**StoreManager: Your Inventory Assistant**

**Introduction**

**Project Title**: StoreManager: Your Inventory Assistant  
**Team Members**:  
• **Team Leader**: [Vaitheswaran G]  
• **Team Member 1**: [Akshaya S]  
• **Team Member 2**: [Diksha B]  
• **Team Member 3**: [Harini R]

**Project Overview**

**Purpose**

StoreManager is designed to help small-to-medium-sized businesses manage their inventory, track product stock levels, and streamline restocking processes through an intuitive dashboard.

**Features**

* User-friendly dashboard for viewing and managing inventory
* Add, update, and delete product information
* Track low-stock items and restock alerts
* Search and filter products by category or availability
* Responsive design optimized for desktops and tablets
* CRUD operations powered by a JSON server backend

**Architecture**

**Component Structure**

* **App** → Root container with routing and layout structure
* **InventoryList** → Fetches and displays list of inventory items
* **ProductForm** → Handles creation and editing of products
* **LowStock** → Displays items below threshold stock level
* **Navbar** → Navigation across dashboard, inventory, and settings
* **Settings** → Manage app preferences like thresholds and categories

**State Management**

* **Local State**:
  + Input fields for product data
  + UI modals and confirmation messages
  + Search and filter states
* **Global State**:
  + Product list and inventory items
  + Managed via useState and updated with Axios to a JSON server

**Routing**

* / → Dashboard overview
* /inventory → Full inventory list
* /low-stock → View low-stock items
* /add-product → Form to add a new product
* Routing handled using **react-router-dom**

**Setup Instructions**

**Prerequisites**

* Node.js & npm
* Git
* Code editor (Visual Studio Code recommended)

**Installation Steps**

1. Clone the repository:  
   git clone https://github.com/yourusername/store-manager.git
2. Navigate into the project directory:  
   cd store-manager
3. Install dependencies:  
   npm install
4. Start the React application:  
   npm start
5. Run the JSON server:  
   json-server --watch ./db/inventory.json

**Folder Structure**

store-manager/

├── src/

│ ├── components/

│ ├── pages/

│ ├── assets/

│ ├── utils/

│ └── App.js

├── db/

│ └── inventory.json

├── package.json

**Utilities**

* **Axios** → For API communication with JSON server
* **Helper Functions** → Used for sorting, searching, and calculating stock levels

**Running the Application**

* **Frontend**:  
  Run npm start
* **Backend (Mock API)**:  
  Run json-server --watch ./db/inventory.json

**Component Documentation**

**Key Components**

* **App** → Root-level container for routing and layout
* **InventoryList** → Renders all products with edit/delete options
* **ProductForm** → Add or update product info via form
* **LowStock** → Shows warning list for items below threshold
* **Settings** → Adjust application settings (stock limits, etc.)

**State Management Summary**

* **Global**:
  + Product list fetched from JSON server and shared across components
* **Local**:
  + Form inputs, modal visibility, search term

**User Interface**

* **Design**: Clean and minimal dashboard interface
* **Frameworks**: Bootstrap + Tailwind CSS
* **Navigation**: Sidebar with clear routing and real-time status indicators

**Styling**

* **CSS Frameworks**:  
  Bootstrap (for layout) and TailwindCSS (for utility-first styling)
* **Theming**:  
  Custom color palette and reusable style classes for buttons, tables, and modals

**Testing**

**Strategy**

* Unit and integration tests using **Jest** and **React Testing Library**

**Coverage**

* Tests written for form validation, inventory calculations, and API interactions
* Coverage report generated via npm run test -- --coverage

