**Crop Elevate**

**[It is a site that gives crops to the farmers.]**

*Mini Project Report*

*Submitted by*

**Aleena Joseph**

**Reg. No.: AJC19MCA-I007**

*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**

**KANJIRAPPALLY**

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

# 2023-2024

## DEPARTMENT OF COMPUTER APPLICATIONS

### AMAL JYOTHI COLLEGE OF ENGINEERING

**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**CROP ELEVATE”** is the bona fide work of **ALEENA JOSEPH (Regno: AJC19MCA-I007)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

**Mr. G.S Ajith Meera Rose Mathew**

**Internal Guide Coordinator**

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

**Head of the Department**

**DECLARATION**

I hereby declare that the project report **“CROP ELEVATE”** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

**Date: ALEENA JOSEPH**

**KANJIRAPPALLY Reg: AJC19MCA-I007**

# ACKNOWLEDGEMENT

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

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Meera Rose Mathew** for his 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 **G.S Ajith** for her inspiration and helping hand.

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

ALEENA JOSEPH

# ABSTRACT

.

***CropElevate***

It is a management project that mainly aims to develop a platform that facilitates the exchange of crops to the farmers. Here the farmers can get the best crop that provides the best productivity within a limited period of time. The farmers of the different Wards who are interested in farming can register and the crops are given only to the farmers who met certain criteria that are given. The members can send notifications to the Secretary and the secretary can select them based on the priority. The crops are given to the farmers by the Panchayath Secretary only after the proper approval from the Ward Member.

• Farmer Registration: Farmers who are interested can create their user accounts on the platform.

• Income Analysis and Verification: The ward Member can access farmer income details and can analyse and verify the accuracy of the provided information.

• User Dashboard: Each user the Panchayath Secretary, Ward Member, and Farmer has a personalized dashboard displaying relevant information.

• Crop Recommendation System: The platform offers a personalized crop recommendation system, using the details provided by the farmers. The Panchayath Secretary can review it and take proper decisions based on it.

• Crop Database: A comprehensive database is maintained, containing information on various crops, expected profits

• Mobile Compatibility: The platform is designed to be accessible and user friendly across different devices including mobile phones. Mainly help the farmers to get the best crop for their field that give the best productivity in a limited period of time.

***Modules***

**Farmer**

• Registration

• Login

• Manage Profile

• Notifications

• Crop Selection

• Recommendation to member

**Secretary (Admin)**

• Login

• Manage Members

• Notifications

• Manage Meetings

• Reports

• Digital Notice board

**Member**

• Login

• Manage Profile

• Notifications

• Verify Framer

**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 Abbreviation

# CHAPTER 1

# INTRODUCTION

### PROJECT OVERVIEW

The project mainly aims to develop a easy to access platform for the users. Here the farmers can register for crops through online and the panchayath will provide the crops for the farmers who met the criterias. The Ward member plays an important role in the approval of the crops to the farmers. The panchayath secretary (Admin) will provide the final approval. The sales person will collect the products (products grown from the crops) and will sold it in the respective shops.

### PROJECT SPECIFICATION

Developing a user-friendly online platform that enables farmers to register their crops. The system will evaluate the criteria set by the panchayat for crop approval. Ward members will play a pivotal role in this approval process, and the final approval will be granted by the panchayat secretary (Admin). Additionally, the platform will facilitate the collection of products grown from these crops by salespersons and their subsequent sale in designated shops. This system should include user registration and authentication, crop evaluation, approval workflows, and inventory management for the sales process. The platform should be accessible, intuitive, and efficient to serve the needs of both farmers and the administrative team.

# CHAPTER 2

# SYSTEM STUDY

### INTRODUCTION

System study is the process of analyzing, designing, and evaluating a system to understand its functioning, identify potential improvements, and ensure it aligns with the organization's goals and requirements.

System study is a critical phase in the system development life cycle, helping organizations make informed decisions about their systems and ensuring they align with business objectives. It can be applied to various types of systems, including software, business processes, and hardware infrastructure.

### EXISTING SYSTEM

**2.2.1 NATURAL SYSTEM STUDIED**

In the current system the farmers can apply for the crop by filling a hand written form and the farmers need to visit the panchayath for the further updates.it is difficult for the farmers.

The farmers will get the current updates only after the meeting conducted by the panchayath secretary with the ward members.so, it is difficult for the farmers.

**2.2.2 DESIGNED SYSTEM STUDIED**

In the current system the farmers can apply for the crop by filling a hand written form and the farmers need to visit the panchayath for the further updates.it is difficult for the farmers.

### DRAWBACKS OF EXISTING SYSTEM

* Time consuming
* Updates Of details is Difficult
* Getting updates is difficult
* Clearing clarifications is difficult

### PROPOSED SYSTEM

The proposed system mainly hep the farmers to apply for the crop samplings easily in an online mode. Through this system the farmers can easily register for the crop samplings. The copy of the application form is sent to the ward member so that the ward member can verify that the farmer is in the correct ward or not. When the member approve the application the message is sent to the farmer that the ward member has approved the application. The copy of the approved application from the ward member is sent to the admin, here it is the Panchayath Secretary. The panchayath secretary will approve the application and the crop is given to the farmer by the panchayath secretary. the up-to-date updates are displayed in the farmer page.so the system will help the farmers to get the updates easily.

### ADVANTAGES OF PROPOSED SYSTEM

* Saves time
* Get current updates
* Easily clarify the doubts

# CHAPTER 3

# REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

A feasibility study is a comprehensive analysis and evaluation conducted to assess the

practicality, viability, and potential success of a proposed project, initiative, or investment. Its

main objective is to determine whether the project is technically, economically, socially, and

operationally feasible before proceeding with implementation. The study involves examining

various aspects, including technical requirements, resource availability, financial

considerations, social and environmental impacts, operational compatibility, and potential

risks. By conducting a feasibility study, decision-makers can gain valuable insights into the

project's likelihood of success, identify potential challenges, and explore alternative solutions.

