**FarmFolio**

*Mini Project Report*

*Submitted by*

**RohithR Nair**

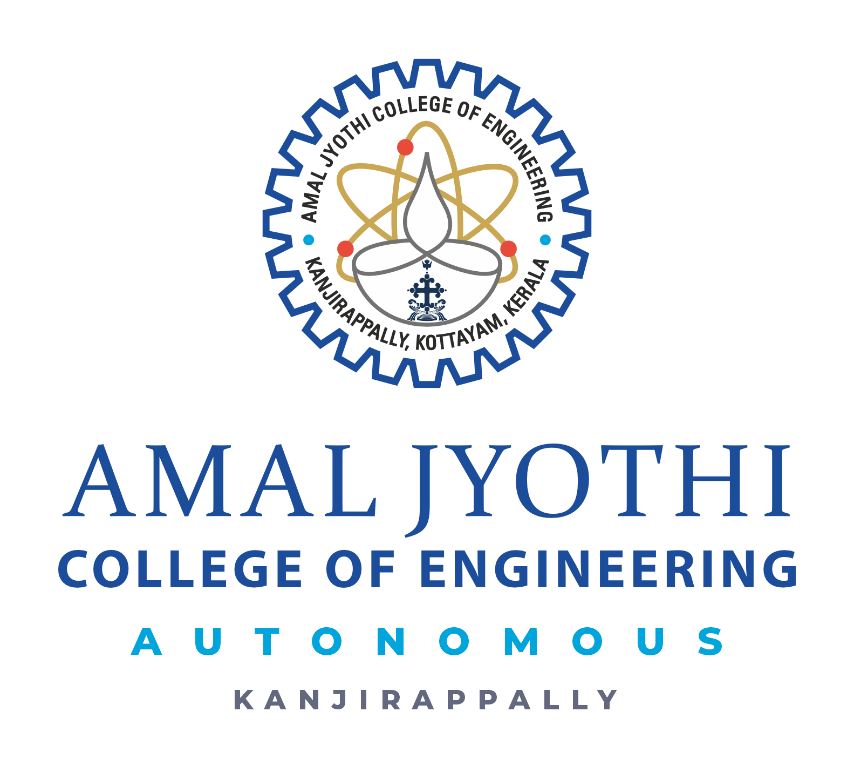
**Reg. No.: AJC22MCA-I050**

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

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

**(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS**

**KANJIRAPPALLY**

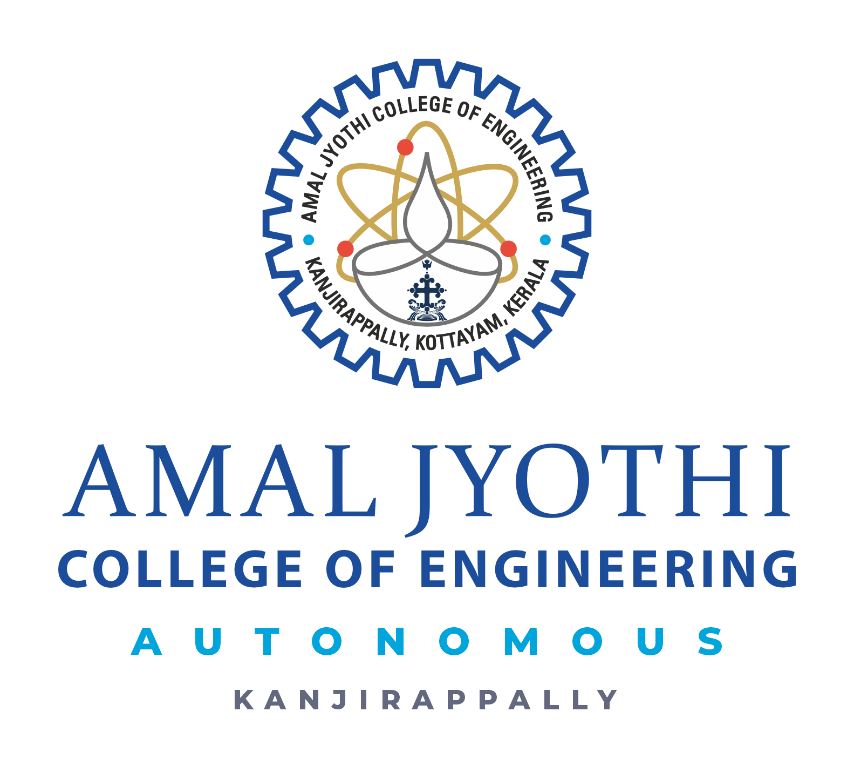
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

# 2024-2025

## DEPARTMENT OF COMPUTER APPLICATIONS

### AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**FARMFOLIO”** is the bona fide work of **ROHITH R NAIR (Regno: AJC22MCA-I050)** carried out in partial fulfillment of the requirements for the award of the **Degree of Integrated Master of Computer Applications** at **Amal Jyothi College of Engineering Autonomous, Kanjirappally,** Affiliated to **APJ Abdul Kalam Technological University**. The project was undertaken during the period from **January 01, 2025 to April 23, 2025.**

**AJITH GS MEERA ROSE MATHEW**

**Internal Guide Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

**DECLARATION**

I hereby declare that the project report **“FARMFOLIO”** is a bona fide work done at **Amal Jyothi College of Engineering Autonomous, Kanjirappally**, Affiliated to **APJ Abdul Kalam Technological University**, towards the partial fulfilment of the requirements for the award of the **Integrated Master of Computer Applications (INMCA)** during the period from **January 01, 2025 to April 23, 2025.**

**Date: ROHITH R NAIR**

**KANJIRAPPALLY Reg: AJC22MCA-I050**

# ACKNOWLEDGEMENT

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

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **MEERA Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ajith GS** for his inspiration and helping hand.

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

ROHITH R NAIR

# ABSTRACT

Farmfolio is an innovative web platform designed to bridge the gap between farm owners and consumers by creating a direct and user-friendly interface. The platform allows farm owners to list their farms, provide details about their products, and share contact information, enabling consumers to connect with them directly. Unlike conventional marketplaces, Farmfolio emphasizes simplicity and transparency, excluding chat functionality while providing direct phone numbers for quick and efficient communication.

**Four types of users:**

* Farm Owners – Who can register their farms, list available products, and update their offerings.
* Consumers – Who can browse through farms, contact owners, and access fresh products.
* Delivery Boys - Who are responsible for delivering the products ordered by consumers from the farms. They can receive delivery requests, track orders, and update the status of deliveries.
* Admin – Responsible for managing platform activities and ensuring smooth operations.

**Farmfolio also introduces two standout features:**

• Farm Ratings and Reviews: Empower consumers to rate and review farms, helping others make informed choices and fostering trust.

• Farm Events: Enable farm owners to promote activities such as farm visits, workshops, and local markets, enriching the consumer experience and supporting community engagement.

The platform is developed using HTML, CSS, JavaScript, PHP and MYSQL ensuring a robust and responsive interface. By streamlining the connection between farms and consumers, Farmfolio aims to support sustainable agricultural practices, promote local businesses, and provide fresh, high-quality items to communities.

**CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL. NO** | | **TOPIC** | **PAGE NO** | |
| **1** | | **INTRODUCTION** |  | |
| **1.1** | | **PROJECT OVERVIEW** |  | |
| **1.2** | | **PROJECT SPECIFICATION** |  | |
| **2** | | **SYSTEM STUDY** |  | |
| **2.1** | | **INTRODUCTION** |  | |
| **2.2** | | **EXISTING SYSTEM** |  | |
| **2.3** | | **DRAWBACKS OF EXISTING SYSTEM** |  | |
| **2.4** | | **PROPOSED SYSTEM** |  | |
| **2.5** | | **ADVANTAGES OF PROPOSED SYSTEM** |  | |
| **3** | | **REQUIREMENT ANALYSIS** |  | |
| **3.1** | | **FEASIBILITY STUDY** |  | |
| **3.1.1** | | **ECONOMICAL FEASIBILITY** |  | |
| **3.1.2** | | **TECHNICAL FEASIBILITY** |  | |
| **3.1.3** | | **BEHAVIORAL FEASIBILITY** |  | |
| **3.1.4** | | **FEASIBILITY STUDY QUESTIONNAIRE** |  | |
| **3.2** | | **SYSTEM SPECIFICATION** |  | |
| **3.2.1** | | **HARDWARE SPECIFICATION** |  | |
| **3.2.2** | | **SOFTWARE SPECIFICATION** |  | |
| **3.3** | | **SOFTWARE DESCRIPTION** |  | |
| **3.3.1** | | **PHP** |  | |
| **3.3.2** | | **MYSQL** |  | |
| **4** | | **SYSTEM DESIGN** |  | |
| **4.1** | | **INTRODUCTION** |  | |
| **4.2** | | **UML DIAGRAM** |  | |
| **4.2.1** | | **USE CASE DIAGRAM** |  | |
| **4.2.2** | | **SEQUENCE DIAGRAM** |  | |
| **4.2.3** | | **STATE CHART DIAGRAM** |  | |
| **4.2.4** | | **ACTIVITY DIAGRAM** |  | |
| **4.2.5** | | **CLASS DIAGRAM** |  | |
| **4.2.6** | | **OBJECT DIAGRAM** |  | |
| **4.2.7** | | **COMPONENT DIAGRAM** |  | |
| **4.2.8** | | **DEPLOYMENT DIAGRAM** |  | |
| **4.2.9** | | **COLLABORATION DIAGRAM** |  | |
| **4.3** | | **USER INTERFACE DESIGN USING FIGMA** |  | |
| **4.4** | | **DATABASE DESIGN** |  | |
| **5** | | **SYSTEM TESTING** |  | |
| **5.1** | | **INTRODUCTION** |  | |
| **5.2** | | **TEST PLAN** |  | |
| **5.2.1** | **UNIT TESTING** | |  |
| **5.2.2** | **INTEGRATION TESTING** | |  |
| **5.2.3** | **VALIDATION TESTING** | |  |
| **5.2.4** | **USER ACCEPTANCE TESTING** | |  |
| **5.2.5** | **AUTOMATION TESTING** | |  |
| **5.2.6** | **SELENIUM TESTING** | |  |
| **6** | **IMPLEMENTATION** | |  |
| **6.1** | **INTRODUCTION** | |  |
| **6.2** | **IMPLEMENTATION PROCEDURE** | |  |
| **6.2.1** | **USER TRAINING** | |  |
| **6.2.2** | **TRAINING ON APPLICATION SOFTWARE** | |  |
| **6.2.3** | **SYSTEM MAINTENANCE** | |  |
| **7** | **CONCLUSION & FUTURE SCOPE** | |  |
| **7.1** | **CONCLUSION** | |  |
| **7.2** | **FUTURE SCOPE** | |  |
| **8** | **BIBLIOGRAPHY** | |  |
| **9** | **APPENDIX** | |  |
| **9.1** | **SAMPLE CODE** | |  |
| **9.2** | **SCREEN SHOTS** | |  |

## List of Abbreviations

# CHAPTER 1

# INTRODUCTION

### PROJECT OVERVIEW

### Farmfolio is a web platform designed to connect farm owners directly with consumers, enabling them to browse farms, access fresh products, and contact owners without intermediaries. The platform features four user roles—Farm Owners, Consumers, Delivery Personnel, and Admin—each with specific functionalities to ensure seamless operations. Key features include farm ratings and reviews for trust-building and farm event promotions to enhance community engagement. Built using HTML, CSS, JavaScript, PHP, and MySQL, Farmfolio streamlines farm-to-consumer interactions, supports local agriculture, and promotes sustainable farming practices.

### PROJECT SPECIFICATION

* **Product & Farm Management** – Farm owners can register, list products, update availability, and promote farm events, while consumers can browse and connect directly.
* **Order & Delivery System** – Delivery personnel receive and track delivery requests, ensuring efficient order fulfillment and real-time status updates.
* **Ratings & Reviews** – Consumers can rate and review farms to foster trust and help others make informed choices.
* **Technology Stack** – Developed using HTML, CSS, JavaScript for the frontend, PHP for backend logic, and MySQL for database management, ensuring a robust and scalable system.

# CHAPTER 2

# SYSTEM STUDY

# 

### INTRODUCTION

Farmfolio is a web-based platform designed to bridge the gap between farm owners and consumers by providing a direct and efficient marketplace for fresh farm products. The platform eliminates intermediaries, allowing consumers to connect with farm owners and purchase directly. Additionally, it includes features for delivery management, farm event promotions, and farm ratings, ensuring a streamlined experience.

### EXISTING SYSTEM

### Currently, farm owners rely on traditional marketplaces, social media, or third-party e-commerce platforms to sell their products. These methods often involve high commission fees, lack of transparency, and limited consumer engagement. Additionally, communication between buyers and farm owners is inefficient, leading to delays and reduced trust.

**2.2.1 NATURAL SYSTEM STUDIED**

The natural system observed is the conventional farm-to-consumer model, where buyers visit local farms, farmers’ markets, or rely on middlemen to purchase fresh produce. This system, while organic and community-driven, lacks scalability, convenience, and accessibility for a broader audience.

**2.2.2 DESIGNED SYSTEM STUDIED**

Several existing online agricultural marketplaces were studied, including e-commerce platforms that allow farmers to list their products. However, most of these systems involve complex interfaces, commission-based transactions, or lack direct communication between buyers and farm owners. Additionally, few platforms focus on farm events and local engagement.

### DRAWBACKS OF EXISTING SYSTEM

 High dependency on middlemen, increasing costs for consumers.

 Lack of a centralized, user-friendly platform for farm owners to manage sales and events.

 Limited trust-building mechanisms such as verified ratings and reviews.

 Inefficient order fulfillment and delivery tracking systems.

 Poor direct communication between consumers and farm owners.

### PROPOSED SYSTEM

Farmfolio aims to resolve these issues by introducing a user-friendly platform where farm owners can list their farms and products, and consumers can browse and contact them directly. Delivery personnel ensure efficient order fulfillment, and the platform supports farm event promotions, enhancing consumer engagement. The system eliminates chat functionality in favor of direct contact via phone numbers for quick communication.

### ADVANTAGES OF PROPOSED SYSTEM

* **Direct Farm-to-Consumer Connection** – Eliminates intermediaries, reducing costs and ensuring fresh produce.
* **Enhanced Trust & Transparency** – Features like farm ratings and reviews build credibility.
* **Efficient Order & Delivery System** – Dedicated delivery personnel streamline logistics.
* **Community Engagement** – Farm event promotions encourage local participation and awareness.
* **User-Friendly & Scalable** – Simple interface built with HTML, CSS, JavaScript, PHP, and MySQL for scalability and ease of use.

# CHAPTER 3

# REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

### Economical Feasibility

### Technical Feasibility

### Behavioral Feasibility

**3.1.4 Feasibility Study Questionnaire**

## SYSTEM SPECIFICATION

### Hardware Specification

### Processor - Dual core processor of higher

RAM - 4GB or higher

Hard disk - Minimum 100GB HDD/SDD

### Software Specification

### Front End - HTML,CSS,JAVASCRIPT,AJAX

### Back End - PHP

### Database - MSQL

### Client on PC - Windows 7 or above

### Technologies used - jS, HTML5, BOOTSTRAP, PHP, CSS

## SOFTWARE DESCRIPTION

### PHP

PHP (Hypertext Preprocessor) is a server-side scripting language widely used for web development. It allows developers to create dynamic web pages and applications by embedding PHP code within HTML. PHP is particularly well-suited for building web applications that interact with databases, making it a suitable choice for the back end of the Canteen Management System.

### MySQL

MySQL is an open-source relational database management system (RDBMS) known for its reliability and performance. It is commonly used for storing and managing structured data in web applications. MySQL is compatible with various programming languages and platforms, making it a popular choice for database management in web development projects like the Canteen Management System.

