*Heaven's Light is Our Guide*

# Name of the Title

**Green Leaf: Plant E-commerce Website for Online Buying, Selling, and Learning**

By

Md. Humayun Khalid

(2110015)

Software Development Project- II (ECE-3100)

Examination Committee: Assistant Prof. Md. Abu Hanif Pramanik

Rajshahi University of Engineering & Technology

Faculty of Electrical & Computer Engineering

Department of Electrical & Computer Engineering

Rajshahi-6204

September 2025

## Acknowledgment

We want to take this opportunity to thank our highly esteemed course teacher, **Md. Abu Hanif Pramanik,** whose guidance, valuable suggestions, and support were very helpful during the process of developing this project. His coaching and feedback were also very instrumental in designing and implementation of the Plant E-commerce Website successfully. The project gave us an applied experience to use our knowledge of HTML, CSS, JavaScript, PHP, and MySQL to create a working online platform and make it interactive. We also appreciate the cooperation and encouragement of our classmates in the process. Lastly, the development, testing, and deployment of the site would have been impossible without the use of XAMPP and other development tools, to which we appreciate the fact that they were used.

*Md. Humayun Khalid*

*Roll: 2110015*

# Table of Contents

# List of Figures

# List of Tables

## Abstract

The Plant E-commerce Website project will be intended to offer an all-inclusive e-commerce site on which to purchase, sell, and educate about different plants. The system incorporates the latest web technologies such as HTML, CSS, JavaScript, PHP, and MySQL in order to provide an interface that is responsive, user-friendly, as well as secure. The site gives them the chance to create accounts, log in, navigate plants, and place items in the carts and buy as well as keeping track of account and order history. Admins can access dedicated access in order to manage products, track orders and monitor user activity. It has several distinctive characteristics such as dynamic product listing, search, cart real-time updates, and secure authentication. The project has focused on the functionality and usability of the project to create a gap between the traditional and modern e-commerce in plant shopping. With the interactive design of the UI and the strong activity of the backend, this system provides an effective, scaling, and easy-to-use solution.

# Chapter 01: Introduction

## 1.1 Introduction

The emerging speed in technology and internet connectivity has changed the way individuals shop goods and services. The old-fashioned shopping has gradually been transformed to online shopping that enables users to view, compare and buy products at the comfort of their homes. Plant selling is a developing niche in the e-commerce that has gained popularity among other niches.

This is an online commerce site, Plant E-commerce Website, which is aimed at offering an online market whereby the users can acquire the knowledge about the plant, its care and purchase the different plants in a safe environment. The system will integrate both user-friendly interface and a well-organized backend to guarantee a user easy shopping and management experience. This allows the customers to shop, save the products in their carts, and make purchases, and the administrators to manage products, users, and orders effectively using an admin panel.

## 1.2 Problem Statement

Buying plants is usually restricted to the physical nurseries or stores, which has a number of problems: The desired variety of plant might not be always available to the customers. Absence of a centralized information of plant care and maintenance. The physical stores may cause ineffectiveness in the processes and restricted customer scope. To the sellers, customer order tracking and inventory control becomes challenging when they are not digitally aided. In order to overcome these shortcomings, a specific e-commerce site designed specifically to deal with plants is needed. The said platform must ease consumers with the purchasing process and offer tools that will allow the administration to manage the sales, inventory, and communication effectively.

## 1.3 Project Objectives & Outcomes:

The project has been worked out with certain purposes: Objectives: To establish and carry out of a reactive e-commerce site to sell plants. To enable the customers to create accounts, log in, shop, and regulate their cart, as well as order items. To incorporate secure order booking and checkout services with database support. To create an administrative dashboard to manage users, products and orders and customer messages. To have seamless database connectivity on the MySQL database to store and retrieve data. Expected Outcomes: An operational site on which users are able to browse and order plants online. A back-end administration program that is able to support daily activities like product management and orders monitoring. Improved user experience as a result of clean navigation and secure sign in and sign out. A solution that can be scaled to add additional features later (e.g. payment gateways, delivery tracking).

## 1.4 Significance of the Study

The project has an academic and practical implication: Academic Value: It shows the combination of various technologies like, HTML, CSS, PHP, JavaScript, and MySQL to create a complete stack project. Students will have a chance to know how such concepts as database manipulation, session management, and CRUD operations are applied to the real world. Practical Value: The site can be used as an example of a real business that intends to sell plants on the Internet. Through the creation of an easily reachable marketplace and allowing the administrators to continue their operations without any disturbances, the project illustrates how technology can help small-scale companies to reach more people.

## 1.5 Scope of the Project

The Plant E-commerce Website deals with the following scope:

- **User Features:**
  - ➢ Registration and system of logging in.
  - ➢ Examining plant information including names, pictures and prices.
  - ➢ Making purchases to the cart and checkout.
  - ➢ Seeing order history and payment status.
  - ➢ Sending a message to the admin.

- **Admin Features:**
  - ➢ Safe administrator log in having dashboard.
  - ➢ The ability to add, update, and remove products.
  - ➢ Tracking the orders and changing payment statuses.
  - ➢ User management (customers and administration account).
  - ➢ Seeing customer requests and messages.

- **Technology Scope:**
- Frontend: HTML, CSS, JavaScript for design and interactivity:
- Backend: PHP for server-side logic.
- Database: MySQL to store structured data.
- Additional Tools: XAMPP to use as a local hosting and testing tool.

This coverage makes sure that the e-commerce process is complete, including the process of browsing to order fulfillment, as well as providing the management system functions to administrators.

# Chapter 2: Requirements & Tools

## 2.1 Introduction

The requirements had to be precisely defined and the correct tools to be used in the project chosen before developing the Plant E-commerce Website.

These requirements cover both the software tools that will be utilized in the development (IDES, version control systems, and local servers) and the technologies that are to be adopted to construct the very system (HTML, CSS, PHP, and MySQL). Each decision was arrived at after taking into account the project requirements like database connectivity, front end design, easy user interface and quality back-end processing.

A combination of these technologies and tools makes the site offer its customers a convenient shopping platform as well as allowing administrators to control the products, orders and users effectively.

## 2.2 Software Requirements

The project needed the following software and tools to be successfully implemented and tested:

- ❖ **XAMPP:** XAMPP is an open-source package of software that offers a full-fledged local development environment. It has the Apache server, the php interpreter as well as mysql database all integrated together, thus it is highly convenient to the developers. In this project, the XAMPP package was employed to host the web pages in the local computer, execute the PHP scripts, and administer the MySQL database, and it did not need an external server.
    - ➢ **Role in the project**: Local hosting environment for server-side scripts and database operations.
    - ➢ **Reason behind being chosen:** Easy to install, cross platform and popular with PHP-MySQL applications.

*Fig 1: XAMPP*

- **Visual Studio Code (VS Code):** Visual Studio Code is a lightweight, but a powerful, Integrated Development Environment (IDE) created by Microsoft. It was used as the primary writing, editing, and debugging tool. The development environment was faster and more efficient due to such features as syntax highlighting, smart code suggestions, terminal integration, and a wide selection of extensions.
  - **Role in the project:** Main IDE coding and debugging.
  - **Reason why it was selected:** It is free, customizable, has a variety of supported programming languages, and it works well with GitHub.

*Fig 2: VS code*

- **Web Browser (Google Chrome / Firefox):** Web Browser is needed to preview and test web sites as it develops. In this project, Google Chrome and Mozilla Firefox were used as browsers to test various pages of the Plant E-commerce Website. They assisted in making sure the design was reactive, the JavaScript functionality was working well, the website was consistent in various platforms of use.
  - **Role in the project**: Testing and previewing the site in the live mode.
  - **Reasons why it was selected:** Stable, popular, and gives adequate developer capabilities to debug.

*Fig 3: Web browsers*

- **GitHub:** GitHub is a version control and collaboration platform, which is based on Git. It was employed to keep various copies of the project, record progress and have an online backup of the code. GitHub made sure that the codebase was well organized, safe and manageable even in an isolated project such as this.

- **Role in the project**: Popular tool for version control, documentation, and online backup.
- **Reason behind being chosen:** Free, safe and common among developers all over the world.

***Fig 4: Github***

## 2.3 Programming Languages & Frameworks

- **HTML (Hypertext Markup Language):** HTML specifies the structure and layout of the web pages. In this project, a skeleton of each page such as the homepage, product listing, login, registration and checkout pages was developed in HTML. With the help of HTML tags, I managed to generate navigation menus, forms, and tables as well as product cards.
  - **Role in the project:** This provides the foundation of all the web pages.
  - **Reason behind being chosen:** It is a standard mark-up language used in the web and can be universally used by every browser.