### Economical Feasibility

Assesses whether the necessary software may bring an organization financial benefits. It includes the expenses related to the software development team, the expected cost of the necessary hardware and software, the expense of conducting a feasibility study, and so on. System that supplies crops to farmers must be carefully evaluated in terms of its financial sustainability and prospective profitability. In this analysis, costs for infrastructure, technology, personnel, crop procurement, transportation, and overhead expenses are all evaluated. The analysis covers various funding sources and looks at the initial investment needed for development and deployment. Pricing policies are developed to find a balance between farmer affordability and system sustainability. As a result of the economic feasibility assessment, stakeholders may make well-informed decisions and assure the system's financial stability, which benefits farmers directly and promotes a long-term crop exchange model within the community.

### Technical Feasibility

Assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget.

The technical feasibility study for a system that supplies crops to farmers looks at how well

the suggested solution can be created, put into practice, and run. The availability and suitability of necessary technology, as well as data collecting and integration skills, are important factors. The system should support existing agricultural technology and databases and have a user-friendly interface that is accessible to users with diverse levels of technical expertise. Making sure the system is technically prepared to promote the exchange of crops to farmers by enabling efficient and reliable operations with the right technology and infrastructure is achieved by conducting an extensive technical feasibility assessment.

### Behavioral Feasibility

Behavioral feasibility, also known as behavioral analysis, is an essential aspect of project or business feasibility studies. It assesses whether a proposed project, system, or business change can be accepted and adopted by the people who will use it. behavioral feasibility is about understanding and addressing the human and cultural aspects of a project to ensure its successful implementation. It is an essential component of feasibility studies to ensure that a project not only makes sense from a technical or financial perspective but is also supported by those it affects.

**3.1.4 Feasibility Study Questionnaire**

1. How the crops are given to the Farmers? Is there any criterias?

The crops are mainly given to the farmers mainly based on the annual income. If there is a farmer with high annual income and a one with the low annual income the crop will be provided to the one with the low income.

1. How the applications forms are submitted?

The application forms are filled and submitted as a hard copy.it is a form that need to be hand written.

1. If any data is not entered or any field is not added how the field can

be updated?

The panchayath together with the Ward Members will evaluate the form and if there is any corrections the Ward Member will inform the farmer to update it and give back the form.

1. How the farmers will get the updates?

The farmers will get the updates in the Gramasabha or they need to visit the panchayath for the new updations.

1. How will you inform them that their registration is approved or not?

We will inform them by calling them and inform them that the application is approved.

1. How the payment is done by the farmers?

The farmers can done the payment by cash.

1. Did you give any advices for growing the crops?

No not at all. We didn’t provide any advices.

1. How will you inform the farmers that their crops had arrived?

We will call or message the corresponding farmers and inform them.

1. From where the farmers will collect the crops?

The farmers can collect the crops from the Krishi Bhavan.

1. Did you provide any shops where the farmers can sell the products

that are grown from their crops?

No we did not have such facilities, farmers can to sell their products by their own.

## SYSTEM SPECIFICATION

### Hardware Specification

Processor - intel core i3

RAM - 4GB

Hard disk - 1TB

### Software Specification

Front End – HTML,CSS

Back End - Django

Database - MYSQL

Client on PC - Windows 7 and above.

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

## SOFTWARE DESCRIPTION

### Django

Django is a high-level Python web framework that simplifies and streamlines web development. It provides a structured and efficient way to build web applications and is known for its "batteries-included" philosophy, meaning it includes many built-in features and tools to help developers. Django is a versatile framework that simplifies web development by providing a robust set of tools and conventions. It is widely used for building web applications, from simple blogs to complex, data-driven websites.

### MySQL

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing.

MySQL is an important component of an open source enterprise stack called LAMP. LAMP is a web development platform that uses Linux as the operating system, Apache as the web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. (Sometimes Perl or Python is used instead of PHP.)

# CHAPTER 4

# SYSTEM DESIGN

* 1. **INTRODUCTION**

A crucial step in the creation of sophisticated software, hardware, and information systems is design of systems. It entails the planning and coordination of numerous elements to produce a unified and effective system that satisfies particular needs. System design offers several advantages in various domains

## UML DIAGRAM

Unified Modeling Language (UML) diagrams are a set of standardized graphical notations used in software engineering, systems engineering, and various other fields to visually represent and document various aspects of a system. These diagrams help communicate and analyze the design, structure, behavior, and interactions within a system. Here are some common UML diagram types and their definitions:

* Class Diagram
* Object Diagram
* Sequence Diagram
* Use-case Diagram
* Activity Diagram
* State-Chart Diagram
* Collaboration Diagram
* Deployment Diagram
* Component Diagram

## USE CASE DIAGRAM

Use-case diagrams aid in capturing system requirements and model a system's behavior in UML.

The scope and high-level functions of a system are described in use-case diagrams. The interactions between the system and its actors are also depicted in these diagrams. Use-case diagrams show what the system does and how the actors use it, but they do not show how the system works internally.

The context and requirements of either the entire system or the key components of the system are illustrated and defined by use-case diagrams. A complex system can be represented by a single use-case diagram, or its various components can be represented by a number of use-case diagrams. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.

The primary components of a use case diagram include:

* Actors:

Actors are outside parties who communicate with the system. Users, roles, or other systems may be included. On the diagram, actors are shown as stick figures.

* Use cases:

Use cases are particular services or functionalities that the system offers. On the diagram, each use case is represented by an oval and corresponds to a distinct aspect of system functionality.

* Associations:

The connections between actors and use cases show how they interact or relate to one another. An association shows that a particular actor is involved in a given use case.

* System boundary:

The system boundary, which is typically a box, encloses all the use cases and actors and establishes the scope of the system being modeled.



## 4.2.2 SEQUENCE DIAGRAM

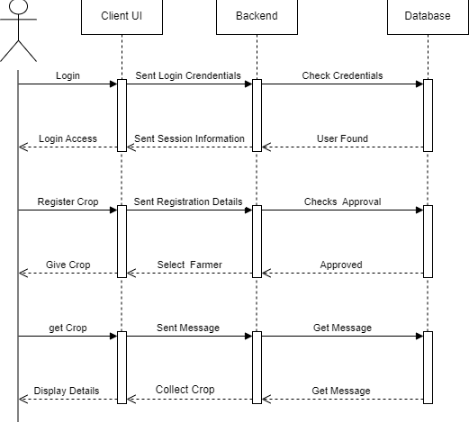
## A sequence diagram is a graphical representation used in software engineering and system design to visualize and describe the interactions and order of execution between various components or objects within a system. It provides a dynamic view of how different parts of a system communicate with one another and the sequence in which messages or method calls are exchanged.

