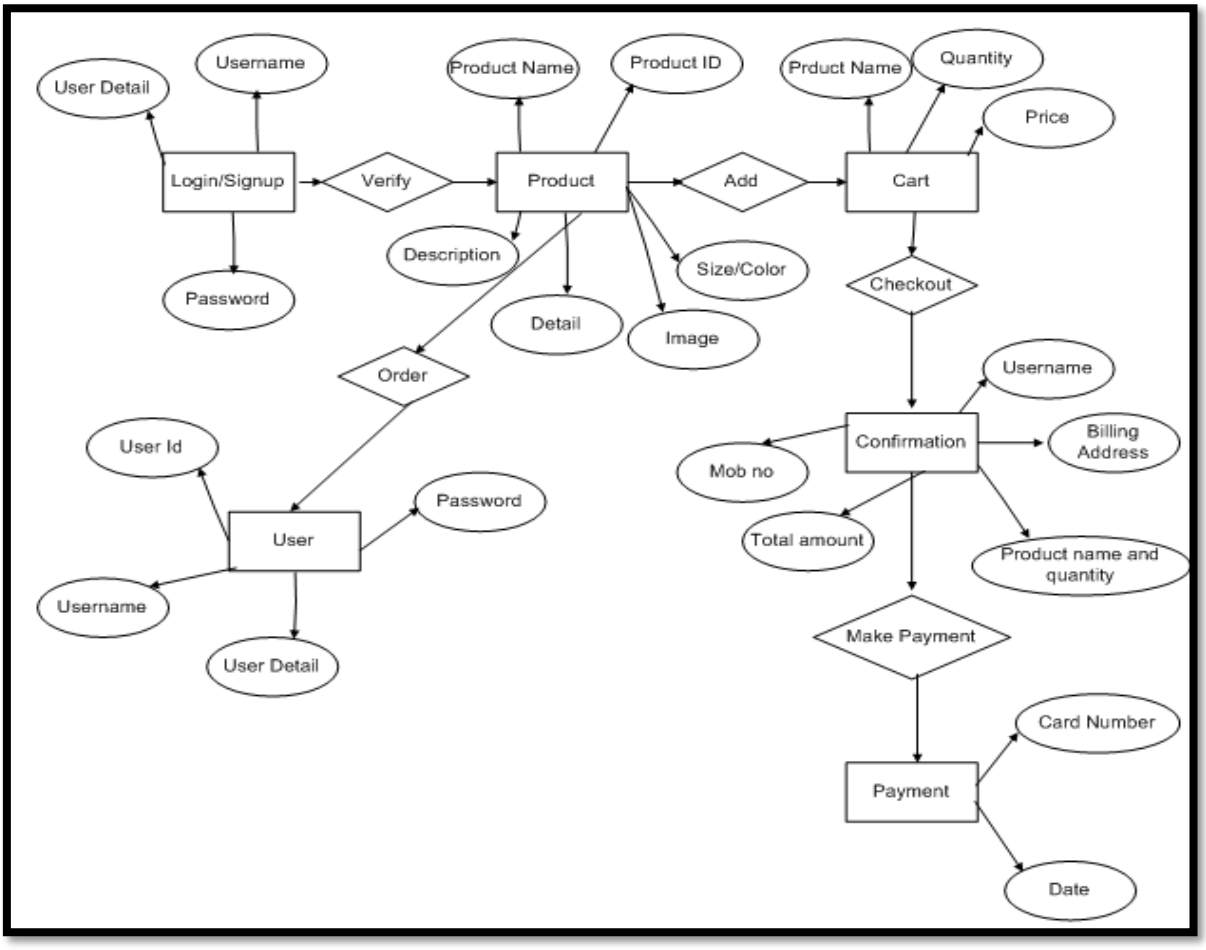


Fig. 5.11 ER Diagram

5.12 COLLABORATION DIAGRAM

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time.

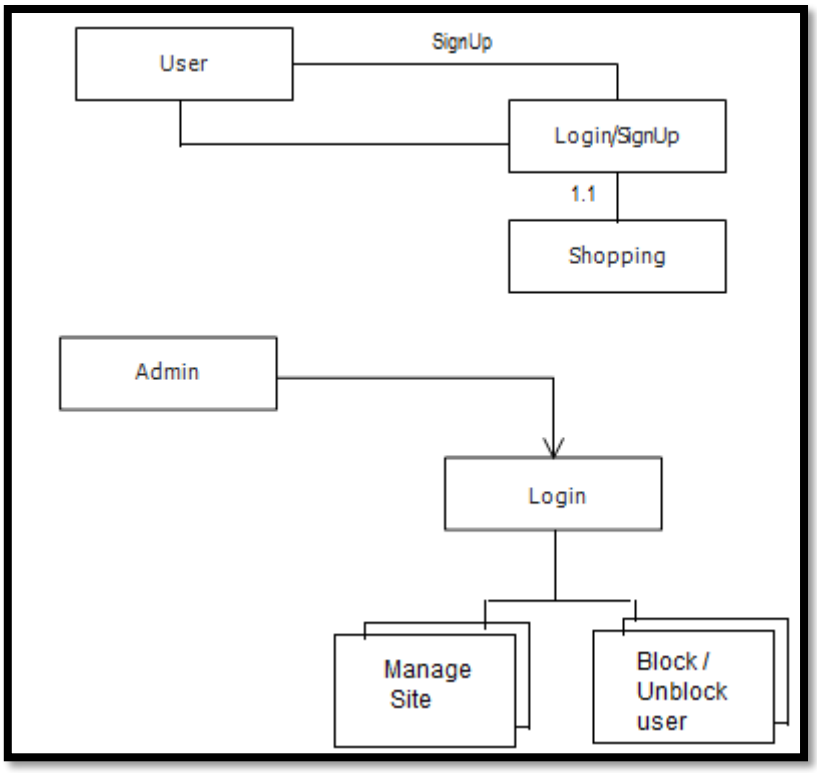


Fig. 5.12 Collaboration Diagram

In Fig 5.11 this diagram shows the relationship between the object are shown as the lines connecting the rectangle

