VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590 018

SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
MINI PROJECT REPORT

On

JIT Flow Inventory Management System

Submitted in partial fulfillment of the requirement for the curriculum of the 5th Semester

Bachelor of Engineering
in
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted by

BALAJI R	**:**	**1VI23CD008**
NITHIN P GOWDA	**:**	**1VI23CD029**
PRAJWAL S K	**:**	**1VI23CD032**
SHASHANK N	**:**	**1VI23CD050**

Under the guidance of

Prof. Kohilambal R
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

VEMANA INSTITUTE OF TECHNOLOGY

BENGALURU – 560034

2025 - 2026

Karnataka ReddyJana Sangha®
VEMANA INSTITUTE OF TECHNOLOGY
Approved by AICTE - New Delhi, Affiliated to VTU Belagavi & Recognized by Govt. Of Karnataka
#1, Mahayogi Vemana Road, 3rd Block, Koramangala, Bengaluru - 560034.
www.vemanait.edu.in
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

CERTIFICATE

This is to certify that the Software Engineering project entitled "JIT Flow Inventory Management System" is a bonafide work carried out jointly by Balaji R(1VI23CD008), Nithin p Gowda(1VI23CD029), Prajwal S K (1VI23CD032) and Shashank N(1VI23CD050), during the academic year 2025-26 in partial fulfillment of the requirement for the 5th Semester course work of the Bachelor of Engineering in Computer Science and Engineering (Data Science) of the Visvesvaraya Technological University, Belagavi. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The activity report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said degree.

Guide	HOD	Principal
Mrs. Kohilambal R	Dr. Mamatha CR	Dr. Vijayasimha Reddy B G

ACKNOWLEDGEMENT

We sincerely thank **Visvesvaraya Technological University** for providing a platform to do a mini-project.

Firstly, we would like to express our deep sense of gratitude to our institute "**Vemana Institute of Technology**" that provided us an opportunity to do a mini-project entitled "**JIT Flow Inventory Management System**".

We thank **Dr. Vijayasimha Reddy B. G,** Principal, Vemana Institute of Technology, Bengaluru for providing the necessary support.

We would like to place on record our regards to **Dr. Mamatha CR,** Associate Professor & Head of the Department, Computer Science and Engineering (Data Science) for his continued support.

We would like to thank our project guide **Mrs. Kohilambal R**, Assistant Professor, Department of Computer Science and Engineering (Data Science) for her continuous support and valuable guidance towards successful completion of the project.

We would also like to thank all the faculty members, batch mates, lab staff, technicians and family members for their constant support and guidance.

BALAJI R: (1VI23CD008)
NITHIN P GOWDA:(1VI2CD029)
PRAJWAL S K: (1VI23CD032)
SHASHANK N: (1VI23CD050)

Date: 10/11/2025
Place: Bengaluru

ABSTRACT

The JIT Flow Inventory Management System is a web-based application designed to automate inventory and order management for businesses. It replaces manual processes such as paper stock registers and Excel-based tracking, which often lead to stock errors, delays, and poor visibility of available items. The system provides a digital platform that enables quick item registration, real-time stock updates, purchase order tracking, and role-based access for administrators, inventory managers, and other authorized users.

Developed using React, TypeScript, Tailwind CSS, and Supabase as the backend service, the application allows users to manage product catalogs, track incoming and outgoing stock, monitor low-stock alerts, and analyze inventory metrics through interactive dashboards. It offers a clean, responsive interface and ensures secure, organized handling of inventory, supplier, and transaction data via a centralized database.

The project reduces operational workload, improves accuracy in stock records, and supports digitalization of supply chain processes. It follows Agile methodologies, using Git and GitHub for iterative development, continuous feedback, and effective project management throughout the lifecycle of the system.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background and Problem Domain

The JIT Flow Inventory Management System is developed to simplify and automate the day-to-day inventory and purchase-order tasks carried out in businesses and warehouses. Traditionally, stock-related information such as item details, quantities, purchase orders, supplier data, and transaction history is maintained manually using registers or spreadsheets. These methods are time-consuming, prone to errors, and difficult to manage as the number of items, suppliers, and orders increases. Organizations require a digital system that can efficiently store, update, and retrieve inventory information in real time without delay.

This project provides a centralized platform that integrates essential modules including Item and Category Management, Supplier Management, Purchase Order Tracking, and Stock Movement Recording. The system is built using React and TypeScript for the interface, with Supabase as the backend database and service layer, enabling quick data access, real-time updates, and easy modifications through a web browser. Each module offers core functionalities such as adding, updating, searching, deleting, and listing inventory and supplier records, making the system highly flexible and user-friendly for managers and staff.

The JIT Flow Inventory Management System reduces operational workload, prevents stock calculation errors, and ensures accurate and transparent material tracking. By digitalizing core inventory operations, the system improves efficiency, enhances data security, and supports faster procurement and planning decisions. Agile development practices, supported by Git and GitHub, were followed to enable continuous improvement, modular design, and effective implementation throughout the project lifecycle

1.2 Proposed Solution:

The proposed JIT Flow Inventory Management System aims to replace traditional manual stock registers and spreadsheet-based tracking with a fully digital, automated platform. Manual inventory handling often leads to inaccurate stock counts, delayed updates, and difficulties in managing growing item and supplier data. By introducing a web-based system, the organization can maintain consistent, error-free, and easily retrievable inventory information across all categories and operations. The system ensures that key inventory and procurement tasks are streamlined for improved efficiency and better decision-making. The system consists of multiple integrated modules, including Item and Category Management, Supplier Management, Purchase Order Management, Stock Movement Tracking, and a central Dashboard with role-based Login. The Item module stores essential details such as item name, SKU, category, unit price, GST/HSN information, and current quantity.

The Supplier module maintains supplier profiles, contact details, and performance information. The Purchase Order module records order number, items, quantities, costs, order dates, and delivery status, while Stock Movement tracking captures all incoming and adjustment transactions so that available quantities are always up to date. Each module supports operations to add, display, update, delete, and search records easily.

To provide a clean and user-friendly interface, the front end is built using React and TypeScript, styled with Tailwind CSS and shadcn/ui components. This combination enables responsive pages for dashboards, tables, and forms, along with smooth navigation between Inventory, Orders, Suppliers, and Analytics screens. Form-based inputs with validation ensure that data entry is structured and controlled, reducing user errors and improving the overall efficiency of inventory operations.

The backend of the system is powered by Supabase, which provides a hosted PostgreSQL database, authentication, and APIs. Inventory data such as items, suppliers, purchase orders, and stock movements is stored in relational tables, enabling fast querying, filtering, and aggregation. Supabase Edge Functions written in TypeScript can be used for more complex server-side logic like recalculating stock, enforcing security rules, or generating summary metrics. This architecture offers scalability and reliability while avoiding the need to manage a custom server.

The development of this system follows Agile methodologies, focusing on incremental progress, frequent testing, and continuous refinement. Features were built in small iterations using Git and GitHub for version control, branch-based development, and code reviews. This iterative approach ensures that the final JIT Flow Inventory Management System is stable, user-friendly, and aligned with real-world inventory and supply-chain processes.

1.3 Significance and Motivation

The JIT Flow Inventory Management System is significant because it digitalizes core inventory and procurement operations such as item management, supplier records, purchase orders, and stock tracking. By replacing manual registers and spreadsheet-based workflows with an automated web system, it improves accuracy in stock records, reduces operational workload, and ensures faster access to real-time inventory data for decision-making. Built using React, TypeScript, Tailwind CSS, and Supabase, the system provides a smooth, secure, and efficient way to manage materials and supplier information.