# CHAPTER 4

# SYSTEM DESIGN

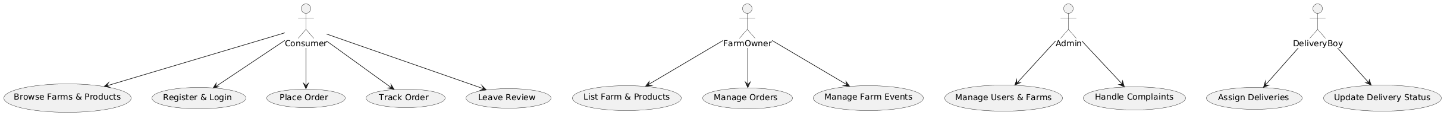
* 1. **INTRODUCTION**

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

## UML DIAGRAM

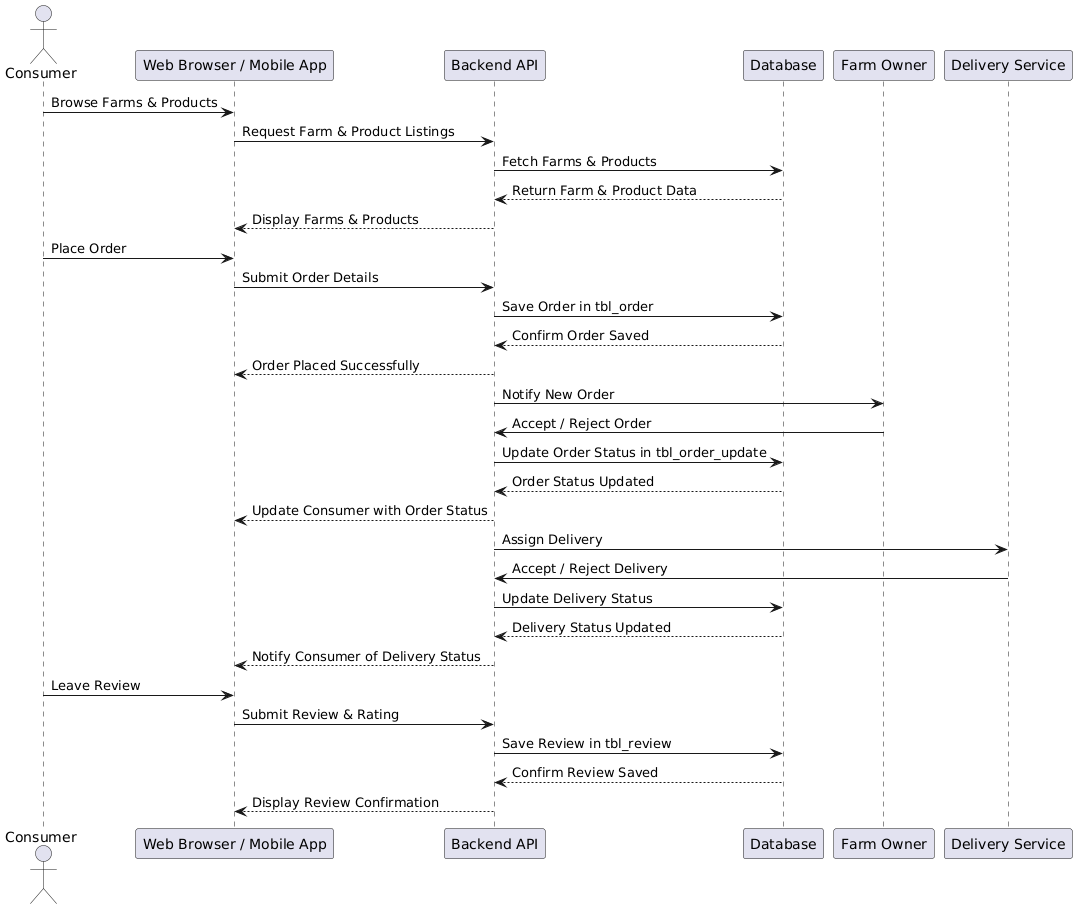
UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. UML stands for Unified Modeling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.

## USE CASE DIAGRAM

This diagram is used to capture the functional requirements of a system from a user's perspective. It consists of actors (users or external systems) and use cases (functions or services provided by the system). Actors are connected to use cases to represent their interactions. Use case diagrams are helpful for understanding the system's overall functionality and the roles different users play.

**4.2.2 SEQUENCE DIAGRAM**

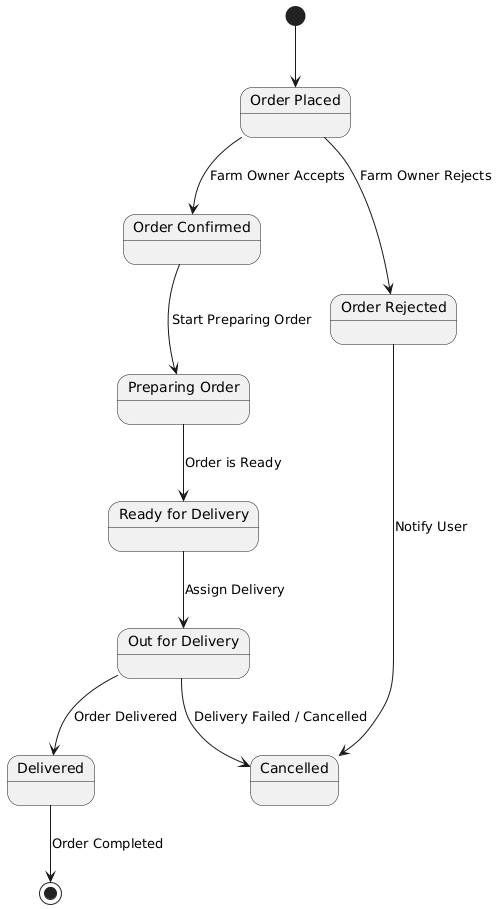
Sequence diagrams visualize the interactions between objects or components in a system over time. They show the order of messages exchanged between objects, including the timing and dependencies of these messages. Sequence diagrams are particularly useful for understanding the dynamic behavior of a system, especially in scenarios where multiple objects collaborate to accomplish a task.



## 4.2.3 State Chart Diagram

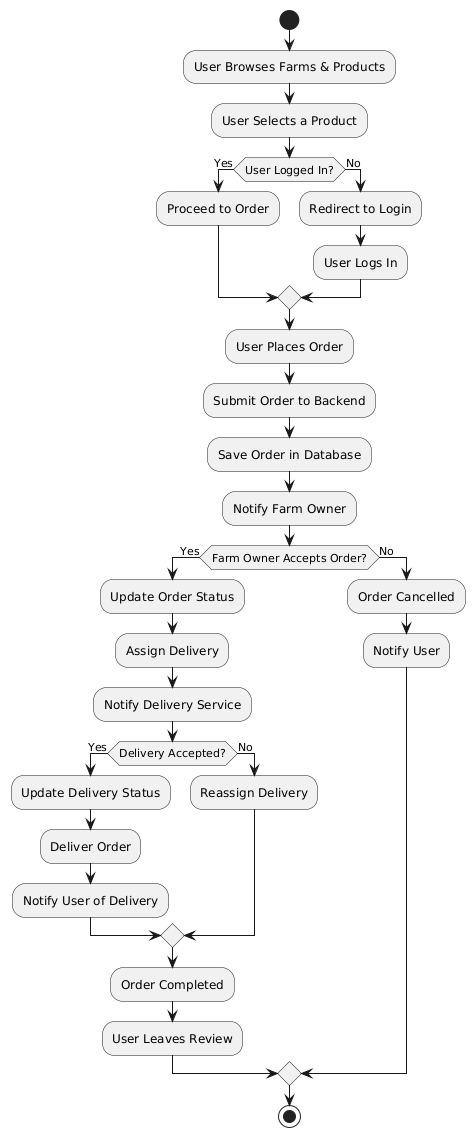
State chart diagrams serve as a vital tool in the arsenal of software engineers and systems analysts, offering a structured way to conceptualize and model the dynamic behavior of objects or entities within a system. These diagrams effectively break down complex systems into manageable components, illustrating the different states that these components can assume and the transitions between them. By visually depicting the possible states and transitions, state chart diagrams help stakeholders grasp the intricacies of system behavior, facilitating a deeper understanding of how the system responds to various inputs, events, and conditions.

One of the key strengths of state chart diagrams lies in their ability to capture not just the static structure of a system, but also its dynamic behavior over time. This temporal dimension allows developers to anticipate and plan for different scenarios, ensuring that the system behaves predictably and robustly under diverse conditions. Whether used in the early stages of system design to explore different architectures and behaviors or as a reference during implementation and testing phases, state chart diagrams serve as invaluable tools for creating software systems that are not only functional but also adaptable and resilient in the face of change.



