SEEDS

Prepared by
Milind Kurma, Punit Malpani, Luke Kim, Sharmisha Parvathaneni

CS 440
at the
University of Illinois Chicago

Fall 2022

Table of Contents

List of Figures

List of Tables

I Project Description

Even now, with technology assisting every sector of employment, agriculture is a sector where technology is not incorporated. In order to assist farmers, we created the SEEDS project.

Seeds is an online program designed to assist farmers in increasing yield by recommending the best crops to plant in a specific type of soil and which pesticides to apply based on the crop. Furthermore, SEEDS can be utilized to predict plant health using image processing techniques.

1 Project Overview

The main goals of this project are to make farmers' lives easier by adding machine learning models and suggesting crops for greater production. The project is broken into two parts.

1. Simple Search: We attempted to fit a variety of soil types and seasons in this feature, based on which the farmer can determine which crops to plant and obtain a higher yield. Pesticide recommendations are also included.

2. Advanced Search: This section includes a Machine Learning model that analyzes an image and predicts whether or not the plant is healthy. If not, what exactly is the issue? Sensor-based measurements for soil acidity, temperature, and moisture are also incorporated, resulting in even better predictions for the farmer.

2 Project Domain

SEEDS is a Full-stack project, web application that uses the full range of web development technologies, from the front-end to the back-end.
We have used the following technologies to achieve our prototype:

- JavaScript
- HTML/CSS
- Python(Sklearn, TensorFlow, Keras)
- Django

3 Relationship to Other Documents

This document is related to the project scenario I and II along with other required documents such as “Object-Oriented Software Engineering” by Bruegge and Dutoit and “UML distilled” by Fowler.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

For this Seed Application, the different variety of soil types can have multiple meanings. Since farmers exposed certain types of soils and the definitions may vary, it is important to clear and standardize the definitions here.

- **Alluvial and volcanic soils**: Alluvial soil is a blend of soils, composed of a combination of clay, silt, sand, and gravel.

- **Black cotton soils:** inorganic clay formed in regions having poor drainage conditions. It contains varieties of mineral elements and is very sensitive to water or moisture.

- **Deep loamy soils:** contain more nutrients, moisture, and humus than sandy soils, have better drainage and infiltration of water and air than silt- and clay-rich soils, and are easier to till than clay soils.

- **Deep,sandy loams:** have a high concentration of sand that gives them a gritty feel. In gardens and lawns, sandy loam soils are capable of quickly draining excess water but can not hold significant amounts of water or nutrients for your plants.

- **Dry Sandy soil:** extremely well-drained and rarely experiences any standing water. Plants that thrive in dry sandy soils are often lower-growing species, due to the reduced availability of both moisture and nutrients.

- **Fertile volcanic red earth or deep sandy loam:** dominated by sand particles, but contain enough clay and sediment to provide some structure and fertility. There are four different types of sandy loam soil that are classified based on the size of the sand particles in the soil.

- **Gray alluvial soils**: lacks phosphorus. The color of the soil varies from light gray to ash. It is great for the cultivation of wheat, Rice, maize, oilseeds, sugarcane, etc.

- **Heavy clay and silt loam:** More fertile than sandy soils, silty soil is the intermediary between sandy and clay soils. Silty soils have a greater tendency than other types to form a *crust*. When dry, silty soils feel floury to the touch, but when wet, you can easily form balls in your hand.

- **Light sandy loams to red clay**: well-drained and retain heat. In warm climate regions, sandy soils make wines that are ‘softer’ with less color, lighter acidity and tannin.

- **Loam soils**: made with a balance of the three main types of soil: sand, silt, and clay soil.

4b UML and Other Notation Used in This Document

UML: Use-Case Diagram:

To generalize the overall application of the Coding project symbols, notations, and diagrams, it is useful to go over the use-case diagram to visualize the web application.

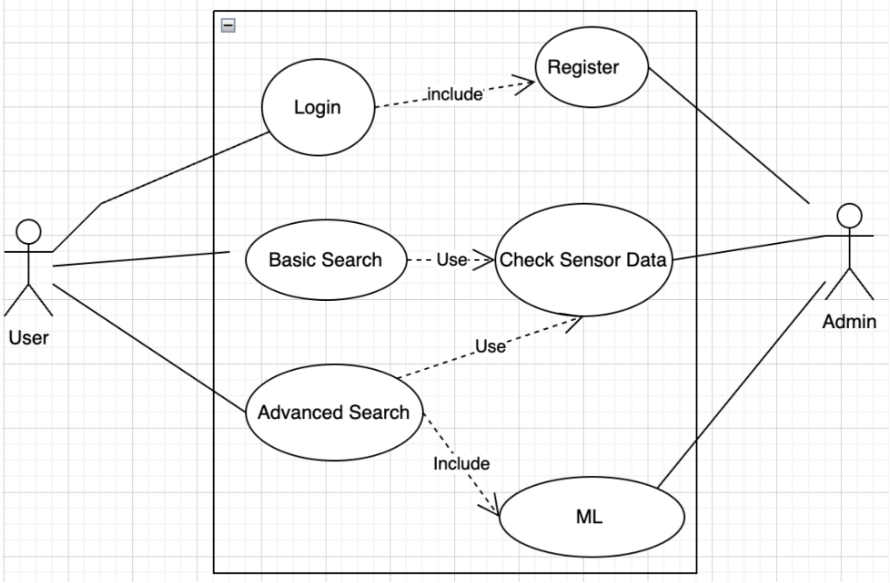


Fig.1 Use-Case Diagram

The graphic nodes that can be included in use-case diagrams are shown. The UML standard indicates that certain elements are typically drawn on certain diagram types, but this is not a prescription.

