**Design Document for workAround**

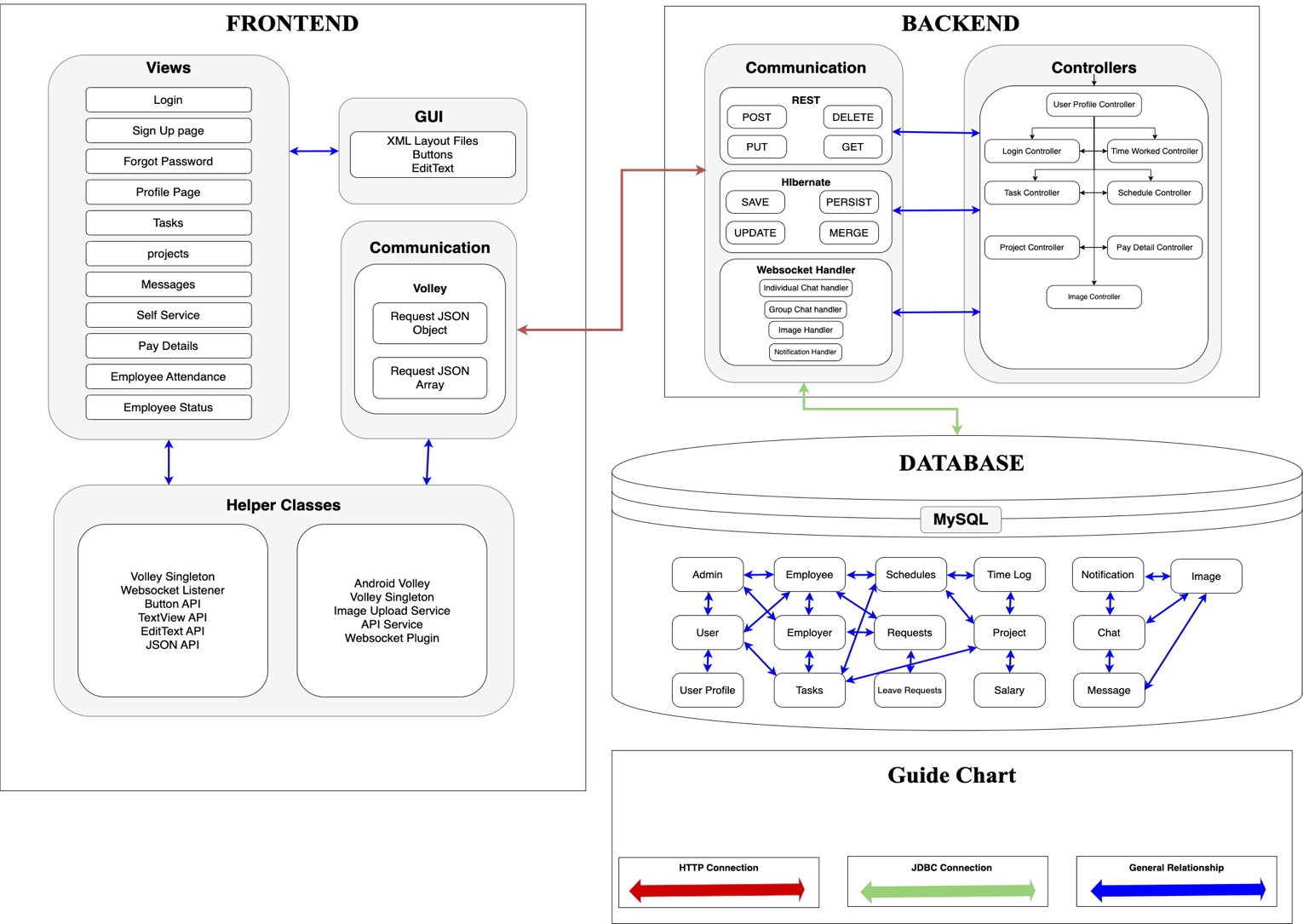
Group 3\_roy\_7

Ethan Roepke: 25% contribution

Ali Mannan: 25% contribution

Mubassir Sudipto: 25% contribution

Thien Nguyen: 25% contribution



**Frontend**

*Login (Admin, Employer, Employee)*

* When login with these fields given below, will be directed to respected page depending on user type. The user plugs in their username and password, if credentials are invalid will not allow login. If credentials are valid then will be logged in. It uses GET to verify credentials.
  + EditText: username
  + EditText: password
  + Button: Sign In

