CS F213 Project Report

Group-10

- Chaitanya Keyal - 2023B4A70727H
- Arpit - 2023B3A70482H
- Abhishek Verma - 2023B4A71256H
- Vishesh Agarwal - 2023B4A70611H

Website: dashboard.livemart.okaybro.dev

GitHub Repository: github.com/Chaitanya-Keyal/LiveMart

Contents

1 Summary

LiveMart is an e-commerce platform designed to unify the fragmented supply chain of local commerce. Unlike traditional platforms that serve either B2C (Business to Consumer) or B2B (Business to Business) markets, LiveMart integrates **Customers**, **Retailers**, **Wholesalers** and **Delivery Partners** into a single, cohesive ecosystem.

The platform solves a critical problem in the retail industry: the disconnect between local inventory and digital visibility. Retailers struggle to source products efficiently from wholesalers while simultaneously failing to reach nearby customers online. LiveMart bridges this gap through a unified platform that enables seamless role switching, allowing users to operate as customers, retailers, wholesalers, or delivery partners within a single account. The system leverages geolocation-powered discovery using Google Maps integration and Haversine distance calculations to prioritize nearby sellers in search results, ensuring customers find local shops within walking distance while retailers discover wholesalers in their vicinity. Products support tiered pricing based on buyer type, enabling wholesalers to offer bulk discounts to retailers while maintaining higher retail prices for individual customers. Dynamic delivery fees are calculated based on distance to ensure fair pricing.

The platform provides comprehensive business automation including automated financial settlements that track all transactions, calculate platform commissions, and generate detailed payout records for transparent reconciliation. Wholesalers can upload thousands of products via CSV files with automated validation and error reporting, while retailers can clone purchased products into their own catalog with a single click. Secure payment processing through Razorpay with webhook verification ensures no successful payment goes unrecorded, supporting credit cards, UPI, and net banking. Orders progress through eleven distinct states with delivery partner assignment and real-time tracking, accompanied by responsive MJML email notifications at each transition. The system includes an admin dashboard for platform oversight, smart coupon management with usage limits and targeted distribution, verified purchaser reviews and ratings, low stock alerts for inventory management, email verification via OTP codes, and Google OAuth for single-click registration and login.

2 Core Functionality and Workflow

The platform operates on a cyclic commerce model:

1. **Wholesale Supply**: Wholesalers upload bulk inventory (via CSV or manual entry) and set pricing tiers.

2. **Retail Procurement**: Retailers browse wholesale catalogs, benefiting from bulk discounts. They place orders, which are tracked and fulfilled through the platform.

3. **Inventory Transformation**: Once a retailer receives stock, they can "clone" these items into their own retail catalog with a single click, setting a new retail price.

4. **Consumer Purchase**: Customers search for products. The system prioritizes nearby sellers using geolocation. They place orders, pay online, and track delivery.

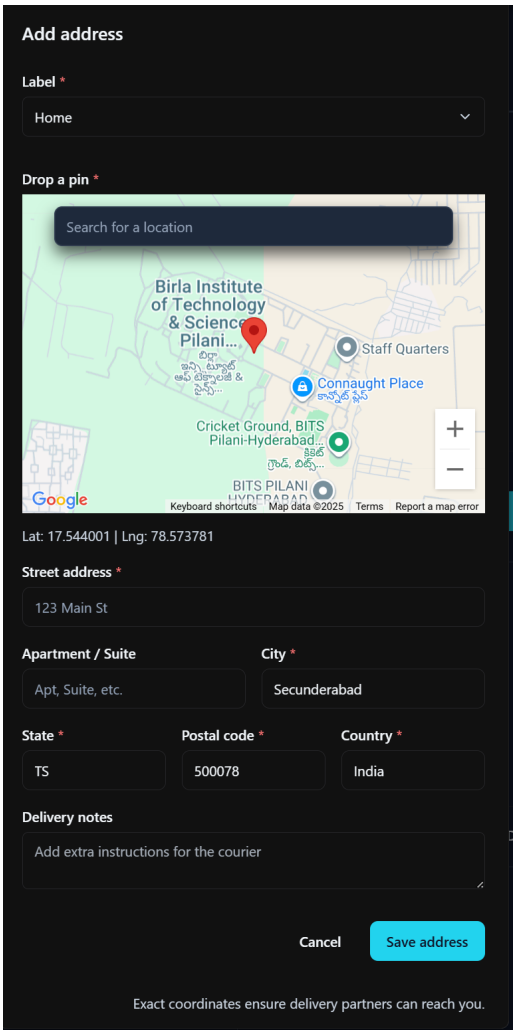
5. **Delivery**: Delivery partners can claim available orders, update statuses, and complete deliveries.

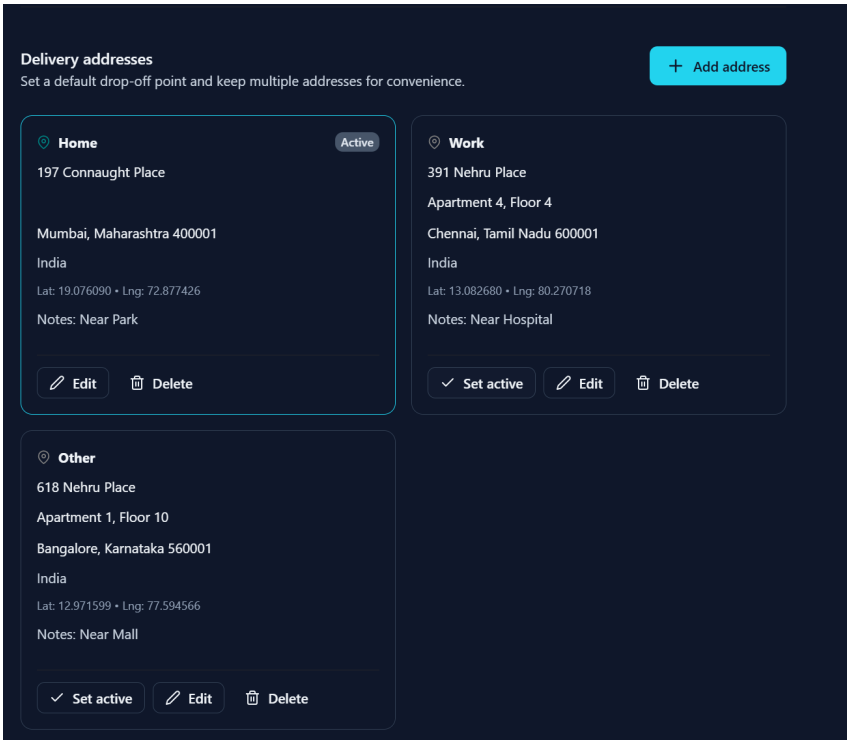
6. **Settlement**: The platform automatically calculates commissions and settles payments to sellers, ensuring a transparent financial flow.

3 Specialized Features

3.1 Google Maps Integration and Geolocation

- **How it works**: When a user searches for a product, the backend calculates the Haversine distance between the user's coordinates and every seller's shop location.
- **The Result**: Search results are sorted by proximity. A customer sees the shop 500 meters away before the one 5 kilometers away.
- **Address Management**: The system uses the Google Maps Autocomplete API to let users pinpoint their exact delivery location, storing the latitude and longitude for precise delivery routing.

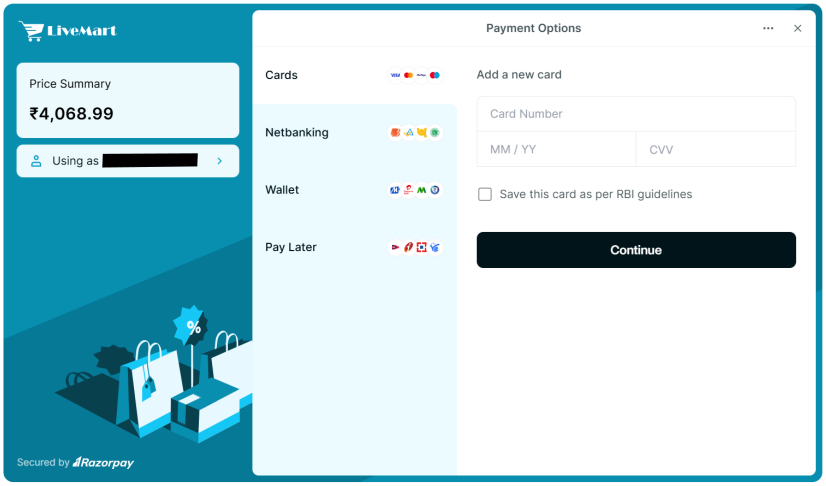


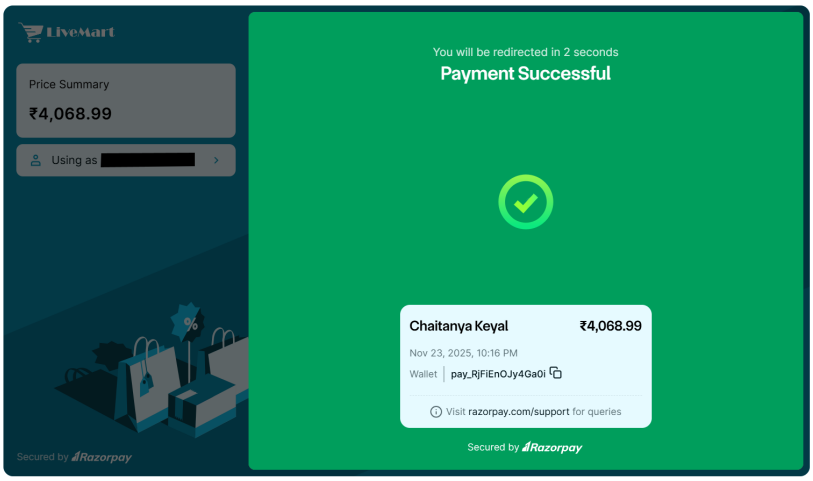


3.2 Razorpay Payment Gateway with Webhooks

The platform implements a robust payment flow that handles the unpredictability of real-world internet connections.

- **The Flow**: When a user initiates payment, the system creates a secure order on Razorpay's servers.
- **Webhooks**: The system relies on webhooks for confirmation. If a user pays but closes their browser immediately, Razorpay sends a secure, signed signal to the backend endpoint (`/payments/razorpay/webhook`).
- **Reliability**: This ensures that no successful payment goes unrecorded. The system automatically clears the cart, generates an invoice, and notifies the seller only after cryptographic verification.

