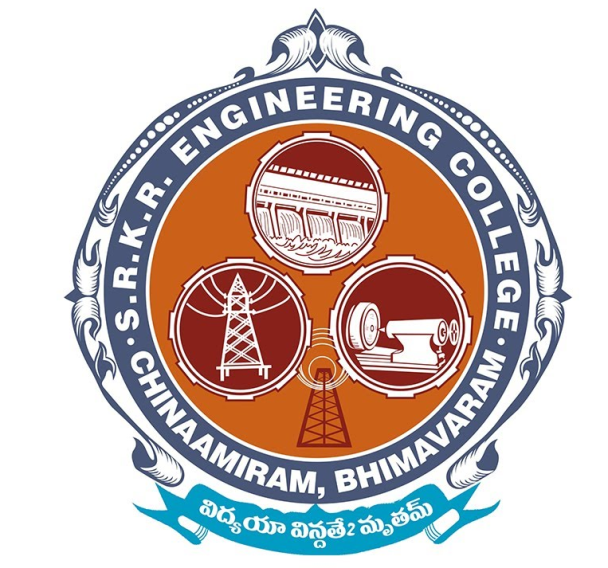
**Project Report**

**On**

**FARMING WEB ASSISTANCE SERVICE**

****

**Submitted By**

**Ms.K. Pushpa lekha Regd No: 315175710106**

**Mr.K. Jaswanth kumar Regd No: 315175710103**

**Mr.K. Sai sudhakar Regd No: 315175710088**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**S.R.K.R ENGINEERING COLLEGE**

**(Affiliated to ANDHRA UNIVERSITY, VISAKHAPATNAM)**

**BHIMAVARAM-534204**

**(2018-2019)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**S.R.K.R ENGINEERING COLLEGE**

(Affiliated to Andhra University)



**CERTIFICATE**

This is to certify that this project work entitled**“FARMING WEB ASSISTANCE**

**SERVICE”** is the bonafied work of **Ms.K. Pushpa lekha(315175710106), Mr. K. Jaswanth kumar (315175710103), Mr. K. Sai Sudhakar (315175710088),** a final

year Bachelor of technology in **object-oriented software engineering lab** have submitted during the academic year 2018-2019.

**Project Guide:**

M.Janaki Devi Prof. G.V. Padma Raju Assistant professor Head of the department

Department of CSE, Department of CSE,

SRKR engineering college, SRKR engineering college,

Bhimavaram Bhimavaram

**ACKNOWLEDGEMENT**

Initially I must be thankful to persons who supported me in all aspects of my project. I convey my sincere thanks to **Sri M.Janaki Devi**, Assistant professor, and Sri Kranthi., Assistant professor, computer science and Engineering, S.R.K.R Engineering College, Bhimavaram for their kind cooperation in the completion of work.

I express my heartful thanks to **Prof. G.V PADMA RAJU,** Head of the Department of computer science and engineering. S.R.K.R Engineering college, Bhimavaram for providing an initiative to this project and giving valuable timely suggestions for the work.

I express my sincere thanks to **Dr. G.P.S.VARMA**, principal, S.R.K.R Engineering College, Bhimavaram for giving me the opportunity for the successful completion of my degree.

I also thanks to other teaching and non-teaching staff for their assistance and help extended. I thank one and all that have contributed directly or indirectly to my project.

**Abstract:**

Farming assistant web service is a web project to help the farmers working with the motive of greater profitability by direct communication between farmer-to- supplier .This service boosts business communication and brings

transparency in the system. This innovative site

allows a good farmer, retailer and supplier

communication.

**INDEX**

**LIST OF FIGURES**

1. **INTRODUCTION**

1.1 SOFTWARE REQUIREMENT SPECIFICATION

1.1.1 INTRODUCTION

1.1.2 SCOPE

1.1.3 EXISTING SYSTEM

1.1.4 PROPOSED SYSTEM

* 1. REQUIREMENT ANALYSIS
     1. FUNCTIONAL REQUIREMENTS
     2. NON-FUNCTIONAL REQUIREMENTS
     3. SOFTWARE REQUIREMENTS
     4. HARDWARE REQUIREMENTS

1. **UML DESIGN**
   1. UML
      1. BASIC BLOCKS OF UML
      2. SOFTWARE ENGINEERING ACTIVITIES
      3. RATIONAL ROSE
   2. USE CASE DIAGRAM
   3. CLASS DIAGRAM
   4. SEQUENCE DIAGRAM
   5. STATE CHART DIAGRAM
   6. ACTIVITY DIAGRAM
   7. COLLABORATION DIAGRAM

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME OF FIGURE** | **FIGURE NO** | **PAGE NO** |
| **1.** | USE CASE DIAGRAM | 2.2 |  |
| **2.** | CLASS DIAGRAM | 2.3 |  |
| **3.** | SEQUENCE DIAGRAM | 2.4.1, 2.4.2, 2.4.3 |  |
| **5.** | STATE CHART DIAGRAM | 2.5,2.5.1,2.5.2 |  |
| **6.** | ACTIVITY DIAGRAM | 2.6,2.6.1,2.6.2 |  |
| **6** | COLLABORATION DIAGRAM | 2.7.1, 2.7.2 |  |
|  |  |  |  |
|  |  |  |  |

**CHAPTER-1**

**INTRODUCTION**

**Software Requirement specification:**

**1.1.1 Introduction:**

This project provides an option of login to farmers and communicates to respective dealers.  The farmers also have an option to submit their grievances and complaints to respective dealers or authorities using their farmer login on a separate complaints page and authorities will get access to that page regularly using their login id and passwords.

**1.1.2 Scope of the project:**

The aim of the project is to help the farmers for their betterment and obtain greater profit by having direct farmer to suppliers communications.

This project aims to create awareness, boost business communication and transparency in the system by having direct farmer to supplier and farmer to farmer communication.

Also it does notify the farmer regarding any advertisement or offer published by the govt. or dealer.

**1.1.3 EXISTING SYSTEM**

The existing system of agriculture is very unpleasant and nasty. The unawareness about the quality of the seed enforce the farmer to buy low price seeds which leads to low productivity and these products are sold to the middlemen for a low price which results in a great loss for the farmer.

**Disadvantages:**

1. Verification of the users cannot be done online. They have to first get themselves verified by visiting before using the system.
2. It requires a large database and memory.

**1.1.4 PROPOSED SYSTEM**

Proposed system is very much constructive and pleasing. In this system farmer can directly communicate with retailers and suppliers without taking help of any third parties. Also they get information regarding the offers and advertisement published by govt. or any other dealers which eradicate the exploitation of middlemen. Farmers may also submit their grievances and complaints which will be taken care by reporting authorities.

**Advantages:**

1. Farmers can directly contact suppliers by searching online.
2. Farmers may submit their grievances online.
3. Farmers get notification of any new offers/schemes.

**1.2 REQUIREMENT ANALYSIS**

Analysis focuses on producing a model of the system called the analysis model, which is correct, complete, consistent and verifiable. Analysis model is composed of three individual models.

* Functional model
* Analysis object model
* Analysis dynamic model

The functional model is represented with use case diagrams, the object model with class diagrams, and the dynamic model with state chart and sequence diagrams.

**1.2.1 FUNCTIONAL REQUIREMENTS**

**Modules:**

* **Supplier:** Once supplier is registered he can log in to the portal by using his credentials. Supplier can post advertisement by specify the name of the crop he required. If any farmer is ready to sell that particular crop supplier can see the details of it in his crop received section.
* **Farmer:** Farmer can view all the advertisement posted by suppliers. If farmer is ready to accept all the terms and conditions given by supplier, he can sell his product to supplier. Even farmer can post his complaints regarding any issue related to product, supplier, sells etc. which will be handled by reporting authorities. Also farmer can view the status of his complaints. Again farmer can view various tips posted by admin which are very much advantageous and useful for farming.
* **Admin:** admin is responsible to view all the complaints posted by farmers and take necessary action. Admin can also post various farming tips for the farmers.

**1.2.2 NON-FUNCTIONAL REQUIREMENTS**

Non-Functional requirements are the limitations or constraints under which system should provide its services to users. Following are the non-functional requirements for virtual classroom system:

**Reliability:**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

**Cost:**

The cost of product development and maintenance is minimum. As this project uses all open source tools to develop, the cost of construction is low.

**Performance:**

This system is developed in the high-level languages and using the advanced front end and back end technologies. It will give response to the end user with in the less time.

**Supportability:**

The system is designed to the cross platform supportable. The system is supported on the wide range of hardware and any software platform, which is having JVM, built into the system.

**Security:**

This avoids the non-authorised users access to the virtual classroom.

**1.2.3 SOFTWARE REQUIREMENTS**

* Operating System: Windows XP or Higher Version
* User Interface: Html/CSS
* Client Side Scripting : Javascript
* Programming Language: Java
* Web Application: JDBC, Servlet and PHP
* Database: Oracle 10g
* Server Deployment: tomcat 5.x C.

**1.2.4 HARDWARE REQUIREMENTS**

* Intel Pentium IV or Higher processor
* 1.80 GHz
* 1 GB of RAM

**CHAPTER- 2**

**UML DESIGN**

**2.1 UML**

The UML is a graphical language for specifying, visualizing, documenting. All the three stages of Object Oriented development such as OOA, OOD and OOP can make use of UML.

Advantages of modelling:

* Makes easy to enhance, and manipulation of existing system
* Helps to carry helps in visualizing the system to be developed
* Permits to specify the structure and behaviour of the system
* Used as a template to construct a proposed system
* For capturing the requirements

**2.1.1 BASIC BLOCKS OF UML**

UML has 3 basic blocks

**1. THINGS**

* 1. **STRUCTURAL THINGS:**

The Structural things define the static part of the model. They represent physical and conceptual elements. Following are the brief description of the structurel things.