*Sign Up (All users)*

* If user has no account, they will go to sign up page and fill in respected fields. This will create a new account. It uses a POST to send data to database and save it
  + EditText: Full Name
  + EditText: Email
  + EditText: username
  + EditText: password
  + EditText: re-enter password
  + Button: create account

*Self Service (All users)*

* This page displays users schedule for shifts and projects scheduled. All users view schedules but Admin creates schedules for Employers and Employers create schedules for Employee. We use a GET to display schedules and a POST to create schedules.
  + TextView: eventTime
  + TextView: eventName
  + calendarView: scheduleCalendar

*Projects/Tasks (All users)*

* Admin views all projects, employer views there projects and tasks for Employees and Employees just view their own tasks. This pages shows all projects and tasks created, with due date, name and description, as well priority level of project. Task has an interactive button letting us know if has been assigned, in progress, or completed. We use a GET to view all projects and tasks and as well a POST so we can create projects and tasks.

**Backend**

The backend is the core of the system, facilitating data processing, business logic implementation, and communication between the frontend and the database. It is divided into three main components: Communication, Controllers, and WebSocket Handler.

1. *Communication:* The communication layer ensures smooth interaction between the frontend, backend, and database by employing multiple protocols and frameworks:

a. *REST API:* REST (Representational State Transfer) provides a stateless architecture for HTTP communication:

* *POST:* Adds new resources to the database. For example, creating a new task, user profile, or schedule.
* *GET:* Retrieves data from the database. This is used for fetching user details, project information, or employee status.
* *PUT:* Updates existing resources in the database. For instance, modifying an employee's schedule or task progress.
* *DELETE:* Deletes specific resources using unique identifiers, such as removing a user profile or canceling a request.

b. *Hibernate Framework:* Hibernate facilitates the mapping between object-oriented programming constructs and the relational database schema. Key operations include:

* *SAVE:* Persists a new object in the database.
* *PERSIST:* Ensures an object is saved with transaction management.
* *MERGE:* Updates detached objects.
* *UPDATE:* Synchronizes the current state of objects with the database.

c. *WebSocket Handler:* WebSockets provide real-time, two-way communication, critical for high-performance features:

* *Individual Chat Handler:* Manages private, user-to-user messages.
* *Group Chat Handler:* Coordinates group chats for collaboration.
* *Image Handler:* Handles image upload, retrieval, and storage processes for user avatars or project-related files.
* *Notification Handler:* Sends real-time notifications to users, such as task reminders or leave request approvals.

2. *Controllers*

Controllers serve as the intermediaries between the backend logic and database. Each controller is responsible for a specific aspect of the system:

* + *User Profile Controller:* Manages CRUD operations for user profiles, ensuring each user has accurate details like name, contact, and role within the system.
  + *Login Controller:* Facilitates user authentication and session management, handling features such as password validation and multi-factor authentication.
  + *Task Controller:* Manages task assignments, updates, and deletions. Ensures task-related data, such as deadlines and priorities, are consistent.
  + *Project Controller:* Handles the creation, update, and retrieval of project data. Links projects with associated tasks and team members.
  + *Schedule Controller:* Manages work schedules for employees, including time slot allocations and adjustments.
  + *Time Worked Controller:* Tracks the hours worked by employees. Useful for payroll and productivity assessments.
  + *Pay Detail Controller:* Ensures proper handling of payroll information, calculating salaries based on hours worked and other factors.
  + *Image Controller:* Handles all aspects of image processing, including uploading, storing references, and linking images to users or tasks.

