Farming Advisory

Design Document

Version 1.0.0
Status: draft

Prepared by

Bhawna 2017CSB1039

Kamal Sharma 2017CSB1084

Nitin K 2017CSB1093

Piyush Lodhi 2017CSB1097

Indian Institute of Technology Ropar

12 February, 2020

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to outline the technical design of the Online screening tool and provide an overview of the Online Portal implementation.

- Provide the link between the Functional Specification and the detailed Technical Design documents
- Detail the functionality which will be provided by each component or group of components and show how the various components interact in the design
- Provide a basis for the detailed design and development of the portal.

1.2. Scope

The Application Design outlined in this document builds upon the scope defined in the Requirements phase through the SRS document and focuses on the design details of the portal.

1.3. Document Organization

Introduction	Provides information related to this document (e.g. purpose, term definitions, etc.)
Design Overview	Describes the approach, architectural details, and constraints used in design and development
Deployment Diagram	Describes the various system components and the integration between them.
Application Architecture	Describes the application architecture in terms of different layers of the application.
Application Implementation	Describes the application implementation in terms of tools and technologies/frameworks to use with the help of the deployment diagram.
Discussion of Design Decisions	Describes the design options available, pros and cons and the design choice adopted.
Assumptions and Constraints	Describes the assumptions made for the choice of design detail and the constraints thus imposed on the portal.

1.4. Audience

The Intended Audience for the project Farming Advisory Portal Design Document are:

- **<u>Farmers</u>**
    - Individual Farmers
    - Farmer Groups
- **<u>Central Government</u>**
    - Department of Agriculture and Farmers' Welfare
    - Academic and Research Institutions
- **<u>State Government</u>**
    - Department of Agriculture, Punjab
    - Attached Offices
    - Regional Training Institutions
    - State Level Academic and Research Institutions
- **<u>Private Sector</u>**
    - Agricultural Business Clinics and Centers
    - Traders, Buyers and Commodity Exchanges
    - Any individuals like Researchers

2. Design Overview

2.1. Architectural Goals and Constraints

<u>Goals</u>

- To build architecture for a software solution for the purpose of farming needs.
- To provide a single-window solution to all the stakeholders by eliminating the need to visit different websites.
- To provide personalized and localized services for farming-related suggestions.
- To suggest crops and crop handling techniques that help farmers to maximize profits.
- To use minimum inputs from the farmer and at the same time suggesting the most suitable crop recommendations.
- To provide a discussion platform for sharing the knowledge and asking queries directly to domain experts.
- To provide the uniform experience to all the users in terms of design, layouts, navigation architecture and interface.

A key Architectural goal is to leverage industry best practices for designing and developing a scalable application. The patterns as well as the development guidelines for the portal are as per the industry-standard for building the Advisory Portal.

2.2. Guiding Principles

Guiding principles provide a foundation upon which to develop the target by architecture for the screening tool, in part setting the standards and measures that the software solution must satisfy.

2.2.1 Scalable

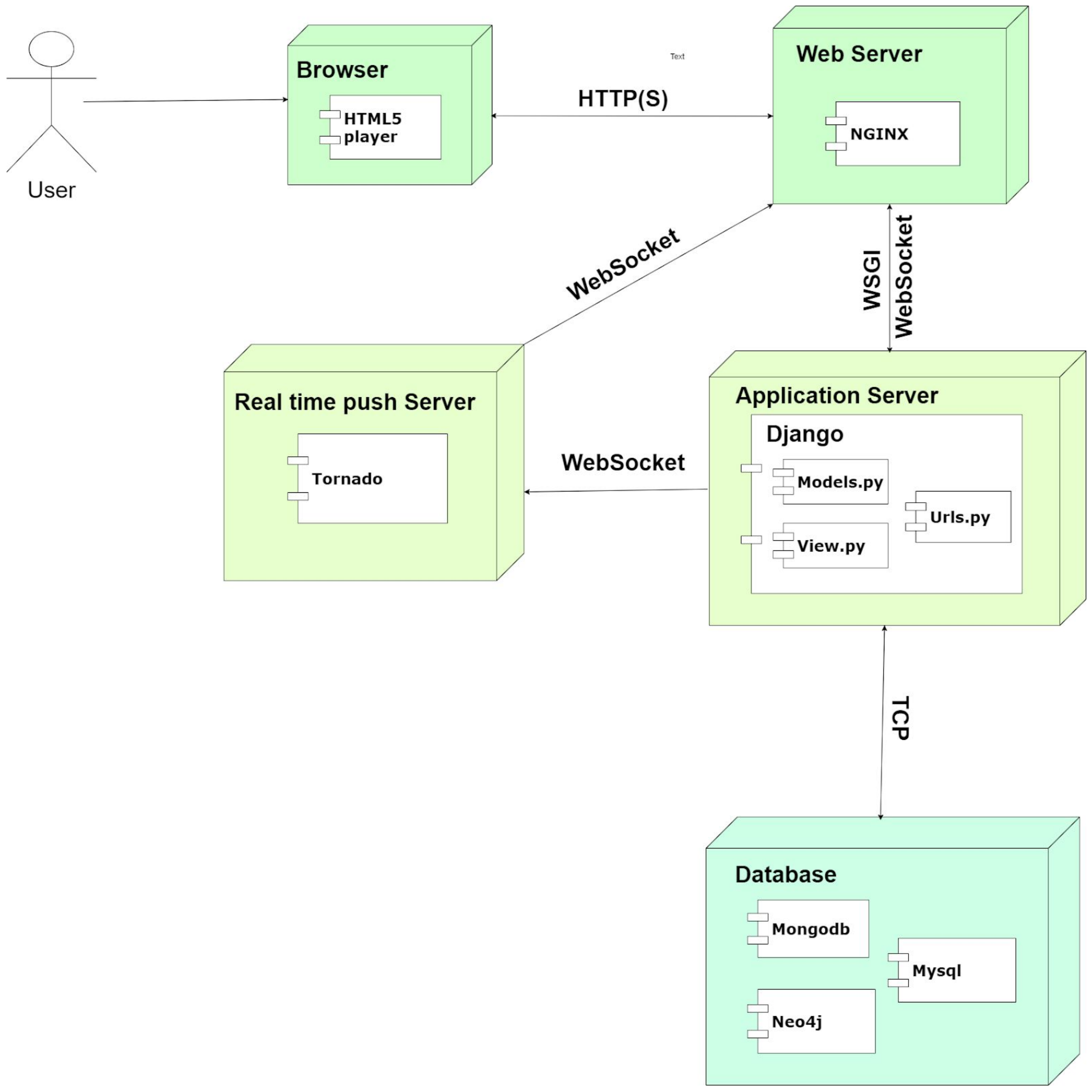
Scalability is the ability of the platform to scale both up and down to support varying numbers of users or transaction volumes. The application should be able to scale horizontally (by adding more servers) or vertically (by increasing hardware capacity or software efficiency).

