Node.js Internship Task Document

Project Overview

The task is to build a **Node.js backend API** using **TypeScript** with **MongoDB** as the database. The project will be deployed on **Vercel**. This will help you understand backend development, CRUD operations, and deployment processes.

Project Name: Transaction Management System

Task to Be Done

Your task involves exposing the following four essential APIs:

1. **Create API**: This API will create new transaction entries in the database.
2. **Update API**: This API will update existing transaction entries in the database.
3. **Delete API**: This API will remove transaction entries from the database.
4. **Read API**: This API will retrieve all transaction entities from the database or specific entries based on unique keys as dictated by request parameters.

To ensure secure access, these APIs will utilize **JWT** (JSON Web Tokens), necessitating the development of a **Verification API** to retrieve the JWT token.

Note: Use the appropriate HTTP methods (e.g., `GET` to read, `POST` to create transactions)

Technologies to Use

1. **Backend Framework**: Node.js with TypeScript.
2. **Database**: MongoDB (e.g., MongoDB Atlas).
3. **Deployment Platform**: Vercel.
4. **Testing**: Postman or Hoppscotch collection for API documentation.

Instructions to Follow

1. **Environment Setup**:
    a. Install Node.js, npm, and TypeScript.
    b. Configure environment variables in a `.env` file (e.g., `MONGO_URI`, `PORT`).

2. **Folder Structure**: Use the following folder structure:

```
project-root
├── src
│   ├── controllers      # Logic for each endpoint
│   ├── models           # Mongoose models/schema
│   ├── routes           # API routes
│   ├── middleware       # Middleware for validation
│   ├── utils            # Helper functions
│   ├── app.ts           # Main app setup
├
├── .env                 # Environment variables
├── tsconfig.json        # TypeScript configuration
└── package.json
```

3. **Development**:
    a. Implement CRUD endpoints.
    b. Use MongoDB for database operations.
    c. Write validation middleware and error handling.

4. **Deployment**:
    a. Link the repository with Vercel.
    b. Configure environment variables on Vercel.
    c. Deploy the project and test the deployment.

5. **Testing**:
    a. Create clear and concise API documentation using Postman or Hoppscotch.