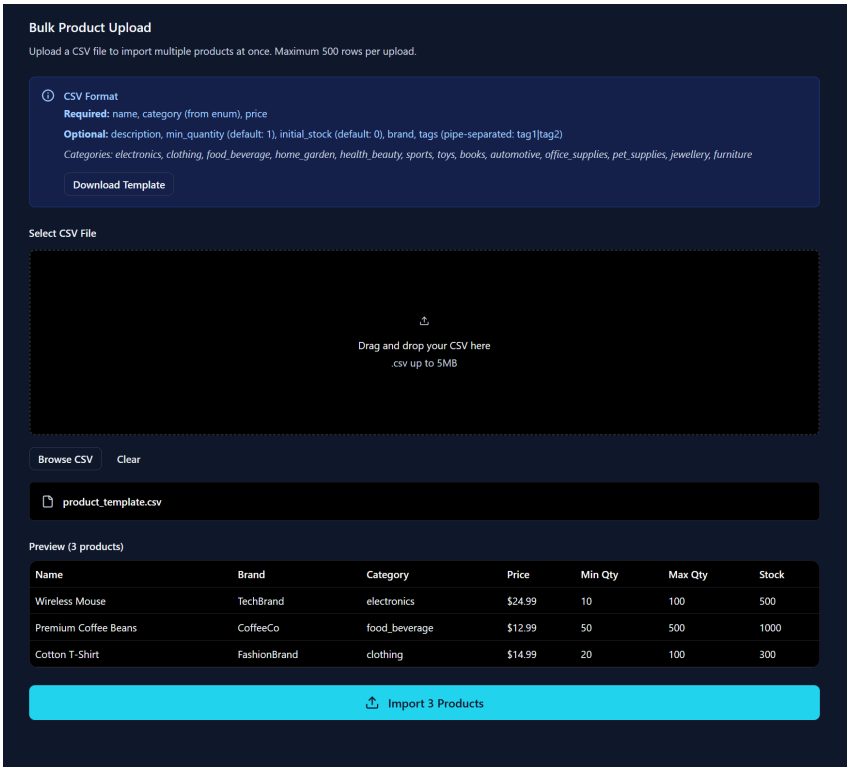




3.3 Bulk Product Upload for Wholesalers

Wholesalers manage thousands of SKUs, making individual product entry impractical.

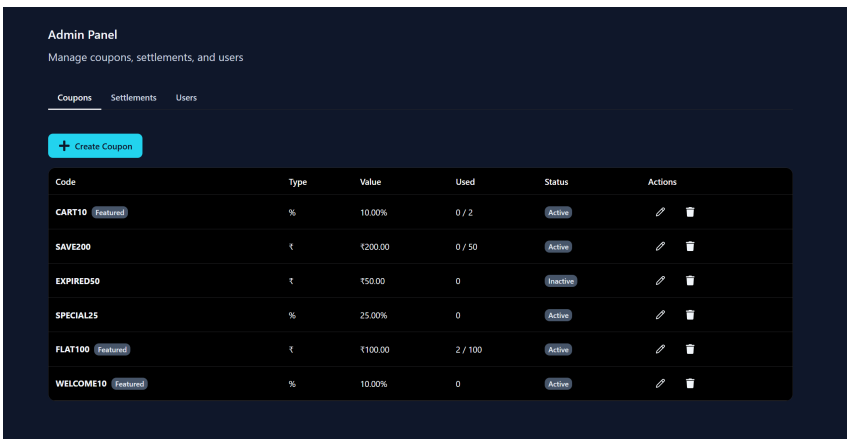
- **The Solution**: The platform provides a dedicated Bulk Import feature (`/products/bulk-import`
- **How it works**: Wholesalers can upload a structured JSON or CSV payload containing hundreds of products.
- **Validation**: The backend processes this data, validating every row for data integrity (ensuring price is greater than zero, stock is valid) before committing to the database. The system provides a detailed report of successes and failures (such as "Row 45 failed: Invalid Category").

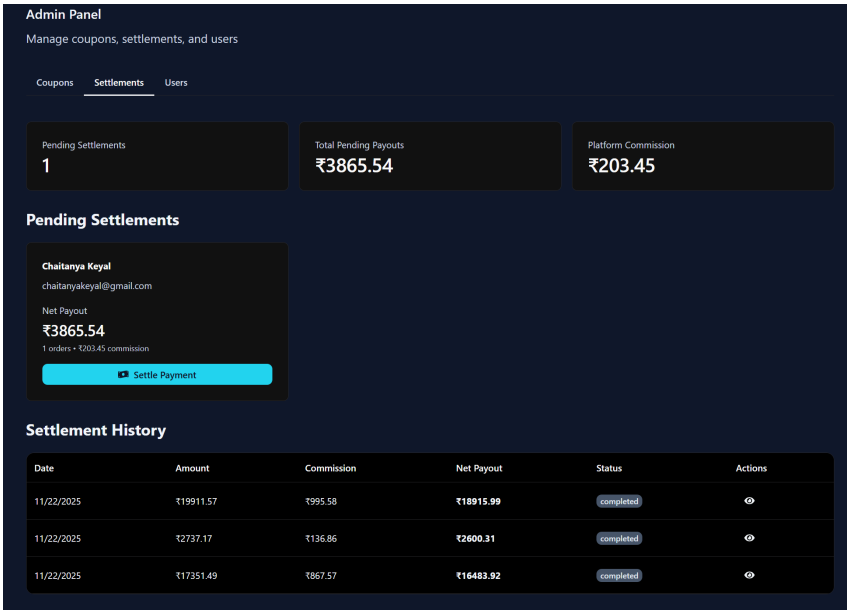


3.4 Admin Panel and Financial Settlements

The platform includes a comprehensive Admin Dashboard for system oversight and financial management.

- **Settlements**: The system tracks every order and calculates:
  - Total Amount Paid
  - Platform Commission (for example, 5%)
  - Net Payable to Seller
- **Reconciliation**: Admins can view pending settlements aggregated by seller. They can mark a payout as completed, which updates the ledger and creates a permanent financial record.

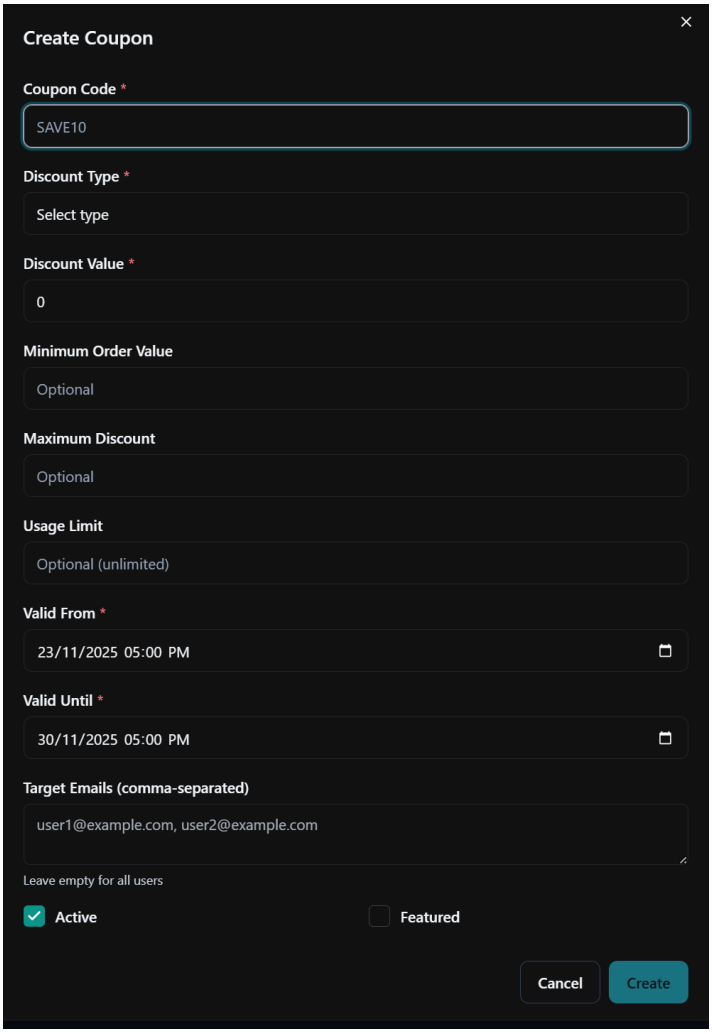


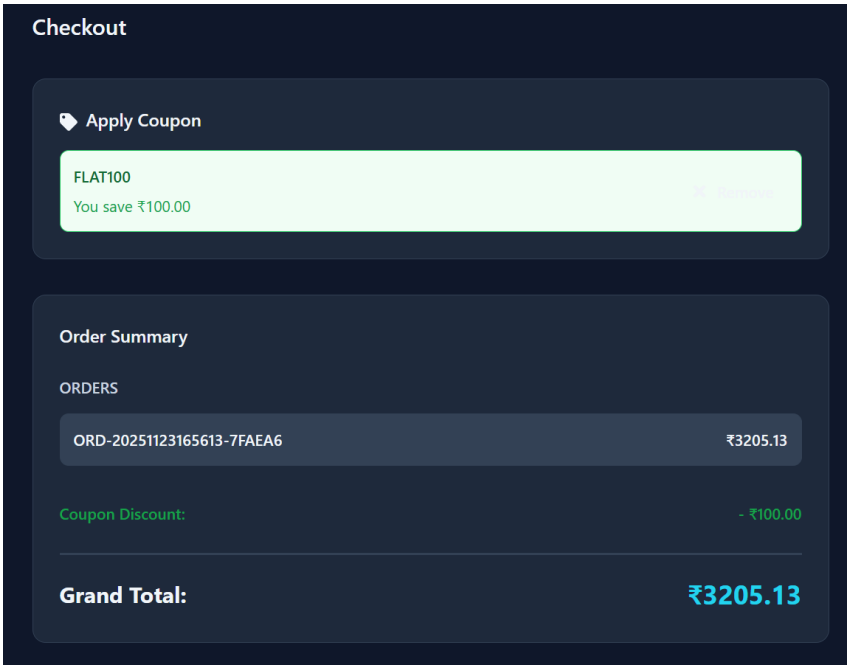


3.5 Smart Coupon System

The platform implements a comprehensive coupon system (`/coupons`) with advanced validation logic.

- **Validation Logic**: The system checks multiple constraints in real-time:
  - Is the coupon active?
  - Has the usage limit been reached?
  - Does the cart meet the minimum order value?
- **Targeting**: Coupons can be public (available to all users) or targeted (sent to specific email lists via background tasks).
- **Usage Tracking**: The system atomically increments usage counters to prevent race conditions where more users apply a limited coupon than allowed.

