

# **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**

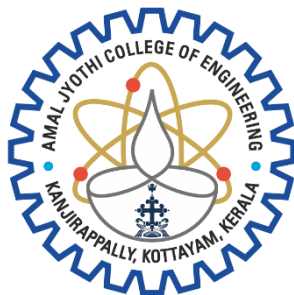


**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS**  
**KANJIRAPPALLY**

[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**

**Internal Guide**

**MEERA ROSE MATHEW**

**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:**  
**KANJIRAPPALLY**

**ROHITH R NAIR**  
**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	

<b>4.2.8</b>	<b>DEPLOYMENT DIAGRAM</b>	
<b>4.2.9</b>	<b>COLLABORATION DIAGRAM</b>	
<b>4.3</b>	<b>USER INTERFACE DESIGN USING FIGMA</b>	
<b>4.4</b>	<b>DATABASE DESIGN</b>	
<b>5</b>	<b>SYSTEM TESTING</b>	
<b>5.1</b>	<b>INTRODUCTION</b>	
<b>5.2</b>	<b>TEST PLAN</b>	
<b>5.2.1</b>	<b>UNIT TESTING</b>	
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	
<b>5.2.4</b>	<b>USER ACCEPTANCE TESTING</b>	
<b>5.2.5</b>	<b>AUTOMATION TESTING</b>	
<b>5.2.6</b>	<b>SELENIUM TESTING</b>	
<b>6</b>	<b>IMPLEMENTATION</b>	
<b>6.1</b>	<b>INTRODUCTION</b>	
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	
<b>6.2.1</b>	<b>USER TRAINING</b>	
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	
<b>7.1</b>	<b>CONCLUSION</b>	
<b>7.2</b>	<b>FUTURE SCOPE</b>	
<b>8</b>	<b>BIBLIOGRAPHY</b>	
<b>9</b>	<b>APPENDIX</b>	
<b>9.1</b>	<b>SAMPLE CODE</b>	
<b>9.2</b>	<b>SCREEN SHOTS</b>	

## List of Abbreviations



# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 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.

## 1.2 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**

#### **2.1 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.

## **2.2 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.

## **2.3 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.

## **2.4 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.

## 2.5 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**

### 3.1 FEASIBILITY STUDY

Farmfolio is a web platform designed to connect farm owners directly with consumers, addressing the gap in accessibility to fresh, locally-produced goods. By creating a streamlined, transparent, and user-friendly interface, the platform eliminates intermediaries, simplifies communication, and promotes sustainable agriculture practices.

#### 3.1.1 Economical Feasibility

The implementation of the **Farmfolio** platform offers substantial cost reduction and operational efficiency in the agricultural marketplace. By enabling direct communication between farm owners and consumers, it eliminates the need for intermediaries, thereby reducing commission costs and increasing profit margins for farmers. The platform streamlines core processes such as farm listing, product updates, event promotion, and delivery tracking, minimizing manual effort and lowering administrative overhead. With features like farm ratings and real-time product availability, it ensures optimized demand forecasting and reduces wastage due to overproduction. Additionally, the use of standard hardware and open-source technologies like PHP, MySQL, HTML, and JavaScript significantly reduces development, deployment, and maintenance costs. This makes Farmfolio an affordable, scalable solution for promoting local produce and empowering rural communities, while maintaining high operational effectiveness.

#### 3.1.2 Technical Feasibility

The development of the **Farmfolio** platform is grounded in strategic decisions regarding its technology stack, development languages, and scalability. Utilizing widely adopted technologies such as HTML, CSS, JavaScript, PHP, and MySQL ensures a flexible, robust, and developer-friendly environment that supports rapid development and maintenance. The use of Apache as the web server enhances platform reliability and cross-platform compatibility. By leveraging open-source tools, Farmfolio significantly reduces licensing costs and improves accessibility for future enhancements. The system is designed with scalability in mind, allowing it to seamlessly adapt to increasing numbers of users, farms, and features without compromising performance. PHP and MySQL provide strong support for dynamic data handling and secure transactions, making the platform resilient, maintainable, and capable of supporting long-term growth and innovation in the agricultural ecosystem.

#### 3.1.3 Behavioral Feasibility

The **Farmfolio** platform emphasizes user acceptance, adaptability, and engagement to ensure smooth integration into daily agricultural and consumer activities. Its intuitive interface is designed to minimize the learning curve for all user roles, including farm owners, consumers, delivery personnel, and administrators. Clear navigation and well-organized features such as farm listings, product availability, and event promotions enhance the overall user experience. Real-time access to fresh produce and direct communication with farm owners encourage frequent use and build consumer trust. The platform's flexibility allows farm owners to easily update products and schedules,

### 3.1.4 Feasibility Study Questionnaire

#### 1. Project Overview?

Farmfolio is a web platform designed to connect farm owners directly with consumers, addressing the gap in accessibility to fresh, locally-produced goods. By creating a streamlined, transparent, and user-friendly interface, the platform eliminates intermediaries, simplifies communication, and promotes sustainable agriculture practices.

#### 2. To what extent the system is proposed for?

Farmfolio is proposed as a full-scale implementation aimed at providing a comprehensive platform for connecting farm owners, consumers, delivery personnel, and administrators. The system goes beyond a prototype or research model, focusing on creating a fully functional and sustainable solution that can be deployed for real-world use.

#### 3. List the Modules included in your System?

- User Management Module
- Farm Management Module
- Product Browsing Module
- Delivery Management Module
- Admin Dashboard Module

#### 4. Identify the users in your project?

**Admin/Platform Management :** Manages user accounts, reviews content, monitors activity, resolves issues, and maintains the system.

**Farm Owners :** Create/manage farms, list products, promote events, handle orders, view/and share contact info.

**Consumers :** Browse farms/products, contact owners, place orders, and leave ratings/reviews.

**Delivery Boy :** Receive delivery requests, update order status, and manage delivery availability.

#### 5. Who owns the system?

The system is owned and managed by the platform administrators responsible for overseeing Farmfolio's operations, ensuring smooth functionality, and maintaining user and content management.



6. System is related to which firm/industry/organization?

The agriculture and agritech industry, focusing on farm-to-consumer connectivity and local food distribution.

7. Details of person that you have contacted for data collection?

Mr Sivadasan Nair (Cow Farm Owner)

8. Questionnaire to collect details about the project? (min 10 questions, include descriptive answers, attach additional docs (e.g. Bill receipts, certificate models), if any?)

1. How frequently do you engage in farm-related transactions or deliveries?

Daily

2. Do you currently use any digital tools or platforms for these activities?

No

3. What is the primary purpose of your cow farm?

Milk production and selling dairy products like cheese, yogurt, and butter.

