VILNIUS UNIVERSITY
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
SOFTWARE ENGINEERING

Software engineering course

Point of sale system modeling

(Group 3: Nojus Cininas, Tomas Martinka, Valdemar Kobelis,
Group 5: Eimantas Gagilas, Rokas Makaveckas)

Vilnius
2025

Turinys

1. Intro

1.1. Problem statement

Bars, cafes and beauty services are using systems that are hard to manipulate and operate for older workers and require a lot of onboarding to learn. These issues introduce friction and hassle for employees while serving and finalizing customer transactions, especially when they are already ready to leave and want to get out quickly or when making reservations on customer behalf / letting them make reservations. This leaves users dissatisfied, and the customers are less likely to use the service again, especially in this world where everything moves quickly and our service is slow. They may even forgo reservation if the process is too difficult to understand.

It is important to resolve these issues, because then we can maximize profits and ensure good customer experience, that makes them want to come back and use our service again.

1.2. Proposed solution

We propose a modern, cloud-based SaaS Point of Sale (POS) system designed for small and medium businesses in the catering and beauty sectors. The system will feature a clean, intuitive user interface accessible via web and mobile devices, ensuring ease of use for all employees.

Our solution will streamline core business operations through two main modules: an advanced Order Management system and a comprehensive Service and Reservation system. Key features include flexible order creation, integrated payment processing (including credit cards via Stripe), split checks, and robust appointment scheduling.

The system is built on a modular, multi-tenant architecture, allowing businesses to subscribe to plans that fit their specific needs. It includes administrative tools for managing products, services, taxes, discounts, and employees, with role-based access control to ensure data security.

2. Business flow

2.1. Client order business flow

1. The client arrives at the business and selects their product
2. The staff proceed to the client
3. The staff inquires about their selection
4. The client verbally explains their choice to the Staff
5. The staff note down the client's selection
6. The gathered data gets relayed to the product producing staff
7. The user might change their mind
   (a) Staff come over to document the change
   (b) They note down the change, either removing or adding additional selections
   (c) The updated info gets relayed to the staff responsible for the production of the product and documented in the receipt
8. The staff make the product
9. The serving staff deliver the product
10. The client asks to pay
11. The serving staff create a receipt
12. The receipt is sent to the client

2.1.1. Client payment business flow

1. The client selects to pay with:
    (a) Cash
        i. The client pays
        ii. The serving staff take the money and add it in the teller, if needed bring change for the client
    (b) Credit card/Apple pay/Google pay etc.
        i. Staff enters the sum in a generic Point of Sale system
        ii. The staff let the client insert/tap their card
        iii. Client taps their card/phone or insert their card
        iv. The transaction gets approved:
            A. A receipt is printed that the transaction is done
        v. The transaction gets denied:
            A. A receipt is printed with the cause of the failure and the failure is shown on the screen
            B. The device provides two choices either to repeat the process or cancel the operation
            C. This is done until the transaction is completed or the problem get resolved other way

2.2. Reservation management business flow

1. Client contacts the business (phone call, email)
2. Receptionist asks about inquiry
3. Client inquires about specific service
4. Receptionist looks up qualified staff in the system
5. Receptionist checks staff schedule in the system
6. Receptionist suggests time and date to the client
7. Client accepts
8. Receptionist adds new entry to schedule in the system
9. System sends SMS notification/reminder to the client (Bonus)
10. Client confirms visit

3. High level architecture

The system is designed with a modular, multi-tenant architecture to serve multiple businesses securely and efficiently. It consists of a frontend application, a backend API gateway, and several microservices, each responsible for a distinct domain of functionality.

- Frontend Application: A responsive web application that provides the user interface for employees.
- API Gateway: A single entry point for all client requests. It handles authentication and routing.
- Identity & Access Management (IAM) Service: Manages users, roles, permissions, and authentication.
- Merchant Service: Manages business-specific information like company details and locations.
- Order Service: Handles all logic related to orders, including creation, modification, and payment.
- Reservation Service: Manages appointment booking, scheduling, and customer notifications.
- Product & Service Catalog Service: A central repository for managing menu items and professional services.
- Inventory Service (Bonus): Tracks stock levels for products.
- Finance Service: Manages taxes, discounts, and service charges.
- Database: Each service has its own database to ensure loose coupling.