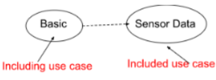
Node Type	Notation	Reference
Association		Association in page 37 “UML Distilled” by Fowler.
Dependency	- - - - - - - - - - - - - - >	Dependency in page 44 “UML Distilled” by Fowler.
Usage	- - - - - - - - - - - - - - ->	Usage in page 118 “UML Distilled” by Fowler.
Include	 Basic Sensor Data Including use case Included use case	Included in page 82 “UML Distilled” by Fowler.
synchronous		

Table 1. UML Notations

UML: Time sequence diagrams

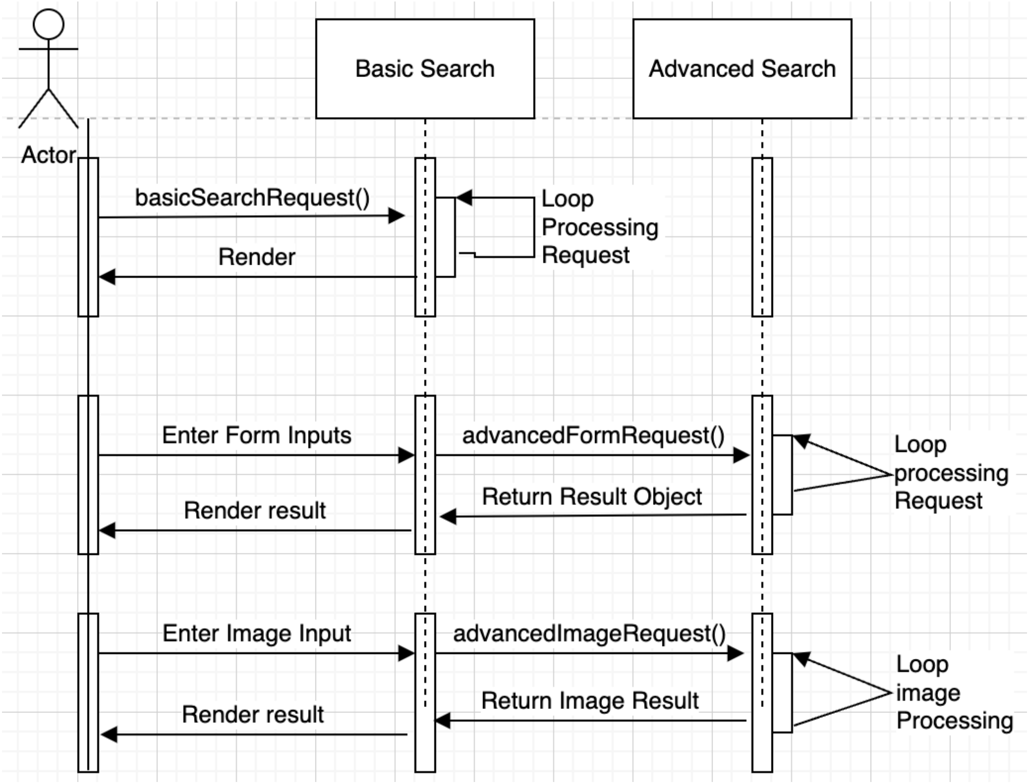


Fig2. UML Sequence Diagram

Time sequence diagram displays the sequence of interaction between an object and actor participating in an action, which consists of the object and the time at which they are interacting with each other.

4c Data Dictionary for Any Included Models

Sensor data dictionary

pH	Most soils have pH values between 3.5 and 10**.** In higher rainfall areas the natural pH of soils typically ranges from 5 to 7, while in drier areas the range is 6.5 to 9. Soils can be classified according to their pH value: 6.5 to 7.5 neutral.
temperature	The degree or intensity of heat present in a substance or object. However, in Seed App, we deal with regional temperature.
humidity	Quantity representing the amount of water vapor in the atmosphere or in a gas.

Table 2. Sensor data dictionary

For simulating the sensor data, we could design a database by taking each sensor values and user inputs. The resulting database is 2NF, so the following diagram shows the database design and model of the project.