4. How do you currently ensure the quality and freshness of your dairy products for customers?

By using cold storage facilities, maintaining strict hygiene during production, and delivering products within a few hours of processing.

5. What challenges do you face in selling your products directly to customers?

Reaching a larger customer base and ensuring timely delivery of fresh products.

6. How do you currently market your farm and products?

Through local word-of-mouth, community markets.

7. Are you interested in hosting farm events, such as farm visits or workshops?

Yes, I would like to host farm tours and workshops to educate people about dairy farming and animal care.

8. How important is it for you to receive customer feedback through ratings and reviews?

Very important. Feedback helps improve product quality and build trust with customers.

9. What delivery methods do you currently use for your products? What delivery methods do you currently use for your products?

Local delivery through our own vehicles and customers picking up products directly from the farm.

10. What features would you find most helpful in a platform like Farmfolio?

Direct customer communication, a simple product catalog system, and an option for delivery scheduling.

## **3.2 SYSTEM SPECIFICATION**

### **3.2.1 Hardware Specification**

**Processor** - **Dual core processor of higher**  
**RAM** - 4GB or higher  
**Hard disk** - Minimum 100GB HDD/SDD

### **3.2.2 Software Specification**

**Front End** - HTML, CSS, JAVASCRIPT, AJAX  
**Back End** - PHP  
**Database** - MS SQL  
**Client on PC** - Windows 7 or above  
**Technologies used** - jS, HTML5, BOOTSTRAP, PHP, CSS

## **3.3 SOFTWARE DESCRIPTION**

### **3.3.1 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.

### **3.3.2 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**

## 4.1 INTRODUCTION

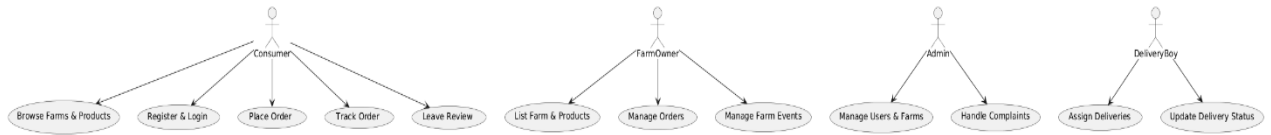
Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system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

## 4.2 UML DIAGRAM

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

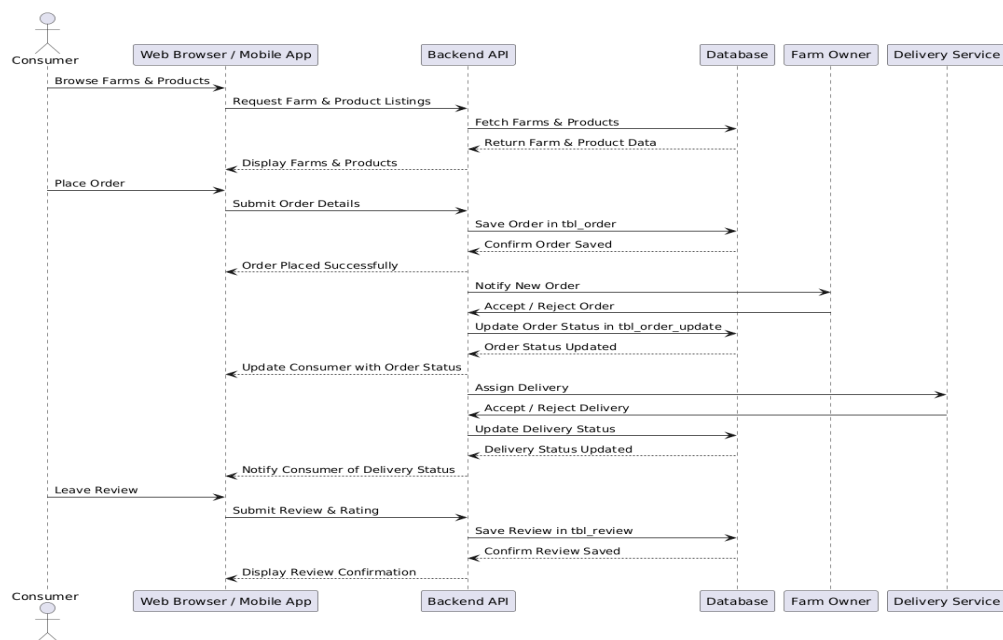
## 4.2.1 USE CASE DIAGRAM

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

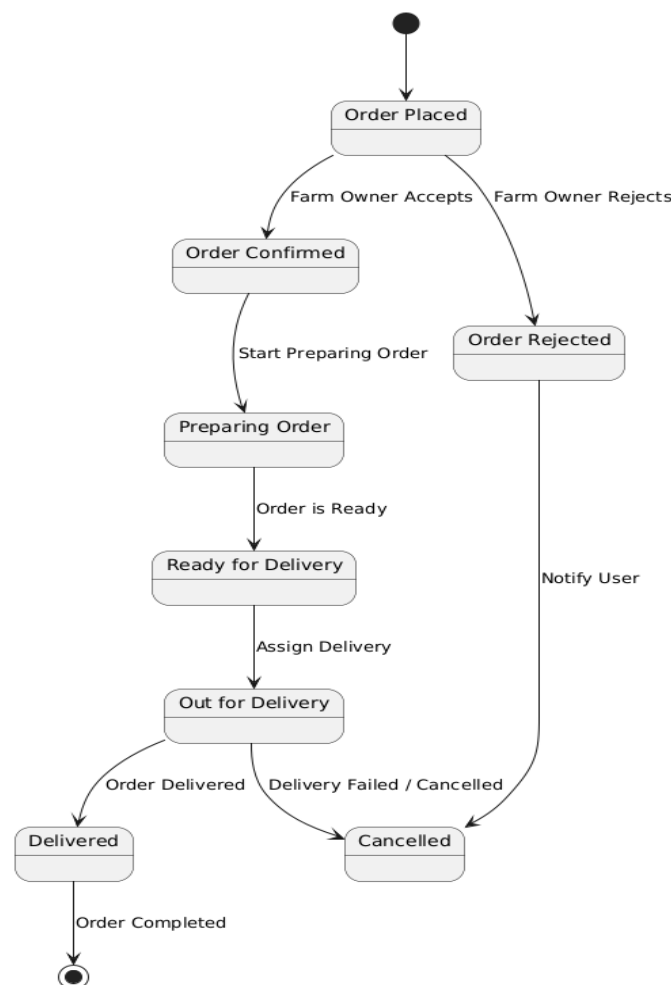
Sequence diagrams visualize the interactions between objects or components in a system over time. They show the order of messages exchanged between objects, including the timing and dependencies of these messages. Sequence diagrams are particularly useful for understanding the dynamic behavior of a system,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