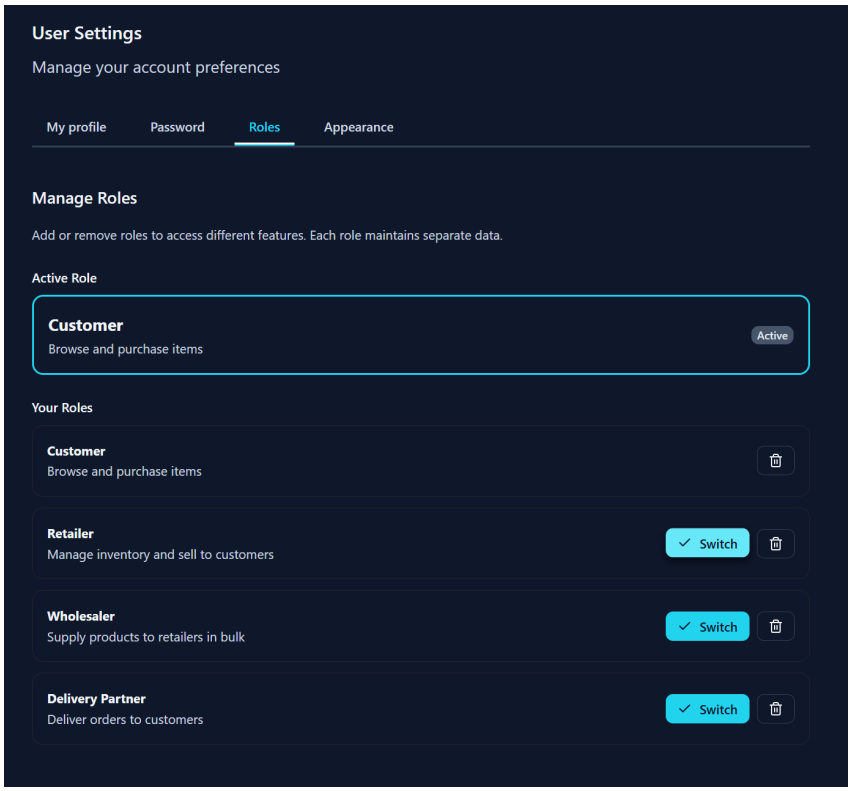

Create Coupon
Coupon Code *
SAVE10
Discount Type *
Select type
Discount Value *
0
Minimum Order Value
Optional
Maximum Discount
Optional
Usage Limit
Optional (unlimited)
Valid From *
23/11/2025 05:00 PM
Valid Until *
30/11/2025 05:00 PM
Target Emails (comma-separated)
user1@example.com, user2@example.com
Leave empty for all users
Active
Featured
Cancel
Create


Checkout
Apply Coupon
FLAT100
You save ₹100.00
Order Summary
ORDERS
ORD-20251123165613-7FAEA6
₹3205.13
Coupon Discount:
- ₹100.00
Grand Total:
₹3205.13

3.6 Dynamic Role Switching

This represents a key architectural feature of the platform.

- **The Problem**: In most applications, users are either buyers or sellers. Supporting both roles typically requires two separate accounts.
- **The Solution**: The system decouples `User` from `Role`. A user can have a list of roles (for example, `[CUSTOMER, RETAILER]`).
- **The Experience**: A toggle switch in the UI allows users to change their active context. When switching to "Retailer", the entire UI transforms: search bars become inventory tools, and cart icons become order management dashboards. The backend validates every request against the active role, ensuring strict security boundaries.



4 Technology Stack

4.1 Frontend

The frontend is built as a Single Page Application (SPA) designed for performance, scalability, and developer experience.

- **Framework**: **React 19** - Leveraging the latest features like Actions and optimistic UI updates for a responsive user experience.
- **Build Tool**: **Vite** - Chosen for its lightning-fast hot module replacement (HMR) and optimized production builds.

- **Language**: **TypeScript** - Ensures type safety across the entire application, significantly reducing runtime errors. The frontend types are auto-generated from the backend OpenAPI schema, guaranteeing 100% contract synchronization.
- **UI Library**: **Chakra UI v3** - A modular, accessible component library. The platform utilizes a custom theme system (`createSystem`) with semantic tokens for consistent design and easy dark mode implementation.
- **State Management & Data Fetching**: **TanStack Query (React Query)** - Handles server state, caching, background refetching, and optimistic updates, eliminating the need for complex global state management for API data.
- **Routing**: **TanStack Router** - A type-safe router that manages URL state, search parameters, and nested layouts (such as `_layout/admin`, `_layout/seller`). It ensures that broken links are caught at compile time.
- **Maps Integration**: **Google Maps API** (@react-google-maps/api) - Powers the location-based features, allowing users to pinpoint addresses and find nearby sellers.
- **Linting & Formatting**: **Biome** - A fast, all-in-one toolchain for linting and formatting, ensuring code consistency.

4.2 Backend

The backend is a high-performance, asynchronous REST API designed for reliability and ease of maintenance.

- **Framework**: **FastAPI** - Selected for its speed (Starlette), ease of use, and automatic OpenAPI documentation generation. It supports asynchronous request handling, making it ideal for I/O-bound operations like database queries and external API calls.
- **Database ORM**: **SQLModel** - A library that combines SQLAlchemy and Pydantic. It allows defining models that serve as both database tables and data validation schemas, reducing code duplication.
- **Database**: **PostgreSQL 16+** - A robust, open-source relational database used for its reliability, complex query support, and JSON capabilities (used for storing snapshots of addresses and order history).
- **Authentication**: **OAuth2 with Password Flow (JWT)** - Secure stateless authentication. Access tokens are short-lived (8 days), and the system supports **Google OAuth** for frictionless onboarding.
- **Security**: **Passlib (bcrypt)** for password hashing. Role-based access control (RBAC) is enforced via dependency injection (`require_role`).
- **Migrations**: **Alembic** - Handles database schema changes, ensuring smooth transitions between versions without data loss.
- **Email**: **MJML** - Responsive email templates are designed in MJML and compiled to HTML. **SMTP** is used for reliable delivery of OTPs, order confirmations, and notifications.

- **Payments**: **Razorpay** - Integrated for secure payment processing, supporting webhooks for real-time payment status updates.
- **Linting & Formatting**: **Ruff** - An extremely fast Python linter and formatter.

4.3 Other Libraries and Tools

- **Containerization**: **Docker & Docker Compose** - The entire stack (Frontend, Backend, DB, Traefik, Adminer) is containerized, ensuring consistency across development, staging, and production environments.
- **Reverse Proxy**: **Traefik** - Automatically discovers Docker services and routes traffic. It handles SSL termination (Let's Encrypt) and load balancing.
- **Database Management**: **Adminer** - A lightweight web interface for managing the PostgreSQL database directly.
- **Version Control**: **Git** - Source code management.

4.4 DevTools, CI/CD, and Deployment

- **CI/CD**: **GitHub Actions** - Automated workflows for:
  - **Linting**: Checks code quality on every push.
  - **Staging Deployment**: Deploys to the staging environment on merges to the main branch.
  - **Production Deployment**: Deploys to production on release tags.
- **Pre-commit Hooks**: Automatically runs Ruff and Biome before commits to prevent bad code from entering the repository.

5 Features Implemented

5.1 Registration and Sign-Up

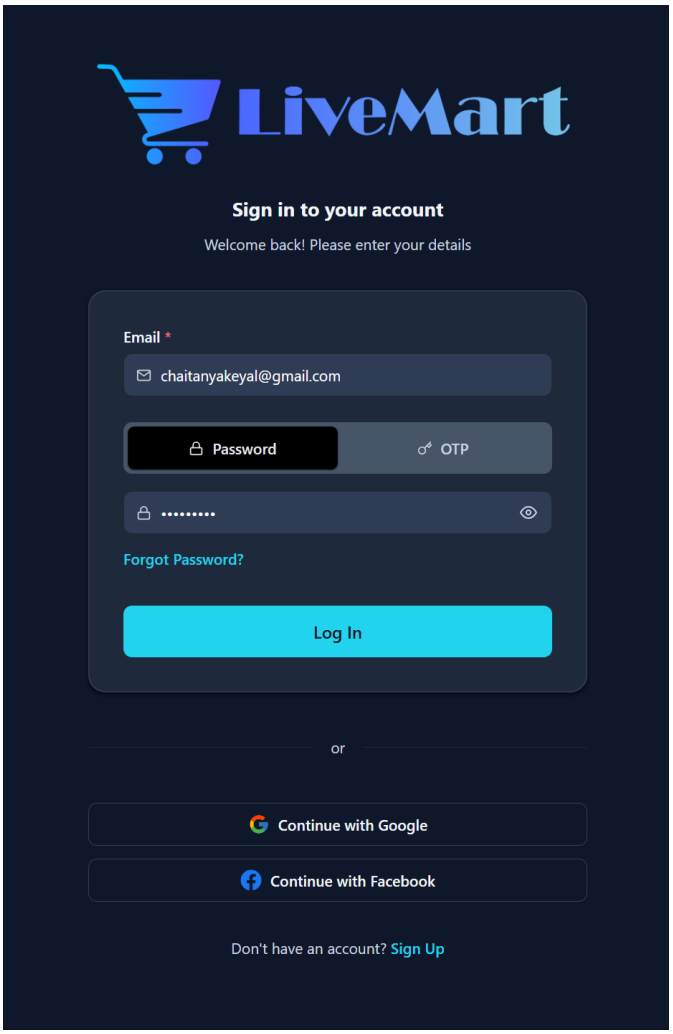
This module handles the onboarding and identity management of users.

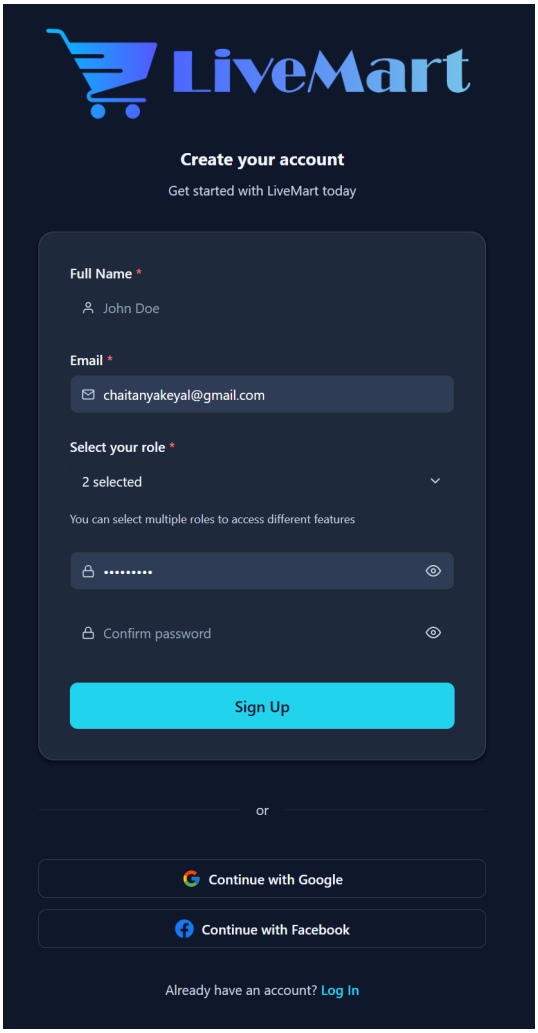
- **Multi-role Registration**: Users select their primary role (Customer, Retailer, Wholesaler) during sign-up. The system allows users to add additional roles later.
- **Secure Authentication**:
  - **Flow**: User enters credentials, backend validates and issues JWT, frontend stores token.
  - **Security**: Passwords are never stored in plain text but are hashed using bcrypt.
- **Email Verification**:
  - **Flow**: After sign-up, the system generates a 6-digit OTP, sends it via email, user enters the OTP, and the account is verified.

