**Project Title**

* **Team Name: Centralized Sales Performance Dashboard (RetailVue)**
* **Team Members: Prathyusha Parameshwar Ganiga (287617), Prerana Joshi (287640)**
* **Problem Statement**: A growing retail business that operates across multiple regions faces challenges in tracking its sales performance. They rely on multiple platforms for transactions, making it difficult to consolidate sales data, monitor customer behavior, and predict future trends. The company needs a solution that offers a real-time, centralized system to visualize key metrics, track performance across regions, and provide insights for informed decision-making.

**1. Understanding of the Problem statement**

1. **Explanation of the Problem Context**

The business in question is a growing retail company operating across multiple regions, likely with a diverse customer base and sales channels. The company faces significant challenges in tracking and managing its sales performance due to reliance on multiple, disconnected platforms for transaction processing.

* **Consolidate Sales Data**: The lack of integration across platforms means sales data is stored in silos, preventing the company from getting a holistic view of performance across different regions and sales channels.
* **Predict Trends**: Without an integrated view of past performance and customer behavior, predicting sales trends, demand fluctuations, and inventory requirements is challenging, which could lead to either overstocking or stockouts.

1. **Key Requirements Identified:**

To address the challenges faced by the business, the solution needs to meet several key requirements, particularly focusing on creating a **Minimum Viable Product (MVP)** that delivers immediate value within the given timeline:

1. **Centralized Sales Dashboard**:

* The dashboard should provide a **single source of truth** for all sales metrics, reducing the need to manually aggregate data from various platforms.

1. **Regional Performance Tracking**:

* The system must allow users to monitor performance across different regions and geographies, highlighting trends, outliers, and areas requiring attention.
* A regional comparison tool should be included to analyze sales performance across various territories.

1. **Basic Predictive Analytics (Trend Forecasting)**:

* While a comprehensive forecasting solution may not be part of the MVP, the system should incorporate basic predictive analytics that uses historical data to estimate sales trends or demand.

1. **Security and Access Control**:

* Data security must be ensured, with proper user authentication and access control mechanisms in place, allowing different stakeholders to access only the data they need.
* Sensitive customer information and transactional data should be encrypted to meet privacy regulations.

**2. Solution Overview**

1. **Solution Summary** The business in question is a growing retail company operating across multiple regions, likely with a diverse customer base and sales channels. The company faces significant challenges in tracking and managing its sales performance due to reliance on multiple, disconnected platforms for transaction processing.

* **Consolidate Sales Data**: The lack of integration across platforms means sales data is stored in silos, preventing the company from getting a holistic view of performance across different regions and sales channels.
* **Predict Trends**: Without an integrated view of past performance and customer behavior, predicting sales trends, demand fluctuations, and inventory requirements is challenging, which could lead to either overstocking or stockouts.

1. **Objective** The primary objective of this solution is to enhance operational efficiency and decision-making across all levels of the retail organization by implementing a secure, flexible, and scalable system.

Key objectives include:

* **Access Control and Role-based Analytics: Allow users to access and view information pertinent to their roles. For example:**
  + **Business Retailers can view analytics across the organization, add new branches, and assign branch retailers.**
  + **Branch Retailers have access to branch-specific analytics, can manage product restocking, and oversee inventory at their branch.**
  + **Sales Reps can create orders and download generated invoices.**
* **User-Specific Functions and Views:** Each user can view **Products, Branches, Orders, Inventory,** and **Customers** relevant to their permissions, ensuring that information is accessible without compromising security.
* **Scalability and Modularity:** By breaking down the system into microservices, we ensure that each component can evolve independently, supporting seamless scaling as the business grows.
* **Enhanced User Experience:** With the role-based structure, each user has a personalized interface that reduces complexity and allows faster access to relevant information, thus improving productivity and customer service.

**3. Features and Functionalities**

**a. Core Features**

**1. Role-Based Access Control**: Secure authorization using JWT tokens with permissions tailored to each role—Business Retailer, Branch Retailer, and Sales Rep.

**2. Microservices Architecture**: Distinct microservices (User, Product, Orders, Customer, and Branch) manage specific functionalities, ensuring modularity and scalability.

**3. Inventory Management**: Allows users to view and manage inventory at branch or organization-wide levels, depending on role.