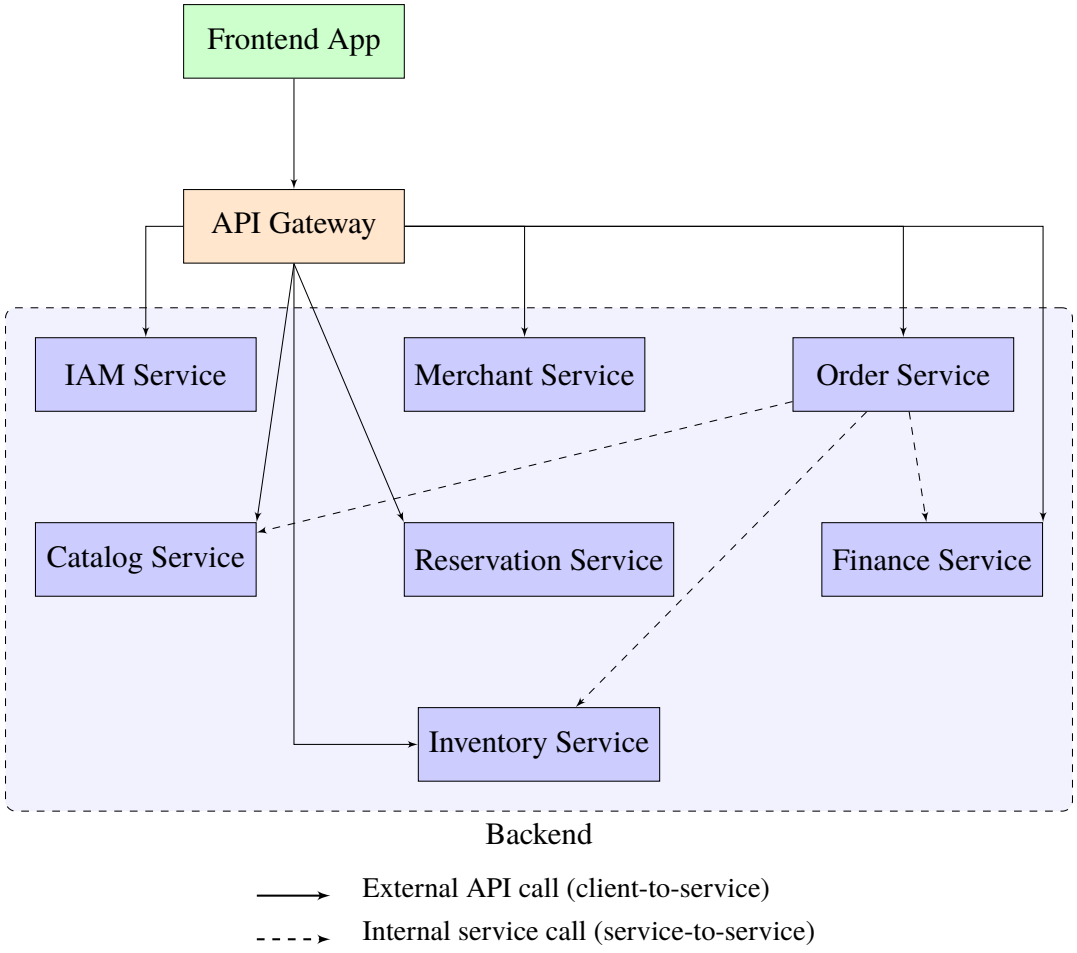

Figure 1: High-Level Package Diagram

4. Lo-Fi Wireframes API Overview

Download the full wireframes API PDF here for better quality: https://github.com/IceHawkey9588/PoS-PSP/blob/main/PoS-Python-wireframes-api.pdf