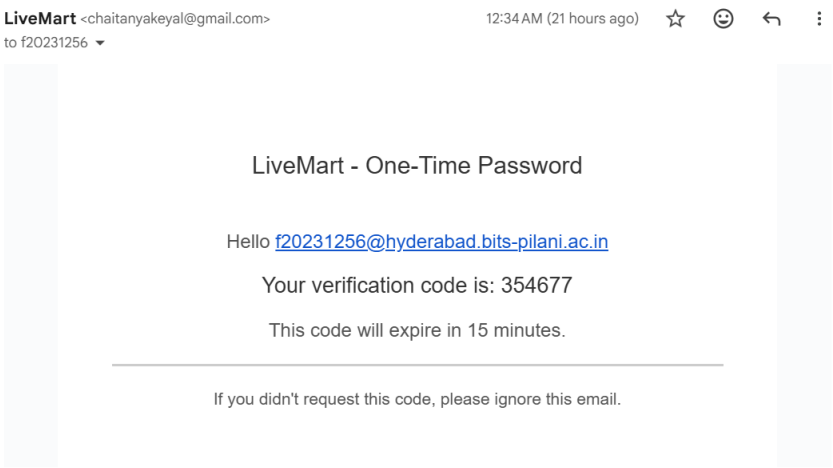
- **Purpose**: Prevents spam accounts and ensures communication channels are valid.

- **Social Login**:
  - **Google OAuth**: Users can sign up or log in with one click. The system automatically retrieves their email and name, creating a seamless entry point.

- **Role Management**:
  - **Dynamic Switching**: Users can switch their active role in the dashboard. For example, a user can switch from Customer mode to Retailer mode to manage their shop without logging out.







5.2 User Dashboards

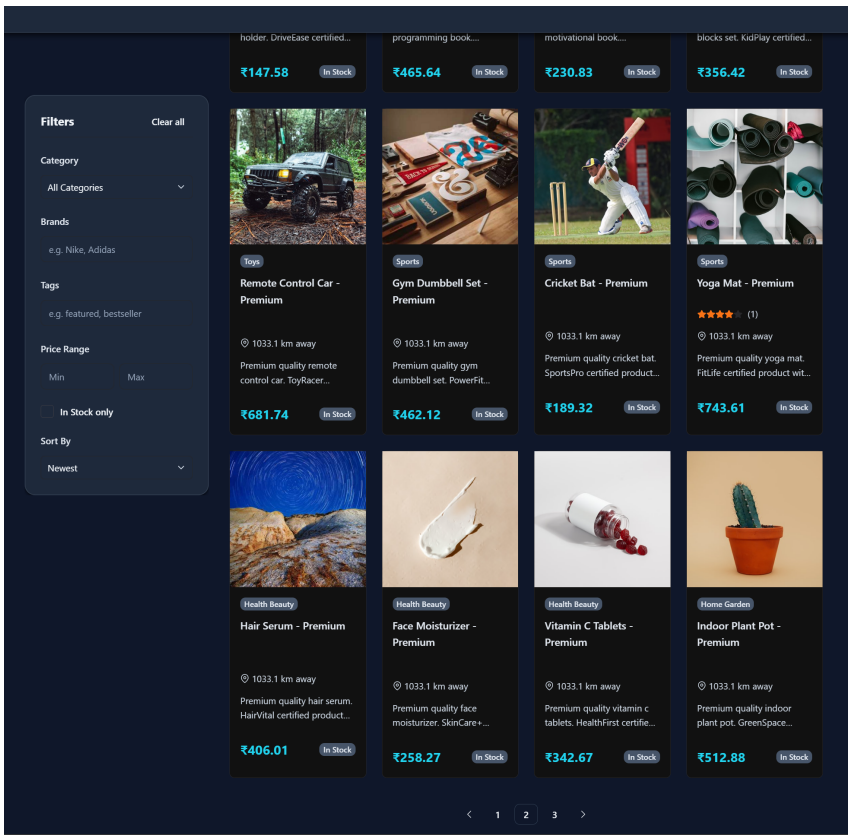
Tailored interfaces for each user type ensure relevant information is always at hand.

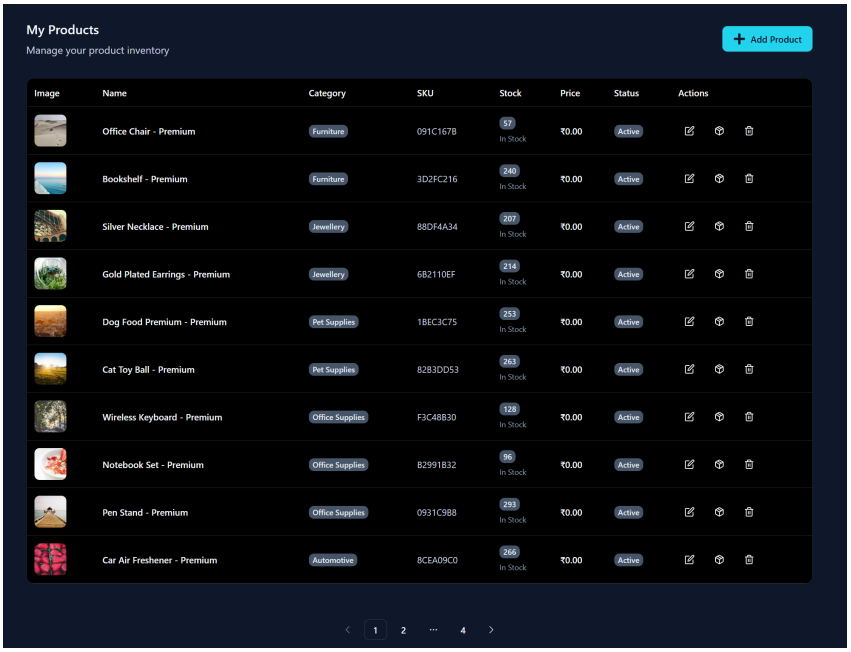
- **Customer Dashboard**:

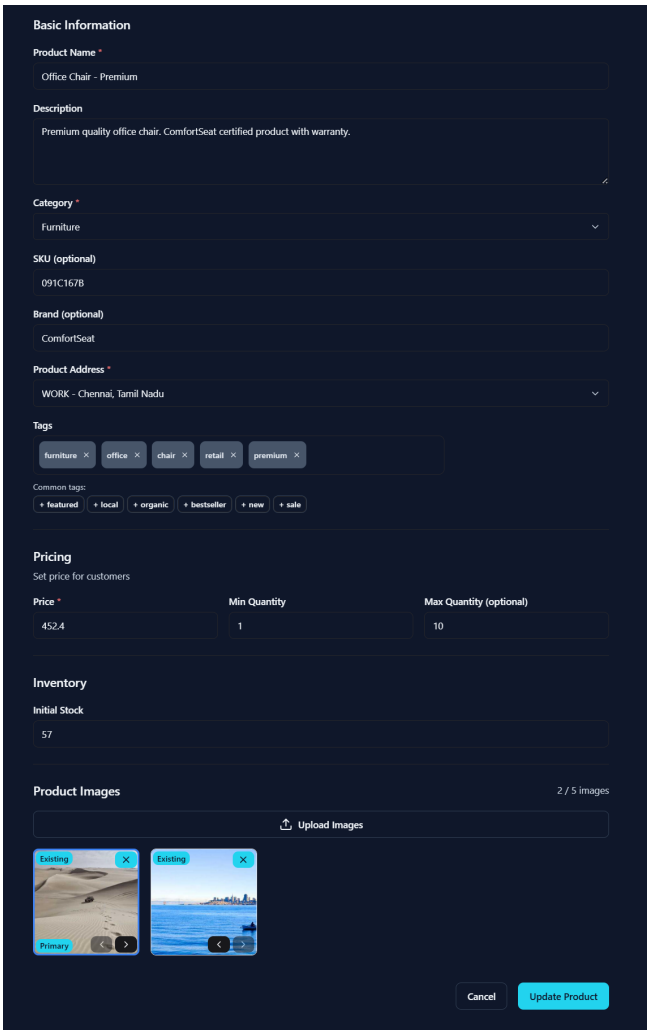
  - **Home Feed**: Personalized product recommendations and nearby shop listings.
  - **Order History**: List of past orders with status tracking.

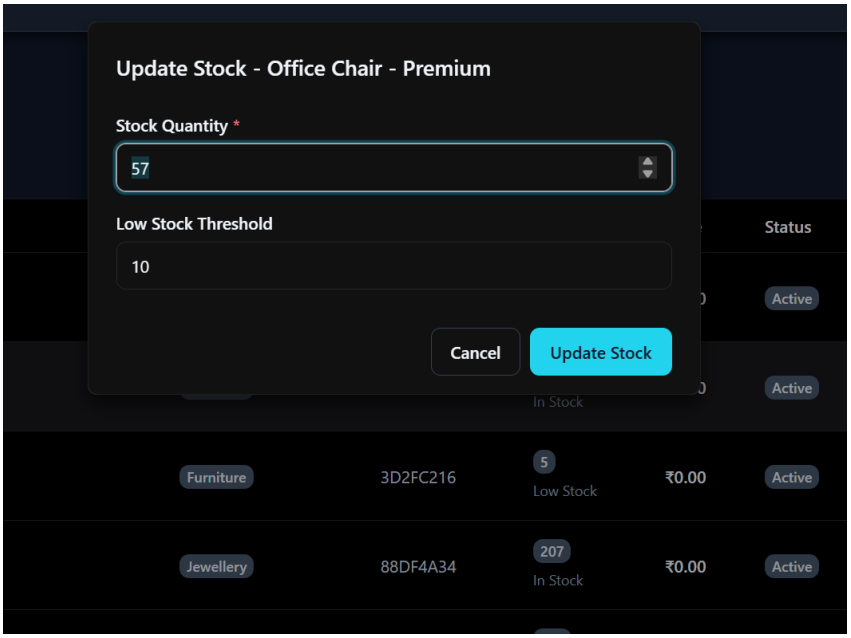
- **Retailer/Wholesaler Dashboard**:
  - **Inventory Management**: CRUD operations for products. Sellers can set stock levels, low-stock alerts, and pricing.
  - **Sales Overview**: Charts and metrics showing daily and weekly sales performance.
  - **Order Management**: Interface to view incoming orders, update status (for example, "Ready to Ship"), and assign delivery partners.