The motivation for developing this system comes from the need to overcome stock mismatches, delayed replenishment, and data mismanagement commonly seen in manual inventory workflows. Using Agile methodologies for this project enabled continuous improvement, iterative feature development, and quick adaptation to feedback from realistic use cases. This helped the system evolve into a more refined, user-friendly, and practically relevant solution for JIT-style inventory management.

1.4 Project Objectives

The primary objective of this project is to create a digital platform that automates and simplifies the major inventory and procurement operations of an organization. The key objectives include:

- To digitalize core inventory operations by replacing manual stock registers and spreadsheets with an automated web system.
- To provide a user-friendly interface for managing items, categories, suppliers, purchase orders, and stock movements.
- To ensure secure and efficient data storage using Supabase (PostgreSQL), enabling fast retrieval, filtering, and updates of inventory records.
- To automate calculations such as current stock levels, total inventory value, low-stock detection, and basic performance indicators.
- To support complete CRUD operations (Create, Read, Update, Delete) across all modules, including items, suppliers, and orders.
- To implement a role-based login system for secure and controlled access for admins and inventory users.
- To reduce operational workload by simplifying data entry, minimizing stock errors, and improving visibility of real-time inventory status.

1.5 Scope and Limitations

INCLUSION

- **User Authentication:**
  Secure login system with role-based access for administrators and inventory users.
- **Inventory Management:**
  Ability to add, edit, update, search, and delete item records, including SKU, category, price, GST/HSN, and current stock levels.
- **Purchase Order Management:**
  Creation and tracking of purchase orders with automatic calculation of quantities, costs, and order status (e.g., pending, in transit, delivered).
- **Supplier Management:**
  Handling supplier details, contact information, and basic performance data for reliable sourcing.
- **Dashboard and Analytics:**
  Central dashboard to view total inventory value, low-stock alerts, active orders, and key inventory metrics.
- **Centralized Navigation:**
  A main layout that allows users to easily access modules such as Dashboard, Inventory, Orders, and Suppliers.
- **CRUD Operations:** Full support for Create, Read, Update, and Delete on items, suppliers, and orders for efficient data management.

Exclusions (Limitations):

The current version of the JIT Flow Inventory Management System has the following limitations:

- **No Online Procurement Integration:** Integration with external supplier portals or ecommerce platforms for automatic ordering is not supported.
- **No Barcode / QR Code Scanning:** Physical barcode or QR code scanning for items is not implemented.
- **No Multi Warehouse Support:** Inventory is managed for a single location; multi warehouse or branch-level tracking is not available.
- **Limited Automation:** Advanced features such as fully automated reordering, predictive demand forecasting, and cost optimization algorithms are not yet implemented.
- **Limited Notification System:** Email/SMS/app notifications for low stock, delayed deliveries, or approvals are not included in this version.

6. Organization of the Report

This report is organized into six chapters:

- **Chapter 1:** Introduction, background, motivation, objectives, and scope.
- **Chapter 2:** technologies used
- **Chapter 3:** project objectives
- **Chapter 4:** System design, including architecture, database design..
- **Chapter 5:** Implementation methodology , technology stack, and development details..
- **References and Appendices**

Chapter 2

TECHNOLOGIES USED

Developing a secure, scalable, and efficient JIT Flow Inventory Management System requires the integration of modern web technologies, cloud-based backend services, and development tools. This chapter explains the various technologies used in the system, along with their roles and reasons for selection. The stack spans frontend, backend (Supabase), database, and developer tools, all contributing to the reliability, performance, and usability of the project.

2.1 Frontend Technologies

2.1.1 React 18

React 18 is used as the core frontend library to build the single-page JIT Flow application. It provides a component-based structure for implementing dashboards, forms, tables, and navigation, ensuring fast rendering and a smooth user experience.

2.1.2 TypeScript

TypeScript adds static typing on top of JavaScript, helping to catch errors at compile time and making complex inventory, order, and supplier data models easier to manage. It improves code reliability, readability, and maintainability in a growing project.

2.1.3 Tailwind CSS

Tailwind CSS is used for styling the interface and maintaining a consistent design across all screens. Utility classes control layout, spacing, colors, and typography, enabling a responsive UI that works well on desktops, tablets, and mobile devices.

2.1.4 shadcn/ui

shadcn/ui provides a rich set of pre-built, accessible React components that speed up UI development for the JIT Flow Inventory Management System. It is used to build consistent and responsive interfaces such as dashboards, forms, dialogs, tables, and navigation elements, while working seamlessly with Tailwind CSS for styling.

2.1.5 TypeScript

TypeScript (a superset of JavaScript) enables rich client-side interaction and enhances user experience in the JIT Flow Inventory Management System by providing:

- Type-safe dynamic content updates on dashboards and tables.
- Reliable form validation for inventory, supplier, and order entries.

2.2 Backend Technologies

2.2.1 Supabase (Backend as a Service)

Supabase is used as the backend platform to handle authentication, database access, and API calls for the JIT Flow Inventory Management System. Key functionalities implemented using Supabase include:

- User login and role-based authentication for admins and inventory users.
- CRUD operations on items, suppliers, purchase orders, and stock movement data.
- Real-time reading and updating of inventory records stored in the PostgreSQL database.
- Secure access to backend data through auto-generated REST APIs and client libraries.

2.3 Database Technology

2.3.1 Supabase PostgreSQL

- Supabase uses PostgreSQL as the backend relational database to securely store:
- Item and category information
- Supplier details
- Purchase orders and stock movement records
- User accounts and authentication data

PostgreSQL provides strong querying capabilities, constraints, indexing, and fast data retrieval, ensuring data integrity and consistency for all inventory operations.

2.4 Server Environment

Because Supabase is a Backend-as-a-Service platform, the server environment (APIs, authentication, and database hosting) is managed in the cloud. The React + Vite frontend communicates with Supabase through secure HTTPS APIs, so no separate Apache/XAMPP server is required.

2.5 Development Tools

2.5.1 Visual Studio Code (VS Code)

VS Code is used as the primary editor for writing React, TypeScript, and Tailwind CSS code. Useful extensions include:

- TypeScript/JavaScript IntelliSense
- Tailwind CSS IntelliSenseIt enhances productivity and reduces coding errors**.**

2.5.2 Git and GitHub

Git provides version control, allowing the project to: Track code changes and history

Create branches for new features and fixes. Collaborate and back up the code using GitHub as the remote repository.

Chapter 3

PROJECT OBJECTIVES

The primary objective of the JIT Flow Inventory Management System is to create a reliable, user-friendly, and fully digitized platform for managing inventory and purchase-order operations in an organization.

3.1 To Digitize Inventory Management Processes

Replace traditional paper-based or spreadsheet-based stock systems with a centralized digital platform that ensures accuracy, reduces effort, and improves accessibility for all inventory data.

3.2 To Implement Secure Role-Based Access

Provide separate login modules for administrators, inventory managers, and (optionally) suppliers with distinct permissions to ensure data confidentiality and controlled access to critical inventory information.

3.3 To Enable Efficient Item and Supplier Data Handling

Allow administrators to add, update, and manage item details, categories, stock levels, and supplier information along with related transaction history.

3.4 To Automate Stock Calculations

Automatically compute current stock levels, reserved quantities, incoming quantities, and low-stock alerts to eliminate manual calculation errors and speed up replenishment decisions.

3.5 To Provide Real-Time Insights and Dashboards

Offer administrators visual analytics on total inventory value, low-stock items, active purchase orders, and item movement trends to support better operational and financial decision-making.

3.6 To Improve Transparency for Students

Enable authorized users to view item details, available quantities, purchase order status, and historical stock movements through an accessible online dashboard.