**Class:** Class represents set of objects having similar responsibilities.

|  |
| --- |
| Class |
| Attributes |
| Operations |

**Interface:** Interface defines a set of operations which specify the responsibility of a class.

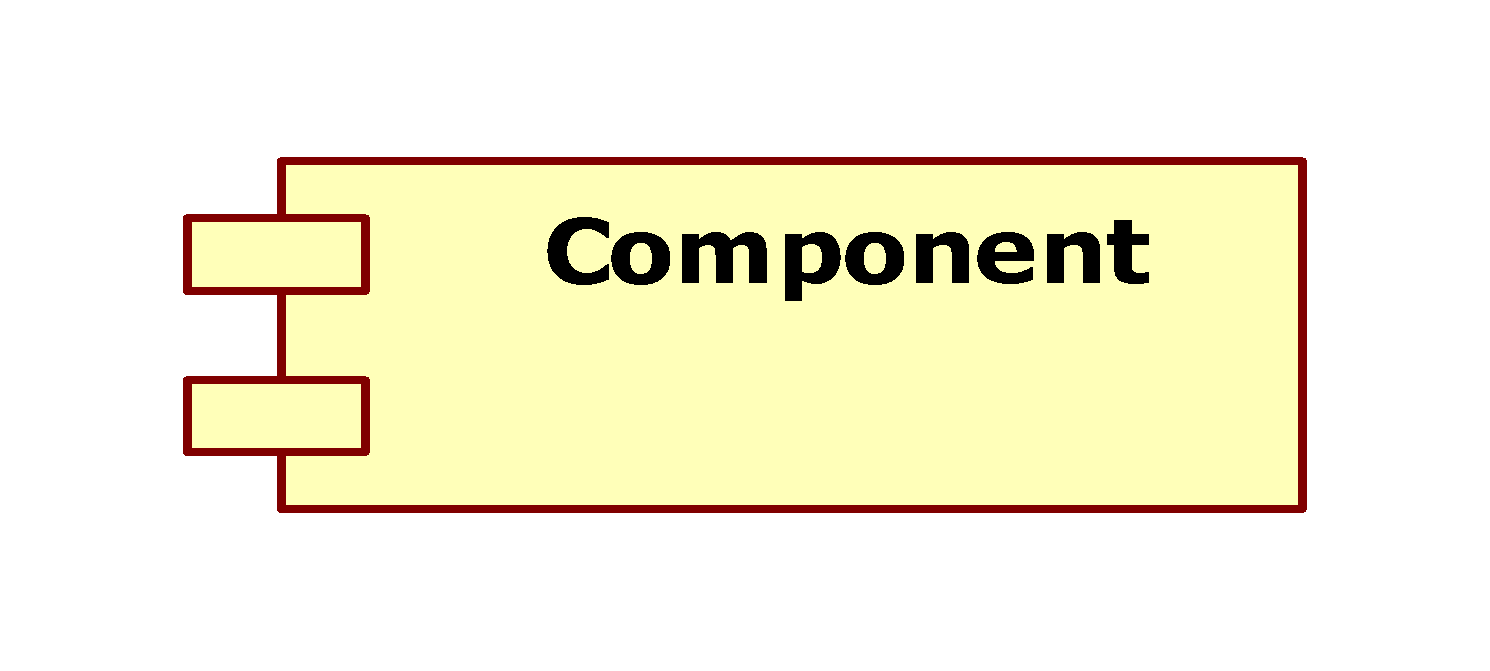
|  |
| --- |
| Interface |
|  |

**Collaboration:** Collaboration defines interaction between elements.

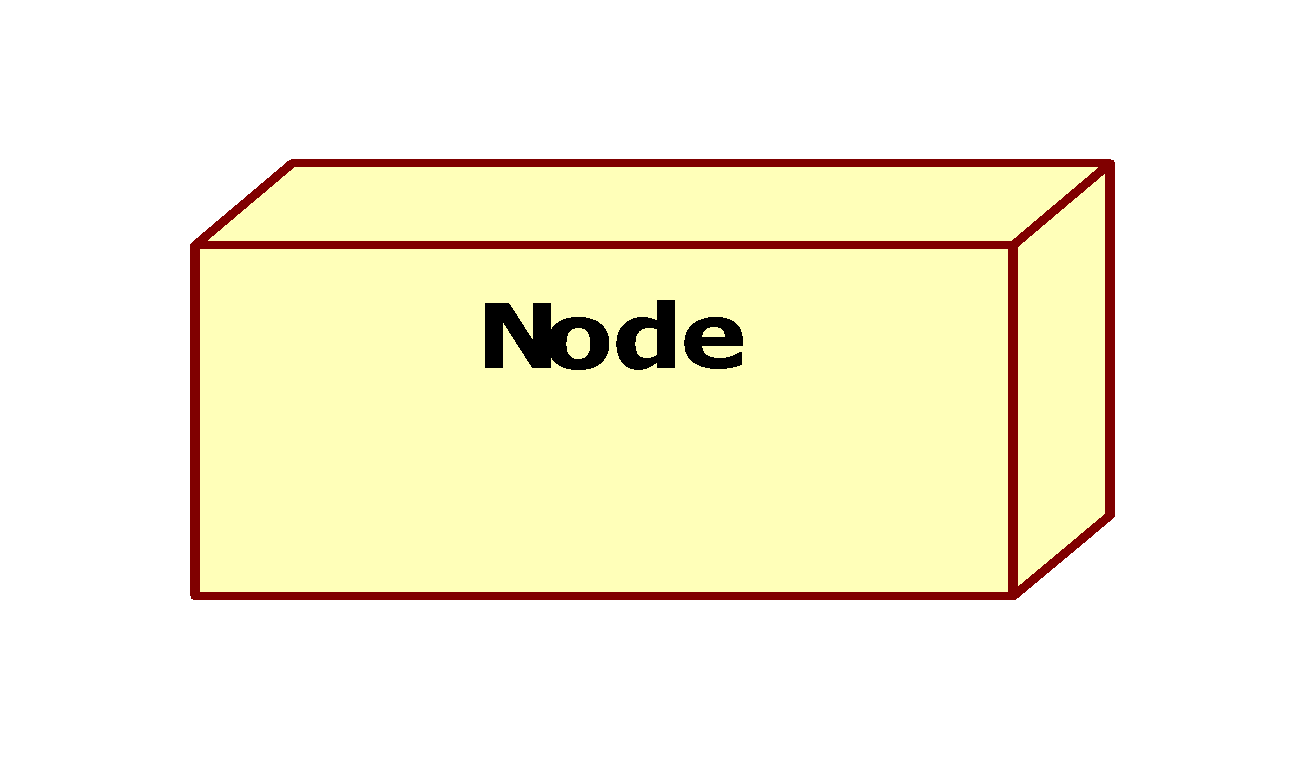
|  |
| --- |
|  |

**Use case:** Use case represents a set of actions performed by a system for a goal.

**Component:** Component describes physical part of a system.



**Node:** A Node can be defined as a physical element that exists at run time.



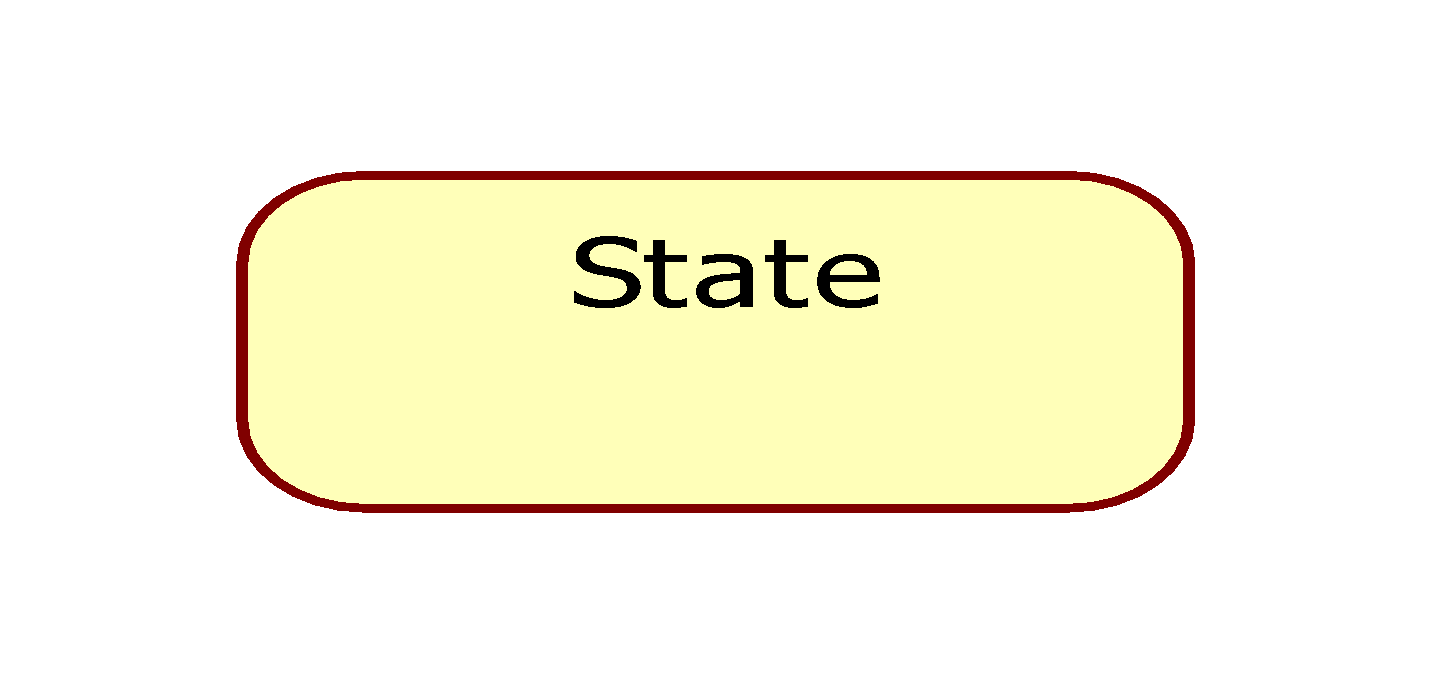
* 1. **BEHAVIOURAL THINGS:**

A Behavioural thing consists of the dynamic parts of UML, models. Following are the behavioural things:

**Interaction:** Interaction is defined as a behaviour that consists of a group of messages exchanged among elements to accomplish a specific task.

Message

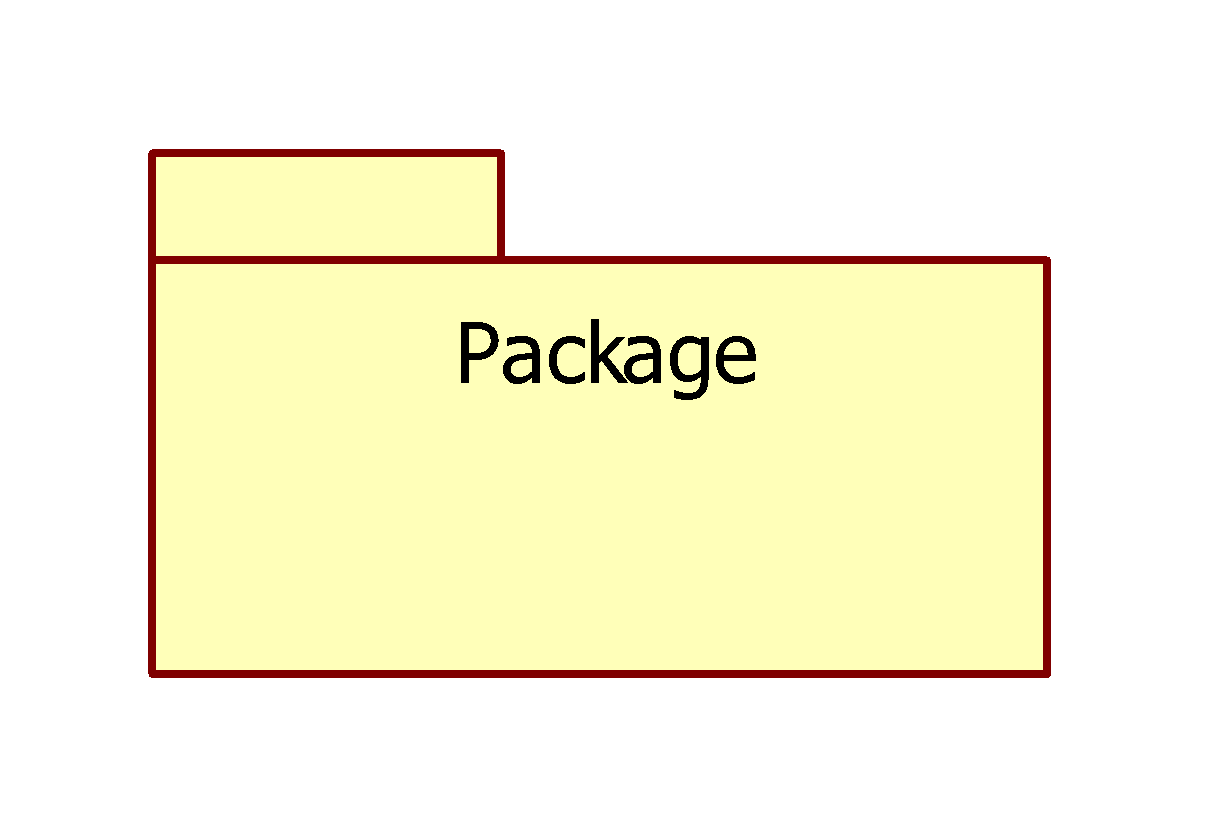
**State Machine:** State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



* 1. **GROUPING THINGS:**

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available;

**Package:** Package is the only one grouping thing available for gathering structural and behavioural things.



* 1. **ANNOTATIONAL THINGS**

Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note is the only one annotational thing available.

**Note:** A note is used to render comments, constraints etc of an UML element.

1. **RELATIONSHIPS**

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

**Dependency:**

Dependency is a relationship between two things change in one element also effects the other one.

