Technical Report – UYP-PMS

Frontend Architecture, Integration & System Behavior

(Personal Technical Contribution)

---

1. Project Overview

Umelaphi Yaodian Pharmacy – Prescription Management System (UYP-PMS) is a full-stack web application designed to model real pharmacy prescription workflows. The system supports role-based access and structured user interactions through a web interface connected to a backend API and a relational database.

This technical report focuses specifically on **my personal contribution**, which consisted of the **entire functional frontend implementation**, including application logic, routing, services, state handling, and frontend–backend integration, as documented in the final project deliverable

---

2. Frontend Responsibilities and Scope

I was responsible for **all frontend development beyond visual styling**, including:

- Application structure and routing
- Business logic implemented in **TypeScript**
- Angular services and API communication
- Role-based behavior and access on the client side
- End-to-end interaction between UI actions and backend responses
- Debugging and stabilizing frontend behavior during final integration

Visual design (HTML/CSS styling and layout) and backend development were handled by other team members.
I was responsible for the complete **functional frontend implementation**, including Angular components, routing, TypeScript logic, services, state management, and frontend–

backend integration.

---

3. Frontend Architecture (Angular)

The frontend is built using **Angular** and follows a modular structure aligned with SPA best practices.

Key architectural elements:

- **Component-based UI** for login, dashboard, prescription creation, and preparation views
- **Angular routing** to control navigation based on user role
- **Services layer** to encapsulate all HTTP communication with the backend
- Centralized handling of authentication state and role context
- Separation between UI rendering and application logic

This structure ensures maintainability, clarity, and scalability of the frontend codebase.

---

4. Frontend–Backend Integration

A major part of my work was ensuring **seamless integration** between the Angular frontend and the NestJS backend.

This included:

- Configuring API endpoints consumption in Angular services
- Handling asynchronous data loading and error states
- Mapping backend responses to frontend models
- Ensuring frontend behavior correctly reflected backend rules and role permissions
- Debugging issues related to:
    - API routes
    - Prisma schema mismatches
    - Authentication and role propagation

Significant effort was required during the final phase to **resolve blocking integration issues** and ensure the application functioned correctly from login to prescription preparation