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:

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

2. Design Overview

2.1. Architectural Goals and Constraints

Goals

- 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

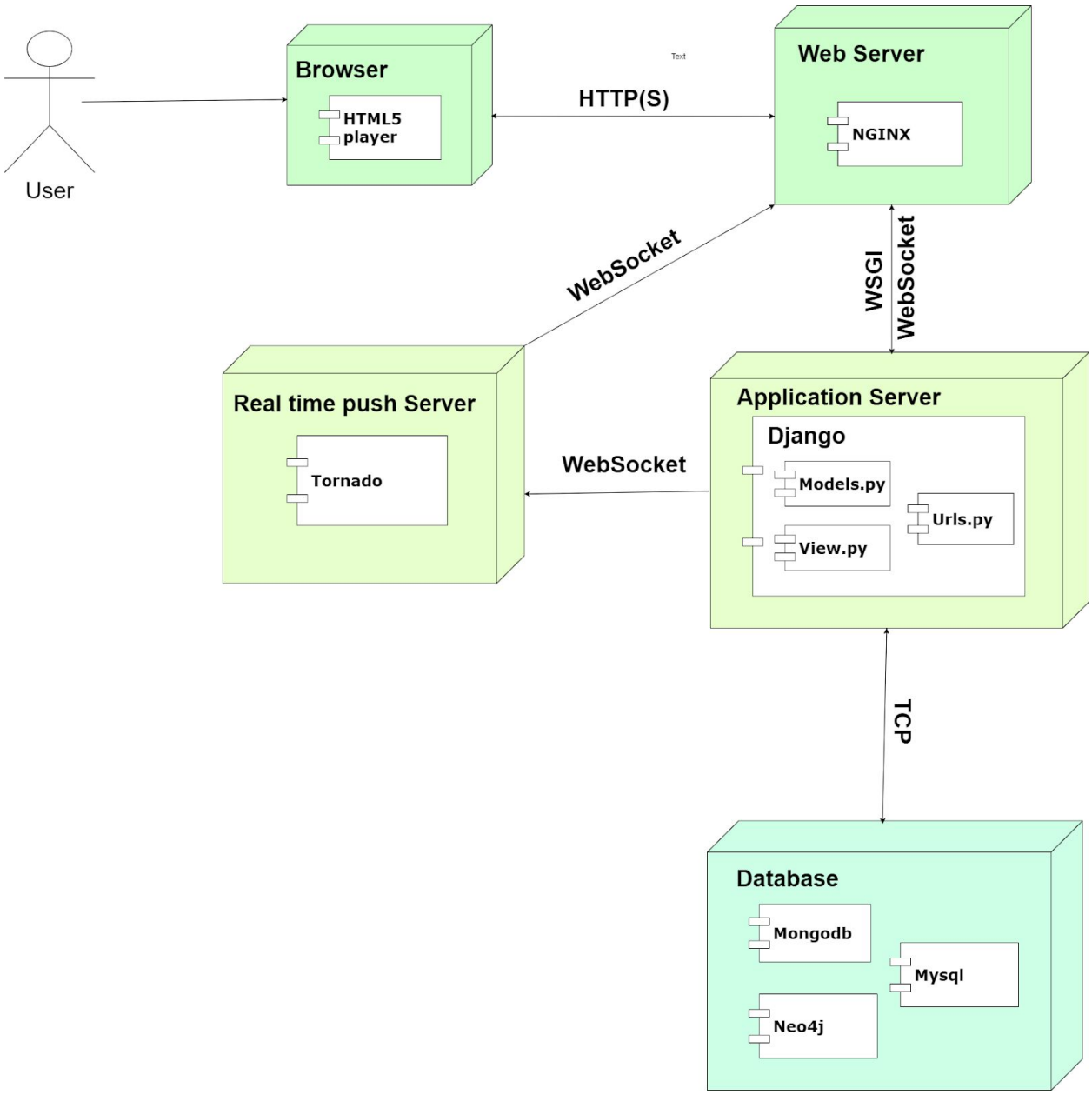
Scalability is the ability of the platform to scale both up and down to support varying numbers of users or transaction volumes. The application should be able to scale horizontally (by adding more servers) or vertically (by increasing hardware capacity or software efficiency).

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: Uses database application to handle tasks such as data analysis and storage.

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 otherwise 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.