5.13 COMPONENT DIAGRAM

The Component Diagram helps to model the physical aspect of an Object-Oriented software system. It illustrates the architectures of the software components and the dependencies between them

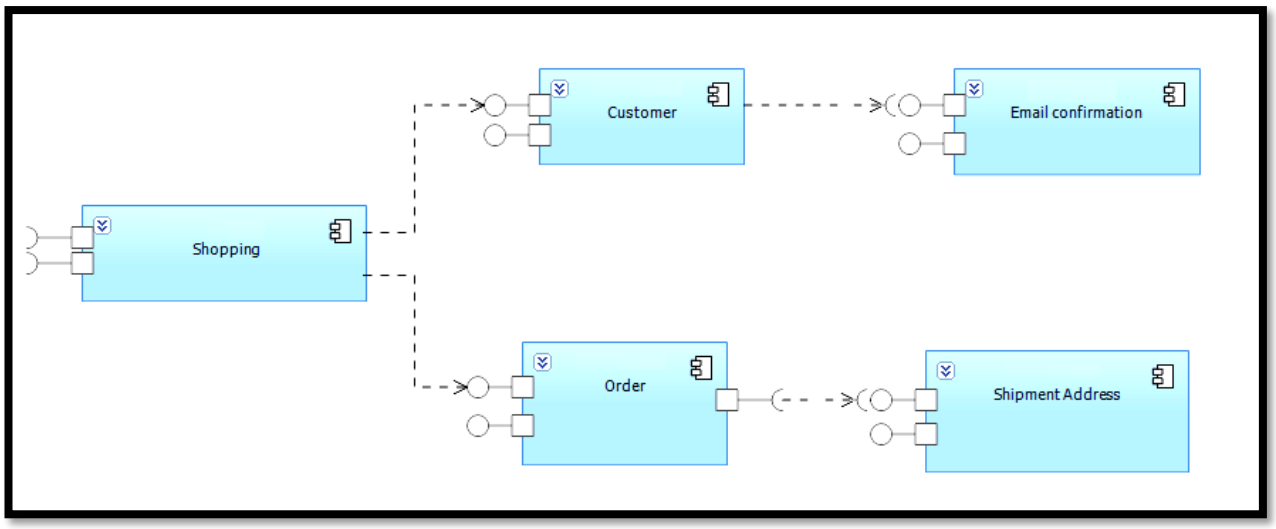


Fig. 5.13 Component Diagram

In Fig 5.12 shows the components are wired together to form larger component of the shopping system

5.14 ASSOCIATION MODEL

Association diagram is a class diagram but unless class diagram which provides an overview of the target system by describing the objects and classes inside the system, Association diagram tells the relationships between them.

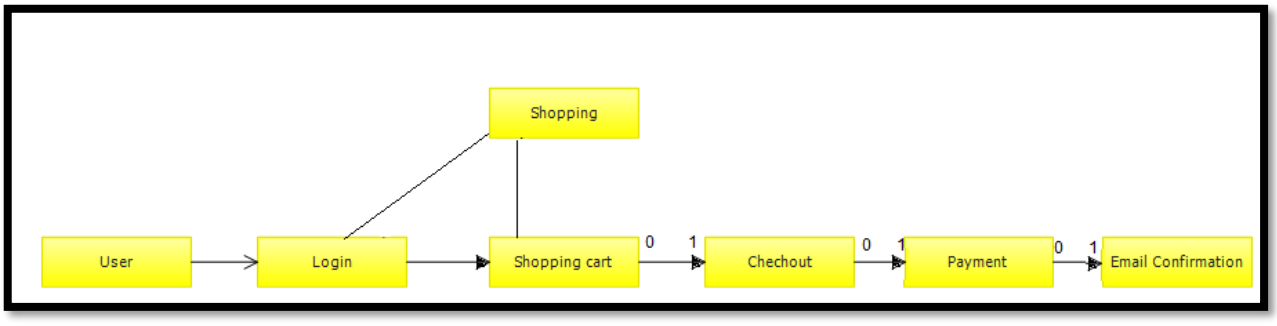


Fig. 5.14 Association Model

In this Fig 5.13 it tells the relationship between the login, shopping cart, checkout, payment and email confirmation

5.1 STATE CHART DIAGRAM

State chart diagram can show the different states of an entity also how an entity responds to various events by changing from one state to another.

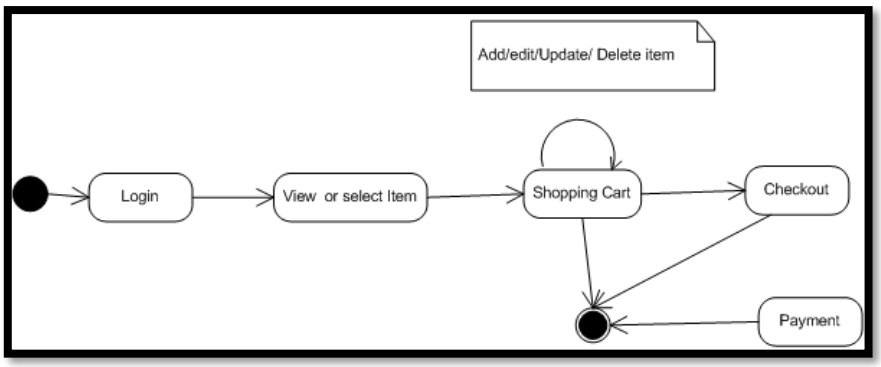


Fig. 5.15 State Chart Diagram

In this Fig 5.14 it shows the state of different nodes that user has to first login then select the item to shopping cart then user can purchase the item

5.16 COMMUNICATION DIAGRAM

Similar to Sequence Diagram, the Communication Diagram is also used to model the dynamic behavior of the use case.