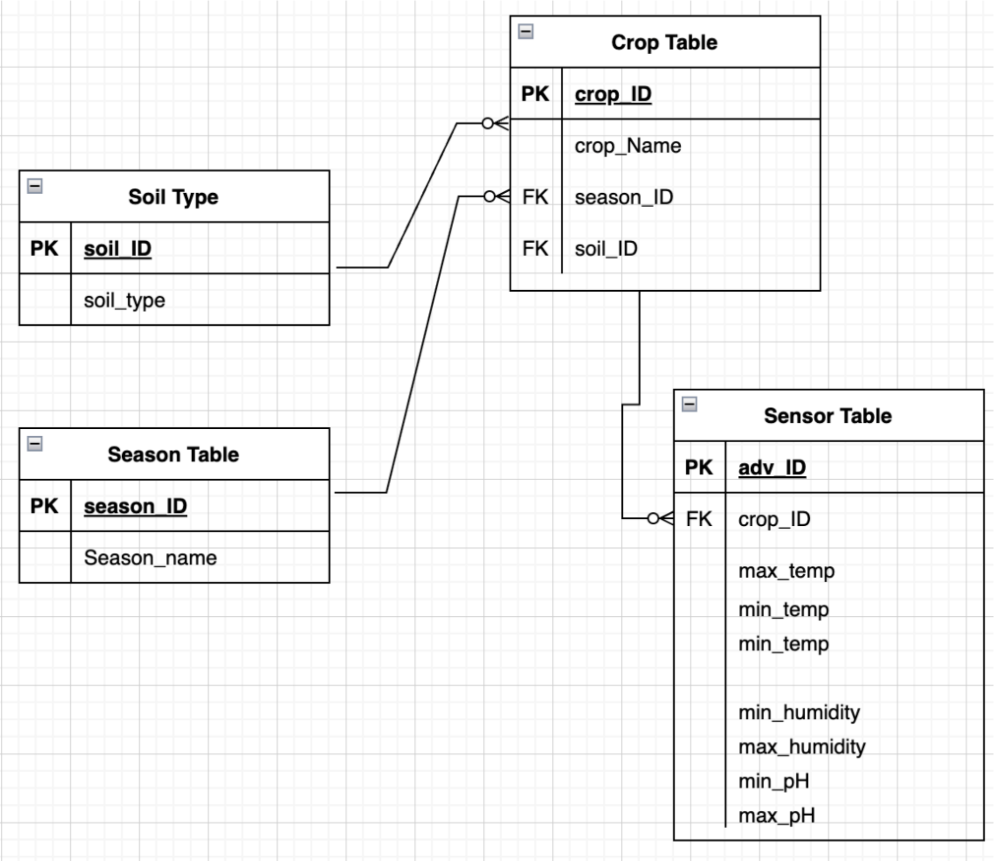


Fig 2. ER-Diagram showing all the tables and the relationship between them

II Project Deliverables

5 First Release

The primary purpose of this project's first release was to provide an attractive, intelligible user interface with correct display of basic search and login pages.

We established a database with descriptions of various types of soils, seasons, and optimum crop suggestions, as well as more information about pesticides that can be employed.

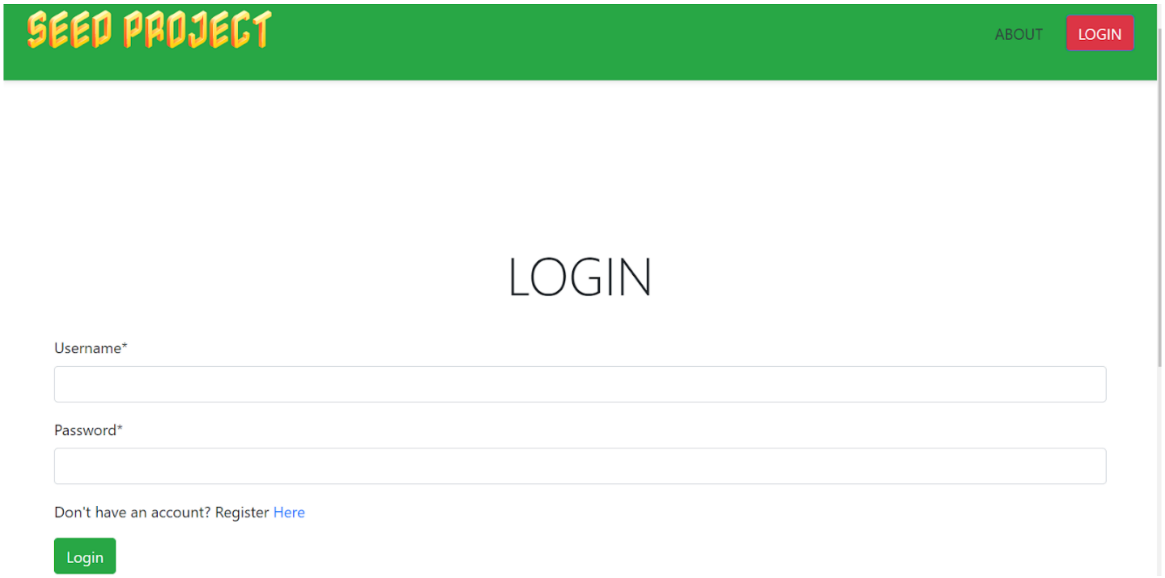
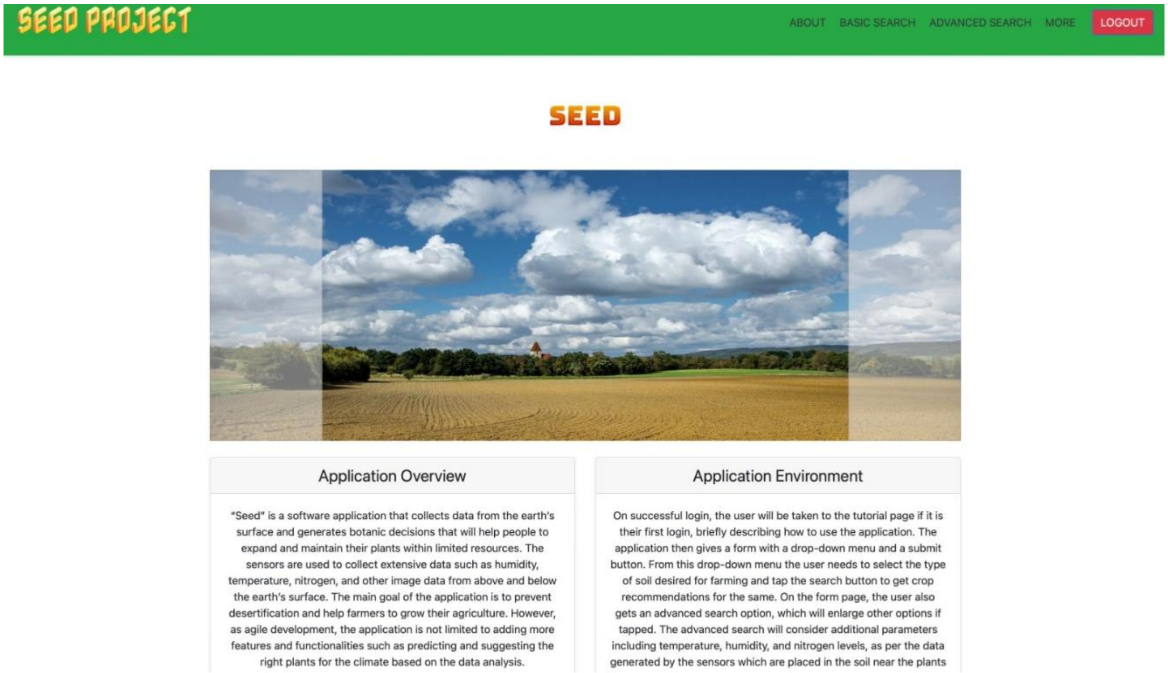


