**FARMER FRIENDLY SERVICES**

***A Mini-Project -I Report submitted in***

***partial fulfilment of the requirements***

***for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

***In***

## COMPUTER SCIENCE & ENGINEERING

|  |  |
| --- | --- |
|  | ***By*** |
| 1. K. GAYATHRI (19B01A0568) | 4. K. LASYA SREE (20B05A0507) |
| 2. P. SOWMYA SRI (19B01A05C9)    3. N. LAVANYA (19B01A05C7) | 5. P. TEJA ARAVINDA VEERA SUPRAJA  (19B01A05C8) |

***Under the esteemed guidance of***

**Dr. V. PURUSHOTHAMA RAJU** Ph.D.

**Professor**



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)

**(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)**

**BHIMAVARAM – 534 202 2020 – 2021**

# SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)

**(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)**

**BHIMAVARAM – 534 202**

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



## CERTIFICATE

*This is to certify that the Mini project- I entitled “****FARMER FRIENDLY SERVICES****”, is being submitted by the members* ***K. GAYATHRI, P. SOWMYA SRI, N. LAVANYA, K. LASYA SREE, P. TEJA ARAVINDA VEERA SUPRAJA*** *bearing the* ***Regd. No. 19B01A0568, 19B01A05C9, 19B01A05C7, 20B05A0507, 19B01A05C8*** *in partial fulfilment of the requirements for the award of the degree of “****Bachelor of Technology*** *in* ***Computer Science & Engineering****” is a record of Bonafede work carried out by them under my guidance and supervision during the academic year 2020–2021 and it has been found worthy of acceptance according to the requirements of the university.*

**Internal Guide Head of the Department**

### ACKNOWLEDGEMENTS

Behind every achievement there lies an unfathomable sea of gratitude to those who activated it, without whom it would never come into existence. To them we lay a word of gratitude imprinted within us.

We express our heartfelt thanks to our Chairman **Sri K.V. VISHNU RAJU**  for providing us with all the facilities necessary to carry out this project successfully.

We also express our heartfelt thanks to our Principal **Dr. G. Srinivasa Rao** and Vice-Principal **Dr. P. Srinivasa Raju** for providing us the necessary facilities to carry out this project.

We also express our sincere thanks to our HOD **Dr. P. Kiran Sree** for providing us the necessary facilities to carry out this project.

We also express our sincere thanks to our guide **Dr. V. Purushothama Raju** Professor of Computer Science and Engineering department for his valuable suggestions and encouragement throughout the project.

We express our thanks to all teaching and non-teaching staff associates of the department for their good wishes and constructiveness.

K. GAYATHRI

P. SOWMYA SRI

N. LAVANYA

K. LASYA SREE

P. TEJA ARAVINDA VEERA SUPRAJA

### ABSTRACT

|  |
| --- |
| This is a web-based project which is useful for farmers and agricultural students. This is an open discussion portal providing solutions to small farmers and agricultural students. It also provides soil analysis for all regions and suggestions on which fertilizers to use where and how much? And which crop, herb, or vegetable to be grown where and in which season? It also helps to make decisions on market and best prices. Information about major crop markets and their current price for the crop will be published daily. NGOs are trying to spread messages to make agriculture more eco-friendly through this site. Training is requested by students, public.    Training provides information about crops, fertilizers, and market details that are requested. Online query handlings for all users. Queries can be posted by students, public through mails. Queries can be directed to a particular officer. Information pages should be dynamic so that agricultural officers and administrator can change it. |
|  |
|  |
|  |
|  |
|  |

**Contents**

**S. No. Topic Page No.**

1. Introduction …………………………………… 1

2. System Analysis …………………………………… 3

2.1 Existing System …………………………………… 3

2.2 Proposed System …………………………………… 3

2.3 Feasibility Study …………………………………… 4

3. System Requirements Specification …………………………………… 5

3.1 Software Requirements …………………………………… 6

3.2 Hardware Requirements …………………………………… 6

3.3 Functional Requirements ….………………………………

3.4 Non-Functional Requirements: …………………………………..

4. System Design …………………………………… 7

4.1 Introduction …………………………………… 7

4.2 UML Diagrams …………………………………… 8

4.4 ER Diagrams …………………………………… 16

5. System Implementation …………………………………… 20

5.1 Introduction ……………………………………

5.2 Project Modules ……………………………………

5.3 Screens ……………………………………

6. System Testing …………………………………… 25

6.1 Introduction …………………………………… 25

6.2 Testing Methods …………………………………… 25

7. Conclusion …………………………………… 27

8. Bibliography …………………………………… 28

9. Appendix …………………………………… 29

9.1 Introduction to JAVA …………………………………… 29

9.2 Introduction to HTML ……………………………………

9.3 Introduction to CSS ……………………………………

9.4 Introduction to MySQL ……………………………………

# 

# INTRODUCTION

**1. INTRODUCTION**

It is an open discussion portal used for agricultural students and farmers. Any public can use this system for knowing the information about various crops, and the usage of fertilizers to those crops and in which soil these crops give more yield and the climatic conditions for those crops. The portal provides soil analysis for various regions and suggestions based on the soil condition and climate. It explores questions such as: “which fertilizers to use where and in what quantity”, “which crop, vegetable, or herb should be grown where and in which season”, etc.

Training is requested by the students, public. These trainings are scheduled by agricultural officer. Information about major crop markets and their current price for the crop will be published daily.

Current prices of the markets are updated daily by the NGO. It gives information regarding all the states in India. Information regarding major crop markets and the prevalent best price for the crops are published on a regular basis.

**Problem statement:**

* Many farmers don’t have a clear idea about which fertilizer is best for the crop, vegetable, herb and what are the percentage of chemicals in the fertilizer, etc.
* This project is useful for farmers to obtain information regarding various crops, the fertilizers usage for these crops.
* The soil condition in which these crops yield more, and the suitable climatic and environmental condition for these crops.

**Objectives:**

* The main objective of the project is to provide information to the farmers.
* It provides better solutions for farmers in providing the information regarding soils and fertilizers.
* It helps the users in getting training regarding various technologies that can be used in farming.

# 

# SYSTEM ANALYSIS

# 

## 2. SYSTEM ANALYSIS

**2.1 Existing system:**

The existing system is very traditional as the data management is very complex. It is difficult to provide sufficient information regarding agricultural products, soils, fertilizers, market details, and climate in a reliable way. Complexity in managing the data related to agricultural products, soils, fertilizers, and market details.

* Lack of security.
* This system does not provide category wise classification of products.
* Inefficiency in querying details.

**2.2 Proposed system:**

The proposed system is built in view of database integration approach; it tries to automate the entire process in an effective, simple, and reliable way. The key features are outlined below:

* Reduces complexity in data management.
* Category-wise classification of information regarding agricultural products, soils, fertilizers, and market details.
* Different access levels for different users from security point of view.
* Rich user interface.
* Querying details is very efficient.
* User queries and answers are maintained.
* Economical, user friendly, and effective.