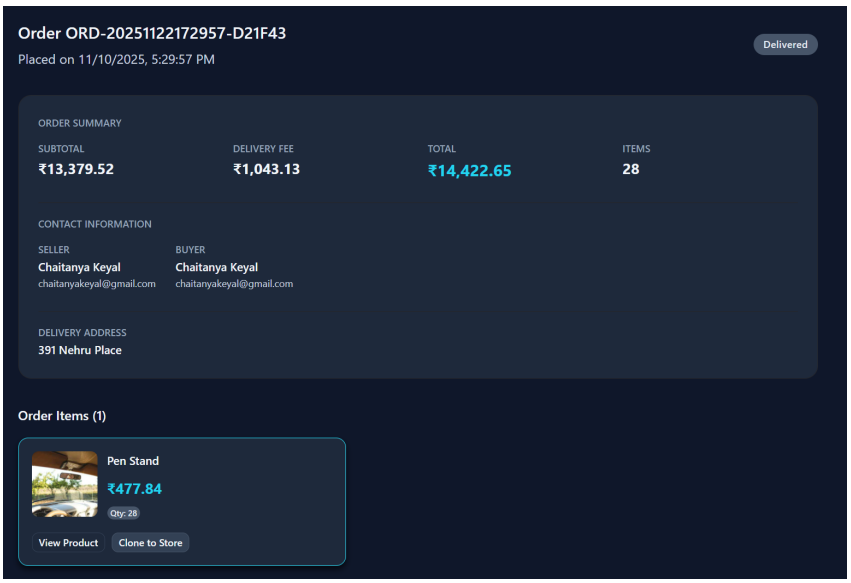
- **Product Management**:
  - **Rich Details**: Support for multiple images, detailed descriptions, and categorization.
  - **Pricing Tiers**: Sellers can define different prices for different buyer types (for example, a Wholesaler sets a lower price for Retailers buying in bulk).




My Products
Manage your product inventory
Add Product
Image
Name
Category
SKU
Stock
Price
Status
Actions
Office Chair - Premium
Furniture
091C167B
57
In Stock
₹0.00
Active
Bookshelf - Premium
Furniture
3D2FC216
240
In Stock
₹0.00
Active
Silver Necklace - Premium
Jewellery
88DF4A34
207
In Stock
₹0.00
Active
Gold Plated Earrings - Premium
Jewellery
6B2110EF
214
In Stock
₹0.00
Active
Dog Food Premium - Premium
Pet Supplies
1BEC3C75
253
In Stock
₹0.00
Active
Cat Toy Ball - Premium
Pet Supplies
82B3DD53
263
In Stock
₹0.00
Active
Wireless Keyboard - Premium
Office Supplies
F3C48B30
128
In Stock
₹0.00
Active
Notebook Set - Premium
Office Supplies
B2991B32
96
In Stock
₹0.00
Active
Pen Stand - Premium
Office Supplies
0931C9B8
293
In Stock
₹0.00
Active
Car Air Freshener - Premium
Automotive
8CEA09C0
266
In Stock
₹0.00
Active
1
2
4


Basic Information
Product Name *
Office Chair - Premium
Description
Premium quality office chair. ComfortSeat certified product with warranty.
Category *
Furniture
SKU (optional)
091C167B
Brand (optional)
ComfortSeat
Product Address *
WORK - Chennai, Tamil Nadu
Tags
furniture
office
chair
retail
premium
Common tags:
+ featured
+ local
+ organic
+ bestseller
+ new
+ sale
Pricing
Set price for customers
Price *
452.4
Min Quantity
1
Max Quantity (optional)
10
Inventory
Initial Stock
57
Product Images
2 / 5 images
Upload Images
Existing
Primary
Existing
Cancel
Update Product



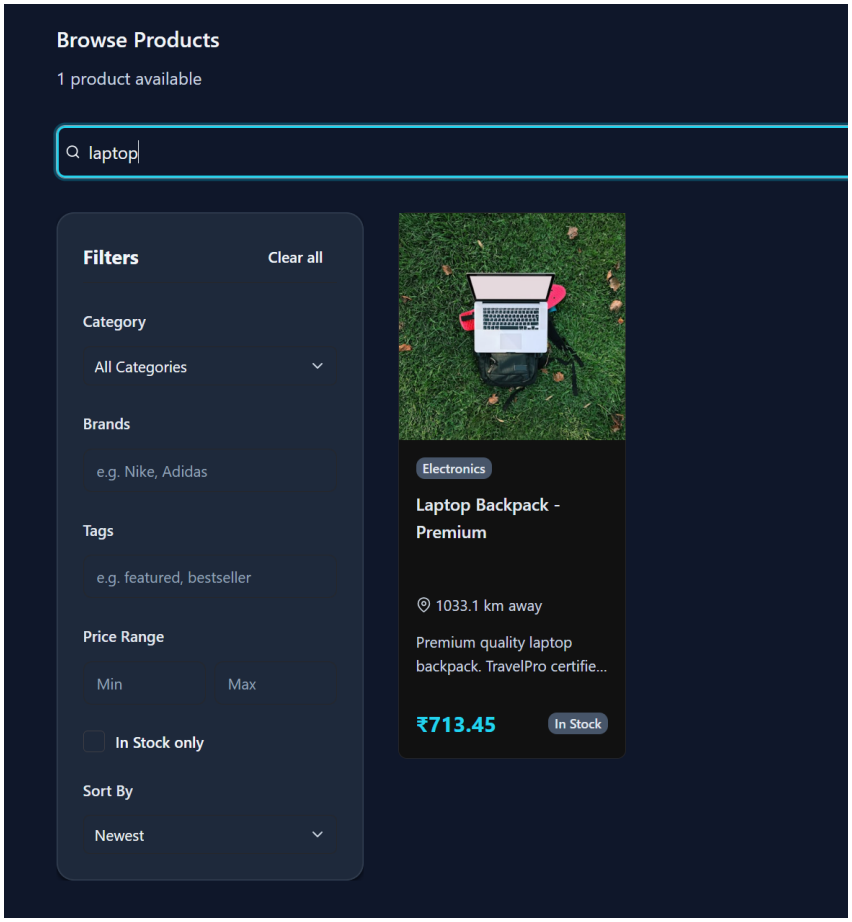


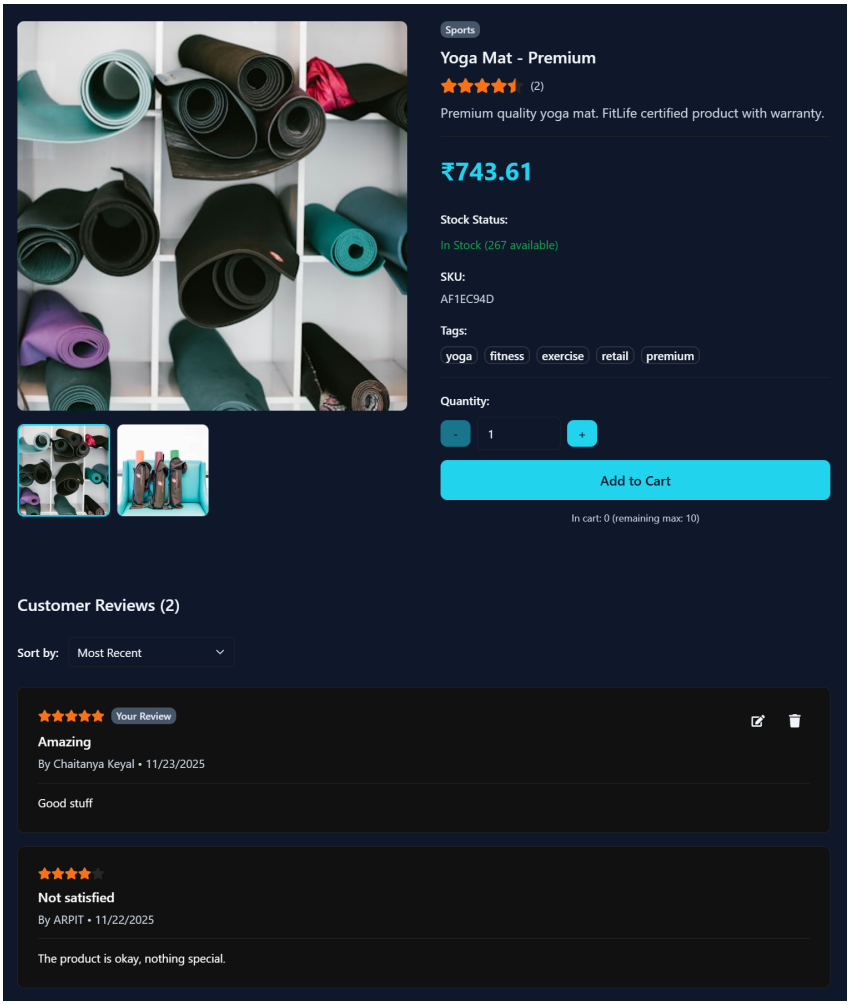
5.3 Search & Navigation

Helping users find what they need quickly and efficiently.

- **Smart Filtering**:
  - **Facets**: Filter by Category, Price Range, Brand, and Availability.
  - **Logic**: Filters are applied server-side for performance.
- **Location-Based Services**:
  - **Geospatial Search**: The platform provides location-based shop discovery. The backend uses the Haversine formula to sort sellers by distance.
- **Product Search**:

– **Full-Text Search**: Users can search for product names, descriptions, or SKUs.



