Architecture

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	Tanmay Sharma	Introduction & Architecture defined
1.1	26 – July - 2023	Naman Malik	Architecture & Architecture Description appended and updated
1.2	27 – July - 2023	Naman Malik	Working of Node.js, Express.js with MongoDB Architecture defined and appended

Approval Status:

Version Comments

Review Date	Reviewed By	Approved By

Content

1. Introduction

1.1. What is Architecture design document?

Any software needs the architectural design to represents the design of software. IEEE defines architecture design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectures.

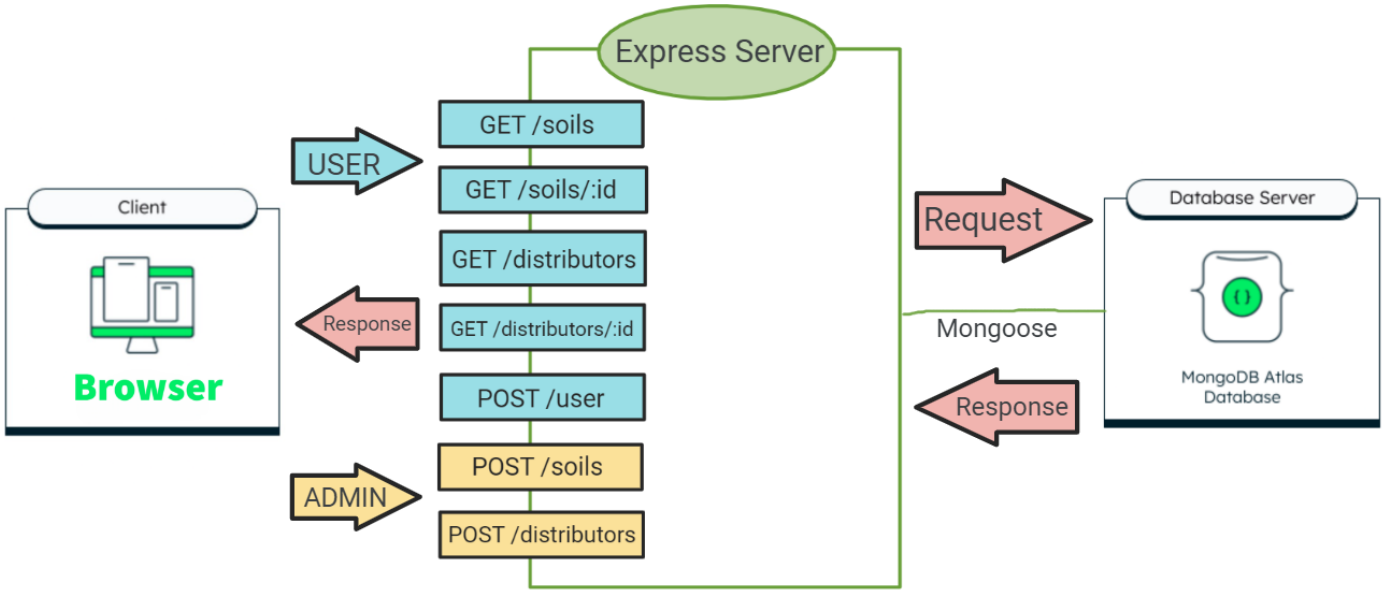
Each style will describe a system category that consists of:

- A set of components (e.g.: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

1.2. Scope

Architecture Design Document (ADD) is an architecture design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performed algorithms. Overall, the design principles may be defined during requirement analysis and then refined during architecture 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.

2.2 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.

2.3 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.

2.4 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.

2.5 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.

2.6 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.

2.7 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.