**2.3 FEASIBILITY STUDY**

Generally, the feasibility study is used for determining the resource cost, benefits and whether the proposed system is feasible with respect to the organization. The proposed system feasibility could be as follows. There are three types of feasibility which are equally important are:

* Technical feasibility
* Economic feasibility
* Behavioural feasibility

**Technical Feasibility**

Technical feasibility deals with the existing technology, software, and hardware requirements for the proposed system. The proposed system “Farmer friendly services” is planned to run on java and PHP. Thus, the project is considered technically feasible for the development. The work for the project can be done with current equipment, existing software technology and available personnel. Hence the proposed system is technically feasible.

**Economic Feasibility**

This method is most frequently used for evaluating the effectiveness of a PHP. It is also called as benefit analysis. In this project “farmer friendly services” is developed on current equipment, existing software technology. Since the required hardware and software for developing the system is already available in the organization, it does not cost must developing the proposed system.

**Behavioural Feasibility**

This project has been implemented by html, CSS and it satisfies all conditions and norms of the organization and the users. This proposed system “farmer friendly services” Application has much behavioural feasibility because users are provided with a better facility.

# 

# SYSTEM REQUIREMENTS SPECIFICATION

**3. SYSTEM REQUIREMENTS SPECIFICATION**

**Introduction:**

A Software Requirements Specification (SRS) is a document, which is used as a communication medium between the customers. A software requirement specification in its most basic form is a formal document used in communicating the software requirements between the customer and the developer.

An SRS document concentrates on WHAT needs to be done and carefully avoids the solution (how to do). It serves as a contract between the development team and the customer. The requirements at this stage are written using end user terminology. If necessary, later a formal requirement specification will be developed from it. SRS is a complete description of the behaviour of a system to be developed and may include a set of use-cases that describes the interactions the users will have with the software.

**Purpose of SRS**

SRS is a communication tool between Customer / Client, Business Analyst, System developers, Maintenance teams. It can also be a contract between purchaser and supplier.

● It will give firm foundation for the design phase.

● Supports project management and control.

● Helps in controlling and evolution of system.

A software Requirement specification should be Complete, Consistent, Traceable, Unambiguous, and Verifiable.

**3.1 Software Requirements:**

Operating System: windows 7.

Web Technologies: HTML, CSS, PHP

Database: MySQL

**3.2 Hardware Requirements:**

Processor: i3 intel core

RAM: 2GB

Hard disk: 500 GB

**3.3 Functional Requirements:**

Functional requirements are the requirements which deals with the operational

requirements of the system and the requirements that are requested by the user.

**3.4 Non- Functional Requirements:**

A non-functional requirement specifies the process that can be used to check the

operations of a system. They are contrasted with the functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is

detailed in the system architecture because they are usually architecturally significant.

requirements.

Some of them are,

* Robustness to cope with errors during execution and cope with erroneous input.
* Performance which tells us whether the input is valid or not and obtained output.
* The cost should be less when compared to the existing system.
* The system acts as platform independent.
* The system is secured.

# SYSTEM-DESIGN

## 4. SYSTEM DESIGN

**4.1 Introduction:**

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from problem domain to the solution domain. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. This document is like a blueprint or plan for the solution, and is used later during implementation, testing and maintenance.

The design activity is often divided into two separate phase-system design and detailed design. System design, which is sometimes also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats, output formats, as well as the major modules in the system and their specifications are decided.

A design methodology is a systematic approach to creating a design by application of set of techniques and guidelines. Most methodologies focus on system design. The two basic principles used in any design methodology are problem partitioning and abstraction. A large system cannot be handled as a whole, and so for design it is partitioned into smaller systems. Abstraction is a concept related to problem partitioning. When partitioning is used during design, the design activity focuses on one part of the system at a time. Since the part being designed interacts with other parts of the system, a clear understanding of the interaction is essential for property designing the part.

**4.2. UML DIAGRAMS**

UML Diagrams is a rich visualizing model for representing the system architecture and design. These diagrams help us to know the flow of the system.

Some of them are.

* Use case diagram.
* Sequence diagram
* Collaboration diagram
* Activity diagram
* Class diagram

**USECASE 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. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behaviour, perhaps the system or use case boundaries should be re-examined. Alternatively, interaction among actors can be part of the assumptions used in the use case.

**Use cases:**

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actors:**

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

**System boundary boxes:**

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not.

Four relationships among use cases are used often in practice.

**Include:**

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case.

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviours from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behaviour of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

**Extend:**

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behaviour of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modellers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

**Generalization:**

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation

**Associations:**

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modelled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

**Identified Use Cases.**

The “user model view” encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of “use case diagrams”. These diagrams describe the functionality provided by

a system to external integrators. These diagrams contain actors, use cases, and their relationships.

**Use case diagram for Farmer friendly services:**



Figure 4.1. Usecase Diagram

**Sequence diagram:**

A **sequence diagram** shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical view of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Benefits of sequence diagrams:

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

* Represent the details of a UML use case.
* Model the logic of a sophisticated procedure, function, or operation.
* See how objects and components interact with each other to complete a process.
* Plan and understand the detailed functionality of an existing or future scenario.

**Notation’s description:**

**Actor:**

* a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data)
* external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject).
* represent roles played by human users, external hardware, or other subjects.

**Lifeline:** A lifeline represents an individual participant in the Interaction.

**Activations:**

* A thin rectangle on a lifeline) represents the period during which an element is performing an operation.
* The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively.

**Call Message:**

* A message defines a particular communication between Lifelines of an Interaction.
* Call message is a kind of message that represents an invocation of operation of target lifeline.

**Return Message:**

* A message defines a particular communication between Lifelines of an Interaction.
* Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.

**Self-Message:**

* A message defines a particular communication between Lifelines of an Interaction.
* Self-message is a kind of message that represents the invocation of message of the same lifeline.

**Recursive Message**

* A message defines a particular communication between Lifelines of an Interaction.
* Recursive message is a kind of message that represents the invocation of message of the same lifeline. It is target points to an activation on top of the activation where the message was invoked from.

**Create Message**

* A message defines a particular communication between Lifelines of an Interaction.
* Create message is a kind of message that represents the instantiation of (target) lifeline.

**Destroy Message**

* A message defines a particular communication between Lifelines of an Interaction.
* Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.

**Duration Message**

* A message defines a particular communication between Lifelines of an Interaction.
* Duration message shows the distance between two-time instants for a message invocation.

**Sequence diagram for Administrator:**



Figure 4.2 Sequence diagram for administrator

**Sequence diagram for User:**

****

Figure 4.3 Sequence Diagram for User

**Sequence diagram for Agriculture officer:**

****

Figure 4.4 Sequence Diagram for Agriculture Officer

**Activity diagram:**

An activity diagram visually presents a series of actions or flow of control in a system like a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activities modelled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

**Basic Activity Diagram Notations and Symbols:**

##### **Initial State or Start Point :**

A small, filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swim lanes, make sure the start point is placed in the top left corner of the first column.

##### **Activity or Action State :**

An action state represents the non-interruptible action of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

##### **Action Flow:**

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

##### **Object Flow :**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.

##### **Decisions and Branching:**

A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labelled with a condition or guard expression. You can also label one of the paths "else."

**Guards:**

In UML, guards are a statement written next to a decision diamond that must be true before moving next to the next activity. These are not essential, but are useful when a specific answer, such as "Yes, three labels are printed," is needed before moving forward.

