Supplier–Consumer Platform (SCP)

Final Software Engineering Report

Suliman Sagindykov, Rustam Amanzhol, Aldiyar Sagat, Dinmukhamed Onglanbek

November 23, 2025

Contents

Contents

***Abstract*—The Supplier–Consumer Platform (SCP) is a digital system designed to streamline interactions between suppliers (owners, managers, sales representatives) and consumers (restaurants and hotels). This final report presents the complete system architecture, UML diagrams, Entity–Relationship Diagram (ERD), interface design, implementation details, testing approach, and individual contributions. Together these components form a coherent end-to-end documentation of the SCP project, demonstrating how requirements were translated into functional software.**

Project Resources

Source Code: GitHub Repository

Video Presentation: Google Drive Video Link

I. Introduction

The Supplier–Consumer Platform (SCP) addresses the operational inefficiencies in business-to-business (B2B) procurement between suppliers and institutional buyers. Traditional ordering relies on manual communication, phone calls, inconsistent catalogs, and slow complaint handling. SCP solves these issues by providing a unified digital platform with verified supplier onboarding, controlled product catalogs, real-time order tracking, and escalation-based complaint resolution.

This report documents the design, architecture, and implementation stages of the SCP project.

II. System Design

A. UML Diagrams Overview

This section contains the full set of UML diagrams created for SCP. Each diagram provides a complementary view of the system's structure and behavior.

Fig. 1: Use Case Diagram for SCP.

1) Use Case Diagram: **Purpose:** Shows all actors (Consumer, Supplier roles, Platform Admin) and their high-level interactions with the system. **Explanation:** Major use cases include managing links, browsing catalogs, placing orders, handling complaints, messaging, and admin moderation.

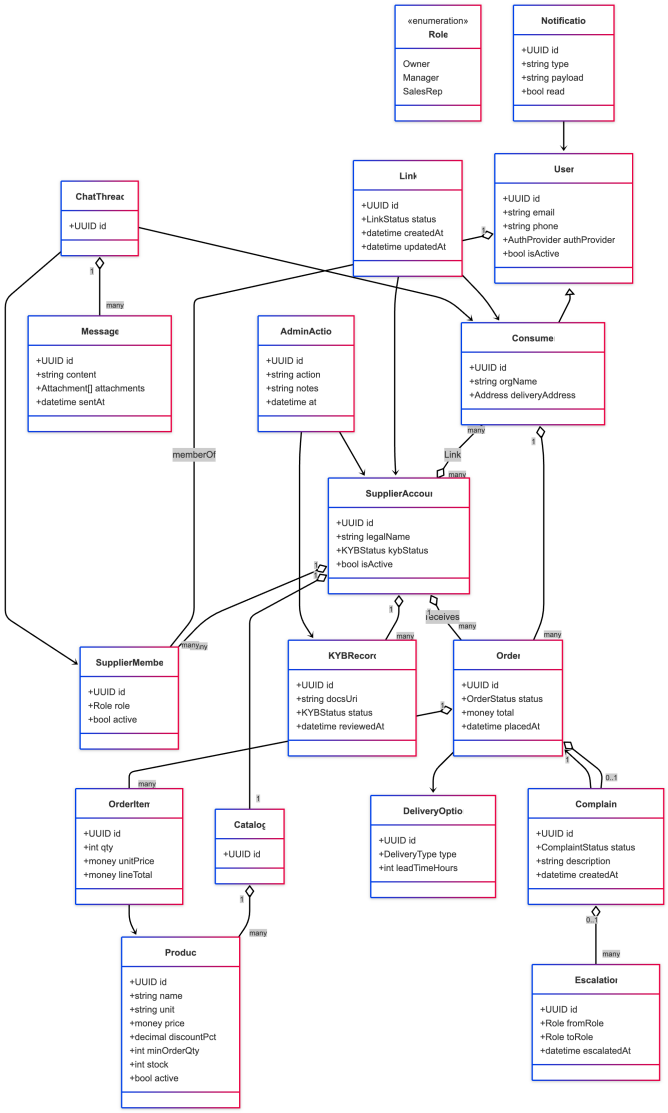

Fig. 2: Domain-level Class Diagram.

2) Class Diagram: **Purpose:** Represents the domain model entities and relationships. **Explanation:** Shows inheritance from User, core entities (Order, Product, Complaint), and administrative structures (KYBRecord, AdminAction).