***Fig 5: HTML***

- **CSS (Cascading Style Sheets):** CSS is the one that makes the site attractive whereas the HTML gives structure. CSS was used to style buttons, forms, navigation bars, product displays and layouts. It has enabled me to customize color, fonts, space, and responsiveness, thus it is responsive to variations in screen size.
  - **Role in the project: Improves** the visual display and the user experience.
  - **Reason behind being chosen:** CSS allows unity and appealing design without making any changes to HTML code.

*Fig 6: CSS*

- **JavaScript:** JavaScript was necessary to provide interactivity to the site. It has transformed the site to be dynamic, through the ability to validate forms, interactive navigation, to search and update the cart without having to reload the site. To illustrate, in case a user clicks on the add to cart button with a plant, the JavaScript will automatically show a change in the number of items in the cart.
  - **Role in the project**: Provides interaction and enhances dynamism.
  - **Reasons why it was selected:** It is the most popular client-side scripting language of web applications.

*Fig 7: JavaScript*

- **PHP (Hypertext Preprocessor):** PHP was the server scripting language used in this project. It provided the functions of user authentication, session management, product insertion, reservation, and communication with the database as a backend. As an example, the use of login.php validates the credentials and redirects them according to the type of user (admin or customer).
  - **Role in the project**: Backend logic, session management, communication with the database.
  - **Reasons why it was selected:** Simple to work with HTML and MySQL, common web hosting platforms.

*Fig 8: PHP*

- **MySQL:** MySQL was used as the relational database in order to store and manage data. It contains user information, plant products, orders, reservations, and administration. I could easily retrieve, add, update and delete data by writing SQL queries.
  - **Role in the project**: Data storage and retrieval are safe and effective.
  - **Reason behind being chosen:** Open-source, lightweight, and reliable database system commonly used with PHP.

*Fig 9: MySQL*

## 2.4 Development Tools (IDE, Version Control, Libraries)

- **IDE:** This project was coded using visual studio code (VS Code). VS Code enabled the development process to be faster, with such features as syntax highlighting, auto-completion, debugging, and integration of version control.

- **Version Control:** As a documentation tool and version control, GitHub was employed. It assisted me in preserving various copies of the code, changes, and worked more efficiently. Although I did the project alone, GitHub provided me with the necessary ability to have my work properly organized and stored in the cloud.

- **Libraries:** Most of the project was developed in raw code, but there was use of some libraries of JavaScript and CSS code to hasten the project and enhance the design. As an example, there were prepared CSS classes used in the development of responsive layouts, and JavaScript functions made it easier to validate forms and interactivity.
  - **Font Awesome:** Font Awesome is used to have icons (cart, user, search, etc.) to enhance the user interface/experience.
  - **CDN Links:** To load stylesheets and scripts faster.

## 2.5 System Architecture Overview

The system is developed based on a two-level client-server architecture.

1. **Frontend (Client Side):**
   a. Constructed in HTML, CSS and JavaScript.
   b. Offers the customer interface through which customers get to interact with the system.

2. **Backend (Server Side):**
    a. PHP acts as the server-side scripting language.
    b. Handles user authentication, product processing, carts and order processing.

3. **Database Layer:**
    a. MySQL contains structured information like the user information, products in-store, order history and the cart.

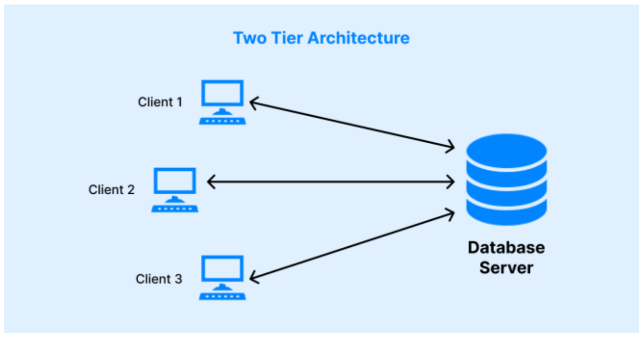


***Fig 10: Two-level client-server architecture***

# Chapter 3: Database Design

## 3.1 Database Requirements:

The database is the foundation of the Plant E-commerce Website, and the system is able to save, categorize, and manage the data in the database. Given that e-commerce applications are sensitive in relationships with user credentials, payment status and product information, it is paramount to consider accuracy, security and scalability of the database. The database is made to manage non-functional and functional features of the Plant E-commerce Website.

### Functional Requirements

These specify the operations that the database will have to support:

- **User Management:** Store and authenticate user accounts, role (user/admin).
- **Product Management:** keep product information (name, price, image).
- **Cart Management:** Store items which have been added by a user, and you can edit or delete products.
- **Order Handling:** Documents final purchases with order information (item, quantity, total price, payment status).
- **Message Handling:** Receive and save customer queries or customer feedbacks posted through the contact form.

### Non-Functional Requirements

The quality qualities of the database system include:

- **Performance Performance:** The queries are expected to perform well regardless of the amount of data.
- **Scalability:** The database must be able to support future expansion which may be in terms of more users, products or orders.
- **Security:** Sensitive information (e.g., passwords) has to be stored safely (hashed/ encrypted).
- **Reliability:** Data integrity should be maintained in transactions (e.g. one cannot make orders without a valid user).
- **Maintainability:** The schema must be readily maintained or expanded when new functionality is introduced.

## 3.2 Entity Relationship Diagram (ERD):

The majority of the main entities, their attributes, and relationships between them are visually presented in the Entity-Relationship Diagram (ERD) of the Plant Shop system. This diagram is essential to learn the structure and the interconnection of data in the system.

### Entities and Attributes

- **Users:** The entity of users contains information on registered users and administrators. Its fields are the unique id (primary key), name, email (unique) and password, user_type (user or admin) and creation and update timestamps (created_at, updated_at).

- **Products:** Products entity holds information about all products in the shop. Some of the main features are id (primary key), name, price, and image.

- **Cart:** The Cart object is an object which holds the things that customers have put in their cart to buy. Each record will be identified by the id (primary key), user_id that connects them to the Users record, product information (name, price, picture), and quantity. Although some of the product information is saved in duplicate form to be displayed, the user_id is the foreign key, which refers to the Users table.

- **Orders:** Orders are where final purchases are registered. Orders are unique in the sense that they have a unique identifier (primary key) and a user_id connecting it to the user who made the order. It contains also the customer information (name, number, email) the method of payment, address, total_products (string list of ordered products), total_price, placed_on timestamp, payment_status. This schema would store the products in form of a string, whereas in a fully normalized design, individual products would be represented in an order_items table.
- **Message:** Message is a storage of user communications. Attributes are id (primary key), user_id (Foreign key to Users), name, email, number and the actual message contents.

### Relationships

- **Users → Orders**: A single user may have several orders and thus there is a one to many (1 → ∞) associations between a user and an order.
- **Users → Cart**: A user may have many items in the cart, also, one-to-many (1 → ∞) relationship.
- **Users → Message**: A user may send multiple messages, creating another one-to-many (1 → ∞) link.
- **Products → Cart**: A product may be presented in the carts of numerous users and, therefore, this is one-to-many (1 → ∞) relationship.
- **Orders → Order_Items:** A single order can include multiple products, forming another one-to-many (1 → ∞) relationship.
- **Products → Order_Items:** Each product can appear in multiple order items, establishing another one-to-many (1 → ∞) link.

This ERD makes the interaction between users, products, carts, orders, order items, and messages well-visualized and conceptualized. It provides consistency of the data, appropriate relational integrity and effective order and product information management in the Plant Shop system.

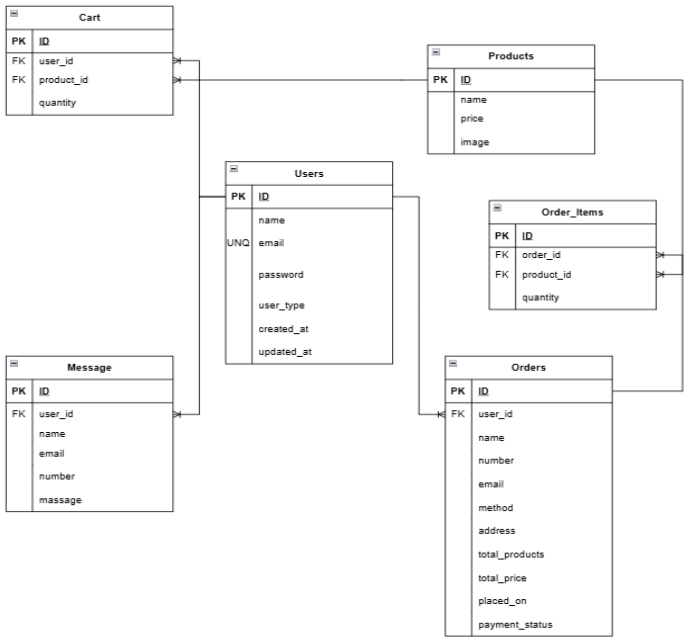


***Fig 11: Entity Relationship Diagram***

## 3.3 Normalization:

In order to get an effective data management and to remove redundancy in the Plant Shop database (store_db), the database was normalized to the Third Normal Form (3NF).