**4. Order Processing and Invoice Generation**: Sales Reps can create orders and generate downloadable invoices.

**5. Analytics and Reporting**: Role-specific analytics for tracking branch-level or organization-wide performance.

**6. Branch Management**: Business Retailers can add branches and assign Branch Retailers to manage them.

**7. Product and Customer Viewing**: All users can view product and customer information within their access permissions.

**b. Additional Features**

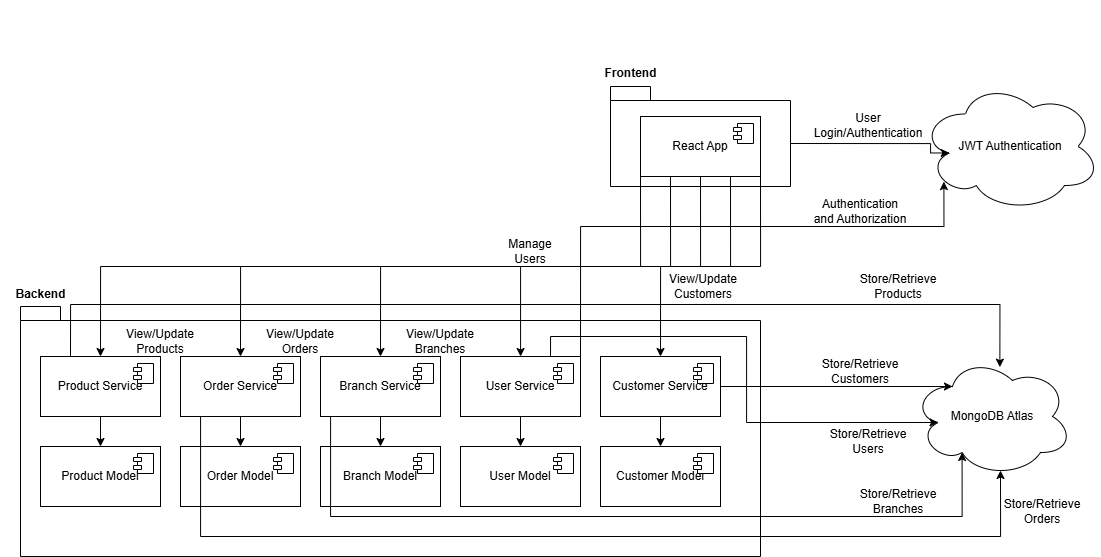
**Advanced Filtering and Search**: Enables users to filter products, orders, and customers based on criteria like date, product name, and branch.

**c. User Flows**

1. **Business Retailer (Organization Head)**
   * **Login**: Authenticates via JWT token, accesses business dashboard.
   * **View Organization Analytics**: Can view comprehensive reports covering all branches.
   * **Manage Branches**: Adds new branches, assigns Branch Retailers, and monitors branch-level analytics.
   * **Monitor Inventory**: Checks stock levels across all branches and identifies restocking needs.
   * **Product and Customer Viewing**: Views product and customer data across the entire organization.
2. **Branch Retailer (Branch Manager)**
   * **Login**: Authenticates via JWT, accesses branch dashboard.
   * **View Branch Analytics**: Sees analytics related to their specific branch, including sales trends and product demand.
   * **Manage Inventory**: Checks stock levels, places restocking orders, and adjusts product availability for the branch.
   * **Product and Customer Viewing**: Views data relevant to their branch.
3. **Sales Rep (Branch Employee)**
   * **Login**: Authenticates via JWT, accesses sales dashboard.
   * **Create Orders**: Adds new orders for customers, views product availability, and places orders based on customer needs.
   * **Generate Invoice**: After order creation, generates and downloads invoices for customers.
   * **View Products, Orders, and Customers**: Accesses data relevant to their branch, enhancing customer service by providing accurate product and order information.

**4. Architecture Diagram**

**a. System Architecture**



Architecture Diagram

1. **User Interface (UI) Layer**:
   * Accessed by **Business Retailers**, **Branch Retailers**, and **Sales Reps** through web applications.
2. **API Gateway**:
   * Central entry point that routes requests to appropriate microservices. Manages load balancing, security (JWT token validation).
3. **Microservices**:
   * Each microservice handles a distinct domain and communicates through REST APIs.
     + **User Service**: Manages user authentication and role-based authorization.
     + **Product Service**: Manages product data, inventory levels, and stock status.
     + **Order Service**: Processes orders, generates invoices, and manages order history.
     + **Customer Service**: Manages customer profiles and data.
     + **Branch Service**: Manages branch-specific operations and data.
4. **Data Storage**:
   * **NoSQL Database**: MongoDB for large-scale storage of analytics and logging data.
5. **Authentication and Authorization Module**:
   * JWT tokens generated and validated here, ensuring secure access control across services.