3.7 To Support Report Generation and Data Export

Allow administrators to export inventory-related data (items, suppliers, purchase orders, stock movements) as CSV files for accounting, auditing, and management reporting purposes.

Chapter 4

SYSTEM OVERVIEW

4.1 System Architecture

The JIT Flow Inventory Management System is designed as a comprehensive web-based platform that simplifies, organizes, and digitalizes the entire inventory and purchase-order process within an organization. It provides two major user modules—Admin Panel and Inventory Dashboard (with optional Supplier View)—each offering dedicated functionalities tailored to the needs of the respective users. This chapter provides a detailed overview of how the system functions, how different modules interact, and how information flows through the application.

4.1 Purpose of the System

The primary purpose of the system is to:

- Provide a centralized platform for storing and managing inventory, supplier, and purchase-order records
- Reduce manual workload and human errors in stock tracking
- Improve the accuracy of inventory and procurement data
- Offer transparency regarding current stock levels and order status
- Enable administrators to track incoming stock, low-stock items, and active orders efficiently
- Ensure easy accessibility through a user-friendly digital interface

The system replaces outdated manual stock registers and scattered spreadsheets with automated records, dynamic dashboards, and secure authentication logic.

4.2 System Modules

The application is divided into two major modules:

4.2.1 Dashboard

The Dashboard is the home screen for administrators and inventory managers. It shows:

- Total inventory value and overall inventory trend.
- System health and alert status.
- Low-stock items that need attention.
- Recent or scheduled purchase orders.

4.2.2 Inventory Module

The Inventory module manages all item-level information. Users can:

- View item cards with SKU, quantity, unit cost, category, and supplier.
- See low/medium stock badges to understand urgency.
- Search items by name, SKU, or category.
- Add new inventory items or update existing ones.

4.2.3 Orders Module

The Orders module handles purchase order creation and tracking. Users can:

- Create new manual or auto-generated purchase orders.
- View order amount, number of items, expected delivery date, and status (pending, auto, etc.).
- Track low-stock reorders and confirm when stock is received.

4.2.4 Suppliers Module

The Suppliers module stores and manages supplier details. Users can:

- Maintain supplier profiles with contact information, address, and communication details.
- View ratings, total orders, and average delivery time for each supplier.
- Use this information to choose reliable suppliers for JIT replenishment.

4.2.5 Alerts Module

The Alerts module shows system notifications related to inventory events. It is used to:

- Inform users about low-stock items, delayed orders, or important system updates.
- Confirm when the system is operating normally (no active alerts).

4.2.6 Auto Orders (Advanced)

The Auto Orders module automates JIT replenishment. Users can:

- Enable automatic reordering when items fall below the reorder point.
- Configure safety stock multiplier and lead-time buffer (days).
- Monitor metrics such as total auto orders, auto order value, and items needing reorder.

4.2.7 Cost Optimization (Advanced)

The Cost Optimization module uses AI to suggest cost-saving actions. It provides:

- Potential and realized savings indicators.
- A list of AI recommendations such as bulk discounts or optimal order quantities.
- Actions to apply or reject each recommendation.

4.2.8 Sustainability (Advanced)

The Sustainability module (from the sidebar) focuses on environmental metrics. Typical usage includes:

- Tracking inventory and logistics decisions that impact carbon footprint.
- Supporting greener choices in sourcing and transportation.

4.2.9 Route Optimizer (Advanced)

The Route Optimizer module calculates efficient delivery routes. Users can:

- Enter origin and destination addresses and pincodes.
- View optimized route distance, time, cost, and estimated CO_2 emissions.
- See a recommended courier or route summary to support logistics planning.

Together, these modules form a complete JIT Flow system that connects inventory, suppliers, orders, automation, and analytics in one integrated web application.