In the beginning, the tables were users, products, cart, orders and message. There were also some problems that were highlighted, including the redundancy of storing product information on the cart table and the presence of several products in one string in the orders table.

**First Normal Form (1NF):** Tables were all updated so that they are atomic, and no repeating groups. The cart table was changed to store the productid instead of having the product information stored directly. The same table like the orders table was also divided by the introduction of an order_items table which will store the individual products in each order, which will replace multi-value fields.

**Second Normal Form (2NF):** The partial dependencies were eliminated by making sure that all the non-key attributes are completely dependent on the primary key. The productid and user_id in the cart table make the combination of the user_id and productid a unique identifier of an entry. The records in the order_items table are dependent on order_id and productid.

**Third Normal Form (3NF):** Dependency transitivity was done away with. An example of such is the user contact details including email and phone that were not included in the orders table because they are already stored in the users table so that all the non-key fields are dependent on the primary key.

**Resulting Normalized Structure:**

- **Users:** id, name, email, password, user_type, created_at, updated_at
- **Products:** id, name, price, image
- **Cart:** user_id, product_id, quantity (composite primary key)
- **Orders:** id, user_id, method, address, total_price, placed_on, payment_status
- **Order_Items:** id, order_id, product_id, quantity, price
- **Messages:** id, user_id, message, created_at

## 3.4 Table structures:

This part is the section where the detailed structure of any table in the database (store_db) of the Plant Shop will be demonstrated. Every table is defined with its attributes, data types, constraints and their purpose in the system. These structures were created to guarantee data consistency, data integrity and economic data storage.

### 1. Users Table

The Users table stores information about all registered users and administrators of the system. It maintains essential user credentials, roles, and timestamps for account management.

**Table 1: Structure of Users Table with Description**

| Column | Type | Constr. | Desc. |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique user ID |
| name | VARCHAR(100) | NOT NULL | Full name |
| email | VARCHAR(100) | UNIQUE, NOT NULL | Email address |
| password | VARCHAR(255) | NOT NULL | Hashed password |
| user_type | ENUM | Default 'user' | User/admin role |
| created_at | TIMESTAMP | Default CURRENT_TIMESTAMP | Creation time |
| updated_at | TIMESTAMP | ON UPDATE CURRENT_TIMESTAMP | Last update |

## 2. Products Table

The Products table stores details of all items available for purchase in the shop. Each product has a unique identifier and associated pricing and image information.

**Table 2: Structure of User Table with Description**

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each product |
| name | VARCHAR(100) | NOT NULL | Name of the product |
| price | DECIMAL(10,2) | NOT NULL | Price of the product |
| image | VARCHAR(255) | NOT NULL | Image path or URL of the product |

## 3. Cart Table

The Cart table keeps track of items that users add for potential purchase. Each entry links a user to a specific product along with the quantity selected.

**Table 3: Structure of  Cart Table with Description**

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique cart entry identifier |
| user_id | INT | FOREIGN KEY → Users(id) | User who added the item |
| product_id | INT | FOREIGN KEY → Products(id) | Product added to cart |
| quantity | INT | NOT NULL | Number of units selected |
| name | VARCHAR(100) | NULL | Product name (optional, for display) |
| price | DECIMAL(10,2) | NULL | Product price (optional snapshot) |
| image | VARCHAR(255) | NULL | Product image (optional for display) |

## 4. Orders Table

The Orders table records all finalized purchases made by users. It captures payment, delivery, and overall order information.

**Table 4: Structure of Orders Table with Description**

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique order identifier |
| user_id | INT | FOREIGN KEY → Users(id) | User who placed the order |
| method | VARCHAR(50) | NOT NULL | Payment method |
| address | VARCHAR(500) | NOT NULL | Delivery address |
| total_price | DECIMAL(10,2) | NOT NULL | Total amount of the order |
| placed_on | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Date and time of order placement |
| payment_status | VARCHAR(20) | DEFAULT 'pending' | Status of payment |

## 5. Order_Items Table

The Order_Items table resolves the many-to-many relationship between orders and products. It records each product included in an order along with quantity and the price at order time.

**Table 5: Structure of  Order_Items Table with Description**

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for the order item |
| order_id | INT | FOREIGN KEY → Orders(id) | Associated order |
| product_id | INT | FOREIGN KEY → Products(id) | Product included in the order |
| quantity | INT | NOT NULL | Number of units ordered |
| unit_price | DECIMAL(10,2) | NOT NULL | Price of the product at the time of order |

## 6. Message Table

The Message table stores all communications sent by users to the system or admin.

**Table 6: Structure of Message Table with Description**

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique message identifier |
| user_id | INT | FOREIGN KEY → Users(id) | Sender of the message |
| name | VARCHAR(100) | NOT NULL | Name of the sender |

| | | | |
|---|---|---|---|
| email | VARCHAR(100) | NOT NULL | Email of the sender |
| number | VARCHAR(12) | NOT NULL | Contact number |
| message | VARCHAR(500) | NOT NULL | Message content |

## 3.5 Database Integration with Backend:

The backend is closely connected with the database on the basis of PHP and MySQLi queries. This will enable dynamic communication between the website interface and data stored.

- **Login /Registration:** PHP queries will verify the user credentials with the users table.
- **Cart Operations:** Adding, updating and deleting items have a direct impact on cart table.
- **Checkout Orders:** Checkout of the orders but cart entries are deleted.
- **Admin Dashboard:** The administrator will be able to retrieve products, order handling and reviewing messages by direct access to the database.

This integration guarantees the real time updates, that is, whatever is done on the site is automatically updated in the database.

# Chapter 4: System Design

## 4.1 Introduction

System design is the transformation of the requirements of the project into a blueprint that specifies the way the system works, communicates and processes information. It incorporates both logical and physical design elements such as Data Flow Diagrams (DFDs), working flowcharts and system architecture. The stage is used to ensure that the plant e-commerce site is designed in such a way that it is easy to navigate, database communication is streamlined, and the user interface and interaction with backend takes place easily.

## 4.2 Data Flow Diagram (DFD)

The DFD represents how data flows through the system.

### DFD Level 0 (Context Diagram)

**External Entities**

- User
- Admin

**Process**

- Plant Shop System

**Data Store**

- MySQL Database (store_db)

**Data Flow**
a) User → Browse Products / Manage Cart / Place Orders / Send Messages → System
b) System → Stores / Retrieves Data → Database
c) System → Sends confirmations, order details, messages → User
d) Admin → Manage Products / Orders / Users → System
e) System → Stores / Retrieves Data → Database
f) System → Displays reports, order info, product lists → Admin

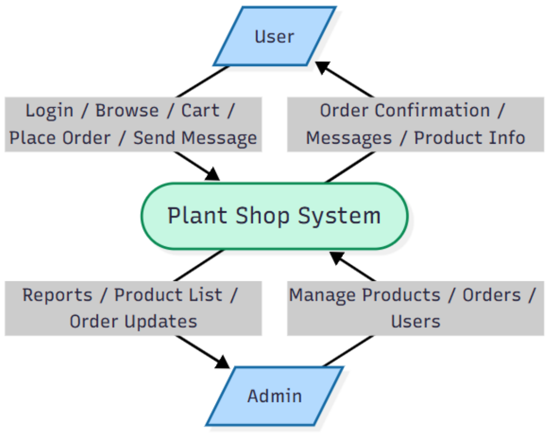

User
Login / Browse / Cart / Place Order / Send Message
Order Confirmation / Messages / Product Info
Plant Shop System
Reports / Product List / Order Updates
Manage Products / Orders / Users
Admin

*Fig 12: Context Diagram*

## DFD Level 1

**Processes**

1. **Login / Register** – User/Admin logs in.
2. **Browse Products** – User retrieves product catalog.
3. **Manage Cart** – User adds/removes items.
4. **Place Order** – User checks out and places an order.
5. **Send Message** – User submits queries/feedback.
6. **Admin - Manage Products** – Admin adds/updates/deletes products.
7. **Admin - Manage Orders** – Admin updates status and monitors orders.
8. **Admin - Manage Users** – Admin manages customer accounts.