## In a sequence diagram:

## Lifelines: These represent the entities (objects, components, or actors) involved in the interaction.

## Messages: Arrows and notations between lifelines indicate the messages or method calls being exchanged between the entities.

## Activation Bars: These show the duration of time during which an entity is active in processing a message.



## 4.2.3 State Chart Diagram

## A state chart diagram, also known as a state machine diagram or state diagram, is a visual representation used in software engineering and system modeling to describe the various states that an object, component, or system can exist in and the transitions between those states. It is particularly useful for modeling the behavior of systems with complex state-dependent logic. Here are the key components of a state chart diagram:

## *State:* A state represents a condition or situation in which an object or system can exist. Each state is depicted as a rectangle with a label describing the state's name or characteristics.

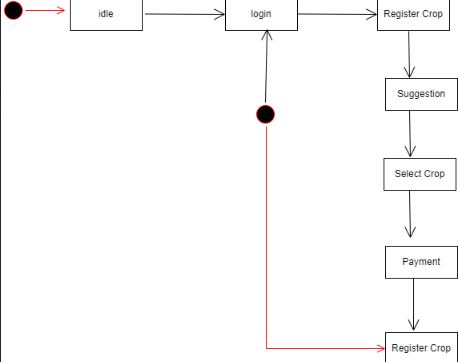
## *Initial State:* An initial state is the starting point of a state machine. It is often marked with a small filled circle connected to the initial state by an arrow.

## *Final State:* A final state represents the end of a particular state or the completion of a process. It is typically marked with a small filled circle enclosed by a larger circle.

## *Transition:* A transition represents a change of state and is depicted as an arrow connecting two states. It is labeled with an event that triggers the transition and a condition (if any) that must be satisfied for the transition to occur.

## *Event:* An event is an occurrence or trigger that initiates a state transition. Events are associated with transitions and can be external stimuli, user actions, or time-based events.

## *Action:* Actions are executable operations or behaviors associated with a state or transition. They represent what happens when a state is entered, exited, or when a transition occurs.



## 4.2.4Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram that is used to represent the workflow or flow of activities within a system, process, or business. It is a visual tool that helps model, understand, and document the steps or actions in a process or system.

Key components of an activity diagram include:

***Activity:*** Activities are represented as rounded rectangles and represent specific tasks, actions, or operations in the process. They can be detailed or high-level, depending on the level of abstraction needed.

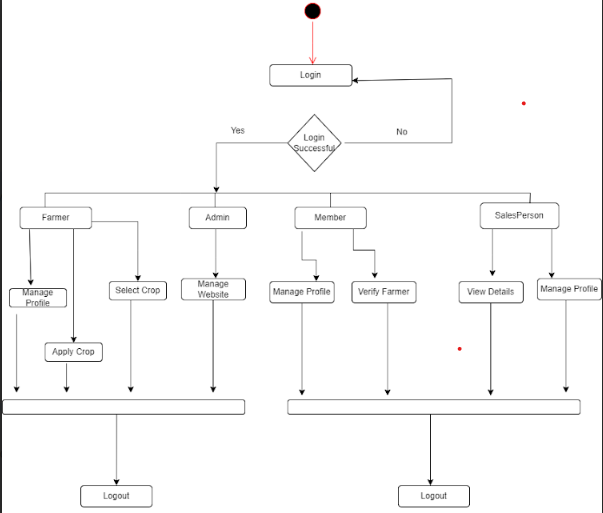
***Action:*** Actions are represented by rectangles with rounded corners and represent individual steps within an activity. They provide a more granular view of the work being done.

***Control Flow:*** Control flow arrows show the sequence in which activities and actions are executed. They connect activities and actions, indicating the order of execution.

***Decision Nodes:*** Decision nodes are represented as diamonds and are used to represent decision points in the process. Depending on conditions, the process can take different paths.

***Merge Nodes:*** Merge nodes, also shown as diamonds, represent points where different paths converge.

***Forks and Joins:*** Forks represent points where the process splits into multiple parallel activities, while joins indicate where those parallel paths converge back into a single flow.

Activity diagrams are valuable for business process modeling, software design, and system analysis. They are especially useful for representing complex processes with many interacting steps, decisions, and concurrent activities. 

## 4.2.5 Class Diagram

A class diagram is a type of UML (Unified Modeling Language) diagram used in software engineering to visualize and document the structure and relationships of classes and objects within a system. It provides a high-level overview of the system's structure and serves as a blueprint for designing and implementing software systems. Class diagrams are an essential part of object-oriented modeling and are commonly used during the software development process.

Components of a Class Diagram:

***Class:*** Represents a blueprint or template for creating objects.

Contains attributes (data members) and methods (functions) that define the behavior of objects of that class.

***Attributes:*** Variables that represent the data or characteristics of a class.

Typically displayed as name: type pairs (e.g., name: string).

***Methods:*** Functions or operations that define the behavior of a class.

Represented with their names, input parameters, and return types.

***Associations:*** Lines connecting classes to indicate relationships between them.

Multiplicity notations (e.g., 0..1, 1, 1..\*) describe how many objects participate in the relationship.

***Inheritance (Generalization):*** Represents an "is-a" relationship where one class (subclass or child) inherits attributes and methods from another class (superclass or parent).

***Interfaces:*** Specifies a contract that a class must adhere to.

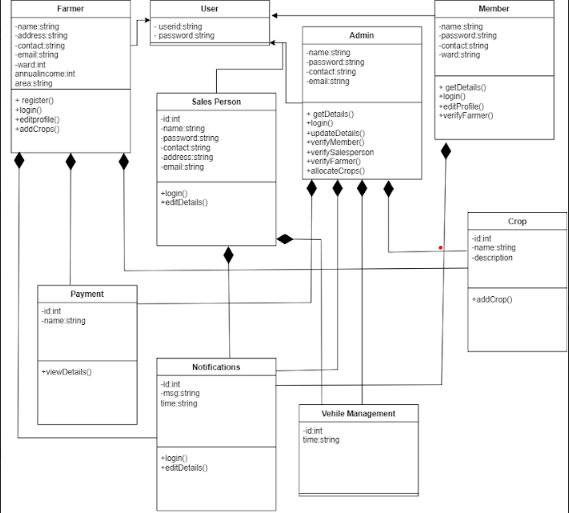
A class can implement one or more interfaces, ensuring that it provides specific methods.