### 4.2.3 State Chart Diagram

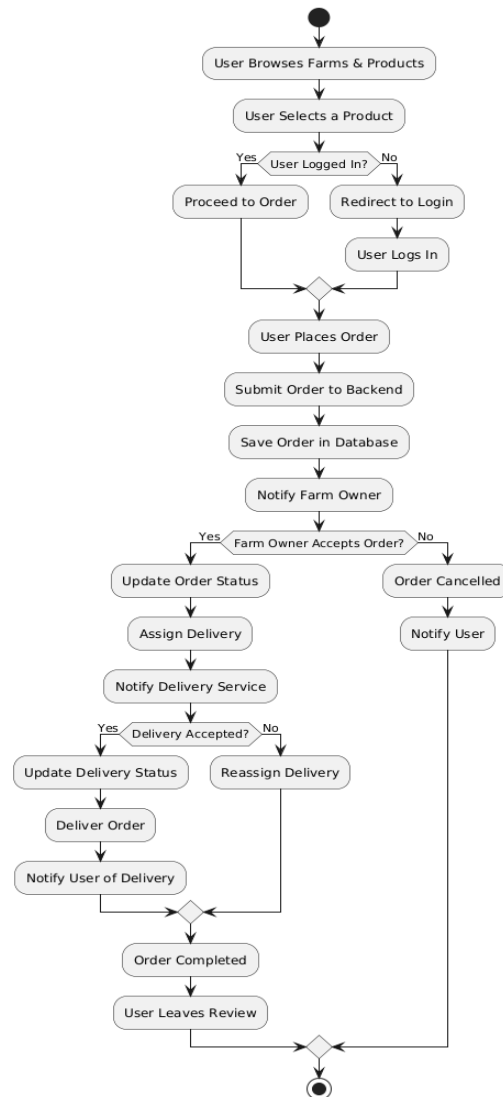
State chart diagrams serve as a vital tool in the arsenal of software engineers and systems analysts, offering a structured way to conceptualize and model the dynamic behavior of objects or entities within a system. These diagrams effectively break down complex systems into manageable components, illustrating 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.