2.2.2 Flexible

Flexibility is the ability of the application to adapt and evolve to accommodate new requirements without affecting the existing operations. This relies on a modular architecture, which isolates the complexity of integration, presentation, and business logic from each other in order to allow for the easy integration of new technologies and processes within the application. We intend to build a Software application design which is compliant with all of such software Engineering practices by having a layered architecture.

3. Deployment Diagram

The diagram below provides an illustration of system architecture including nodes such as hardware or software execution environments-



Components of System Architecture-

Web Server : Web server is responsible for serving web pages via the HTTP protocol to clients. The Web server sends out web pages in response to requests from browsers. A page request is generated when a client clicks a link on a web page in the browser

Application Server : Application server hosts the Online Screening application and hosts the business logic and the business model classes of applications. It serves requests for dynamic HTTP web pages from Web servers.

Runtime push Server: Tornado-based server to push updates,handling websocket connections that can pass messages from client to client , improve the security of the websocket connection, and incorporate Redis as a message broker to improve the future scalability of the server.

Database Server: It is used to store and maintain the data and information of the websites.

How do components of System Architecture interact?

- The client makes a request to our web server with his web browser.
- Our Web Server (NGINX) passes WSGI or WebSocket request to Application Server.
- If it is a WebSocket request then pass it to tornado if not then Django will handle the request.
- Wait for Django response or WebSocket response.
- Return this response to the client.
- Different type of database for storing and retrieving essential information

4. Application Architecture

4.1. User Layer: Farmers and Experts.

4.2. Presentation Layer

This Layer defines the   user interface for the application. The design patterns we will use in this layer is Model View Controller (MVC) . For this Model Component is implemented in Business Layer.

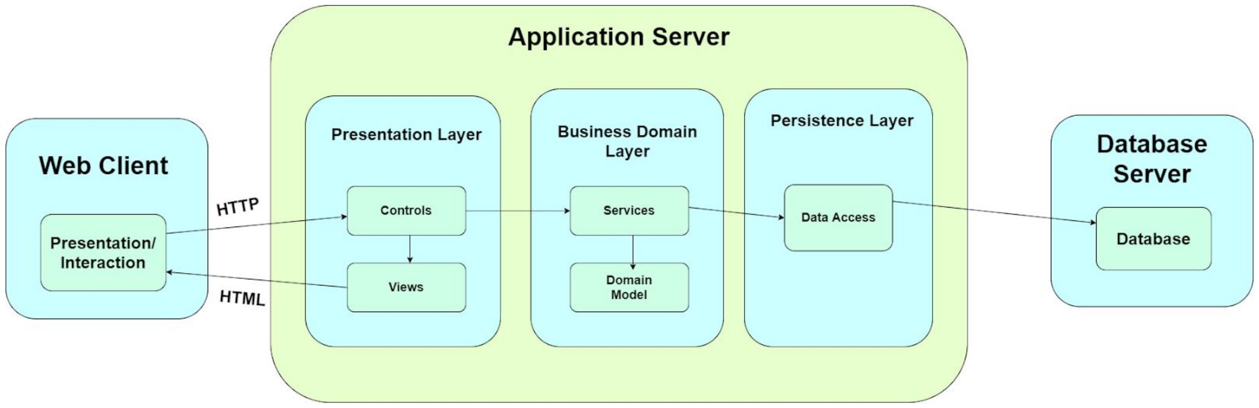
4.3. Business Layer

This Layer contains the main logic of our application .It contains Business Components, Business entity ,Business Workflow .Business Components include production cost, Selling cost ,query on discussion portal objects .
Business Entities include Farm Details ,Individual discussion thread .
Business Workflow Includes the whole process of farmers giving farm inputs to giving crop suggestions to farmers based on input .

4.4. Persistence Layer

Does storing and retrieving data from data stores . For farmer objects, its attributes are received from the persistence layer from the data store . Data Access Object (DAO) is used to access data .

Design Issue: Session and Context: Session starts with a valid login. It has a temporary life cycle which ends either by a logout, time out or exceptions like connection closing. When a user opens the portal, he is prompted to login to the system. This provides authentication context. During the session, requests are made, information is accumulated and processed . This kind of information is generalized as data context.



5. Application Implementation

5.1. UML diagrams for dynamics of static structure behavior of the system

The UML diagrams representing the dynamics of static structure are as follows.