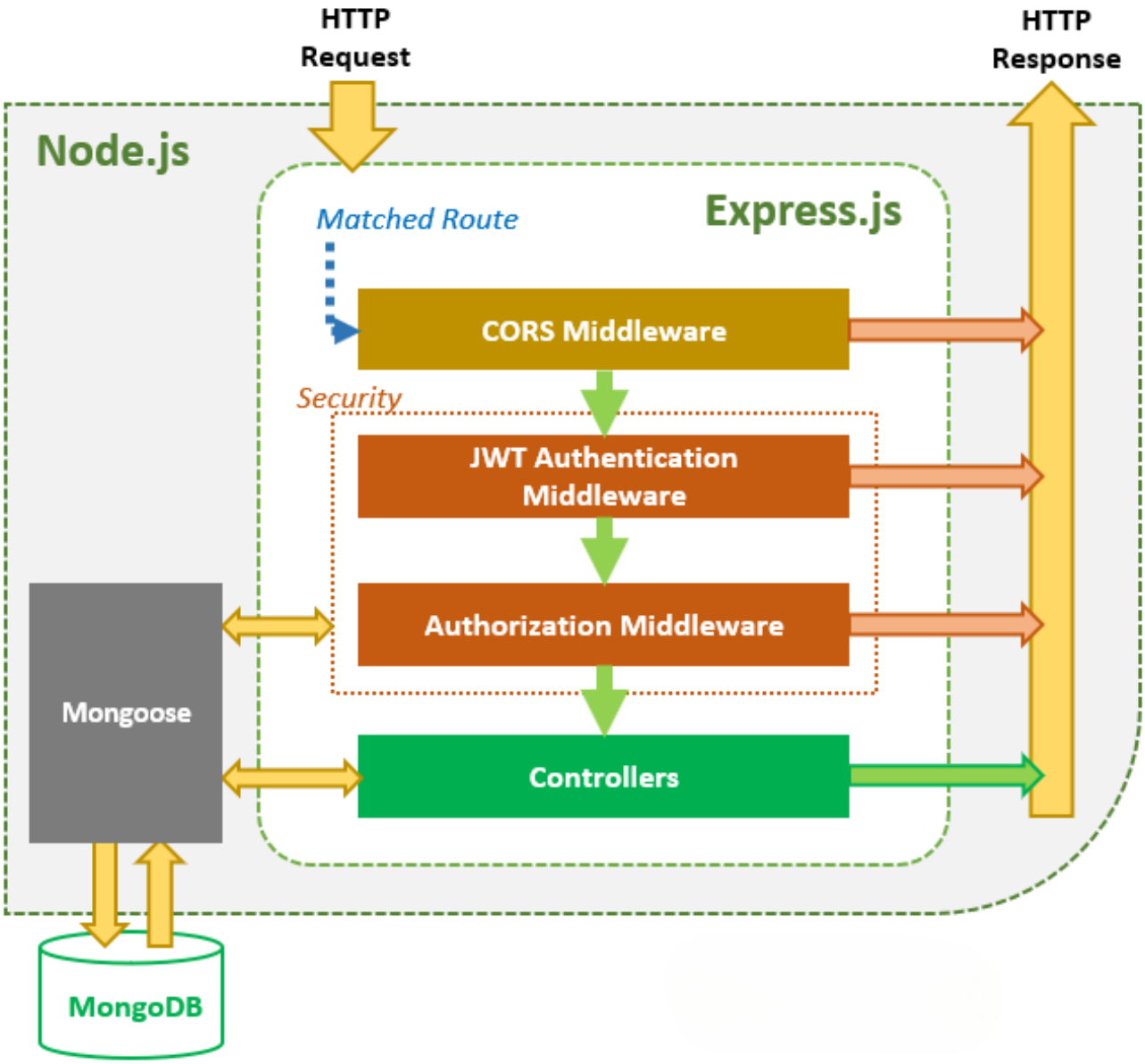
2.8 Data Management

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.

2.9 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. Working of Node.js Express.js with MongoDB



3.1 Detailed Working

1) Client Interaction:

- Clients (web browsers, mobile apps, etc.) send HTTP requests to the backend server (Node.js + Express.js) through specific routes.

2) CORS Middleware:

- CORS (Cross-Origin Resource Sharing) middleware is used to handle Cross-Origin requests. It allows or restricts requests from different origins (domains) to access resources on the server.
- When a client from different origin makes a requests, CORS middleware can add necessary headers to the response to allow or deny access based on configured rules.

3) JWT Authentication Middleware:

- JWT (JSON Web Token) authentication middleware is responsible for user authentication. It verifies the authenticity of users based on the presence of a valid JWT token in the request headers.
- When a user logs in or registers, a JWT token is generated on the server and sent back to the clients. Subsequent requests from the client include this token in the headers for authentication.
- The JWT authentication middleware verifies the token's signature and expiration to validate the user's identity.

4) Authorization Middleware:

- Authorization middleware checks whether the authenticated user has the necessary permissions to access specific routes or resources.
- Based on the user's role or other criteria, authorization middleware can grant or deny access to certain parts of the application.

5) Controllers:

- Controllers act as the bridge between the routing system and the business logic of the application.
- When a request matches a specific route, the corresponding controller function is invoked to process the request.
- Controllers handle the data processing, data retrieval from MongoDB using Mongoose, and formulate the response to be sent back to the client.

6) Mongoose (Object Data Modeling):

- Mongoose is an Object Data Modeling (ODM) library used to interact with MongoDB in a structured manner.
- It allows developers to define schemas for MongoDB documents and create models based on these schemas.
- The models provide an abstraction layer, simplifying the data retrieval, validation, and manipulation tasks with MongoDB.

7) Response Generation:

- After processing the request and performing necessary database operations, the controller generates an HTTP response containing the requested data or appropriate status codes.

8) Sending Response to Clients

- The generated response is sent back to the client (web browser or mobile app) over the HTTP connection.

This architecture allows for secure user authentication and authorization, seamless data interaction with MongoDB using Mongoose, and a structured approach to handling client requests. It offers an efficient and scalable solution for building modern web applications with Node.js, Express.js, and MongoDB.