***Aggregation and Composition:*** Aggregation represents a "has-a" relationship where one class contains or is part of another class.

Composition is a stronger form of aggregation, indicating that the parts belong exclusively to the whole.

***Dependency:*** Shows that one class relies on another class but is not tightly coupled to it.

Class diagrams are used for design and documentation purposes, helping software developers, designers, and other stakeholders understand the structure and relationships within a software system.



## 4.2.6 Object Diagram

An object diagram is a structural diagram in the Unified Modeling Language (UML) that represents instances of classes or objects at a specific point in time within a system or a specific scenario. It provides a detailed view of the relationships and interactions between objects, offering a snapshot of how objects collaborate at a particular moment.

Key Features of Object Diagrams:

***Objects:*** In an object diagram, individual objects or instances of classes are depicted. These objects can represent real-world entities or software components.

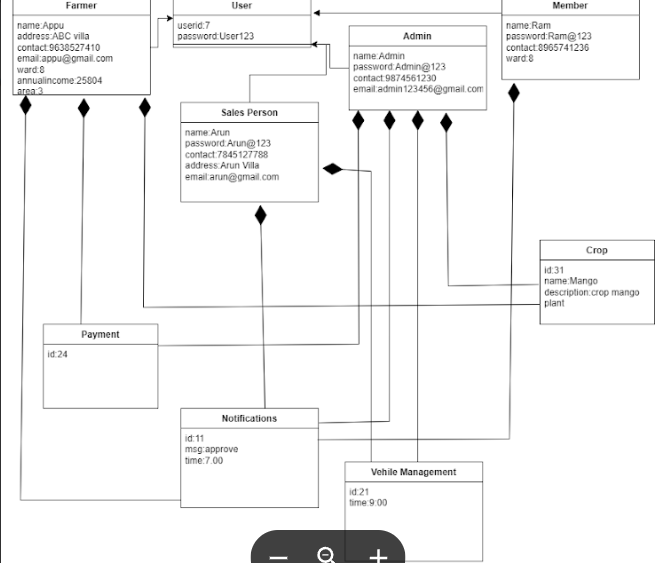
***Links:*** Object diagrams illustrate the relationships or associations between objects. Links between objects show how they are connected or interact with each other.

***Attributes:*** Object diagrams may display object attributes and their values, providing a detailed look at the state of objects at the given moment.

***Multiplicity:*** Object diagrams can show the cardinality of associations, indicating how many objects participate in a particular relationship.

***Scenario-Specific:*** Object diagrams are typically scenario-specific, meaning they depict objects and their relationships within a specific context or use case.

Object diagrams are useful for visualizing and verifying how objects collaborate and communicate within a system during a particular scenario or event.



## 4.2.7 Component Diagram

A component diagram is a type of structural diagram in the Unified Modeling Language (UML) that represents the high-level architecture and structure of a system or application. It focuses on the organization of components, which can be software modules, libraries, or other encapsulated parts of a system. Component diagrams are used to model the physical and logical aspects of a system's components and how they interact.

Key Features of Component Diagrams:

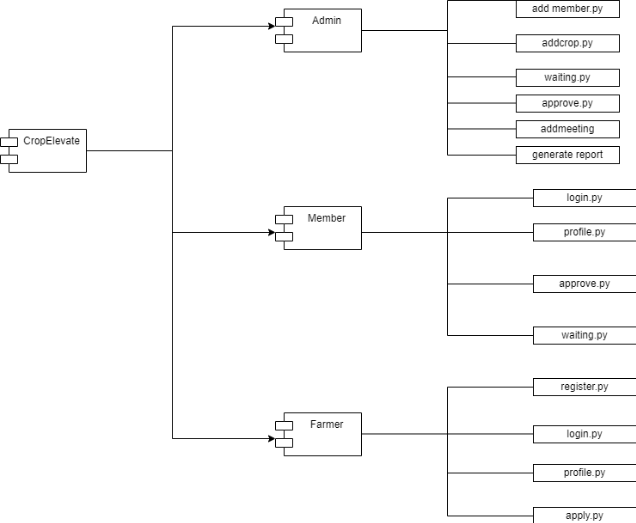
***Components:*** In a component diagram, components are represented as boxes or rectangles, each depicting a discrete part of the system. These components can be at different levels of abstraction, such as individual classes, packages, or larger system modules.

***Relationships:*** Component diagrams show the relationships and dependencies between components. These relationships can include associations, dependencies, aggregations, and more, indicating how components interact with each other.

***Interfaces:*** Components can have interfaces that define the services or functionality they provide or require from other components. Interfaces are depicted using lollipops or sockets.

***Ports:*** Ports are used to represent the interaction points of components. They show where connections, such as communication channels or data flows, are established between components.

***Connectors:*** Connectors are lines or arrows that illustrate the connections between components. They depict the flow of data or communication pathways between components.



**4.2.8 Deployment Diagram**

A deployment diagram is a type of diagram in the Unified Modeling Language (UML) that focuses on the physical deployment of software components within a system or application. It illustrates how software artifacts, such as executable files, libraries, and components, are distributed across hardware nodes and interconnected to form a complete system.

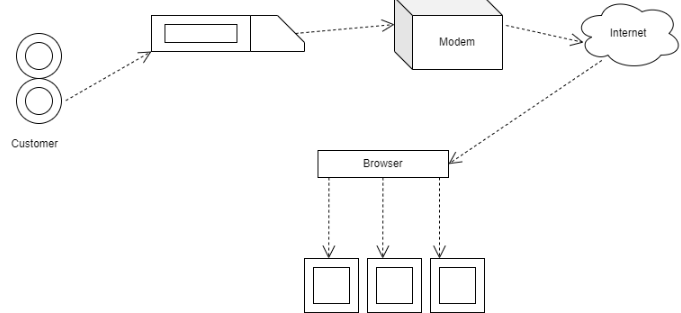
Key Features of Deployment Diagrams:

***Nodes:*** In a deployment diagram, nodes represent hardware entities such as servers, workstations, or devices. These nodes serve as the execution environments for the software components.