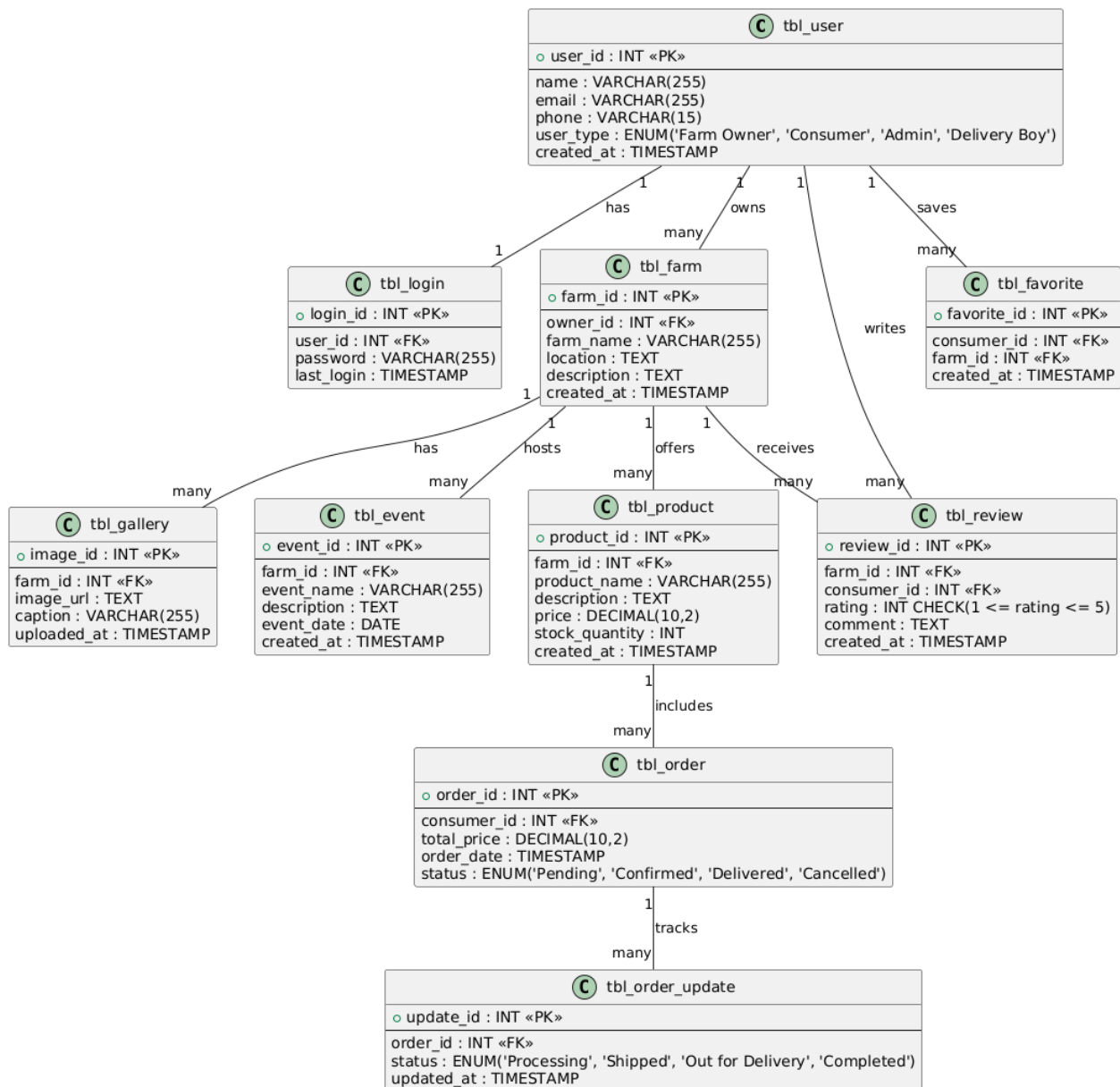
#### 4.2.4 ACTIVITY DIAGRAM

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

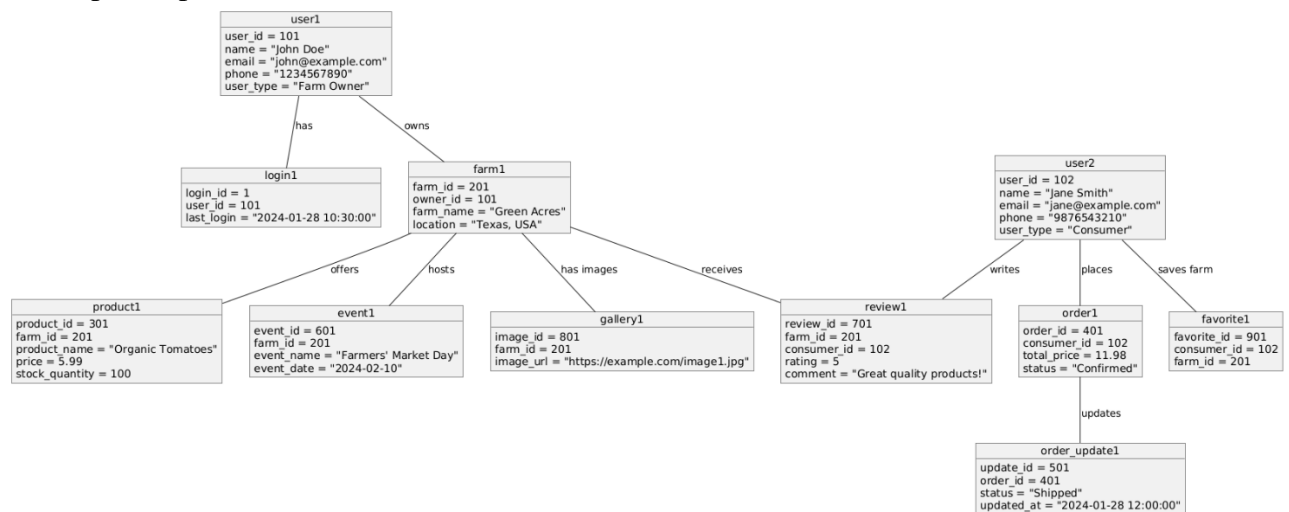
Class diagrams provide a static view of a system's structure by depicting the classes, attributes, methods, and relationships between classes. They show the blueprint of the system's objects and how they interact with each other. Class diagrams are fundamental for object-oriented analysis and design,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





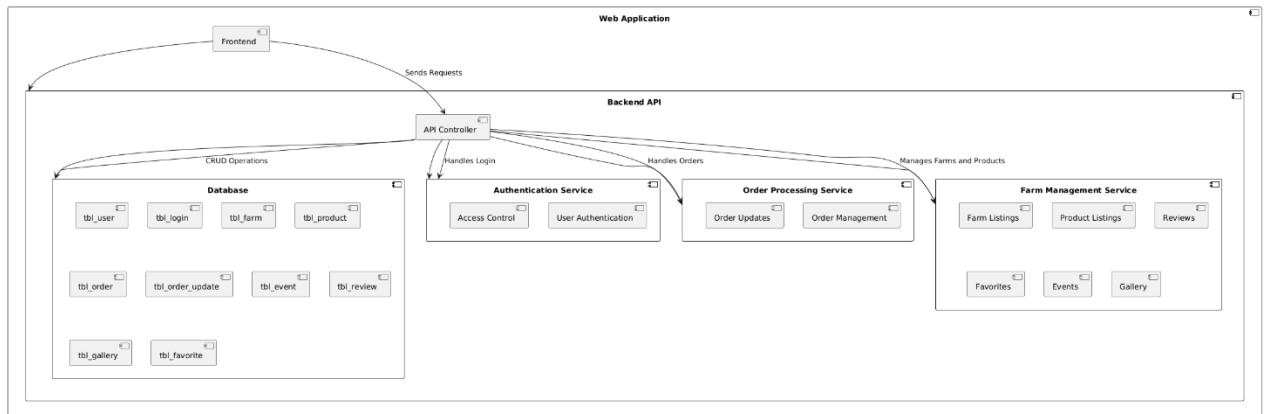
#### 4.2.6 OBJECT DIAGRAM

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

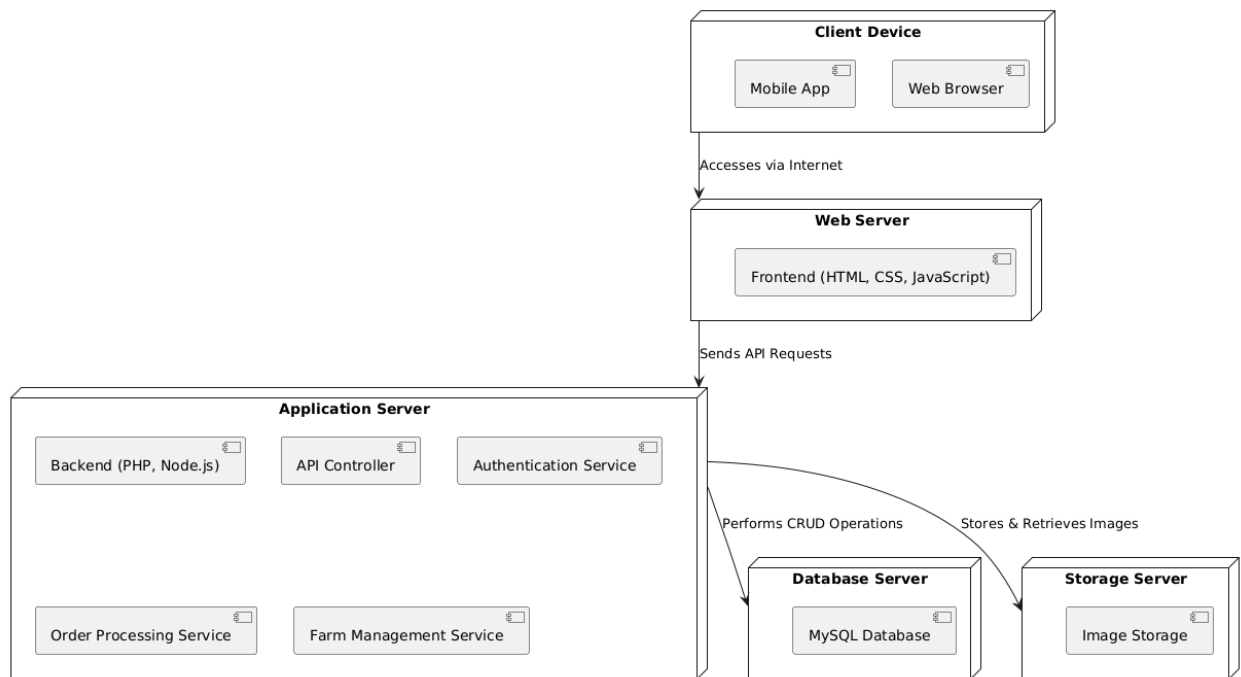
Component diagrams provide a visual representation of the relationships between various components, including their interfaces, dependencies, and interactions. By highlighting these connections, stakeholders can gain insights into the flow of data, control, and communication within the system, aiding 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