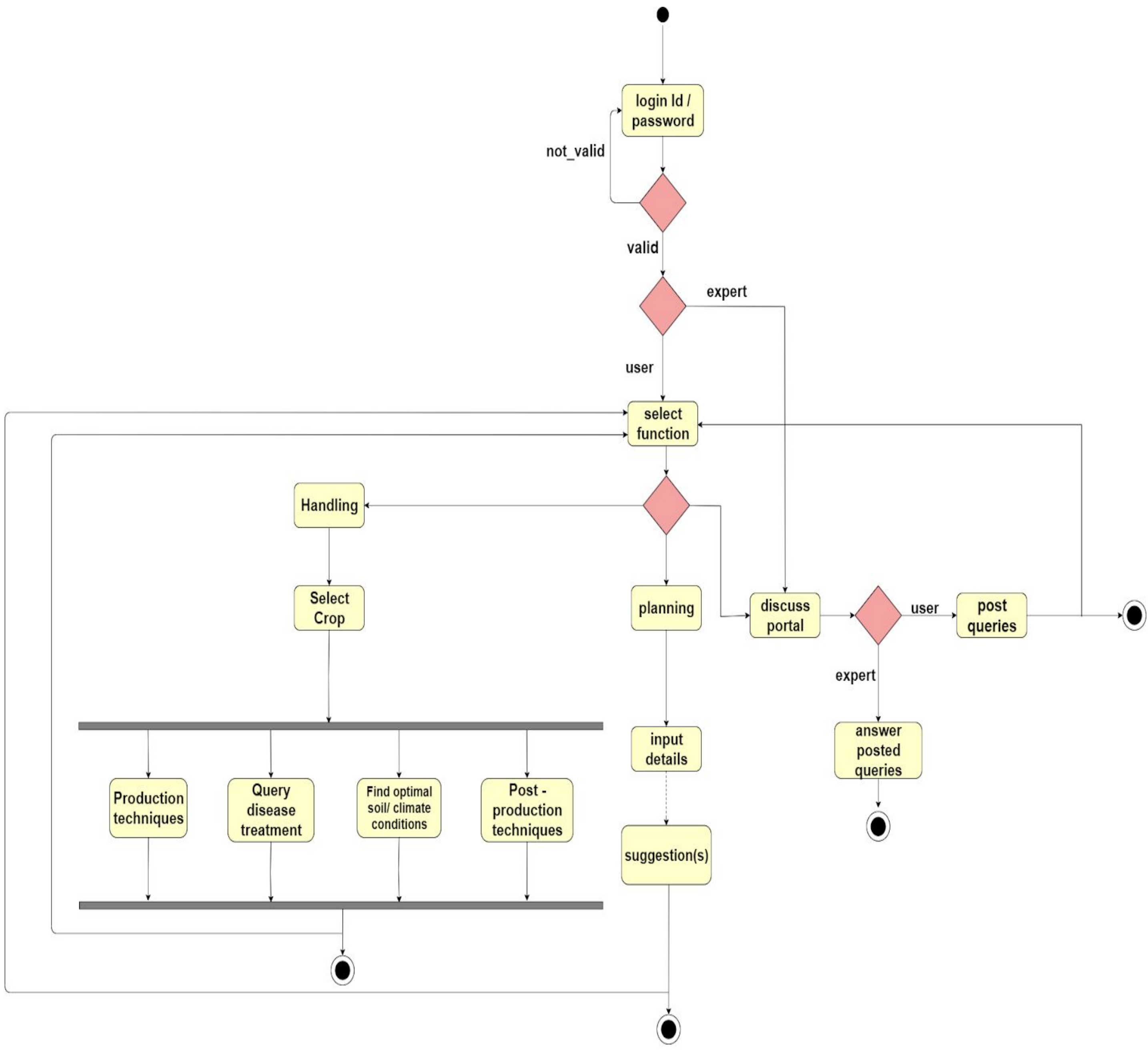
The dynamics of the above static components in the class diagram are described by the diagrams below.

<u>Sequence diagram</u>:

The sequence diagram is quite large to fit in a document page, please click on the link given in the next line to see the dynamics of static components.

The sequence diagram for a user as a farmer is at the drive link.
The sequence diagram for a user as an expert is at the drive link.

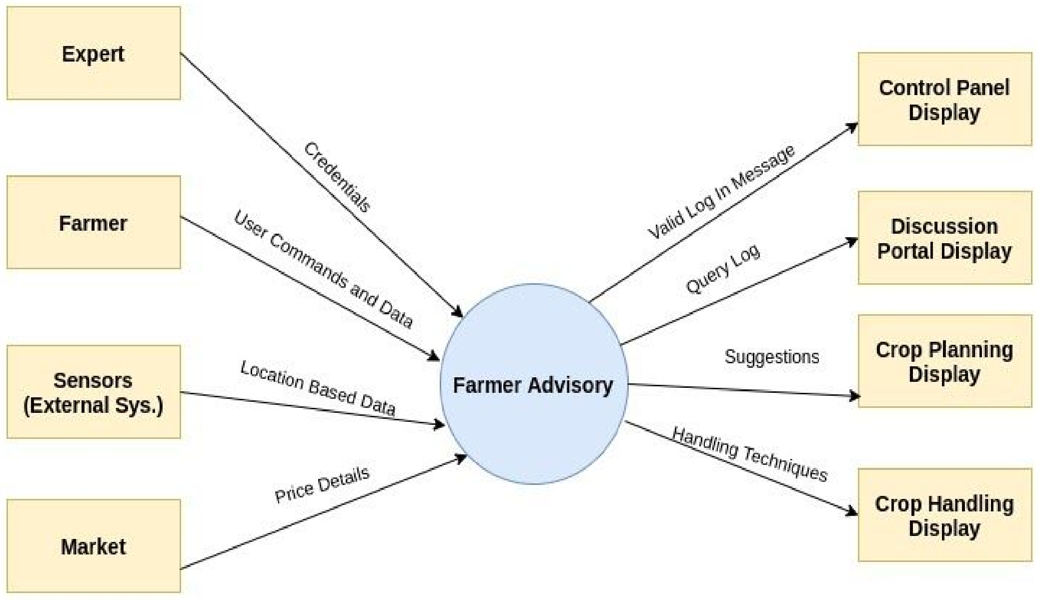
Activity Diagram:



* Non registered users can also see the crop handing techniques.