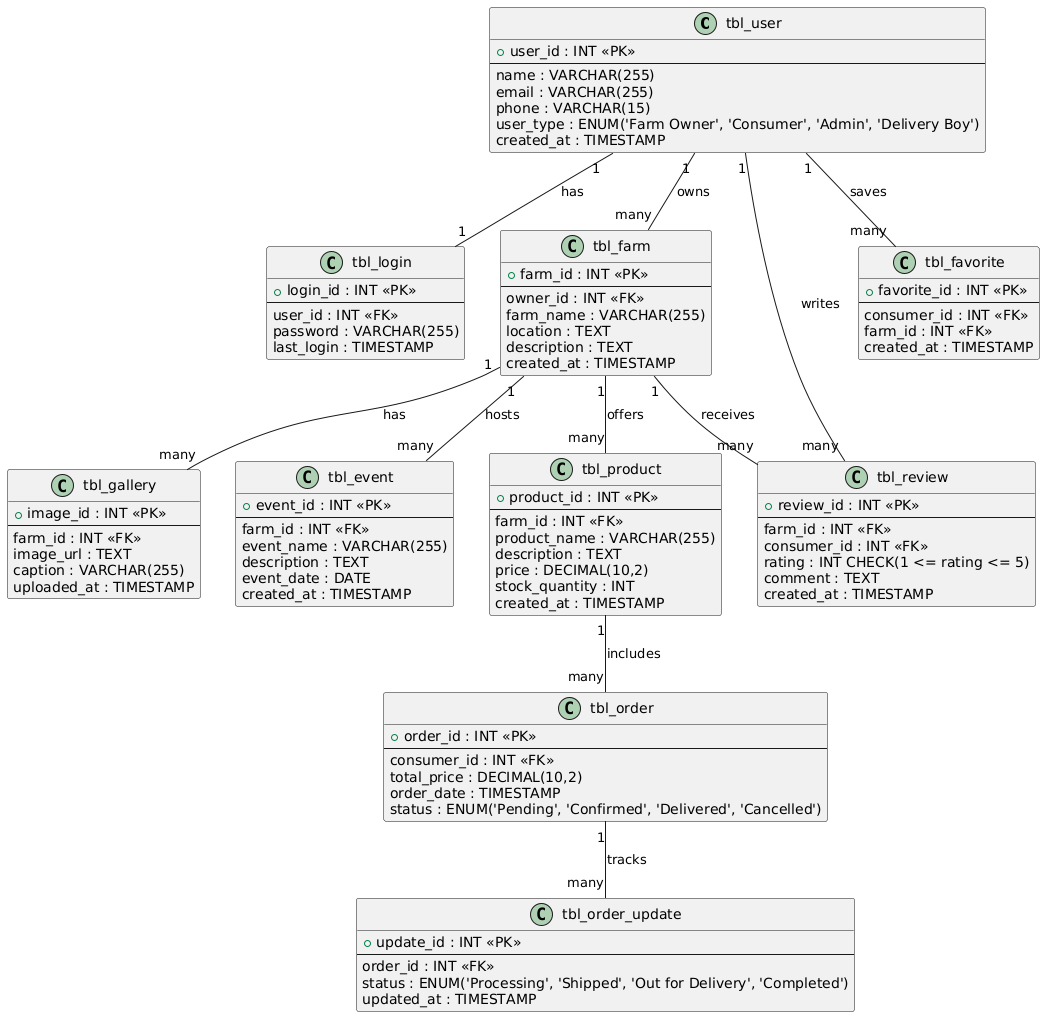
**4.2.4 ACTIVITY DIAGRAM**

Activity diagrams represent the flow of activities or processes within a system. They consist of nodes representing actions, decision points, and control flows connecting these nodes to indicate the sequence of activities. Activity diagrams are commonly used for modeling business processes.



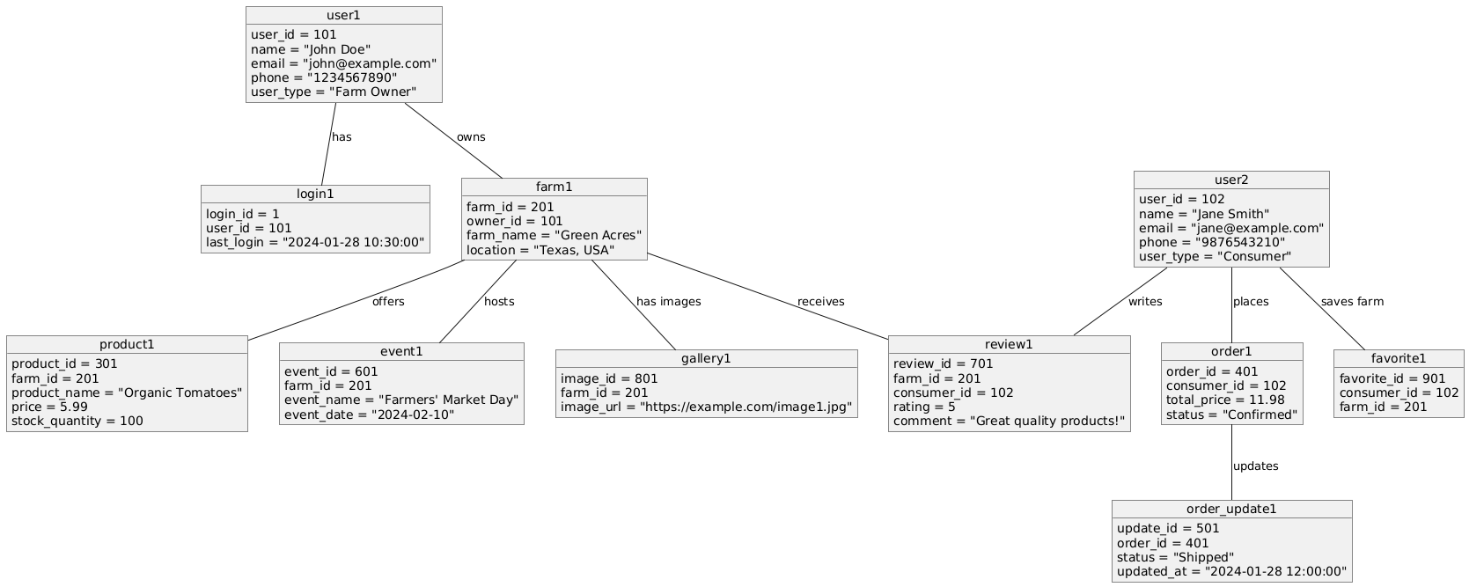
**4.2.5 CLASS DIAGRAM**

Class diagrams provide a static view of a system's structure by depicting the classes, attributes, methods, and relationships between classes. They show the blueprint of the system's objects and how they interact with each other. Class diagrams are fundamental for object-oriented analysis and design, serving as a foundation for other diagrams and implementation.

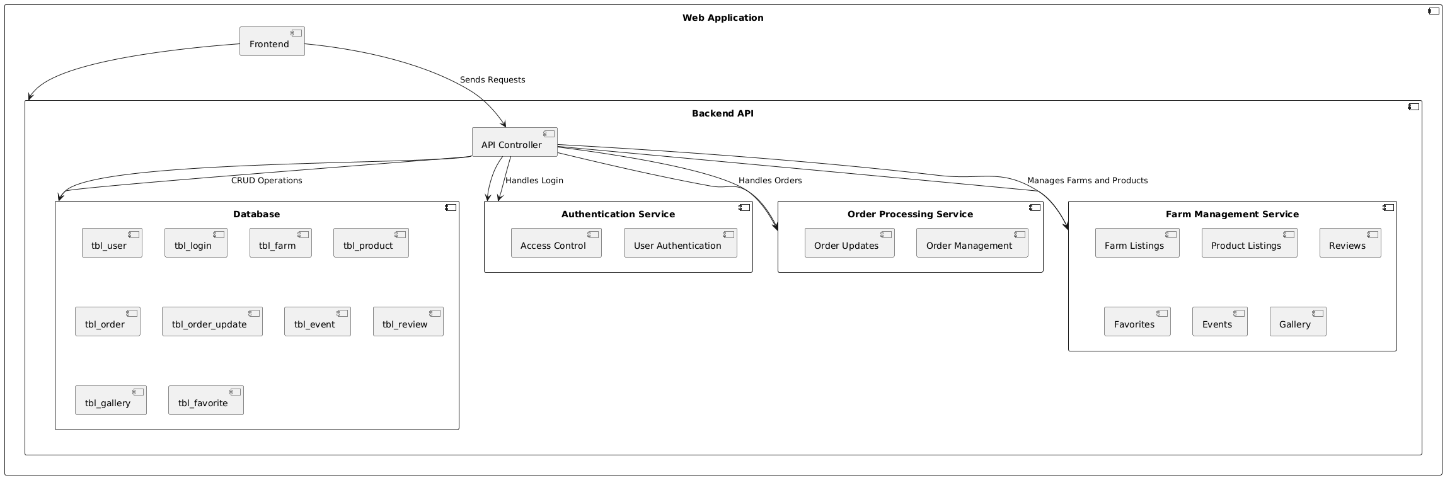
****

4.2.6 OBJEXT DIAGRAM

Object diagrams serve as visual representations that encapsulate the specific state of a system by illustrating instances of classes and their relationships at a particular moment in time. This approach provides a concrete and tangible view of the system’s structure, enabling stakeholders to verify the accuracy of class diagrams and gain a deeper understanding of system behavior through practical examples. By showcasing objects and their interactions, object diagrams serve as invaluable tools for developers, architects, and stakeholders alike, facilitating discussions, clarifying design decisions, and aiding in the identification of potential design flaws or inconsistencies. Moreover, object diagrams play a crucial role in the software development lifecycle by serving as documentation artifacts that capture system configurations, dependencies, and constraints, thus ensuring alignment between conceptual models and implementation details throughout the development process.