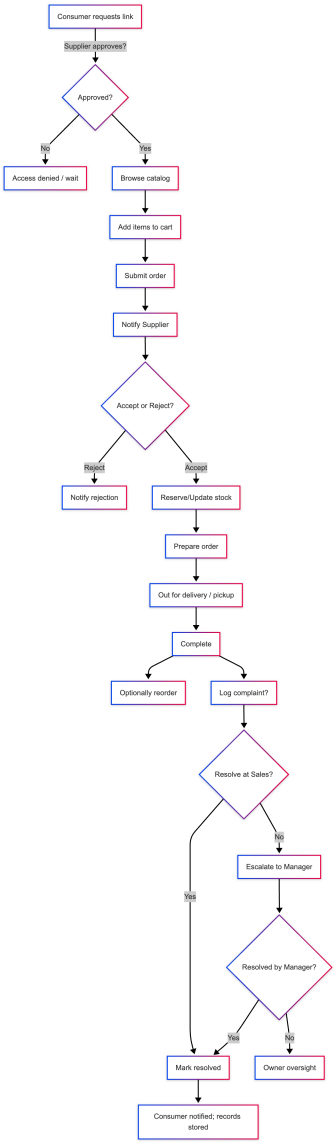

Fig. 3: Order & Complaint Handling Activity Diagram.

3) Activity Diagram: **Purpose:** Illustrates ordering flow, complaint escalation, and decision points.

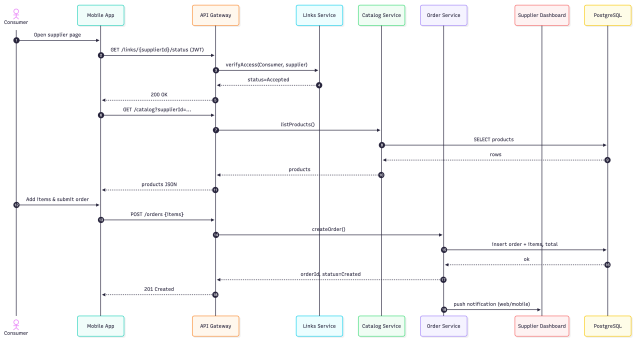

Fig. 4: Sequence Diagram: Place Order.

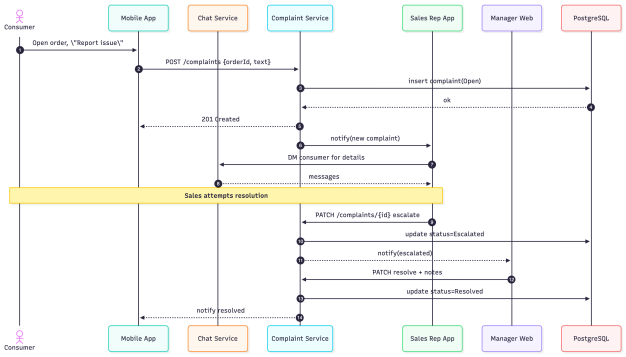

Fig. 5: Sequence Diagram: Complaint Escalation.

4) Sequence Diagrams: **Purpose:** Describes runtime messaging between client apps, API gateway, and backend microservices.

III. Entity–Relationship Diagram (ERD)

A. ERD Diagram

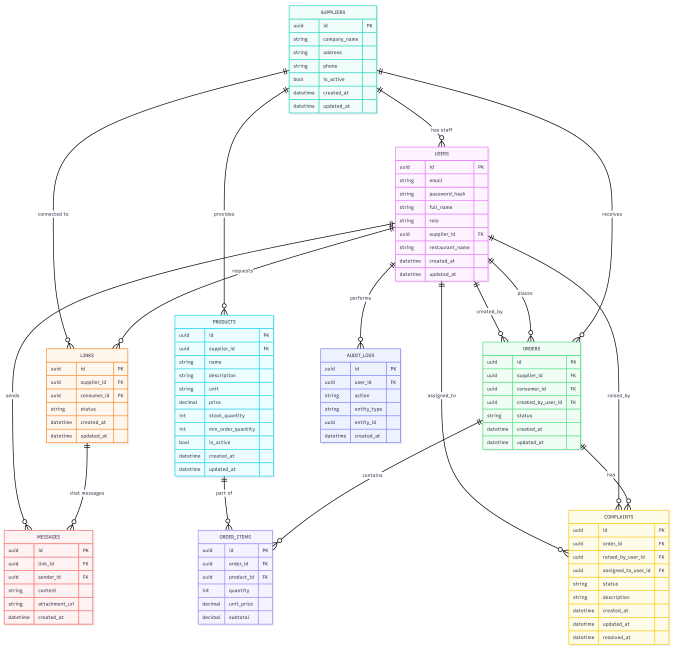

Fig. 6: Entity–Relationship Diagram for SCP.