**b. Key Components**

1. **User Interface (UI)**: Provides tailored dashboards and interfaces for each user role, allowing interactions with different services based on permissions.
2. **API Gateway**: Routes and manages all incoming requests, enforces security, and provides a unified API entry point for the UI.
3. **User Service**: Responsible for managing user accounts, roles, and JWT authentication, enabling secure role-based access.
4. **Product Service**: Manages product catalog, inventory, and restocking functions. Supports branch-specific product data for inventory management.
5. **Order Service**: Handles order creation, processing, and invoicing. Integrates with other services for inventory checks and customer data.
6. **Customer Service**: Manages customer profiles, providing relevant data for Sales Reps and Retailers to enhance customer experience.
7. **Branch Service**: Manages branch-specific operations, such as analytics for Branch Retailers and inventory restocking for each location.
8. **Data Storage**:
   * **NoSQL Database - MongoDB** stores unstructured and large-scale analytics data, enabling fast access to reporting information.
9. **Authentication and Authorization Module**: Generates and validates JWT tokens, enforcing secure access across all services based on user roles.

**5. Technical Stack**

**a. Frontend**

* **Framework**: **React with TypeScript** – React is used to build a responsive, modular frontend with TypeScript for improved type safety and maintainability.
* **Styling**: **Tailwind CSS** – A utility-first CSS framework that enables rapid UI development with a consistent, responsive design.
* **Wireframing**: **Figma** – Used for designing and prototyping UI/UX wireframes, ensuring the frontend design aligns with user needs and provides a clean, intuitive layout.

**b. Backend**

* **Framework**: **Node.js with TypeScript** – Provides a scalable, non-blocking backend environment. TypeScript enhances code readability and stability.
* **REST APIs**: Developed to manage communication between microservices and to facilitate interaction with the frontend. Each microservice (User, Product, Orders, Customer, Branch) has its own set of APIs tailored to specific functionalities, enabling modular, maintainable code.

**c. Database**

* **Database**: **MongoDB Atlas** – A managed cloud version of MongoDB that ensures scalability and high availability. MongoDB’s NoSQL document model is well-suited for flexible, dynamic data structures, such as user data, product catalogs, and order details.
* **Design Considerations**:
  + **Collections**: Separate collections for each service (e.g., Users, Products, Orders, Branches, Customers) to allow faster queries and optimized access.

**d. Other Technologies and Tools**

* **Version Control**: **Git** – Used for source code management, enabling collaboration, version tracking, and efficient branching strategies.
* **Cloud Database**: **MongoDB Atlas** – Managed database service that provides automated backups, monitoring, and scaling, freeing up resources for other development tasks.
* **JWT (JSON Web Token)**: For secure, stateless user authentication and role-based authorization.

**6. Prerequisites and Requirements**

**a. Technical Requirements**

1. **Hardware**:
   * Development machines with at least 8GB RAM and a modern processor for running frontend and backend services smoothly.
2. **Software and Development Environments**:
   * **Frontend**: Node.js environment (latest stable version) for running React applications, TypeScript, and necessary libraries.
   * **Backend**: Node.js and TypeScript setup, with Express framework and npm for dependency management.
   * **Database**: Access to MongoDB Atlas for managed cloud database hosting.
3. **Cloud Services**:
   * **MongoDB Atlas** for database management.

**b. Data Requirements**

1. **Datasets**:
   * **User Data**: Sample data for users with various roles (Business Retailer, Branch Retailer, and Sales Rep) to test role-based access control.
   * **Product Data**: Initial product dataset to verify catalog and inventory management features.
   * **Branch Data**: Sample branch data to set up analytics and branch-level operations.
   * **Order Data**: Test data for order processing and invoice generation, essential for verifying the order service functionality.
   * **Customer Data:** Sample customer data to test customer management, order history tracking.