**Data Flows**

a) User → [Login/Register] → Users DB
b) User → [Browse Products] → Products DB
c) User → [Manage Cart] ↔ Cart DB
d) User → [Place Order] → Orders DB
e) User → [Send Message] → Messages DB
f) Admin → [Manage Products] ↔ Products DB
g) Admin → [Manage Orders] ↔ Orders DB
h) Admin → [Manage Users] ↔ Users DB

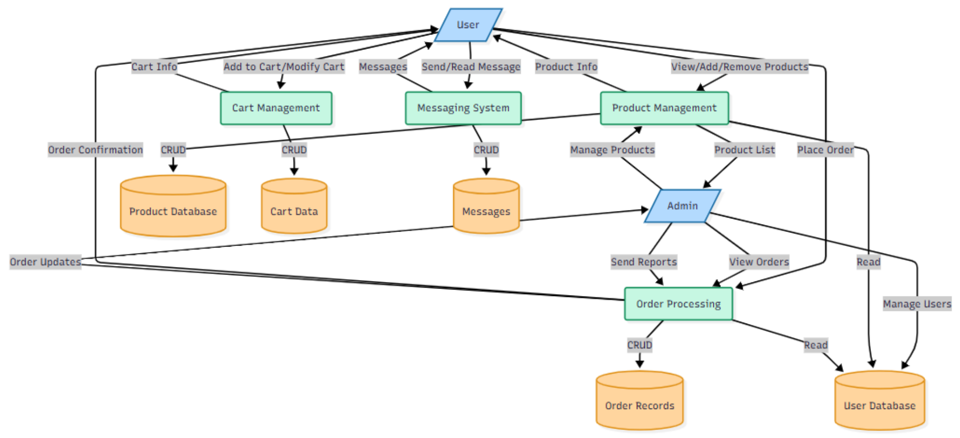


*Fig 13: Level 1 DFD*

## DFD Level 2 (Detailed Breakdown)

### (A) Place Order (User side)

1. **Select Products** – User chooses items from Products DB.
2. **Add to Cart** – Items stored in Cart DB.
3. **Checkout** – User confirms cart.
4. **Enter Shipping & Payment Info** – User provides address, method, etc.
5. **Confirm Order** – System stores order in Orders DB and updates User.

**Data Flows**
a) User → [Select Products] → Products DB
b) [Select Products] → [Add to Cart] → Cart DB
c) [Add to Cart] → [Checkout]
d) [Checkout] → [Enter Shipping & Payment Info] → Users DB
e) [Enter Shipping & Payment Info] → [Confirm Order] → Orders DB
f) [Confirm Order] → User

### (B) Manage Products & Orders (Admin side)

1. **Add / Update / Delete Products** – Changes stored in Products DB.
2. **View Orders** – Retrieves order details from Orders DB.
3. **Update Order Status** – System updates Orders DB.

**Data Flows**
a) Admin → [Add/Update/Delete Products] → Products DB
b) Admin → [View Orders] → Orders DB

c) Admin → [Update Order Status] → Orders DB
d) Orders DB → Admin

1. **User Subsystem**
    a. Register/Login → User Table.
    b. Browse/Add Plants → Product Table.
    c. Place Order → Order Table.
2. **Admin Subsystem**
    a. Manage Products (Add/Edit/Delete).
    b. Manage Users.
    c. Manage Orders.

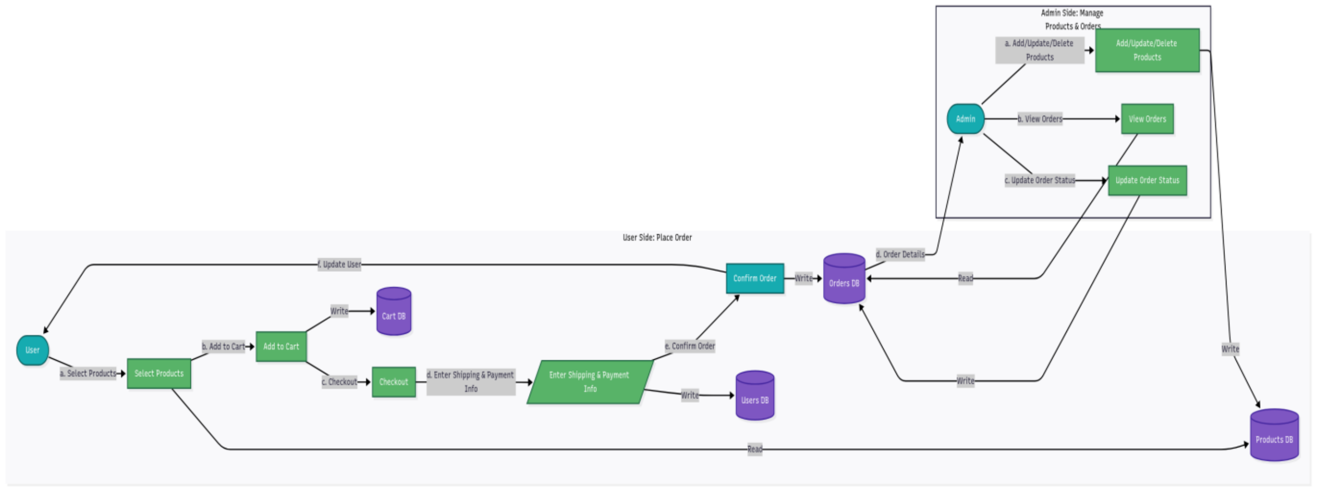


*Fig 14: Level 2 DFD*

## 4.3 Use Case Diagram

**Purpose:** Demonstrate actors, as well as actions on the high level. This assists the reader in getting the idea about functional scope and with whom he/she interacts.

**Actors:**

- **Customer (User)** - browse products, register/login, add to cart, update cart, checkout/ place order, order history, contact the administration.
- **Administrator (Admin)** - access to an area with administration, add/edit/delete products, access/edit/orders (paid/unpaid), users management, messages.

**Primary use cases (grouped):**

- Register / Login

- Browse Products / Search
- Add to Cart / Update Cart / Remove Item
- Checkout / Payment (simulated) / Place Order
- View Orders / Order History
- Contact Admin (Send Message)
- Admin: Manage Products (Create / Update / Delete)
- Admin: Manage Orders (View / Update Status / Delete)
- Admin: Manage Users / View Messages

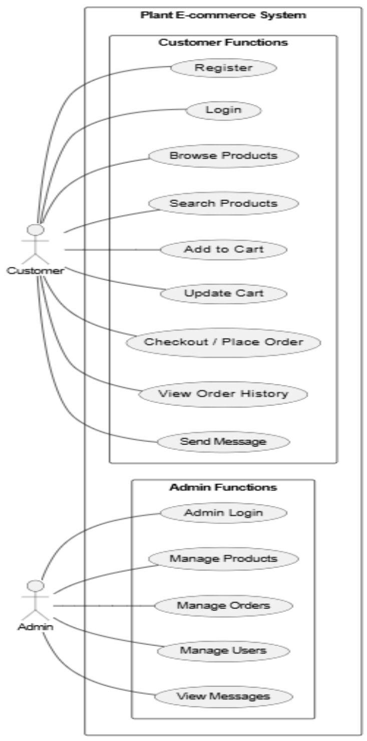


***Fig 15: Use Case Diagram: Key actors and system actions for the Plant E-commerce Website.***

## 4.4 Class Diagram

**Purpose:** Explain the key data objects, attributes and key operations. This assists in the translation of database design to the object/logic design.

