**1. TITLE PAGE**

* **Project Title:** **Inventory Store Manager**
* **Team ID:** [Your Team ID]

**Team Leader SANTHOSH E**

*santhoshguru184@gmail.com*

* **Team Members:**

**i. SANTHOSH E** [**santhoshguru184@gmail.com**](mailto:santhoshguru184@gmail.com)

**ii. HARISH N** [**harish8148285177@gmail.com**](mailto:harish8148285177@gmail.com)

**iii. SUMAN N** [**s93973880@gmail.com**](mailto:s93973880@gmail.com)

**iv. KARTHICK M** [**kavikarthick483@gmail.com**](mailto:kavikarthick483@gmail.com)

**2. ABSTRACT**

The "Inventory Store Manager" is a web-based application designed to streamline and automate inventory management for small to medium-sized businesses. The primary purpose of this project is to replace inefficient manual tracking systems with a centralized, real-time solution. The application allows users to add new products, track current stock levels, monitor sales, and generate comprehensive reports. Developed using React.js for the frontend, Node.js and Express.js for the backend, and MongoDB for the database, this system provides an intuitive and secure platform to enhance business efficiency and accuracy.

**3. INTRODUCTION**

* **3.1 Project Overview:** The Inventory Store Manager is a full-stack application built to simplify the complexities of inventory management. It provides a user-friendly dashboard for business owners to gain a clear overview of their stock.
* **3.2 Problem Statement:** Manual inventory tracking is often time-consuming, prone to human error, and lacks real-time data visibility. This can lead to stockouts, overstocking, and inaccurate financial reporting.
* **3.3 Objectives:**
  + - 1. To develop a secure and user-friendly web application for inventory management.
      2. To enable real-time tracking of products, stock levels, and sales.
      3. To provide data-driven insights through automated reports.
      4. To create a scalable solution suitable for small businesses.
* **3.4 Scope:** The project includes core inventory management features such as product creation, stock updates, and sales recording. It does not include advanced features like supply chain management, integrated payment gateways, or a built-in e-commerce platform.

4. TECHNOLOGY STACK

* **Frontend:** The user interface is built with React.js, ensuring a dynamic and responsive user experience. It utilizes Bootstrap and Material UI for a clean and modern design.
* **Backend:** The server-side logic and API endpoints are managed by Node.js and Express.js.
* **Database:** MongoDB is used to store all data, including user credentials, product information, and sales records.

**5. SYSTEM ARCHITECTURE**

The application follows a standard MERN stack architecture, consisting of a client-server model. The React frontend sends requests to the Node.js/Express.js backend, which then interacts with the MongoDB database. Data is securely transferred between the components using RESTful APIs.

6. FOLDER STRUCTURE

inventory-store-manager/

|-- client/

| |-- public/

| |-- src/

| | |-- components/

| | |-- pages/

| |-- package.json

|-- server/

| |-- routes/

| |-- models/

| |-- controllers/

| |-- package.json

7. INSTALLATION & SETUP

* 7.1 Prerequisites: You will need to have Node.js, MongoDB, and Git installed on your machine.
* 7.2 Installation Steps:
* Clone the repository: git clone [repository URL]
* Navigate to the project root: cd inventory-store-manager
* Install client dependencies: cd client and then npm install
* Install server dependencies: cd ../server and then npm install
* **7.3 Running the Application:**
  + - Start the backend server: cd server and then npm start
    - Start the frontend application: cd ../client and then npm start
* **7.4 Access:** The application will be accessible via your web browser at http://localhost:3000.

8. API DOCUMENTATION

* **User Authentication:**

/api/user/register (POST) - Registers a new user.

/api/user/login (POST) - Authenticates and logs in a user.

* **Products:**

/api/products/add (POST) - Adds a new product to the inventory.

/api/products (GET) - Fetches all products.

/api/products/:id (PUT) - Updates an existing product.

* **Sales:**

/api/sales/add (POST) - Records a new sale.

/api/sales/reports (GET) - Generates sales reports.

9. USER INTERFACE

* **Dashboard:** Provides a quick overview of key metrics, such as total products and recent sales.
* **Products Page:** Allows users to view, add, edit, and delete products.
* **Sales Report Page:** Displays sales data in a clear, easy-to-read format.

10. TESTING

The application was primarily tested through manual testing throughout the development lifecycle to ensure all features function as intended. Tools such as Postman were used for API endpoint testing and Chrome Dev Tools for debugging the frontend interface.

11. KNOWN ISSUES

* Data validation could be more robust to handle edge cases.
* Error handling for network failures can be improved.

12. FUTURE ENHANCEMENTS

* Implement a barcode scanner feature for faster product entry.
* Add a feature to generate PDF or CSV reports.
* Develop a separate dashboard for administrators with more control over user accounts.
* Integrate a payment gateway to process transactions directly within the application.