**c. Access Permissions**

1. **Source Code Repository**: GitHub repository access for team collaboration and version control.
2. **Database Access**: Administrator-level access to MongoDB Atlas for managing database configuration, permissions, and data replication settings.

**d. Other Dependencies**

1. **Libraries and Plugins**:
   * **Frontend**: React, Tailwind CSS, TypeScript, Axios for API calls, JWT library for authentication.
   * **Backend**: Express, TypeScript, Mongoose for MongoDB integration, JSON Web Token (JWT) for authentication.
2. **Versions**:
   * Node.js and MongoDB.

**7. Future Improvements**

1. **Advanced Analytics and Live Data Integration**:
   * Integrate live data feeds for real-time insights, such as inventory levels, sales trends, and customer behavior.
   * Implement a more robust analytics dashboard with interactive charts and visualizations tailored to different user roles.
2. **Enhanced Role-Based Permissions**:
   * Expand role-based access control by adding more granular permissions within each user role.
   * Allow custom roles for businesses with specific requirements, providing flexibility to control access to resources.
3. **Integration with Third-Party Services**:
   * Connect with external ERP, CRM, and payment systems to streamline business processes and enhance operational efficiency.
   * Integrate with messaging platforms for sending notifications, order confirmations, and alerts.
4. **Enhanced Reporting and Export Options**:
   * Offer downloadable reports in various formats (PDF, CSV, Excel) to facilitate data sharing and compliance.

**8. Conclusion**

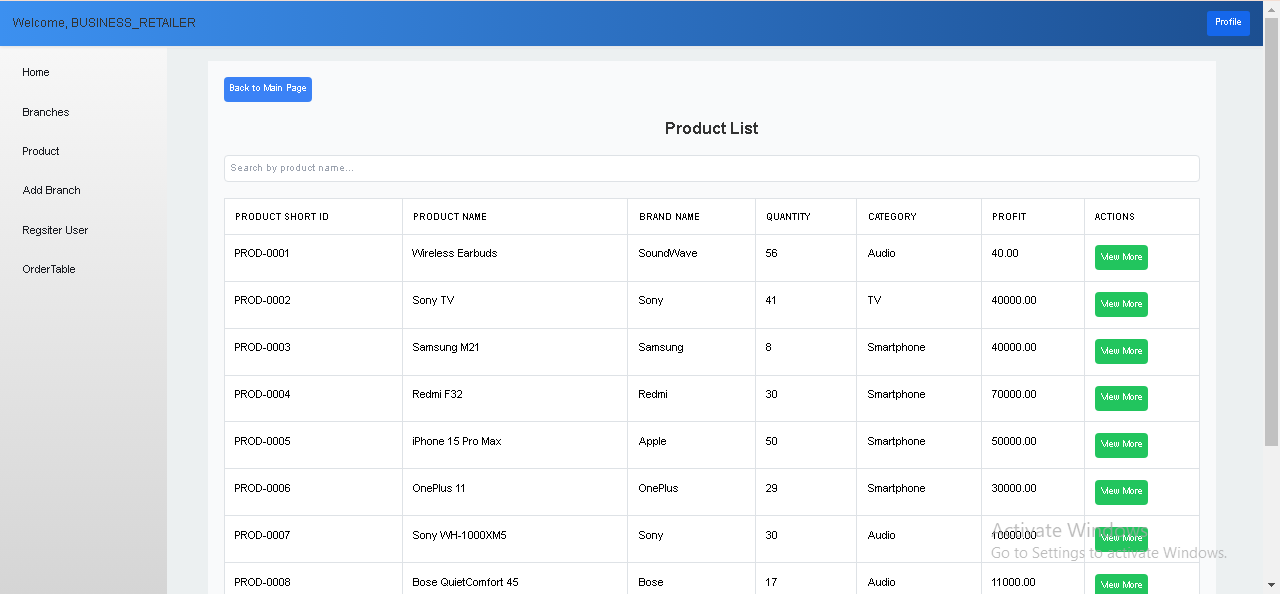
**a. Summary of Achievements**

* **Modular Microservices Architecture**: Successfully implemented a microservices-based solution with distinct services (User, Product, Orders, Customer, Branch) that offers flexibility, scalability, and easier maintenance.
* **Role-Based Access and Security**: Established a robust role-based authentication and authorization system using JWT, allowing for secure access control among three user roles (Business Retailer, Branch Retailer, and Sales Rep).
* **Custom Analytics and Reporting**: Created an analytics dashboard tailored to user roles, enabling Business Retailers to view organization-wide data, Branch Retailers to view branch-specific metrics, and Sales Reps to focus on sales orders and invoicing.
* **End-to-End Order and Inventory Management**: Implemented features for viewing products, managing branches, placing orders, restocking inventory, and generating invoices, providing comprehensive support for business operations.
* **Scalable Backend and Cloud Database**: Utilized Node.js with MongoDB Atlas, ensuring a backend capable of handling significant growth in users and data volume, as well as enhanced resilience through cloud-based hosting.