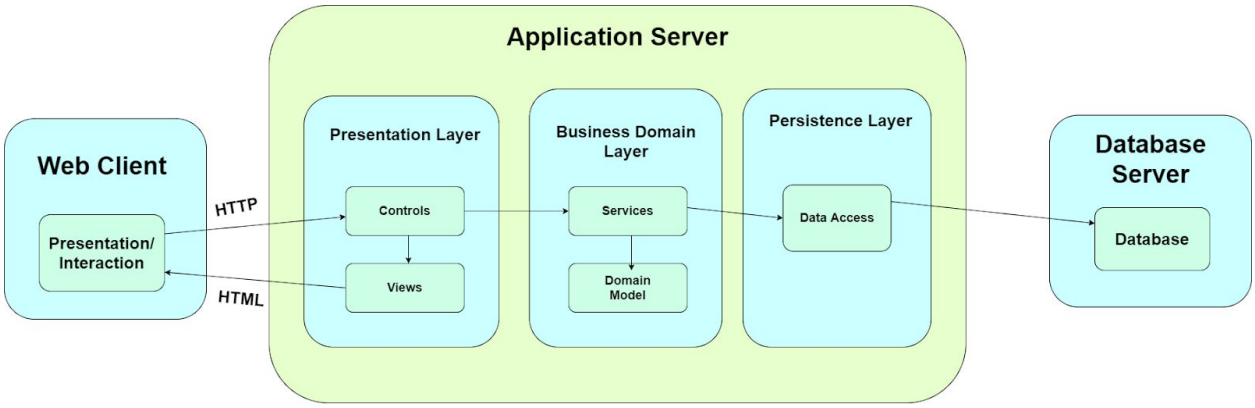
4.3. Business Layer

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 of 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

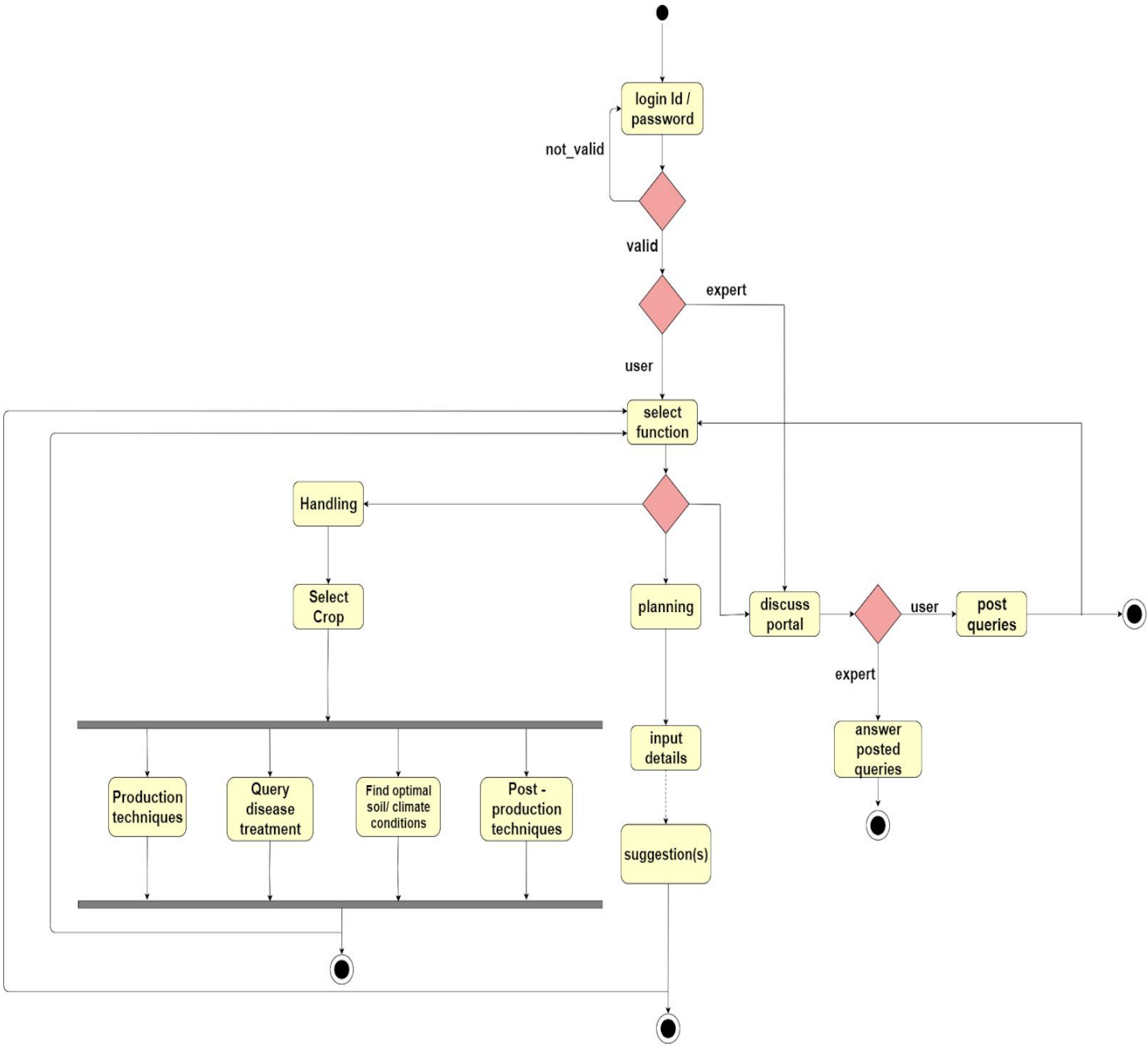
The UML diagrams representing the dynamics of static structure are as follows.