**Association:**

Association is basically a set of links connects elements of an UML model. It also describes how many objects are taking part in that relationship.

**Generalisation:**

Generalisation can be defined as a relationship which connects a specialised element with a generalised element. It basically describes inheritance relationship in world of objects.

**Realisation:**

Realisation can be defined as a relationship in which two elements are connected. One element describes some relationship which is not implemented and the other one implements them. This relationship exists in case of inheritance.

1. **DIAGRAMS**

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

UML includes the following nine diagrams and there are two broad categories of diagrams and then are again divided into subcategories.

**3.1 STRUCTURAL DIAGRAMS:**

**Class Diagram:**

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interface, associations and collaboration. Class diagrams basically represent the object oriented view of a system which a static in nature. Active class is used in a class diagram to represent the concurrency of the system. Class diagrams represents the object orientation of a system. So it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

**Object diagram:**

Object diagrams can be described as an instance of class diagram. So these diagrams are more close to real life scenarios where we implement a system. Object diagrams are a set of objects and their relationships just like class diagrams and also represent the static view of the system. The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from practical perspective.

**Component Diagram:**

Component diagrams represent a set of components and their relationships. These components consists of classes, interface, interfaces or collaborations. So component diagrams represent the implementation view of a system. During design phase software artifacts (Classes, interfaces etc) of a system are arranged in different groups depending upon their relationship. Now these groups are known as components. Finally, component diagrams are used to visualise the implementation.

**Deployment Diagram:**

Deployment diagrams are a set of nodes and relationships. These nodes are physical elements where the components are deployed. Deployment diagrams are used for visualising deployment view of a system. This is generally used by the deployment team.

**3.2 BEHAVIOURAL DIAGRAMS:**

**Use case diagram:**

Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. So, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

**Sequence diagram:**

A sequence diagram is an interaction diagram. From the name it is clear that the diagram deals with some sequences, which are the sequence of message flowing from one object to another. Interaction among the components of a system is very important from implementation and execution perspective. So, sequence diagram is used to visualise the sequence of calls in a system to perform a specific functionalities.

**Collaboration Diagram:**

Collaboration diagram is another form of interaction diagram. It represents the structural organisation of a system and the messages sent/received. Structural organisation consists of objects and links.The purpose of collaboration diagram is similar to sequence diagram. But the specific purpose of collaboration diagram is to visualize the organisation of objects and their interactions.

**State chart diagram:**

Any real time system is expected to be reached by some kind of internal/external events. These events are responsibilities for state change of the diagram. State chart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc. Statechart diagram is used to visualise the reaction of a system by internal/external factors.

**Activity diagram:**

Activity diagram describes the flow of control in a system. So, it consists of activities and links. The flow can be sequential, concurrent or branched. Activities are nothing but the functions of as system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

**2.1.2 SOFTWARE ENGINEERING ACTIVITIES**

1. Requirement Elicitation

2. Analysis

3. System Design

4. Object Design

5. Implementation

6. Testing

**2.1. 3 RATIONAL ROSE**

The Rational Rose scripting language is an extended version of the summit basic script language. The Rational Rose extensions to basic scripting allow you to automatic Rational Rose specific functions, and in some cases to perform functions that are not available through the Rational Rose user interface.

The application’s method recommends the use of static and dynamic views of a logical model and a physical model to capture the in-process products of object oriented analysis and design. Using the notation, the application enables you to create and refine these views within an overall model representing your problem domain and software system.

This overall model contains classes use cases, objects, packages, operation, component packages, components, processors, devices and the relationship between them. Each of these model elements processes model properties that identify and characterize them. The notation provides graphical icons to represent each kind of model element and relationship.

A model also contains diagrams and specifications, which provide a means of visualizing and manipulating the model’s element and their model properties. Since diagrams are used to illustrate multiple views of a model, icons representing the model element can appear in none, one, several of a model’s diagrams. The application therefore enables you to control which element, relationship, and property icons appear on each diagram, using facilitate provided by its application window, and each specification in a specification window.

**2.2 USE CASE DIAGRAM**

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

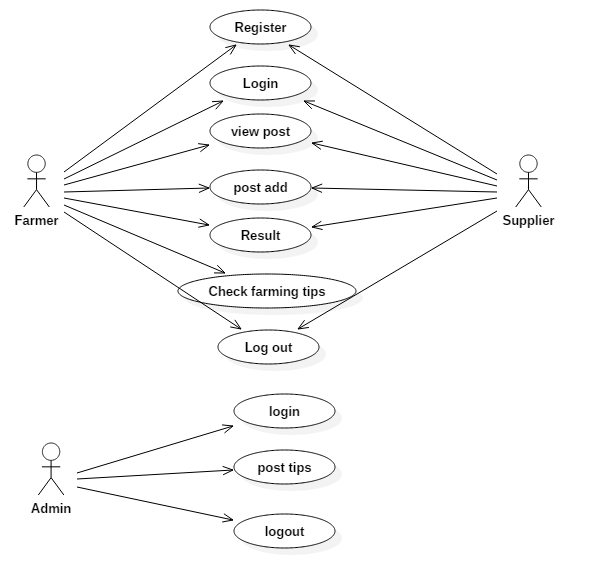


Fig 2.2: Use case diagram

**2.3 CLASS DIAGRAM**

A class diagram shows a set of classes, interfaces, collaborations and their relationships. The diagrams are most common diagram founding modelling object-oriented systems. Class diagram is a collaboration of attributes, operations, objects, classes and relationships among the classes. It represents overall structure of project. It is the main diagram for overall project.

Class diagram represents class name, attributes and method. It shares the same attributes, operations, relationships and semantics.

**Class:** It is description of a set of objects.

**Attributes:** It represents the property of a class.

**Method:** Performing operations on date.

**CLASS DIAGRAM REPRESENTATION:**

|  |
| --- |
| Class name |
| Attributes |
| Operations |
| Responsibility of class |

**ACCESS ADORNMENT:**

**Public:**

Public access means that the members of a class are accessible to all clients. This is the default access. You can also click on the public button in the access section of the relationship specification.

**Protected:**

Protected access means that the members of a class are accessible only to subclasses, friends or to the class itself. To apply a protected access adornment to a relationship, click on the protected button in the access section of the applicable relationship specification.

**Private:**

Private access means that the members of a class are accessible only to the class itself or to its friends. To apply a private access adornment to a relationship click on the private button in the access section of the relationship specification.

**Multiplicity**:

The cardinality field specifies the number of expected instances of the class. You can set a specific cardinality value for the client class, supplier class, or both.

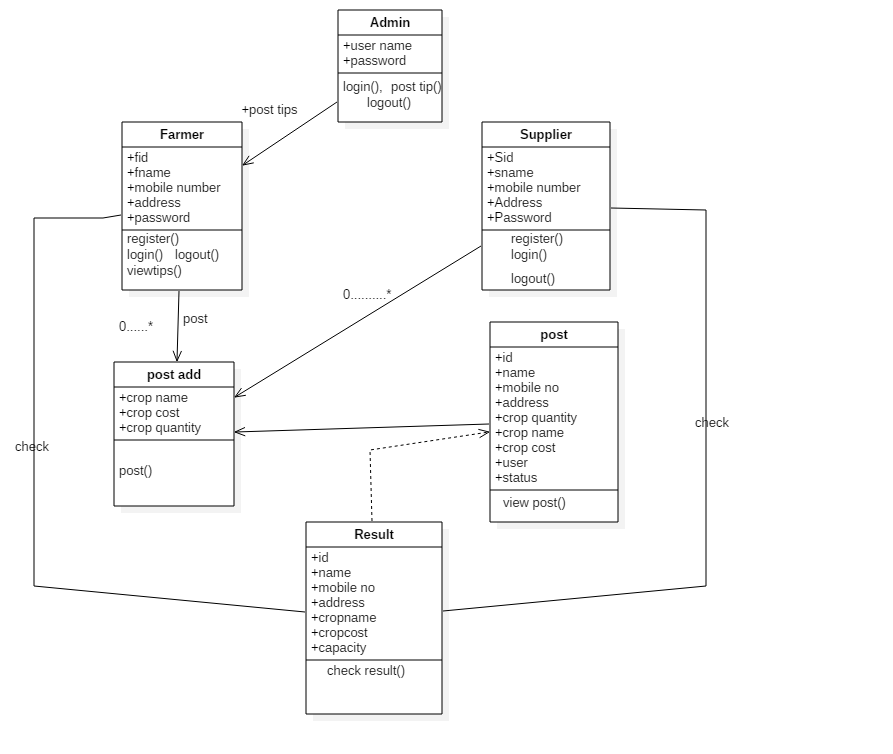
****

Fig 2.3: Class diagram

**2.4 SEQUENCE DIAGRAM**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence Chart. Sequence diagram are sometimes called event diagrams, event scenarios, and timing diagrams.

Sequence diagram contains:

* Object
* Swim lance/life line
* Message passing
* Object message
* Return message
* Self message