##### **Synchronization:**

A fork node is used to split a single incoming flow into multiple concurrent flows. It is represented as a straight, slightly thicker line in an activity diagram.

A join node joins multiple concurrent flows back into a single outgoing flow.

A fork and join mode used together are often referred to as synchronization.

##### **Time Event:**

This refers to an event that stops the flow for a time; an hourglass depicts it.

##### **Merge Event:**

A merge event brings together multiple flows that are not concurrent.

##### **Sent and Received Signals:**

Signals represent how activities can be modified from outside the system. They usually appear in pairs of sent and received signals, because the state can't change until a response is received, much like synchronous messages in a sequence diagram. For example, an authorization of payment is needed before an order can be completed.

##### **Interrupting Edge:**

An event, such as a cancellation, that interrupts the flow denoted with a lightning bolt.

##### **Swim lanes:**

Swim lanes group related activities into one column.

##### **Final State or End Point:**

An arrow pointing to a filled circle nested inside another circle represents the final action state.

**Activity diagram Administrator:**

****

Figure 4.5 Activity Diagram for Administrator

**For agricultural officer:**



Figure 4.6 Activity Diagram for Agriculture Officer

**For User:**

****

Figure 4.7 Activity Diagram for User

**Class Diagram:**

A class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations.

**Classes:**

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

**Active classes:**

This will initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

**Visibility:**

Use visibility markers to signify who can access the information contained within a class. Private visibility, denoted with a - sign, hides information from anything outside the class partition. Public visibility, denoted with a + sign, allows all other classes to view the marked information. Protected visibility, denoted with a # sign, allows child classes to access information they inherited from a parent class.

**Associations:**

Associationsrepresent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

**Multiplicity:**

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for just one company.

**Composition and aggregation**:

It is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate composition with a filled diamond. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond ends in both composition and aggregation relationships point toward the "whole" class (i.e., the aggregation).

**Generalization:**

It is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So, the class Honda would have a generalization relationship with the class car.

**Class diagram for farmer friendly services:**



Figure 4.8 Class Diagram

**4.3 ER Diagrams:**

An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities, and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

## Components of an ER Diagram

## As shown in the above diagram, an ER diagram has three main components: 1.Entity 2.Attribute 3. Relationship

### 1. Entity

An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.  
For example: In the following ER diagram we have two entities Student and College, and these two entities have many to one relationship as many students study in a single college. We will read more about relationships later, for now focus on entities.

**WeakEntity:**

An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

### 2. Attribute

An attribute describes the property of an entity. Ana attribute is represented as Oval in an ER diagram. There are four types of attributes :

* 1. Key attributes
  2. Composite attributes
  3. Multivalued attributes
  4. Derived attributes

#### **1. Key attribute**:

A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**.

#### **2. Composite attribute:**

An attribute that is a combination of other attributes is known as composite attribute. For example, in student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.

#### **3. Multivalued attribute**:

An attribute that can hold multiple values is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

#### **4. Derived attribute:**

A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

### 3. Relationship

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

* 1. One to One
  2. One to Many
  3. Many to One
  4. Many to One

#### **1. One to One Relationship**

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.

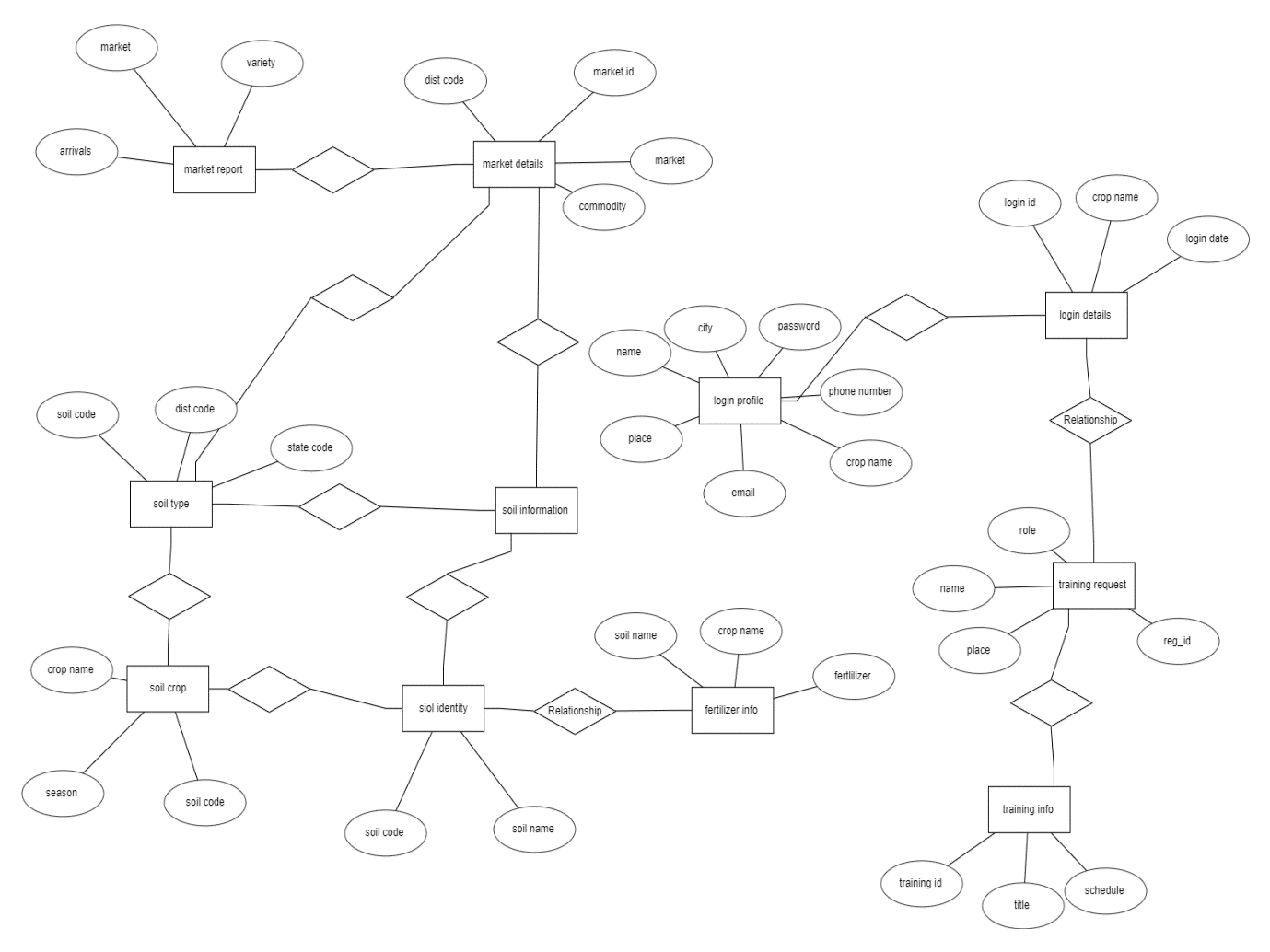
#### **2. One to Many Relationship**

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationships. For example – a customer can place many orders but a order cannot be placed by many customers.

#### **3. Many to One Relationship**

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college, but a student cannot study in many colleges at the same time.