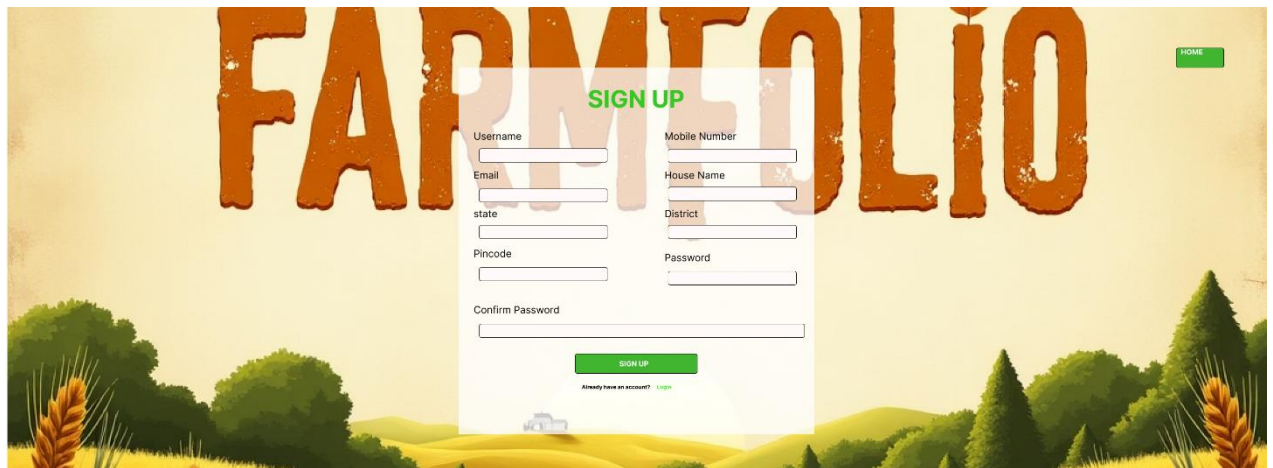
Deployment diagrams depict the physical deployment of software components on hardware nodes, such as servers, computers, or devices. They show how software artifacts are distributed across the hardware infrastructure and how they communicate with each other. Deployment diagrams are essential for understanding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

Explanation, Diagram



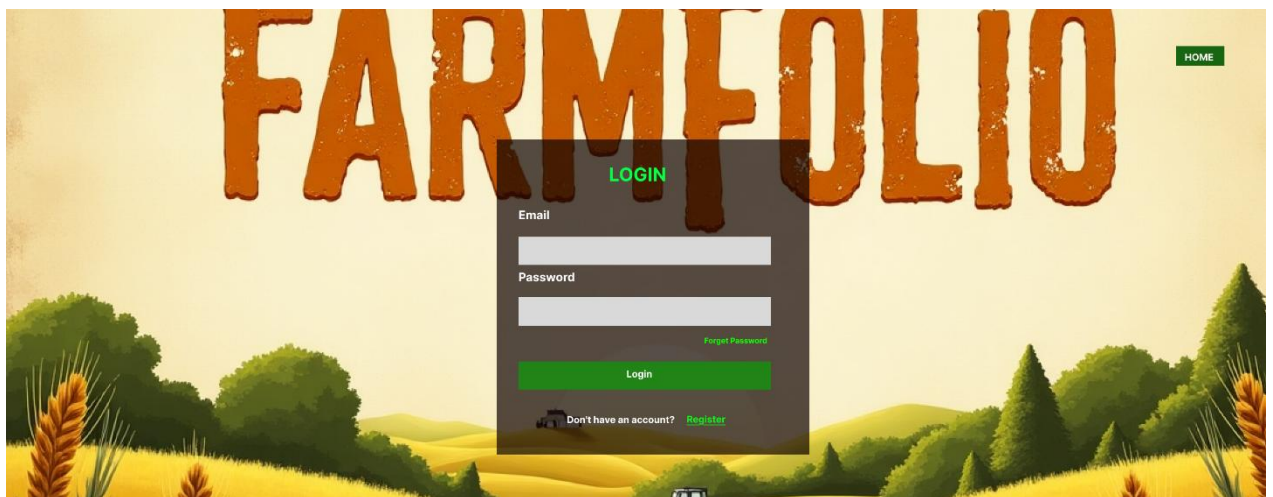
## 4.3 USER INTERFACE DESIGN USING FIGMA

### Form Name: SIGNUP



The Signup form is centered on a background featuring the 'FARMFOLIO' logo and a rural landscape. It includes input fields for Username, Mobile Number, Email, House Name, state, District, Pincode, Password, and Confirm Password. A green 'SIGN UP' button is at the bottom, with a link to 'Already have an account? Login'.

### Form Name: LOGIN



The Login form is centered on the same background as the Signup form. It includes input fields for Email and Password, a 'Forgot Password' link, and a green 'Login' button. At the bottom, there is a link to 'Don't have an account? Register'.

### Form Name: User Dashboard



The User Dashboard features a green sidebar with navigation links: Browse Farms, My Orders, Favorite Farms, Farm Events, and Profile. The main content area displays the FarmFolio logo, a 'Welcome consumer' message, and a 'Logout' button. Below these are three placeholder boxes for user information. The footer contains copyright information and links to Terms of Service, Privacy Policy, Contact, and UsFAQ.

**Form Name: Farm Dashboard**

Dashboard

Products

Farm Images

Events

Reviews

Orders

About

FarmFolio

Welcome Farm

Logout

Active Peoducts

6

Events Lisated

1

Reviews

4.8

© 2025 Farmfolio . All rights reserved.

[Terms of Service](#) [Privacy Policy](#) [Contact](#) [UsFAQ](#)

**Form Name: Delivery Dashboard**

Dashboard

Assigned Deliveries

Delivery History

Earnings

Profile

FarmFolio

Welcome Delivery Boy

Logout

Pending Deliveries

3

Delivery History

5

Total Earnings

500.00

**Form Name: Admin Dashboard**

Home

Users

Farms

Products

Deliveries

Reviews

Analytics

Settings

FarmFolio

Welcome Admin

Logout

Total Users

24

Total Farms

12

Activities

© 2025 Farmfolio . All rights reserved.