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

Sequence diagram:

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.

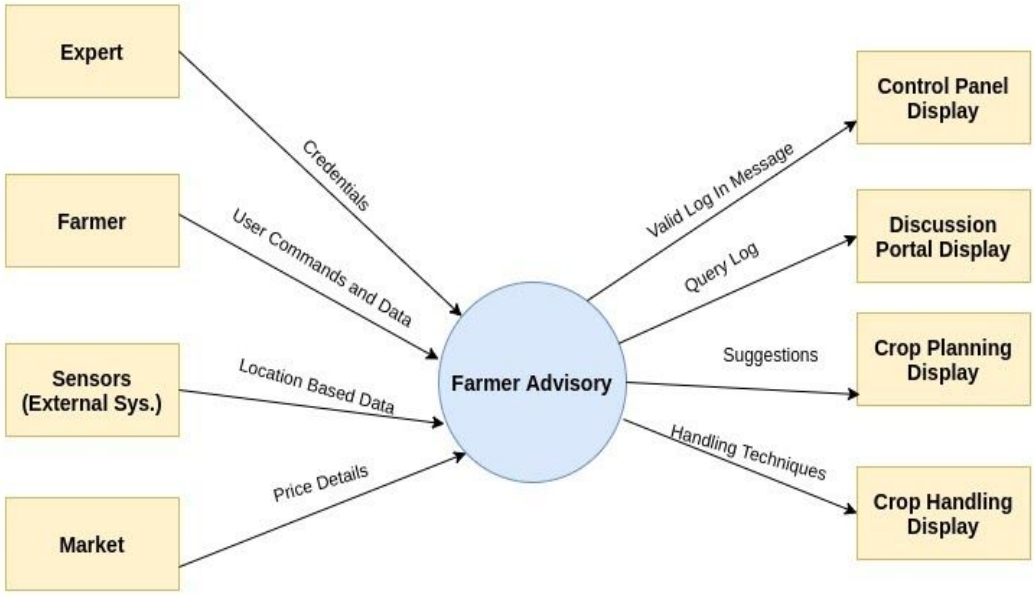
Activity Diagram:



* 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

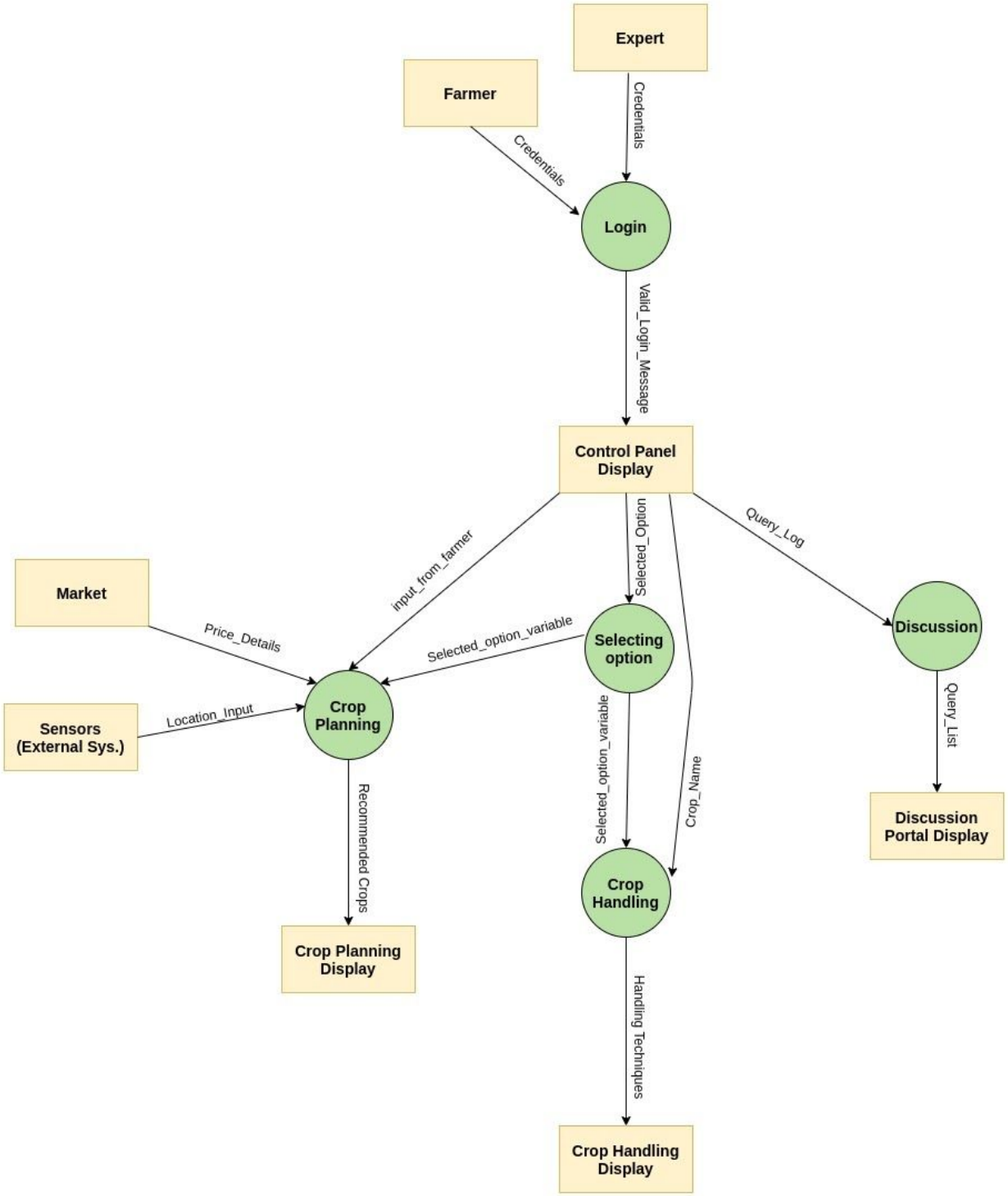
Suggestions = string denoting which crop(s) should be grown