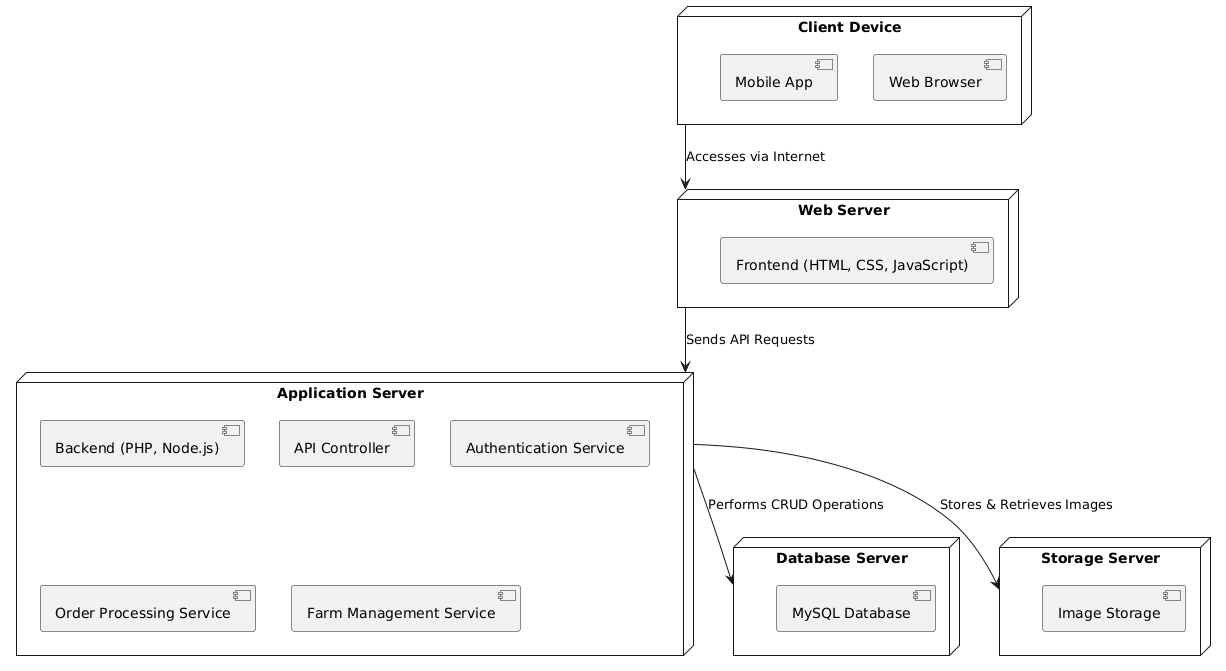
**4.2.7 COMPONENT DIAGRAM**

Component diagrams provide a visual representation of the relationships between various components, including their interfaces, dependencies, and interactions. By highlighting these connections, stakeholders can gain insights into the flow of data, control, and communication within the system, aiding in the identification of potential bottlenecks or areas for optimization. Additionally, component diagrams can serve as a blueprint for software development teams, guiding the division of labor and facilitating collaboration by clearly defining each component’s responsibilities and interfaces. Furthermore, these diagrams can be used to assess the impact of changes or updates to individual components, helping to minimize the risk of unintended consequences during system evolution. Overall, component diagrams play a crucial role in system design and development, serving as a bridge between high-level architectural concepts and concrete implementation details.

**4.2.8 DEPLOYMENT DIAGRAM**

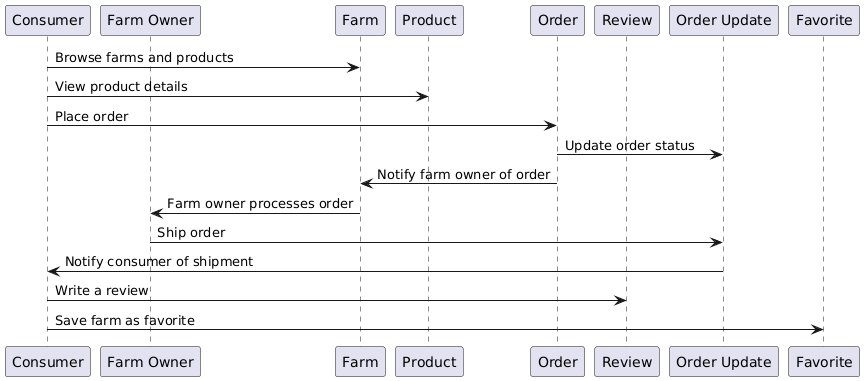
Deployment diagrams depict the physical deployment of software components on hardware nodes, such as servers, computers, or devices. They show how software artifacts are distributed across the hardware infrastructure and how they communicate with each other. Deployment diagrams are essential for understanding the system's deployment architecture, including scalability, reliability, and performance considerations.

Explanation, Diagram



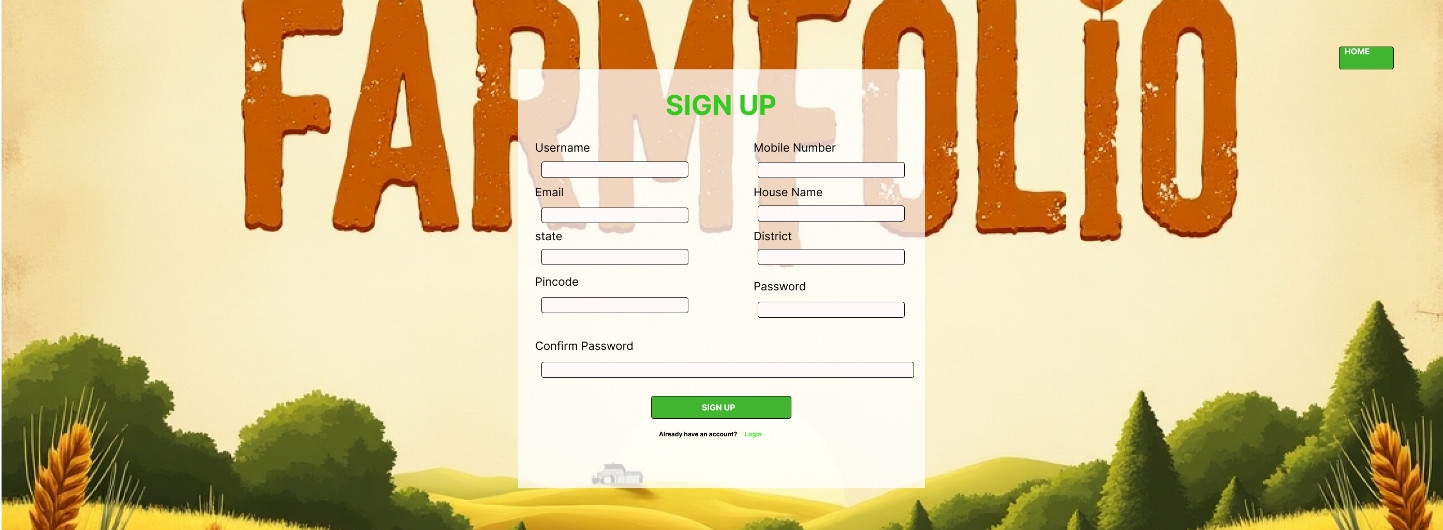
**4.2.9 COLLABORATION DIAGRAM**

Collaboration diagrams, also known as communication diagrams, illustrate the interactions between objects or components in a system, emphasizing the messages exchanged between them rather than the sequence of events over time. They focus on the structural organization of objects and the connections between them to achieve specific tasks or functions. Collaboration diagrams are particularly useful for modeling real-time systems or systems with concurrent behavior.