***Artifacts:*** Software artifacts, including components, files, and databases, are represented in the diagram to illustrate their deployment on specific nodes.

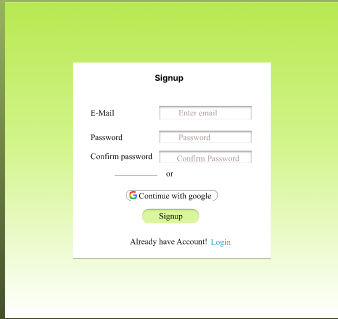
***Relationships:*** Deployment diagrams show the relationships between nodes and artifacts. These relationships can include associations, dependencies, and usage relationships, indicating how software components interact with hardware nodes.

***Deployment Specifications:*** Deployment specifications can be associated with artifacts to provide additional information about deployment options, configurations, and requirements.

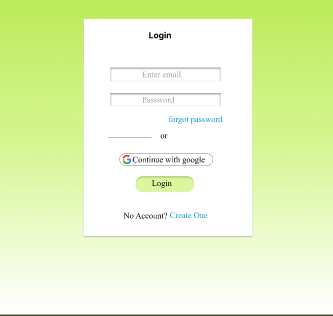
Deployment diagrams are valuable for system architects and developers to visualize the physical infrastructure and software distribution of a system. They help in planning and understanding how software components will be deployed on hardware, including considerations for scalability, performance, and redundancy.

## 4.3 USER INTERFACE DESIGN USING FIGMA

**Form Name: signup**



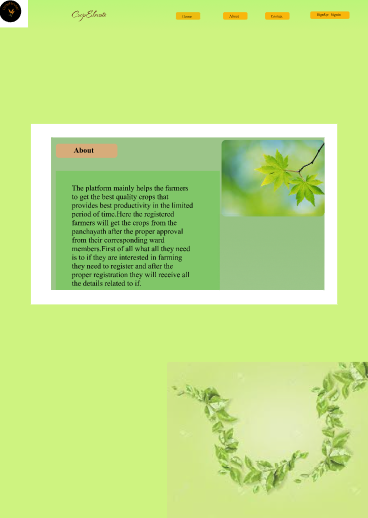
**Form Name: Login**



**Form Name: Homepage**



**Form Name: About**



**Form Name: Contact**



## 4.4 DATABASE DESIGN

## Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

## The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

### 4.4.1 Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) is a type of database management system that organizes and stores data in a structured format, primarily using tables with rows and columns. It is based on the principles of the relational model, which was introduced by Edgar F. Codd in the 1970s. RDBMSs are designed to manage and manipulate data efficiently while maintaining the integrity and relationships between different data elements.

Key characteristics of RDBMS:

***Tables:*** Data is organized into tables, also known as relations, where each table consists of rows (tuples) and columns (attributes). Each row represents a unique record, while each column represents a specific piece of data.

***Schema:*** RDBMSs define a schema that specifies the structure of tables, including data types, constraints, and relationships between tables. This ensures data consistency and integrity.

***SQL (Structured Query Language):*** RDBMSs use SQL as the standard language for querying and manipulating data. SQL allows users to perform operations like selecting, inserting, updating, and deleting data.

***ACID Properties:*** RDBMSs ensure data integrity through ACID (Atomicity, Consistency, Isolation, Durability) properties. These properties guarantee that database transactions are processed reliably.

***Data Relationships:*** RDBMSs support the establishment of relationships between tables, enabling the creation of complex queries that retrieve related data from multiple tables.

***Normalization:*** RDBMSs promote data normalization, which reduces data redundancy and enhances data integrity by minimizing the risk of update anomalies.

Popular RDBMSs include MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite. These systems are widely used in various applications, ranging from enterprise-level data management to web development and mobile applications, due to their reliability and data consistency features.

### 4.4.2 Normalization

Normalization in the context of databases refers to the process of organizing and structuring data within a relational database to eliminate data redundancy and ensure data integrity. It involves breaking down large tables into smaller, related tables and establishing relationships between them. The main objective of normalization is to reduce the potential for data anomalies, such as update anomalies, insert anomalies, and delete anomalies, which can occur when data is not properly structured. The process typically involves defining a set of rules or normal forms, including First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), and beyond, to guide the design and organization of database tables. Each normal form has specific criteria that a table must meet to be considered normalized. Normalization is a critical step in database design, ensuring efficient data storage, retrieval, and maintenance while minimizing data inconsistencies.

***First Normal Form (1NF):*** In 1NF, a table is considered normalized if it has no repeating groups (columns with multiple values), and all attributes contain atomic (indivisible) values. It ensures that each cell in a table contains a single, indivisible value.

***Second Normal Form (2NF):*** 2NF builds on 1NF and further eliminates partial dependencies. To be in 2NF, a table must meet 1NF criteria, and all non-key attributes must depend on the entire primary key, not just part of it. This avoids redundancy related to composite primary keys.

***Third Normal Form (3NF):*** 3NF extends 2NF by removing transitive dependencies. A table is in 3NF if it meets 2NF criteria and has no attributes that depend on other non-key attributes. It eliminates indirect relationships between attributes.

***Boyce-Codd Normal Form (BCNF):*** BCNF is a stricter form of 3NF that ensures that for every non-trivial functional dependency, the left-hand side is a superkey. It helps avoid anomalies related to key attributes.

***Fourth Normal Form (4NF):*** 4NF deals with multi-valued dependencies, ensuring that if an attribute is multi-valued, it's stored in a separate table with a relationship to the main table.

***Fifth Normal Form (5NF):*** 5NF addresses cases where an attribute depends on a combination of super keys and is often used in specialized situations to further reduce data redundancy.