**ER Diagram for farmer friendly services:**



# SYSTEM IMPLEMENTATION

## 5. SYSTEM IMPLEMENTATION

**5.1 Introduction:**

Today we are living in a technologically advanced era where everything from education, business, and finance to entertainment via a computer device using various platforms. Due to extensive usage of web applications, all types of business are progressively shifting to latest gadgets.

To serve in this recent growing trend we have proposed a web application. It will be used to establish connections between users and local professional workers to provide repairing and maintenance services and will enhance employment opportunities. This system provides the solutions of all problems related to home, office, or any other place.

Farmer friendly services is a contribution towards solution for farmers who are unaware of the using the technology to yield more. This application is designed and developed using PHP (PHP database connectivity). Website backend is developed using JavaScript. Frontend is designed with CSS, HTML.

The main objective of this work is to develop a system comprising of a website. Which includes the following features.

* The main objective of the project is to provide information to the farmers and

agricultural students. It provides better solutions for farmers in providing the information regarding soils, fertilizers.

* It helps the users in getting training regarding various technologies that can be used in farming.

**5.2 Project Modules**

The modules present in our project are:

1. User Module
2. Soils and Fertilizers
3. Crop Details
4. Market Details
5. videos
6. **User Module:**

First, to enter this system the users have to login to this system. Basically, there are 3 types of users in this system.

* + Admin users - Has full access to all the modules of this system.
  + Farmers and Agriculture Students – Has restricted access. i.e., Normal users have access to some of the modules only.
  + Agricultural officers: Has also restricted access.

**2.Soils and Fertilizers:**

This module is used to maintain the various Soils and Fertilizers Details. This module will be enabled only to the admin type of users.

This module contains:

* + A separate screen should be provided to maintain the Soils and Fertilizers Details.
  + If a new Soil Information is received it should be added to the System.
  + If a new Fertilizer information is received it should be added to the system with the corresponding details like soil name, crop type, crop name etc.

**3.Crop Details:**

This module is used to maintain the various details about crops. This module will be enabled only to the admin type of users.

This module contains:

* + A separate screen should be provided to maintain the Crops Information. It should provide a way to add, modify and delete the crop details.
  + If a new crop information is received, it should be added to the system with the corresponding details like Soil Name, Crop Type, Crop Name and Season.

**4.Market Details:**

In this module we can maintain the market details. This module will be enabled only to the admin type of users.

This module contains:

* A separate screen should be provided to maintain the market related information. It should provide a way to add, modify and delete the market related information.
* Administrator type of user can add the commodities in the market.
* He can add the information about new markets into the system.
* He can add the market report into the system regarding a particular market and commodities prices details in that market in a day.

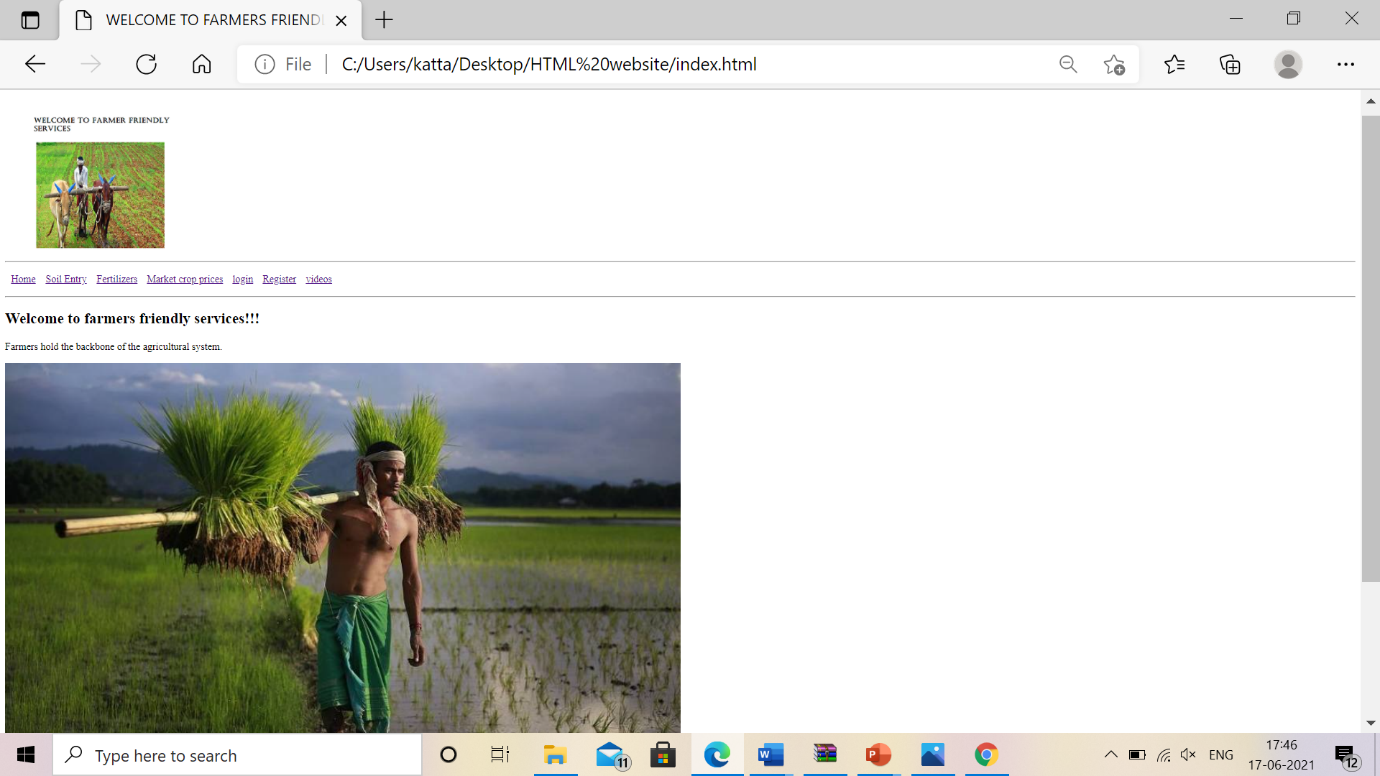
**5.videos:**

This module provides the mail system to every user in the system who are authenticated.