B. ERD Description

The ERD models all persistent data structures in the SCP:

Major Entities:

- User (PK: user_id)
- SupplierAccount, Consumer (inherit PK from User)
- SupplierMember (FK → SupplierAccount)
- Catalog, Product
- Order, OrderItem
- Link (Consumer–Supplier relationship)
- Complaint, Escalation

- Notification
- ChatThread, Message

Relationship Rules:

- One SupplierAccount owns many SupplierMembers.
- Consumer and Supplier relate via Link (M:N resolved through Link entity).
- Supplier has one Catalog; Catalog has many Products.
- Order is 1:N with OrderItem.
- Complaint may escalate many times (1:N).
- ChatThread is shared by two users; contains many messages.

C. Normalization Note

The schema satisfies:

- **1NF:** All attributes are atomic.
- **2NF:** No partial dependencies; all non-key attributes depend on the full PK.
- **3NF:** No transitive dependencies (e.g., message → user email is avoided).

IV. Interface Design

This section presents the key user interface components of the Supplier–Consumer Platform (SCP). Screenshots illustrate the primary workflows for both Consumers and Supplier Owners. Each figure is accompanied by a short explanation describing its functionality and user interaction flow.

A. Authentication Interfaces

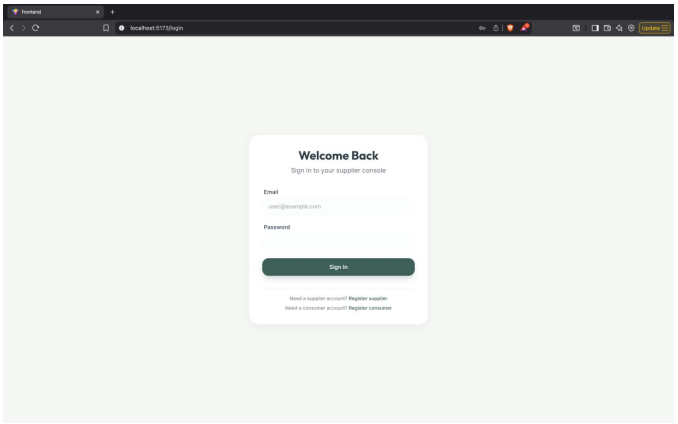

Fig. 7: Sign In Interface

1) Sign In Page: The sign-in interface provides authenticated access to the platform. Users enter their email and password, which are validated through the FastAPI backend using JWT authentication. Upon successful login, the user is redirected to their respective dashboard based on their role (Consumer or Owner).

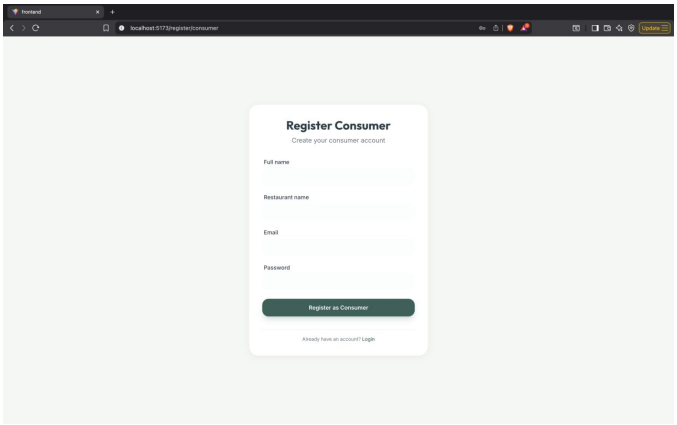

Fig. 8: Consumer Registration Page

B. Consumer Interfaces

1) Consumer Registration Page: This interface allows new consumers to create an account by providing their name, restaurant details, email, and password. Validation ensures correct input formats and prevents duplicate accounts.