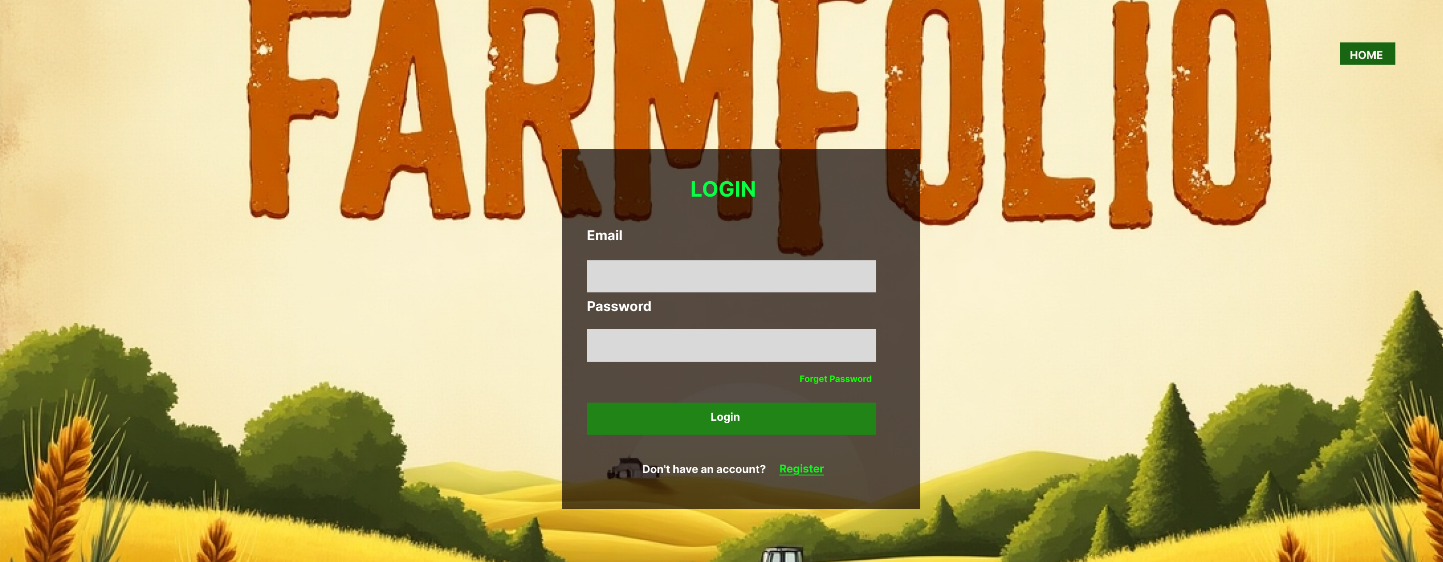


## 4.3 USER INTERFACE DESIGN USING FIGMA

**Form Name: SIGNUP**



**Form Name: LOGIN**



## 4 DATABASE DESIGN

### 4.4.1 Relational Database Management System (RDBMS)

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the Data base. This ensures both Referential and Entity Relationship Integrity. A domain D is set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

### 4.4.2 Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys, and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form. As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These

include:

✓ Normalize the data

✓ Choose proper names for the tables and columns.

✓ Choose the proper name for the data.

**First Normal Form**

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute.

### 

### Table1

### Conversion to first normal form

### 

### Second Normal Form

### A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

### 

### Table1

### Conversion to second normal form

### Third Normal Form

### A relation will be in 3NF if it is in 2NF and no transition dependency exists.

### 

### Table 1 Conversion to Third normal form

### 

### 

### 

### 4.4.3 Sanitization

An automated procedure called "sanitization" is used to get a value ready for use in a SQL query. This process typically involves checking the value for characters that have a special significance for the target database. To prevent a SQL injection attack, you must sanitize(filter) the input string while processing a SQL query based on user input. For instance, the user and password input is a typical scenario. In that scenario, the server response would provide access to the 'target user' account without requiring a password check.

**4.4.4 Indexing**

By reducing the number of disk accesses needed when a query is completed, indexing helps a database perform better. It is a data structure method used to locate and access data in a database rapidly. Several database columns are used to generate indexes. The primary key or candidate key of the table is duplicated in the first column, which is the Search key. To make it easier to find the related data, these values are kept in sorted order. Recall that the information may or may not be kept in sorted order.

### 4.5 TABLE DESIGN

### 1 .tbl\_signup

Primary key: **userid**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Field name** | **Datatype (Size)** | **Key Constraints** | **Description of the field** |
| 1 | userid | Int(11) | PRIMARY KEY NOT NULL | Unique identifier for each user |
| 2 | username | varchar(50) | NOT NULL | Username for user |
| 3 | mobile | varchar(11) | NOT NULL | Mobike numbem |
| 4 | email | varchar(100) | NOT NULL | Email |
| 5 | house | varchar(255) | NOT NULL | House name |
| 6 | district | varchar(100) | NOT NULL | District |
| 7 | state | varchar(100) | NOT NULL | State |
| 8 | pin | Char(6) | NOT NULL | Pin |
| 9 | password | Varchar(255) | NOT NULL | Password |
| 10 | Signup\_time | datetime | NOT NULL | Store the signup date and time |

**2.tbl\_login**

Primary key: **login\_id**

Foreign key: **userid** references table **tbl\_sighup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | login\_id | int(11) | Primary Key, Not Null | Unique identifier for each login entry |
| 2 | email | varchar(255) | Not Null | User's email address for login |
| 3 | password | varchar(255) | Not Null | Encrypted password for authentication |
| 4 | type | int(11) | Not Null | User type (e.g., Admin, Consumer) |
| 5 | login\_time | timestamp | Not Null, Default current\_timestamp() | Timestamp of login entry |
| 6 | userid | int(11) | Foreign Key, Not Null | Reference to the user in tbl\_user |
| 7 | username | varchar(50) | Not Null | Username associated with the user |

**3.tbl\_farms**

Primary key: **farm\_id**

Foreign key: **user\_id** references table **tbl\_signup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | farm\_id | int(11) | Primary Key, Not Null | Unique identifier for each farm |
| 2 | user\_id | int(11) | Foreign Key, Not Null | References the owner of the farm (from tbl\_signup) |
| 3 | farm\_name | varchar(255) | Not Null | Name of the farm |
| 4 | location | varchar(255) | Not Null | Address or geographical location of the farm |
| 5 | description | text | Default NULL | Additional details about the farm |
| 6 | created\_at | timestamp | Not Null, Default current\_timestamp() | Timestamp of when the farm was added |
| 7 | status | enum('pending','active','rejected') | Not Null, Default 'pending' | Current status of the farm (pending, active, or rejected) |

**4.tbl\_category**

Primary key: **category\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | category\_id | int(11) | Primary Key, Not Null | Unique identifier for each category |
| 2 | category | varchar(255) | Not Null | Name of the main category |
| 3 | sub | varchar(100) | Not Null | Name of the subcategory |
| 4 | status | enum('0','1') | Not Null, Default '1' | Status of the category (0 = inactive, 1 = active) |

**5.tbl\_fc**

Primary key: **id**

Foreign key: **farm\_id** references table **tbl\_farms,category\_id references table tbl\_category**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL | Unique identifier for each record |
| 2 | farm\_id | INT(11) | FOREIGN KEY (References tbl\_farm(farm\_id)), NOT NULL | Refers to the farm in the farm table |
| 3 | category\_id | INT(11) | FOREIGN KEY (References tbl\_category(category\_id)), NOT NULL | Refers to the category in the category table |

**6.tbl\_products**

Primary key: **product\_id**

Foreign key: **farm\_id** references table **tbl\_farms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** | |
| 1 | product\_id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL | Unique identifier for each product |
| 2 | farm\_id | INT(11) | FOREIGN KEY (References tbl\_farms(farm\_id)), NOT NULL | Refers to the farm in the farm table |
| 3 | product\_name | VARCHAR(255) | NOT NULL | Name of the product |
| 4 | price | DECIMAL(10,2) | NOT NULL | Price of the product |
| 5 | stock | INT(11) | NOT NULL | Available stock quantity |
| 6 | description | TEXT | NULL | Product description |
| 7 | created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP, NOT NULL | Timestamp when the product was added |
| 8 | category\_id | INT(11) | FOREIGN KEY (References tbl\_category(id)), NOT NULL | Refers to the category in the category table |
| 9 | unit | ENUM('kg', 'g', 'l', 'm') | NOT NULL | Measurement unit of the product |
| 10 | status | ENUM('0', '1') | DEFAULT '0', NOT NULL | Status of the product (0 = inactive, 1 = active) |

**7.tbl\_favorites**

Primary key: **favorite\_id**

Foreign key: **user\_id** references table **tbl\_signup, farm\_id references table tbl\_farms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | favorite\_id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL | Unique identifier for each favorite entry |
| 2 | user\_id | INT(11) | FOREIGN KEY (References tbl\_users(user\_id)), NULL | Refers to the user who favorited the farm |
| 3 | farm\_id | INT(11) | FOREIGN KEY (References tbl\_farm(farm\_id)), NULL | Refers to the farm that is favorited |
| 4 | created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP, NOT NULL | Timestamp when the favorite was added |

**8.tbl\_farm\_images**

Primary key: image\_**id**

Foreign key: **farm\_id** references table **tbl\_farms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | image\_id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL | Unique identifier for each image |
| 2 | farm\_id | INT(11) | FOREIGN KEY (References tbl\_farm(farm\_id)), NULL | Refers to the farm to which the image belongs |
| 3 | path | VARCHAR(255) | NOT NULL | File path or URL of the image |