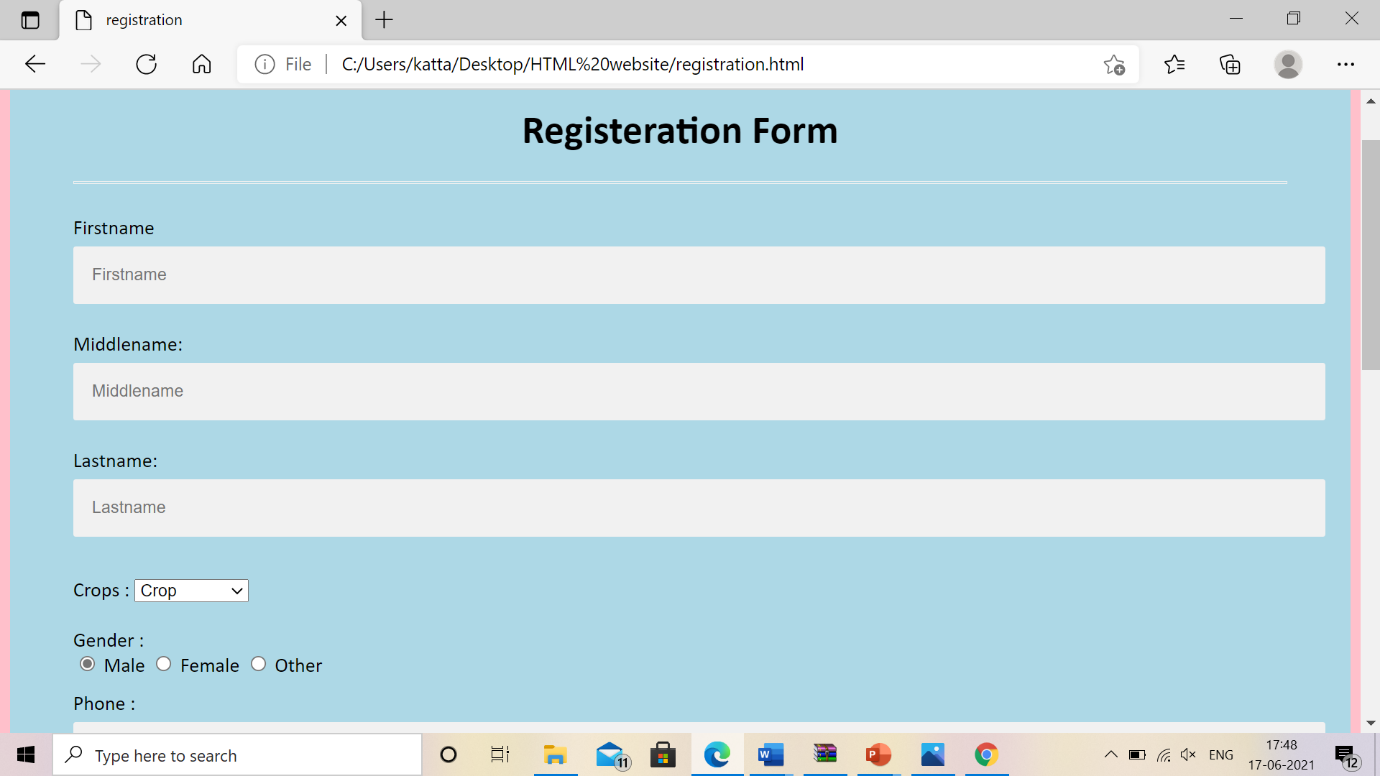
* He can view the videos which will help them for better understandings.