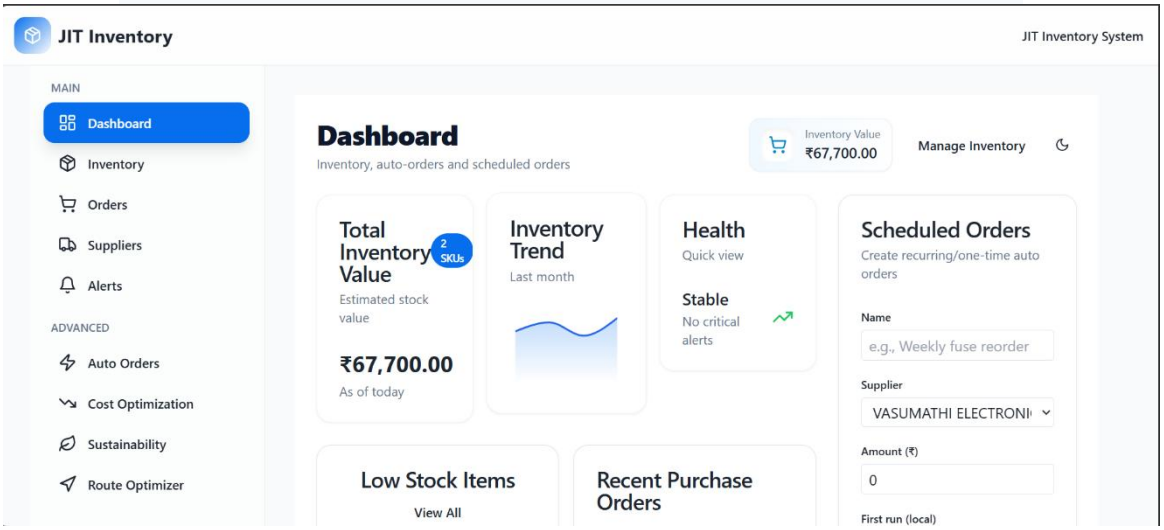


FIGURE 4.1 – Dashboard of the JIT

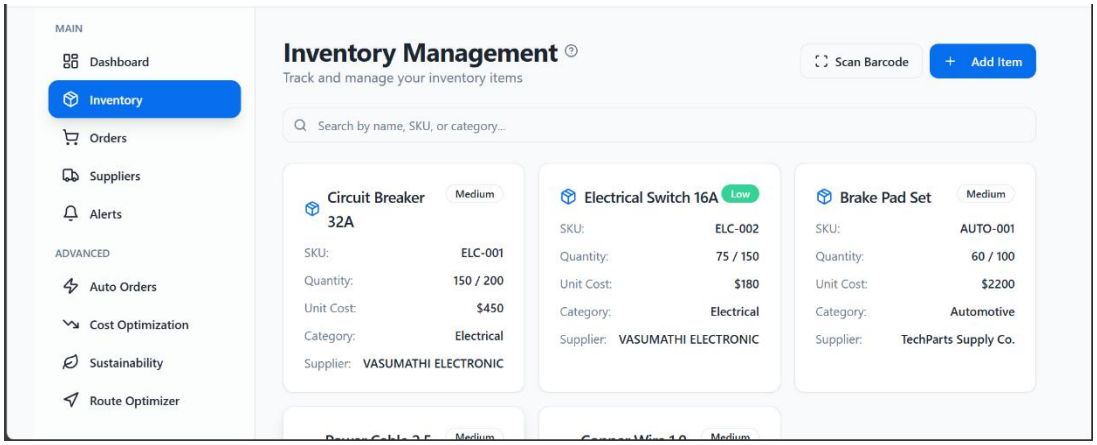


FIGURE 4.2 – Inventory Management

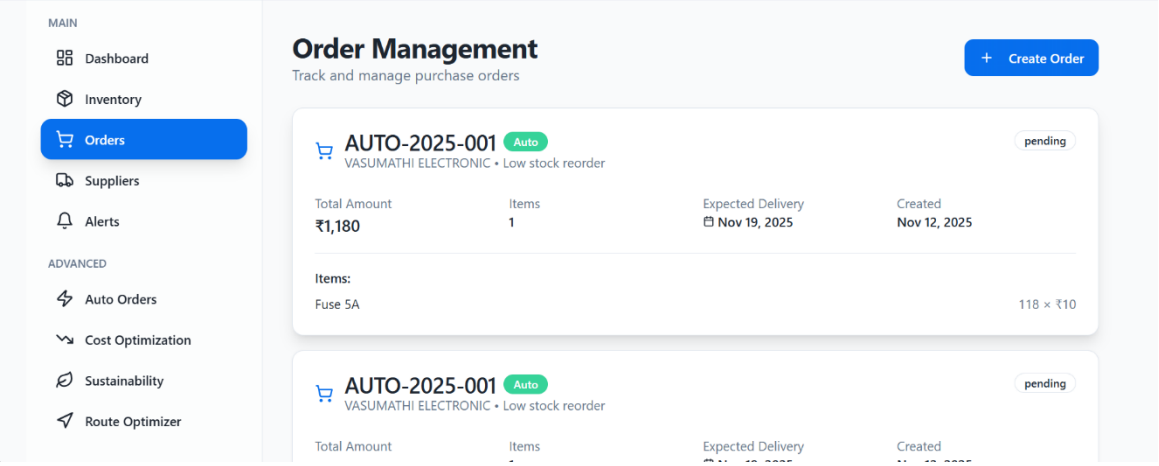


FIGURE 4.3 – Order Management Dashboard

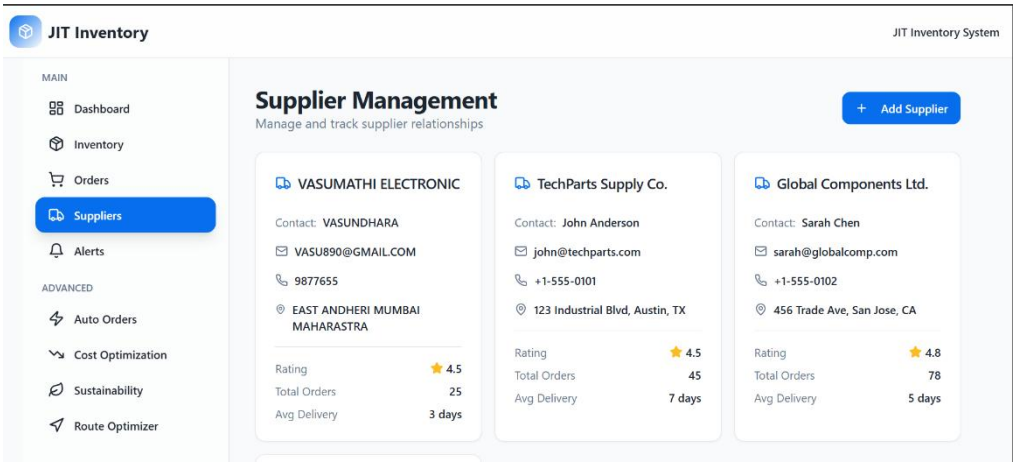


FIGURE 4.4 – Supplier Management Dashboard

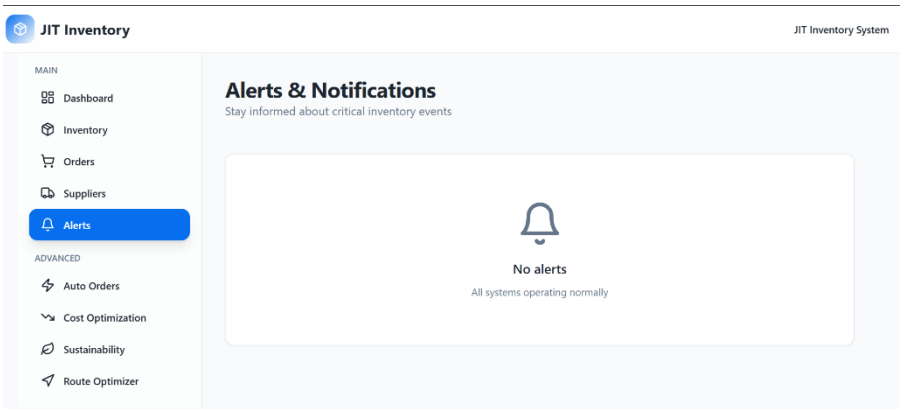


FIGURE 4.5 – Low Stock Alerts

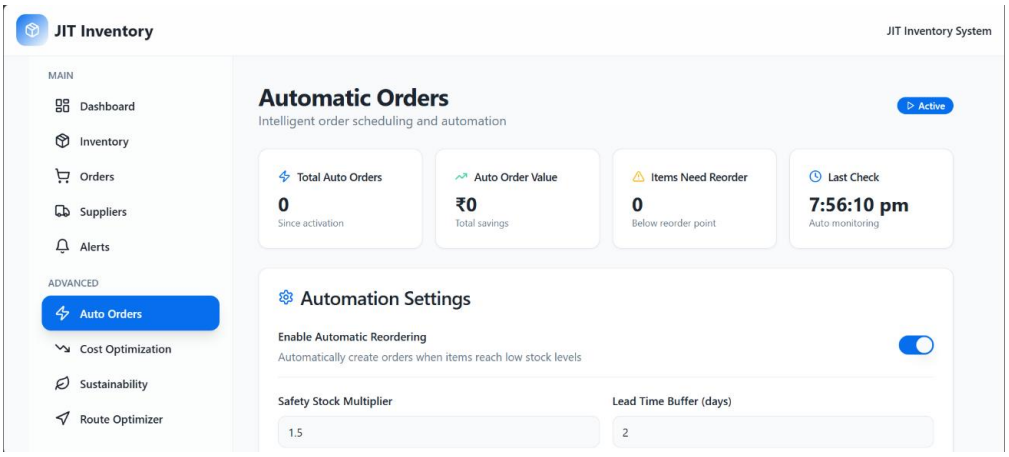


FIGURE 4.6 – Auto Order panel

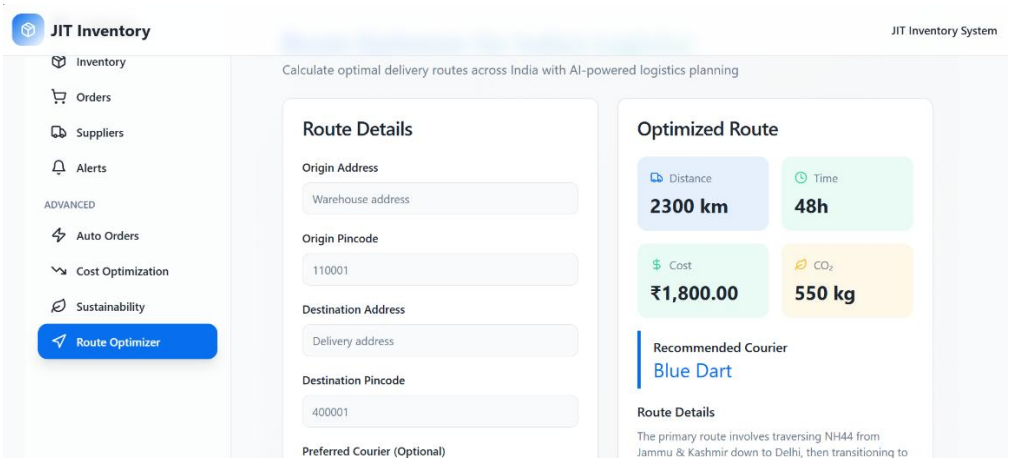


FIGURE 4.7 – Route Optimzer

4.2 Workflow of the System

The overall JIT Flow system workflow follows a clear sequence designed for simplicity and efficiency:

User Authentication

- Admin or inventory user enters login details.
- Credentials are validated through the React frontend and Supabase authentication service.