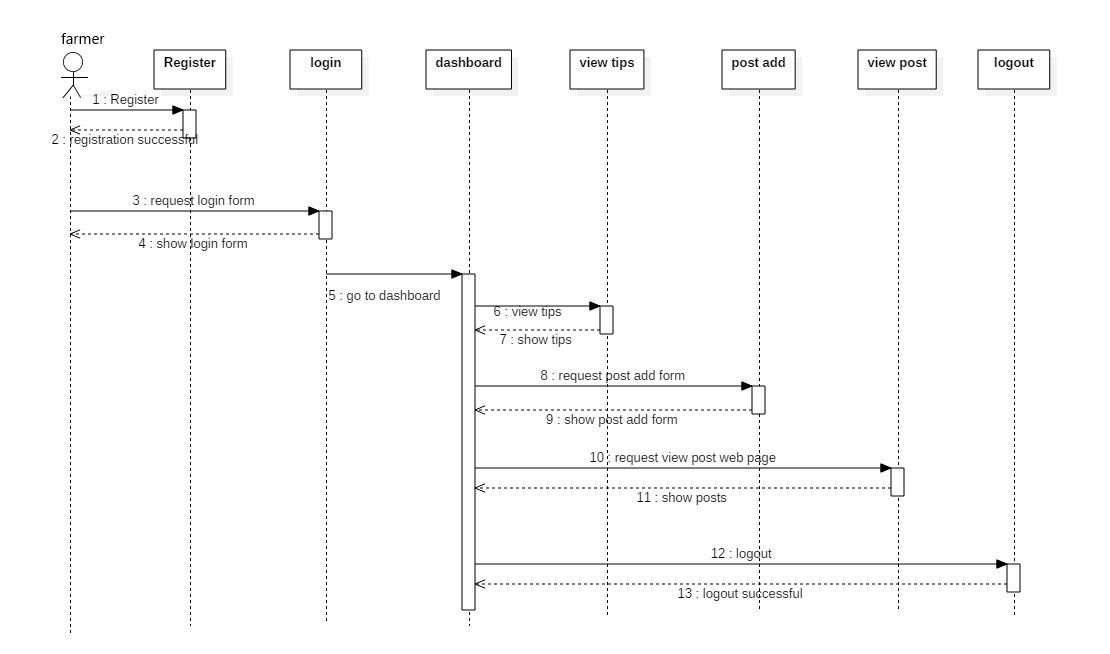


Fig 2.4.1: Sequence diagram for farmer

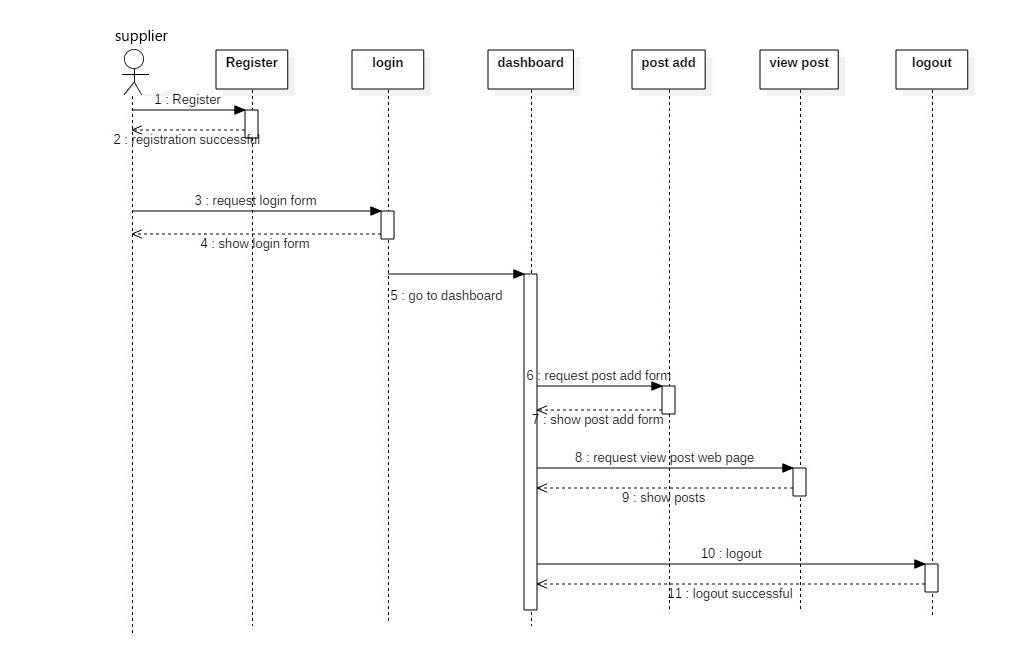
****

Fig 2.4.2: Sequence diagram for supplier

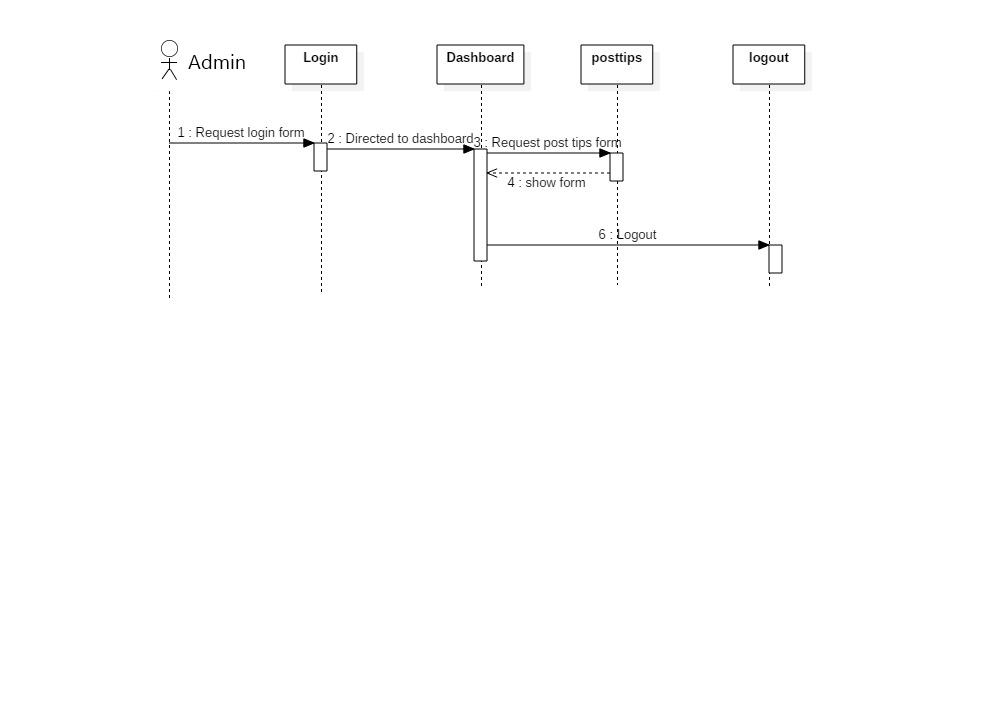
****

Fig 2.4.3: Sequence diagram for Admin

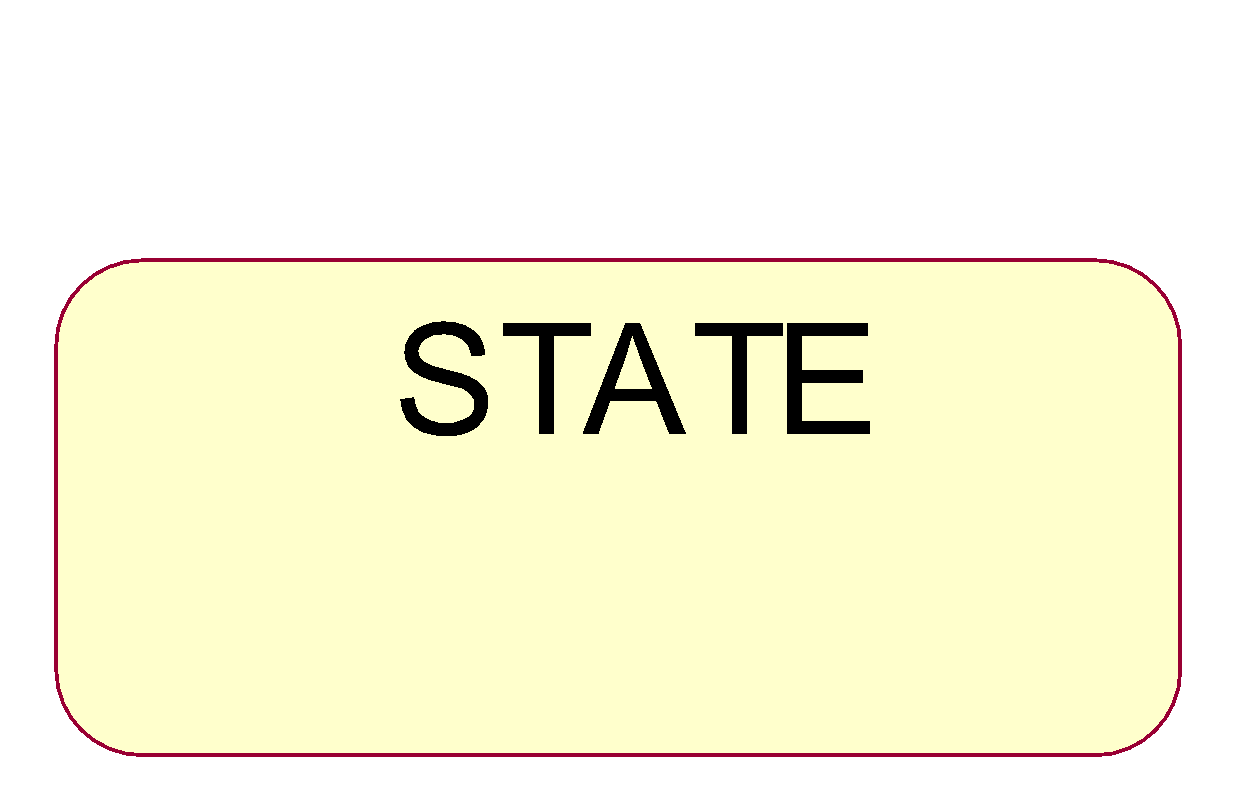
**2.6 STATECHART DIAGRAM**

**2.6.1 INTRODUCTION TO STATECHART DIAGRAM**

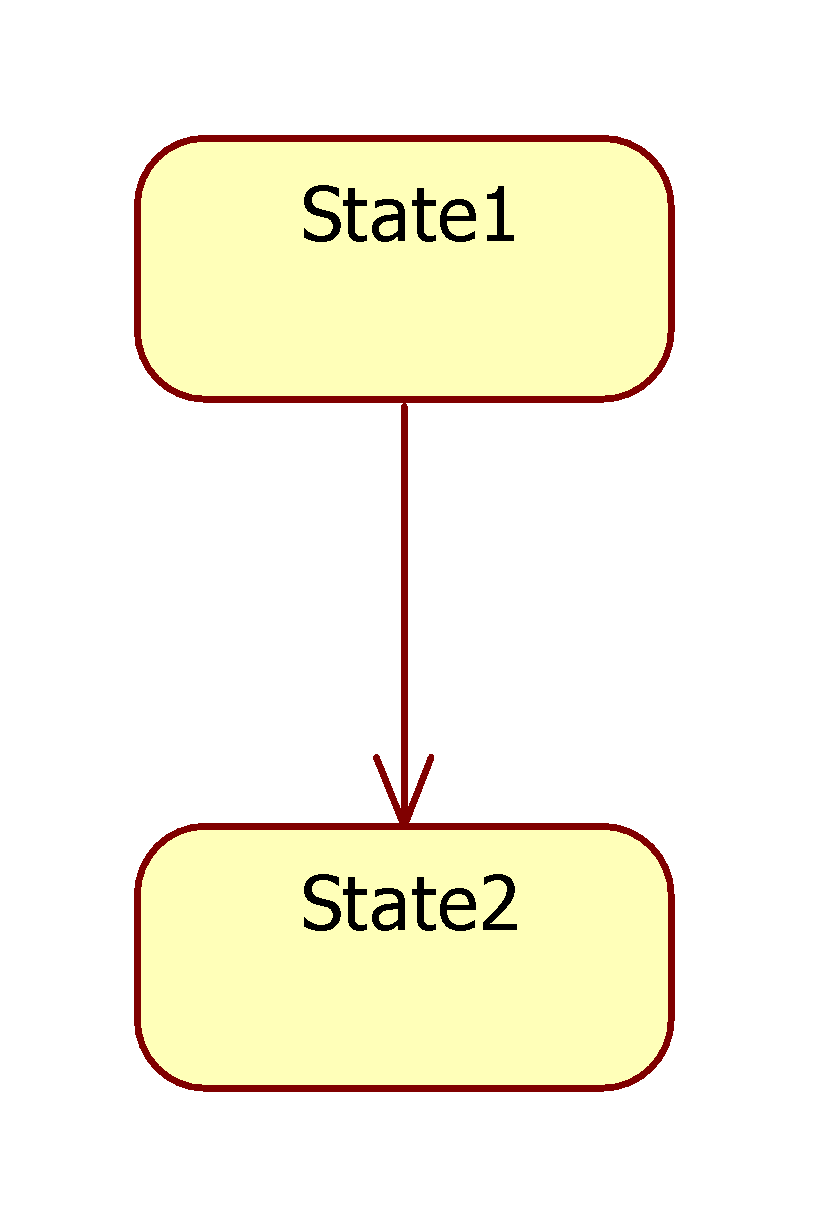
A state chart diagram is also called “state transition diagram”. It is used to identify the listed objects. It is going to represent how one object is going to change the states to complete a single use case. A state diagram shows a state machine, consisting of states, transitions, events, activities. A state diagram shows the dynamic view of an object. They are especially important in modelling the behaviour of an interface and class or collaboration and emphasize the event-ordered behaviour of an object, which is especially useful in modelling reactive systems.

