Low Level Design

Soil Farming Agent

Written By	Naman Malik, Tanmay Sharma
Document Version	1.2
Last Revised Date	27–July-2023

Document Control

Change Record:

Version	Date	Author	Comments
1.0	25 – July - 2023	Naman Malik	Introduction & Architecture defined
1.1	26 – July - 2023	Naman Malik	Architecture & Architecture Description appended and updated
1.2	27 – July - 2023	Tanmay Sharma	Unit Test Cases defined and appended

Approval Status:

Version Comments

Review Date	Reviewed By	Approved By

Content

1. Introduction

2. Architecture

3. Architecture Description

4. Unit Test Cases

1. Introduction

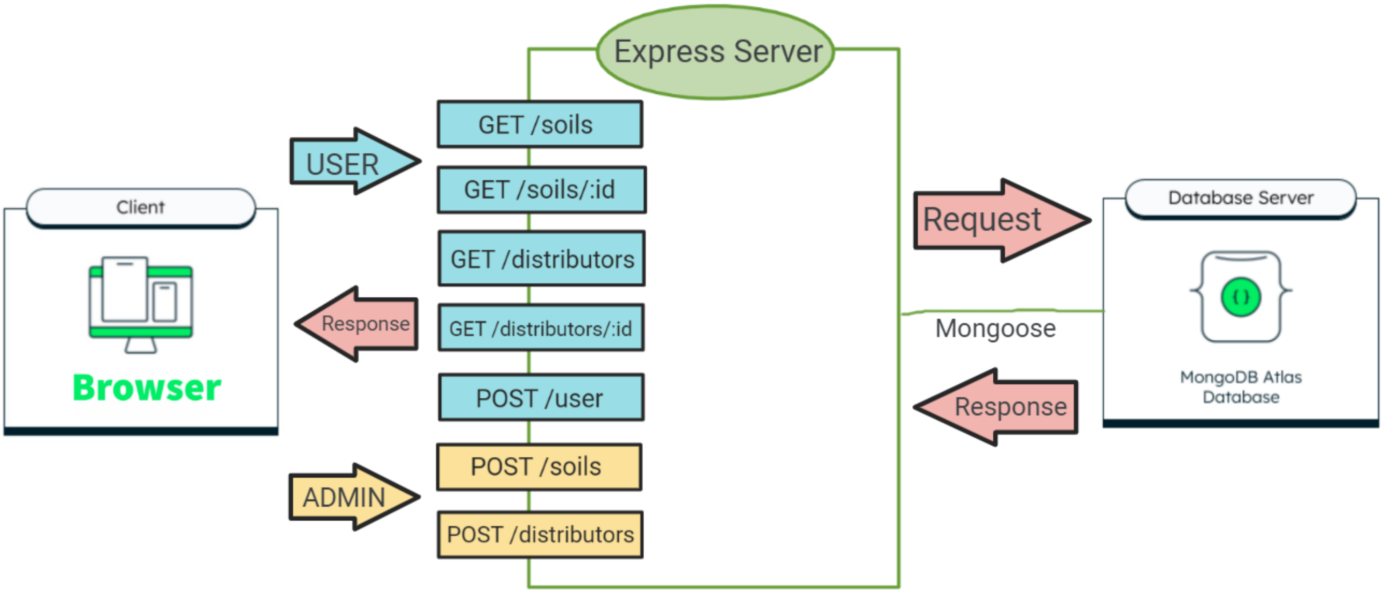
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Soil Farming Agent. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture



2.1 Components of our Architecture?

The Soil Farming Agent is a modern web-based platform designed to provide comprehensive soil information and connect soil distributors with agricultural stakeholders. It is built using technologies such as Node.js, Express.js, and MongoDB.

1) Node.js is used as a Server-Side programming language to provide services to HTML, CSS and JavaScript Files.
2) Express.js is used to develop an Interface between the Database and Frontend.
3) MongoDB is a N0-SQL Database that we are using in our project.

3. Architecture Description

3.1 Frontend

The frontend of the Soil Farming Agent project is developed using HTML, CSS, and JavaScript. The User Interface (UI) is designed to be intuitive and user friendly, allowing users to easily access and interact with soil-related data and distributors listings. The Frontend Communicates with the backend through RESTful APIs to fetch data and update the user interface in real-time.