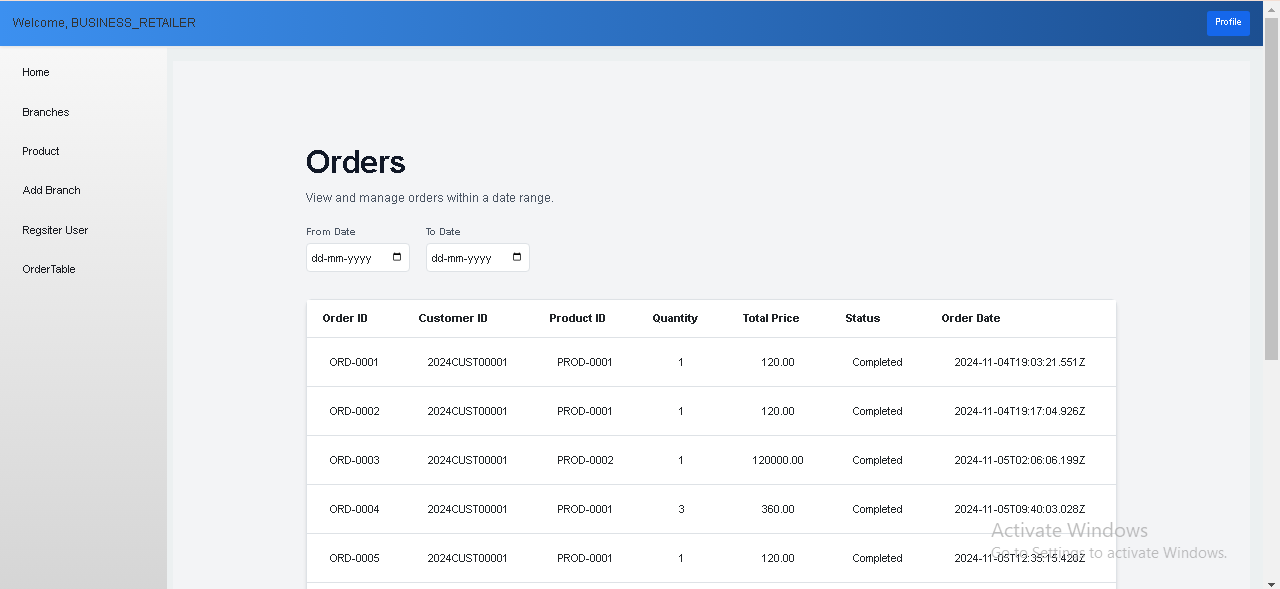
**b. Value Provided**

* **Enhanced Operational Efficiency**: The system automates various business processes, such as inventory restocking, order creation, and invoice generation, reducing manual workload and improving efficiency.
* **Real-Time, Role-Based Insights**: The solution provides tailored analytics and data visibility for different user roles, enabling informed decision-making at all levels within the organization.
* **Improved Scalability and Flexibility**: The microservices architecture allows each service to be independently scaled and modified as business needs evolve, ensuring longevity and adaptability of the system.
* **Secure and Reliable Access**: With JWT-based role management, users can securely access and manage resources, which builds trust and protects sensitive business data.