### 4.4.3 Sanitization

sanitization refers to the process of cleansing and validating input data to ensure that it is safe and does not pose security risks to the database or the application using it. Sanitization is a critical practice to prevent various security vulnerabilities, including SQL injection, Cross-Site Scripting (XSS), and other forms of cyberattacks.

sanitization is a fundamental practice in database design to protect against vulnerabilities and ensure that data is stored and retrieved safely. It is a crucial element in maintaining the integrity and security of the database and the applications that interact with it.

**4.4.4 Indexing**

Indexing in the context of database design refers to the process of creating data structures that improve the speed of data retrieval operations on a database table. These data structures, called indexes, work like an organized catalog of the data in a table, making it easier and faster to find specific rows based on the values in one or more columns. Indexing is crucial for optimizing query performance in database systems.

### TABLE DESIGN

***1. Tbl\_Login***

Primary key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key constraint** |
| 1 | Login\_id | int | Primary key, Auto Increment |
| 2 | Status | int |  |
| 3 | Email | VARCHAR | NOT NULL |
| 4 | Password | VARCHAR | NOT NULL |
| 5 | Approve | VARCHAR | NOT NULL |

***2.Tbl\_Profile***

Primary key: Profile\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key constraint** |
| 1 | Profile\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Name | VARCHAR | NOT NULL |
| 4 | Address | VARCHAR | NOT NULL |
| 5 | Place | VARCHAR | NOT NULL |
| 6 | Ward | VARCHAR | NOT NULL |
| 7 | Contact | VARCHAR | NOT NULL |
| 8 | Dob | VARCHAR | NOT NULL |
| 9 | Land | VARCHAR | NOT NULL |
| 10 | Annual income | VARCHAR | NOT NULL |
| 11 | Licence no | VARCHAR | NOT NULL |

3. ***Tbl\_Land***

Primary key:Land\_id

Foreign key:profile\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Land \_id | Int | Primary Key, Auto Increment |
| 2 | Profile\_id | Int | Foreign Key |
| 3 | Land (acer) | VARCHAR | NOT NULL |
| 4 | Ward | VARCHAR | NOT NULL |

***4. Tbl\_Blog***

Primary key: Blog\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Blog\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Description | Text | NOT NULL |
| 4 | Author | VARCHAR | NOT NULL |
| 5 | Date | VARCHAR | NOT NULL |
| 6 | Approve | VARCHAR | NOT NULL |

***5. Tbl\_Feedback***

Primary key: Field\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key constraints** |
| 1 | Field\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Feedback | VARCHAR | NOT NULL |
| 4 | Status | VARCHAR | NOT NULL |

***6. Tbl\_Contact***

Primary key: Contact\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Contact\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Contact no | Int | NOT NULL |
| 4 | Email | VARCHAR | NOT NULL |
| 5 | Message | VARCHAR | NOT NULL |

***7. Tbl\_Crop***

Primary key: Crop\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Crop\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Crop Name | VARCHAR | NOT NULL |
| 4 | Yield (After how many years) | VARCHAR | NOT NULL |

***8. Tbl\_farmer\_crop***

Primary key: Fc\_id

Foreign key: Login\_id, Crop\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Fc\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Crop\_id | Int | Foreign Key |
| 4 | Details | VARCHAR | NOT NULL |

***9. Tbl\_Suggestion***

Primary key: Suggestion\_id

Foreign key: Login\_id, Fc\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Suggestion\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Fc\_id | Int | Foreign Key |
| 4 | Suggestion | VARCHAR |  |

***10. Tbl\_Approval***

Primary key: Approval\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Approval\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Approval message | VARCHAR | NOT NULL |
| 4 | Approval Date | VARCHAR | NOT NULL |
| 5 | Approval time | VARCHAR | NOT NULL |

***11. Tbl\_Distribution***

Primary key: Distribution\_id

Foreign key: Login\_id, Crop\_id, Approval\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Distribution\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Crop\_id | Int | Foreign Key |
| 4 | Approval\_id | Int | Foreign Key |
| 5 | Distributor name | VARCHAR | NOT NULL |
| 6 | Date | VARCHAR | NOT NULL |

***12. Tbl\_Collection\_Crop***

Primary key: Collection\_id

Foreign key: Login\_id, Crop\_id, Approval\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Collection\_id | Int | Primary Key, Auto Increment |
| 2 | Approval\_id | Int | Foreign Key |
| 3 | Login\_id | Int | Foreign Key |
| 4 | Crop\_id | Int | Foreign Key |
| 5 | Collection Date | VARCHAR | NOT NULL |

***13. Tbl\_Collection\_SalesPerson***

Primary key: Product\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Product\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Product Details | VARCHAR | NOT NULL |
| 4 | Date | VARCHAR | NOT NULL |

***14. Tbl\_Distribution\_Shop***

Primary key: Shop\_id

Foreign key: Login\_id, Product\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Shop\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Product\_id | Int | Foreign Key |
| 4 | Details(amount of Product) | VARCHAR | NOT NULL |

***15. Tbl\_Payment***

Primary key: Payment\_id

Foreign key: Login\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Data\_Type** | **Key Constraints** |
| 1 | Payment\_id | Int | Primary Key, Auto Increment |
| 2 | Login\_id | Int | Foreign Key |
| 3 | Amount | VARCHAR | NOT NULL |
| 4 | Message | VARCHAR | NOT NULL |

# CHAPTER 5

# SYSTEM TESTING

* 1. **INTRODUCTION**

System testing is the process of assessing a software application or system as a whole to verify that it performs according to its specified requirements and functions effectively in its intended environment. This type of testing evaluates the system's behavior and interactions to identify defects, ensure reliability, and confirm that it aligns with user expectations.

Key aspects of system testing include:

Functional Testing: Verifying that the software performs its intended functions correctly, addressing features, and requirements as specified in the design and requirements documents.

Non-Functional Testing: Evaluating non-functional aspects, such as performance, scalability, security, and usability to ensure the system's quality and robustness.

Integration Testing: Assessing how different components or modules within the system work together and interface seamlessly.

End-to-End Testing: Testing the system as a whole to ensure that data and processes flow smoothly from end to end.

Regression Testing: Ensuring that system changes or updates do not introduce new defects or disrupt existing functionalities.