Fig.4 -Login Page



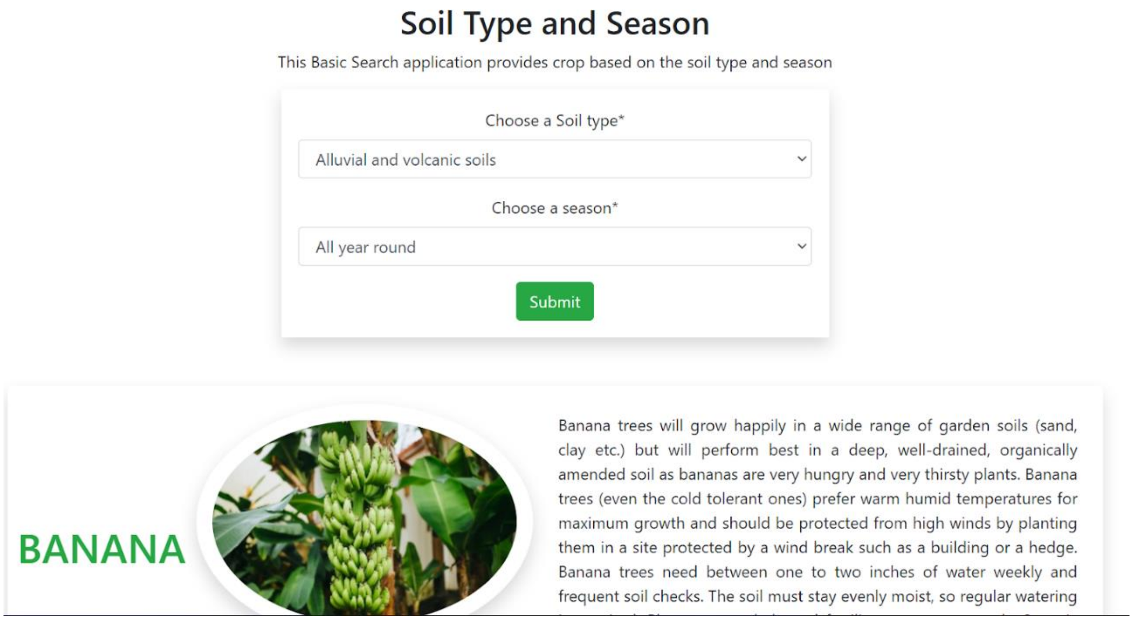
Application Overview

"Seed" is a software application that collects data from the earth's surface and generates botanic decisions that will help people to expand and maintain their plants within limited resources. The sensors are used to collect extensive data such as humidity, temperature, nitrogen, and other image data from above and below the earth's surface. The main goal of the application is to prevent desertification and help farmers to grow their agriculture. However, as agile development, the application is not limited to adding more features and functionalities such as predicting and suggesting the right plants for the climate based on the data analysis.