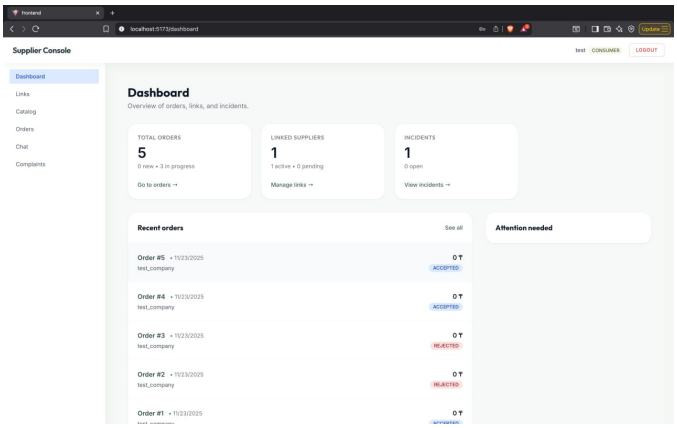

Fig. 9: Consumer Dashboard Overview

2) Consumer Dashboard: The consumer dashboard provides an overview of active orders, supplier links, and any open incidents or complaints. It serves as the central navigation hub for all consumer operations.

3) Links Management (Consumers): Consumers can send link requests to suppliers. Upon approval, the link unlocks access to the supplier's catalog, ordering system, and the chat feature. The interface displays link statuses (Pending, Approved, Declined, Blocked).

4) Product Catalog (Consumers): After linking to a supplier, the consumer can browse the supplier's product catalog. Products include unit type, minimum order quantity, price, and availability. Items can be added directly to the order cart.

5) Orders Page (Consumers): Consumers may create, track, and review orders. Each order displays item details,

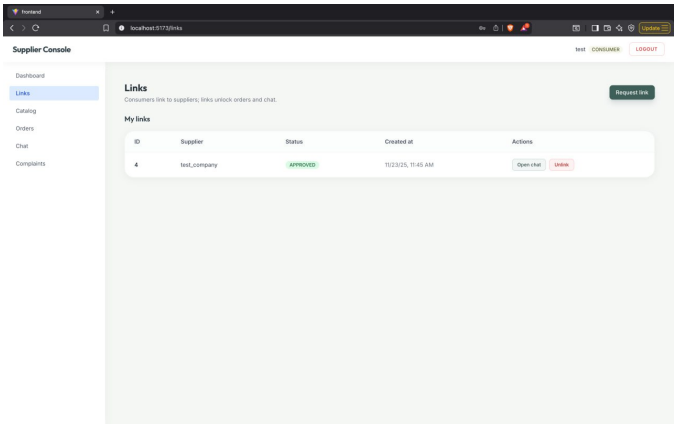

Fig. 10: Consumer Supplier Link Management

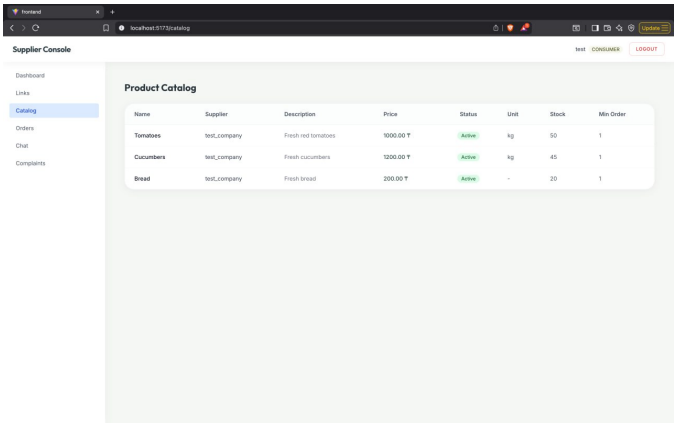

Fig. 11: Consumer Product Catalog

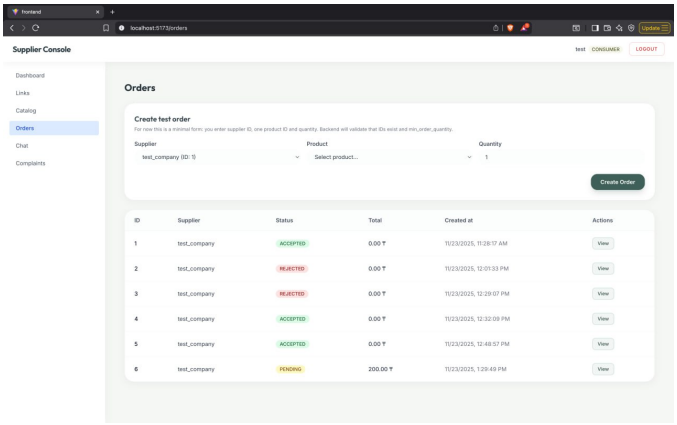

Fig. 12: Consumer Orders Interface

pricing, and current status (Pending, Accepted, Completed, etc.). Past orders remain available for reference.