**Business Retailer View**

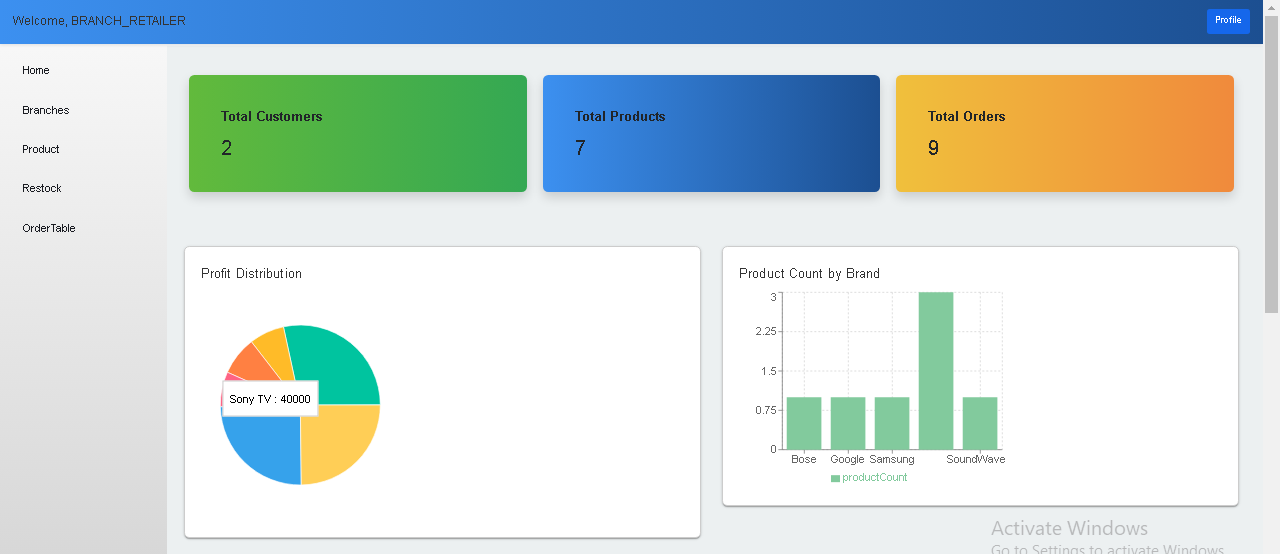


Product List

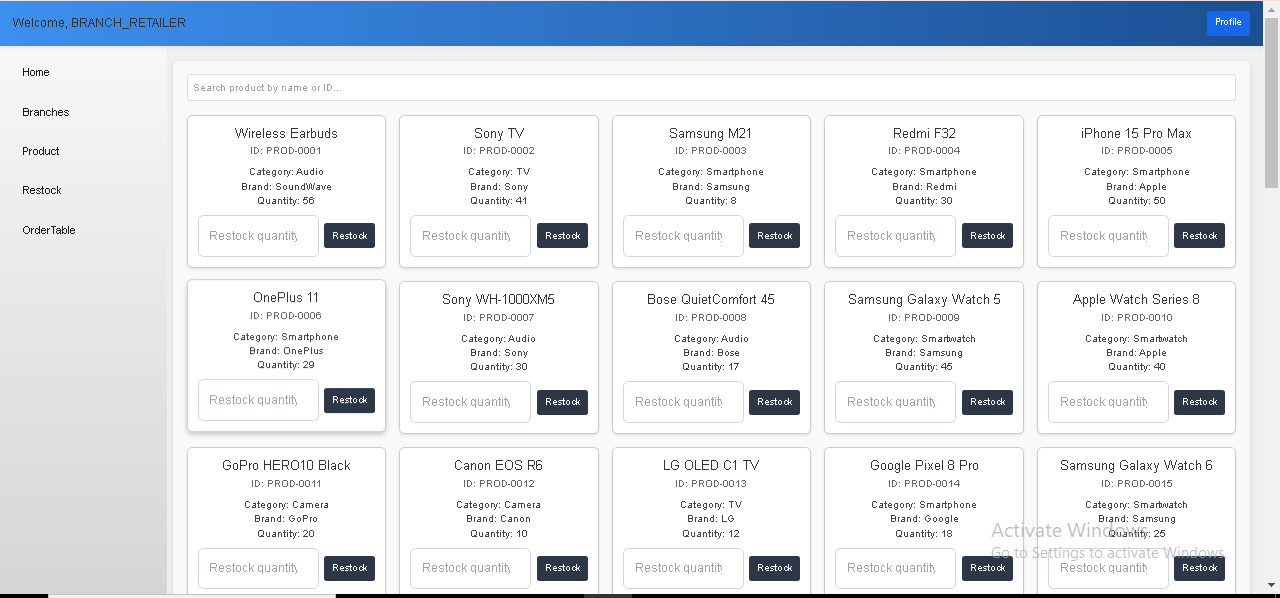


List of Orders