[Terms of Service](#) [Privacy Policy](#) [Contact](#) [UsFAQ](#)

## 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.

ORDER_ID	CUSTOMER_NAME	PRODUCT_NAMES	QUANTITY
101	John Doe	Apples, Oranges	5, 3
102	Jane Smith	Bananas	2
103	Sam Brown	Grapes, Apples	4, 6

Table1



Conversion to first normal form

ORDER_ID	CUSTOMER_NAME	PRODUCT_NAME	QUANTITY
101	John Doe	Apples	5
101	John Doe	Oranges	3
102	Jane Smith	Bananas	2
103	Sam Brown	Grapes	4
103	Sam Brown	Apples	6

## 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.

ORDER_ID	PRODUCT_ID	CUSTOMER_NAME	PRODUCT_NAME	QUANTITY
101	P1	John Doe	Apples	5
101	P2	John Doe	Oranges	3
102	P3	Jane Smith	Bananas	2
103	P1	Sam Brown	Apples	6
103	P3	Sam Brown	Bananas	4

Table1

Conversion to second normal form

ORDER_ID	CUSTOMER_NAME
101	John Doe
102	Jane Smith
103	Sam Brown

ORDER_ID	PRODUCT_ID	PRODUCT_NAME	QUANTITY
101	P1	Apples	5
101	P2	Oranges	3
102	P3	Bananas	2
103	P1	Apples	6
103	P3	Bananas	4



### Third Normal Form

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

ORDER_ID	CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_CITY
101	C1	John Doe	New York
102	C2	Jane Smith	Los Angeles
103	C3	Sam Brown	Chicago
104	C1	John Doe	New York

Table 1

Conversion to Third normal form

ORDER_ID	CUSTOMER_ID
101	C1
102	C2
103	C3
104	C1

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_CITY
C1	John Doe	New York
C2	Jane Smith	Los Angeles
C3	Sam Brown	Chicago

### **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	<b>participant_id</b>	<b>int(11)</b>	<b>Primary Key, Auto Increment</b>	<b>Unique ID for participant entry</b>
2	<b>event_id</b>	<b>int(11)</b>	<b>Foreign Key (Events)</b>	<b>ID of the event the user registered for</b>
3	<b>user_id</b>	<b>int(11)</b>	<b>Foreign Key (Users)</b>	<b>ID of the user participating in the event</b>
4	<b>registration_date</b>	<b>timestamp</b>	<b>Not Null, Default: current_timestamp()</b>	<b>Date of registration</b>
5	<b>status</b>	<b>enum('Pending', 'Confirmed', 'Cancelled')</b>	<b>Default: 'Pending'</b>	<b>Status of the registration</b>

**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	<b>review_id</b>	<b>int(11)</b>	<b>Primary Key, Auto Increment</b>	<b>Unique ID for a review</b>
2	<b>farm_id</b>	<b>int(11)</b>	<b>Foreign Key (Farms)</b>	<b>ID of the farm being reviewed</b>
3	<b>user_id</b>	<b>int(11)</b>	<b>Foreign Key (Users)</b>	<b>ID of the user who left the review</b>
4	<b>rating</b>	<b>int(11)</b>	<b>Not Null</b>	<b>Rating given to the farm</b>
5	<b>comment</b>	<b>text</b>	<b>Nullable</b>	<b>User's review comment</b>
6	<b>created_at</b>	<b>timestamp</b>	<b>Not Null, Default: current_timestamp()</b>	<b>Timestamp of review creation</b>

**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	<b>order_id</b>	<b>INT(11)</b>	<b>PRIMARY KEY, AUTO_INCREMENT</b>	<b>Unique identifier for each order</b>
2	<b>user_id</b>	<b>INT(11)</b>	<b>FOREIGN KEY (Users)</b>	<b>Identifies the user who placed the order</b>
3	<b>total_amount</b>	<b>DECIMAL(10,2)</b>	<b>NOT NULL</b>	<b>Total cost of the order</b>
4	<b>order_status</b>	<b>ENUM('pending', 'processing', 'shipped', 'delivered')</b>	<b>DEFAULT 'pending'</b>	<b>Tracks the current status of the order</b>
5	<b>order_date</b>	<b>DATETIME</b>	<b>NOT NULL</b>	<b>The date and time when the order was placed</b>
6	<b>delivery_address</b>	<b>TEXT</b>	<b>NULLABLE</b>	<b>Address where the order should be delivered</b>
7	<b>phone_number</b>	<b>VARCHAR(15)</b>	<b>NULLABLE</b>	<b>Contact number for delivery</b>
8	<b>payment_method</b>	<b>VARCHAR(10)</b>	<b>DEFAULT 'cod'</b>	<b>Payment method used (e.g., COD, online)</b>
9	<b>payment_status</b>	<b>ENUM('pending', 'paid', 'failed')</b>	<b>DEFAULT 'pending'</b>	<b>Payment status for the order</b>
10	<b>delivery_boy_id</b>	<b>INT(11)</b>	<b>FOREIGN KEY (Delivery_Boys) NULLABLE</b>	<b>Identifies the delivery person handling the order</b>
11	<b>updated_at</b>	<b>TIMESTAMP</b>	<b>DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP</b>	<b>Tracks the last update time of the order</b>
12	<b>processing_date</b>	<b>TIMESTAMP</b>	<b>NULLABLE</b>	<b>The time when the order was moved to "processing" status</b>

<b>13</b>	<b>shipped_date</b>	<b>TIMESTAMP</b>	<b>NULLABLE</b>	<b>The time when the order was shipped</b>
<b>14</b>	<b>delivered_date</b>	<b>TIMESTAMP</b>	<b>NULLABLE</b>	<b>The time when the order was delivered</b>

### 13 . tbl\_order\_items

Primary key: **item\_id**

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

<b>No</b>	<b>Field Name</b>	<b>Datatype (Size)</b>	<b>Key Constraints</b>	<b>Description of the Field</b>
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**

## 5.1 INTRODUCTION

Software Testing is a crucial process of evaluating software in a controlled manner to determine if it functions as specified. System testing aims to assess the overall system specifications, encompassing various components of a computer-based system beyond just the software itself. The software inevitably interacts with other hardware and software systems. Therefore, System Testing involves conducting a range of tests aimed at fully exercising the entire computer-based system.

### Two Category of Software Testing

- Black Box Testing
- White Box Testing

Categorized as a form of software testing, System Testing belongs to the black box testing category, which focuses on examining the external behavior and functionality of the software application from the end user's perspective. Conversely, White box testing pertains to evaluating the internal workings or code of the software.

## 5.2 TEST PLAN

It is a thorough document that outlines the resources, test strategy, goals, timetable, estimation, deliverables, and testing process for a software product. A test plan plays a critical role in estimating the extent of effort needed to ensure the quality of an application. To guarantee that software testing procedures adhere to a defined methodology, the test manager closely supervises and regulates every facet of the test plan.

It is always the software developers' job to test each of the program's individual parts to ensure that it serves the intended function. There is an independent test group (ITG) to address the problems that come from letting the developer assess what they have created.

### 5.2.1 Unit Testing

Unit testing is the initial stage of functional testing that involves testing individual components or units of a software application. The goal of this testing is to verify the performance of each unit and ensure its accuracy. Developers typically employ the white box testing approach to perform unit testing, with the aim of validating the correctness of isolated code.

Upon completion and submission of the application, the test engineer will commence an independent and sequential evaluation of each component or module. This evaluation process is known as unit testing.