5.2. Context Diagram

The context diagram of the system is represented by a Data Flow Diagram of Level 0 i.e. showing the input and output data objects at the system level.
The diagram for our Farming Advisory Portal is as below:



Handling Techniques = Production Techniques + Disease Treatment + Optimal Soil Climate Conditions + Post-Production Techniques

Query log = posted_by + post_time + query + replied_by + reply_time

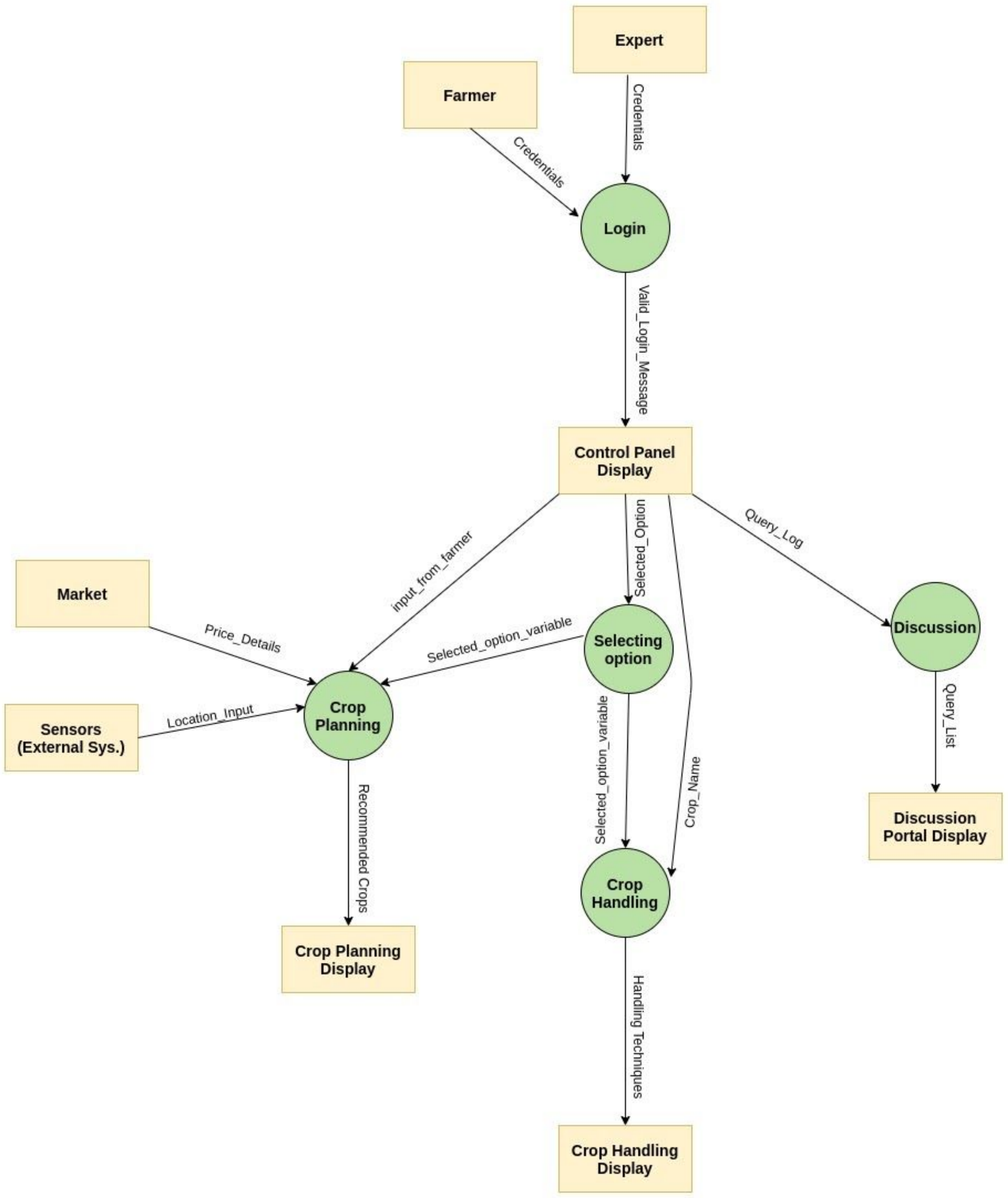
Suggestions = string denoting which crop(s) should be grown

All the Primitive data is in string

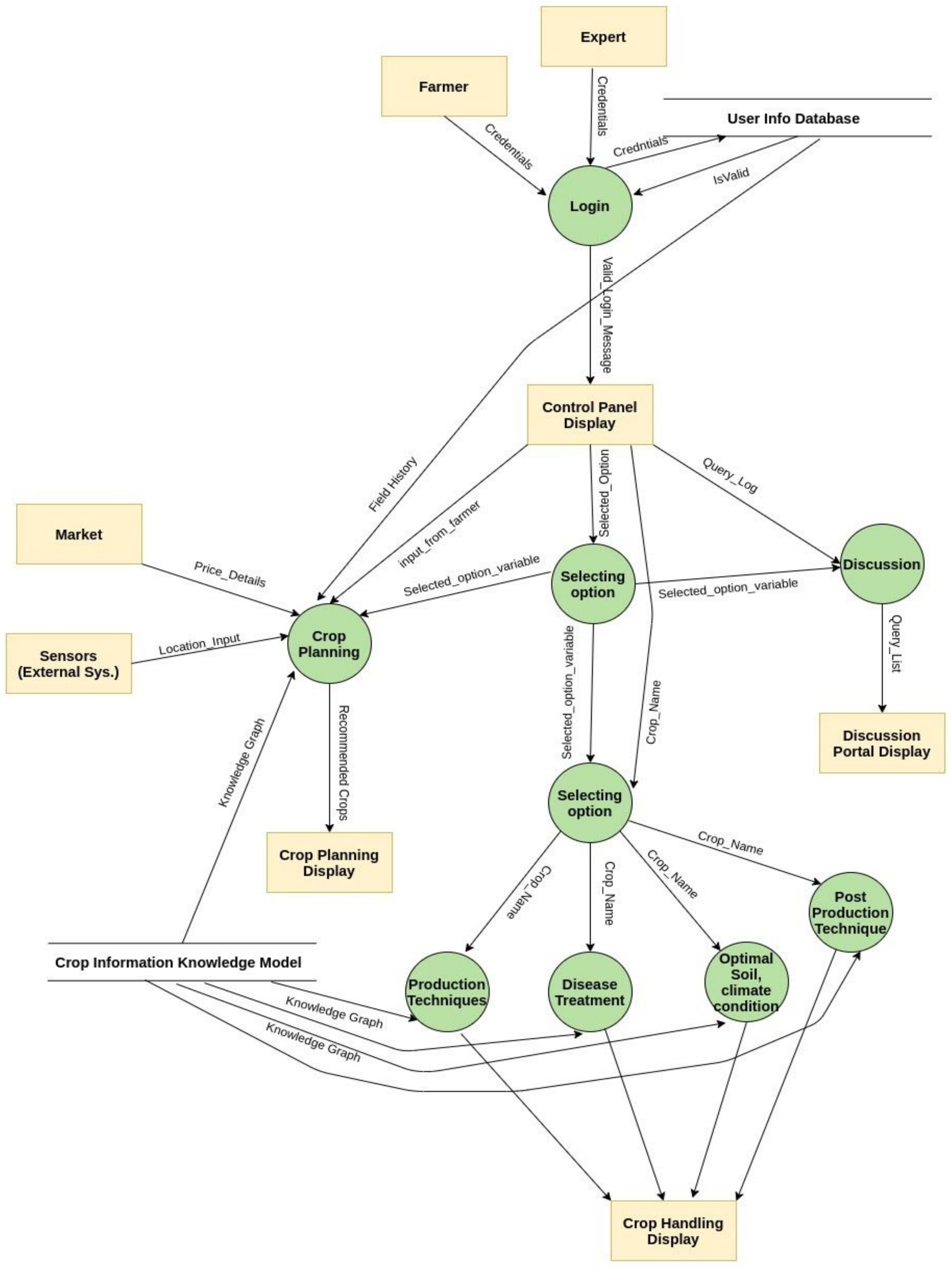
5.3. Data Flow Diagrams Upto Use Case Level