System testing is performed after unit and integration testing and serves as a comprehensive assessment before the software's release to ensure that it meets user expectations and performs reliably in its operational environment.

## TEST PLAN

A test plan is a formal document that defines the scope, objectives, resources, schedule, and approach for testing a software system. It outlines the testing strategy, test cases, and criteria for success, serving as a guide for testers, developers, and stakeholders. The primary goal of a test plan is to ensure that the software is thoroughly tested to identify defects and validate that it meets its intended requirements and quality standards.

Key components of a test plan include:

Introduction: Provides an overview of the software under test, its purpose, and the objectives of testing.

Scope: Defines what aspects of the software will be tested and any items that are out of scope.

Testing Objectives: Specifies the goals of testing, including what needs to be achieved or validated.

Test Strategy: Outlines the overall approach to testing, including the types of testing (e.g., functional, performance, security), and the testing environments.

Test Schedule: Provides a timeline for testing activities, including start and end dates for different testing phases.

Test Cases: Describes individual test cases, including input data, expected results, and dependencies.

Resources: Identifies the roles and responsibilities of team members, as well as the tools and infrastructure required for testing.

Risks and Mitigation: Highlights potential risks during testing and strategies to mitigate them.

Exit Criteria: Specifies the conditions that must be met to consider testing complete.

Approvals: Identifies stakeholders responsible for approving the test plan.

### Unit Testing

Unit testing is a software testing technique where individual units or components of a software application are tested in isolation. A unit, in this context, is the smallest testable part of the software, typically a function, method, or class. The primary goal of unit testing is to verify that each unit of code performs as expected and meets its specified requirements.

### Integration Testing

Integration testing is a software testing technique that evaluates the interactions and data flow between interconnected components, modules, or services within a software application. The primary objective is to identify and rectify issues that may arise when these units are combined. This testing phase ensures that various parts of the software work harmoniously, data is exchanged correctly, and interfaces between components function as intended.

### Validation Testing or System Testing

Validation testing and system testing are both integral parts of the software testing process, each serving distinct purposes in ensuring software quality and reliability. Validation testing, often referred to as acceptance testing, is the process of evaluating a software system to determine whether it meets the specified requirements and satisfies the needs of the end users or stakeholders. It focuses on validating that the software performs as intended and delivers the expected functionality.

### Output Testing or User Acceptance Testing

Output Testing, also known as User Acceptance Testing (UAT), is a crucial phase in the software testing process. It focuses on verifying that the software system produces the correct output or results and meets the end-users' requirements and expectations.

* + 1. **Automation Testing**

Automation Testing is a critical aspect of software testing that involves using automated tools and scripts to perform tests on a software application.

Automation testing is the process of using automation tools, frameworks, and scripts to execute test cases and verify the functionality of a software application automatically.

* + 1. **Selenium Testing**

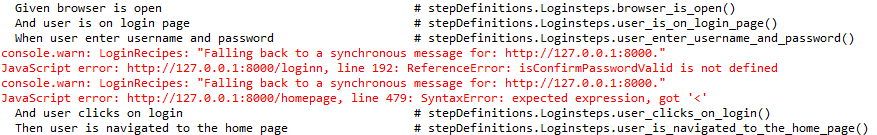
Selenium Testing is a widely-used open-source tool for automating web browsers. It allows testers and developers to write scripts in various programming languages to interact with web applications and automate browser actions.

**Test Case 1:**

**Code**



**Screenshot**

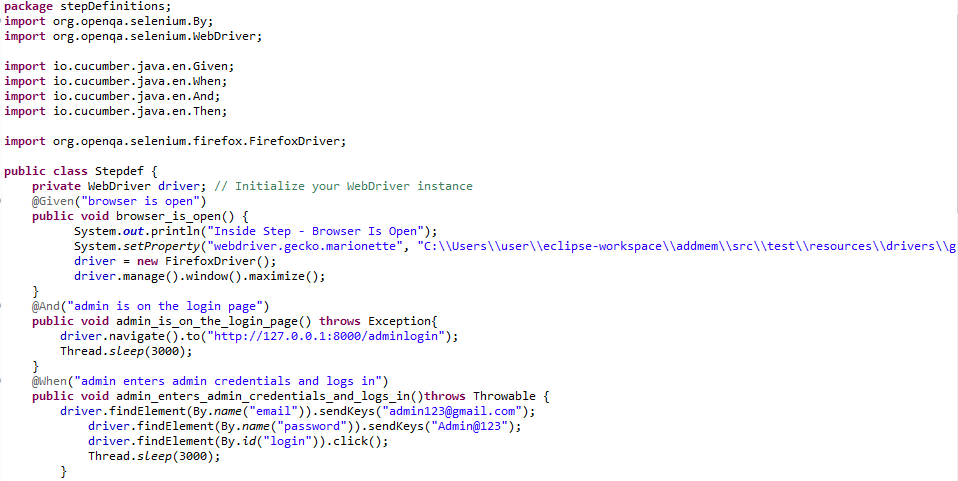


**Test Report**

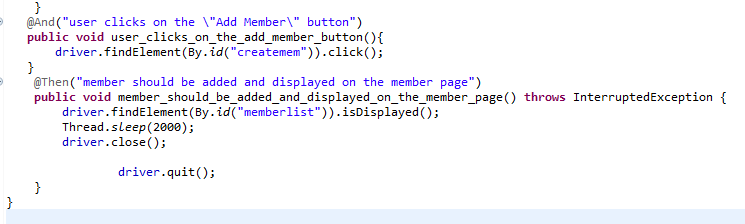
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name: CropElevate** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID:** 1 | | | **Test Designed By:** Aleena Joseph | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** | | |
| **Module Name**: Login Page | | | **Test Executed By:** Mr. G S Ajith | | |
| **Test Title:** User Login | | | **Test Execution Date:** | | |
| **Description**: Verify the login page with valid email and password | | |  | | |
| **Pre-Condition:** User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Navigate to login page |  | Dashboard should displayed. | Login page is displayed | Pass |
| 2 | Provide valid email | anna@gmail.com | User should be able to login | User logged in the user dashboard is displayed | pass |
| 3 | Provide valid password | Anna@123 |
| 4 | Clicks on login button |  |
| 5 | Navigate to homepage |  |  |  |  |
|  |  |  |  |  | pass |
| **Post-Condition:** | | | | | |

