Design Document for <<NAILSYNC>>

Group <2 Swarna 7>

Member1 Jacob: 25% contribution

Member2 Jordan: 25% contribution

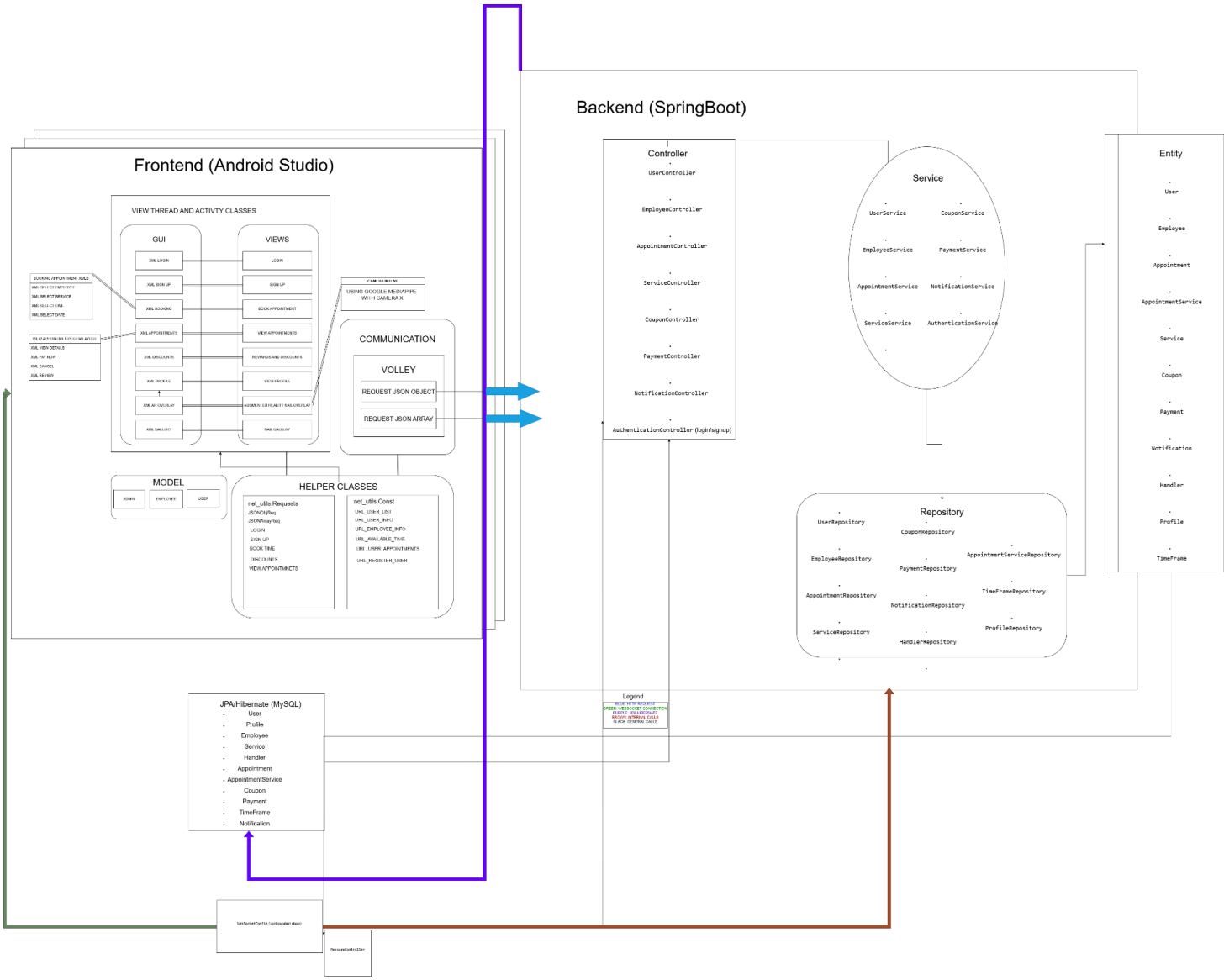
Member3 Phong: 25% contribution

Member4 Bao: 25% contribution

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE! (Create the picture using pencil or drawIO)

Please zoom in for better views:

https://drive.google.com/file/d/18KtnICNiUyO0Rx3g-MlVV0cdE_6m1ec7/view?usp=sharing



Use this third page to describe complex parts of your design.

Frontend:

The frontend of the NAILSYNC application is designed as a responsive web application that provides an intuitive user experience for customers, employees, and future admin users.

It is organized into multiple screens corresponding to major use-cases such as Login/Signup, Dashboard, Appointment Booking, Profile Management, Messaging, and Password Reset.

Each screen contains key UI elements (buttons, input fields, displays) and supports actions like booking appointments, chatting, editing profiles, and viewing offers. The frontend communicates with the backend server via HTTP (REST APIs) for normal data interactions, and WebSocket connections for real-time messaging and notifications.

The app is designed with cross-platform compatibility in mind, ensuring that it works smoothly on both desktop and mobile browsers. Future expansion includes turning it into a PWA (Progressive Web App) or a native mobile application.

To enhance security, sensitive actions like profile updates or account deletion require verification via a secure code sent to the user's registered email.

Overall, the frontend emphasizes ease of use, speed, and security to deliver a seamless salon booking and communication experience.

Backend:

Our backend system for NAILSYNC is implemented using Spring Boot with Apache Tomcat as the embedded server. It is organized into three primary packages following a layered architecture: controller, service, and repository, along with entity models under a model package.

The Controller layer handles HTTP requests (e.g., appointment booking, user login, chatting) and passes them to the Service layer, which contains the business logic. The Service layer then interacts with the Repository layer to perform CRUD operations on the database using Spring Data JPA with MySQL as the database engine.

WebSocket technology is used to support real-time chat between customers and employees and to deliver instant notifications.

Security is enforced through verification codes for critical actions such as password resets, profile edits, and account deletions. The modular structure ensures high scalability, security, and performance, supporting multiple concurrent users and future feature expansions (e.g., admin role management).

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)