The DFD (Data Flow Diagram) at Level 0 is described by the context diagram shown in Section 5.2 above.
The DFDs at Level 1 and Level 2 (the Use Case Level) for the Farming Advisory

Portals are shown below.



DFD Level 1



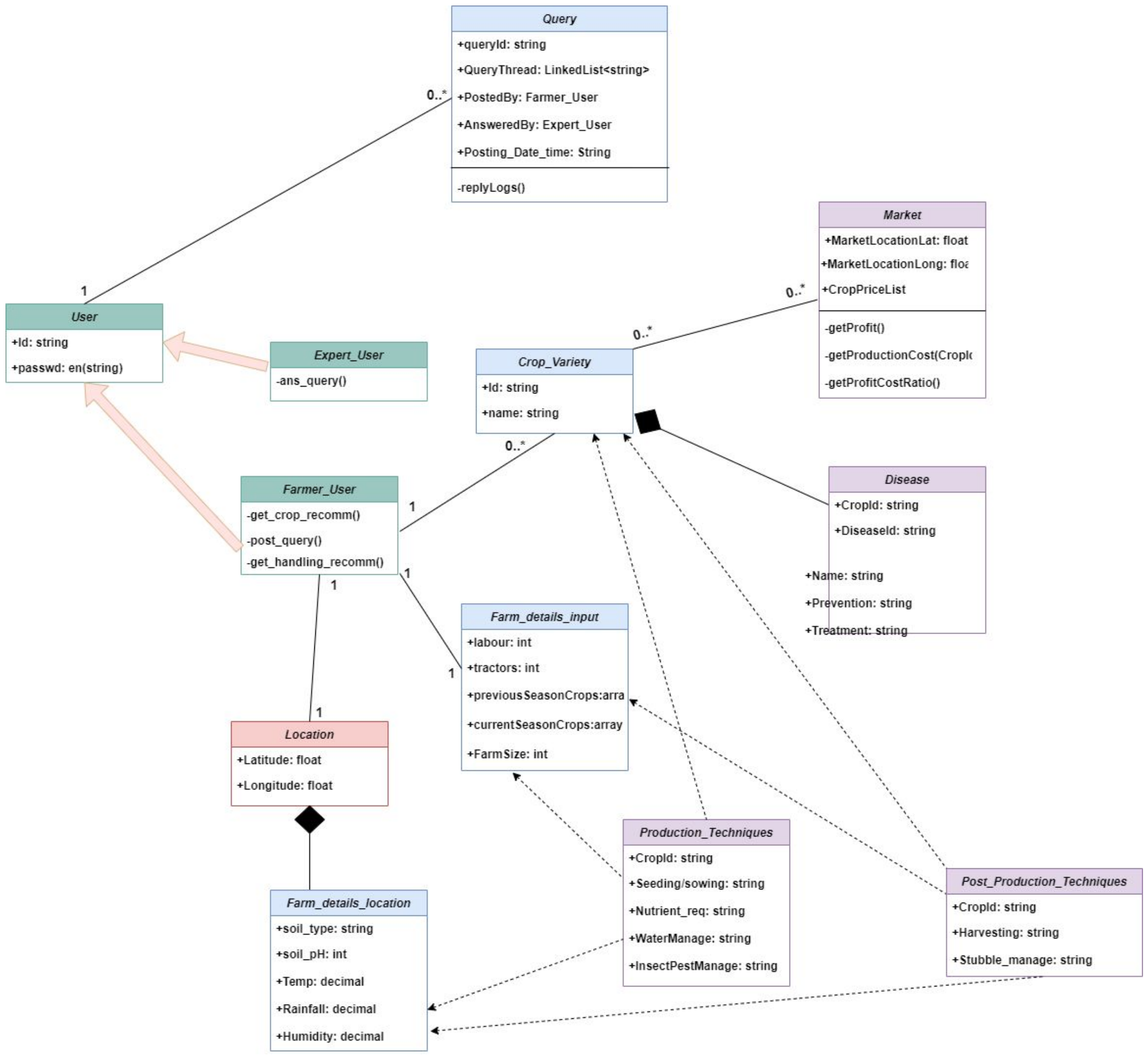
DFD Level 2 (Use Case Level)

6. Database Architecture

6.1. Data Model (Class Diagram)

Class Diagram represents the static view of an application describing the attributes and operations of a class and also the constraints imposed on the system.

The class diagram for Farming Advisory Portal is as shown below.



6.2. Database Deployment

The Database for the portal is to be maintained in three different formats for Different Use Cases.

