Design Document for Saint Cy's Hospital

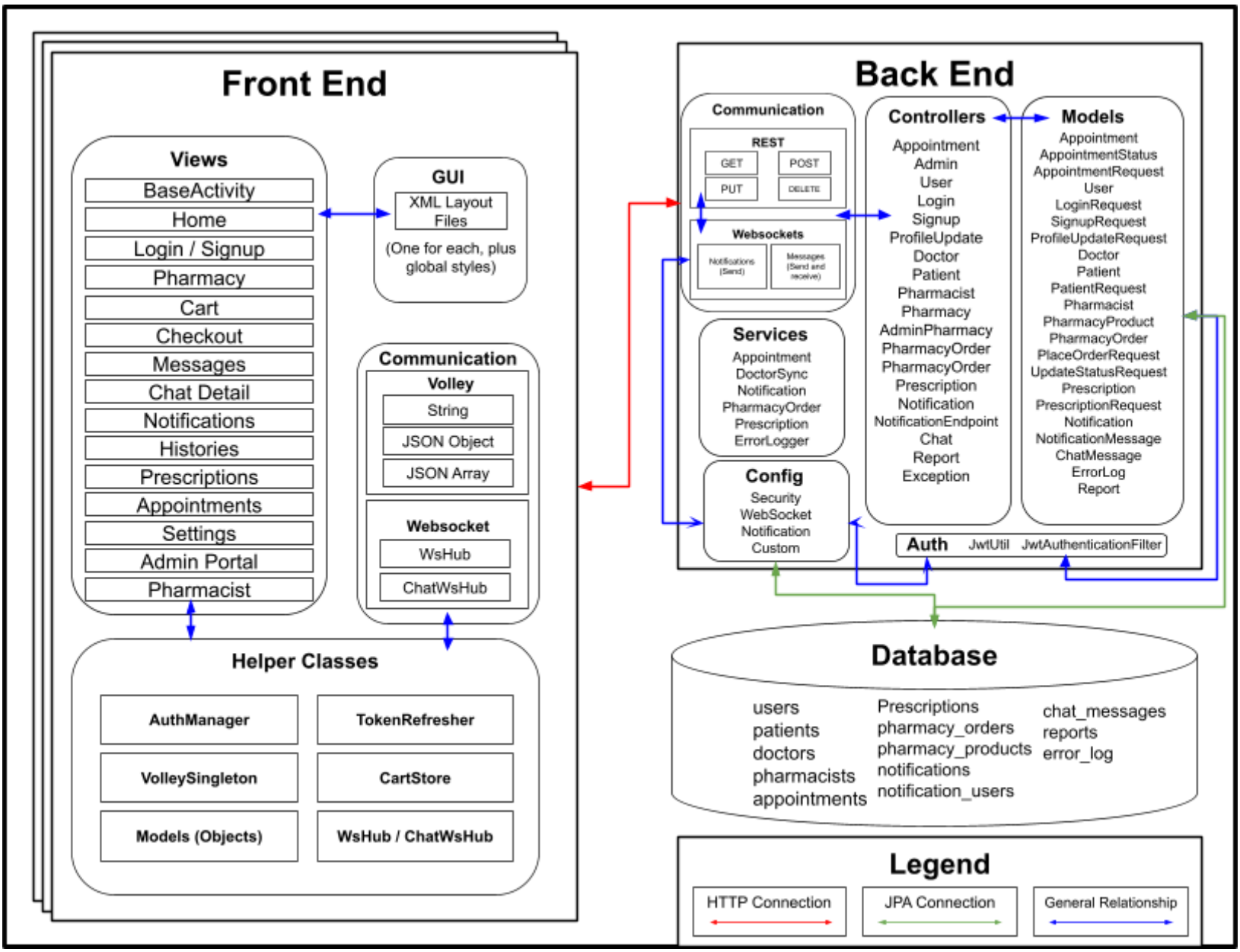
Group <3_hosen_5>

Trice Buchanan: 25% contribution

Neal Sharma: 25% contribution

Quinn Beckman: 25% contribution

Devank Uppal: 25% contribution


Front End
Views
BaseActivity
Home
Login / Signup
Pharmacy
Cart
Checkout
Messages
Chat Detail
Notifications
Histories
Prescriptions
Appointments
Settings
Admin Portal
Pharmacist
GUI
XML Layout Files
(One for each, plus global styles)
Communication
Volley
String
JSON Object
JSON Array
Websocket
WsHub
ChatWsHub
Helper Classes
AuthManager
TokenRefresher
VolleySingleton
CartStore
Models (Objects)
WsHub / ChatWsHub
Back End
Communication
REST
GET
POST
PUT
DELETE
Websockets
Notifications (Send)
Messages (Send and receive)
Controllers
Appointment
Admin
User
Login
Signup
ProfileUpdate
Doctor
Patient
Pharmacist
Pharmacy
AdminPharmacy
PharmacyOrder
PharmacyOrder
Prescription
Notification
NotificationEndpoint
Chat
Report
Exception
Models
Appointment
AppointmentStatus
AppointmentRequest
User
LoginRequest
SignupRequest
ProfileUpdateRequest
Doctor
Patient
PatientRequest
Pharmacist
PharmacyProduct
PharmacyOrder
PlaceOrderRequest
UpdateStatusRequest
Prescription
PrescriptionRequest
Notification
NotificationMessage
ChatMessage
ErrorLog
Report
Services
Appointment
DoctorSync
Notification
PharmacyOrder
Prescription
ErrorLogger
Config
Security
WebSocket
Notification
Custom
Auth
JwtUtil
JwtAuthenticationFilter
Database
users
patients
doctors
pharmacists
appointments
Prescriptions
pharmacy_orders
pharmacy_products
notifications
notification_users
chat_messages
reports
error_log
Legend
HTTP Connection
JPA Connection
General Relationship