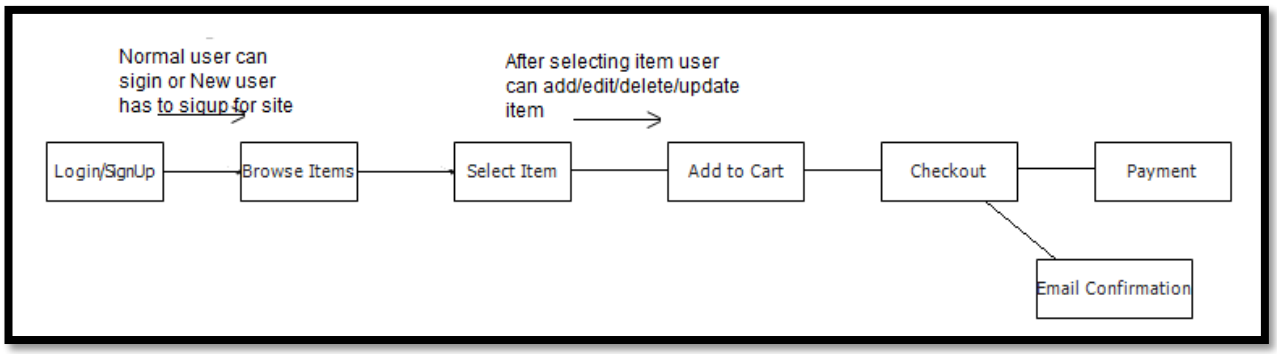


Fig. 5.16 Communication Diagram

In Fig 5.15 shows the collaboration of objects rather than the time sequence

CHAPTER 6

Algorithm And Its Complexity

6.1 Algorithm and Pseudocode

An algorithm is a step-by-step solution to a specific problem. In the context of *Foody*, algorithms are used in areas like login checking, product search, order sorting, etc.

A **pseudocode** is a human-readable version of the logic written in plain structured language. It helps in designing algorithms without focusing on syntax of a programming language.

6.2 Complexity of Code

In *Foody*, evaluating algorithm complexity helps optimize performance—especially important in operations like product searches and login validation.

Cyclomatic Complexity (CCN) measures the number of independent paths through a program's source code:

- CCN 1–4: Low complexity
- CCN 5–7: Moderate complexity
- CCN 6–10: High complexity
- CCN >10: Very complex

Understanding code complexity allows us to write clean, maintainable logic in critical sections like order processing and admin access control.

6.3 Dry Run Algorithm

A **dry run** is a manual simulation of an algorithm or code execution.

In *Foody*, a dry run is useful during:

- Testing the **user login process**
- Verifying **search functionality**
- Simulating the **checkout process**

Dry runs are generally done using tables, showing how variables change step-by-step.

6.4 Search Algorithm (Binary Search Example)

Used when the product catalog is sorted (e.g., by name or price). Here's a sample binary search logic to find a product:

Pseudocode:

```
location = -1
first = 0
last = total number of products - 1

while (first <= last AND target not found)
    middle = (first + last) / 2
    if product[middle] == target
        location = middle
    else if (target < product[middle])
        last = middle - 1
    else
        first = middle + 1
return location
```

6.4.1 Complexity Analysis for Search

Model	Number of Comparisons (n = 100000)	Comparisons as Function of n
Best Case	1	O(1)
Worst Case	16	$O(\log_2 n)$

Table 6.3

Running Time:

- Best: O(1) → target found at first attempt
- Worst: $O(\log_2 n)$ → keeps halving the list until one item is left

6.5 Login Check Algorithm

In *Foody*, login validation is critical for both **admin** and **users**.

Algorithm Steps:

1. Start
2. Get username and password
3. Match input with database
4. If matched:
    - Start session
    - Redirect user to homepage (user) or admin dashboard (admin)
5. If not matched:
    - Return to login
6. Exit

Pseudocode:

```
PROGRAM userFirstLoginCheck
    user dialing dialer
    WHILE reading user session from database
    END WHILE

    IF session found
        PRINT "Not first login"
    ELSE
        PRINT "First Login"
        Update user login info in DB
    END IF
END
```

6.5.1 Complexity Analysis for Login Check

To evaluate search, count the number of comparisons in the best case and worst case. This analysis omits the average case, which is a bit more difficult, and ignores any differences between algorithms in the amount of computation corresponding to each comparison.

The best case occurs if the middle item happens to be the target. Then only one comparison is needed to find it. As before, the best case analysis does not reveal much.

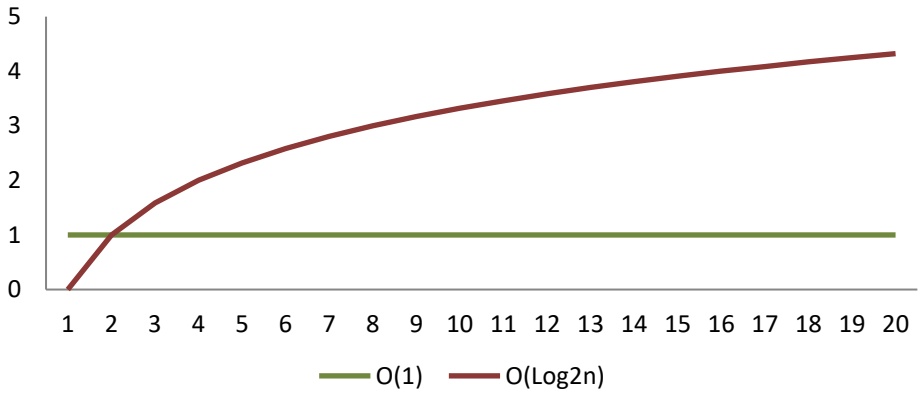
When does the worst case occur? If the target is not in the array then the process of dividing the list in half continues until there is only one item left to check.