Application Environment

On successful login, the user will be taken to the tutorial page if it is their first login, briefly describing how to use the application. The application then gives a form with a drop-down menu and a submit button. From this drop-down menu the user needs to select the type of soil desired for farming and tap the search button to get crop recommendations for the same. On the form page, the user also gets an advanced search option, which will enlarge other options if tapped. The advanced search will consider additional parameters including temperature, humidity, and nitrogen levels, as per the data generated by the sensors which are placed in the soil near the plants

Fig.5 - Home Page



Soil Type and Season

This Basic Search application provides crop based on the soil type and season

Choose a Soil type*: Alluvial and volcanic soils

Choose a season*: All year round

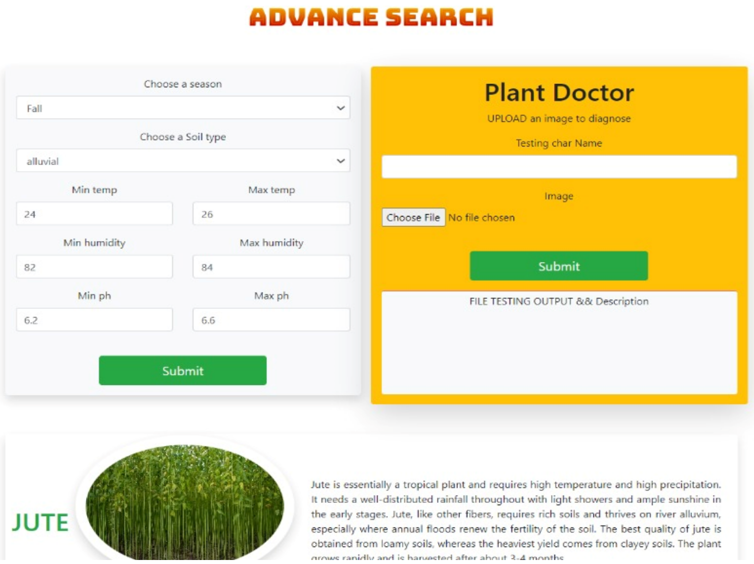
BANANA

Banana trees will grow happily in a wide range of garden soils (sand, clay etc.) but will perform best in a deep, well-drained, organically amended soil as bananas are very hungry and very thirsty plants. Banana trees (even the cold tolerant ones) prefer warm humid temperatures for maximum growth and should be protected from high winds by planting them in a site protected by a wind break such as a building or a hedge. Banana trees need between one to two inches of water weekly and frequent soil checks. The soil must stay evenly moist, so regular watering

Fig.6 - Basic Search Page

6 Second Release

We were motivated to build an advanced search that analyzes sensory data provided by sensors placed in the soil and generates the best crop selections for the second edition of our project. Along with this, we created a component called "Plant Doctor" , a machine learning model, which takes a user-supplied leaf photograph and decides if the plant is healthy or not. If not, what is the plant's problem?


ADVANCE SEARCH

Choose a season: Fall

Choose a Soil type: alluvial

Min temp: 24
Max temp: 26

Min humidity: 82
Max humidity: 84

Min ph: 6.2
Max ph: 6.6

Plant Doctor

UPLOAD an image to diagnose

Testing char Name:

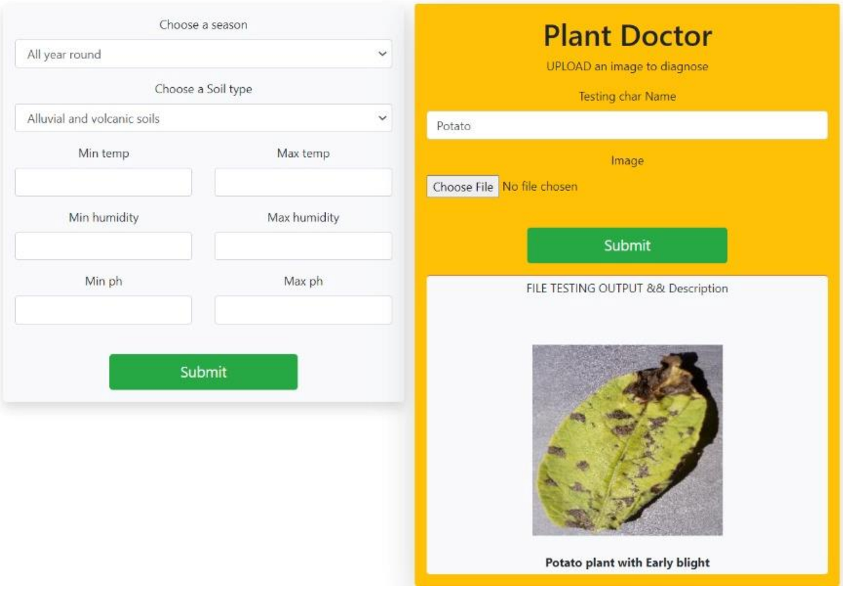
Image: Choose File No file chosen

FILE TESTING OUTPUT && Description

JUTE

Jute is essentially a tropical plant and requires high temperature and high precipitation. It needs a well-distributed rainfall throughout with light showers and ample sunshine in the early stages. Jute, like other fibers, requires rich soils and thrives on river alluvium, especially where annual floods renew the fertility of the soil. The best quality of jute is obtained from loamy soils, whereas the heaviest yield comes from clayey soils. The plant grows rapidly and is harvested after about 3-4 months

Fig.7 - Implementation of the Advance Search


Choose a season: All year round

Choose a Soil type: Alluvial and volcanic soils

Min temp:
Max temp:

Min humidity:
Max humidity:

Min ph:
Max ph:

Plant Doctor

UPLOAD an image to diagnose

Testing char Name: Potato

Image: Choose File No file chosen

FILE TESTING OUTPUT && Description

Potato plant with Early blight

Fig.8 - Implementation of Plant Doctor

7 Comparison with Original Project Design Document

The use case diagram depicted above is from the section [5] "II- Requirements - 9.Product Use Cases - 9a.Use Case Diagrams" of the project design document.