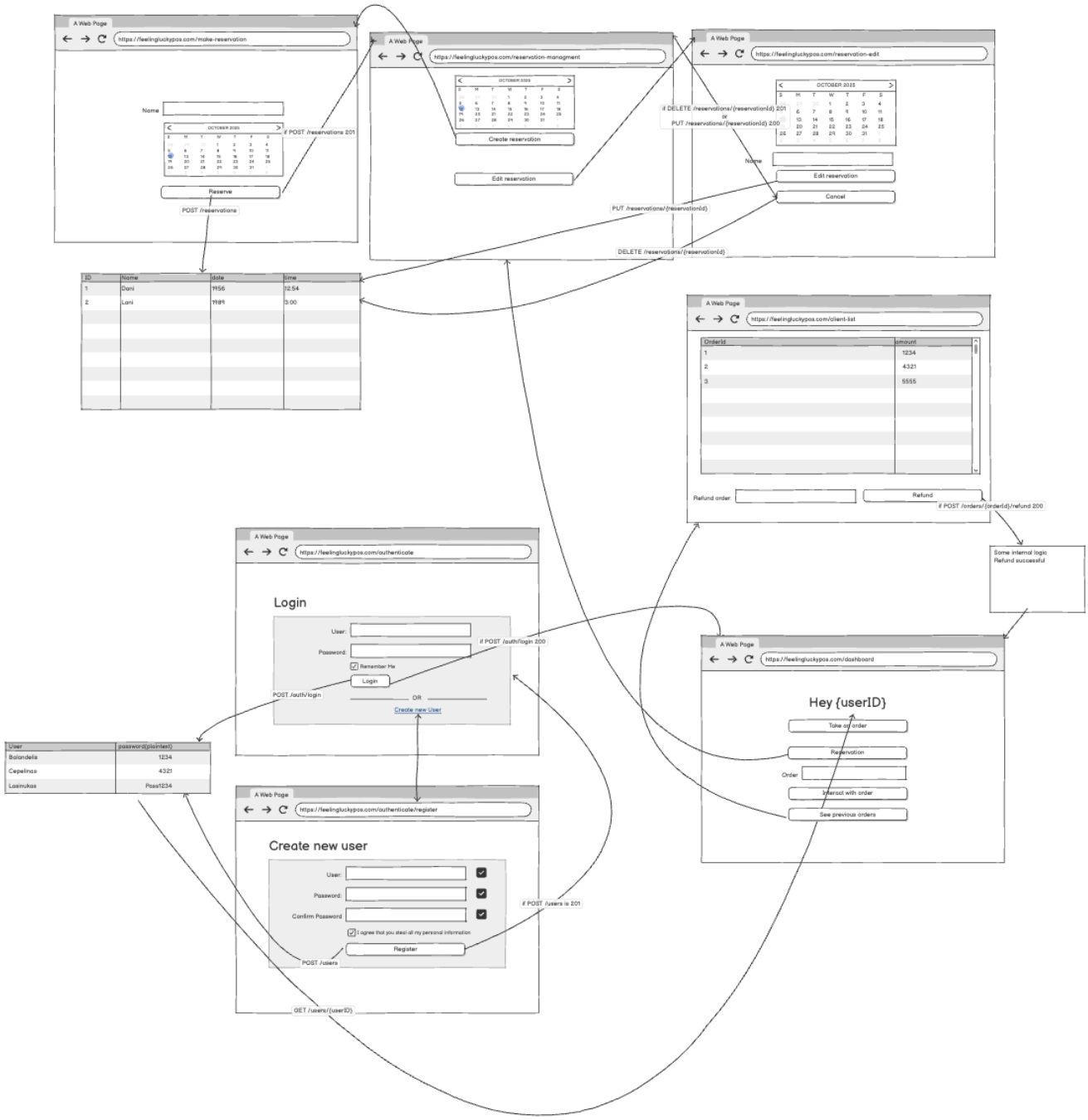

Figure 2: Auth, Reservations, Refunds

Description.

- Authentication flow: login and create new user (registration) with validation.
- Dashboard provides quick navigation to reservations, orders, and refunds.
- Reservation management: create from calendar, edit time/assignee, or cancel; list/table view shows upcoming bookings.
- Refund flow: select a settled payment or order line and issue a full/partial refund; confirmation banner on success.
- Data views emphasize clarity for front-desk staff: tables for lists, calendars for availability, and concise action buttons.
- Typical API actions: auth (login/register), reservations (create/edit/cancel/list), and payments (refund).