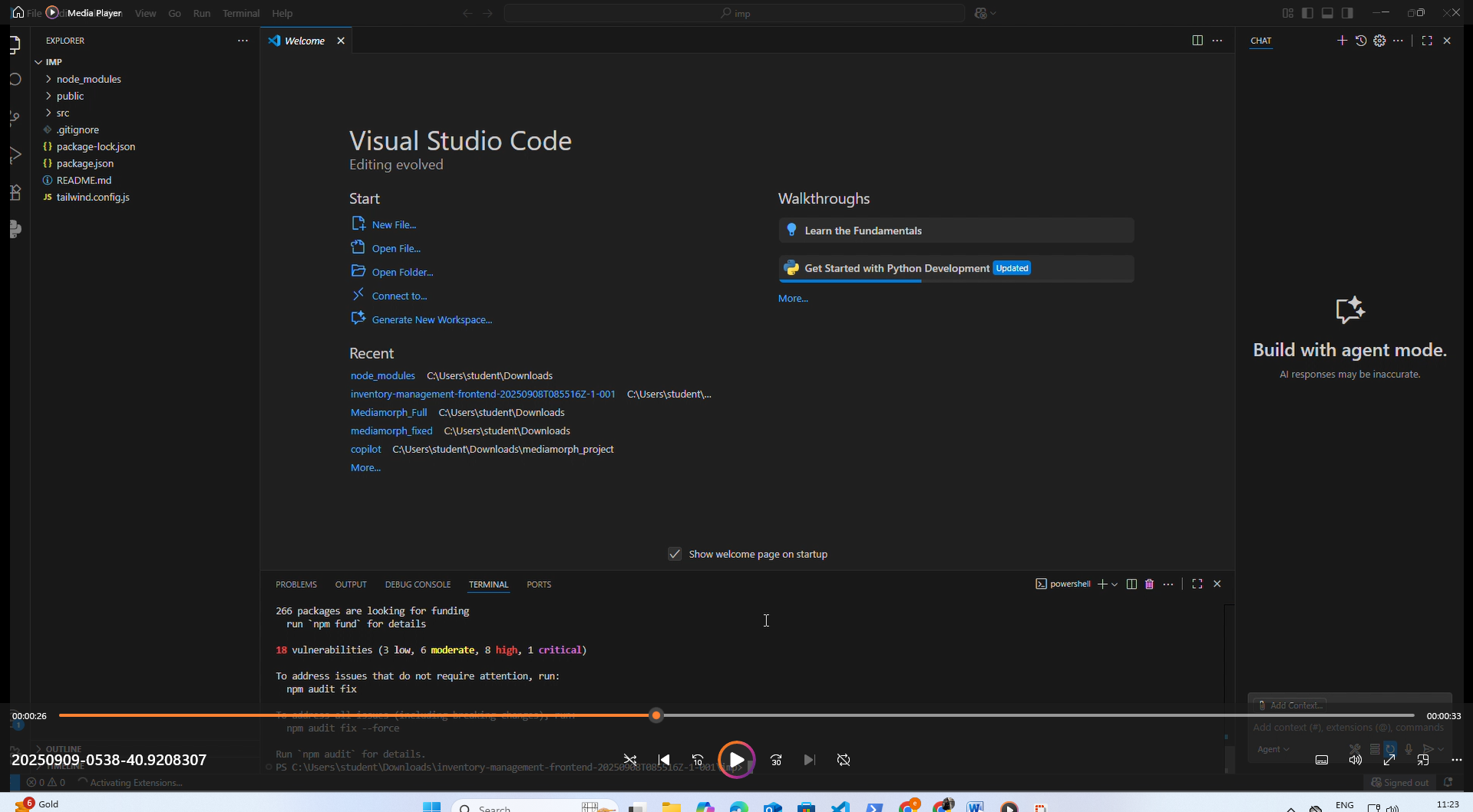
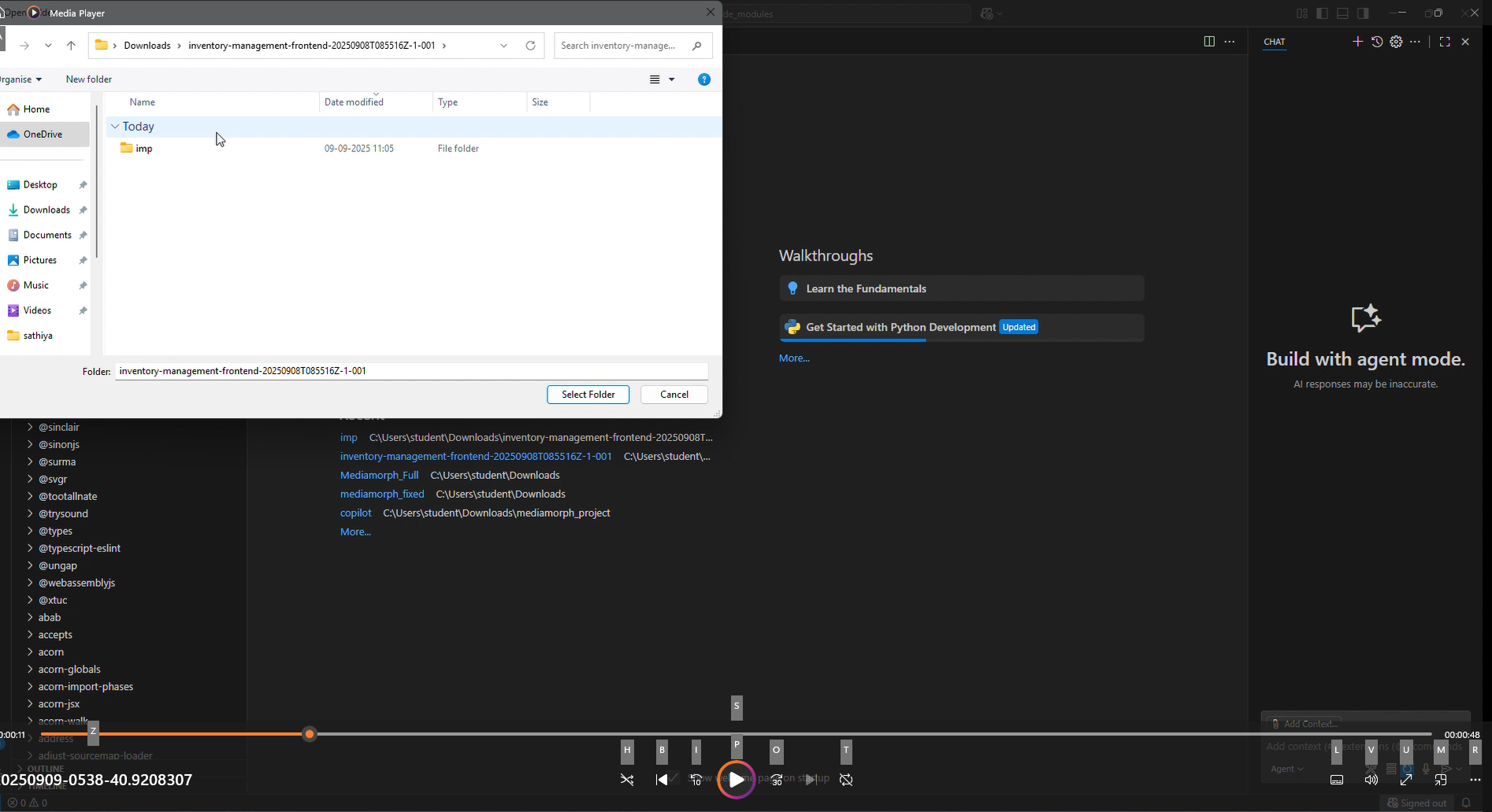
**Screenshots or Demo**

* **Demo Video**: Click Here
* **GitHub Repository**: Click Here

**Known Issues**

* Lag in UI updates when multiple API calls happen in rapid succession
* JSON server occasionally drops changes if not saved properly

**Future Enhancements**

* User authentication and role-based access
* Integration with real-time database (e.g., Firebase or MongoDB)
* PDF and CSV export for inventory reports
* Email alerts for low-stock items
* Mobile app version for quick inventory management