The prototype created meets the previous team's use case diagram and contains an ML model that processes the image and gives the user information on the plant's health.

Functional needs can be found in the preceding design paper's section [5] “II-Requirements- 10.Functional Requirements.” When the prototype is compared to the suggested document, it is evident that the developed project meets all of the functional requirements.

The design paper's section [5] “II-Requirements- 11. Data Requirements” contains data requirements.

This has two requirements:

- Save Farmers Account Data

- Save Botany Data

These conditions are addressed satisfactorily in the SEEDS prototype, which may be viewed on the server.

III Testing

8 Items to be Tested

The following table highlights the identified major items to be tested:

Sr. No	Items to Test
1.	Usability of Web Program
2.	Accessibility of Website
3.	Color Contrast Testing
4.	Agile Methodologies Testing

Table 3. Test Items

9 Test Specifications

ID#1 - Usability Testing:

Description: to test usability of the website, we recruit current or potential users to

1. Give a series of tasks for users to accomplish on the prototype or current site.
2. Using varying methods, observe the user's behavior to gauge the site's usability.

Items covered by this test: navigating website, clickables, and user experience.

Requirements addressed by this test: No issue

Environmental needs: Zoom Link

Intercase Dependencies: User successfully navigated and completed the scenario for the usability test.

Test Procedures: Two-part Process

1. Gather data to determine project requirements
   - Contextual inquiry
   - Directed interviews
   - Non-directed interviews

2. Synthesize that data to improve usability.
   - Surveys and Questionnaires

Input Specification: User asked to put common sensor values into basic form and advanced form to retrieve the result. Image was uploaded to test the ML processing.

Output Specifications: User gets expected result.

Pass/Fail Criteria: Passed, but users complain that complex advanced form input fields are not user friendly.

ID#2 - Accessibility Testing:

Description: In order to test the accessibility of the website, we used https://wave.webaim.org/ to test our web accessibility.

Items covered by this test: missing HTML tags, such as "alt" for image.

Requirements addressed by this test: 5 minor issues.

Environmental needs: Automation test pasting URL into the testing website.

Intercase Dependencies: No steps needed.

Test Procedures:

1. Create project URL, paste the URL into the https://wave.webaim.org/ website.

Input Specification: No

Output Specifications: Result shown on Wave.webaim.org.

Pass/Fail Criteria: Pass.

ID#3 - Color Contrast Testing:

Description: To function as a proper application, we tested color contrast using webaim.org to use contrast checker.

Items covered by this test: color choices and contrast for color blind test.

Requirements addressed by this test: No issue

Environmental needs: None

Intercase Dependencies: None

Test Procedures: Pasting two (background and other) color hex values into https://webaim.org/resources/contrastchecker/

Input Specification: hex values

Output Specifications: Contrast

Pass/Fail Criteria: Passed.

ID#4 – Agile Methodology and Integration Testing:

Description: As an agile development, we test each method and its functionality while implementing the project requirements. To test the methods, we use both web visual and integration testing to make sure that the methods and data model works properly.

Items covered by this test: method functionality, database model, compatibility.

Requirements addressed by this test: using dynamically typed language, Python, we could both test and develop the application.

Environmental needs: Github, VSCode, Python 3.4 and above.

Intercase Dependencies: Testing methods that handle HTTP requests requires intercase dependencies such as HTML, network, database and compatibility of the currently running application.

10 Test Results

ID#1 - Accessibility Test

Expected Results: Pass

Actual Results: Pass

Test Status: 5 minor issues.

ID#2 - Usability Test

Expected Results: Pass

Actual Results: Pass

Test Status: minor complaints on complex form.

ID#3 - Contrast Test

Expected Results: Pass

Actual Results: Pass

Test Status: Pass.

ID#4 - Agile Methodology and Integration Testing:

Expected Results: Pass

Actual Results: Pass

Test Status: Pass.

11 Regression Testing

The modification to the advanced search form into the same page with image processing was a risky procedure that failed regression tests when one of the form submissions was triggered. However, as agile development with progressive testing, we fixed the issue by rendering empty objects to the front-end. Each subtle changes requires the regression testing

IV Inspection

The aim of inspection throughout this project work was to make out defects in coding and other areas. The inspection process followed for this project is similar to traditional inspection stepwise methods. Everyone in the group contributed significant code fragments and other group members inspected those fragments. Apart from code fragments, inspection process was also applied on the documentation. Contributions of individual members of the group was inspected by the other group members.