**Branch Retailer View**

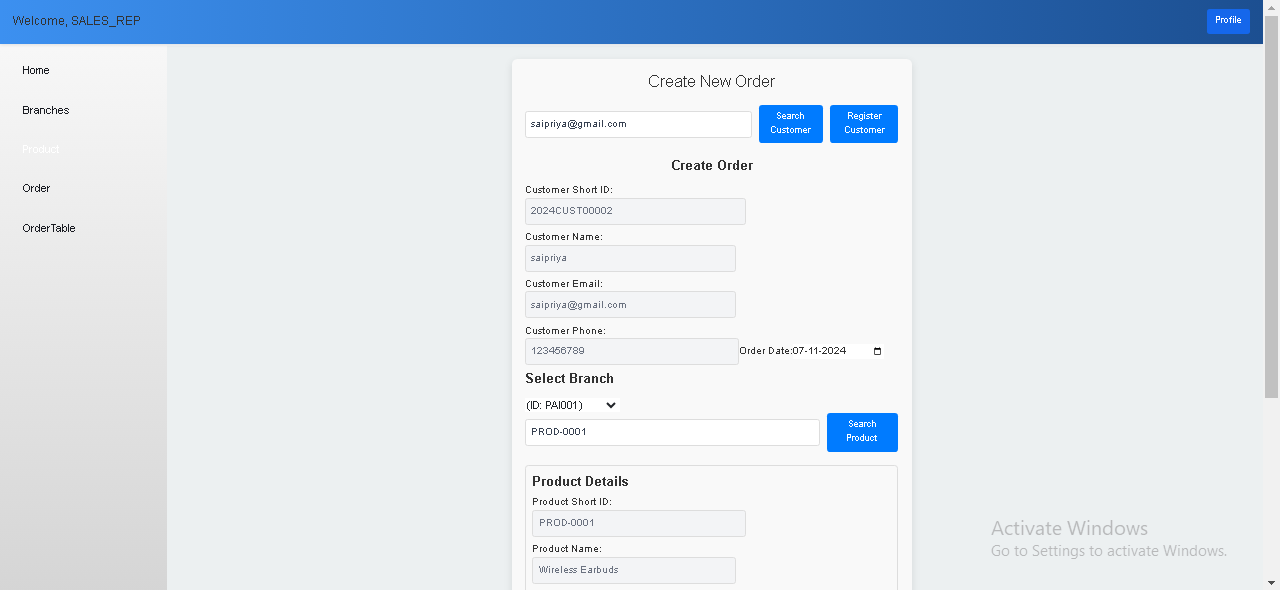


Dashboard for Branch Retailer

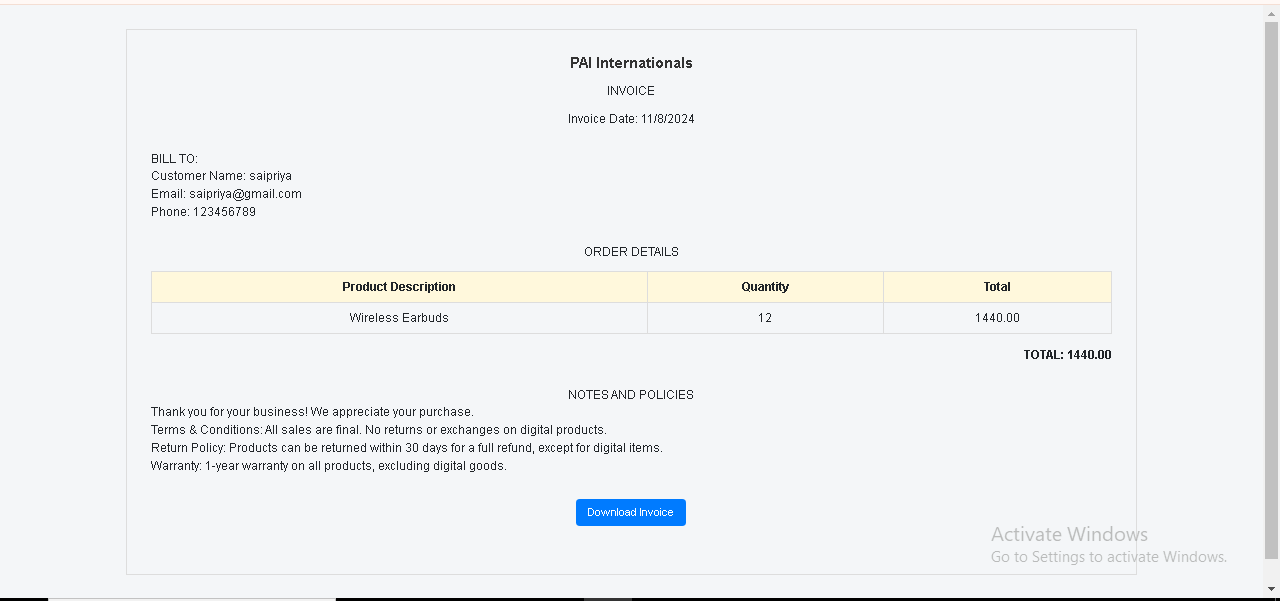


Update restock Quantity