**Key classes:**

- **User**
  - Attributes: userId:int, name:string, email:string, password:string, userType:string, createdAt:datetime

    - Methods: register(), login(), logout(), updateProfile()
- **Admin** *(can be modeled as a specialization/inheritance of User)*
    - Attributes: (inherits User)
    - Methods: addProduct(), updateProduct(), deleteProduct(), viewAllOrders()
- **Product**
    - Attributes: productId:int, name:string, price:decimal, imagePath:string, stock:int
    - Methods: getDetails(), updateStock()
- **CartItem**
    - Attributes: cartItemId:int, userId:int, productId:int, quantity:int
    - Methods: updateQuantity(), removeItem()
- **Order**
    - Attributes: orderId:int, userId:int, totalPrice:decimal, totalProducts:string, method:string, address:string, placedOn:datetime, paymentStatus:string
    - Methods: placeOrder(), calculateTotal(), cancelOrder()
- **OrderItem** *(recommended for normalized design — links Order and Product)*
    - Attributes: orderItemId:int, orderId:int, productId:int, quantity:int, price:decimal
- **Message**
    - Attributes: messageId:int, userId:int, name:string, email:string, number:string, messageText:string, createdAt:datetime
    - Methods: sendMessage(), deleteMessage()

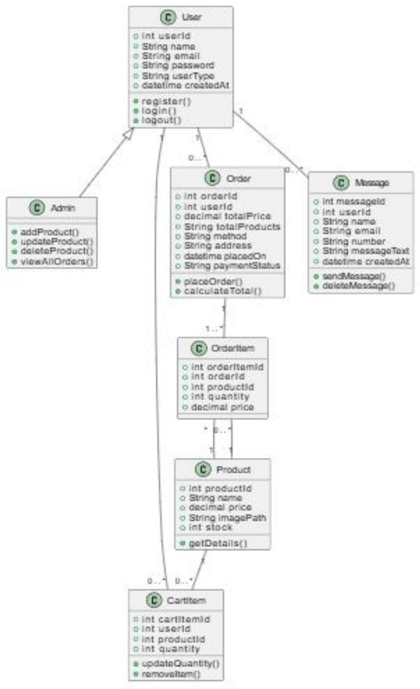


***Fig 16: Class Diagram***

**Relationships:**

- User 1 — * Order (one user makes many orders)
- User 1 — * CartItem (one user can have many cart items)
- Order 1 — * OrderItem (one order has many order items)
- Product 1 — * OrderItem (one product can appear in many orders)
- Product 1 — * CartItem (one product can be in many carts)
- User 1 — * Message (user can send many messages)
- Admin is a specialized User (inheritance)

## 4.5 Working Flowchart

Flowchart is a logical illustration of how the project works.

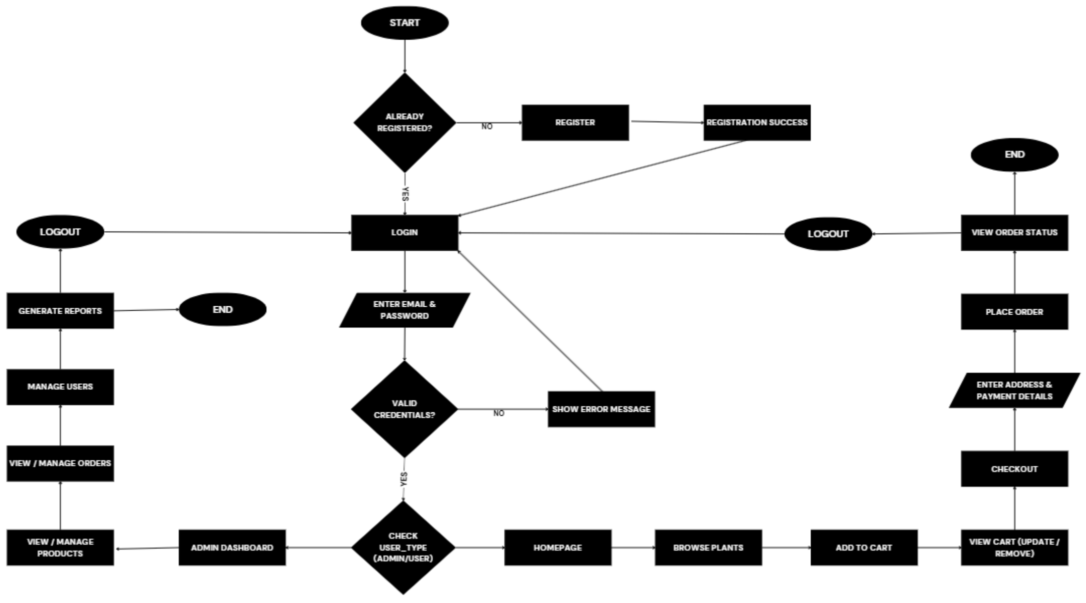


*Fig 17: Working Flowchart*

## 4.6 Database Design

The system database will be configured so as to provide efficiency in storing and retrieving information.

**Main Tables:**

1. **Users Table** – Holds the user information (ID, name, email, password, user type).
2. **Products Table** – This table will hold product details (ID, name, category, price, stock, image).
3. **Orders Table** – Records the orders (order ID, user ID, product ID, quantity, status).
4. **Messages Table** – Stores messages posted through the contact form.

**Relationships:**

- There are numerous orders that can be made by one user.
- A single order may contain several products.
- Admin is able to control all three (users, products, orders).

## 4.7 User Interface Design

The UI is designed to be simple, responsive, and user-friendly. Major pages include:

- **Homepage** – shows price tagged plants.
- **Login & Registration Page** – Handles authentication.
- **Cart Page** – User can view items that are selected with the option to delete/edit.
- **Order Page** – Checks out and confirming orders.
- **Admin Dashboard** – Gives the management features to the users, products and orders
- **Contact Page** – Allows the users and a administrator to communicate.

# Chapter 5: System Implementation

## 5.1 Introduction

This chapter provides the real process of creation and realization of the various elements of the Plant E-commerce Website. Although the previous chapters have discussed design, architecture and planning this section defines how the designs were converted into working modules by means of coding, testing and integration. The project was developed step by step with the first step being the database along with the user authentication system followed by the frontend interface, product management and lastly the admin panel and handling of orders.

## 5.2 Module-wise Implementation

The site has been split in several functional modules that had a particular task to do:

1. **User Authentication Module (login.php & register.php)**
    a. Manages user registration and safe log in.
    b. The implementation of sessions was used to preserve the user identity across pages.
    c. Hashing techniques of storing passwords.
2. **Home Page and Informational Pages (home.php, about.php, contact.php)**
    a. Showed warm greetings, advertisement areas and plant details.
    b. Contact page enabled the user to send messages, and they were registered in the database.
3. **Product Management Module (shop.php, product display)**
    The store management of the store can be well achieved using a number of ways.
    a. Displayed all existing plants in the database including their pictures, names, and prices.
    b. Inbuilt "Add to Cart" feature.
4. **Shopping Cart & Checkout Module (cart.php, checkout.php)**
    a. Granted access to users to preview and edit the chosen items or delete products.
    b. Checkout received the details of delivery and payment methods and stored orders in the database.
5. **Admin Module (admin_page.php and related files)**
    a. Ensured that the administrator had an interface to control products, track user orders and review messages.
    b. Only available to users who are in a position of the admin.

## 5.3 Backend Feature Implementation

1. **Database Connection (config.php):** Database connection was centralized so that all files related to the same MySQL database were connected.
2. **Session Management:** Guaranteed that the user would be staying in one or more pages until the time when he/she logged out.
3. **Dynamic Product Handling:** The products were required to be fetched out of the database and displayed dynamically and also bound to the cart system.
4. **Form Handling:** The validation of all input forms (login, registration, contact and checkout) was done both on the client (JavaScript) and server (PHP) level.

## 5.4 Code Quality & Structure

In order to be readable and scalable, the code had a modular design:

1. **Separation of Concerns:**
   - header.php and footer.php added on account of layout uniformity.
   - CSS, JS stored in external files (style.css, script.js) to be managed.
2. **Error Handling:**
   - SQL queries in error-checking statements.
   - Friendly error messages shown to the users rather than uncoded database errors.
3. **Naming Conventions:**
   - Variable and file names are descriptively named.
   - Tables in the database that are normalized to prevent redundancy.

## 5.5 Security Features

Even though it was of light weight, there were a number of security features that were included to secure the system:

- **Password Protection:** User passwords are coded and then stored.
- **SQL Injection Prevention:** SQL queries are made in a form that minimizes risks.
- **Session Control:** Only allowed to view such pages as checkout.php or admin_page.php in case a user was logged in.
- **Input validation:** Email, phone number and text were all checked to ensure that wrong or malicious data would not be entered.

# Chapter 6: Testing & Debugging

## 6.1 Testing Strategy

The testing was an essential step of the project to make sure that the Plant E-commerce Website worked correctly and provided the user-friendly experience. The strategy of the multi-level testing was implemented, which includes the following approaches:

- **Unit Testing:** This is concerned with testing of specific units like the check of the validity of a login, product additions and cart updates. This contributed to the detection of problems at the early stage of development.
- **Integration Testing:** Tested the connection between modules- e.g. determining whether the registration process properly entered the data into the database and whether it was possible to log in.
- **System Testing:** Tested the whole site, to ensure that all the features such as browsing of the products, to making of orders, etc. had been performed as expected.
- **User Acceptance Testing (UAT):** This was done to test the scenarios that are likely to occur in the real world bearing in mind that the site had to be user-friendly and match the standards of an end-user.

This stratified method guaranteed that the small parts as well as the entire system were put to test.