Model	**Number of Comparisons** (for n = 100000)	**Comparisons as a function of n**
Best Case (Least/fewest Comparisons)	1 (target is middle item)	1
Worst Case (Most comparisons)	16 (target not in array)	Log_2n

Table 6.3

The following table 6.1 summarizes the analysis for searching algorithm.

6.6 Graphical Representation



Examples:

- Binary Search Tree for product lookup
- Login flowchart: login → session check → role-based redirect

CHAPTER 7

DEVELOPMENT PHASE

7.1 Development Phase – 1: Layout / Form Designing

Objective:

To design the visual structure and user interface (UI) of **"Foody: Your Organic Marketplace"**, focusing on form creation and static layout. The aim is to simulate the complete look and feel of the final web application before functionality is implemented.

Key Activities:

1. Wireframing and UI/UX Design:

- **Wireframes Created For:**
  - Homepage displaying featured organic products and categories.
  - Product listing and product detail pages.
  - User-side pages like Registration, Login, Shopping Cart, and Order History.
  - Admin-side pages like Dashboard, Product Upload, and Contact Message View.
- **Mockups and Styling:**
  - Designed mockups using **HTML5 and CSS3**.
  - Adopted **organic, nature-inspired themes** with green tones and clean typography.
  - Incorporated consistent design across all pages.
- **Responsive Design:**
  - Integrated **Bootstrap 5** to ensure full responsiveness.
  - Designed layouts that automatically adjust for **mobile, tablet, and desktop** devices.

2. Form Designing:

- **User-Side Forms:**
  - **Registration Form:** Fields for name, email, password, contact number.
  - **Login Form:** Email and password inputs with validation.
  - **Password Recovery Form:** Email input for password reset request.
- **Admin-Side Forms:**
  - **Admin Login Form:** Simple username-password verification.
  - **Product Management Forms:**
    - Add New Product: Name, price, description, image upload, organic tag, and category.
    - Update Product: Form to modify existing product details.
    - Delete Product: Product ID-based deletion form.

3. Static Content Creation:

- **Navigation Bar:**
  - Included links to **Home**, **Shop**, **About Us**, **Contact Us**, **Login/Register**, and **Cart**.
  - Dynamic highlighting and dropdown menus added using Bootstrap.
- **Footer Design:**
  - Added sections for **Quick Links**, **Newsletter Subscription**, and **Social Media Icons**.
  - Incorporated brand logo and copyright.
- **Placeholder Content:**
  - Used **placeholder images (from Unsplash and Pexels)** and **Lorem Ipsum** text.
  - Helped in aligning future backend data integration.

Outcome:

- Fully designed and responsive **static frontend pages** for both user and admin modules.
- Clean and consistent **form structure** to handle user data, product uploads, and admin interactions.
- Created a ready-to-develop base for integrating backend PHP and MySQL functionalities in future phases.
- Ensured smooth user navigation and intuitive layout, setting a professional tone for the online organic food platform.

7.2 Development Phase – 2: Layout / Form Design with Code and Database Integration

Objective:

To integrate static frontend designs with server-side scripts using **PHP** and connect them to a **MySQL** database. This phase transforms the static user interface into a dynamic and interactive web application by connecting forms and layouts to functional backend operations.

Key Activities:

1. Database Design and Table Creation

A relational database named **foody marketplace** was designed using structured tables that handle user registration, product listings, orders, messages, and admin activity logs.

Table 1: users

Field Name	Data Type	Constraints	Description
user_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each user
first_name	VARCHAR(50)	NOT NULL	User's first name
last_name	VARCHAR(50)	NULL	User's last name (optional)
email	VARCHAR(100)	NOT NULL, UNIQUE	User's email address
mobile	VARCHAR(15)	NOT NULL	User's mobile number
password	VARCHAR(255)	NOT NULL	Hashed password
role	ENUM	('customer','seller','admin') DEFAULT 'customer'	User type
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Registration date and time

Table 2: categories

Field Name	Data Type	Constraints	Description
category_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each category
category_name	VARCHAR(100)	NOT NULL, UNIQUE	Name of the product category
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Creation timestamp

Examples: Fruits, Vegetables, Dairy, Organic Grains, Herbs, etc.

Table 3: products

Field Name	Data Type	Constraints	Description
product_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique product ID
seller_id	INT(11)	FOREIGN KEY (users.user_id)	Seller who listed the product
product_name	VARCHAR(150)	NOT NULL	Product name
description	TEXT	NOT NULL	Detailed product description
price	DECIMAL(10,2)	NOT NULL	Product price
category_id	INT(11)	FOREIGN KEY (categories.category_id)	Product category
image_path	VARCHAR(255)	NOT NULL	Path of uploaded image file
unique_url	VARCHAR(255)	NOT NULL, UNIQUE	SEO-friendly product URL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Upload timestamp

Table 4: orders

Field Name	Data Type	Constraints	Description
order_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique order ID
user_id	INT(11)	FOREIGN KEY (users.user_id)	Customer placing the order
order_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Order timestamp
total_amount	DECIMAL(10,2)	NOT NULL	Total price of the order
status	ENUM	DEFAULT 'pending'	Order status (pending, shipped...)