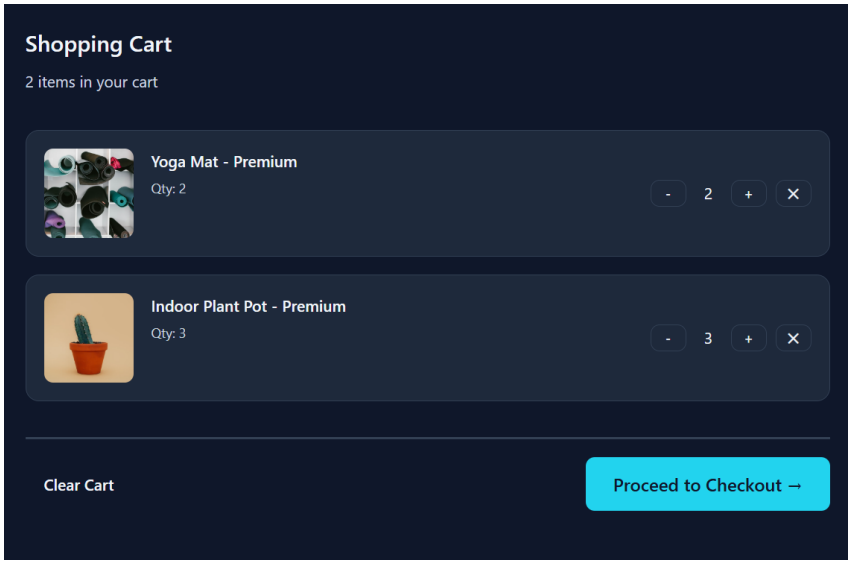


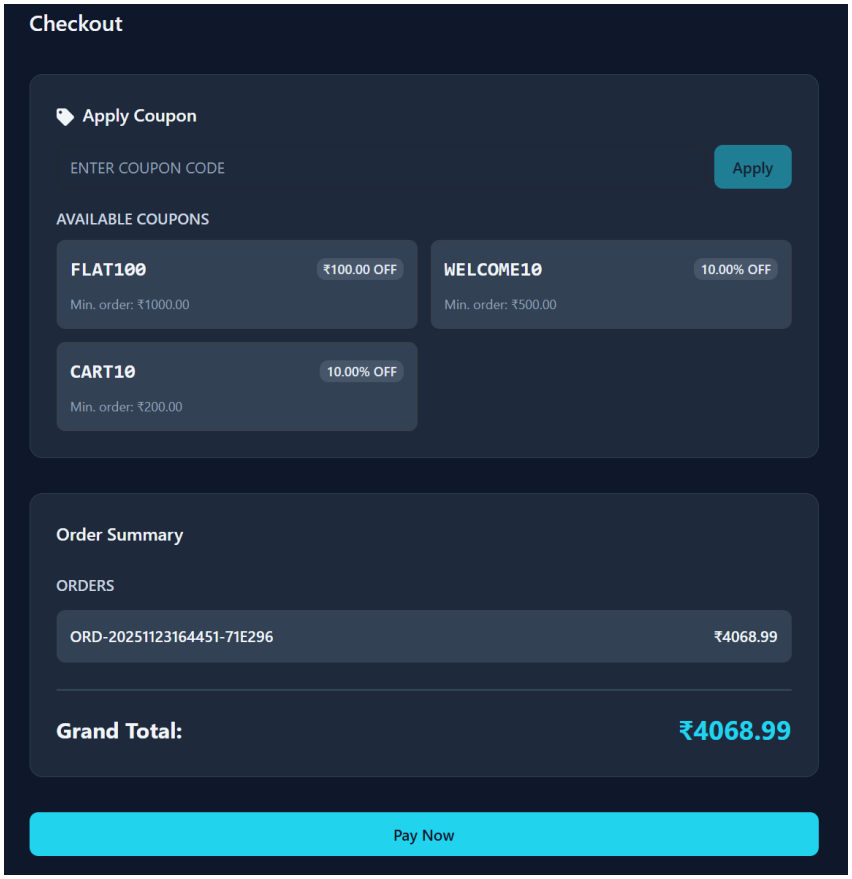
5.4 Order & Payment Management

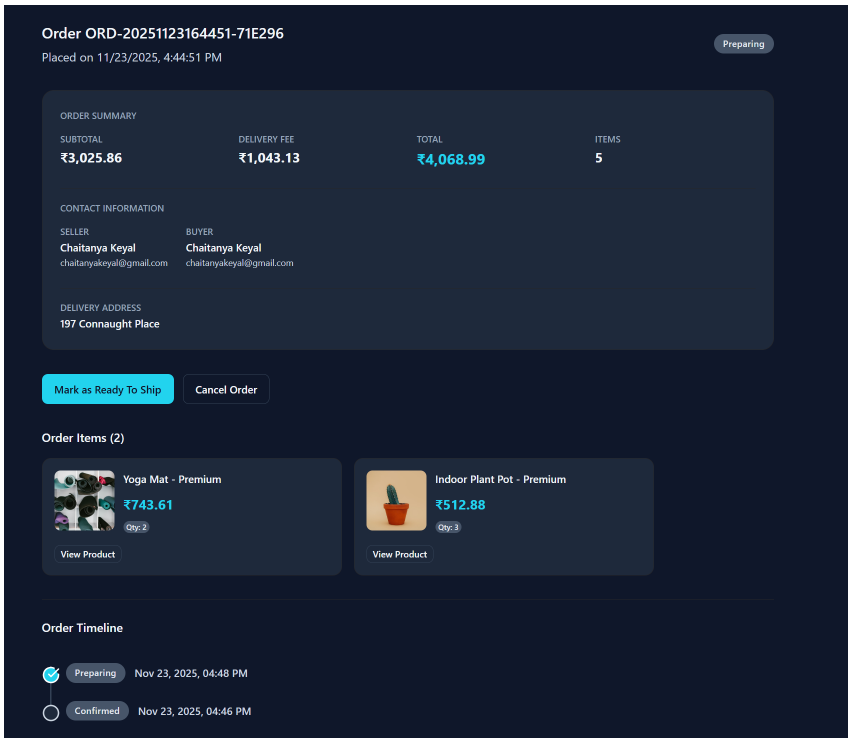
The core commerce engine of the platform.

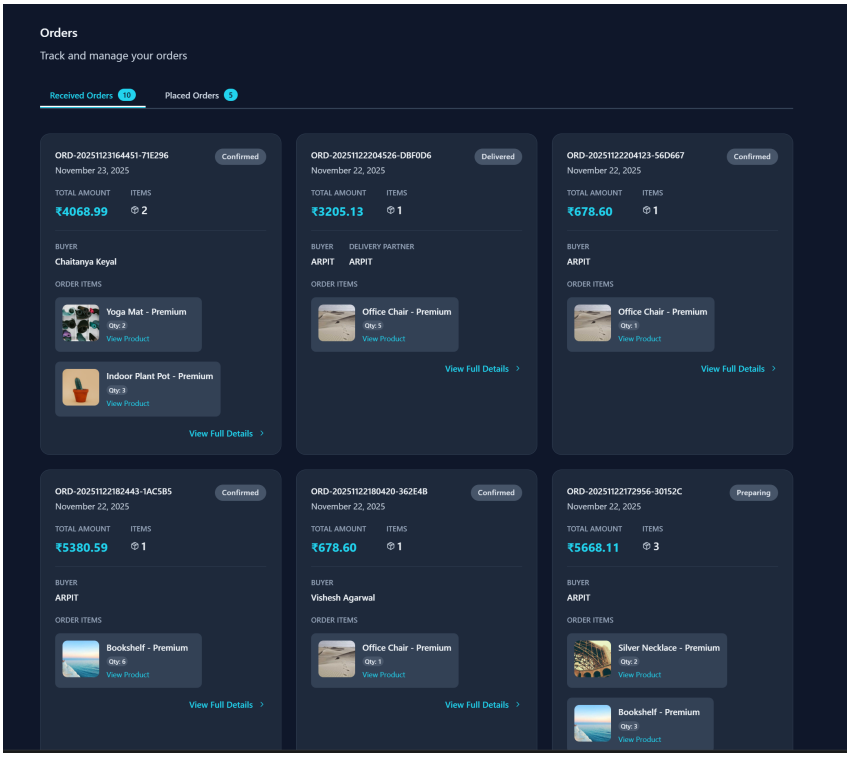
- **Cart System**:
  - **Persistence**: Cart items are stored in the database, so they persist across devices.
  - **Validation**: Real-time stock checks prevent adding out-of-stock items.
- **Checkout Process**:
  - **Address Selection**: Users choose from saved addresses or add a new one using Google Maps autocomplete.
  - **Coupon Application**: Users can apply discount codes. The system validates validity, usage limits, and minimum order values.
- **Payment Gateway**:
  - **Razorpay Integration**: Seamless flow for credit cards, UPI, and net banking.
  - **Webhooks**: The backend listens for Razorpay webhooks to automatically mark orders as "Paid" even if the user closes the browser.

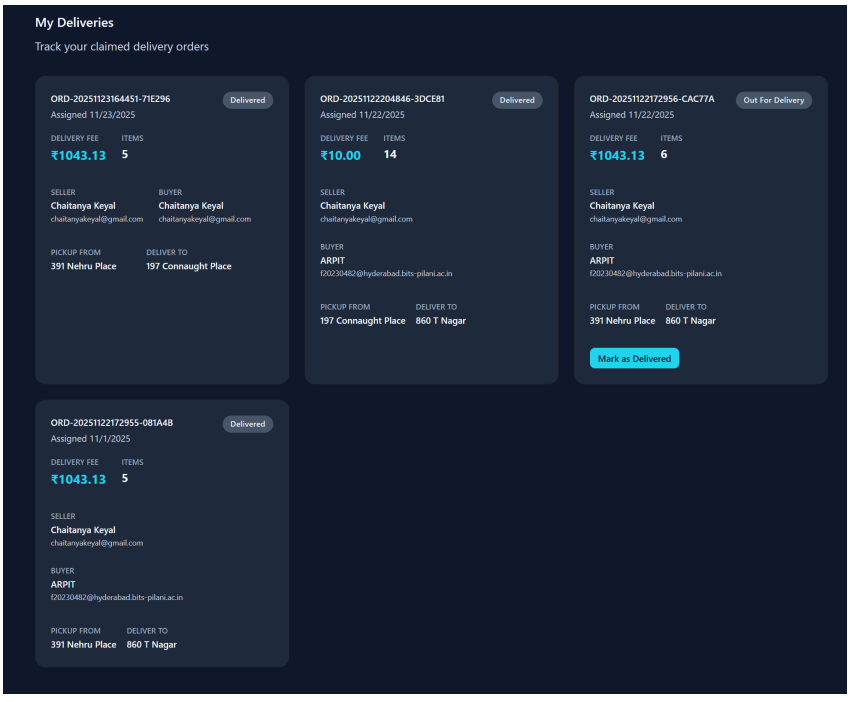
- **Order Tracking**:
  - **Granular Statuses**: Orders progress through multiple states: Pending, Confirmed, Preparing, Ready to Ship, Out for Delivery, and Delivered.
  - **History**: A timeline view shows when each status change occurred.
- **Settlements**:
  - **Automated Logic**: The system calculates the platform commission and the net amount due to the seller.
  - **Transparency**: Sellers can view a report of all settlements and their status (Pending or Completed).

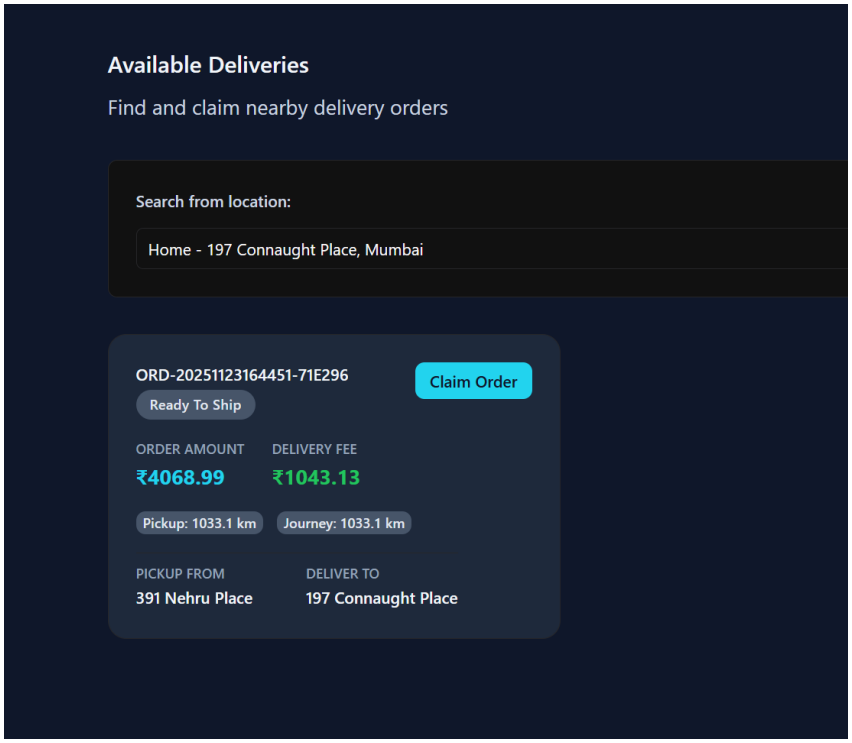



Checkout
Apply Coupon
ENTER COUPON CODE
Apply
AVAILABLE COUPONS
FLAT100
₹100.00 OFF
Min. order: ₹1000.00
WELCOME10
10.00% OFF
Min. order: ₹500.00
CART10
10.00% OFF
Min. order: ₹200.00
Order Summary
ORDERS
ORD-20251123164451-71E296
₹4068.99
Grand Total:
₹4068.99
Pay Now