Redirect to Respective Dashboard

- Admin / Inventory Manager logs in → accesses dashboard, inventory, orders, suppliers, alerts, and advanced modules.

Data Retrieval from Database

- System fetches item, supplier, order, and analytics data from the Supabase (Postgres) database.

Operations & Updates

- Admin updates item details, stock levels, and purchase orders.
- System runs auto-order checks, cost-optimization analysis, and alert rules.

Database Synchronization

- Every update is written back to Supabase.
- All dashboards show near real-time inventory, order, and alert information.

4.4 Advantages of the System

- Centralized digital storage of inventory, supplier, and order data.
- Faster and more accurate stock monitoring and purchasing decisions.
- Reduction in manual paperwork and spreadsheet dependency

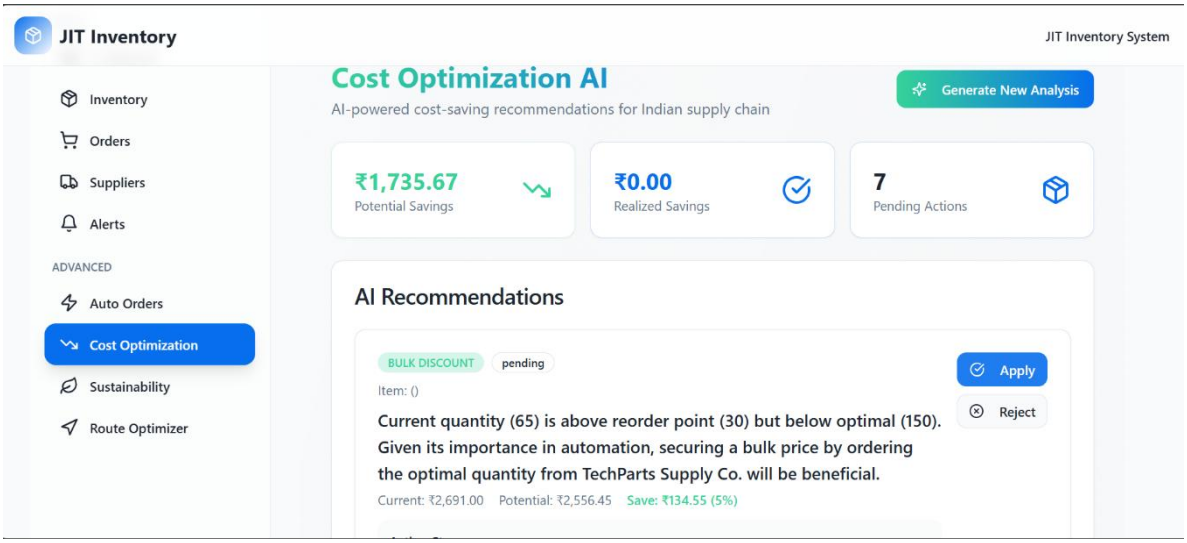


FIGURE 4.8 – Cost Optimization With Ai (Gemini-flash-2.0)

FIGURE 4.9 – Sidebar Navigation For Better Access of the pages

4.5 SYSTEM ARCHITECTURE

The architecture of the JIT Flow Inventory Management System follows a structured, layered model that ensures efficient data processing, smooth user interaction, secure authentication, and reliable storage of operational information. The system is built using a three-tier architecture—Presentation Layer, Application Layer, and Data Layer—improving maintainability, scalability, and ease of adding new features like advanced analytics or mobile access.

4.5.1 Overview of the Architecture

The system supports two main user roles:

- Administrator / Inventory Manager
- (Optional) Viewer-only roles for management or other stakeholders

Regardless of role, all interactions follow the same request–response cycle handled by the React frontend, Supabase APIs, and the Postgres database.

Core components of the architecture include:

- Frontend: React, TypeScript, Tailwind CSS, shadcn/ui, Recharts
- Backend / APIs: Supabase Auth, Supabase REST / Edge Functions
- Database: Supabase Postgres

4.5.2 Three-Tier Architecture

4.5.2.1 Presentation Layer (Frontend Layer)

The Presentation Layer is responsible for everything the user interacts with. It consists of:

- Login forms
- Main dashboard UI
- Inventory, Orders, Suppliers, Alerts, and Advanced (Auto Orders, Cost Optimization, Sustainability, Route Optimizer) pages
- Tables, cards, charts, and forms

This layer is built using:

- React + TypeScript → component structure and behavior
- Tailwind CSS & shadcn/ui → styling and responsiveness
- Recharts → graphs and data visualization

Functions of the Presentation Layer:

- Collects user input (login credentials, item/order forms, filter options).
- Displays stock levels, purchase orders, AI recommendations, alerts, and route results.
- Provides navigation between all modules via the sidebar.

This layer does not store inventory data permanently; it only communicates user actions to the backend and renders responses.

4.5.2.2 Application Layer (Backend Layer)

This is the core processing unit of the system, implemented through Supabase services and Edge

Functions. It acts as a mediator between the React UI and the database.

The Application Layer handles:

- Login authentication and session management via Supabase Auth.
- CRUD operations on items, suppliers, purchase orders, and alerts.
- Business logic for auto-reordering, cost optimization calculations, and route optimization calls.
- Data validation and authorization checks.
- Dynamic aggregation of metrics for dashboards (inventory value, low-stock count, order stats).

Whenever a user interacts with the interface—such as creating an order, updating stock, or running cost optimization—the backend receives the request, performs the necessary logic, and returns updated data to the UI**.**

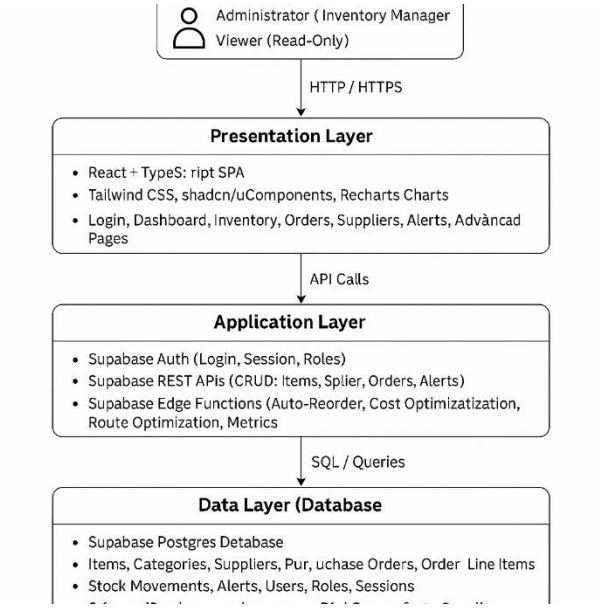


FIGURE 4.10 – Overall Architecture Diagram

4.5.2.3 Data Layer (Database Layer)

The Data Layer consists of the Supabase Postgres database which stores all essential data, including:

- Items and categories
- Suppliers and contact details
- Purchase orders and order line items
- Stock movements and alerts
- User accounts and roles (via Supabase Auth tables)
- The database is designed with:

- Primary keys and foreign keys (e.g., orders linked to suppliers and items).
- Indexing on frequently queried fields (SKU, status, supplier).
- Normalized tables to avoid duplication and maintain consistency.