State transition diagram provide a way to model the various states in which the class diagrams show a static picture of the classes. Elements in state chart diagram are

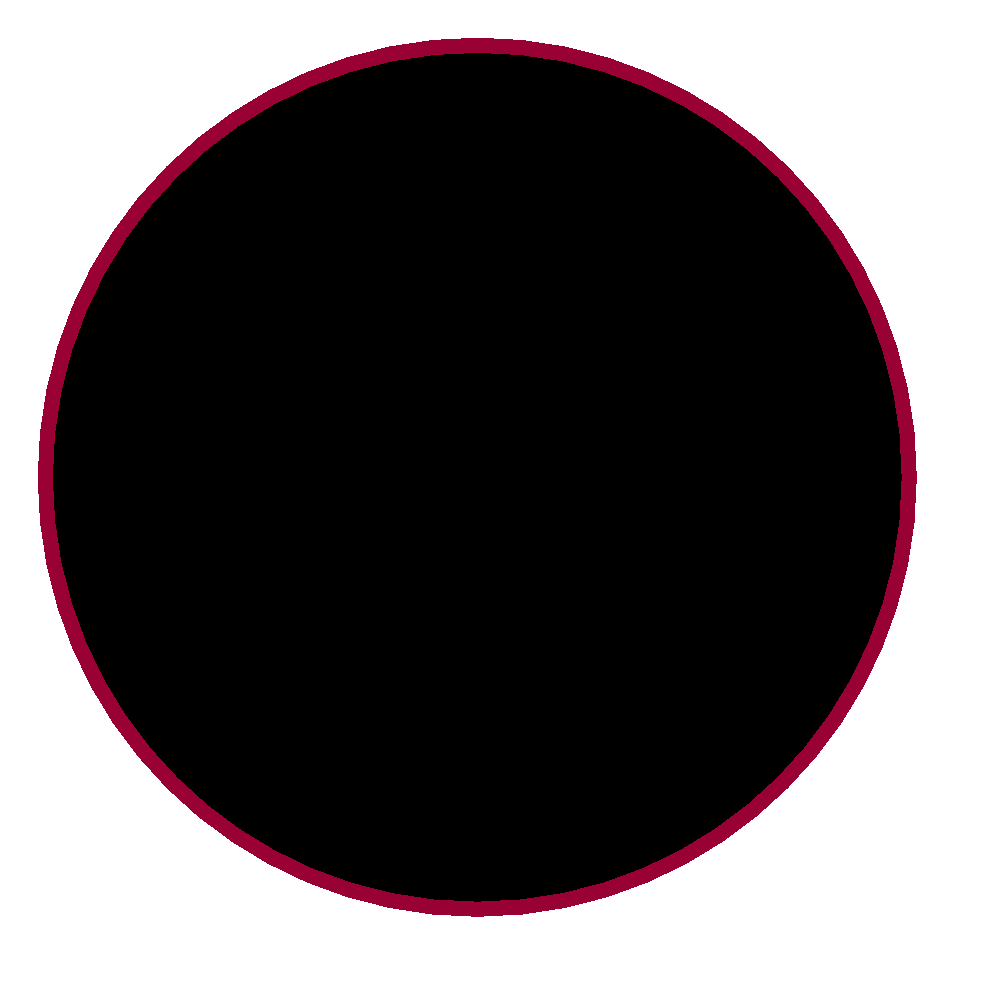
**State:** Means the range of values for all properties of an object. It is represented by



**Transition:** A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relation between two states, two activities or between an activity and a state. It is represented by



**Start state:** The starting state is represented by the symbol



**End state:** The ending state is represented by the symbol

* State chart diagram is used to model the various states in which an object can exist.
* While a class diagram shows the static picture of classes and their relationships.
* A state chart diagram shows the behaviour of an object.

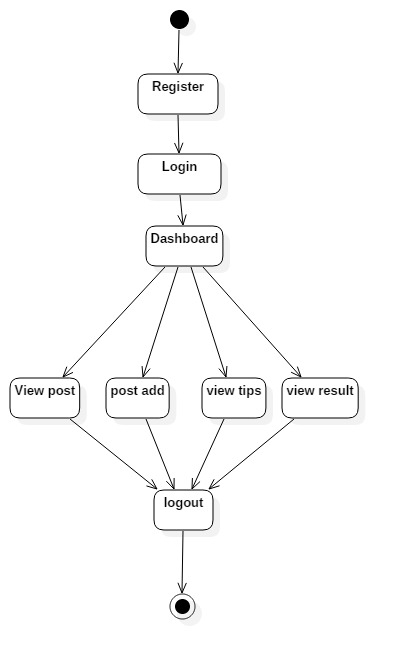


Fig 2.5: State chart diagram of farmer

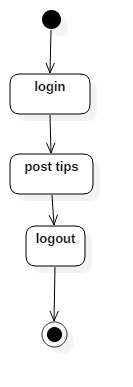
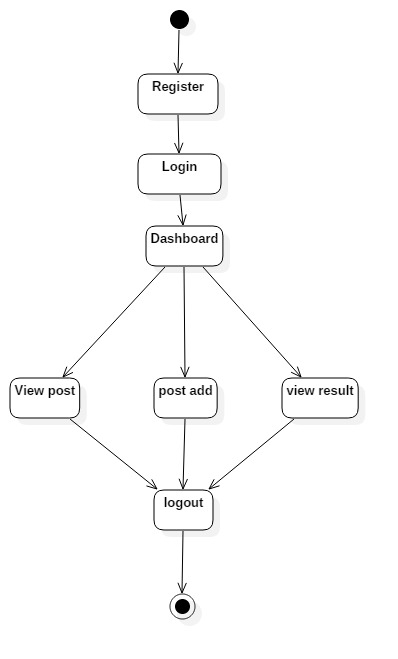


Fig 2.5.1: State chart diagram of suppliers Fig 2.5.2: State chart diagram of admin

**2.7 ACTIVITY DIAGRAM:**

Activity diagrams are graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagram can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

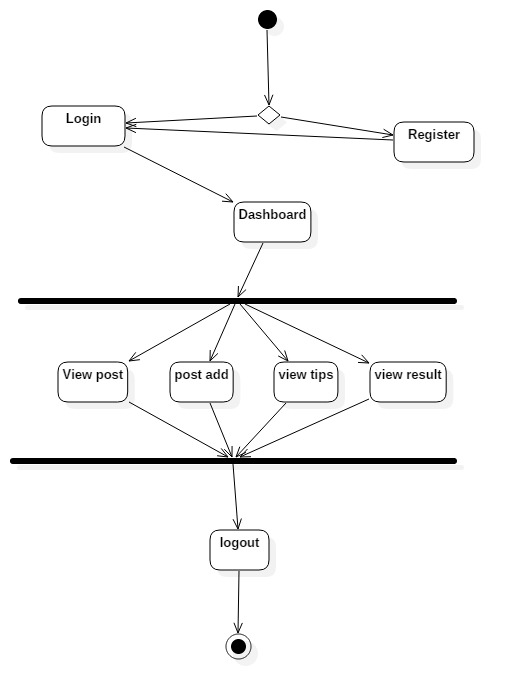
****

Fig 2.6: Activity diagram for farmer

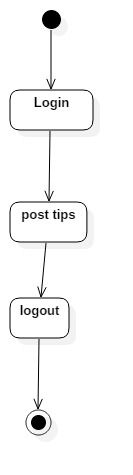
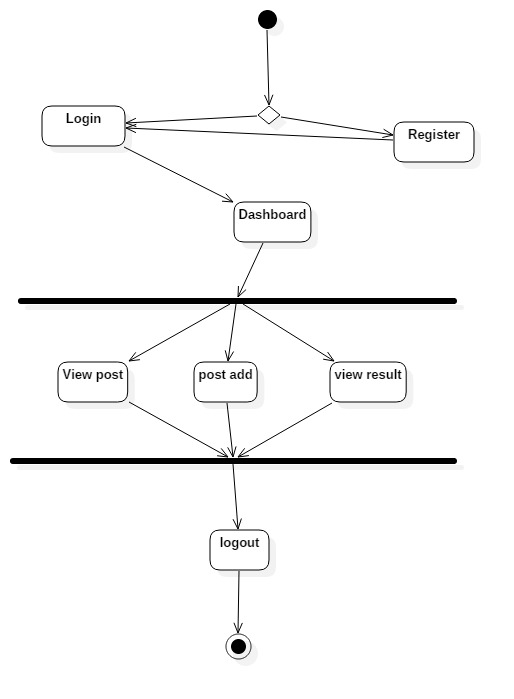


Fig 2.6.1: Activity diagram for admin

Fig 2.6.2: Activity diagram for supplier

**2.5 COLLABORATION DIAGRAM**

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time based sequence.Collaboration diagrams show objects, their links and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

The create collaboration diagram command creates a collaboration diagram from information contained in the sequence diagram. The create sequence diagram command creates sequence diagram from information contained in the interactions collaboration diagram. The go to sequence diagram and go to collaboration diagram commands traverse between an interaction’s two representations.Collaboration diagrams contain icons representing objects. You can create one or more collaboration diagrams to depict interactions for each logical package in your model. Such collaboration diagrams are themselves contained by the logical package enclosing the objects they depict. An object specification enables you to display and modify the properties and relationships of an object. The information can also be displayed inside the icons representing objects in collaboration diagrams.

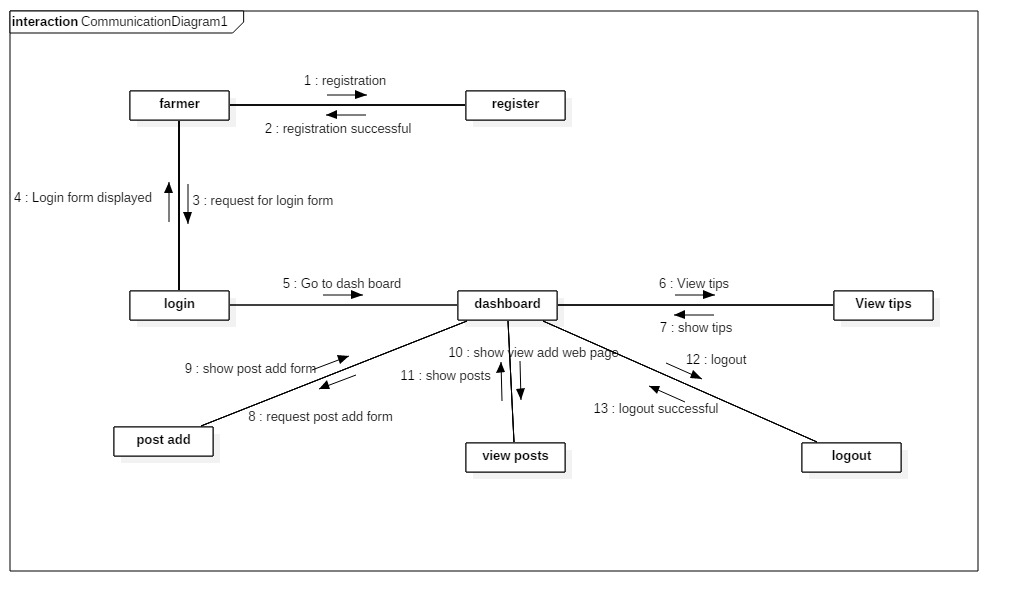
****

Fig 2.7.1: Collaboration diagram for farmer

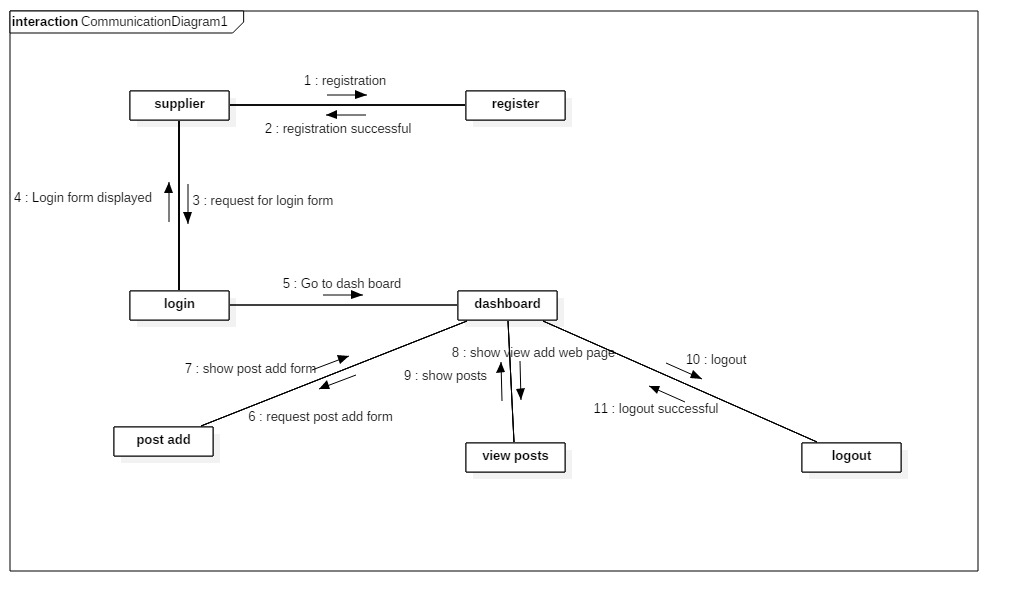
****

Fig 2.7.2: Collaboration diagram for upload supplier

**SYSTEM DESIGN DOCUMENT**

**3.1 DATABASE DESIGN**

**Schema**:

farmer TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Field name** | **Data type** | **Description** |
| 1. | Farmer Id | Int(20) | ID of the farmer |
| 2. | Farmer name | Varchar(30) | Name of the farmer |
| 3. | Address | Varchar(30) | place or city address of farmer |
| 4. | mobile number | Int(20) | phone number of farmer |
| 5. | Password | Varchar(10) | password |
|  |  |  |  |