Order ORD-20251123164451-71E296
Placed on 11/23/2025, 4:44:51 PM
Preparing
ORDER SUMMARY
SUBTOTAL
₹3,025.86
DELIVERY FEE
₹1,043.13
TOTAL
₹4,068.99
ITEMS
5
CONTACT INFORMATION
SELLER
Chaitanya Keyal
chaitanyakeyal@gmail.com
BUYER
Chaitanya Keyal
chaitanyakeyal@gmail.com
DELIVERY ADDRESS
197 Connaught Place
Mark as Ready To Ship
Cancel Order
Order Items (2)
Yoga Mat - Premium
₹743.61
Qty: 2
View Product
Indoor Plant Pot - Premium
₹512.88
Qty: 3
View Product
Order Timeline
Preparing
Nov 23, 2025, 04:48 PM
Confirmed
Nov 23, 2025, 04:46 PM


Orders
Track and manage your orders
Received Orders 10
Placed Orders 5
ORD-20251123164451-71E296
Confirmed
November 23, 2025
TOTAL AMOUNT
₹4068.99
ITEMS
2
BUYER
Chaitanya Keyal
ORDER ITEMS
Yoga Mat - Premium
Qty: 2
View Product
Indoor Plant Pot - Premium
Qty: 3
View Product
View Full Details
ORD-20251122204526-DBF0D6
Delivered
November 22, 2025
TOTAL AMOUNT
₹3205.13
ITEMS
1
BUYER
ARPIT
DELIVERY PARTNER
ARPIT
ORDER ITEMS
Office Chair - Premium
Qty: 5
View Product
View Full Details
ORD-20251122204123-56D667
Confirmed
November 22, 2025
TOTAL AMOUNT
₹678.60
ITEMS
1
BUYER
ARPIT
ORDER ITEMS
Office Chair - Premium
Qty: 1
View Product
View Full Details
ORD-20251122182443-1AC5B5
Confirmed
November 22, 2025
TOTAL AMOUNT
₹5380.59
ITEMS
1
BUYER
ARPIT
ORDER ITEMS
Bookshelf - Premium
Qty: 6
View Product
View Full Details
ORD-20251122180420-362E4B
Confirmed
November 22, 2025
TOTAL AMOUNT
₹678.60
ITEMS
1
BUYER
Vishesh Agarwal
ORDER ITEMS
Office Chair - Premium
Qty: 1
View Product
View Full Details
ORD-20251122172956-30152C
Preparing
November 22, 2025
TOTAL AMOUNT
₹5668.11
ITEMS
3
BUYER
ARPIT
ORDER ITEMS
Silver Necklace - Premium
Qty: 2
View Product
Bookshelf - Premium
Qty: 3
View Product

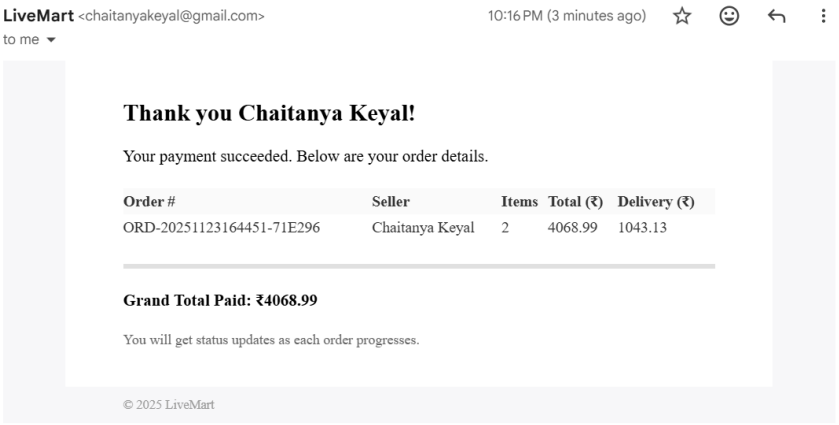

My Deliveries
Track your claimed delivery orders
ORD-20251123164451-71E296
Delivered
Assigned 11/23/2025
DELIVERY FEE
₹1043.13
ITEMS
5
SELLER
Chaitanya Keyal
chaitanyakeyal@gmail.com
BUYER
Chaitanya Keyal
chaitanyakeyal@gmail.com
PICKUP FROM
391 Nehru Place
DELIVER TO
197 Connaught Place
ORD-20251122204846-3DCE81
Delivered
Assigned 11/22/2025
DELIVERY FEE
₹10.00
ITEMS
14
SELLER
Chaitanya Keyal
chaitanyakeyal@gmail.com
BUYER
ARPIT
f20230482@hyderabad.bits-pilani.ac.in
PICKUP FROM
197 Connaught Place
DELIVER TO
860 T Nagar
ORD-20251122172956-CAC77A
Out For Delivery
Assigned 11/22/2025
DELIVERY FEE
₹1043.13
ITEMS
6
SELLER
Chaitanya Keyal
chaitanyakeyal@gmail.com
BUYER
ARPIT
f20230482@hyderabad.bits-pilani.ac.in
PICKUP FROM
391 Nehru Place
DELIVER TO
860 T Nagar
Mark as Delivered
ORD-20251122172955-081A4B
Delivered
Assigned 11/1/2025
DELIVERY FEE
₹1043.13
ITEMS
5
SELLER
Chaitanya Keyal
chaitanyakeyal@gmail.com
BUYER
ARPIT
f20230482@hyderabad.bits-pilani.ac.in
PICKUP FROM
391 Nehru Place
DELIVER TO
860 T Nagar



5.5 Feedback and Dashboard Updates

Ensuring quality and keeping users informed.

- **Reviews & Ratings**:
  - **Verified Purchase Only**: Only users who have bought a product can review it, ensuring authenticity.
  - **Aggregates**: Product pages show average ratings and review counts.
- **Notifications**:
  - **Email Triggers**: System sends emails for: OTP, Order Confirmation, Status Updates, Coupons.
- **Admin Panel**:
  - **Superuser Access**: A special role for platform administrators.
  - **Capabilities**: Manage global settings, view all users, manage coupons, and oversee platform-wide settlements.



6 Key Platform Highlights

- **Dynamic Role Switching Architecture**: Unlike traditional platforms where a user is either a buyer or a seller, LiveMart treats roles as attributes. A single `User` entity can possess multiple `UserRole` links. The `active_role` field on the user model determines the current context, allowing for a fluid user experience where a Retailer can buy stock (as a customer of a Wholesaler) and sell goods (to end Customers) using the same account.

- **Tiered Pricing Model**: The `ProductPricing` model allows a single product to have multiple price points based on the `BuyerType`. This enables complex B2B and B2C scenarios within a single catalog.
- **Automated Settlements System**: The platform includes a dedicated `PaymentSettlement` module. It aggregates orders, calculates the platform commission, and generates payout records for sellers. This automates the financial reconciliation process, which is often a manual pain point.

7 Unique Selling Points (USPs)

1. **Unified Supply Chain Ecosystem**: LiveMart is not just a B2C store or a B2B marketplace; it is both. It connects the entire chain (Wholesaler to Retailer to Customer) in one application, reducing friction and intermediaries.
2. **Hyper-Local Focus**: The architecture is designed to prioritize proximity. Search results and shop listings default to the user's location, supporting local commerce initiatives and reducing delivery times.
3. **Transparent Operations**: Real-time visibility into stock levels and order statuses builds trust. Retailers can track when their wholesale stock will arrive, and Customers can track their delivery status.
4. **Developer-Friendly Architecture**: The project uses a schema-first approach. The backend OpenAPI schema drives the frontend TypeScript client generation. This means if a backend developer changes an API response, the frontend build will fail immediately, preventing subtle runtime bugs.

8 System Architecture

The system follows a modern, containerized microservices-ready architecture. While currently deployed as a modular monolith for simplicity, the clear separation of concerns allows for easy splitting into microservices in the future.

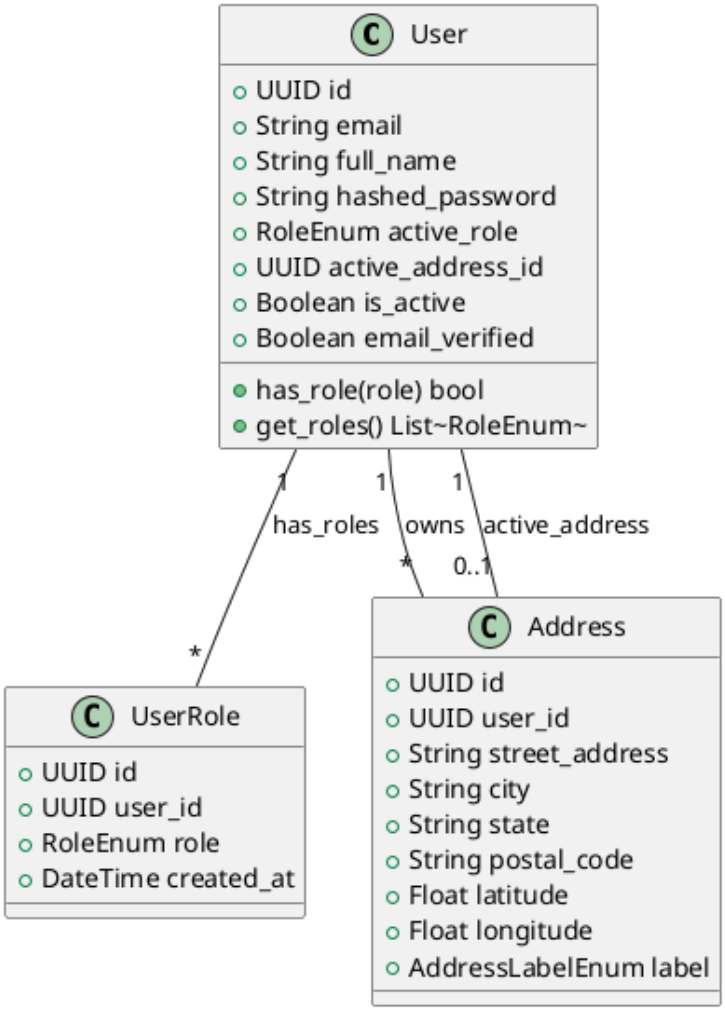
8.1 Directory Structure and Organization

- `backend/app/models`: Contains SQLModel definitions. These serve as both database tables and Pydantic schemas.
- `backend/app/api`: Holds the route handlers, organized by resource (users, products, orders).
- `backend/app/crud`: Encapsulates all database logic. The API layer does not access the database directly but calls CRUD functions.
- `frontend/src/routes`: Follows the file-based routing convention of TanStack Router.
- `frontend/src/client`: Contains the auto-generated API client SDK.