This section explains the main parts of the Saint Cy's Hospital system that required the most coordination between the Android app and the backend. These areas stood out as the most involved sections of the project: Pharmacy ordering, real-time messaging, authentication and token management, and the appointment scheduling flow. The goal is to describe how each part works in practice without over-formal wording.

1. Pharmacy Ordering and Checkout

The pharmacy system enables patients to browse available medications, review details, and place orders. On Android, PharmacyActivity loads the list of items through simple GET requests using the Volley library. When a patient submits an order, the checkout screen sends the selected medicine IDs and quantities to the backend.

The backend handles this through PharmacyController, PlaceOrderRequest, PharmacyRepository, and related classes. The service layer checks that the requested items are in stock, reduces the inventory, and records the prescription. Since multiple tables are changed simultaneously, the backend uses transactions to maintain consistency. This part took extra care to ensure that the app and backend always displayed the correct inventory and order status.

2. Real-Time Messaging

The chat system is one of the most interactive features in the project. The Android app uses WebSocketClient, WsHub, and ChatDetailActivity to open a connection to the server. When a user enters a chat room, the app connects and listens for incoming messages.

On the backend, classes like ChatController, ChatMessageRep, and ChatRoomServer handle incoming messages. Each message is saved to the database and then sent to everyone else in the same room. The app receives messages right away, so conversations update instantly. Because this involves background listeners, UI updates, and database writes all happening simultaneously, it was one of the more challenging parts to get right.

3. Authentication and Token Refresh

The project supports various user types, including patients, doctors, administrators, and pharmacists. After a user logs in, the Android app stores the JWT token and attaches it to all requests. TokenRefresher ensures the user is not logged out unexpectedly by refreshing the token when necessary.

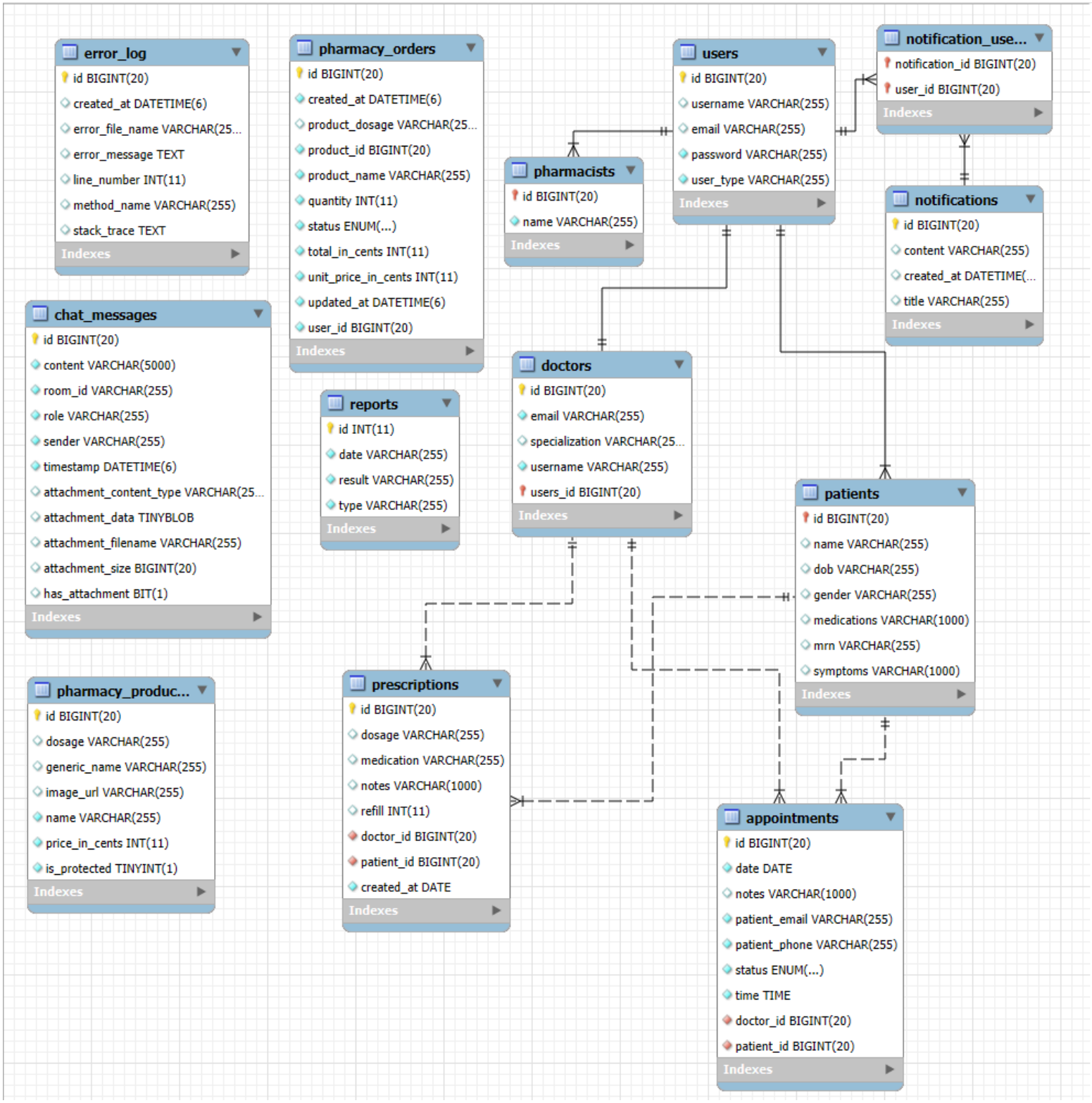
On the backend, classes such as JwtAuthenticationFilter, JwtUtil, and SecurityConfig verify and validate each token. Different controllers only allow access to users with the correct roles and permissions. Since this system is used in every major part of the project, getting the token flow right was crucial for avoiding errors and ensuring that users only see the features they are authorized to access.