Table 5: order_items

Field Name	Data Type	Constraints	Description
order_item_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique order item ID
order_id	INT(11)	FOREIGN KEY (orders.order_id)	Related order
product_id	INT(11)	FOREIGN KEY (products.product_id)	Product ordered
quantity	INT(11)	NOT NULL	Quantity purchased

Table 6: admin_logs (Optional)

Field Name	Data Type	Constraints	Description
log_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique log ID
admin_id	INT(11)	FOREIGN KEY (users.user_id)	Admin performing the action
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Action timestamp

Table 7: messages (Contact Us Form)

Field Name	Data Type	Constraints	Description
message_id	INT(11)	PRIMARY KEY, AUTO_INCREMENT	Unique message ID
user_id	INT(11)	FOREIGN KEY (users.user_id), NULL	Optional user ID (if logged in)
name	VARCHAR(100)	NOT NULL	Sender's name
email	VARCHAR(100)	NOT NULL	Sender's email
subject	VARCHAR(150)	NOT NULL	Subject of the message
message_body	TEXT	NOT NULL	Full message content
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Message timestamp

2. Backend Integration

2.1 Database Connection Using PHP:

```
// config.php

$dbHost = 'localhost';

$dbUser = 'root';

$dbPass = '';

$dbName = ‘shopping’;

$conn = new mysqli($dbHost, $dbUser, $dbPass, $dbName);

if ($conn->connect_error) {

   die("Connection failed: " . $conn->connect_error);

}
```

2.2 User Registration and Login Functionality:

- Captures data via forms and inserts it into users table after hashing passwords.
- Login verifies entered credentials and starts session.

2.3 Product Upload System:

- Allows sellers to upload new organic products.
- Image upload and validation are handled.
- Unique SEO-friendly URL is generated for each product.

2.4 Dynamic Product Display Using PHP + SQL:

```
SELECT p.product_id, p.product_name, p.price, p.image_path, c.category_name

FROM products p

JOIN categories c ON p.category_id = c.category_id

WHERE p.seller_id = ?;
```

- Products are displayed dynamically with category filters.

3. Navigation and Content Linking

- Used PHP include for **header.php** and **footer.php** to maintain consistency across pages.
- Navigation bar dynamically adapts based on user login status (Customer / Seller / Admin).

Outcome:

- The static frontend was successfully connected to a dynamic backend.
- The website now supports:
  - User registration and login
  - Product uploads and listings
  - Order and message handling
- A **functioning prototype** has been established that interacts with a well-structured MySQL database, laying the groundwork for advanced features like cart, order tracking, payment integration, and admin analytics.

7.3 Development Phase – 3: Layout / Form Designing with Validation and Enhanced Security

Objective:

To enhance the system with comprehensive validation mechanisms and critical security practices that ensure data integrity, user privacy, and a smooth user experience.

Key Activities:

1. Client-Side Validation:

- **JavaScript-Based Validation:**
  - **Registration Form Validation:**
    - Required fields: First Name, Email, Password, Mobile Number.
    - **Email Format:** Regular expression validation (e.g., `/^\S+@\S+\.\S+$/`).
    - **Mobile Number:** Exactly 10 digits using regex.
    - **Password:** Minimum 6–8 characters with complexity (uppercase, lowercase, number).
  - **Product Upload Form Validation:**
    - Mandatory fields: Product Name, Description, Price (> 0), and Image file.
    - Validates file types (e.g., JPG, JPEG, PNG) and size (e.g., <2MB).
  - **Real-Time Feedback:**
    - Error messages appear instantly when the user inputs incorrect or incomplete data.

2. Server-Side Validation (Using PHP):

- **Sanitization and Validation:**

Inputs are cleaned using `htmlspecialchars()` and `mysqli_real_escape_string()`.

Duplicate Email Check (Example):

```
$email = $_POST['email'];
$sql = "SELECT * FROM users WHERE email = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();
if($result->num_rows > 0) {
    // Handle error: email already exists
}
```

- **Image Validation:**
  - Only allows images with MIME types `image/jpeg`, `image/png`, etc.
  - Restricts max size (e.g., 2MB) and checks for upload errors.

3. Error and Success Handling:

- **User-Friendly Feedback:**
  - Errors are shown near the relevant input fields.
  - On successful registration or form submission, confirmation messages like "Registered Successfully" or "Product Uploaded" are displayed.
- **Redirection:**
  - After successful login, users are redirected to their respective dashboards (e.g., Customer Panel, Seller Dashboard, or Admin Panel).

4. Security Enhancements:

- **Password Hashing:**
  - Passwords are securely hashed using:

    ```
    $hashed = password_hash($password, PASSWORD_DEFAULT);
    ```

  - Password verification is done with `password_verify()`.
- **SQL Injection Prevention:**
  - All database queries are written using prepared statements.
- **Session Management:**

  Secure PHP sessions track logged-in users and block unauthorized access to restricted pages.

  Example:

```
session_start();
if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
     }
```

- **CSRF Protection (Optional):**
  - Forms can include CSRF tokens to verify valid requests and prevent cross-site request forgery attacks.

Validation Summary Table:

Form	Field	Client-Side Validation	Server-Side Validation
Registration	Email	Regex (`/^\S+@\S+\.\S+$/`)	Check duplicates + sanitize
	Password	Min 6 chars, mixed case + digit	Hashing + password_verify
	Mobile Number	10-digit numeric	Regex + numeric check
Product	Price	Must be > 0	`is_numeric()` + range check
Upload	Image	Type + size check	MIME type + file error check

Bonus Security Notes:

- **XSS Protection:** Use `htmlspecialchars()` when echoing user data to prevent cross-site scripting.
- **Session Timeout:** Auto logout users after inactivity (e.g., `$_SESSION['last_activity']` timeout logic).
- **HTTPS Enforcement:** Suggest using HTTPS in live hosting for secure data transmission.