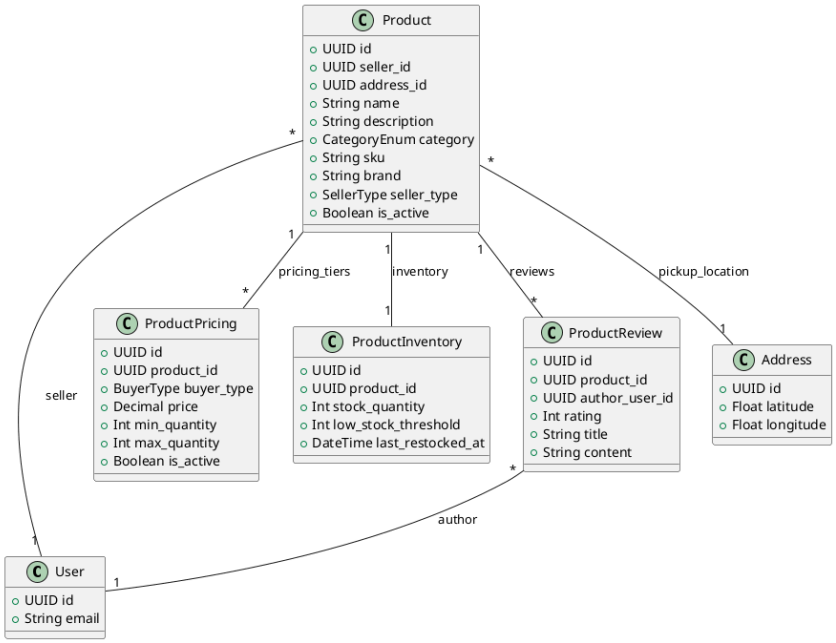
8.2 Database Schema Diagrams

The following diagrams show the core domain models and their relationships, split by functional area for clarity.

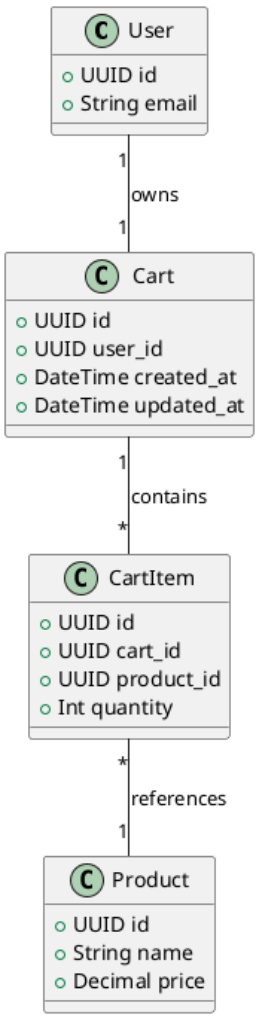
8.2.1 User and Authentication Domain



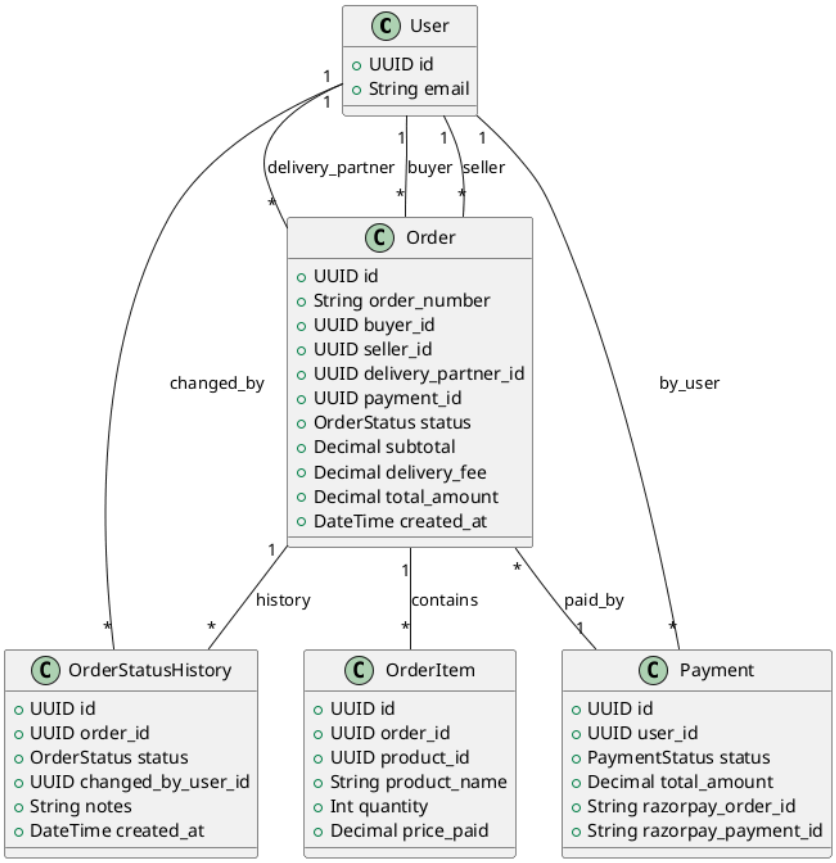
8.2.2 Product Catalog Domain



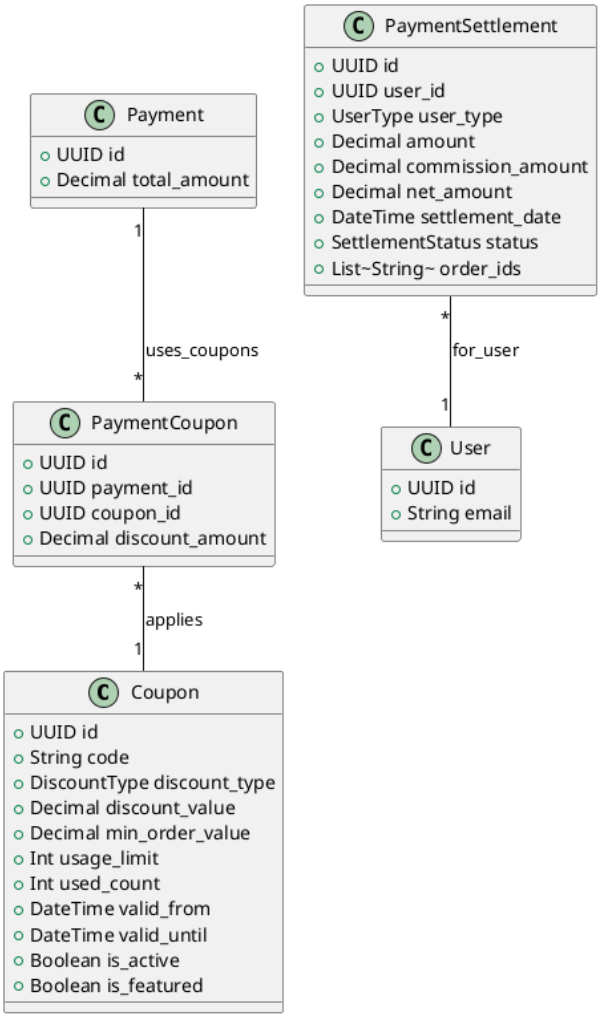
8.2.3 Shopping Cart Domain



8.2.4 Order and Payment Domain



8.2.5 Coupon and Settlement Domain



9 Database Overview

The database is normalized to 3NF to ensure data integrity.

- `user`: The central identity table. Stores credentials and profile info.
- `user_role`: A many-to-many link table between users and roles.
- `address`: Stores geocoded locations. Linked to users. Used for both shipping addresses and shop locations.
- `product`: The main catalog table. Contains static data (name, description).
- `product_pricing`: One-to-Many with Product. Allows different prices for different buyer types.
- `product_inventory`: One-to-One with Product. Separates volatile stock data from static catalog data to reduce locking contention.
- `cart` **and** `cart_item`: Transient data for shopping sessions.

- `order`: The central transaction record. Contains snapshots of addresses to preserve history even if the user changes their address later.
- `order_item`: Details of what was bought. Stores a snapshot of the price at the time of purchase.
- `payment`: Tracks Razorpay transaction details and statuses.
- `payment_settlement`: Records the financial payouts to sellers.

10 Deployment and Developer Guide

10.1 Prerequisites

- **Docker Engine and Docker Compose** (v2.0+)
- **Node.js 20+** (Only for local frontend development outside Docker)
- **Python 3.12+** (Only for local backend development outside Docker)

10.2 Quick Start with Docker (Recommended)

This is the easiest way to get the full stack running.

1. **Clone the repository**:

```
git clone https://github.com/Chaitanya-Keyal/LiveMart.git
cd LiveMart
```

2. **Configure Environment**:
   - Copy `.env.example` to `.env`.
   - Generate a secure secret key: `openssl rand -hex 32`.
   - Set `SECRET_KEY`, `POSTGRES_PASSWORD`, `FIRST_SUPERUSER`, and `FIRST_SUPERUSER_PASSWORD`.
   - (Optional) Add Google Maps API Key and Razorpay Keys for full functionality.
3. **Run the Stack**:

```
docker compose watch
```

   - The `watch` command enables hot-reloading for both frontend and backend containers.
4. **Access the App**:
   - **Frontend**: http://localhost:5173
   - **Backend API Docs**: http://localhost:8000/docs
   - **Adminer (Database UI)**: http://localhost:8080
   - **Traefik Dashboard**: http://localhost:8090

10.3 Local Development (Hybrid)

If you prefer running services natively for faster debugging:

- **Backend**:
  1. Navigate to `backend/`.
  2. Install dependencies: `uv sync` (or `pip install -r requirements.txt`).
  3. Run the server: `fastapi dev app/main.py`.
  4. The API will be available at http://localhost:8000.
- **Frontend**:
  1. Navigate to `frontend/`.
  2. Install dependencies: `npm install`.
  3. Run the dev server: `npm run dev`.
  4. The UI will be available at http://localhost:5173.

10.4 Deployment to Production

The project is designed for easy deployment using Docker Compose and Traefik.

1. **Provision a Server**: A VPS (such as AWS EC2, DigitalOcean Droplet) with Docker installed.
2. **DNS Configuration**: Point your domain (such as `livemart.com`) and subdomains (`api.livemart.com`, `dashboard.livemart.com`) to the server IP.
3. **Traefik Setup**:
   - Deploy the `docker-compose.traefik.yml` stack.
   - This sets up the reverse proxy and automatic SSL certificate management (Let's Encrypt).
4. **App Deployment**:
   - Set `ENVIRONMENT=production` in `.env`.
   - Run `docker compose -f docker-compose.yml up -d`.
   - Traefik will automatically detect the new containers and route traffic securely via HTTPS.