**9. tbl\_events**

Primary key: event\_**id**

Foreign key: **farm\_id** references table **tbl\_farms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | event\_id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL | Unique identifier for each event |
| 2 | farm\_id | INT(11) | FOREIGN KEY (References tbl\_farm(farm\_id)), NOT NULL | Refers to the farm hosting the event |
| 3 | event\_name | VARCHAR(255) | NOT NULL | Name of the event |
| 4 | event\_date | DATE | NOT NULL | Date of the event |
| 5 | event\_description | TEXT | NULL | Description of the event |
| 6 | created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP, NOT NULL | Timestamp when the event was created |
| 7 | updated\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, NOT NULL | Timestamp when the event was last updated |
| 8 | status | ENUM('0', '1') | DEFAULT '1', NOT NULL | Event status (0 = inactive, 1 = active) |

**10. tbl\_cart**

Primary key: cart\_**id**

Foreign key: **product\_id** references table **tbl\_products, user\_id references table tbl\_signup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | cart\_id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL | Unique identifier for each cart entry |
| 2 | product\_id | INT(11) | FOREIGN KEY (References tbl\_products(product\_id)), NULL | Refers to the product added to the cart |
| 3 | quantity | INT(11) | DEFAULT '1', NULL | Quantity of the product in the cart |
| 4 | user\_id | INT(11) | FOREIGN KEY (References tbl\_users(user\_id)), NULL | Refers to the user who added the product |
| 5 | added\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP, NOT NULL | Timestamp when the item was added to the cart |

**11. tbl\_favorites**

Primary key: favorite\_**id**

Foreign key: **farm\_id** references table **tbl\_farms , user\_id references table tbl\_signup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | favorite\_id | int(11) | Primary Key, Auto Increment | Unique ID for favorite entry |
| 2 | user\_id | int(11) | Foreign Key (Users) | ID of the user who favorited a farm |
| 3 | farm\_id | int(11) | Foreign Key (Farms) | ID of the farm that is favorited |
| 4 | created\_at | timestamp | Not Null, Default: current\_timestamp() | Timestamp of when the favorite was added |

**12 .tbl\_participants**

Primary key: participant\_**id**

Foreign key: **event\_id** references table **tbl\_events , user\_id references table tbl\_signup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| **1** | **participant\_id** | **int(11)** | **Primary Key, Auto Increment** | **Unique ID for participant entry** |
| **2** | **event\_id** | **int(11)** | **Foreign Key (Events)** | **ID of the event the user registered for** |
| **3** | **user\_id** | **int(11)** | **Foreign Key (Users)** | **ID of the user participating in the event** |
| **4** | **registration\_date** | **timestamp** | **Not Null, Default: current\_timestamp()** | **Date of registration** |
| **5** | **status** | **enum('Pending', 'Confirmed', 'Cancelled')** | **Default: 'Pending'** | **Status of the registration** |

**13 . tbl\_reviews**

Primary key: **review\_id**

Foreign key: **farm\_id** references table **tbl\_farms , user\_id references table tbl\_signup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | review\_id | int(11) | Primary Key, Auto Increment | Unique ID for a review |
| 2 | farm\_id | int(11) | Foreign Key (Farms) | ID of the farm being reviewed |
| 3 | user\_id | int(11) | Foreign Key (Users) | ID of the user who left the review |
| 4 | rating | int(11) | Not Null | Rating given to the farm |
| 5 | comment | text | Nullable | User's review comment |
| 6 | created\_at | timestamp | Not Null, Default: current\_timestamp() | Timestamp of review creation |

**13 . tbl\_orders**

Primary key: **review\_id**

Foreign key:  **user\_id references table tbl\_signup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| **1** | **order\_id** | **INT(11)** | **PRIMARY KEY, AUTO\_INCREMENT** | **Unique identifier for each order** |
| **2** | **user\_id** | **INT(11)** | **FOREIGN KEY (Users)** | **Identifies the user who placed the order** |
| **3** | **total\_amount** | **DECIMAL(10,2)** | **NOT NULL** | **Total cost of the order** |
| **4** | **order\_status** | **ENUM('pending', 'processing', 'shipped', 'delivered')** | **DEFAULT 'pending'** | **Tracks the current status of the order** |
| **5** | **order\_date** | **DATETIME** | **NOT NULL** | **The date and time when the order was placed** |
| **6** | **delivery\_address** | **TEXT** | **NULLABLE** | **Address where the order should be delivered** |
| **7** | **phone\_number** | **VARCHAR(15)** | **NULLABLE** | **Contact number for delivery** |
| **8** | **payment\_method** | **VARCHAR(10)** | **DEFAULT 'cod'** | **Payment method used (e.g., COD, online)** |
| **9** | **payment\_status** | **ENUM('pending', 'paid', 'failed')** | **DEFAULT 'pending'** | **Payment status for the order** |
| **10** | **delivery\_boy\_id** | **INT(11)** | **FOREIGN KEY (Delivery\_Boys) NULLABLE** | **Identifies the delivery person handling the order** |
| **11** | **updated\_at** | **TIMESTAMP** | **DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP** | **Tracks the last update time of the order** |
| **12** | **processing\_date** | **TIMESTAMP** | **NULLABLE** | **The time when the order was moved to "processing" status** |
| **13** | **shipped\_date** | **TIMESTAMP** | **NULLABLE** | **The time when the order was shipped** |
| **14** | **delivered\_date** | **TIMESTAMP** | **NULLABLE** | **The time when the order was delivered** |

**13 . tbl\_order\_items**

Primary key: **item\_id**

Foreign key:  **order\_id references table tbl\_orders , product\_id references table tbl\_products**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | item\_id | INT(11) | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for each item in an order |
| 2 | order\_id | INT(11) | FOREIGN KEY (Orders) | Identifies the order to which this item belongs |
| 3 | product\_id | INT(11) | FOREIGN KEY (Products) | Identifies the product in the order |
| 4 | quantity | INT(11) | NOT NULL | Number of units of the product ordered |
| 5 | price | DECIMAL(10,2) | NOT NULL | Price per unit of the product |
| 6 | subtotal | DECIMAL(10,2) | NOT NULL | Total cost for this product (quantity \* price) |

# CHAPTER 5

# SYSTEM TESTING

* 1. **INTRODUCTION**

Explanation

## TEST PLAN

Explanation

### Unit Testing

explanation

### Integration Testing

Explanation

### Validation Testing or System Testing

Explanation

### Output Testing or User Acceptance Testing

explanation.

* + 1. **Automation Testing**

explanation.

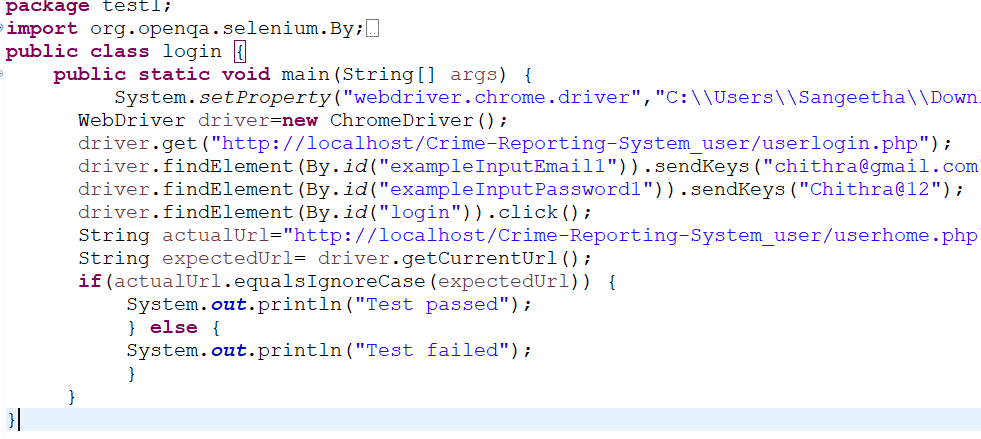
* + 1. **Selenium Testing**

explanation.