Outcome:

- The system becomes more secure and robust with multi-layered validation.
- User interaction improves with real-time feedback and clear instructions.
- All submitted data is well-validated, protected from attacks, and stored securely in the database.

Summary of Database Interactions:

- **User Registration/Login:**
  Data flows from registration forms → validated (client/server-side) → stored in the `users` table.
- **Product Upload:**
  Seller fills the product upload form → file upload and product data are validated → saved in the `products` table with a generated `unique_url`.
- **Order Management:**
  Customer places an order → order data is captured and stored in the `orders` and `order_items` tables.
- **Category Management:**
  Products are categorized based on the `categories` table to facilitate easy filtering on the front end.
- **Message Handling:**
  Feedback or inquiries via the "Contact Us" form are stored in the `messages` table for admin review.

CHAPTER 7

TESTING

7.0 TESTING PHASE

The testing phase plays a vital role in ensuring the quality and stability of the **Foody Online Marketplace** system. Various testing methods were implemented to identify and rectify bugs, validate functionality, and ensure a smooth user experience. The testing process included both manual and automated approaches for frontend and backend functionalities.

7.0.1 TYPES OF TESTING

7.0.1.0 Gray Box Testing

Verifies that the product meets customer-specified requirements. Typically conducted by the customer for externally developed systems.

7.0.1.1 Black Box Testing

Tests functionality without knowledge of internal code structure. Focuses on input/output behavior.

7.0.1.2 Compatibility Testing

Ensures compatibility with different browsers, operating systems, and devices. Can be manual or automated.

7.0.1.3 Conformance Testing

Checks implementation against industry standards to ensure interoperability, portability, and compliance.

7.0.1.4 Functional Testing

Validates that the application performs all required functions as per specifications using both valid and invalid inputs.

7.0.1.5 Integration Testing

Tests the interaction between integrated modules (e.g., databases, APIs) after unit testing but before system testing.

7.0.1.6 Load Testing

Measures how the application performs under expected and peak load conditions. Often includes performance and stress testing.

7.0.1.7 Performance Testing

Evaluates system responsiveness and stability under various workloads. Identifies bottlenecks and establishes benchmarks for regression.

7.0.1.8 Regression Testing

Ensures new changes haven't affected existing functionality. Usually automated for consistency and speed.

7.0.1.9 Smoke Testing

Basic functional checks to verify that major features work. If it fails, further testing is paused.

7.0.2.0 Stress Testing

Pushes the system beyond normal load limits to observe how it handles high traffic or data volume. Focuses on failure behavior.

7.0.2.1 System Testing

Validates the complete and integrated system against its requirements. Conducted without access to the internal logic.

7.0.2.2 Unit Testing

Tests individual components/functions of the system in isolation to ensure correct behavior before integration.

7.0.2.3 White Box Testing

Tests internal structures or workings of an application, often used to ensure full branch/path coverage.

7.1 PERFORMANCE TESTING

Performance testing was conducted to measure the system's responsiveness, stability, and behavior under varying load conditions.

- **Tools Used**: Chrome DevTools, Apache JMeter
- **Tests Performed**:
  - Page Load Speed
  - Server Response Time
  - Product Search Speed
- **Results**:

- Homepage load time: 1.8 seconds (average)
- Cart page response time: 2.2 seconds
- API response time: < 1.5 seconds

Conclusion: The system performed efficiently under normal load and maintained stability during concurrent requests.

7.2 STRESS TESTING

Stress testing was applied to determine the system's breaking point by pushing it beyond normal operational limits.

- **Scenarios Tested**:
  - Simulated 100+ users placing orders simultaneously.
  - Bulk registration of 200 users.
- **Observations**:
  - Minor delay in database processing (~3.5 seconds lag).
  - No crashes or data loss occurred.

Conclusion: The system handled high load conditions gracefully, though slight optimization is needed for bulk operations.

7.3 LOAD TESTING

Load testing was performed to observe system behavior under expected user load.

- **Test Load**: 50 concurrent users browsing, adding products to cart, and checking out.
- **Outcome**:
  - No system downtime
  - All transactions completed successfully
  - Server CPU usage peaked at 78%

Conclusion: System is well optimized for real-world usage under expected traffic levels.

7.4 FUNCTIONAL TESTING

Functional testing ensured each feature of the platform behaves as expected under valid and invalid scenarios.

Feature	Test Scenario	Expected Result	Status
User Registration	Valid/Invalid data	User created/Error messages	Pass
Login Authentication	Correct/Wrong credentials	Redirect/Error shown	Pass
Product Add to Cart	Add, remove, update items	Cart reflects actions	Pass
Checkout System	Submit with complete/incomplete data	Order placed/Error shown	Pass
Admin Login	Secure login and redirection	Admin Dashboard loaded	Pass
Contact Form	Message submission	Stored and viewed in backend	Pass

Conclusion: All major modules passed functional testing with valid and invalid data inputs.