All the Primitive data is in string

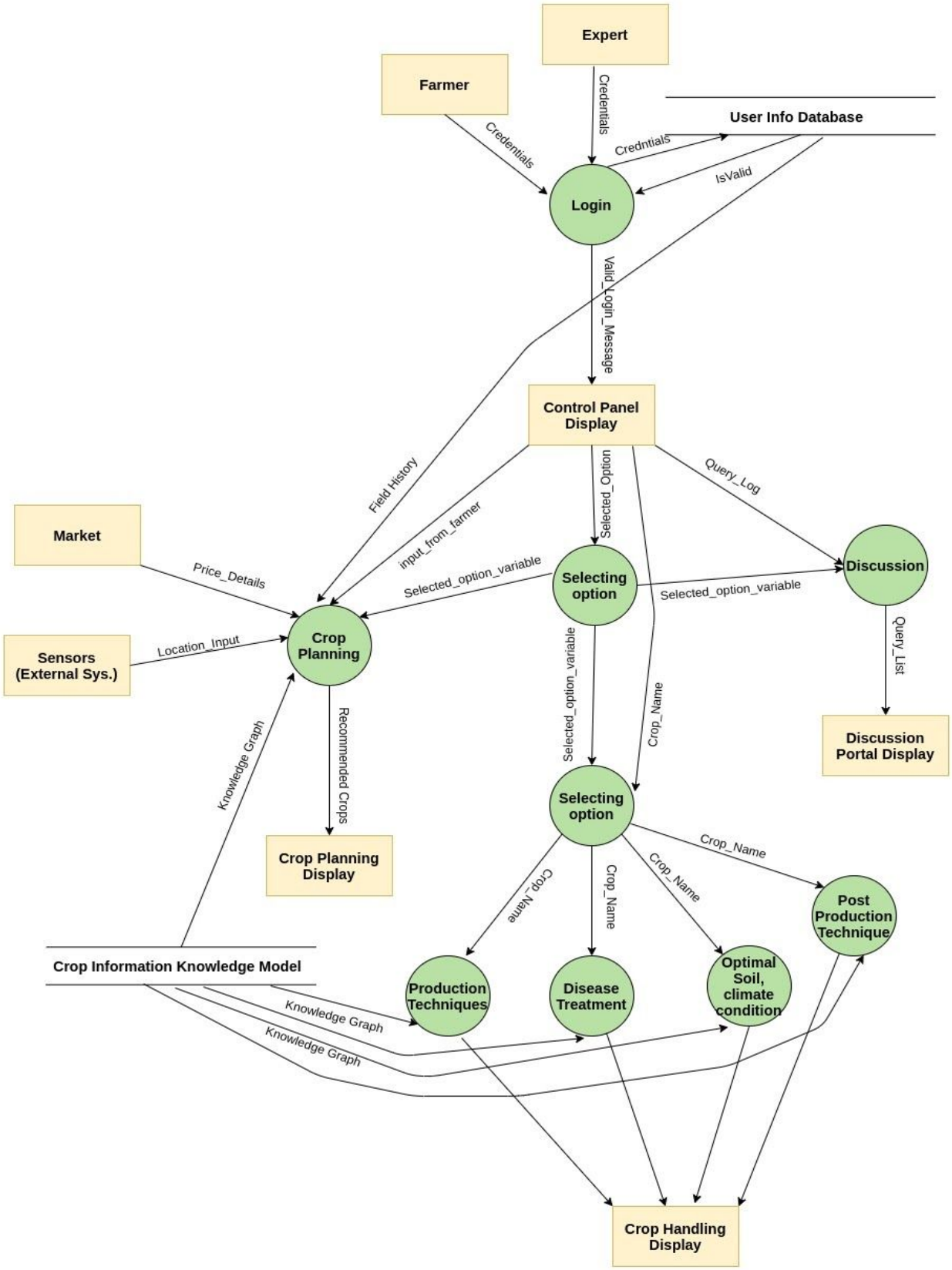
5.3. Data Flow Diagrams Upto Use Case Level