### **5.2.2 Integration Testing**

Once unit testing is completed, the software testing process advances to the integration testing phase, which involves testing multiple units or software components together. In integration testing, the primary goal is to identify potential flaws that may surface when integrated units or components interact with each other.

Unit testing makes use of modules, and integration testing assembles and tests these modules. To make sure that all of the modules are properly communicating, integrity testing is carried out.

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

Requirements validation is a static practice used to review and verify the specific requirements for a particular development stage, whereas product testing is a dynamic practice performed after the development phase to confirm that the final product satisfies customer requirements. Unlike testing, requirements validation does not involve running any code. Its aim is to ensure that the product meets the client's actual needs and to demonstrate that the product performs as intended when deployed in the appropriate environment.

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

During the user acceptance testing phase, the system is evaluated to ensure it meets the needs of the organization. The software must remain relevant to the user and be adaptable to changes as needed. The testing is conducted with respect to the input and output screen designs, using various types of test data. The success of the system testing depends heavily on the preparation of test data. Once the test data is ready, the system is tested using the data, and any errors uncovered are corrected using the testing steps mentioned above. Any necessary corrections are noted for future use.

### **5.2.5 Automation Testing**

Automated testing plays a crucial role in assessing software and other technological products to guarantee compliance with rigorous standards. The aim of this testing is to validate that the equipment or software operates exactly as intended, and to detect bugs, defects, and other issues that may arise during product development. While some types of testing can be performed manually, such as regression or functional testing, automation testing provides numerous advantages.



## 5.2.6 Selenium Testing

Selenium is a highly popular open-source suite for automating Web UI (User Interface) testing that was first created by Jason Huggins as an internal tool at Thought Works in 2004. Selenium enables automation testing on diverse browsers, platforms, and programming languages.

Selenium is a valuable and open-source automated testing tool that is important for web application developers to know about. When performing a test using Selenium, it is commonly called Selenium automation testing. However, Selenium is not a single tool, but a suite of tools that are designed to meet various automation testing needs for Selenium.

### Test Case 1

```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestLogin():
    def setup_method(self, method):
        self.driver = webdriver.Chrome()
        self.vars = {}

    def teardown_method(self, method):
        self.driver.quit()

    def test_login(self):
        self.driver.get("http://localhost/mini%20project/login/login.php")
        self.driver.set_window_size(1060, 804)
        self.driver.find_element(By.ID, "email").click()
        self.driver.find_element(By.ID, "email").click()
        self.driver.find_element(By.ID, "email").send_keys("Farmfoliomini@gmail.com")
        self.driver.find_element(By.ID, "password").click()
        self.driver.find_element(By.ID, "password").send_keys("Admin@2004")
        self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
        self.driver.find_element(By.CSS_SELECTOR, ".logout-btn").click()
        self.driver.find_element(By.CSS_SELECTOR, ".container").click()
        self.driver.find_element(By.ID, "email").click()
        self.driver.find_element(By.ID, "email").send_keys("rohithreghu482@gmail.com")
        self.driver.find_element(By.ID, "password").click()
        self.driver.find_element(By.ID, "password").send_keys("Abin@2004")
        self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
        self.driver.find_element(By.CSS_SELECTOR, ".popup-logout-btn").click()
```

```
PS E:\xampp\htdocs\mini project\test> python -m pytest test_login.py
===== test session starts =====
platform win32 -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: E:\xampp\htdocs\mini project\test
collected 1 item

test_login.py
DevTools listening on ws://127.0.0.1:57060/devtools/browser/10164836-eb80-4c6c-a7e0-e1b3acb22524
[17672:8096:0408/192152.859:ERROR:interface_endpoint_client.cc(725)] Message 0 rejected by interface blink.mojom.WidgetHost

===== 1 passed in 6.55s =====
```

## Test report

Test Case 1					
Project Name:FarmFolio					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Rohith R Nair		
TestPriority(Low/Medium/High) : Medium			Test Designed Date: 06/04/2025		
Module Name: Login Module			Test Executed By :		
Test Title : login test			Test Execution Date:		
Description: Testing the login modu					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Navigate to login page		User should be able to login	Login page displayed	Pass
2	Provide valid email	Email – farmfoliomini@gmail.com	Admin should be logged in	Admin logged in and redirected to dashboard	Pass
3	Provide valid password	Password – Admin@2004			
4	Click login button	-			
5	Provide valid email	Email – rnairrohith17@gmail.com	Farmer should be logged in	Farmer logged in and redirected to dashboard	Pass
6	Provide valid password	Password – Abin@2004			
7	Click login button	-			
Post-Condition: Users navigated to there dashboards					

## Test Case 2

### Code

```
# Generated by Selenium IDE
import pytest
import
import
from se
from se
from se
from se
from se
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestEvents():
    def setup_method(self, method):
        self.driver = webdriver.Chrome()
        self.vars = {}

    def teardown_method(self, method):
        self.driver.quit()

    def test_events(self):
        self.driver.get("http://localhost/mini%20project/login/login.php")
        self.driver.set_window_size(1055, 802)
        self.driver.find_element(By.ID, "email").click()
        self.driver.find_element(By.ID, "email").send_keys("rnairrohith17@gmail.com")
        self.driver.find_element(By.ID, "password").click()
        self.driver.find_element(By.ID, "password").send_keys("Rohith@2004")
        self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
        self.driver.find_element(By.CSS_SELECTOR, "li:nth-child(4) span").click()
        self.driver.find_element(By.CSS_SELECTOR, ".add-event-btn").click()
        self.driver.find_element(By.ID, "event_name").click()
        self.driver.find_element(By.ID, "event_name").send_keys("training")
        self.driver.find_element(By.ID, "event_date").click()
        self.driver.find_element(By.ID, "event_date").send_keys("0002-04-12")
        self.driver.find_element(By.ID, "event_date").send_keys("0020-04-12")
        self.driver.find_element(By.ID, "event_date").send_keys("0202-04-12")
        self.driver.find_element(By.ID, "event_date").send_keys("2025-04-12")
        self.driver.find_element(By.ID, "event_description").click()
        self.driver.find_element(By.ID, "event_description").send_keys("training")
        self.driver.find_element(By.ID, "submitBtn").click()
        self.driver.find_element(By.CSS_SELECTOR, ".logout-btn").click()
```

```
PS E:\xampp\htdocs\mini project\event> python -m pytest test_events.py
===== test session starts =====
platform win32 -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: E:\xampp\htdocs\mini project\event
collected 1 item

test_events.py
DevTools listening on ws://127.0.0.1:55454/devtools/browser/689ebf35-9a09-4eda-a723-16bbe451d58b
[100%]

===== 1 passed in 11.96s =====
```