12 Items to be Inspected

12a Major and Minor Code defects

The first and most important section to be inspected was identified to be the code fragment defects. All group members submitted their individual codes and these were inspected by the other members. The following table highlights the identified major and minor code defects:

Sr. No	Code defect	Type
1.	Using the Global System Python Environment for Project Dependencies	Minor
2.	Not Pinning Project Dependencies in a requirements.txt File	Minor
3.	Grammatical / Spelling Errors on Webpage	Minor
4.	Webpage Form Validation	Major
5.	HTML Validation	Major
6.	No crop recommendations match	Major
7.	Image dimensions fix before being applied to the ML model	Major
8.	URL Security	Major

Table 4. Inspection- Identified Code Defects

12b Major defects Proportion

The Major defects proportion is the ratio of the significant faults to all defects detected. In other words, it is the number of major defects found to total found. With only the first inspection, this ratio was large. Individuals in the group proof-reed their own code fragments and eliminated certain defects and the inspection process was restarted. After eliminating the defects found in the code fragments this proportion was recalculated and the ratio reduced significantly.

12c Document

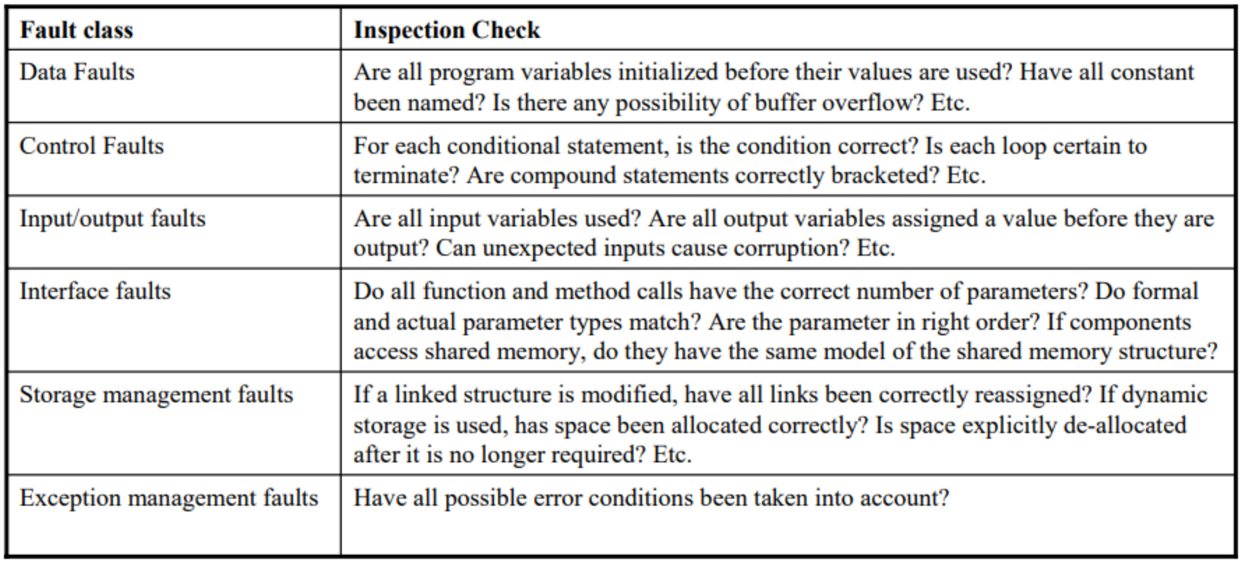
The formal documentation written for this project was also considered during the inspection phase. The documentation was initially divided into four parts and distributed among various group members. After completion of the individual sections, these sections were inspected by the other three members of the group. Requisite suggestions were put forward in during weekly meeting and appropriate changed were applied.

13 Inspection Procedures

The inspection was held in stages: 1) Overview stage, 2) Preparation Stage, 3) Meeting Stage, 4) Rework stage

Work during these stages was accomplished via communicating through one in-person weekly meet and online conference meetings where all group members participated. During the testing and inspection phase three meetings were conducted and progress was made.

During the Overview stage, major sections to be inspected were decided and a layout was planned accordingly. Using a checklist as a reference, individuals would seek out the defects in their contribution of the code. For the inspection of document, user referenced report guidelines that needed to be followed.



Fault class	Inspection Check
Data Faults	Are all program variables initialized before their values are used? Have all constant been named? Is there any possibility of buffer overflow? Etc.
Control Faults	For each conditional statement, is the condition correct? Is each loop certain to terminate? Are compound statements correctly bracketed? Etc.
Input/output faults	Are all input variables used? Are all output variables assigned a value before they are output? Can unexpected inputs cause corruption? Etc.
Interface faults	Do all function and method calls have the correct number of parameters? Do formal and actual parameter types match? Are the parameter in right order? If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	If a linked structure is modified, have all links been correctly reassigned? If dynamic storage is used, has space been allocated correctly? Is space explicitly de-allocated after it is no longer required? Etc.
Exception management faults	Have all possible error conditions been taken into account?

Fig 9. Inspection Check list reference

This inspection checklist was used as a reference and it aided in figuring out possible defects in the code fragments.

14 Inspection Results

	Member1's code	Member2's code	Member3's code	Member4's code
Member1	-	Sensitive URLs displayed Web Page Input Form Validation Error	-	No image Display in Plant Doctor section
Member2	Error when no crop recommendation match in Basic Search	-	ML model low accuracy	
Member3	-	Grammatical / Spelling errors in web pages HTML	-	Error with distinct size image upload to plant doctor
Member4	User input min parameter values		Not Pinning Project	-

	> max parameter values		Dependencies in a requirements.txt file	

Table 5. Individual member’s Inspection for other members’ code fragments

After the overview stage, each member’s contributed piece of code was inspected by the remainder of the group in the preparation stage. During the semester’s twelfth, thirteenth-, and fourteenth-weeks’ meeting minutes, individual member’s inspection for other members’ code fragments (as shown in Table 2.) were discussed and noted. Members again worked their part of the code to resolve the unprecedented faults in their codes.

