PROJECT DOCUMENTATION

MEDICAL INVENTORY MANAGEMENT

INTRODUCTION

- **PROJECT TITLE**

  Medical Inventory Management

- **TEAM MEMBERS**
  - ***Saradappa Gari Sireesha(Team Leader)***
  - ***Jadavallu Swathi***
  - ***Vaishnavi Sravanthi Paturu***
  - ***Yashwitha p***

PROJECT OVERVIEW

- **PURPOSE**
  - ***Streamline inventory management***: Automate and optimize inventory management processes for medical supplies, equipment, and pharmaceuticals.
  - ***Improve visibility and tracking***: Provide real-time visibility into inventory levels, locations, and movement, enabling better decision-making and reducing stockouts or overstocking.
  - ***Enhance patient care***: Ensure that medical supplies and equipment are available when needed, enabling healthcare professionals to provide high-quality patient care.
- ***GOALS***
  - ***Inventory tracking and management***: Track inventory levels, locations, and movement in real-time, including tracking of batch numbers, expiration dates, and serial numbers***.***
  - ***User-friendly interface:*** Provide a user-friendly interface for inventory management, enabling healthcare professionals to easily manage inventory and focus on patient care
- ***FEATURES***
  - ***Inventory Tracking:*** Track inventory levels, locations, and movement in real-time.
  - ***Product Catalog:*** Manage a catalog of medical products, including product descriptions, pricing, and vendor information.
  - ***3. Barcode Scanning:*** Use barcode scanning to track inventory movement and reduce errors.
  - ***4. Automated Workflows:*** Automate inventory management workflows, such as reorder points, stock transfers, and inventory adjustments***.***
  - ***5. Reporting and Analytics:*** Provide insights into inventory usage, trends, and optimization opportunities through reporting and analytics.
  - ***6. Integration with Other Systems***: Integrate with other healthcare systems, such as electronic health records (EHRs) and enterprise resource planning (ERP) systems.

- ***7. Security and Access Control:*** Ensure that sensitive inventory data is secure and accessible only to authorized personnel.

➢ *FUNCTIONALITIES*

1. Inventory Management: Manage inventory levels, track inventory movement, and perform inventory adjustments.
2. Order Management: Manage orders, including purchase orders, stock transfers, and sales orders.
3. Vendor Management: Manage vendor information, including contact details, pricing, and delivery terms.
4. Product Management: Manage product information, including product descriptions, pricing, and inventory levels***.***
5. Warehouse Management: Manage warehouse operations, including receiving, storing, and shipping inventory.
6. Inventory Reporting: Generate reports on inventory levels, movement, and usage***.***
7. Alerts and Notifications: Send alerts and notifications for low inventory levels, stockouts, and other inventory-related issues.

ARCHITECTURE

➢ *FRONTEND*

The frontend will handle user interactions, displaying inventory data, and sending requests to the backend API. Here's a high-level overview:
1. Components:
- InventoryList: Displays a list of medical inventory items***.***
- InventoryItem: Represents an individual inventory item with details like name, quantity, and expiration date.
- AddInventoryForm: A form for adding new inventory items***.***
- EditInventoryForm: A form for editing existing inventory items***.***
2. State Management:
- Use React's Context API or Redux to manage state globally, such as inventory data and user authentication status.
3. API Calls:
- Use Axios or Fetch API to make HTTP requests to the backend API for CRUD operations on inventory items.
4. Routing:
- Implement React Router for client-side routing, with routes for inventory list, add inventory, edit inventory, and login/signup.

➢ *BACKEND*

The backend will handle API requests, interact with the database, and perform business logic.
1. API Endpoints:
- GET /api/inventory: Retrieves a list of all inventory items***.***
- POST /api/inventory: Creates a new inventory item***.***
- GET /api/inventory/:id: Retrieves a single inventory item by ID.
- PUT /api/inventory/:id: Updates an existing inventory item***.***
- DELETE /api/inventory/:id: Deletes an inventory item.