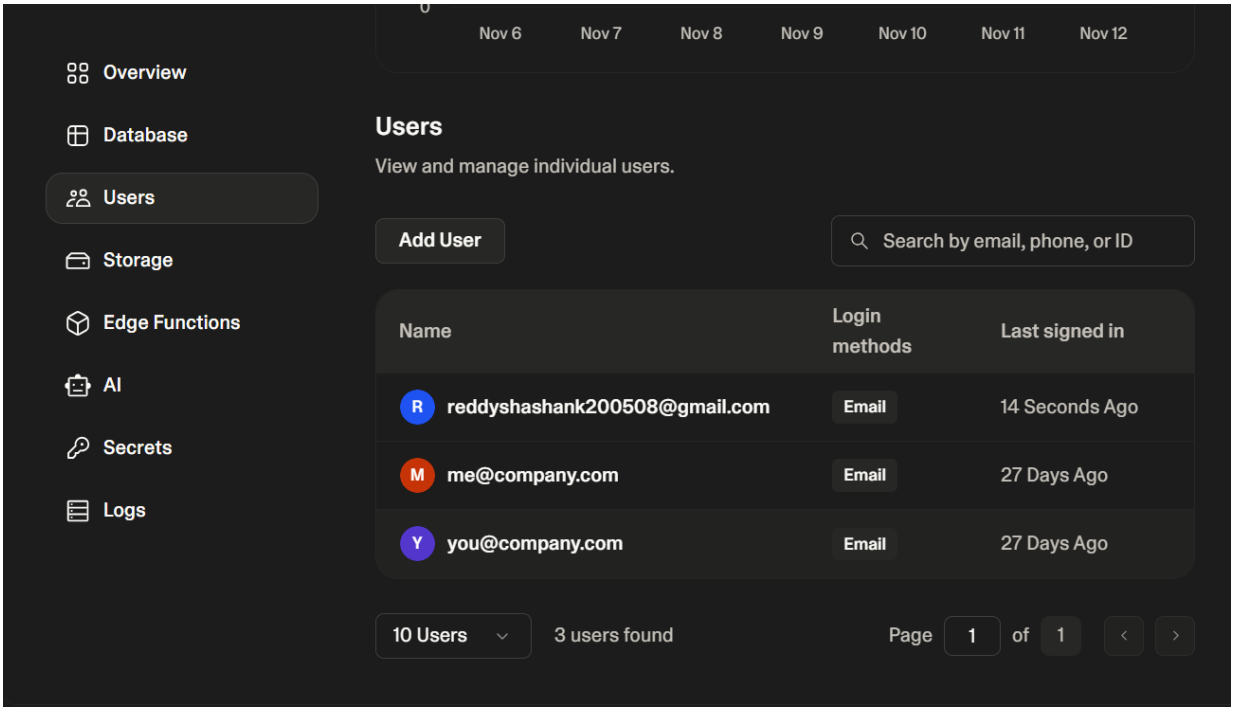


FIGURE 4.11 – Supabase User Database

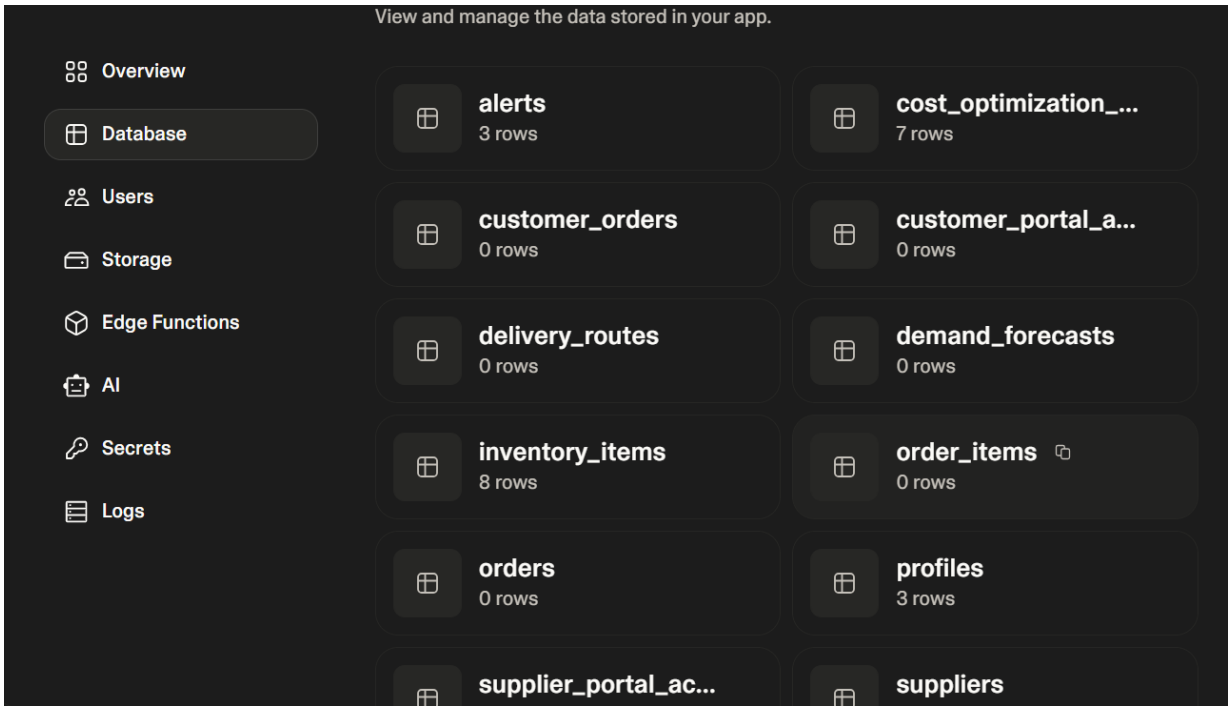


FIGURE 4.11 – Overall Database

4.6 AGILE DEVELOPMENT

The The development of the JIT Flow Inventory Management System followed Agile Software Development Methodology, using short, iterative cycles to deliver working features. Agile made it possible to prioritize core inventory functionality first and then add advanced modules like Auto Orders, Cost Optimization, Sustainability, and Route Optimizer in later sprints.

4.6.1 Agile Workflow Overview

The Agile workflow included stages such as requirement gathering, sprint planning, development, testing, review, and deployment. Each cycle delivered a usable increment—for example, first the basic dashboard and inventory list, then orders and suppliers, then automation features. This iterative approach allowed regular refinement of UI, performance, and usability.

4.6.2 Sprint Planning and Execution

Sprint planning was carried out at the beginning of each cycle. Each 5–7 day sprint focused on a specific module such as:

- Authentication and Dashboard
- Inventory Management
- Order Management
- Supplier Management
- Auto Orders and Alerts
- Cost Optimization, Sustainability, and Route Optimizer

During each sprint:

1. Requirements for that module were refined.
2. Code was developed in feature branches.
3. Unit and UI tests were executed.
4. Features were integrated into the main branch.
5. A sprint review was held to collect feedback and plan improvements.

4.6.3 Daily Scrum Routine

A lightweight scrum routine was followed with short daily check-ins covering:

- What was completed yesterday (e.g., new page, API integration).
- What is planned for today.
- Any blockers (bugs, design doubts, integration issues).

This ensured continuous progress and quick resolution of problems.

4.6.4 Tools Used During Agile Development

- Git & GitHub – Used for version control, branching, pull requests, and issue tracking. Each module or feature was developed on a separate branch and merged after review.
- Visual Studio Code – Main development environment with extensions for React, TypeScript, Tailwind, and Supabase integration.
- ESLint & TypeScript – Enforced coding standards and caught type errors early.