**5.3 Screens:**

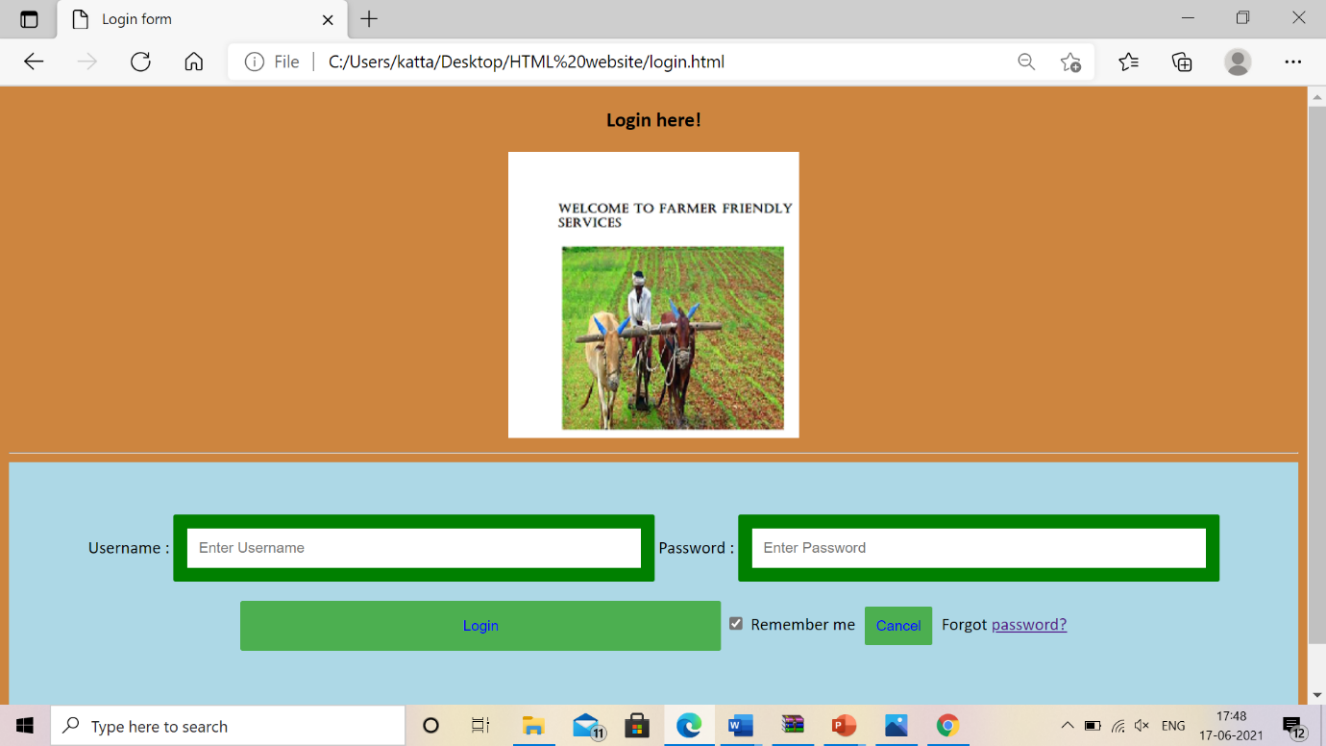
**1.Home:**



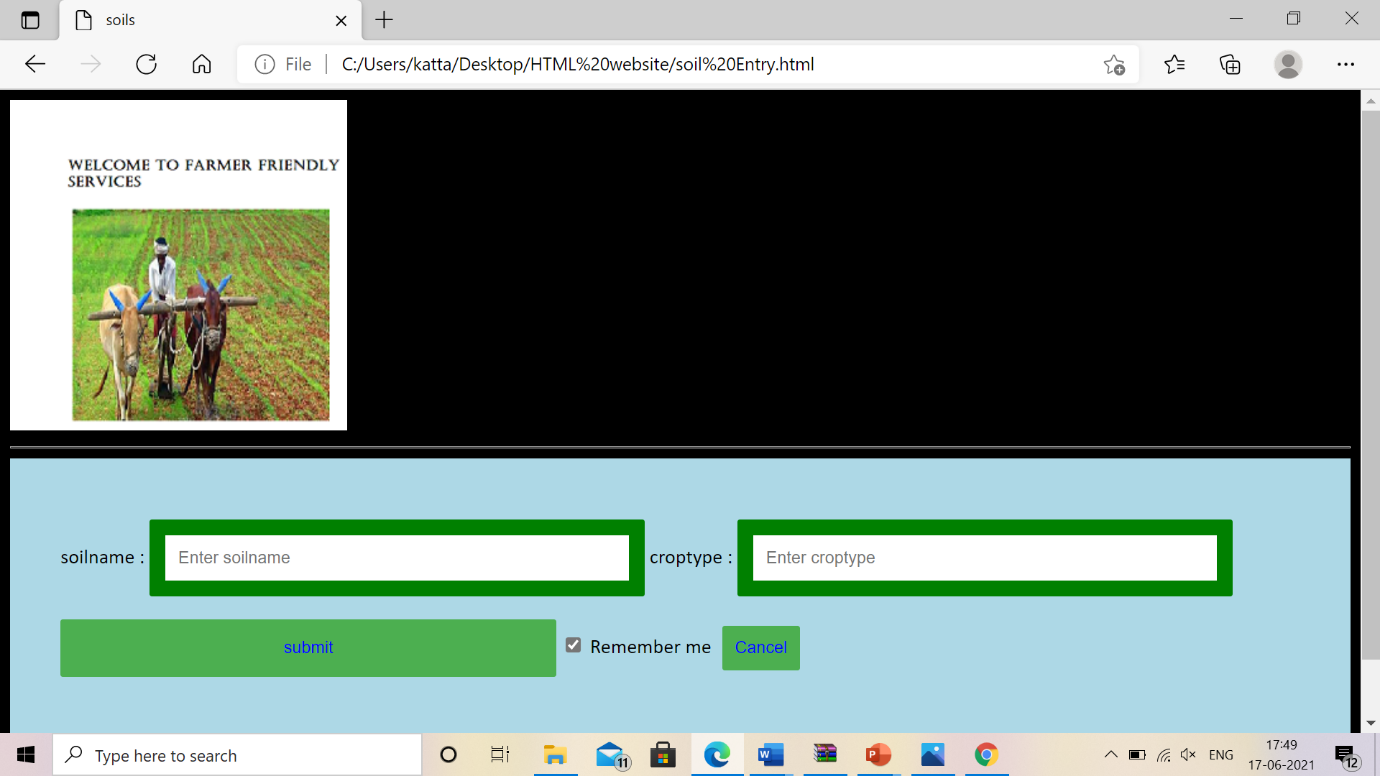
**2.Registration:**

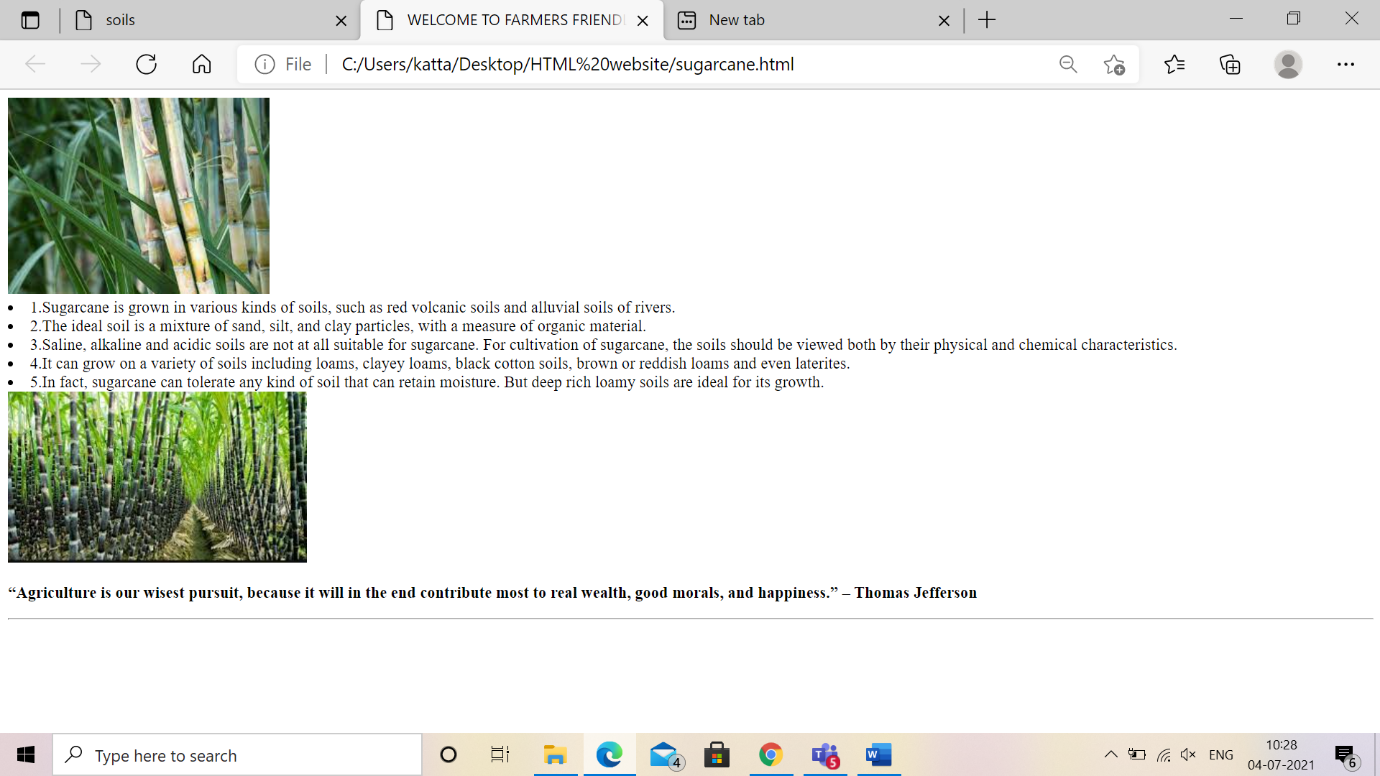


1. **Login:**



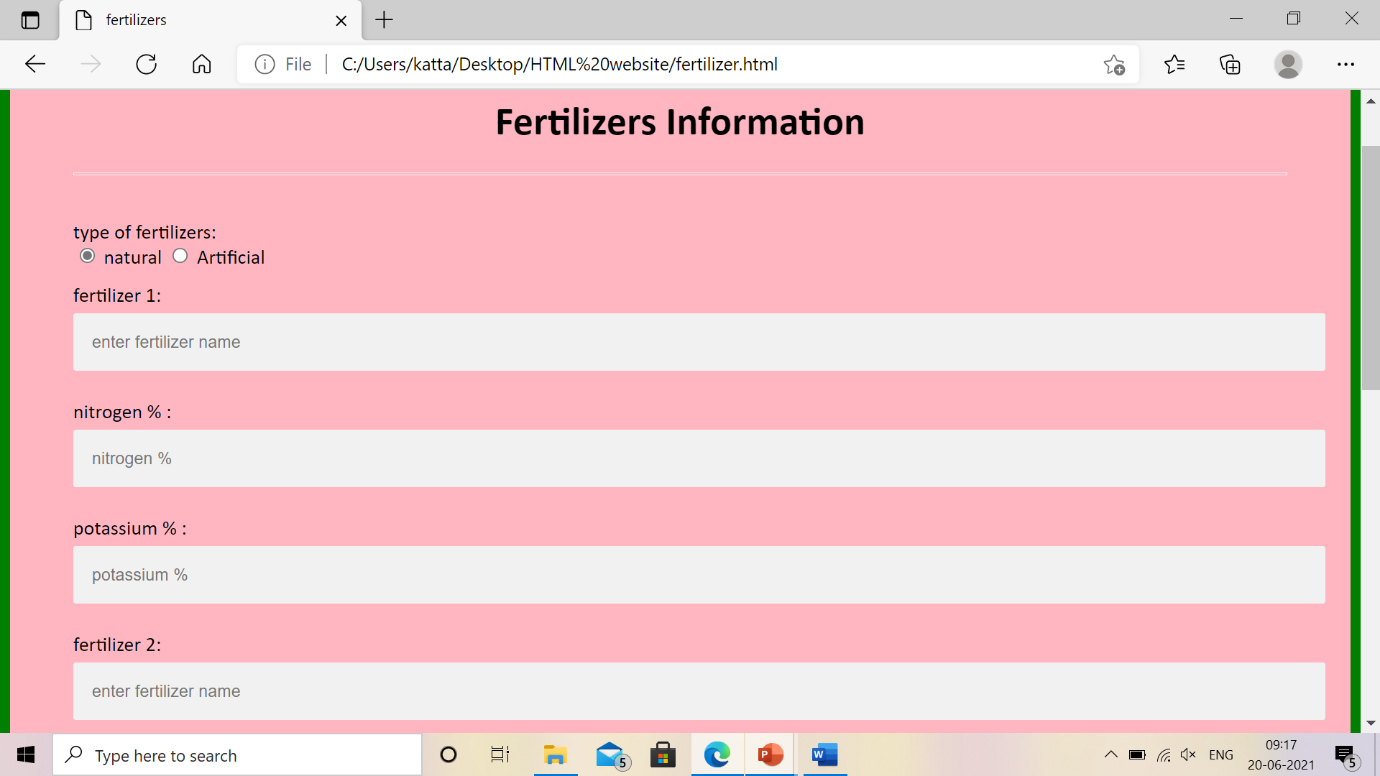
1. **Soil entry:**



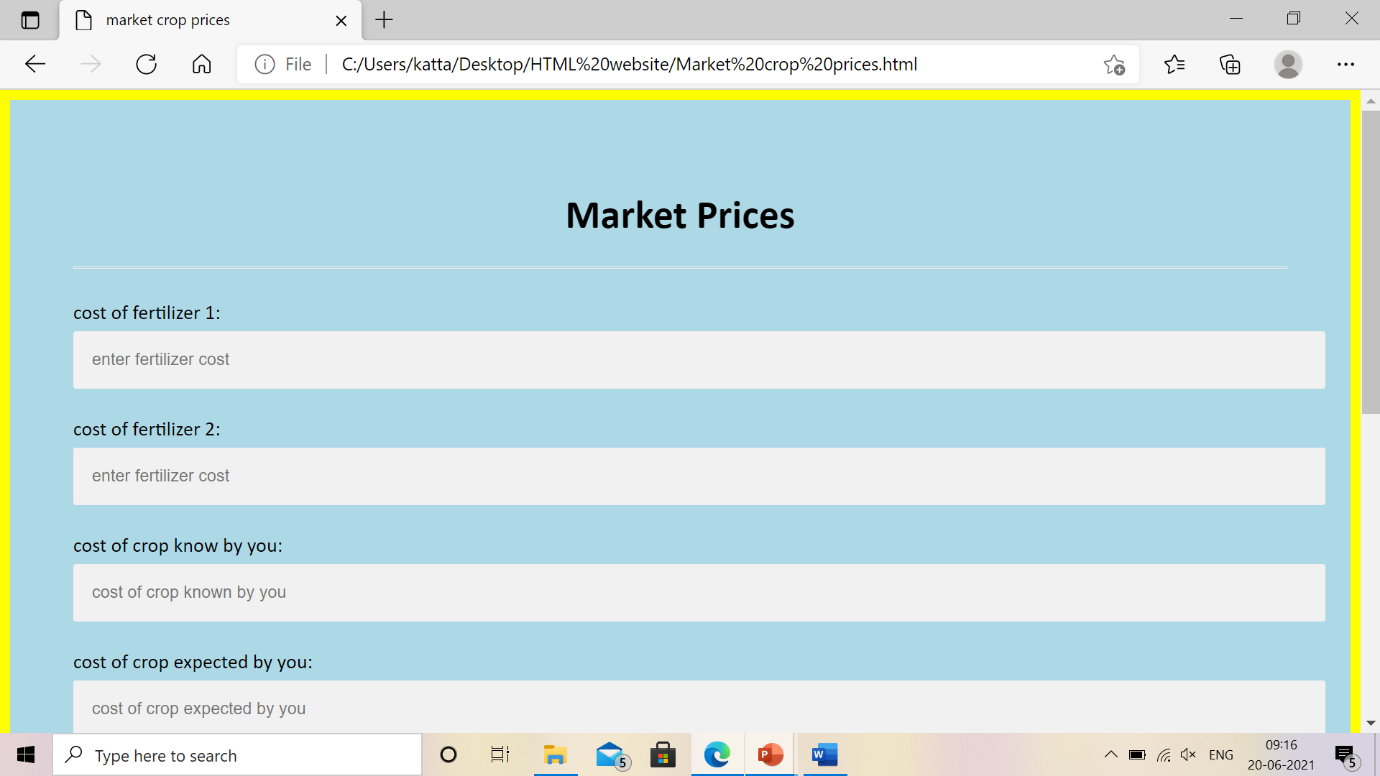




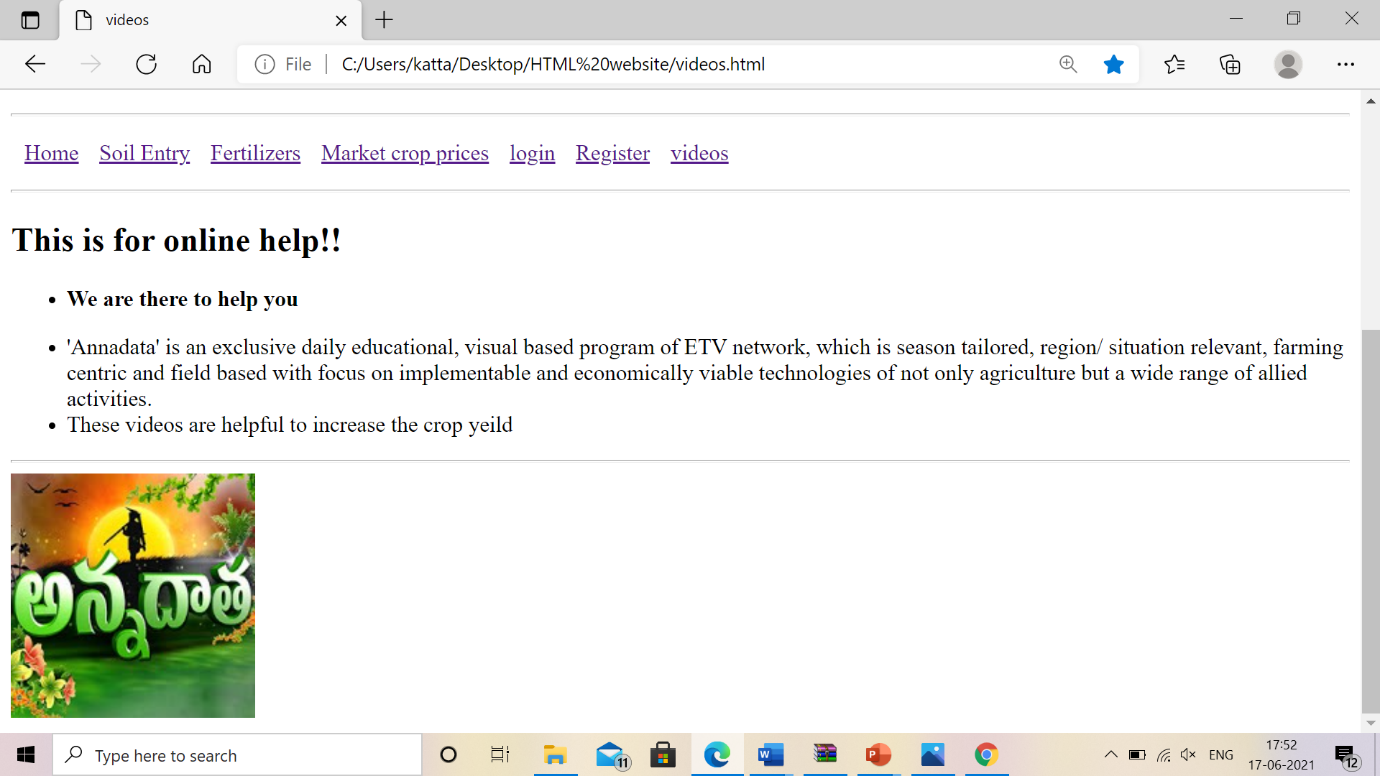
**5.Fertilizers:**



**6.Market crop prices:**



**7.online videos:**



# SYSTEM TESTING

## 6. SYSTEM TESTING

**6.1 Introduction:**

Software Testing is an important element of the software quality assurance and represents the ultimate review of specification, design, and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for III planned through testing.

**TESTING OBJECTIVES**

These are several rules that can save as testing objectives:

* Testing is a process of executing program with the intent of finding an error.
* A good test case is one that has a high probability of finding an undiscovered error.

**Test Levels**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or darkness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**6.2. TESTING METHODS**

**6.2.1 Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.

**6.2.2 Integration Testing**

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

**6.2.3 Functional Testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

**6.2.4 System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test.

**6.2.5 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user.

# 

# 

# CONCLUSION

## 7.CONCLUSION

By this project, we provide various information required for farmers and agricultural students and providing solutions to them about queries posted by them. This makes agriculture more eco-friendly, and this portal is very useful to farmers and agricultural students

**BIBLIOGRAPHY**

**8. BIBLIOGRAPHY**

* HTML 5 Black Book, Covers CSS 3, JavaScript, XML, XHTML, AJAX, PHP and jQuery, 2ed by [DT Editorial Services]
* HTML 5 Black Book, Covers CSS 3, JavaScript, XML, XHTML, AJAX, PHP
* Java The Complete Reference - Eleventh Edition
* SQL The Complete Reference, 3rd Edition.
* <https://www.mysqltutorial.org>
* <https://www.javatpoint.com/mysql-tutorial>
* <https://html.com>
* <https://www.slideshare.net/florianletsch/html-for-beginners-30020829>
* <https://www.javatpoint.com/css-tutorial>
* [www.w3schools.com/Css/css\_intro.asp](http://www.w3schools.com/Css/css_intro.asp)