- For User Login
  The user login system requires a database of registered user credentials to be maintained which is done in a relational database through MySQL.
- For Knowledge Models
  The Knowledge Models built from Research Literature in Agriculture and which is used for answering queries is to be maintained in a Graph Database which is maintained in Neo4j.
- For Discussion Portal
  The Query Database is maintained in MongoDB.

7. Discussion of Design Decisions

1. Database

Options considered : Relational database (MySql), schema-less database (mongoDb), Graph database (Neo4j)

MySql :

Pros:

- ❏ For structured data (as a relational database with a predefined scheme)
- ❏ Supports transaction more easily
- ❏ Fits complex queries better

Cons:

- ❏ Not generally suitable for unstructured data requires tables with well-defined columns or attributes, means cannot be used in the task where we don't know the predefined schema of the table.
- ❏ Maximum size of the string or VARCHAR is 255 characters

mongoDb :

Pros:

- ❏ Useful for unstructured data
- ❏ Easier to scale
- ❏ The more clear object structure

❏ Support larger size for strings

Cons:

❏ It might be slower than the corresponding relation based design.

Graph database(Neo4j) :

Pros:

❏ Easier to apply the object-oriented way of programming and thinking while building the software. Easy to implement multidimensionality

❏ Data captured can be easily changed and extended for additional attributes and objects

❏ Graph databases are naturally indexed by relationships, hence provides speedy data retrieval for connected data.

Cons:

❏ not good at handling queries that span the entire database.

❏ have to store all the data on *one* server.

Which database we will use in our project and why?

We will use all the three databases we mentioned above. Reason for using all the three databases mentioned is the following:

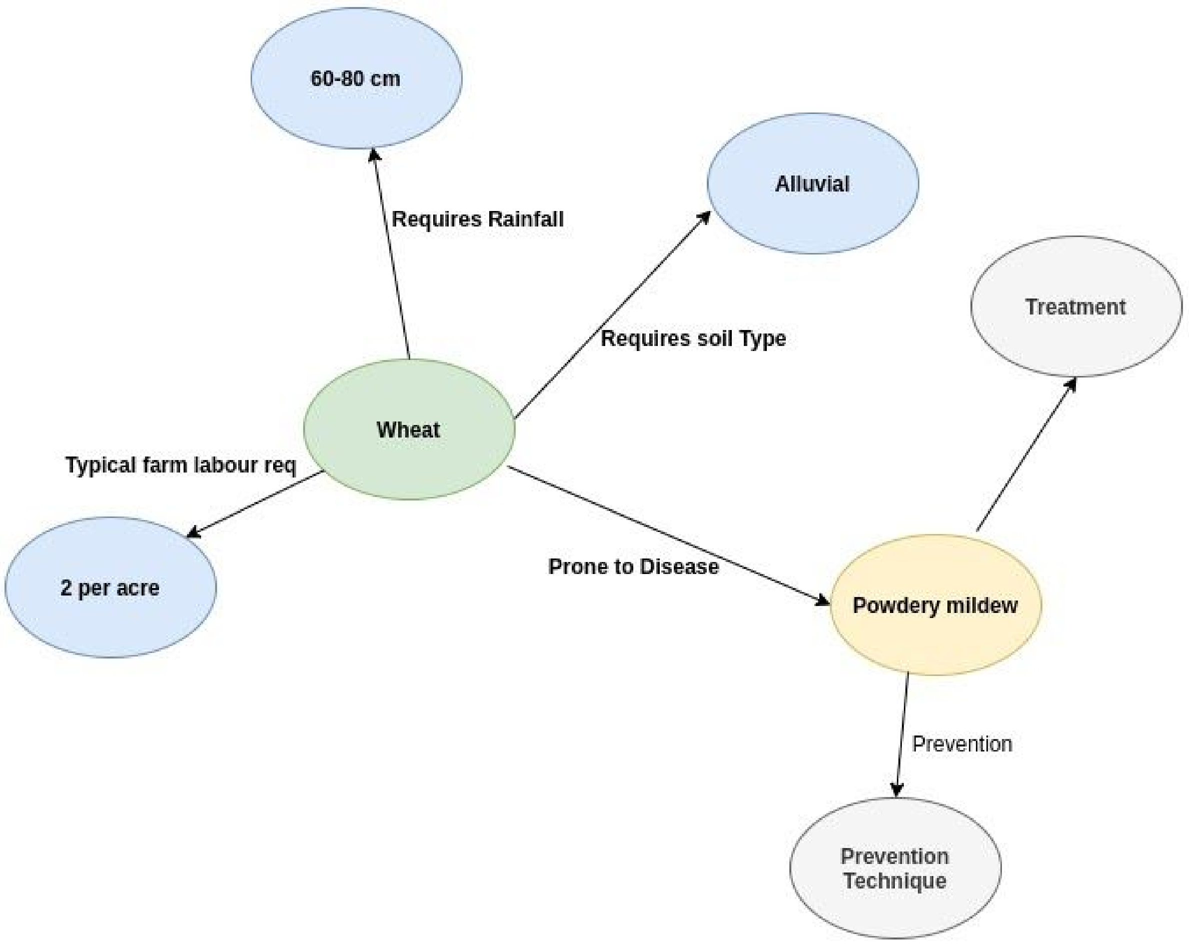
- In this project, we planned to make a discussion portal where a farmer can enter his/her queries (queries are entered in the text format as a general question which farmer wants to ask to the experts of the agriculture domain) which are to be answered by the experts. Here for any particular query, we want to bind all the replies of query with it. Here we don't know the size of replies and queries text. Hence we exploit the property of **mongoDB** that the maximum size of a reply/query is not limited as in case of MySql (string size is at max 255 characters).
- We planned to make a login portal also. (Why we planned to make a login portal is another design choice which is explained in this section as well -> explained in design issue 3.). To store login credentials(userID, password, IsUserAnExpert/Farmer) of the user, we have a predefined schema. When a user enters userID and password and if a user is valid(userID, password matches) when he/she is logged in as an expert or a farmer. Similarly to store farm details(which helps in better crop recommendations) of a farmer we have a predefined schema (farm details includes FarmSize, LabourAvailable, EquipmentsAvailable, PreviousSeasonCrops,

CurrentSeasonCrops). As MySql supports queries when we have a structured data or predefined schema, we will use **MySql** for this purpose.