6) Chat Interface (Consumers): The chat interface facilitates communication between consumers and suppliers. It supports text messages and attachments. Conversations are grouped per supplier link.

7) Complaints Page (Consumers): Consumers may submit complaints regarding specific orders. Complaints

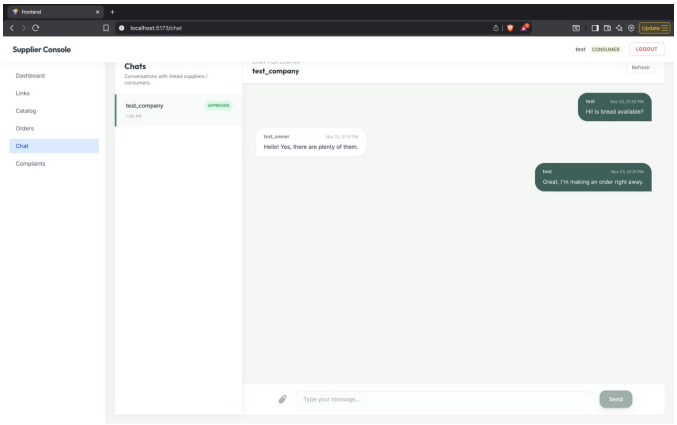

Fig. 13: Consumer Chat Interface

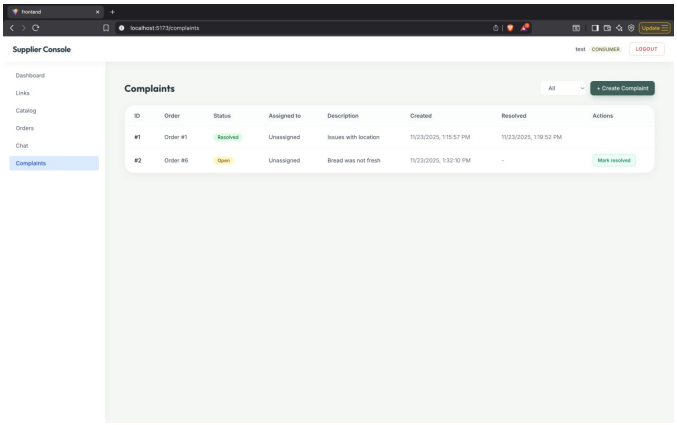

Fig. 14: Consumer Complaint Submission Interface

include a description, timestamps, and current resolution status.

C. Supplier Owner Interfaces

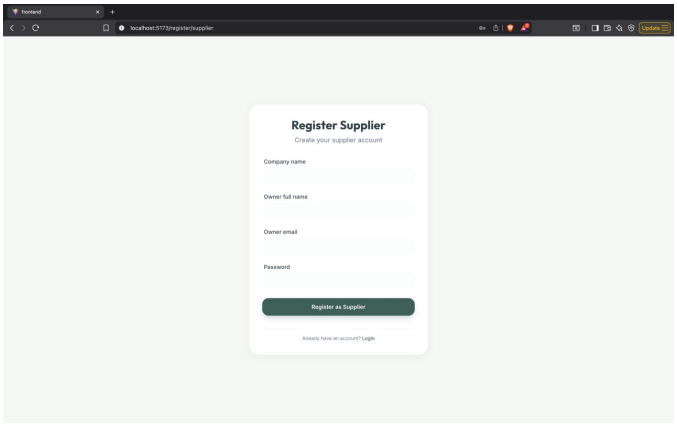

Fig. 15: Supplier Owner Registration Page

1) Owner Registration Page: Supplier owners register their business by entering the company name, address,

phone number, and admin credentials. The system then provisions a new supplier account.