2. Database Interactions:

- Use Mongoose to interact with the MongoDB database, defining a schema for inventory items.

3. Authentication and Authorization:

- Implement authentication using JSON Web Tokens (JWT) or Passport.js to secure API endpoints***.***

- ***DATABASE***

The database schema will define the structure of inventory items.

1. Inventory Schema:

- name: String, required
- quantity: Number, required
- expirationDate: Date
- description: String

2. Mongoose Model:

- Create a Mongoose model for the inventory schema, allowing for CRUD operations.

SETUP INSTRUCTIONS

- ***PREREQUISITES***
  - ✓ ***Salesforce Developer Edition:*** Ensure you have a Salesforce Developer Edition org.
  - ✓ ***Git:*** Install Git on your local machine.
  - ✓ ***Node.js***: Install Node.js (LTS version) on your local machine.
  - ✓ ***Salesforce CLI:*** Install Salesforce CLI on your local machine***.***

- ***INSTALLATION***

STEP BY STEP GUIDE TO CLONE

- Open your terminal or command prompt.
- Navigate to the directory where you want to clone the repository.
- ***Run the command:*** git clone https://github.com/username/medical-inventory-management.git (replace username with the actual repository owner).

- ***Install Dependencies***
  - ***Navigate to the project directory***: cd medical-inventory-management
  - ***Run the command***: npm install or yarn install to install dependencies.
- ***Set up Environment Variables***
  - ***1. Create a .env file in the project root directory.***
  - ***2. Add the following environment variables:***
  - ***- SALESFORCE_USERNAME: Your Salesforce username.***
  - ***- SALESFORCE_PASSWORD: Your Salesforce password.***
  - ***- SALESFORCE_TOKEN: Your Salesforce security token.***
  - ***- SALESFORCE_URL: Your Salesforce instance URL (e.g., https://login.salesforce.com).***
  - ***3. Example .env file:***
  - ***SALESFORCE_USERNAME=myusername***

- ***SALESFORCE_PASSWORD=mypassword***
- ***SALESFORCE_TOKEN=mysecuritytoken***
- ***SALESFORCE_URL=https://login.salesforce.com***

FOLDER STRUCTURE

- ***CLIENT***
- ***SERVER***

RUNNING THE APPLICATION

- Provide commands to start the frontend and backend servers locally
- ***FRONTEND***

  1. Open a terminal or command prompt.
  2. Navigate to the client directory:
  bash
  cd client
  3. Install dependencies (if not already installed):
  bash
  npm install
  4. Start the frontend server:
  bash
  npm start
  5. The React app should now be running at http://localhost:3000.
- ***BACKEND***

  1. Open a new terminal or command prompt.
  2. Navigate to the server directory:
  bash
  cd server
  3. Install dependencies (if not already installed):
  bash
  npm install
  4. Start the backend server:
  ```bash
  npm start
  5. The Node.js server should now be running at http://localhost:8080/ (or the port specified in your server configuration).

## *API DOCUMNETATION*
  ```