- In our system, we will be using **Knowledge Graphs** for maintaining the database on agricultural research for answering the farmer queries on crop planning and crop handling. We use a graph database because Database teams can implement multidimensionality with a relatively ease. To store crop planning and handling related information software developers can easily implement object oriented thinking.

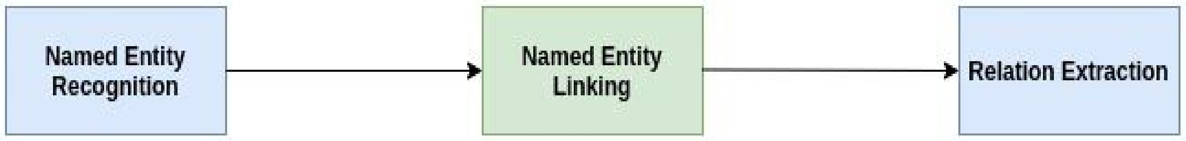
2. Building the Knowledge Graph

Here the main issue is to discuss the technique for extracting structured data relevant for the Knowledge Graph creation. These techniques depend on the source of data. We aim to build knowledge graphs similar to the example shown below for wheat from the unstructured natural language data.



We consider two alternatives to choose from which have been described below:

- **Required Knowledge Extraction with Entity Identification:**



❑ **Named Entity Recognition:**

In this step, we parse the text and perform **POS (parts of speech) tagging** by finding the nouns, adjectives, determiners etc in each sentence. The output of this parsing after identifying all the nouns in the data is to classify the identified data into nouns useful to us which mainly are:

- Crop names
- Environment Variables such as temperature
- Soil Type
- Crop Disease Names
- Location Names

For Noun classification:

A supervised classifier to be trained on agricultural entities using the dataset available from agricultural books (such as a list of crop variety names). This task will be performed with NLTK tokenizer or LSTM classification whichever yields better results.

❑ **Named Entity Linking:**

Once we have obtained the the agricultural entities, we need to map them to a knowledge base. We intend to map the entities to in vector space using **wordnet and word2vec** mapping models.

The aim of this process is to disambiguate any ambiguous entities identified in the Named Entity Recognition Task. For example, 'Wheat' and 'crop' might refer to ' Wheat' entity only. So it would be helpful to consider them as the same object being considered in a relation.

- ❑ **Relation Extraction:**

We intend to identify the relations such as crop_entity_a ***'requires'*** climate_entity_b or crop_entity_a ***'prone to'*** disease_entity_b. We will be using ACE task dataset to train a classifier to find parts of sentence reflecting relations similar to given input relation.

- **Web scraping:**

The aim is to use structured data obtained from web scraping (such as Google Search and Pre-processing). This structured data is then used to build a knowledge graph for various entities reflected in the search query. For example, in order to identify the soil type requirement of rice crop, we can use google search to get relevant documents and then use these documents to extract intended search output and add to the knowledge graph.

Pros:

- ❑ Structured Data: Data extracted through scaping is relatively more structured and precise than using native unstructured data source. This is the biggest point in favour web scraping techniques making knowledge graph from relatively structured data is easy.

Cons:

- ❑ It fails for many complex websites. So for reliability we have to select some specific websites to scrape data from.
- ❑ Difficult to maintain, if the web service updates the html page rendering way, then we need to update our backend code to get updated information.

Which alternative we select for Knowledge Graph creation?

We will first try our best to create the Knowledge Graph from Natural Language Processing techniques (based on Entity identification and relationing the entities) on unstructured text from authentic agricultural books, Government jonourals and farming practice manuals. In case the project deadline is on the verge and nothing works in favour of NLP on unstructured text, we will keep the web scraping and indexing techniques as our backup plan for the agile product. But for sure our first preference is to create a Knowledge graph for unstructured authorized text by entity identification and relationing the entities.

3. Making a login portal or not

We prefer to make a login portal for the system. Reasons for that is the following:

1. We want to increase the user experience. If the farmer has a login id and password, he/she only has to enter his/her farm details(farm size, labour availability, equipment available, previous crop sown the field) once. When he/she logged in next time he/she doesn't have to fill the same data again which reduces his/her overhead to input farm details every time he logged in. Obviously he had a choice to change his input.
2. With a login id and password, it will be easy to distinguish between farmer and expert i.e. who can post a query on the discussion portal and who can reply to it.
3. People who only want to know information about crop handling(inc. crop disease treatment, market price, production cost) can also view the crop handling section of the website without login. But only the logged in users can post tet queries on the discussion portal to interact with the expert.
4. We will provide buttons for all the options available for logged in users at the menu bar, but when a user clicks on that he/she has to sign in. This will attract more users to make a signup(account) on the website, as they can utilize more features like crop planning and other agricultural recommendations based on their farm details means they can get more prioritized results.

4. Architectural Design: UI Based Query System V/s Text Query Based System

Options Considered: UI Based Query System and Text Query Based System

UI Based Query System :

The user, in this case, is provided with the queries based on the options to be selected from the portal. For example, if the user intends to know the requirements for a crop, the user selects the crop and gives the required input parameters such as field size and previous crops.

Pros:

- ❏ Easier to use for the majority of the users who might not be able to put their questions in standard forms.

Cons:

- ❏ It requires a more precise database to be maintained which may sometimes be difficult to obtain.
- ❏ Knowledge Model Graphs need to be used for faster data access.
- ❏ NLP techniques to be used for building structured graphs out of unstructured natural language data.

Text Query Based System :

The user, in this case, is provided with inputs to get questions based on farming queries in natural language sentences. All the queries are made in a text search format and the responses to the queries are made by providing the relevant data from the Agricultural Science literature and research Database maintained in the system.

Pros:

- ❏ Easier to implement Natural Language Processing (NLP) Techniques as the task is to find a similarity between the question asked and the relevant articles or research information available.

