**Audit Bazaar**

**Backend Technical Documentation**

## **Introduction**

This document provides an overview and technical details for the backend service of Audit Bazaar platform. The backend service is responsible for handling most of the functionality of the platform such as user activities, audits creation, etc.

## **Table of Contents**

* Technologies Used
* Setup
* API Endpoints
* Middlewares
* Database
* Swagger Documentation
* Test Cases
* Dockerfile

### 1. Technologies Used

The backend service of Audit Bazaar utilizes the following technologies:

* Node.js: A JavaScript runtime environment.
* Express: A web application framework for Node.js, used to build APIs and handle middleware functionalities.
* MongoDB: A NoSQL database used for data storage.
* Swagger: A tool to generate API documentation and interact with APIs through a user-friendly interface.
* Jest and Supertest: Testing frameworks for creating and running test cases.
* Docker: Used for containerization and deployment.

### 2. Setup

To set up two Node.js servers using the provided read me files, you can follow these steps for each server.

\*\*Server 1: Audit Bazaar Backend Service\*\*

1. Clone the repository:

git clone <repository-url>

2. Navigate to the project directory:

cd <project-directory>

3. Create an environment file:

- In the root directory, you should find an `env.example` file. Duplicate it and name the new file `.env`.

- Update the `.env` file with your configuration settings. Make sure to configure the MongoDB server URL and other required variables.

4. Install dependencies:

npm install

5. Build and run the server using npm:

npm start

This will start the server, and it will listen on <http://localhost:9000/api/v1>

6. Access the Swagger documentation for this server at:

<http://localhost:9000/swagger/>

Docker Setup:

If docker already installed skip this step, otherwise visit the official docker installation guide : <https://docs.docker.com/engine/install/>

After successfull installation, run following commands

=> sudo docker build -t audit-bazar .

=> docker run -d -it audit-bazar-

\*\*Server 2: Another Node.js Server\*\*

1. Clone the repository for the second server:

git clone <repository-url-for-second-server>

2. Navigate to the project directory for the second server:

cd <second-server-project-directory>

3. Create an environment file for the second server:

- In the root directory of the second server, you should find an `env.example` file. Duplicate it and name the new file `.env`.

- Update the `.env` file with your configuration settings, including any necessary environment variables.

4. Install dependencies for the second server:

npm install

5. Build and run the server for the second application using npm:

npm start

This will start the second server, and it will listen on a specific port as defined in the `.env` file or by default.

Docker Setup:

If docker already installed skip this step, otherwise visit the official docker installation guide : <https://docs.docker.com/engine/install/>

After successfull installation, run following commands

=> sudo docker build -t audit-bazar-cron .

=> docker run -d -t audit-bazar-cron

Now you have set up two Node.js servers using the provided read me files. Each server has its own unique functionality and Swagger documentation for API reference. You can customize the environment variables and configurations according to your specific needs for each server.

### 2. API Endpoints

The backend service provides the following API endpoints:

* /register: Endpoints related to user onboarding.
* /login: Endpoints related login after onboarding.
* /getUserInfo: Endpoints related to user info management.
* /updateProfile: Endpoints related to user profile updation.

Detailed documentation for each API endpoint, including request parameters, response formats, and authentication requirements, can be found in the Swagger documentation .

### 3. Middlewares

The backend service utilises the following middlewares:

* Body Parser: Responsible for parsing incoming request bodies in JSON format, making the data accessible in the request object.
* CORS: Enables Cross-Origin Resource Sharing to handle requests from different domains.
* Dotenv: Loads environment variables from a .env file, facilitating configuration management.
* Jwt Validation: Involves a JWT validation check after login ensuring restricted usage of platforms for genuine users.

### 4. Database

The backend service uses MongoDB, a popular NoSQL database, to store and manage data related to the Audit Bazaar platform. MongoDB was chosen for its flexibility and scalability, which aligns with the platform's requirements.

### 5. Swagger Documentation

The backend service is equipped with Swagger documentation, which provides a user-friendly interface to interact with the APIs. It includes detailed information about each API endpoint, such as available methods, request parameters, and response formats. Developers can access the Swagger documentation at /swagger endpoint.

### 6. Test Cases

To ensure the reliability and correctness of the backend service, a comprehensive suite of test cases has been developed using Mocha and Chai testing frameworks. These test cases cover different functionalities of the service, including API endpoints and middleware functionalities. Regular testing is conducted to identify and fix any issues that may arise during development or maintenance.

### 7. Dockerfile

For easy deployment and distribution, the backend service includes a Dockerfile. The Dockerfile defines the environment and dependencies required to run the service, making it easy to deploy the application as a container on various platforms.

### 8. Conclusion

This technical documentation provides an overview of the backend service for the Audit Bazaar platform. It describes the technologies used, available API endpoints, implemented middlewares, database choice, Swagger documentation, test cases, and Dockerfile for containerization. Developers and stakeholders can refer to this documentation for a better understanding of the backend service and its functionalities.

## 