Supplier TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Field name** | **Data type** | **Description** |
| 1. | supplier Id | Int(20) | ID of the supplier |
| 2. | supplier name | Varchar(30) | Name of the supplier |
| 3. | Address | Varchar(30) | place or city address of supplier |
| 4. | mobile number | Int(20) | phone number of supplier |
| 5. | Password | Varchar(10) | password |
|  |  |  |  |

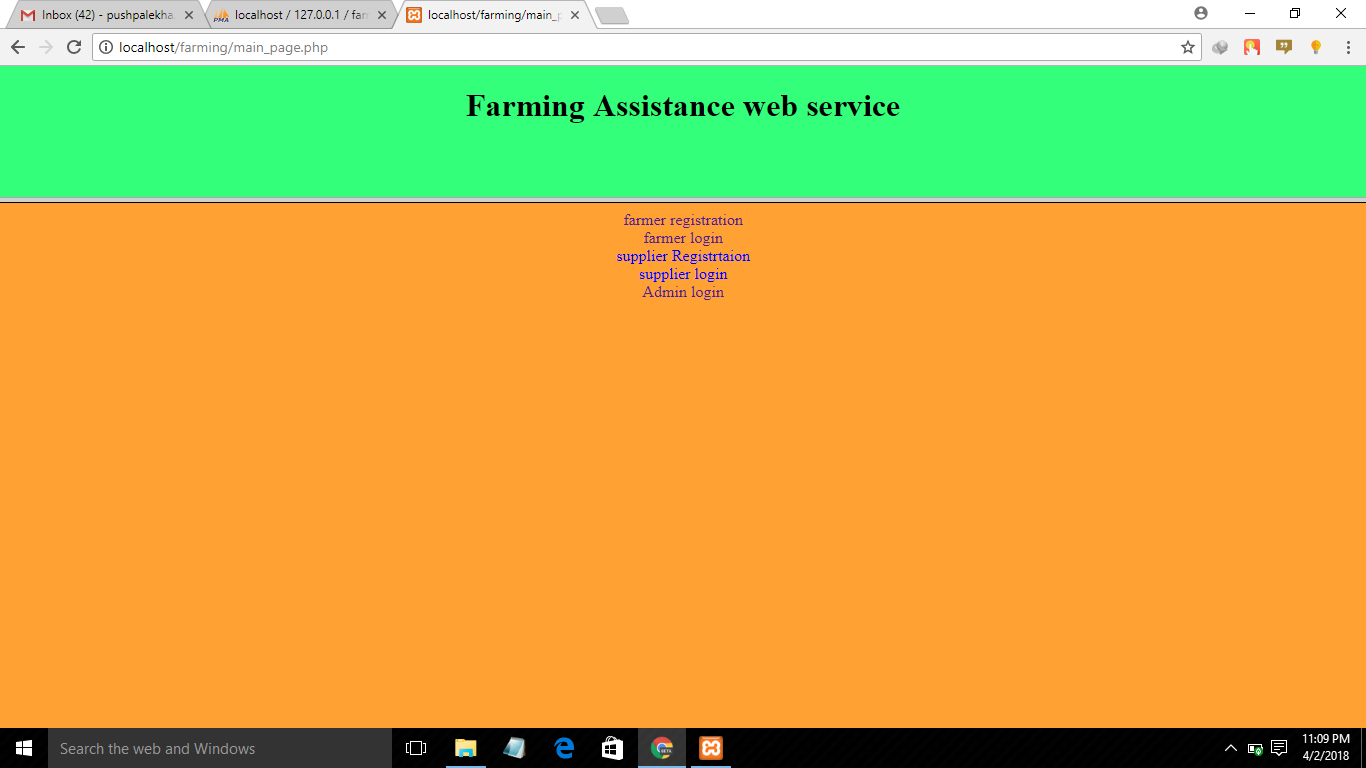
**post add table:**

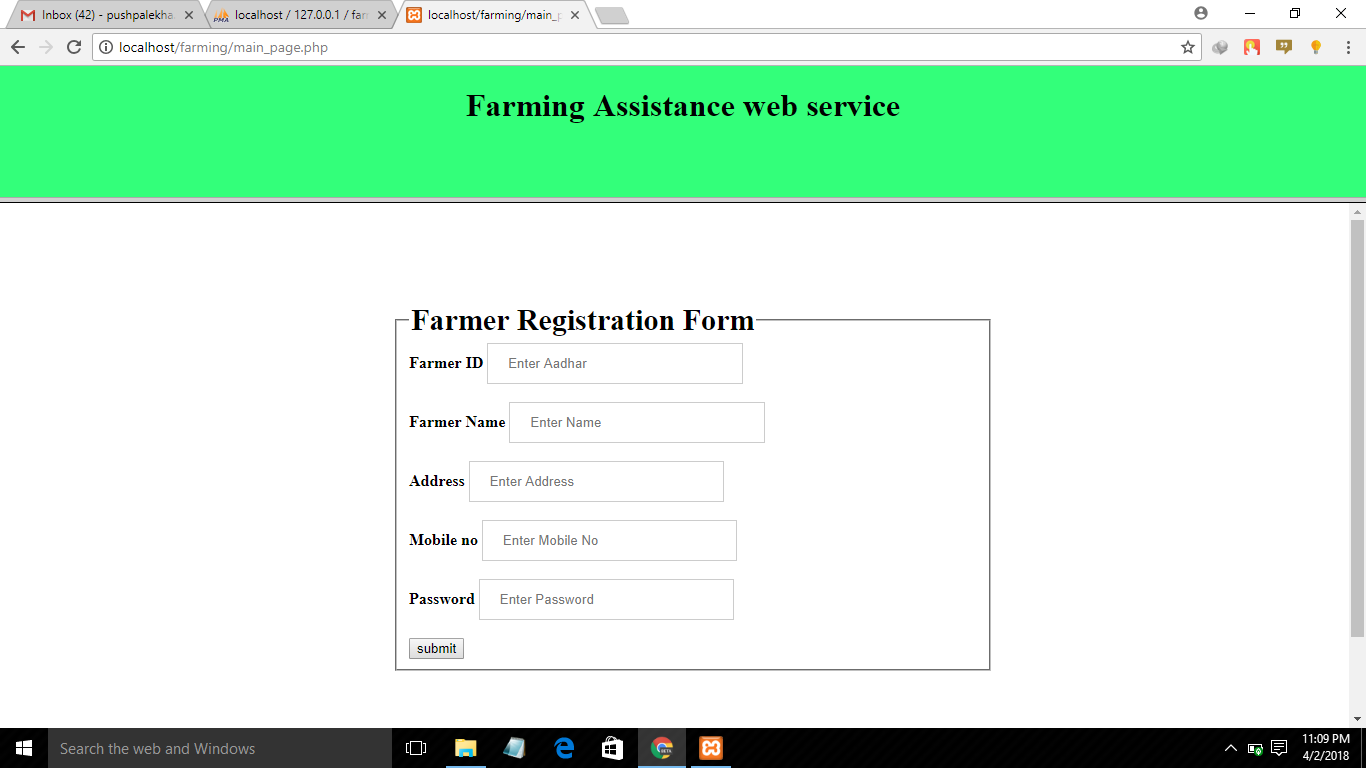
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Field Name** | **Type** | **Description** |
| 1. | crop name | Varchar(30) | name of the crop |
| 2. | crop quantity | int (10) | quantity required |
| 3. | crop cost | Int(10) | cost for that crop |