7.5 BUGS & ISSUE REPORTING

During testing, the following bugs were encountered and resolved:

Bug ID	Description	Severity	Status	Fix Applied
B001	Cart not updating item count in navbar	Medium	Fixed	Added JavaScript event binding
B002	Error message not showing on login fail	Minor	Fixed	Corrected PHP session handling
B003	Contact form submission alert missing	Minor	Fixed	Added JS alert on form submit
B004	Mobile view menu overlap	Medium	Fixed	Adjusted CSS flexbox and media query

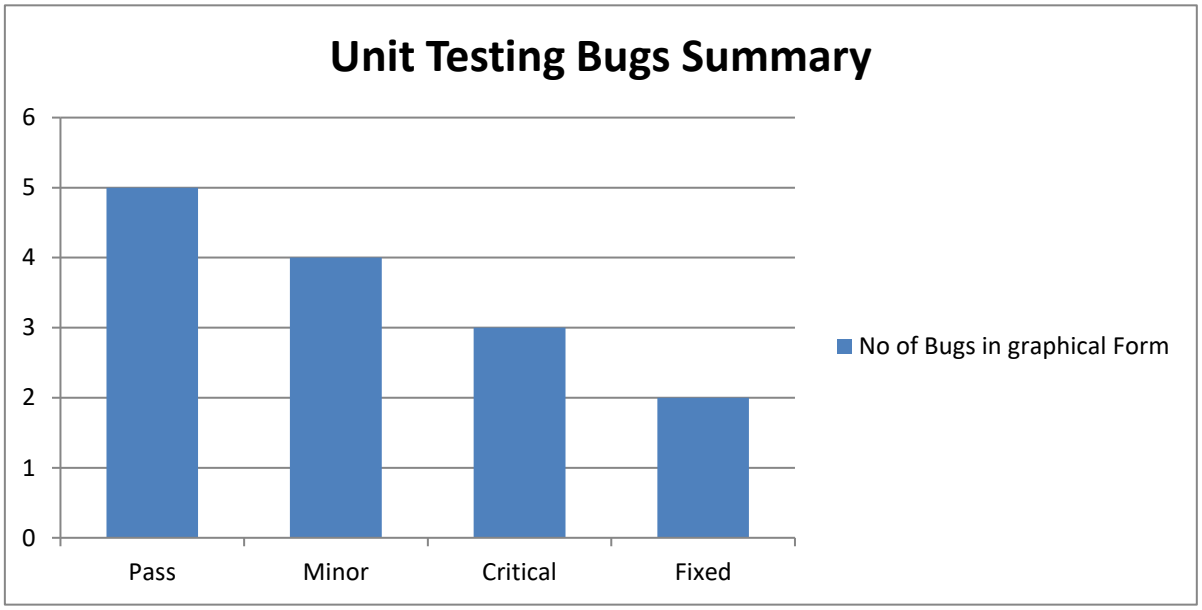


Figure 7.4.7Unit testing Bugs Graph

7.6 TEST CASES SUMMARY

Test Case ID	Module	Input	Expected Output	Actual Output	Status
TC001	Registration	Valid user data	Account created	Account created	Pass
TC002	Login	Invalid credentials	Error message shown	Error message shown	Pass
TC003	Cart	Add multiple items	Items appear in cart	Items appear in cart	Pass
TC004	Checkout	Incomplete address	Validation error	Validation error	Pass
TC005	Admin Login	Correct credentials	Redirect to dashboard	Redirect to dashboard	Pass
TC006	Contact Form	Valid message data	Confirmation shown	Confirmation shown	Pass

7.7 TESTING TOOLS USED

The following tools and platforms were used for comprehensive testing:

- **Manual Testing** – Frontend navigation, form validation, and user experience
- **Google Chrome DevTools** – Inspecting network requests, load times, and debugging
- **Apache JMeter** – Load and stress testing
- **Postman** – Backend API testing
- **PHP Unit (optional)** – Unit testing for PHP functions (admin-side modules)

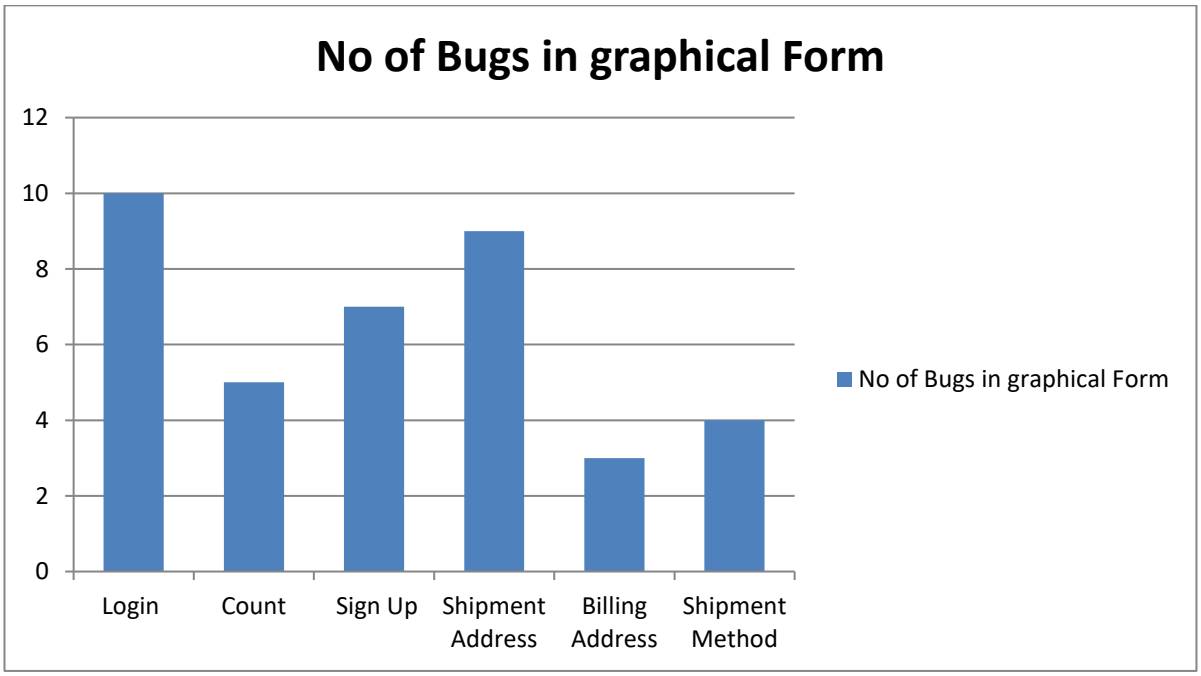


Figure 7.4.7Unit testing Bugs Graph

7.8 USER ACCEPTANCE TESTING (UAT)

A prototype version of the website was presented to sample users and faculty for review.