4. Appointment Scheduling

Patients schedule appointments through a simple doctor-first workflow. The Android app loads the list of doctors, and once a user picks one, the app requests that doctor's available times. After selecting a slot and appointment type, the app sends the request to the backend.

The backend checks whether the time is available and creates the appointment. Classes like AppointmentController, AppointmentRequest, and AppointmentService handle the logic, while the database stores the final result. This part required coordination between the app's calendar interface and the backend checks to avoid conflicts.

Overall, these features combine several moving parts, including live updates, secure access, multi-step workflows, and database operations. They represent the most involved and team-focused sections of our Saint Cy's Hospital project.


error_log
id BIGINT(20)
created_at DATETIME(6)
error_file_name VARCHAR(25...
error_message TEXT
line_number INT(11)
method_name VARCHAR(255)
stack_trace TEXT
Indexes
pharmacy_orders
id BIGINT(20)
created_at DATETIME(6)
product_dosage VARCHAR(25...
product_id BIGINT(20)
product_name VARCHAR(255)
quantity INT(11)
status ENUM(...)
total_in_cents INT(11)
unit_price_in_cents INT(11)
updated_at DATETIME(6)
user_id BIGINT(20)
Indexes
users
id BIGINT(20)
username VARCHAR(255)
email VARCHAR(255)
password VARCHAR(255)
user_type VARCHAR(255)
Indexes
notification_use...
notification_id BIGINT(20)
user_id BIGINT(20)
Indexes
pharmacists
id BIGINT(20)
name VARCHAR(255)
Indexes
notifications
id BIGINT(20)
content VARCHAR(255)
created_at DATETIME(...
title VARCHAR(255)
Indexes
chat_messages
id BIGINT(20)
content VARCHAR(5000)
room_id VARCHAR(255)
role VARCHAR(255)
sender VARCHAR(255)
timestamp DATETIME(6)
attachment_content_type VARCHAR(25...
attachment_data TINYBLOB
attachment_filename VARCHAR(255)
attachment_size BIGINT(20)
has_attachment BIT(1)
Indexes
reports
id INT(11)
date VARCHAR(255)
result VARCHAR(255)
type VARCHAR(255)
Indexes
doctors
id BIGINT(20)
email VARCHAR(255)
specialization VARCHAR(25...
username VARCHAR(255)
users_id BIGINT(20)
Indexes
patients
id BIGINT(20)
name VARCHAR(255)
dob VARCHAR(255)
gender VARCHAR(255)
medications VARCHAR(1000)
mrn VARCHAR(255)
symptoms VARCHAR(1000)
Indexes
pharmacy_produc...
id BIGINT(20)
dosage VARCHAR(255)
generic_name VARCHAR(255)
image_url VARCHAR(255)
name VARCHAR(255)
price_in_cents INT(11)
is_protected TINYINT(1)
Indexes
prescriptions
id BIGINT(20)
dosage VARCHAR(255)
medication VARCHAR(255)
notes VARCHAR(1000)
refill INT(11)
doctor_id BIGINT(20)
patient_id BIGINT(20)
created_at DATE
Indexes
appointments
id BIGINT(20)
date DATE
notes VARCHAR(1000)
patient_email VARCHAR(255)
patient_phone VARCHAR(255)
status ENUM(...)
time TIME
doctor_id BIGINT(20)
patient_id BIGINT(20)
Indexes