Following are the resolutions to flaws detected:

Sr. No	Code defects	Resolutions
1.	Not Pinning Project Dependencies in a requirements.txt File	Dependencies in the project were included and pinned in the requirements.txt file
2.	Grammatical / Spelling Errors on Webpage	HTML text portions were scanned and corrections were made
4.	Webpage Form Validation User input min parameter values > max parameter values	User input data in signup and search forms were restricted with appropriate limits
5.	HTML Validation	HTML code was checked on https://validator.w3.org and code was refined
6.	Error when no crop recommendation match in Basic Search	Empty object with text message was returned to eliminate Internal Server Error in case no recommendations exist
7.	Error with distinct size image upload to plant doctor	Image dimensions were refined to static before being applied to the ML model
8.	Sensitive URLs displayed	Get requests were replaced with Post where necessary
9.	ML model low accuracy	Model was re-trained with more

		classified image data set and higher epoch count to achieve better accuracy

Table 6. Resolutions to flaws

V Recommendations and Conclusions

With the end of testing & inspection phase, the inspection code fragments flaws were recognized and corrected during the inspection process. Further, in order to improve the robustness of the system, following are recommended:

- The size of artifacts (documents, pages) involved could be optimized
- Code length could be reduced using more sophisticated and optimized algorithms
- Cloud servers with powerful hardware running significantly help speed up the Plant Doctor feature, which works on the supervised ML model
- Inspection can be further extended by including parameters like Defect detection rate – the number of major defects found per review hour and Defect Density- Total defects found / size

In conclusion, our team was delighted with the turn of events and the knowledge we acquired in this course. It has provided with the knowledge and experience to start or work on designing and developing any real-world software applications.

VI Project Issues

15 Open Issues

Following are some open issues that currently needs solving:

- Scalability in IoT systems has become an issue due to many devices demanding simultaneous connectivity. Despite these efforts, obstacles remain, such as the requirement for IoT nodes to provide a greater variety of services, such as functional scalability, access control, data storage, fault tolerance, and privacy and security, to mention a few.

- The lack of privacy standards and end-to-end security solutions has long been a worry for traditional IoT deployments, and wireless IoT has much greater difficulties in these areas.

Several hardware and software solutions are aimed at resolving privacy and security challenges. In terms of hardware, RFID, as well as later versions of 5G and other local network protocols, are critical in addressing security concerns. In terms of software, the Key Management System (KMS), which includes a zero-trust network feature, and blockchain are rapidly tackling privacy and trust problems with enhanced security features.

- Due to the expansion of IoT nodes, a paradigm change from Internet-of-Things to Internet-of-Everything is underway, necessitating new methods of autonomic administration to make the network proactive rather than reactive. The primary idea behind self-organization in IOT systems is to automatically and coordinately adjust to changing surroundings through the use of one or more control loops that reconfigure system behavior on-demand to maintain it within specified parameters.

16 Waiting Room

Implementations that we would like to have but couldn’t get to are listed below.
While developing Seeds applications, data from different types of sensors could be used, like water temperature and soil temperature data which can help seeds applications to run smoothly.

A function developed in this application to calculate the results and provide recommendations on data provided by different sensors could provide more data to give accurate results.

A camera module and a GPS module attached to an agricultural machine or drone could be used to capture live images of the field and affected areas.

A live chat option could be provided to the user so that an expert could assist them with their questions and issues.

17 Ideas for Solutions

We could add additional parameters to the ML model to make accurate recommendations. Some of the Seeds application issues will be easy and quick fixes that can come up based on the year the application is being released. A camera module could upload live images to the database, which can be virtually processed to understand the field's condition and the infection's exact location.

A chatbot could be developed to address some of the FAQs and redirect the user to an expert when more help is needed that requires technical expertise.

18 Project Retrospective

The people in the group made a difference in software engineering. We were open enough to discuss the strengths and weak points of each group member at the beginning of the semester, and doing so helped us focus on working on the weakness. Pair programming was the central theme in our group, as that helped the better coders to teach others. We also partnered up to increase our presentation and writing skills. Our group followed an agile methodology, and we all agreed this worked best and had an excellent outcome. For the development project, we also went back and fixed the report based on the given reviews. The one thing we could have done better was time management, and coordination this would have helped us relieve some of the stress, but in conclusion, we were delighted with the turn of events and the knowledge we earned in this course because this knowledge will help us to design and develop any type of software application in future easily.

VII Glossary

IoT - The Internet of things (IoT) describes physical objects (or groups of such objects) with sensors processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.

ML model- A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

VIII References / Bibliography

[1] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] Robertson and Robertson, Mastering the Requirements Process.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[5] Markup Validation Service, https://validator.w3.org/

[6] Software Inspection, https://www.ece.uvic.ca/~itraore/seng426-07/notes/qual07-2.pdf