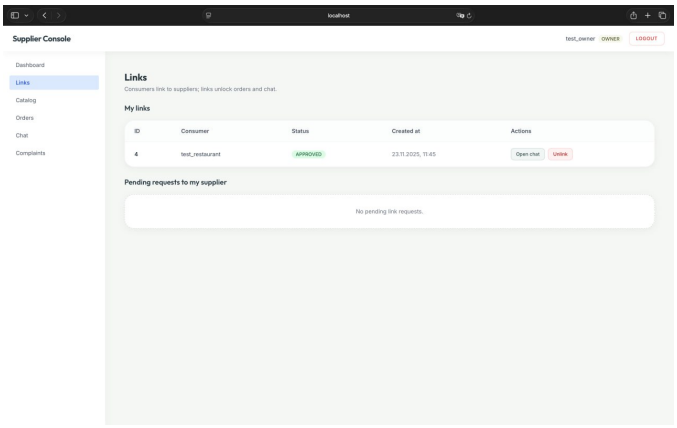

Fig. 16: Supplier Owner Links Management

2) Supplier Links (Owner View): Owners manage incoming link requests from consumers. Approved links enable order placement and chat communication. Owners can block or decline requests if needed.

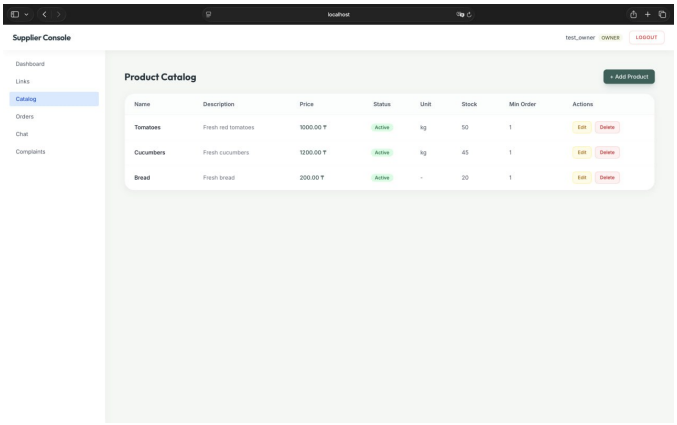

Fig. 17: Supplier Catalog Management Interface

3) Catalog Management (Owners): Owners can create, update, or remove products from their catalog. Editing a product updates information such as price, stock, unit type, and minimum order quantity.

4) Chat Interface (Owners): Supplier owners communicate directly with linked consumers. The interface mirrors the consumer chat experience for consistency.

5) Complaints Panel (Owners): Owners can view complaints submitted by consumers. The system allows owners to assign complaints to themselves or other staff members for resolution.

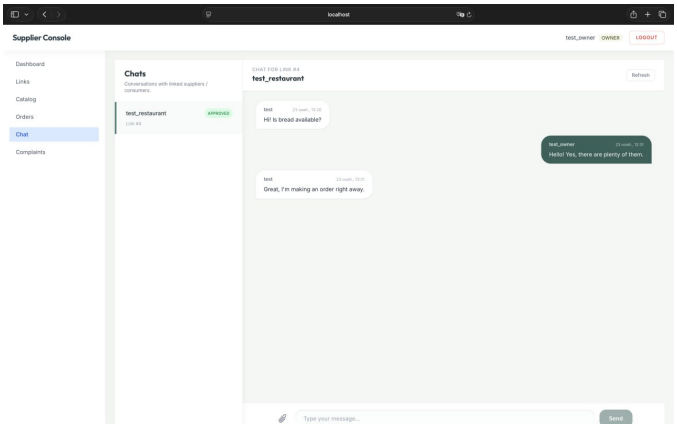

Fig. 18: Supplier Chat Interface

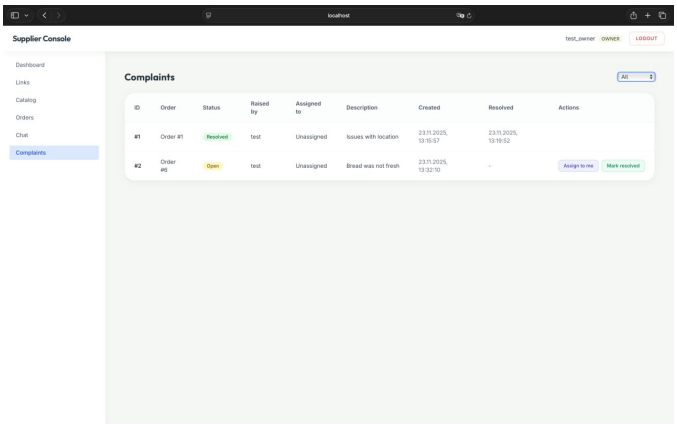

Fig. 19: Supplier Owner Complaint Management Interface

V. Implementation Details

The Supplier–Consumer Platform (SCP) was implemented using a modern full-stack architecture that combines a Python backend, a cross-platform mobile application, and a responsive web frontend.