## 6.2 Test Cases & Results

Some of the most important test cases that have been conducted, the results to be expected and the actual result are summarized in the following table:

**Table 7: Table of Test Cases with Results**

| Test Case | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| User Registration | Register with valid details (unique email) | Account created and stored in database | Successfully registered | Pass |
| Duplicate Registration | Register with existing email | Error message: "Email already exists" | Correct error displayed | Pass |
| Login (Valid) | Login with correct email & password | Redirect to home.php or admin_page.php | Redirected successfully | Pass |
| Login (Invalid) | Login with incorrect credentials | Error: "Incorrect email or password" | Error displayed correctly | Pass |
| Add Product (Admin) | Admin adds a product with all fields filled | Product saved in database and displayed on homepage | Works as expected | Pass |

| | | | | |
|---|---|---|---|---|
| Add Product (Missing Fields) | Product added without name/price | Error message shown | Proper validation error | Pass |
| Cart Functionality | User adds product to cart | Cart updates with selected product | Works as expected | Pass |
| Remove from Cart | User removes an item from cart | Cart updates accordingly | Works as expected | Pass |
| Place Order | Checkout with valid details | Order stored in database, confirmation shown | Works successfully | Pass |
| Contact Form | Submit a message with valid details | Message saved in database/admin notified | Works as expected | Pass |
| Session Expiry | Login and then close browser | Session cleared, requires re-login | Works correctly | Pass |
| SQL Injection Attempt | Enter ' OR '1'='1 in login form | Blocked, no unauthorized login | Prevented as expected | Pass |

These test cases were both functional (that is, login, cart and checkout) and security tests (that is, SQL injection prevention). The findings indicated that the system worked well in all the conditions that it was tested in.

## 6.3 Debugging & Error Handling

A few problems were met during the development and fixed with the help of debugging:

- **Database Connection Errors:** In the beginning, there were connection failures brought about by inappropriate database credentials. This was corrected by correcting the config.php settings.
- **Session Management Problems:** Sessions were not destroyed correctly when they were being logout. The problem was resolved by adding explicit session destroy.
- **Form Validation Flaws:** There were forms that had incorrect input checks. JavaScript and PHP validation layers were implemented to deal with blank fields and non-valid inputs.
- **CSS Styling Issues:** There were problems with alignment of various components of the UI between browsers. Brower developer tools provided me with debugging to resolve the problems with cross-browsers compatibility.

These steps of debugging made the end system stable, secure and consistent.

## 6.4 Performance Evaluation

The site was tested in three aspects on the basis of loading speed, responsiveness, and reliability:

- **Speed of loading:** Pages were loaded in 1-2 seconds when tested in the local server.

- **Responsiveness:** The site was responsive to various screen sizes (desktop, laptop, and mobile).
- **Database Performance:** The queries were performed efficiently even when several test entries were inserted into the product and user table.
- **Scalability:** The code structure is designed to be modular, which means that it can be expanded in the future with the addition of a new product or a user base.

## 6.5 Limitations & Future Improvements

Although the project reached its major goals, the following constraints were noticed:

- **No Online Payment Gateway:** The system does not have any online payment solutions at the moment, and the orders are placed manually only.
- **Minimal Search Process:** The search and filter options are restricted and can be improved to facilitate products searching.
- **Minor Security Improvements:** SQL injection is already deployed but more security (e.g. hashed passwords, SSL encryption) can be added.
- **None Cloud Hosting:** The system is locally hosted using XAMPP. It would be made available all over the world because it would be deployed on a live server.

**Future Improvements:**

- Install safe online payment technologies, i. e. PayPal, Stripe, or mobile banking.
- Improve the filtering and search system (price filters, plant types, availability filters).
- Enhance UI/UX using modern frameworks (e.g. React, Bootstrap) to give it a more refined appearance.
- Include the support of the multi-language to make the site reach more people.
- Add email/SMS confirmation on orders, shipments or offers.
- Real-world deployment can be made accessible on a cloud-based hosting (e.g., AWS, Heroku) to be scalable.
- Add an analytics dashboard to help the administration monitor the sales trends, user behavior.

# Chapter 7: Project Management & Participation

## 7.1 Development Methodology

The construction of the Plant E-commerce Website was based on the hybrid approach, sticking to both Waterfall and Agile.

**Waterfall Approach:** The initial phases of the project, including requirement gathering, problem identification, system design and database design were done in a sequential fashion. One step was done thoroughly and then another step was proceeded to. Indicatively, the coding phase was preceded by the completion of the ER diagram and database schema to take a firm base.

**Agile Approach:** When coding began, it became necessary to implement Agile principles to introduce the aspect of flexibility and iterative development. Small modules were created to include the features of user login, display of the products, shopping cart and order management. The modules were tested as soon as they were implemented and amendments done where appropriate. This was a cycle of develop-test-improve till all the key features were developed.

## 7.2 Timeline & Gantt Chart

It was structured into weekly phases of the project, which guaranteed a systematic progress. Frequent meetings with the course supervisor served to keep the work in line with expectations in academic terms.

- Week 1-2: Problem identification, requirement analysis and project planning.
- Week 3: Database design and creation of ER diagram.
- Week 4-6: Frontend development (HTML, CSS, JavaScript).
- Week 7-9: Backend development (PHP, MySQL integration).
- Week 10: The creation of the admin panel and the implementation of features.
- Week 11: Testing, debugging and performance evaluation.
- Week 12: Final presentation and documentation and report writing.

**Supervisor Feedback and Meetings.**

- **Meeting 1 (Week 2):** Discussed project proposal and accepted to go ahead. Feedback aimed at the definition of functional vs non-functional requirements.
- **Meeting 2 (Week 4):** Presented ER diagram, database schema. Normalization and validation rules were proposed by the supervisor.

- **Meeting 3 (Week 6):** Showed first frontend design. Feedback on making it more user-friendly and having a consistent layout.
- **Meeting 4 (Week 9):** Demonstrated working backend with product features and a login. Handling of errors and security checks advised by supervisor.
- **Meeting 5 (Week 11):** Final review on the entire system was done. Feedback was concerned with small things, flow between modules and readiness to the presentation.

## 7.3 Team Roles & Contributions

This project was conducted on a case-by-case basis. This is why I had to handle all positions and duties:

- **Frontend Development:** Using HTML, CSS, and JavaScript the layout, styling and interactivity of the website were designed and implemented.
- **Backend Development:** PHP coded user authentication scripts, product management, shopping cart, and order processing scripts.
- **Database Design:** Designed and maintained the MySQL database, which has adequate relationships and normalization.
- **Testing & Debugging:** Validation, performance tested and removed bugs both on the client and server side.
- **Documentation:** Developed project report, created diagrams and prepared support documentation

## 7.4 Attendance & Participation

Ensuring that there was active participation was a significant contributor to the success of this project. I was present in every planned project development meeting and session with the supervisor. The feedbacks of every meeting were constructive and had direct impact on the quality of the final system.

For example:

- Database normalization recommendations in Week 3 were a way to prevent redundancy.
- Week 6 Week 6 had improved usability through UI feedback.
- Week 11 was spent on final refinements to have a polished presentation.

# Chapter 8: Documentation & Report

## 8.1 Objectives & Design Rationale

This project was written down to provide the entire picture of the development process starting with the initial idea up to the final implementation. Documenting the project properly will allow the project to be comprehended by others, recreated where necessary, and expanded in future. It is also an individual document of the obstacles encountered, and strategies implemented in the process of growth.

The design rationale provides the reasoning behind all the significant technology and tools choices:

**PHP & MySQL (Backend):** They are selected because of their simplicity, reliability, and good integration with one another. PHP is effective in processing dynamic content and database communication, and MySQL provides structured storage of data.

**HTML, CSS, JavaScript (Frontend):** Chosen to plan a user-friendly interface which should be responsive. These foundation web technologies made sure that the system had a modern appearance, device compatibility, and interactivity.

**XAMPP:** It is an open-source project that is based on Apache, PHP and MySQL, which is configured as a local server environment. This allowed easy testing of the site in the process of development without external server.

**GitHub (Version Control):** It is used to keep the track of the progress, archive of the backups, and various versions of the project code. It was also the collaborative and organizational tool although the project was made on an individual basis.

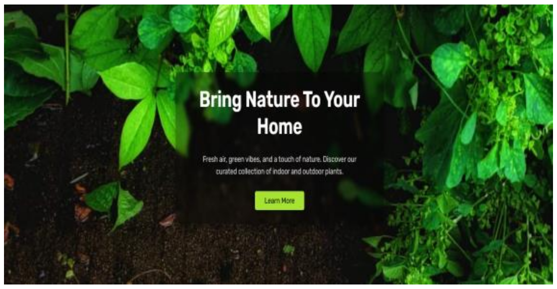
These tools combined to make the system scalable, manageable and efficient enabling easy transition between coding and deployment.