- **Feedback Collected**:
  - "Easy to navigate and use"
  - "Smooth shopping and checkout experience"
- **Suggestions**:
  - Add order confirmation mail (implemented later)
  - Improve search filter (added dropdown-based category search)

Conclusion: The system was accepted by the target audience with positive feedback and minor enhancement suggestions.

CHAPTER 8

CONCLUSION

8.0 CONCLUSION

After thorough development and observation, it is clear that **e-commerce has transformed into an indispensable part of modern life**. As people increasingly rely on digital platforms for convenience, businesses must adapt to stay relevant. For small enterprises, especially in niche markets like organic food, having a robust online presence is not just beneficial—it's essential.

The **Foody Online Marketplace** was developed to meet this growing demand by providing a platform for healthy, organic, and sustainable food options. It offers customers a convenient, secure, and intuitive way to explore, select, and purchase organic products from the comfort of their homes.

While traditional shopping has the advantage of personal interaction and physical inspection, **Foody compensates for these limitations** with features such as:

- Detailed product descriptions and high-quality visuals
- Live support and customer care
- Secure payment gateways and smooth checkout processes

8.1 Project Achievements

The development of Foody resulted in:

- A fully responsive web platform for buyers and sellers
- Secure user registration, login, and authentication
- Real-time cart and checkout system
- Admin backend for product and order management
- Integration of feedback and contact form systems
- Performance-optimized code and load testing success

8.2 Benefits of Foody

- **Persistent connection** with customers through accounts, subscriptions, and offers
- **New value generation** via curated organic product categories
- **Access to a wider audience**, especially those seeking eco-friendly alternatives

8.3 Future Scope

- Implementation of **mobile apps** for Android and iOS
- Addition of **AI-driven recommendation systems**
- Integration with **logistics APIs** for real-time order tracking

- Customer loyalty programs and digital wallets
- Support for multiple languages and regional pricing

8.4 Social and Environmental Impact

By promoting locally-sourced organic produce, Foody contributes positively to:

- Reducing environmental impact due to shorter supply chains
- Supporting small-scale farmers and local businesses
- Promoting a healthier lifestyle for consumers

8.5 Final Thoughts

Understanding how e-commerce affects businesses and the broader economy is crucial. Projects like Foody not only demonstrate how small businesses can thrive digitally but also highlight the social responsibility of building platforms that are sustainable, inclusive, and user-friendly.

As technology evolves, Foody will continue to grow—guided by user feedback, technological innovation, and a commitment to health and sustainability.

CHAPTER 9

FURTHER ENHANCEMENT AND RECOMMENDATIONS

FURTHER ENHANCEMENT AND RECOMMENDATIONS

The project **"Foody: Online Marketplace"** has been successfully implemented and performs accurately on the hardware and software environment detailed in Chapter 2. While the current version meets core functional and design expectations, several areas can be further enhanced for a better user experience and system efficiency.

Areas of Improvement:

- **Shopping Cart Design:**
  Although the web application adheres to **Human-Computer Interaction (HCI)** principles and offers a user-friendly interface, the **shopping cart UI can be enhanced** for better clarity and usability—such as implementing real-time updates, item previews, and an improved summary layout.

Recommended Enhancements:

- **Mobile App Integration:**
  Developing native Android/iOS apps to improve accessibility and expand user reach.
- **AI-Based Recommendations:**
  Adding product suggestions based on user behavior and preferences to improve user engagement.
- **Order Tracking System:**
  Integrating third-party logistics APIs for real-time tracking of deliveries.
- **Customer Loyalty System:**
  Implementing reward points, discounts, and referral programs to retain users.
- **Voice Search Integration:**
  Enabling voice-based product search to enhance accessibility.
- **Multilingual Support:**
  Allowing users to browse the site in regional languages to cater to a broader audience.

These recommendations will help Foody evolve into a more robust, scalable, and user-centric e-commerce platform.

CHAPTER 10

REFERENCES/BIBLIOGRAPHY

1. Google Search Engine for general information
2. *Web Technologies Black Book*
3. Keeves – *PHP & MySQL*
4. *Beginning PHP5* – Wrox Press
5. www.w3schools.org – Online Tutorials
6. *Murach's PHP/MySQL*

Additional Academic Sources:

- Arnold, M. J. et al. (2005). *Customer delight in retail*. *Journal of Business Research*.
- Martin, B. A. S. (2012). *Interpersonal touch effects*. *Journal of Consumer Research*.
- Harada, M. et al. (1996). *Personal ID by face images*. *IEICE Report*.
- Goldberg, L. R. (1993). *Personality traits*. *American Psychologist*.
- John, O. P. et al. (2008). *Big-Five Trait Theory*. Guilford Press.
- Phillips, J. J. et al. (2001). *Human Resource ROI*. ASTD Publications.
- Kaner, C. (2006). *Exploratory Testing*. QAI Conference.
- Kitchenham, B. et al. (2002). *Software Metrics*. IEEE Xplore.
- Runeson, P., & Höst, M. (2009). *Case Studies in Software Engineering*. Springer.
- *Stanford Online Courses* – online.stanford.edu