Design Document for ISU PULSE
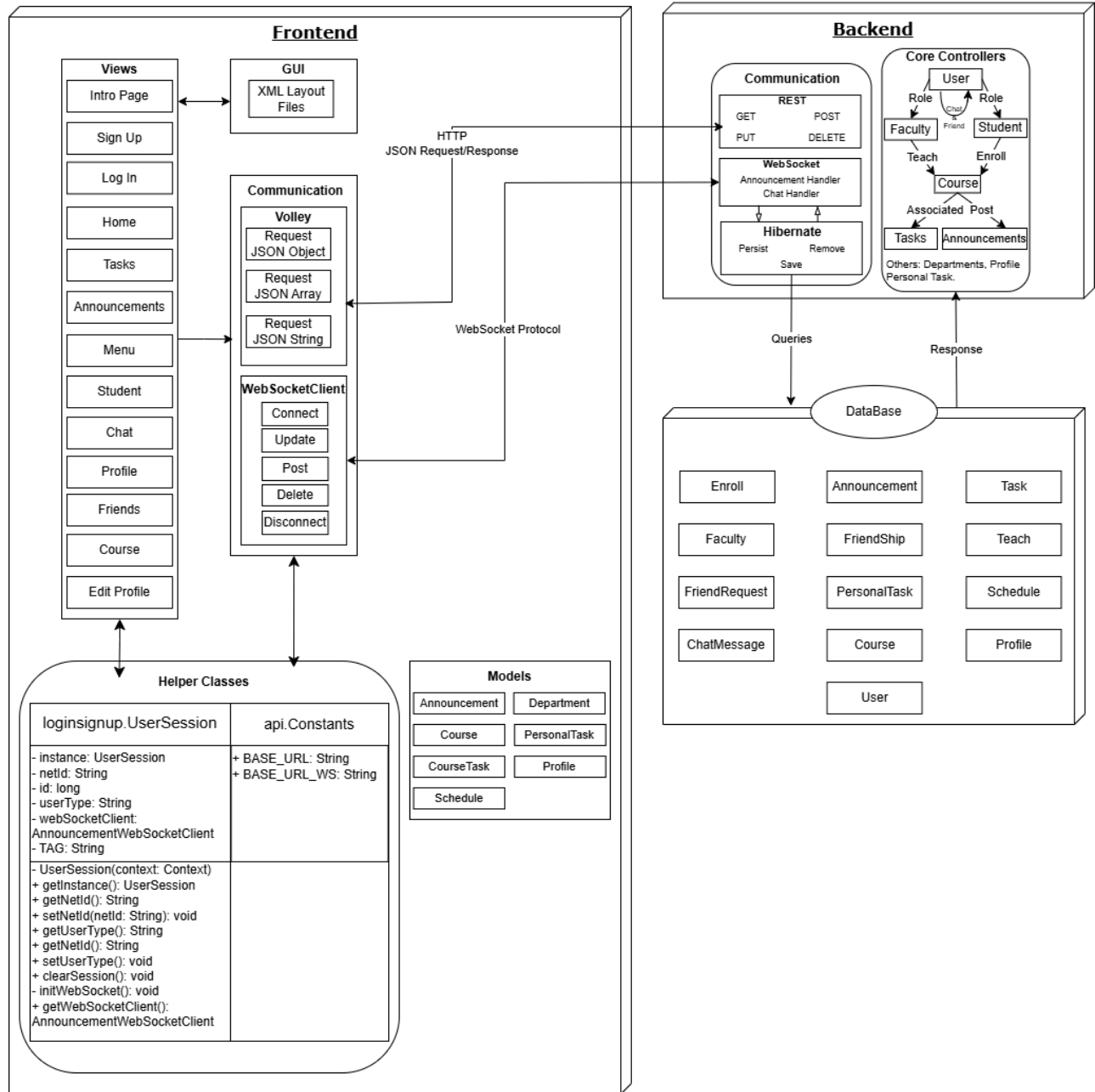
Group 3_roy_2
Nhat Le: 25% contribution
Bach Nguyen: 25% contribution
Autrin Hakimi: 25% contribution
Minh Nguyen: 25% contribution

Frontend

Views
- Intro Page
- Sign Up
- Log In
- Home
- Tasks
- Announcements
- Menu
- Student
- Chat
- Profile
- Friends
- Course
- Edit Profile

GUI
- XML Layout Files

Communication

Volley
- Request JSON Object
- Request JSON Array
- Request JSON String

WebSocketClient
- Connect
- Update
- Post
- Delete
- Disconnect

Helper Classes

loginsignup.UserSession	api.Constants
- instance: UserSession - netId: String - id: long - userType: String - webSocketClient: AnnouncementWebSocketClient - TAG: String	+ BASE_URL: String + BASE_URL_WS: String
- UserSession(context: Context) + getInstance(): UserSession + getNetId(): String + setNetId(netId: String): void + getUserType(): String + getNetId(): String + setUserType(): void + clearSession(): void - initWebSocket(): void + getWebSocketClient(): AnnouncementWebSocketClient	

Models
- Announcement
- Department
- Course
- PersonalTask
- CourseTask
- Profile
- Schedule

HTTP JSON Request/Response

WebSocket Protocol

Backend

Communication

REST
GET   POST

PUT   DELETE

WebSocket
Announcement Handler
Chat Handler

Hibernate
Persist   Remove

Save

Core Controllers

User

Role   Role

Faculty   Chat   Student

Friend

Teach   Enroll

Course

Associated Post

Tasks   Announcements

Others: Departments, Profile Personal Task.