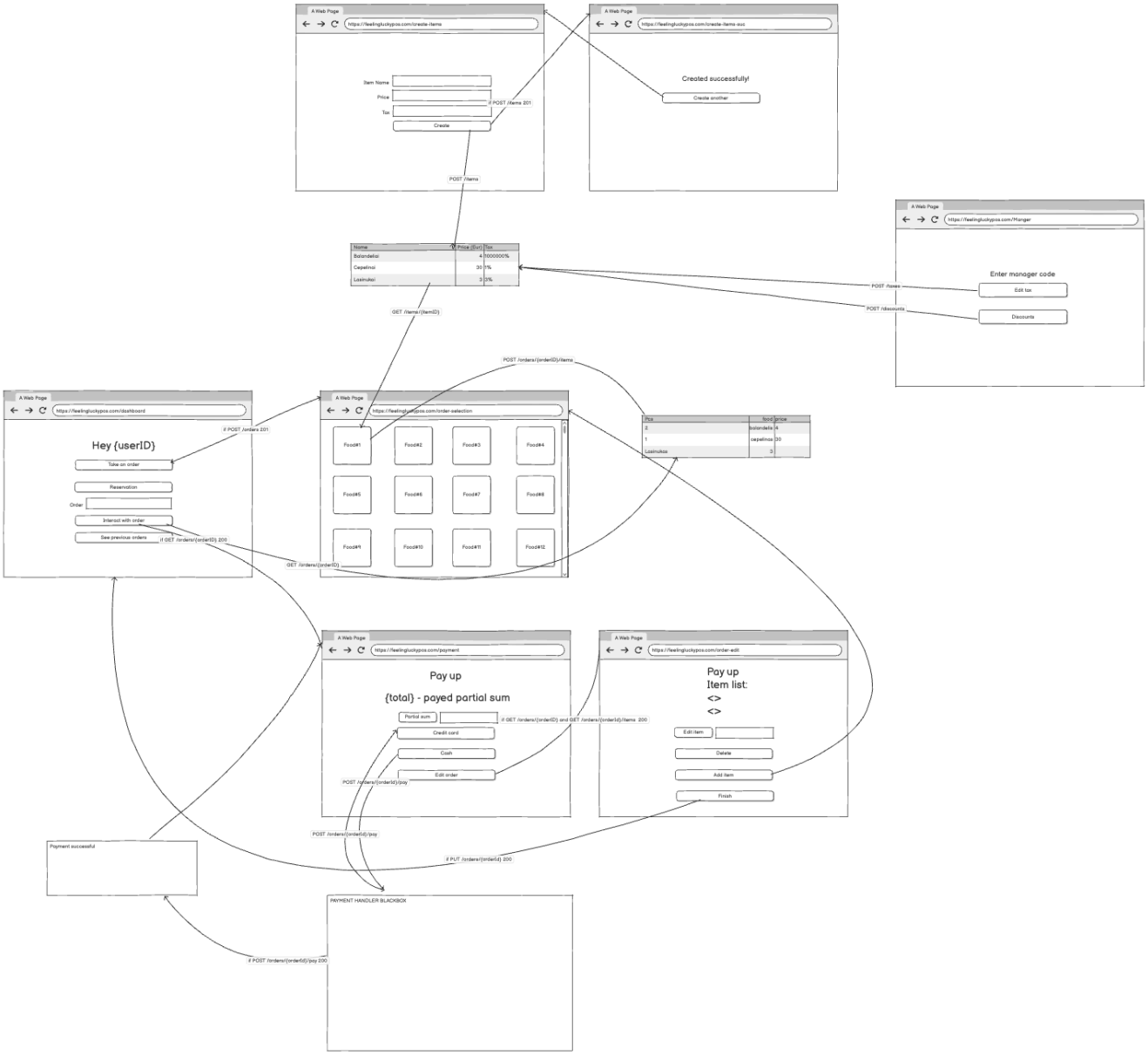

Figure 3: Ordering and payments

Description.

- Staff logs in and lands on a simple dashboard with quick actions (start new order, view/open orders, apply manager functions).
- Menu grid shows selectable items; selections are added to the current order with running totals and item table.
- Manager code screen gates privileged actions (e.g., applying discounts, editing menu).
- “Pay up” view supports split/partial payments; shows total and already paid sum.
- Payment methods: cash or card/wallet; success/failure banners confirm result and receipt handling.
- Order lifecycle: create order, add/remove items, optionally apply discount, take payment(s), and close the check.

5. Data model

The data model is designed to be robust and scalable, capturing the relationships between the core entities of the system. A class diagram is used to illustrate the entities, their attributes, and their associations.

Key entities include:

- User: For authentication and authorization.
- Role: Defines user permissions.
- Merchant: The root entity for a business.
- Product: For the catering menu or saleable goods.
- ProductVariation: For different product options.
- Service: For bookable appointments in the beauty sector.
- Order: To track customer sales.
- OrderItem: Individual item within an order.
- Payment: To handle multiple payment types and split checks.
- Reservation: To manage appointments.
- TaxRate: For current financial calculations.
- HistoricalTaxRate: For historical financial calculations.
- Discount: For price reduction calculations.