4.6.5 Benefits of Using Agile

- Fast Development: Features were delivered in small, manageable increments.
- Early Testing: Issues were found and fixed early in each sprint.
- Flexibility: Design changes and new ideas (like AI cost optimization) were easily incorporated.
- Better Collaboration: Version control and reviews on GitHub supported teamwork.
- User-Centric Output: Frequent UI iterations led to a clean, easy-to-use JIT inventory interface.

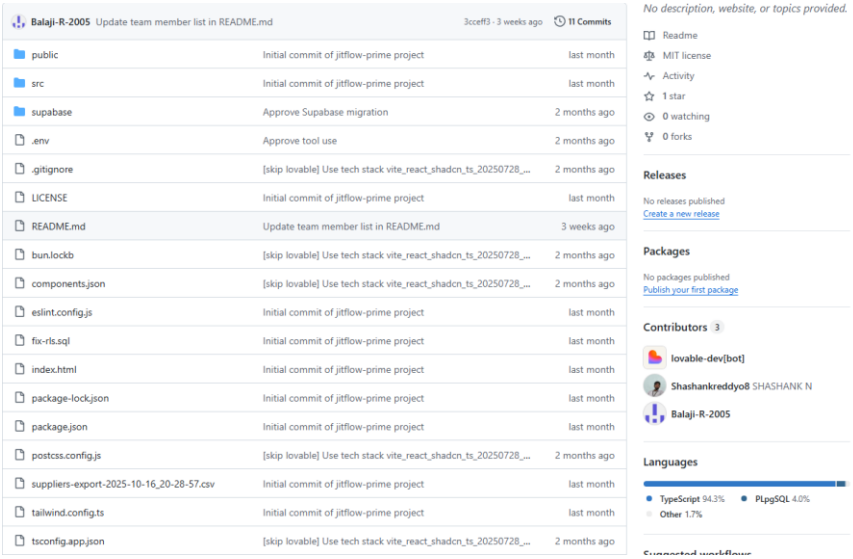


Figure 4.9 : GitHub Repo

4.7 DATABASE DESIGN

The database design of the JIT Flow Inventory Management System uses a relational Postgres model in Supabase to ensure secure, accurate, and efficient data storage. It supports operations such as item management, supplier management, purchase orders, auto-orders, and alerts.

4.7.1 Overview

Key entities stored in the database include:

- Items – product master data, SKU, category, unit cost, current quantity.
- Suppliers – supplier profiles and contact details.
- Purchase Orders – header information for each order (supplier, status, dates, totals).
- Order Items / Stock Movements – line-level quantities and movements affecting inventory.
- Users / Roles – authentication and authorization data managed by Supabase.

These entities form the core data of the system and are linked through foreign-key relation

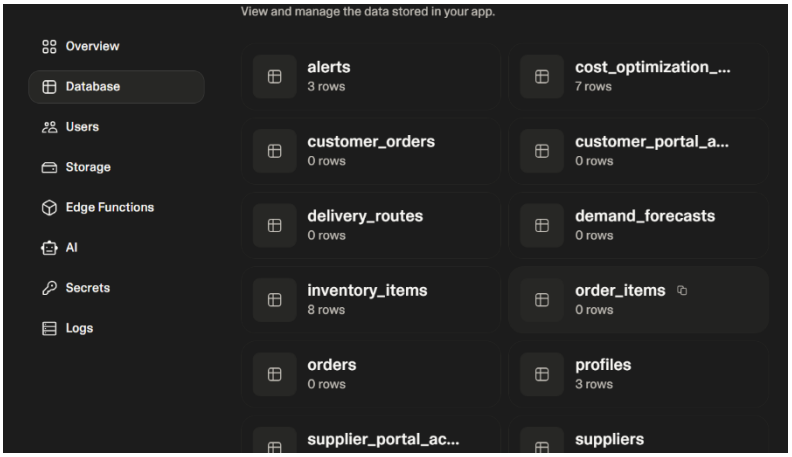


Figure 4.11: Database of SUPABASE

4.7.3 Table Descriptions

a) Items Table

Stores master information about inventory items.

- item_id (PK) – Unique ID for each item
- sku – Unique item code
- name – Item name
- category – Category (Electrical, Automotive, Cables, Wires, etc.)
- unit_cost – Cost per unit
- current_qty – Current available quantity
- reorder_point – Threshold at which auto-orders are triggered

b) Suppliers Table

Stores supplier details.

- supplier_id (PK) – Unique supplier ID
- name – Supplier name
- contact_email, contact_phone – Contact information
- address – Supplier address
- rating – Performance rating
- avg_delivery_days – Average delivery lead time

c) Purchase_Orders Table

Stores purchase order headers.

- po_id (PK) – Unique purchase order ID
- supplier_id (FK) – Linked supplier
- status – Pending, In Transit, Delivered, Cancelled
- total_amount – Total value of the order
- created_at, expected_delivery – Dates for tracking lead time

d)Order_Items / Stock_Movements Tables

Store line-item details and quantity changes.

- line_id (PK) – Line record ID
- po_id (FK) – Linked purchase order
- item_id (FK) – Linked item
- ordered_qty – Quantity ordered
- received_qty – Quantity received
- movement_type – IN, ADJUSTMENT, etc.

4.7.4 Normalization

Tables are normalized (up to Third Normal Form) to ensure:

- No repeating groups of data.
- No partial dependencies on composite keys.
- No transitive dependencies between non-key attributes.

This reduces redundancy and keeps inventory and supplier data consistent.

4.7.5 Summary

The database design provides a robust structure to support all JIT inventory operations. Using Supabase Postgres ensures fast data access, secure storage, strong relational links between entities, and seamless integration with the React + TypeScript frontend. This design enables accurate stock tracking, reliable automation (auto-orders and alerts), and scalable reporting for future growth.

CHAPTER 5

SYSTEM IMPLEMENTAION

The System Implementation phase focuses on converting the planned system architecture, database design, and user interface mockups into a fully functional web-based JIT Flow Inventory Management System. This chapter describes how each module was developed, integrated, tested, and deployed in a structured manner. React with TypeScript was used for the frontend, Supabase provided authentication and database services, and Tailwind CSS/shadcn-ui formed the UI layer.

5.1 Development Environment Setup

The entire system was implemented in a local development environment to allow fast testing and debugging. The following software tools were used:

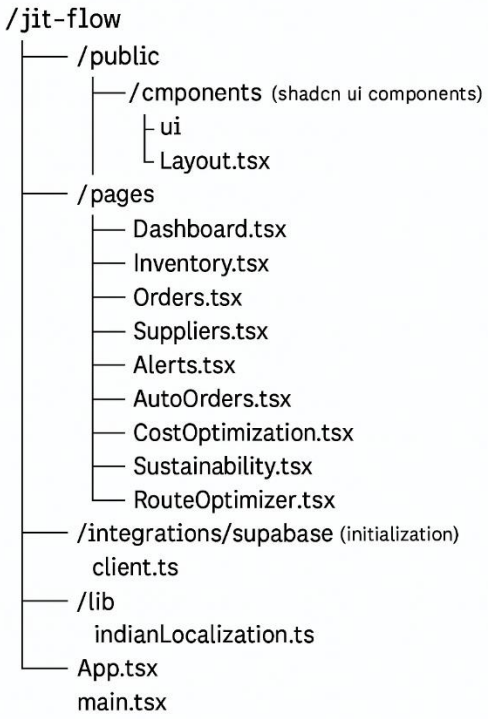
- Node.js and npm – runtime and package manager
- Vite – development server and build tool
- Visual Studio Code – coding environment
- Supabase – hosted Postgres database, authentication, and Edge Functions
- Git & GitHub – version control and collaboration
- Chrome/Edge Browser – runtime testing

The project is run locally with:

- npm install
- npm run dev → served at http://localhost:5173 (Vite default).