# APPENDIX

## 9. APPENDIX

**9.1 Introduction to Java:**

JAVA was developed by Sun Microsystems Inc in 1991, later acquired by Oracle Corporation. It was developed by James Gosling and Patrick Naughton. It is a simple programming language.  Writing, compiling and debugging a program is easy in java.  It helps to create modular programs and reusable code.

### Java is a platform independent language.

### Compiler (javac) converts source code (.java file) to the byte code (.class file). As mentioned above, JVM executes the byte code produced by compiler. This byte code can run on any platform such as Windows, Linux, Mac OS etc. which means a program that is compiled on windows can run on Linux and vice-versa. Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems. That is why we call java as platform independent language.

### Java is an Object-Oriented language.

Object oriented programming is a way of organizing programs as collection of objects, each of which represents an instance of a class.

4 main concepts of Object-Oriented programming are:

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

### Simple:

### Java is considered as one of simple language because it does not have complex features like Operator overloading, [Multiple inheritance](https://beginnersbook.com/2013/05/java-multiple-inheritance/), pointers and Explicit memory allocation.

### Robust Language

Robust means reliable. Java programming language is developed in a way that puts a lot of emphasis on early checking for possible errors, that’s why java compiler is able to detect

errors that are not easy to detect in other programming languages. The main features of java that makes it robust are garbage collection, Exception Handling and memory allocation.

**Secure**

We do not have pointers and we cannot access out of bound arrays (you get ArrayIndexOutOfBoundsException if you try to do so) in java. That’s why several security flaws like stack corruption or buffer overflow are impossible to exploit in Java.

### Java is distributed.

Using java programming language, we can create distributed applications. RMI (Remote Method Invocation) and EJB (Enterprise Java Beans) are used for creating distributed applications in java. In simple words: The java programs can be distributed on more than one system that are connected to each other using internet connection. Objects on one JVM (java virtual machine) can execute procedures on a remote JVM.

### Multithreading

Java supports multithreading. Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilisation of CPU.