**CHAPTER-4**

**IMPLEMENTATION**

* 1. **SCREEN SHOTS**

****

**fig: screenshot for home page**  ****

**fig: screenshot for registration page**

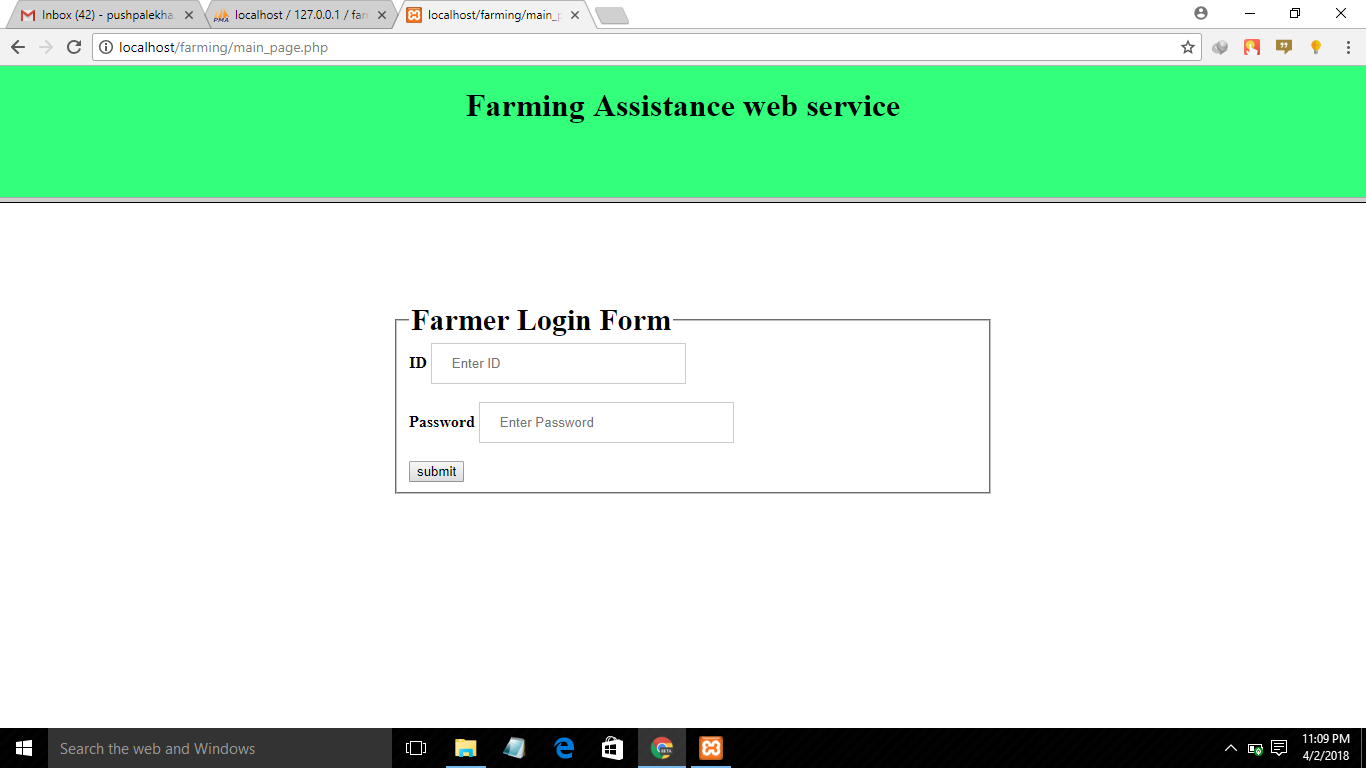
****

Fig: Screen shot for farmer login



Fig: Screen shot for admin login

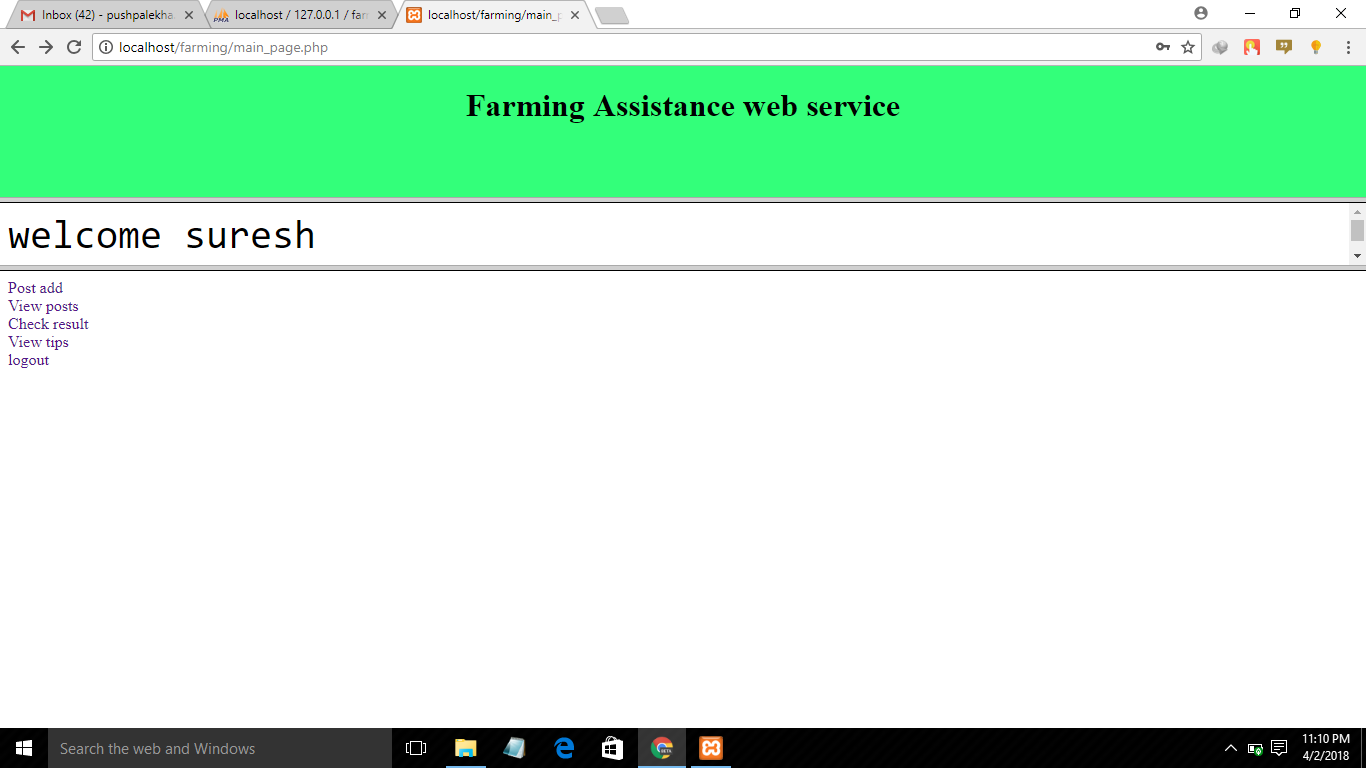


Fig: Screen shot for user dashboard

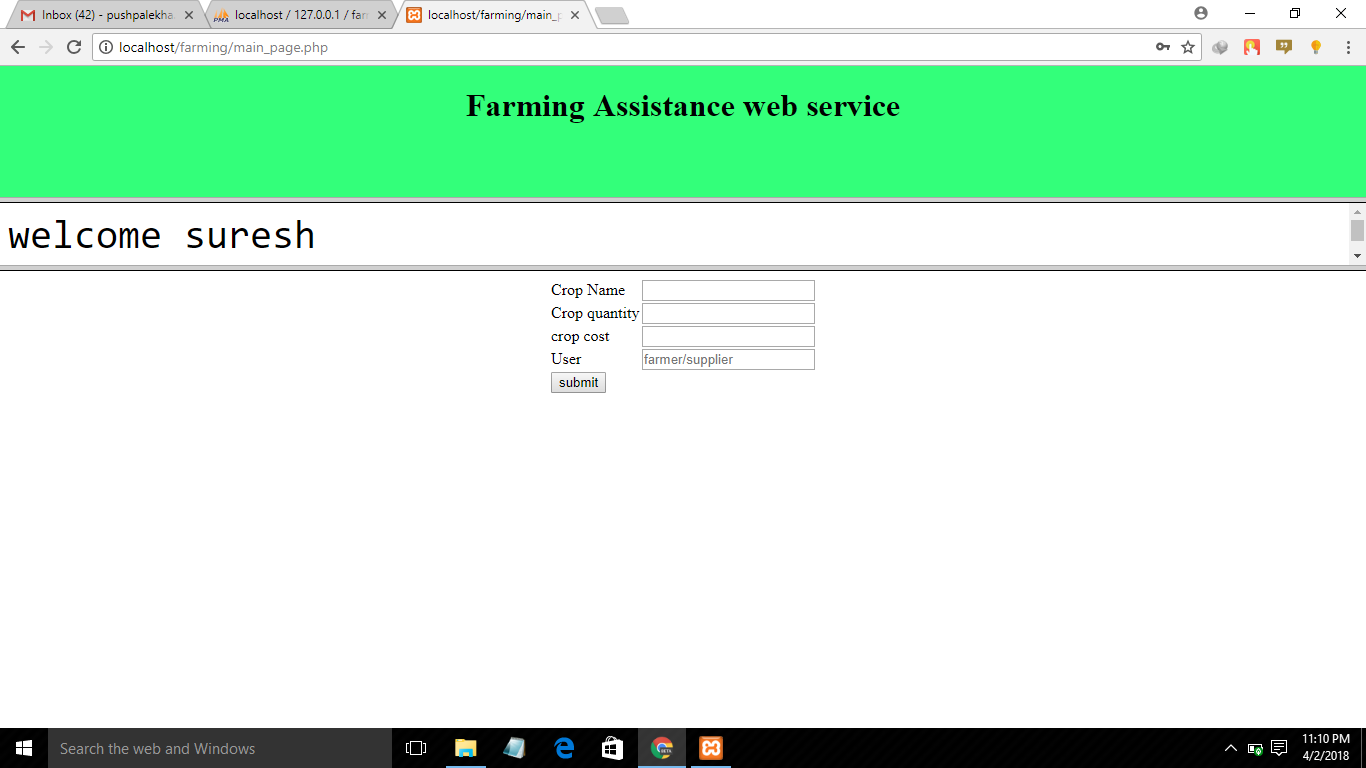


Fig: Screen shot for post add

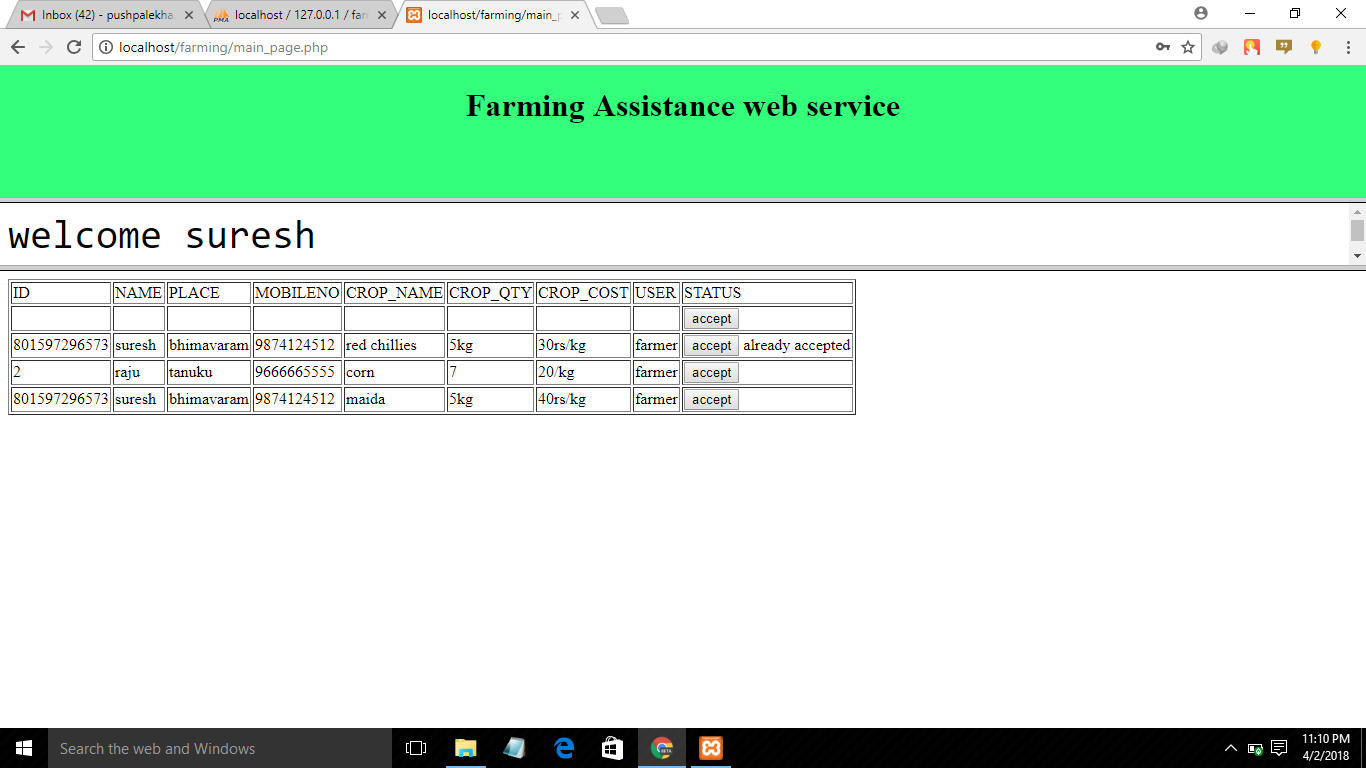


Fig: Screen shot for view add

**CHAPTER-5**

**TESTING**

**5.1 INTRODUCTION**

Testing is the process to find any deviation from the expected working of the system. If there is no deviation from the expected behaviour of the system then the project is successful otherwise failure. Testing can’t be done in a full-fledged manner because of the time and budget constraints.

**Test case**: In general a test case is a set of test data and test programs and their expected results. A test case 1 software engineering normally consists of a unique identifier, requirements references from a design specification, preconditions, events, a series of steps to follow, input, output and it validates one more system requirements and generates a pass or fail.

**5.2 BLACK BOX AND WHITE BOX TESTING**

1. Black box testing is tests the inputs and the corresponding outputs.

2. White box testing is tests the code where exactly the fault occurs can be identified by this means of testing.

**5.3 UNIT TESTING**

Unit testing is method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Unit tests are created by programmers or occasionally by white box testers.

**5.4 INTEGRATION TESTING**

Integration testing is the activity of finding faults when testing the individual components together. Structural testing is the culmination of integration testing involving all the components of the system.