## 8.2 Screenshots & Results

Screenshots of various modules of the site were captured to show how the system works as well as its design. These screenshots bring out the experience on the front end to users and the backend to administrators.

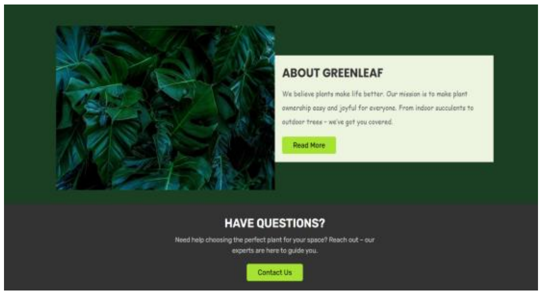
Some examples include:

- **Homepage:**


Bring Nature To Your Home

Fresh air, green vibes, and a touch of nature. Discover our curated collection of indoor and outdoor plants.

Learn More


ABOUT GREENLEAF

We believe plants make life better. Our mission is to make plant ownership easy and joyful for everyone. From indoor succulents to outdoor trees – we've got you covered.

Read More

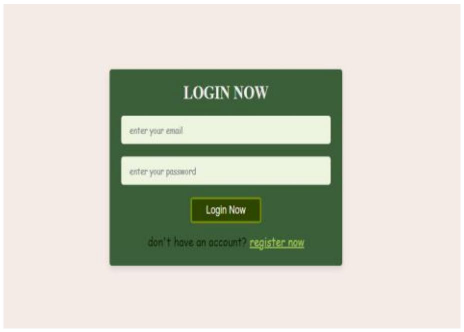
HAVE QUESTIONS?

Need help choosing the perfect plant for your space? Reach out – our experts are here to guide you.

Contact Us

*Fig 18: Screenshot of Hero Section and About Section of the Homepage*

- **User Login & Registration Forms:**

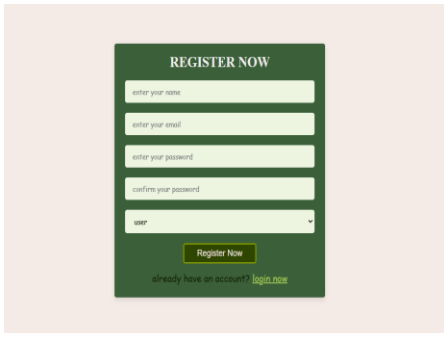

LOGIN NOW

enter your email

enter your password

Login Now

don't have an account? register now


REGISTER NOW

enter your name

enter your email

enter your password

confirm your password

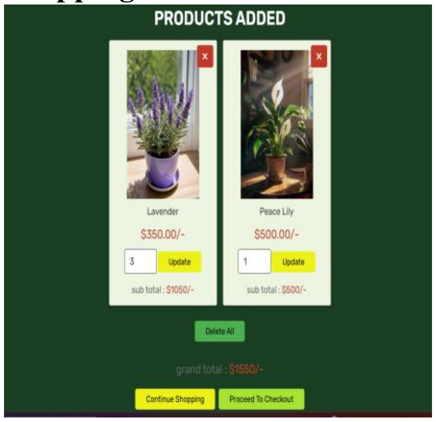
user

Register Now

already have an account? login now

*Fig 19: Screenshot of Login & Registration forms*

- **Shopping Cart & Order Placement Page:**

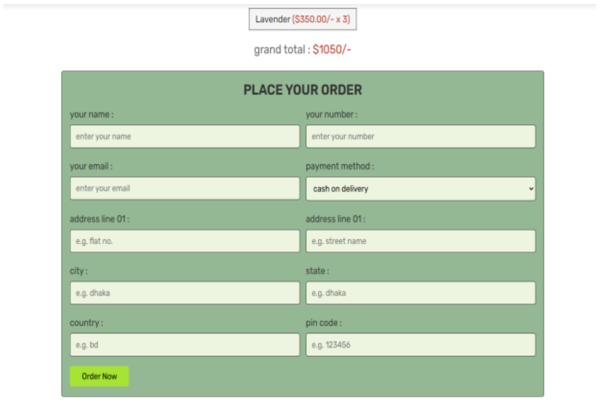

PRODUCTS ADDED

| | |
|---|---|
| Lavender | Peace Lily |
| $350.00/- | $500.00/- |
| 3 Update | 1 Update |
| sub total : $1050/- | sub total : $500/- |

Delete All

grand total : $1550/-

Continue Shopping | Proceed To Checkout


Lavender ($350.00/- x 3)

grand total : $1050/-

PLACE YOUR ORDER

| | |
|---|---|
| your name : enter your name | your number : enter your number |
| your email : enter your email | payment method : cash on delivery |
| address line 01 : e.g. flat no. | address line 01 : e.g. street name |
| city : e.g. dhaka | state : e.g. dhaka |
| country : e.g. bd | pin code : e.g. 123456 |

Order Now

*Fig 20: Screenshot of Cart & Orders page*

- **Shop Page:**

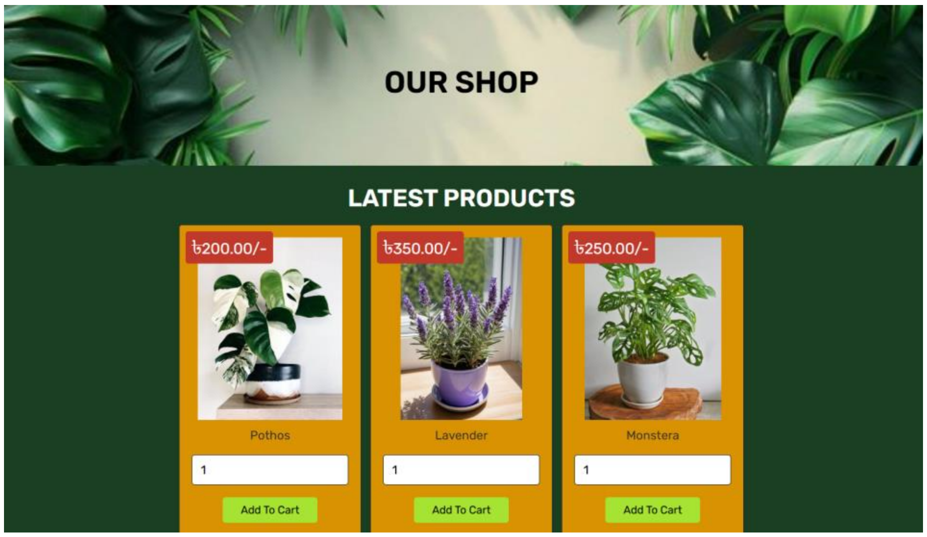
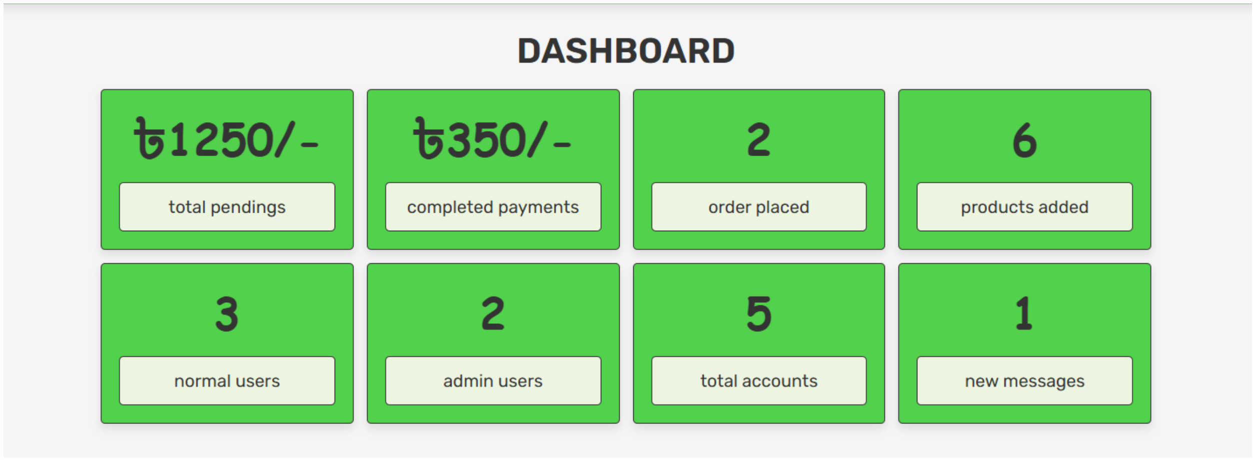