### Portable

As discussed above, java code that is written on one machine can run on another machine. The platform independent byte code can be carried to any platform for execution that makes java code portable.

**9.2 Introduction to HTML:**

HTML is the standard markup language for creating Web pages.

* HTML stands for Hyper Text Markup Language
* HTML describes the structure of a Web page.
* HTML consists of a series of elements.
* HTML elements tell the browser how to display the content.
* HTML elements are represented by tags.
* HTML tags label pieces of content such as "heading", "paragraph", "table", and so on.
* Browsers do not display the HTML tags, but use them to render the content of the page.

**Below is a visualization of an HTML page structure:**

<html>

<head>

<title>Page title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

</body>

</html>

**Introduction to CSS :**

**Styling HTML with CSS:**

**CSS** stands for **C**ascading **S**tyle **S**heets.

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

* **Inline** - by using the style attribute in HTML elements.
* **Internal** - by using a <style> element in the <head> section.
* **External** - by using an external CSS file.

The most common way to add CSS, is to keep the styles in separate CSS files. However,

here we will use inline and internal styling because this is easier to demonstrate.

## Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

This example sets the text color of the <h1> element to blue:

### Example

<h1 style="color:blue;">This is a Blue Heading</h1>

## Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element:

<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {background-color: powderblue;}  
h1   {color: blue;}  
p    {color: red;}  
</style>  
</head>  
<body>  
<h1> This is a heading

</h1>  
<p>Thisisaparagraph.</p>  
</body>  
</html

## External CSS

An external style sheet is used to define the style for many HTML pages.

***With an external style sheet, you can change the look of an entire web site, by changing one file!***

To use an external style sheet, add a link to it in the <head> section of the HTML page:

<!DOCTYPE html>  
<html>  
<head>  
  <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
  
<h1> This is a heading.

</h1>  
<p> This is a paragraph.

</p>  
  
</body>  
</html>

**9.3Introduction to MySQL:**

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

49| Page

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons –

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc. SQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.