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



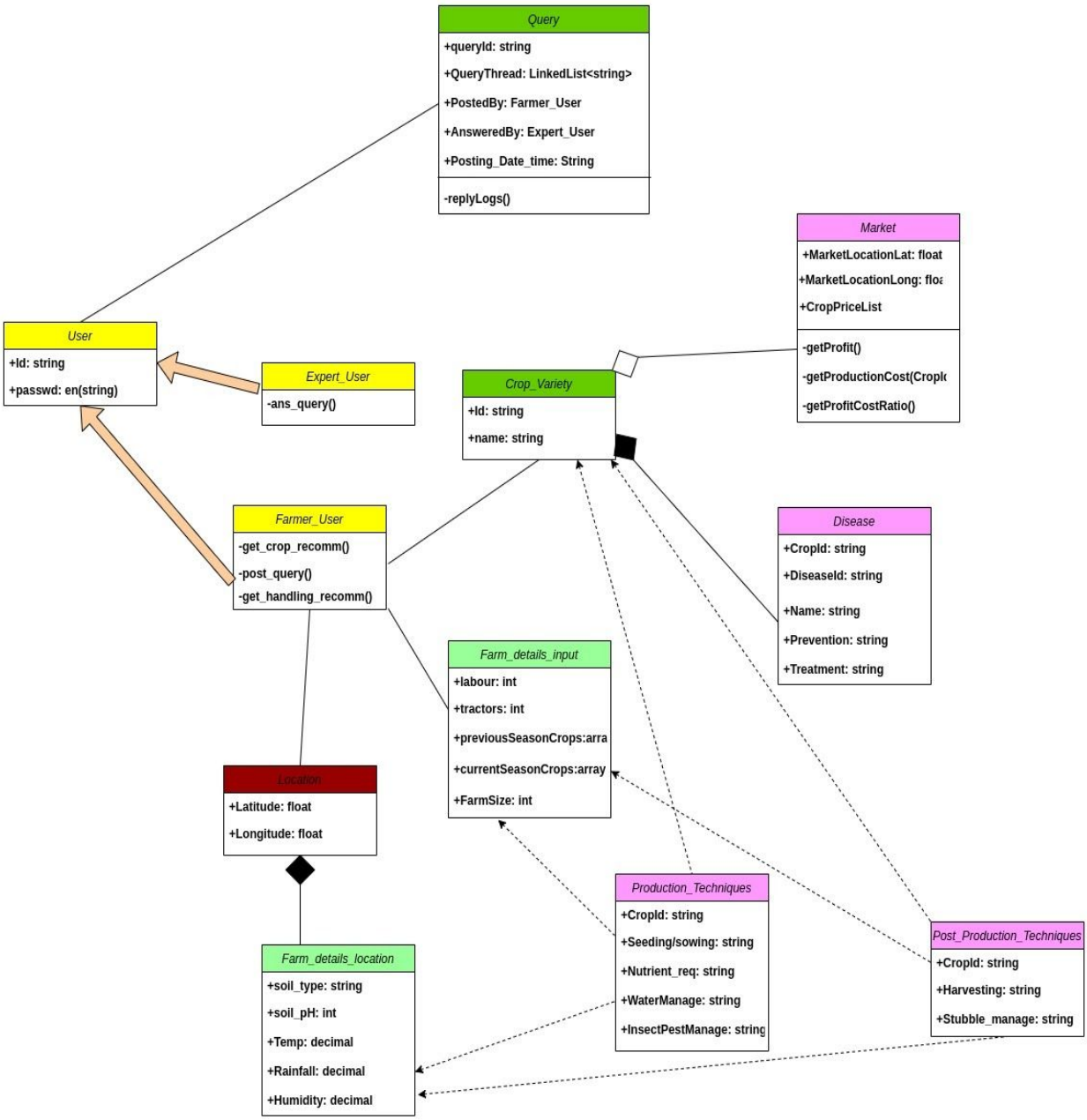
DFD Level 2 (Use Case Level)

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

- Architectural Design: UI Based Query System V/s Text Query Based System

  Options Considered:

  1. 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.
  2. 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.

Since both of the above approaches have their pros and cons, we chose a hybrid approach in which the majority of the times, user can get replies/responses for the queries through a precise GUI based system but is also provided with a system to post queries on which the replies are given by some expert users. We 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.

- <u>Database of the System</u>

  Options Considered:

  - Graph Database (Knowledge Graph)

    Pros:

    - Easier to apply the object-oriented way of programming and thinking while building the software.
    - Database teams can add to the existing graph structure without endangering current functionality.

    Cons:

    - Difficult to build a large Knowledge Graph from unstructured data as text.
  - MongoDB (No SQL (Non Relational) Database)

    Pros:

    - Useful for unstructured data
    - Easier to scale
    - The more clear object structure
    - Easier to make reply threads for a particular query

        - Schema-less

      Cons:

        - It might be slower than the corresponding relation based design.
    - MySQL (Relational Database)

      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

  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.
  The discussion portal database (maintaining query_log : posted_by, post_time, query, replied_by, reply_time) will be maintained through MongoDB.
  Relational Database is to be used for storing user credentials having predefined credential field names.

- <u>Apache V/S NGINX Server</u>

  Options Considered:

    - Apache
    - NGINX

  Apache uses a process driven approach and creates a new thread for each request, but NGINX uses event driven architecture to handle multiple requests in a single thread.

  Apache consumes more memory, because of more threads. Hence we prefer NGINX over Apache as, NGINX is faster.

- <u>Degree of Separation Between Content and Presentation</u>

  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.

We chose to go with the second option considered for making the software more modular and hence maintainable.

- 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.

  We chose to 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.

- 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