3.2 Backend

The backend of Soil farming Agent is built using Node.js and Express.js. **Node.js** serves as the runtime environment, allowing the server-side code to be written in JavaScript. **Express.js** provides a robust framework for Handling HTTP requests, routing, and middleware.

3.3 RESTful APIs

The backend exposes RESTful APIs that enable communication between the frontend and the server. These APIs handle requests for user authentication, Soil and distributor update.

3.4 User Authentication

The project incorporates a user authentication system to secure data and personalize user experiences. User can register, log in, and access personalized profiles. This authentication mechanism helps us to protect our system from Unauthorized calls such as ADD Soil, Add Distributor and

Update Profile.

This system is possible throughout the help of **Json Web Token** and the **cookies**.

3.5 Middleware

Middleware plays a crucial role in the backend architecture. Express.js middleware functions are used to process incoming requests before they reach the appropriate route handler. Middleware functions can perform various tasks, such as authentication, logging, data validation, error handling, and more. They help modularize the backend codebase and enhance the overall maintainability and scalability of the application. Middleware functions can be applied globally to all routes or specific to certain routes, depending on the application's requirements.

In This Project, Two Middleware are used:

1) Auth: This middleware is used to authorize our users or admins to gain access of some specific routes. Auth contains two middleware functions auth.admin and auth.user .In These functions, Json Web Tokens are verified.
2) FileStorage: This middleware is used to upload an Image from the Browser to the Server Using a Post request with encryption of multipart/form-data. It includes 3 separate middleware functions to upload Profile, Soil and Distributor in 3 different paths. In these functions, multer module is used.

3.6 MongoDB Database

The Soil farming Agent uses MongoDB as the backend database. It is a NoSQL database that offers flexibility and scalability, making it suitable for handling diverse soil-related data. MongoDB's document-oriented nature allows data to be organized in a JSON-like format, simplifying data retrieval and manipulation. Mongoose is used as a MongoDB Driver in this project.

This project stores data of 3 collections:

1) User : This collection contains records of all the users whether it is an admin or end user. Each record of user contains information such as User_Name, UserID, Email, Password, Age, User_type, ProfilePhoto, etc.
2) Soil: This collection contains records of all the Soils. Each record of Soil contains information such as Soil_Name, Description, Places where it is found, Suitable Cropes, Nature of Soil, Soil_Image etc.
3) Distributor: This collection contains records of all the Distributors. Each record of Distributor contains information such as Distributor_Name, Description, Address of Distributor, Soils sells by him, Distributor_Image etc.

3.7 Data Management

The backend handles data management tasks, such as CRUD operations (Create, Read, Update, Delete), data validation, and data aggregation. The server interacts with MongoDB database to store and retrieve data efficiently. The data management layer ensures the accuracy and integrity of soil information, enhancing the platform’s reliability.

3.8 Data Visualization

In the End Personalized data is visualized in front of the user through help of HBS Template Engine. It renders the html file with soil and distributor details along with user personalized details and his profile.

3.9 Deployment

The project is deployed on a web server, making it accessible to users through the internet. Cloud-based hosting solutions or dedicated servers are used for deployment, ensuring scalability and availability.

4. Unit Test Cases

4.1 Accessiblity Test Cases

Test Case Description	Pre-Requistite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when URL is accessed	1. Application URL is accessible. 2. Application is deployed	The Application should load completely for the user when the URL is accessed

4.2 Authentication Test Cases

Test Case Description	Pre-Requistite	Expected Result
Verify whether the User is able to sign up in the application	1. Application is accessible	The User should be able to register in the application
Verify whether the User is able to successfully login to the application	1. Application is accessible 2. User is signed up to the application	User should be able to successfully login to the application

4.3 User Test Cases

Test Case Description	Pre-Requistite	Expected Result
Verify whether user is able to see soil and distributor details	1. Application is accessible. 2. User is logged in to the application	User should be able to see soil and distributor details.
Verify whether user is able to see and update his profile	1. Application is accessible. 2. User is logged in to the application	User should be able to see and update his profile.

4.4 Admin Test Cases

Test Case Description	Pre-Requistite	Expected Result
Verify whether admin is able to see and update soil details	1. Application is accessible. 2. User is logged in to the application 3. Logged in User is registered as admin in the application	Admin should be able to see and update soil details.
Verify whether admin is able to see and update distributor details	1. Application is accessible. 2. User is logged in to the application 3. Logged in User is registered as admin in the application	Admin should be able to see and update distributor details.