שולה
ספריית ציוד שכונתית

Technical Guide

Shula Neighborhood Equipment Library

Connecting Shula Library System Components

1. Development Environment

- **VSCode**: The recommended integrated development environment (IDE) for this project. It provides excellent support for **ReactJS**, **Node.js**, and **MongoDB** development with extensions for code highlighting, linting, and debugging.
- **Git**: Version control is managed using Git. The codebase should be hosted on a platform like GitHub or GitLab.
  - **Version Control**: Git is essential for tracking changes, managing different features through branching, and ensuring collaborative development.
  - **Branching Strategy**: A standard branching model (e.g., Git Flow or a simple feature-based workflow) is used to maintain code integrity.

2. Backend Technologies

- **Node.js & Express.js**: The backend is built using **Node.js** with the **Express.js** framework. This provides a fast, scalable, and non-blocking server environment to handle API requests.
- **Libraries**:
  - **Mongoose**: An Object Data Modeling (ODM) library for MongoDB. It defines structured schemas and provides an abstraction layer to simplify database interactions.
  - **bcrypt**: A library used for hashing and salting passwords before they are stored in the database, which is a critical security measure.
  - **jsonwebtoken (JWT)**: Used for creating and verifying JSON Web Tokens for secure, stateless user authentication and session management.
  - **dotenv**: A zero-dependency module that loads environment variables from a .env file into process.env, keeping sensitive information out of the codebase.

3. Database Management

- **MongoDB**: The primary database for all application data. MongoDB is a NoSQL, document-oriented database that offers flexibility and horizontal scalability.
  - **Collections**: Data is organized into collections, which are analogous to tables in a relational database. The main collections are products, customers, and rentals.
  - **Schemas**: Mongoose schemas enforce a structure on the documents within each collection, ensuring data consistency and integrity.

- **Connection**: The backend connects to the MongoDB instance using a connection string, which is securely stored in a .env file.

4. Frontend Technology

- **ReactJS**: The frontend is a Single-Page Application (SPA) built with **ReactJS**. It provides a component-based architecture for building a dynamic and responsive user interface.
- **Key Libraries**:
  - **React Router**: Manages client-side routing, enabling navigation between different pages (/, /products, /dashboard) without a full page reload.
  - **Axios**: A promise-based HTTP client for making API requests from the frontend to the backend. It is used to fetch product data, submit forms, and handle user authentication.
  - **Redux Toolkit (Optional)**: A state management library for handling the application's global state, such as user login status or the shopping cart contents.

5. Security and Configuration

- **Secrets Management (.env)**: The .env file is used to securely store sensitive information, such as the MongoDB connection string, JWT secret keys, and API credentials.
  - **Security**: This file is never committed to Git by adding .env to the .gitignore file, preventing credentials from being exposed.
  - **Example .env file**:

```
MONGO_URI=mongodb://localhost:27017/rentalstore
JWT_SECRET=your_super_secret_jwt_key
```

שולה

ספריית ציוד שכונתית

Installation Guide for the Online Rental Store

This guide outlines the steps to set up and run the project locally.

Please refer to the README.md and PROJECT_GUIDE.md files for more details.

1. **Prerequisites**
   - **Git**: For cloning the repository.
   - **Node.js & npm**: The backend is built on Node.js, and npm (Node Package Manager) is used to install dependencies.
   - **MongoDB**: A local or cloud-based MongoDB instance to serve as the database.
2. **Cloning the Repository**
   - Open a terminal and run the following command to clone the project:

     Bash
     ```
     git clone https://github.com/TymorIbrahim/ShulaWebApp2.git
     ```
   - Navigate into the project directory:

     Bash
     ```
     cd ShulaWebApp2
     ```
3. **Setting Up the Backend**
   - Navigate to the backend folder: cd backend
   - Install backend dependencies:

     Bash
     ```
     npm install
     ```
   - Create a .env file in the backend folder and add your MongoDB connection string and JWT secret.
   - Start the backend server:

     Bash
     ```
     npm start
     ```
4. **Setting Up the Frontend**
   - Navigate to the frontend folder: cd ../frontend
   - Install frontend dependencies:

     Bash
     ```
     npm install
     ```
   - Start the React application:

     Bash
     ```
     npm start
     ```

5. **Accessing the Application**
   - The backend will typically run on http://localhost:5000 or a similar port.
   - The React frontend will be accessible at http://localhost:3000.

Maintenance Guide for the Online Rental Store

To ensure the smooth functioning of the application, follow these maintenance guidelines.

1. **Regular Updates**
   - **Dependencies**: Regularly update both the frontend and backend dependencies to address security vulnerabilities and benefit from new features.
   - **Backend**: Navigate to the backend folder and run npm update.
   - **Frontend**: Navigate to the frontend folder and run npm update.
2. **Data Integrity**
   - **Database Backups**: Regularly back up your MongoDB database to prevent data loss. You can use MongoDB's mongodump command or a cloud provider's backup services.
   - **Data Validation**: Periodically audit the database to ensure all documents adhere to the defined schemas and check for any data inconsistencies.
3. **Error Handling and Logging**
   - **Logging**: Implement a logging solution (using libraries like Winston) to track errors, API failures, and other critical events on the backend. This helps in proactive issue resolution.
   - **Frontend Error Boundaries**: Use React's **Error Boundaries** to gracefully handle errors in the UI, preventing the entire application from crashing.
4. **Security Best Practices**
   - **Secrets Management**: Periodically rotate your JWT secret key and other sensitive credentials stored in the .env file to enhance security.
   - **HTTPS**: When deploying the application to a public server, ensure it runs over HTTPS to secure data in transit. You can use services like Let's Encrypt for free SSL certificates.
   - **User Passwords**: Ensure all new and existing user passwords are a strong hash using bcrypt and are never stored as plain text.