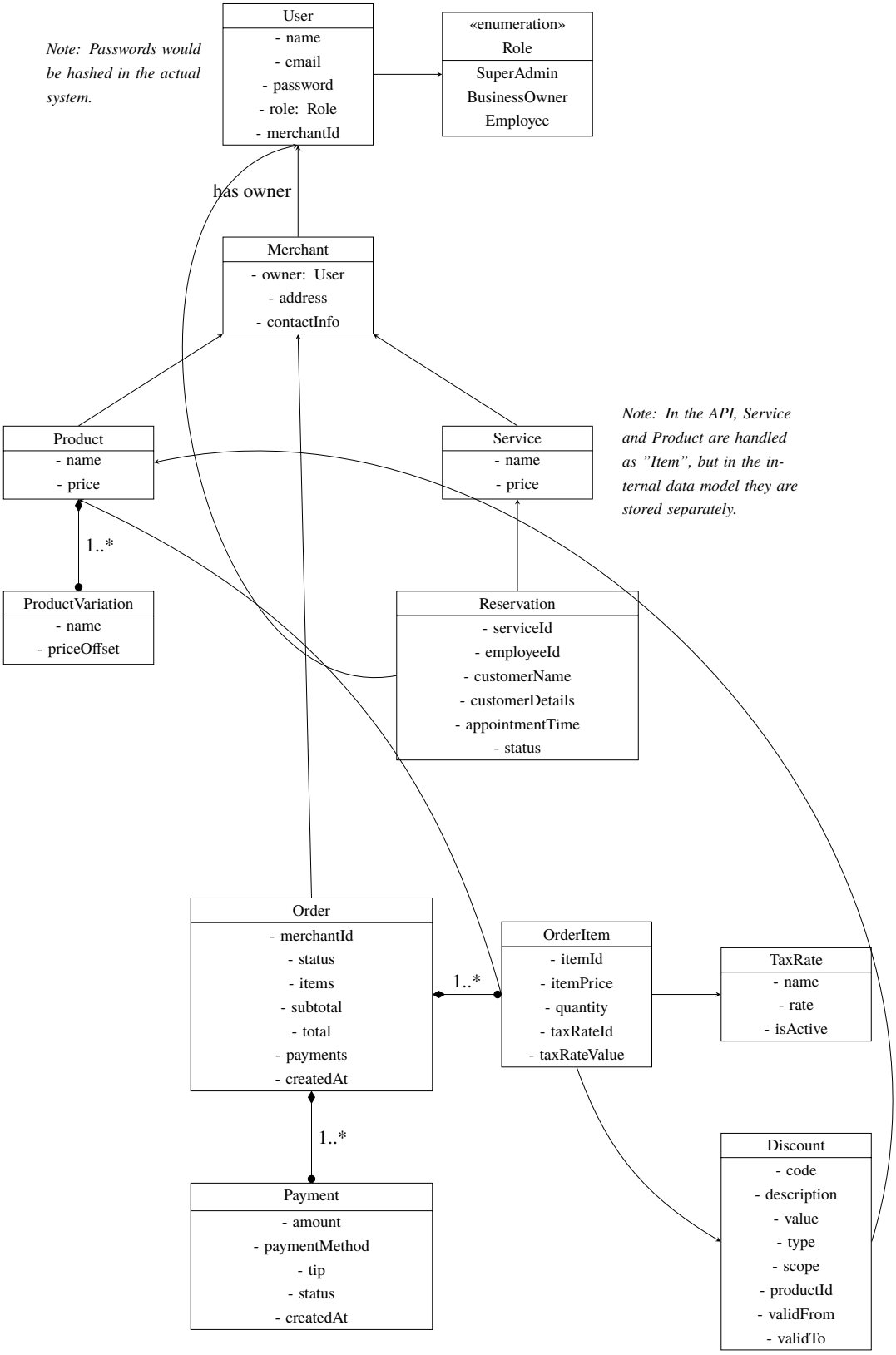

Figure 4: System Data Model (Class Diagram)

6. Swagger API documentation

Can be accessed here:
https://github.com/IceHawkey9588/PoS-PSP/blob/main/pos-python.yaml