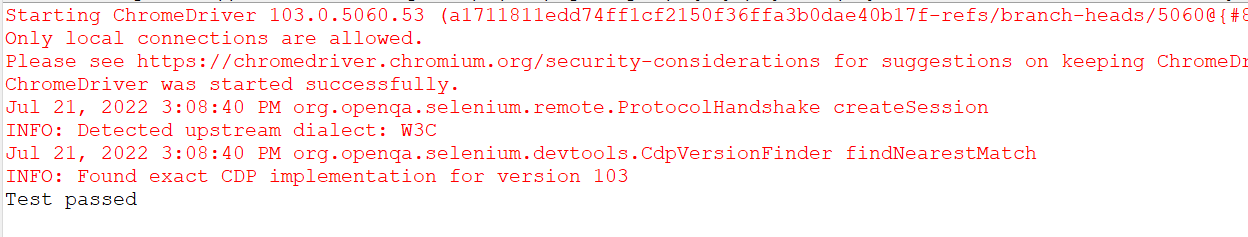
**Example:**

**Test Case 1**

**Code**



**Eg.Screenshot**



**Eg.Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name:** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID: Test\_1** | | | **Test Designed By:** | | |
| **Test Priority(Low/Medium/High):** | | | **Test Designed Date:** | | |
| **Module Name**: | | | **Test Executed By :** | | |
| **Test Title :** | | | **Test Execution Date:** | | |
| **Description:** | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |  |  |  |
|  |  |  |  |  |  |
| 6 |  |  |
| 7 |  |  |  |  |  |
|  |  |  |  |  |
| **Post-Condition:** | | | | | |

**Test Case 2:**

**Code**

**Screenshot**

**Test report**

**Minimum 4 test cases (1 login 3 functionalities)**

# CHAPTER 6

# IMPLEMENTATION

## INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working

system. It can be the most crucial stage in achieving a successful new system, gaining the users'

confidence that the new system will work effectively and accurately. Implementation is primarily

concerned with user training and documentation. Conversion usually takes place about the same

time the user is being trained or later. Implementation simply means converting a new system design

into operation, which is the process of converting a new revised system design into an operational

one.

At this stage, the main workload, the greatest upheaval, and the major impact on the existing system

shift to the user department. If the implementation is not carefully planned or controlled, it can

create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to

the new system. The new system may be a totally new one, replacing an existing manual or

automated system, or it may be a modification to an existing system. Proper implementation is

essential to provide a reliable system to meet organizational requirements. The process of putting

the developed system into actual use is called system implementation. This includes all those

activities that take place to convert from the old system to the new system. The system can be

implemented only after thorough testing is done and if it is found to be working according to the

specifications. The system personnel check the feasibility of the system. The more complex the

system being implemented, the more involved will be the system analysis and design effort required

to implement the three main aspects: education and training, system testing, and changeover. The

implementation stage involves the following tasks:

- Careful planning.

- Investigation of system and constraints.

- Design of methods to achieve the changeover

## IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to

the satisfaction of the intended uses and the operation of the system. In many organizations someone

who will not be operating it, will commission the software development project. In the initial stage

people doubt about the software but we have to ensure that the resistance does not build up, as one

has to make sure that:

➢ The active user must be aware of the benefits of using the new system. Their confidence in the

software is built up.

➢ Proper guidance is imparted to the user so that he is comfortable in using the application. Before

going ahead and viewing the system, the user must know that for viewing the result, the server

program should be running in the server. If the server object is not up running on the server, the

actual process will not take place

### User Training

### 

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

### Training on the Application Software

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

### System Maintenance

System maintenance is an effective component such that System maintenance refers to the ongoing activities required to ensure that a system or application operates effectively and efficiently after it has been implemented. It involves regular updates, bug fixes, and performance optimizations to keep the system running smoothly and securely. System maintenance is essential to ensure that a system remains operational and effective after implementation. By establishing maintenance procedures and following them consistently, project teams can ensure that the system operates smoothly, remains secure, and continues to meet the needs of the end-users

* + 1. **Hosting**

Hosting refers to the process of providing a location or platform where a website, application,

or service can be stored, accessed, and made available to users on the internet. It involves storing the files and data associated with the website or application on a server that is connected to the internet, allowing users to access the content remotely.

There are various types of hosting services available, including shared hosting, virtual private servers (VPS), dedicated hosting, cloud hosting, and more. Each type of hosting offers different levels of resources, scalability, security, and management options, catering to different needs and requirements. Hosting services typically include features such as storage space, bandwidth allocation, server management tools, security measures, and technical support. Choosing the right hosting provider and plan is crucial to ensure that your website or application performs well, remains

secure, and meets the needs of your users.

**000Webhost**

000Webhost is a popular web hosting provider that offers free hosting services. It provides users with the opportunity to host websites without any cost, making it an attractive option for individuals and small businesses looking to establish an online presence.

**Key features of 000Webhost include:**

➢ Free Hosting: 000Webhost offers free web hosting services with no hidden fees or charges.

➢ Website Builder: It provides a user-friendly website builder tool that allows users to create and

customize websites without the need for coding knowledge.

➢ WordPress Integration: Users can easily install and manage WordPress websites on 000Webhost

with one-click installation.

➢ PHP and MySQL Support: The hosting platform supports PHP programming language and

MySQL databases, enabling users to create dynamic and interactive websites.

➢ No Ads: Unlike some free hosting providers, 000Webhost does not display ads on hosted

websites.

➢ Limited Resources: Free hosting plans typically come with limitations on storage space,

bandwidth, and other resources compared to paid hosting plans.

➢ Technical Support: While 000Webhost offers customer support, free hosting users may have

limited access to technical support compared to paid plans.

Overall, 000Webhost is a suitable option for individuals and small businesses looking for a cost

effective way to host a website, especially for those who are just starting and have minimal

requirements. However, users should be aware of the limitations and consider upgrading to a paid

plan as their website grows and requires more resources and support.

**Procedure for hosting a website on 000Webhost:**

**Step 1: Sign Up**

Explanation: Visit the 000Webhost website (www.000webhost.com) and locate the sign-up section.

Fill in the required information, including your email address, password, and chosen website name.

This step creates your account on 000Webhost.

**Step 2: Verify Email**

Explanation: After signing up, you'll receive a verification email from 000Webhost. Check your

inbox and click on the verification link provided in the email to confirm your account. This step

ensures the security of your account and enables you to proceed with setting up your website..

**Step 3: Log In**

Explanation: Once your account is verified, return to the 000Webhost website and log in using the credentials you provided during sign-up. Logging in grants you access to your account dashboard, where you can manage your website and hosting settings.

**Step 4: Choose Website Building Method**

Explanation: Upon logging in, you'll be prompted to choose a website building method.

000Webhost offers options such as WordPress, Website Builder, and Upload Own Website. Select the method that best suits your needs and preferences.

**Step 5: Set Up Domain**

Explanation: If you have a domain name, you can connect it to your 000Webhost account. Navigate to the domain settings and follow the instructions to set up your domain with 000Webhost. Alternatively, you can choose a free subdomain provided by 000Webhost.

**Step 6: Upload or Create Website**

Explanation: Depending on the method you chose in Step 4, you can now upload your existing

website files or create a new website using the built-in tools. Follow the on-screen instructions to complete this step.

**Step 7: Customize Website (if applicable)**

Explanation: If you're using a website builder or content management system like WordPress, you can customize your website's design, layout, and content. Explore the customization options available and make adjustments according to your preferences.

**Step 8: Configure Settings**

Explanation: Access the settings section of your 000Webhost account to configure various aspects of your website, such as security settings, email accounts, database management, and more. Adjust these settings according to your requirements.

**Step 9: Test Website**

Explanation: Before making your website live, it's important to test it thoroughly to ensure

everything is functioning as expected. Check for broken links, formatting issues, and functionality across different devices and web browsers.

**Step 10: Publish Website**

Explanation: Once you're satisfied with the testing results, you can publish your website to make it accessible to visitors. Use the publishing or deploy feature provided by 000Webhost to make your website live on the internet.

**Step 11: Monitor Performance**

Explanation: After launching your website, regularly monitor its performance using tools provided by 000Webhost or third-party analytics platforms. Keep an eye on website traffic, loading speed, and other key metrics to ensure optimal performance.

**Hosted Website:**

**Hosted Link: https://abc.000webhostapp.com**

**Hosted Link QR Code**

**Screenshot**

# CHAPTER 7

# CONCLUSION AND FUTURE SCOPE

## CONCLUSION

## 

.

* 1. **FUTURE SCOPE**

.

# CHAPTER 8

# BIBLIOGRAPHY

### REFERENCES:

* + - ..#Books
    - ..
    - ..
    - ..
    - ...

### WEBSITES:

* + - [..](http://www.w3schools.com/)
    - [..](http://www.jquery.com/)
    - [..](http://homepages.dcc.ufmg.br/%7Erodolfo/es-1-03/IEEE-Std-830-1998.pdf)
    - [..](http://www.agilemodeling.com/artifacts/useCaseDiagram.html)

# CHAPTER 9

# APPENDIX

## Sample Code

Main functionalities

## Screen Shots