**Test Case 2:**

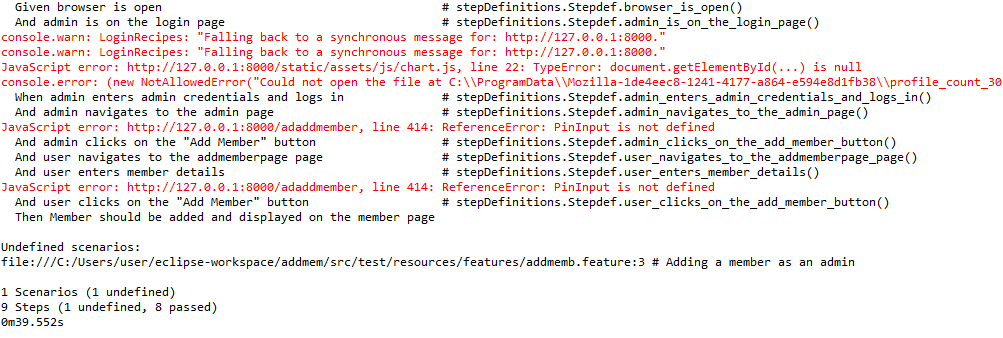
**Code**

****

****

****

**Screenshot**

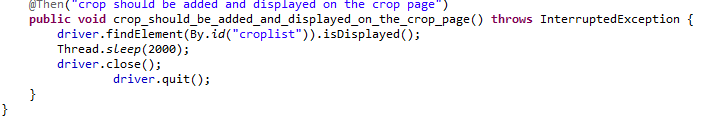
****

**Test report**

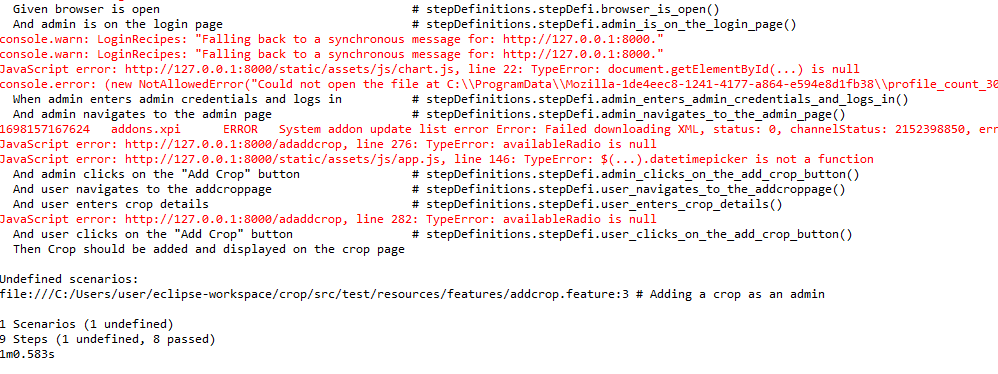
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 2** | | | | | |
| **Project Name: CropElevate** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID:** 2 | | | **Test Designed By:** Aleena Joseph | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** | | |
| **Module Name**: Add Member Page | | | **Test Executed By:** Mr. G S Ajith | | |
| **Test Title:** Adding Member | | | **Test Execution Date:** | | |
| **Description**: To add a member by the admin | | |  | | |
| **Pre-Condition:** admin has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Admin is on login page |  | Dashboard should displayed. | Login page is displayed | Pass |
| 2 | Provide the Email | Admin123@gmail.com | Admin login | Admin logged in the admin dashboard is displayed | pass |
| 3 | Provide valid password | Admin@123 |
| 4 | Admin navigates to admin page |  |
| 5 | Admin clicks on add member button. |  | Admin click on add member button | Admin clicks on add member button |  |
|  |  |  | Admin enters the member details | Admin enters the member details | pass |
| 6 | Navigates to add member page |  | clicks on the create member button the member is created | clicks on the create member button the member is created | Pass |
| 7 | Enter the member details |  |  |  |  |
| 8 | Click on add member button. |  |  |  |  |
| 9 | Member should be added and displayed in the admin member page |  |  |  |  |
| **Post-Condition:** | | | | | |

**Test Case 3:**

**Code**

**Screenshot**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 3** | | | | | |
| **Project Name: CropElevate** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID:** 3 | | | **Test Designed By:** Aleena Joseph | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** | | |
| **Module Name**: Add crop Page | | | **Test Executed By:** Mr. G S Ajith | | |
| **Test Title:** Adding Crop | | | **Test Execution Date:** | | |
| **Description**: To add a crop by the admin | | |  | | |
| **Pre-Condition:** Admin has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Admin is on login page |  | Dashboard should displayed. | Login page is displayed | Pass |
| 2 | Provide the Email | Admin123@gmail.com | Admin login | Admin logged in the admin dashboard is displayed | pass |
| 3 | Provide valid password | Admin@123 |
| 4 | Admin navigates to admin page |  |
| 5 | Admin clicks on add crop button. |  | Admin click on add crop button and | Admin clicks on add crop button and |  |
|  |  |  | Admin enters the crop details | Admin enters the crop details | pass |
| 6 | Navigates to add crop page |  | clicks on the create crop button the crop is created | clicks on the create crop button the crop is created | Pass |
| 7 | Enter the crop details |  |  |  |  |
| 8 | Click on add crop button. |  |  |  |  |
| 9 | Crop should be added and displayed in the admin crop page |  |  |  |  |
| **Post-Condition:** | | | | | |

# CHAPTER 6

# IMPLEMENTATION

## INTRODUCTION

Explanation

## IMPLEMENTATION PROCEDURES

Explanation

### User Training

Explanation

### Training on the Application Software

Explanation

### System Maintenance

Explanation

# CHAPTER 7

# CONCLUSION AND FUTURE SCOPE

## CONCLUSION

## 

.

* 1. **FUTURE SCOPE**

.

# CHAPTER 8

# BIBLIOGRAPHY

### REFERENCES:

* + - ..
    - ..
    - ..
    - ..
    - ...

### 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