A. Backend Technologies

- **FastAPI** — High-performance Python web framework powering all REST API endpoints.
- **SQLAlchemy** — ORM used for data modeling and database interaction.
- **PostgreSQL** — Relational database used for persistent storage of system data.
- **Alembic** — Handles database schema migrations and version control.
- **JWT (PyJWT)** — Ensures secure authentication through JSON Web Tokens.
- **Passlib (Bcrypt)** — Provides secure password hashing for user accounts.

B. Frontend (Web) Technologies

- **React** — JavaScript framework used to develop the SCP web dashboard for suppliers and admins.

- **Tailwind CSS** — Utility-first CSS framework enabling fast, consistent, and responsive UI design.

C. Mobile Application

- **Flutter** — Cross-platform mobile framework used to build the mobile application for consumers and suppliers, ensuring native performance and unified UI.

This combined tech stack provides a scalable backend, secure authentication workflow, and a seamless user experience across both mobile and web platforms.

VI. API Documentation (Swagger UI)

Swagger UI provides an automatically generated, interactive documentation interface for all REST API endpoints of the Supplier–Consumer Platform (SCP). The FastAPI backend exposes a complete OpenAPI specification, enabling developers and testers to inspect endpoints, validate input/output schemas, and execute requests directly from the browser.

A. Swagger Overview

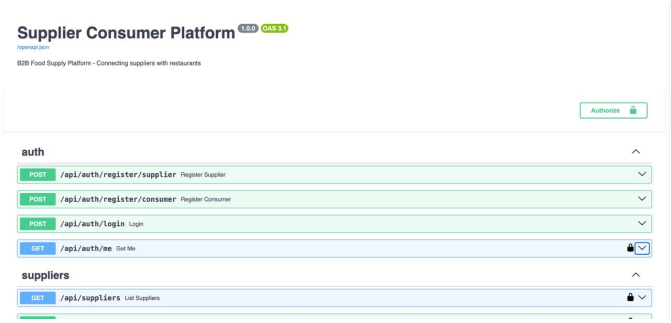

Fig. 20: Swagger UI: Authentication, Supplier, and Link Endpoints

Figure 20 displays the first part of the API documentation. It includes:

- User registration and authentication endpoints
- Supplier listing and creation APIs
- Link request workflow (request, approve, reject, block)

These endpoints define the foundation of the supplier–consumer relationship in the platform.

Fig. 21: Swagger UI: Products, Orders, Complaints, and Messaging

Figure 21 shows the operational part of the API:

- Product catalog management (supplier-side)
- Order creation, status updates, and order tracking
- Complaint submission, escalation, and resolution
- Messaging endpoints for real-time communication

Together, these APIs ensure full coverage of platform functionality for mobile and web clients.

B. API Use in Development

The Swagger UI was actively used throughout development to:

- Test endpoint correctness and error handling
- Validate JSON schemas for all request/response models
- Debug authentication and role-based access restrictions
- Ensure smooth integration between Flutter, React, and FastAPI

This interactive documentation significantly improved backend testing efficiency and reduced integration errors across the system.

VII. Testing and Evaluation

Testing methods:

- Unit testing (Jest/PyTest)
- Integration tests for API endpoints
- UI testing of mobile screens
- Load tests for high-volume ordering

VIII. Individual Contributions

Suliman Sagindykov: Backend services, API gateway, database schema. **Rustam Amanzhol:** Mobile application. **Dinmukhamed Onglanbek:** Organization, Task assignment, Business flow, UML diagrams, ERD, documentation. **Aldiyar Sagat:** UI/UX design, frontend implementation.

IX. Conclusion and Future Work

SCP successfully digitizes B2B supplier–consumer operations with ordering, catalog management, complaints, and chat. Future work includes:

- AI-assisted order predictions
- Supplier recommendation engine
- Advanced reporting dashboards