AUTHENTICATION

- ***AUTHENTICATION***
  - ***OAuth 2.0:*** Salesforce uses OAuth 2.0 to authenticate users and authorize access to data. OAuth 2.0 provides a secure way to access Salesforce data without sharing passwords.
  - ***Session Management***: Salesforce uses session IDs to manage user sessions**.** When a user logs in, a session ID is generated and used to authenticate subsequent requests**.**
  - ***JSON Web Tokens (JWTs):*** JWTs can be used for authentication and authorization in Salesforce. JWTs contain user information and are digitally signed to prevent tampering**.**
- ***AUTHORIZATION***
  - ***Role-Based Access Control (RBAC):*** Salesforce uses RBAC to control access to data based on user roles. Users are assigned roles, and each role has specific permissions and access levels.
  - ***Object-Level Security***: Salesforce provides object-level security to control access to specific objects and fields. Administrators can set permissions for each object and field to determine who can view, edit, or delete data.
  - ***Field-Level Security:*** Field-level security allows administrators to control access to specific fields within an object. This ensures that sensitive data is only accessible to authorized users.
- ***TOKENS AND SESSIONS***
  - ***Access Tokens***: Access tokens are used to authenticate and authorize API requests. They are typically valid for a short period and can be refreshed using refresh tokens.
  - ***Refresh Tokens***: Refresh tokens are used to obtain new access tokens when the existing one expires. This allows for long-term access to Salesforce data without requiring users to re-authenticate.
  - ***Session IDs***: Session IDs are used to manage user sessions and authenticate requests. Session IDs can be obtained through various authentication flows, including OAuth 2.0.

USER INTERFACE

TESTING

- ***Testing Strategy***
  - ***Unit Testing***: Focus on individual components or classes within the Salesforce application, ensuring each unit functions correctly.
  - ***Integration Testing***: Verify interactions between different components, such as integrations with external systems or between custom and standard Salesforce features.
  - ***System Testing***: Validate the entire system, simulating real-world scenarios to ensure the application behaves as expected.

- ✓ ***User Acceptance Testing (UAT):*** Engage end-users to test the application, ensuring it meets business requirements and is user-friendly.

➢ ***Testing Tools***

- ***Salesforce DX***: A set of tools that enables developers to develop and test Salesforce applications in a more agile and efficient way.
- ***Apex Testing***: Salesforce's built-in testing framework for writing unit tests for Apex code.
- ***Lightning Testing Service (LTS):*** A tool for testing Lightning components, allowing developers to write tests for JavaScript and Apex code.
- ***Selenium***: An open-source tool for automating web browsers, useful for end-to-end testing and UAT.
- ***Jenkins:*** A popular CI/CD tool that can be integrated with Salesforce to automate testing and deployment processes.

SCREENSHOTS OR DEMO

KNOWN ISSUES

- ***API Limitations:*** Reaching API limits can cause operational problems, integration issues, and data synchronization failures. To manage API usage, consider batch processing, caching strategies, request optimization, and off-peak scheduling**.**
- ***Security Issues:*** Vulnerabilities can expose sensitive data. Implement field-level security, quarterly access reviews, permission set rationalization, login monitoring, and the principle of least privilege.
- ***Performance Issues***: Poor configuration can lead to slow performance. Optimize reports, implement caching strategies, build custom indexes, and schedule resource-consuming operations during quiet hours.
- ***Integration Failures:*** Outdated API versions, inconsistent data formats, and overlooked system limits can cause integration issues. Monitor integrations, ensure API versions are current, and test thoroughly before deployment.
- ***Data Integrity Problems:*** Duplicate or outdated data can lead to inefficiency and miscommunication. Regularly cleanse data, implement validation rules, and use deduplication tools.

FUTURE ENHANCEMENTS

- ***Automated Inventory Tracking:*** Implement barcode scanning and RFID tracking to ensure accurate and real-time inventory updates
- ***Predictive Analytics***: Leverage machine learning algorithms to forecast demand and optimize inventory levels, reducing stockouts and overstocking
- ***Smart Reordering:*** Automate reordering processes based on predefined thresholds, lead times, and supplier constraints
- ***Inventory Visibility:*** Provide real-time visibility into inventory levels, locations, and movement across multiple warehouses and facilities
- ***Salesforce Health Cloud Integration:*** Integrate with Salesforce Health Cloud to leverage its capabilities in managing medical inventory, tracking product information, and streamlining sales processes
- ***Electronic Health Records (EHRs) Integration:*** Integrate with EHR systems to ensure seamless data exchange and accurate inventory management

- ***Supplier Integration:*** Integrate with supplier systems to enable real-time inventory updates, automated ordering, and streamlined communication