## Test report

<b>Test Case 2</b>					
<b>Project Name: FarmFolio</b>					
<b>Add Event Test Case</b>					
<b>Test Case ID: Test_2</b>			<b>Test Designed By: Rohith R Nair</b>		
<b>Test Priority (Low/Medium/High) : Medium</b>			<b>Test Designed Date: 06/04/2025</b>		
<b>Module Name: Event Module</b>			<b>Test Executed By :</b>		
<b>Test Title : Event test</b>			<b>Test Execution Date:</b>		
<b>Description: Testing the event Adding module</b>					
<b>Pre-Condition : User has valid username and password</b>					
<b>Step</b>	<b>Test Step</b>	<b>Test Data</b>	<b>Expected Result</b>	<b>Actual Result</b>	<b>Status (Pass/Fail)</b>
1	Navigate to login page	URL: http://localhost/mini-project/login/login.php	Login page should be displayed	Login page displayed	Pass
2	Provide valid email	Email – rnairrohith17@gmail.com	User should be logged in	User logged in and redirected	Pass
3	Provide valid password	Password – Rohith@2004			
4	Click login button	-			
5	Navigate to event section	Click on event tab	Event should be saved and listed	Event Added successfully	Pass
6	Click Add Event button and add event Details	-			
7	Submit the event	Click Submit			
<b>Post-Condition: Event is added and visible in the event list</b>					

## Test Case 3

### Code

```

y > test_category.py > ...
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestCategory():
    def setup_method(self, method):
        self.driver = webdriver.Chrome()
        self.vars = {}

    def teardown_method(self, method):
        self.driver.quit()

    def test_category(self):
        self.driver.get("http://localhost/mini%20project/login/login.php")
        self.driver.set_window_size(1058, 804)
        self.driver.find_element(By.ID, "email").click()
        self.driver.find_element(By.ID, "email").send_keys("farmfoliomin@gmail.com")
        self.driver.find_element(By.ID, "password").click()
        self.driver.find_element(By.ID, "password").send_keys("Admin@2004")
        self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
        self.driver.find_element(By.ID, "email").click()
        self.driver.find_element(By.ID, "email").send_keys("farmfoliomini@gmail.com")
        self.driver.find_element(By.ID, "password").click()
        self.driver.find_element(By.ID, "password").send_keys("Admin@2004")
        self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
        self.driver.find_element(By.ID, "category").click()
        self.driver.find_element(By.ID, "category").click()
        self.driver.find_element(By.ID, "category").send_keys("fruits")
        self.driver.find_element(By.ID, "subcategory").click()
        self.driver.find_element(By.ID, "subcategory").send_keys("mango")
        self.driver.find_element(By.NAME, "add_category").click()
        self.driver.find_element(By.CSS_SELECTOR, ".logout-btn").click()
        self.driver.close()

```

### Screenshot

```

PS E:\xampp\htdocs\mini project\category> python -m pytest test_category.py
===== test session starts =====
platform win32 -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: E:\xampp\htdocs\mini project\category
collected 1 item

test_category.py
DevTools listening on ws://127.0.0.1:51818/devtools/browser/4559e8ca-a17c-43a6-b221-7862fe58f7d7
[22812:4440:0408/102633.472:ERROR:interface_endpoint_client.cc(725)] Message 0 rejected by interface blink.mojom.WidgetHost
[100%]

===== 1 passed in 10.52s =====
PS E:\xampp\htdocs\mini project\category>

```

## Test report

### Test Case 4

```

search > test_search.py > testSearch > teardown_method
1  # Generated by Selenium IDE
2  import pytest
3  import time
4  import json
5  from selenium import webdriver
6  from selenium.webdriver.common.by import By
7  from selenium.webdriver.common.action_chains import ActionChains
8  from selenium.webdriver.support import expected_conditions
9  from selenium.webdriver.support.wait import WebDriverWait
10 from selenium.webdriver.common.keys import Keys
11 from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
12
13 class TestSearch():
14     def setup_method(self, method):
15         self.driver = webdriver.Chrome()
16         self.vars = {}
17
18     def teardown_method(self, method):
19         self.driver.quit()
20
21     def test_search(self):
22         self.driver.get("http://localhost/mini%20project/login/login.php")
23         self.driver.set_window_size(1060, 804)
24         self.driver.find_element(By.ID, "email").click()
25         self.driver.find_element(By.ID, "email").send_keys("rohithreghu842@gmail.com")
26         self.driver.find_element(By.ID, "password").click()
27         self.driver.find_element(By.ID, "password").send_keys("Hari@2004")
28         self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
29         self.driver.find_element(By.CSS_SELECTOR, ".logout-btn").click()
30         self.driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2)").click()
31         self.driver.find_element(By.ID, "email").click()
32         self.driver.find_element(By.ID, "email").send_keys("rohithreghu482@gmail.com")
33         self.driver.find_element(By.ID, "password").click()
34         self.driver.find_element(By.ID, "password").send_keys("Abin@2004")
35         self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(5)").click()
36         self.driver.find_element(By.LINK_TEXT, "Browse Farms").click()
37         self.driver.find_element(By.ID, "searchInput").click()
38         self.driver.find_element(By.ID, "searchInput").send_keys("eggs")
39         self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(3)").click()
40         self.driver.find_element(By.ID, "searchInput").click()
41         self.driver.find_element(By.ID, "searchInput").send_keys("mango")
42         self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(3)").click()
43         self.driver.find_element(By.ID, "searchInput").click()
44         self.driver.find_element(By.ID, "searchInput").send_keys("selam ")
45         self.driver.find_element(By.CSS_SELECTOR, "button:nth-child(3)").click()
46
47

```

```

PS E:\xampp\htdocs\mini project\search> python -m pytest test_search.py
===== test session starts =====
platform win32 -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: E:\xampp\htdocs\mini project\search
collected 1 item

test_search.py
DevTools listening on ws://127.0.0.1:52767/devtools/browser/f066d55b-a4f4-42cb-9a60-fb0dea92f459 [100%]
===== 1 passed in 9.06s =====

```

## Test Report

# **CHAPTER 6 IMPLEMENTATION**

## 6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be 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



## 6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system. Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application. Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process will not take place

### 6.2.1 User Training

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

### 6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the 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.

### 6.2.3 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

### 6.2.4 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.

**Hosted Website:**

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

**Hosted Link QR Code**

**Screenshot**



## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## **7.1 CONCLUSION**

.

## **7.2 FUTURE SCOPE**

.

## **CHAPTER 8**

## **BIBLIOGRAPHY**

---

**REFERENCES:**

- ..#Books
- ..
- ..
- ..
- ...

**WEBSITES:**

- ..
- ..
- ..
- ..

## **CHAPTER 9**

### **APPENDIX**



## **9.1 Sample Code**

Main functionalities

## **9.2 Screen Shots**