Cons:

- ❏ It might not always give the user precise information to his/her queries.
- ❏ The search process might get slower due to the multiple steps of asking a question and then skimming through various research outputs provided by the system.

Which query system we will use in our project and why?

We planned to use a **hybrid approach** in which the majority of the time, users can get replies/responses for the queries through a precise GUI based system. But users(famers) can also post queries on which the replies are given by experts. We will ensure that most of the common questions such as those on crop planning and crop handling are provided through a precise system based on GUI. Except the discussion portal where farmers can post queries and experts reply to their queries, all other recommendations and suggestions regarding crop planning and handing and driven by **GUI based query system**.

5. Python based Framework

Python is a high level programming language which is used to express concepts without writing lengthy code. Python is the most popular language to write web applications with a clean, readable and maintainable code base. There are multiple frameworks which are written in Python. But we will prefer Django.

Why we prefer Django Framework :

- **Django Supports MVC patterns:** MVC patterns simplify and speed up the development process of the complex web applications by keeping their business logic and user interface layer separated. By the use of Django, developers don't have to concentrate on the interaction between the model and the view. They just have to map them (model ,view and template) to a specific URL.
- **Strong and Robust Security features:** This framework enhances the security by preventing various types of security attacks like SQL injection, cross-site scripting.
- **Highly Scalable and Customizable:** Django is much more scalable and flexible than any other framework written in Python. Developers can easily customize the entire web application just by making modifications to a specific component.
- **Exclusive built in template System:** Built template System of Django provides Developers to keep the code base maintainable by keeping HTML and Python code separated.

6. Apache V/S NGINX Server

Options Considered: Apache and NGINX

Apache

Pros:

- Provides a variety of multi-processing modules(MPMs) to handle clients' requests. MPM provides flexible Architecture for choosing different connections and request handling algorithms.
- Uses a process driven approach and creates a new thread for each request. Apache handles both static and dynamic content without relying on external processors.
- Larger Community Support.

Cons:

- Consumes More RAM under heavier load.

- ❏ Spawns a new process for request which results in less efficient architecture.

NGINX

Pros:

- ❏ More lightweight hence it requires fewer resources or memory.
- ❏ Designed to use an asynchronous, non-blocking, event-driven architecture to handle multiple requests in a single thread.
- ❏ NGINX’s design architecture is much faster when it comes to serving static content.

Cons:

- ❏ NGINX depends on external processors for handling dynamic content.
- ❏ It was developed after Apache hence it has less developer community support.

Which server we will use in our project and why?

We are using **NGINX** because of its quicker interpretation and response. We are building our application with Django(or actually Python) and because of it we shall never be able to use a lot of features provided by Apache and if we would go for Apache then it has to pay with considerably higher memory usage and lower performance, hence We prefer NGINX.

7. Communication Protocols

Options Considered : HTTP(S) and Websockets

HTTP(S) :

HTTP or HyperText Transfer Protocol, is the protocol used by the World Wide Web. It defines what actions browsers and servers should take in response to certain messages or commands. HTTPS is the more secure version of the HTTP. It means all the connections between the browser and website are encrypted.

Pros**:**

- ❏ Security and encryption (Only browser and server can decrypt the communicated information).
- ❏ Server Authentication means HTTPS gives an opportunity to the client to leave before feeding any person data to invalided server.

Cons:

- ❏ Communication speed between server and client can decrease because of complex encryption and decryption.
- ❏ Request and Response overhead in real time communication.

WebSockets :

The idea behind socket is that it is a port through which data flows in and out of. WebSockets are just an extension of socket idea. Websocket allowed people to use websocket protocol ,which is very flexible for transferring data to and from servers from browsers.

Pros:

- ❏ Full-duplex asynchronous messaging means both(client and server) can stream messages to each other independently.
- ❏ Low latency communication for client to server messages.
- ❏ Good security model.
- ❏ Websockets can pass through most firewalls without any reconfiguration.

Cons:

- ❏ WebSockets keeps the connection open on the server for the duration of the time the user is interacting with the page. This will increase the demand on the server , and means We shall always have to scale out rather than up.
- ❏ WebSockets don’t automatically recover when connections are terminated.

Which communication protocol we will use in our project and why?

We will use the hybrid approach because we have considered the option of query portal where farmer and expert would communicate. Hence, in the cases of real time communication (communication between farmer and expert), WebSocket protocol is much better than HTTPS because of request and response overhead in HTTPS. To handle real time communication We will use a

real time push server (**tornado**) because of its high performance. Tornado is a python base framework which gives built in support for user authen
HTTPS protocol will only be used for communication between browser and web server because of its secure configuration.

8. Degree of Separation Between Content and Presentation

Options Considered :

- All of the data to be presented as well as the presentation code kept together in a module based on HTML/CSS or JavaScript for the use of presentation thus combining the presentation and content layer. The method is easier to implement in code but difficult to maintain as a programming technique and also makes the software less modular.
- The other option is to consider a content layer based on Python or any Database and then allow the presentation layer (based on CSS) to access only the required data from the Content layer.

Which option we have considered?

We will go with the second option for making the software more modular and hence maintainable.

9. Visual Hierarchy v/s Availability of All Options on UI

Options Considered :

- A visual hierarchy showing all the steps the user has taken to enter a particular part of the portal.
- A portal showing all the available options on the portal the user can take in the current screen.

Which option we have considered?

We will go with the second option, as the options provided in our portal are not in any way hierarchical and hence can be used by the user while on any given screen.

10. Handling Data Exceptions

In case there is a data exception such as empty input or compulsory option not selected, following two options can be considered.

Options Considered:

Reloading the page and asking for new input.

Pros:

- ❏ More secure.
- ❏ Easier to implement.

Cons:

- ❏ Overhead for users to fill out the inputs again.

Asking to fill out empty inputs only.

Pros:

- ❏ Less overhead for users.

Cons:

- ❏ Less secure.

References:

- https://readthedocs.org/projects/django-tornado-websockets/downloads/pdf/latest/
- Design Document Template: https://www.sampletemplates.com/business-templates/design-document.html