**5.5 SYSTEM TESTING**

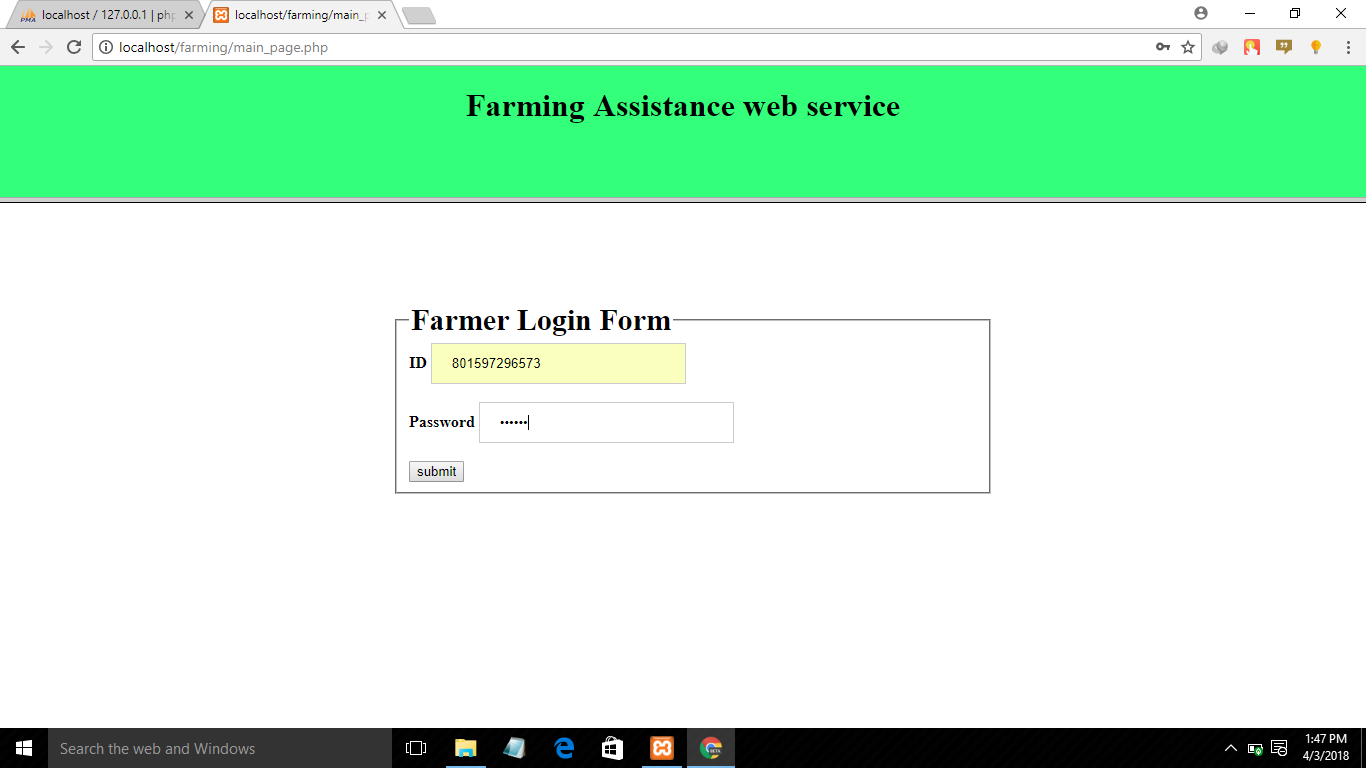
System testing tests all the components together, seen as a single system to identify faults with respect to scenarios from the problem statements and the requirements and design goals identify in the analysis and system design.

**TESTING ON LOGIN PAGE:**

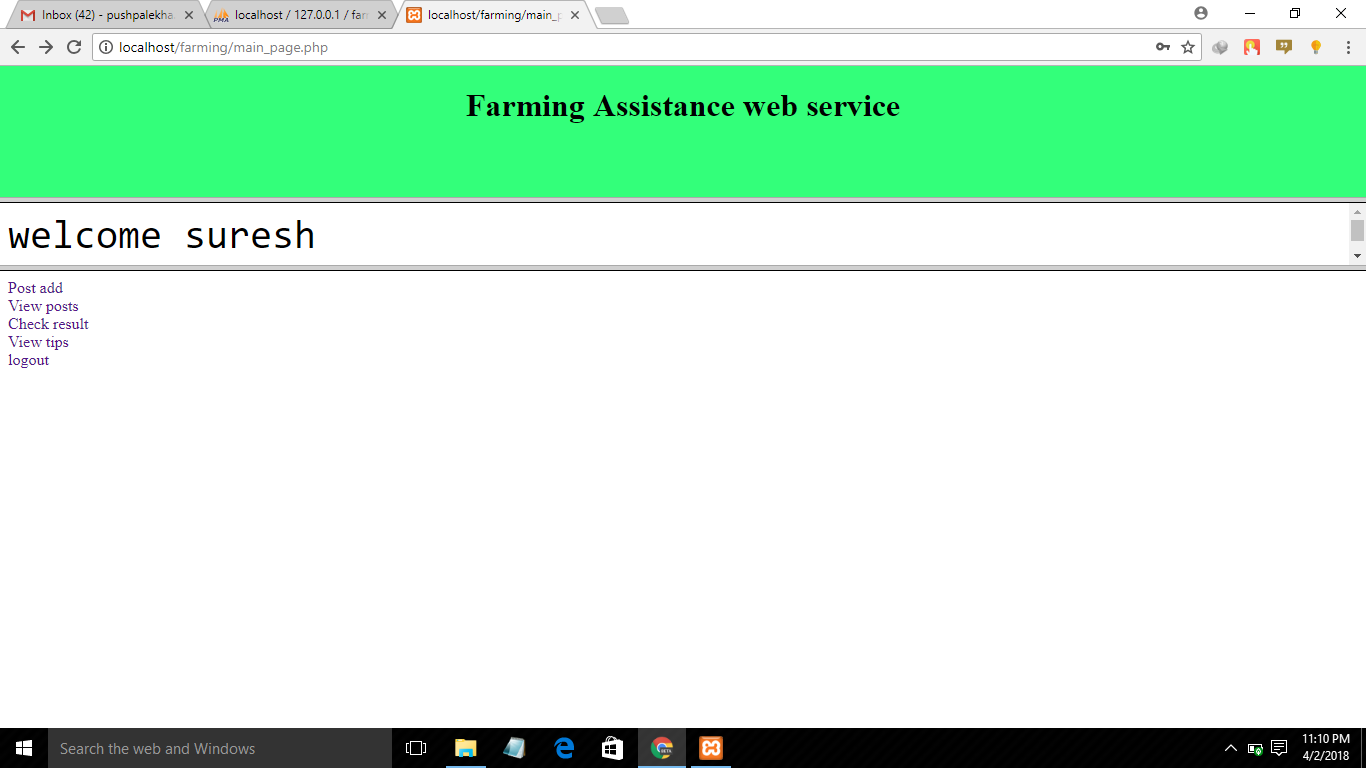
Validating login of the farmer:

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST CONDITION** | **INPUT SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
| Enter farmer Id and password | 1. If correct specifications are entered 2. If wrong specifications are entered | Direct to next page  Display enter wrong credentials message | Pass  Fail |

Screen shot for login:



registration successful and dashboard



**CHAPTER-6**

**CONCLUSION**

The main purpose of this farming web assistance application is to remove third party brokers and provide better profits for farmers and suppliers

**FUTURE ENHANCEMENTS**

* The aim of the project is to help the farmers for their betterment and obtain greater profit by having direct farmer to supplier and farmer to farmer communications.
* This project aims to create awareness, boost business communication and transparency in the system by having direct farmer to supplier and farmer to farmer communication.
* Also it does notify the farmer regarding any advertisement or offer published by the govt. or dealer.

**CHAPTER-7**

**REFERENCES**

1. Object-oriented software engineering-Berned Bruegge, Allen H. Dutoit.
2. Web technology and design-C. Xavier.
3. [www.w3.schools.com](http://www.w3.schools.com)
4. www.projects-forum.com
5. www.uml-diagrams.com

**CHAPER-8**

**APPENDIX**

* 1. **SAMPLE CODE**

**Source code for former login**

<?php session\_start();

?>

<!DOCTYPE html>

<html>

<style>

form{

margin-top: 50px;

margin-bottom: 50px;

margin-right: 50px;

margin-left: 200px;

width:600px;

padding:50px 100px 100px 185px;

}

legend{

font-size: 30px;

}

/\* Full-width input fields \*/

input[type=text], input[type=password] {

width: 45%;

padding: 12px 20px;

margin: 0 0;

display: inline-block;

border: 1px solid #ccc;

box-sizing: border-box;

}

/\* Set a style for all buttons \*/

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 40%;

}

/\* Extra styles for the cancel button \*/

.cancelbtn {

padding: 14px 20px;

background-color: #f44336;

}

/\* Float cancel and signup buttons and add an equal width \*/

.cancelbtn,.signupbtn {

align:center;

width: 30%;

}

/\* Add padding to container elements \*/

.container {

align="center;

}

/\* Clear floats \*/

.clearfix::after {

content: "";

clear: both;

display: table;

}

/\* Change styles for cancel button and signup button on extra small screens \*/

@media screen and (max-width: 300px) {

.signupbtn {

width: 30%;

align:center;

}

}

</style>

<body>

<form action="farmer\_login\_php.php" method="post">

<fieldset>

<legend><b>Farmer Login Form</b></legend>

<div class="container">

<label><b>ID</b></label>

<input type="text" placeholder="Enter ID" name="username" required><br><br>

<label><b>Password</b></label>

<input type="password" placeholder="Enter Password" name="password" required><br><br>

<input type="submit" value="submit" name="submit">

</div>

</fieldset>

</form>

</body>

</html>

**Source code for former login php**

<?php session\_start()

?>

<?php

if(isset($\_POST["submit"])){

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "farming";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$myusername = mysqli\_real\_escape\_string($conn,$\_POST['username']);

$mypassword = mysqli\_real\_escape\_string($conn,$\_POST['password']);

$sql = "SELECT \* FROM farmer WHERE farmerid = '$myusername' and password = '$mypassword'";

$result = mysqli\_query($conn,$sql);

$row = $result->fetch\_assoc();

$count = mysqli\_num\_rows($result);

if($count == 1) {

$\_SESSION['farmerid'] = $myusername;

$\_SESSION['farmername'] = $row[farmername];

$\_SESSION['address'] = $row[address];

$\_SESSION['moblieno'] = $row[mobileno];

header("Location:wel\_farm.php");

}

else {

header("Location: main\_page.html");

$error = "Your Login Name or Password is invalid";

echo $error;

}

$conn->close();

}

?>

**Source code for admin login:**

<?php session\_start();

?>

<!DOCTYPE html>

<html>

<style>

form{

margin-top: 50px;

margin-bottom: 50px;

margin-right: 50px;

margin-left: 200px;

width:600px;

padding:50px 100px 100px 185px;

}

legend{

font-size: 30px;

}

/\* Full-width input fields \*/

input[type=text], input[type=password] {

width: 45%;

padding: 12px 20px;

margin: 0 0;

display: inline-block;

border: 1px solid #ccc;

box-sizing: border-box;

}

/\* Set a style for all buttons \*/

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 40%;

}

/\* Extra styles for the cancel button \*/

.cancelbtn {

padding: 14px 20px;

background-color: #f44336;

}

/\* Float cancel and signup buttons and add an equal width \*/

.cancelbtn,.signupbtn {

align:center;

width: 30%;

}

/\* Add padding to container elements \*/

.container {

align="center;

}

/\* Clear floats \*/

.clearfix::after {

content: "";

clear: both;

display: table;

}

/\* Change styles for cancel button and signup button on extra small screens \*/

@media screen and (max-width: 300px) {

.signupbtn {

width: 30%;

align:center;

}

}

</style>

<body>

<form action="admin\_login\_php.php" method="post">

<fieldset>

<legend><b>Admin Login Form</b></legend>

<div class="container">

<label><b>username</b></label>

<input type="text" placeholder="Enter username" name="username" required><br><br>

<label><b>Password</b></label>

<input type="password" placeholder="Enter Password" name="password" required><br><br>

<input type="submit" value="submit" name="submit">

</div>

</fieldset>

</form>

</body>

</html>