*Fig 21: Screenshot of Shop Page*

- **Admin Dashboard:**

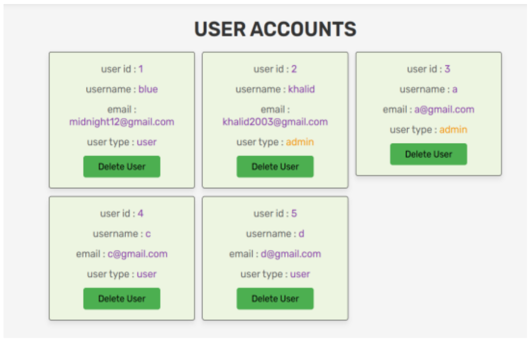


DASHBOARD

| | | | |
|---|---|---|---|
| ৳1250/- | ৳350/- | 2 | 6 |
| total pendings | completed payments | order placed | products added |
| 3 | 2 | 5 | 1 |
| normal users | admin users | total accounts | new messages |

*Fig 22: Screenshot of Admin Dashboard*

- **Admin Users & Admin Orders Form:**



USER ACCOUNTS

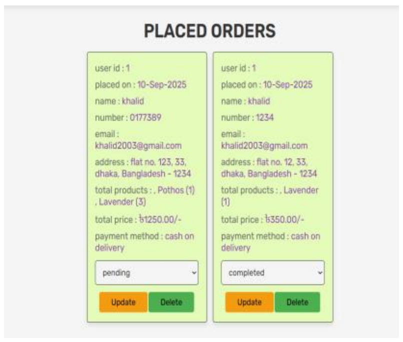
user id : 1
username : blue
email : midnight12@gmail.com
user type : user
Delete User

user id : 2
username : khalid
email : khalid2003@gmail.com
user type : admin
Delete User

user id : 3
username : a
email : a@gmail.com
user type : admin
Delete User

user id : 4
username : c
email : c@gmail.com
user type : user
Delete User

user id : 5
username : d
email : d@gmail.com
user type : user
Delete User



PLACED ORDERS

user id : 1
placed on : 10-Sep-2025
name : khalid
number : 0177389
email : khalid2003@gmail.com
address : flat no. 123, 33, dhaka, Bangladesh - 1234
total products : , Pothos (1) , Lavender (3)
total price : ৳1250.00/-
payment method : cash on delivery
pending
Update Delete

user id : 1
placed on : 10-Sep-2025
name : khalid
number : 1234
email : khalid2003@gmail.com
address : flat no. 12, 33, dhaka, Bangladesh - 1234
total products : , Lavender (1)
total price : ৳350.00/-
payment method : cash on delivery
completed
Update Delete

***Fig 23: Screenshot of Admin Users & Admin Order Page***

## 8.3 Conclusion

Plant E-commerce Website is a project that I have been happy to develop and has incorporated both theory and practice. The project is a good example of how technological advances in web technology can be incorporated to develop a fully functional e-commerce platform.

Through this system:

- It gives customers the advantage of having an easy shopping experience and going through plants and buying them easily.
- A powerful management panel is provided to the administrators making it easy to handle products, monitor users, and track orders.
- The backend database will provide security in storing and retrieving sensitive user and order information.

To recapitulate on the project objectives, I can say with certainty that they were met. Its functionality was found not only to fulfill the essential purposes as the display of the products on the site, cart system, and handling of orders, but also helped to position the site right in the future development.

Possible enhancement opportunities may be:

- Incorporation of online payment gateways on real transactions.
- Improving user experience by adding advanced search and filtering.
- Enhance security, including encrypting of data and 2-factor authentication.

Summing up, the project can also be seen as an indication of the significance of planning, choosing the right tools, and developing them progressively. Not only does it focus on my technical development, but it also illustrates how documentation, feedback of supervisors, and systematic approach can all be useful in ensuring a successful project deliverable.

# Chapter 9: References

[1] W. S. Jawadekar, *Software Engineering: Principles and Practice*, 5th ed. New Delhi, India: Tata McGraw-Hill, 2019.

[2] D. Flanagan, *JavaScript: The Definitive Guide*, 7th ed. Sebastopol, CA: O'Reilly Media, 2020.

[3] B. Duckett, *HTML & CSS: Design and Build Websites*, 2nd ed. Indianapolis, IN: Wiley, 2014.

[4] M. Nixon, *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*, 5th ed. Sebastopol, CA: O'Reilly Media, 2020.

[5] XAMPP Project, "XAMPP – Apache Friends," [Online]. Available: https://www.apachefriends.org/index.html. [Accessed: Sep. 24, 2025].

[6] MySQL Documentation, "MySQL 8.0 Reference Manual," [Online]. Available: https://dev.mysql.com/doc/. [Accessed: Sep. 24, 2025].

[7] X. Yuan and E. B. Fernandez, "Patterns for Business-to-consumer E-Commerce Applications," *arXiv*, 2011. [Online]. Available: https://arxiv.org/abs/1108.3342. [Accessed: Sep. 24, 2025].
*Referenced in:* E-commerce design patterns and system architecture.

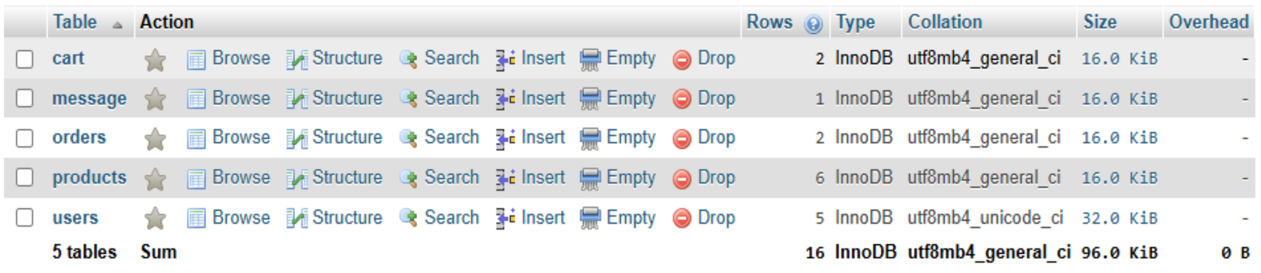
[8] A. K. Luhach, S. K. Dwivedi, and C. K. Jha, "Designing a logical security framework for e-commerce system based on SOA," *arXiv*, 2014. [Online]. Available: https://arxiv.org/abs/1407.2423. [Accessed: Sep. 24, 2025].
*Referenced in:* Security framework considerations for the e-commerce platform.

# Chapter 10: Appendices

## Appendix A: Source Code / GitHub Link

**Source Code:** https://github.com/HKR-XV-20-R3/Plant-ecom-project

## Appendix B: Additional Screenshots



| | Table | Action | | | | | | | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | cart | ★ | Browse | Structure | Search | Insert | Empty | Drop | 2 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | message | ★ | Browse | Structure | Search | Insert | Empty | Drop | 1 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | orders | ★ | Browse | Structure | Search | Insert | Empty | Drop | 2 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | products | ★ | Browse | Structure | Search | Insert | Empty | Drop | 6 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | users | ★ | Browse | Structure | Search | Insert | Empty | Drop | 5 | InnoDB | utf8mb4_unicode_ci | 32.0 KiB | - |
| | 5 tables | Sum | | | | | | | 16 | InnoDB | utf8mb4_general_ci | 96.0 KiB | 0 B |

*Fig 24: Database of All Data Table*

## Appendix C: User Manual

This section gives the users a short description on how to use the Plant E-commerce Website. The interface is also made easy and user-friendly so that users and administrators find it easy to manage orders, products, and accounts.

- **Browse Plants:** It allows users to view what plants are availed on the homepage or use the search box to locate a certain plant. On clicking a plant, one will see all the details such as price, description, and care tips.
- **Add to Cart:** When a plant has been chosen, you may add it to your shopping cart by selecting the Add to Cart button. The quantities can be updated, or the items can be removed by the users in the cart.
- **Checkout:** Once the cart is reviewed, then checkout where delivery and payment method can be selected. Confirm the purchase to make the purchase. A confirmation message is created when the system is submitted successfully.
- **User Registration and Login:** New users are able to register themselves by completing the registration form with their name, email, password and contact information. Shopping features are accessible through the use of credentials by the registered users. Admins who log in will be redirected to the admin dashboard.
- **Admin Dashboard:** Admins have an opportunity to operate with products, orders, and users. The dashboard shows the total number of products, the number of outstanding orders and registered users. Admins are able to add and edit or delete plants, change the status of orders and monitor the activity of users.
- **Export Data (Admin):** Admins will be able to export product and order data to maintain records and report on it.