5.2 Project Directory Structure

A clean and modular directory structure was maintained for readability and easy updates in the future:



5.3 Backend Integration (Supabase)

5.3.1 User Authentication

- Uses Supabase Auth for email/password sign-in.
- Stores user session in the frontend and protects routes (e.g., dashboard, inventory).
- Only authenticated users can access operational modules.

5.3.2 Inventory, Orders, and Suppliers

- CRUD operations for items, suppliers, and purchase orders are implemented through Supabase client calls.
- Business rules such as low-stock detection and reorder point checks are applied before updates.

5.3.3 Auto Orders and Cost Optimization

- Auto Orders module periodically checks items below reorder point and creates purchase orders using Supabase Edge Functions or scheduled logic.

Cost Optimization uses stored data (prices, quantities, suppliers) to compute potential savings and show AI recommendations on the frontend

5.3.4 Logging and Analytics

- Important events (order creation, stock adjustments) are stored as rows in stock-movement or activity tables.
- Dashboard KPIs (inventory value, low-stock count, active orders) are calculated using aggregate queries from Supabase.

5.4 Database Connectivity

- A central Supabase client file (supabase/client.ts) holds the project URL and public API key.
- All pages and hooks import this client to perform select, insert, update, and delete operations.
- This approach avoids duplicated connection code and makes changing configuration simple.

5.5 Frontend Implementation (React, TypeScript, Tailwind, shadcn-ui)

5.5.1 React + TypeScript

- Each major page (Dashboard, Inventory, Orders, Suppliers, Alerts, Advanced modules) is a React component.
- TypeScript interfaces define shapes for items, suppliers, orders, and recommendations, reducing runtime errors.

5.5.2 Tailwind CSS & shadcn-ui

Used to create:

- Responsive layouts with sidebar navigation and top bars.
- Cards, buttons, tables, badges (e.g., Low stock, Pending).
- Dialogs and forms for adding/editing items, suppliers, and orders.

5.5.3 Recharts and UI Logic

- Recharts renders graphs for inventory trends, savings, and performance metrics.
- JavaScript/TypeScript logic handles client-side validation, filtering, searching, and small UI interactions.

5.6 Admin / Inventory Panel Implementation

5.6.1Dashboard

Displays:

- Total inventory value and overall inventory trend.
- Health status and low-stock items.
- Recent or scheduled purchase orders.

5.6.2 Inventory Management

- Add / Edit / Delete item details.
- Show SKU, quantity, category, unit cost, supplier, and stock status (Low / Medium).

5.6.3 Order Management

- Create and manage purchase orders.
- Display amount, items, expected delivery, and status (pending, auto, delivered).

5.6.4 Supplier Management

- Maintain supplier cards with contact information, rating, total orders, and average delivery time.

5.6.5 Alerts & Advanced Modules

- Alerts page shows important system notifications.
- Auto Orders, Cost Optimization, Sustainability, and Route Optimizer pages use backend data and logic to provide advanced JIT features.

5.7 User Access and Transparency

Authorized users can:

- Log in securely.
- View dashboards and item/order details based on role.
- Check low-stock items, active orders, and AI recommendations. This eliminates the need for manual spreadsheets and repeated phone calls for stock checks.

5.8 Module Integration and Testing

Modules were integrated step-by-step:

- Authentication and routing tested first.
- Supabase connection verified.
- Inventory, Orders, and Suppliers pages integrated with database.
- Auto Orders, Cost Optimization, Sustainability, and Route Optimizer modules added.
- UI responsiveness and behavior tested across devices and browsers.

Testing ensured:

- Smooth navigation between pages.
- Correct calculations of stock, order totals, and savings.
- Consistent database state after create/update/delete actions.
- No broken links or unauthorized access to protected routes.

5.9 TESTING AND VALIDATION

Testing and validation ensured that all implemented modules of the JIT Flow Inventory Management System work correctly, efficiently, and securely. Each component—including Dashboard, Inventory, Orders, Suppliers, Alerts, and advanced automation/analytics modules—was tested to verify functional correctness, error handling, and data integrity.

SOURCE CODE

APP.TSX

```
import { Toaster } from "@/components/ui/toaster";
import { Toaster as Sonner } from "@/components/ui/sonner";
import { TooltipProvider } from "@/components/ui/tooltip";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./components/Layout";
import Index from "./pages/Index";
import Login from "./pages/Login";
import Dashboard from "./pages/Dashboard";
import Inventory from "./pages/Inventory";
import Orders from "./pages/Orders";
import Suppliers from "./pages/Suppliers";
import Alerts from "./pages/Alerts";
import Achievements from "./pages/Achievements";
import HelpCenter from "./pages/HelpCenter";
import NotFound from "./pages/NotFound";
import Sustainability from "./pages/Sustainability";
import RouteOptimizer from "./pages/RouteOptimizer";
import CostOptimization from "./pages/CostOptimization";
import AutoOrders from "./pages/AutoOrders";
import Logout from "./pages/Logout";
import Admin from "./pages/Admin";
import AdminUsers from "./pages/AdminUsers";
const queryClient = new QueryClient();
const App = () => (
  <QueryClientProvider client={queryClient}>
    <TooltipProvider><Toaster />
```

```
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Index />} />
        <Route path="/dashboard" element={<Layout><Dashboard /></Layout>} />
        <Route path="/login" element={<Login />} />
        <Route path="/logout" element={<Logout />} />
        <Route path="/admin" element={<Admin />} />
        <Route path="/admin/users" element={<AdminUsers />} />
        <Route path="/inventory" element={<Layout><Inventory /></Layout>} />
        <Route path="/orders" element={<Layout><Orders /></Layout>} />
        <Route path="/suppliers" element={<Layout><Suppliers /></Layout>} />
        <Route path="/alerts" element={<Layout><Alerts /></Layout>} />
        <Route path="/achievements" element={<Layout><Achievements /></Layout>} />
        <Route path="/help" element={<Layout><HelpCenter /></Layout>} />
          <Route path="/sustainability" element={<Layout><Sustainability /></Layout>} />
          <Route path="/route-optimizer" element={<Layout><RouteOptimizer /></Layout>} />
          <Route path="/cost-optimization" element={<Layout><CostOptimization /></Layout>} />
          <Route path="/auto-orders" element={<Layout><AutoOrders /></Layout>} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </BrowserRouter>
  </TooltipProvider>
 </QueryClientProvider>
);
export default App;
```

REFERENCES

1. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education.
2. Pressman, R. S., & Maxim, B. R. (2015). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.
3. Laudon, K. C., & Laudon, J. P. (2020). Management Information Systems: Managing the Digital Firm (16th ed.). Pearson Education.
4. Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.
5. Supabase. (2025). Supabase Documentation – Auth, Database, Edge Functions, and Client Libraries. **https://supabase.com/docs/**
6. React Team. (2025). React Documentation – React 18, Components, and Hooks. **https://react.dev/**
7. TypeScript Team. (2025). TypeScript Handbook and Reference. **https://www.typescriptlang.org/docs/**
8. Tailwind Labs. (2025). Tailwind CSS Documentation – Utility-First CSS Framework. **https://tailwindcss.com/docs**
9. shadcn. (2025). shadcn/ui Documentation – Re-usable Components for React + Tailwind. **https://ui.shadcn.com/**
10. Recharts. (2025). Recharts – A Composable Charting Library for React. **https://recharts.org/en-US/**
11. GitHub Docs. (2025). Using Git and GitHub for Version Control and Collaboration. **https://docs.github.com/**