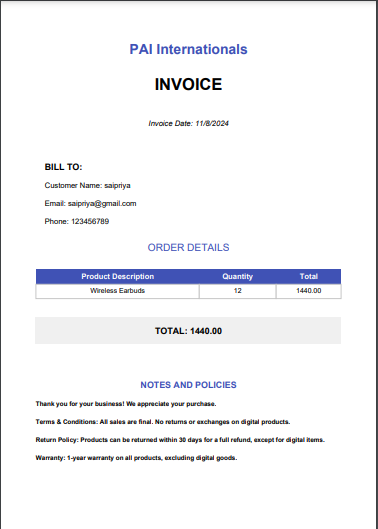
**Sales Rep View**



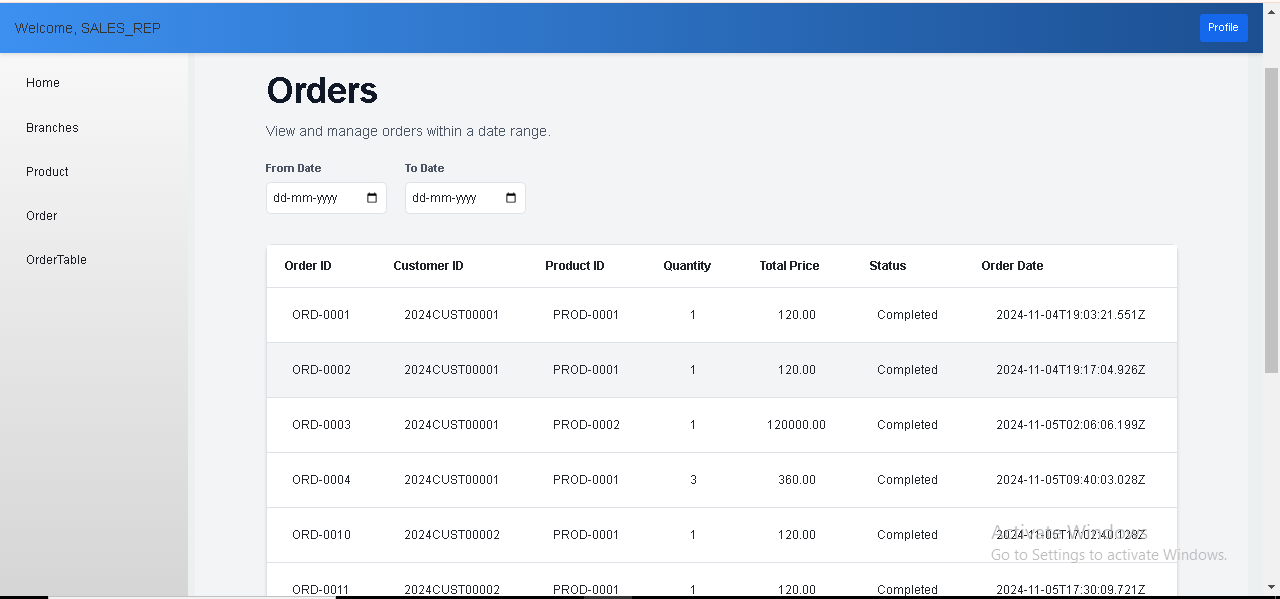
Order Creation



View Invoice



Invoice Generated



List of Orders