Queries

Response

DataBase

- Enroll
- Announcement
- Task
- Faculty
- FriendShip
- Teach
- FriendRequest
- PersonalTask
- Schedule
- ChatMessage
- Course
- Profile
- User

Block Diagram Description

Frontend

- Login & Signup (User can be specified as Teacher or Student): The user signs up, which creates a REST POST request and adds it to the request queue, or signs in, which sends a GET request to the backend (through the Volley library). The passwords are hashed. Once logged in, even quitting the app or turning off the emulator will not log out the user.
- Home/Dashboard: It will be opened after signing in or signing up and can be accessed through the menu, too. Fetches the Tasks, personal or course assignments, and announcements. Shows a calendar. By clicking on the "ADD TASK" button, students can create a task. They can also edit a task by clicking on the "EDIT TASK" button. There is also a "Delete" button for tasks. Tasks, announcements, and calendar have their own special adaptors.
- Profile: The user can click on Profile through the menu and see their name, profile image, bio, LinkedIn, and any external URL. There is also a log-out button.
- Courses: Users can view all courses they are taking in the profile page (Course name, course description, time, recurring pattern, credits).
- Announcements: The teacher can send announcements by going to the course page and clicking on "Post Announcement." The announcements are handled by WebSockets.
- Edit Profile: User can edit their profile, such as passwords, bio, external and LinkedIn URLs. They first need to validate their netID and password.
- Friends: User can view their friend profile, friend list, friend requests, send requests, and friend suggestions.
- Chat: Two users can chat with each other by WebSocket, and the application can also fetch all message history between two users.

Backend

Communication:

The backend uses REST controllers to communicate with the database through Hibernate.

1. Post: Register a new user (student/ faculty).
2. Get: Fetch a course faculties teaching, students taking, fetch user profile, list of enrollments.
3. Put: Update the user's profile, account information, and task information.
4. Delete: Delete user accounts.

Along with that, we also use WebSocket protocol with custom handlers.

1. Chat handler: The GET handler fetches all relevant chat history between 2 users, whereas the POST handler establishes a real-time web socket between 2 users, facilitating a two-way real-time chat room between them.
2. Announcement handler: Post Handler, Update Handler, Delete Handler, which all broadcast real-time updates so that students can instantly see the changes on their ends.

Controllers:

Our database stores user data (email, hashed password, names, etc.), friendships, tasks, schedules, courses, and announcements. Users can be students or faculty, each with a profile (one-to-one). Students are enrolled - one-to-many - in a schedule (a section of a course with recurring times), can view course tasks (many-to-one), manage personal tasks (one-to-many), send/receive friend requests (many-to-many), chat with others, and receive course announcements. Faculty can view their schedules and manage announcements (post, edit, delete) for specific schedules (one-to-many).

friend_ship
- 🔑 id BIGINT(20)
- ◇ user1_id BIGINT(20)
- ◇ user2_id BIGINT(20)
- Indexes ▶

user
- 🔑 id BIGINT(20)
- ◇ email VARCHAR(255)
- ◇ first_name VARCHAR(255)
- ◇ hashed_password VARCHAR(255)
- ◇ last_name VARCHAR(255)
- ◇ net_id VARCHAR(255)
- ◇ profile_picture_url VARCHAR(25...
- ◇ user_type ENUM(...)
- Indexes ▶

friend_requ...
- 🔑 id BIGINT(20)
- ◇ status ENUM(...)
- ◇ receiver_id BIGINT(20)
- ◇ sender_id BIGINT(20)
- Indexes ▶

personal_task
- 🔑 id BIGINT(20)
- ◇ description VARCHAR(25...
- ◇ due_date DATETIME(6)
- ◇ title VARCHAR(255)
- ◇ user_id BIGINT(20)
- Indexes ▶

chat_message
- 🔑 id BIGINT(20)
- ◇ content TEXT
- ◇ timestamp DATETIME(...
- ◆ recipient_id BIGINT(20)
- ◆ sender_id BIGINT(20)
- Indexes ▶

enroll
- 🔑 id BIGINT(20)
- ◇ schedule_id BIGINT(2...
- ◇ student_id BIGINT(20)
- Indexes ▶

profile
- 🔑 id BIGINT(20)
- ◇ description VARCHAR(255)
- ◇ external_url VARCHAR(25...
- ◇ linkedin_url VARCHAR(255)
- ◇ user_id BIGINT(20)
- Indexes ▶

task
- 🔑 id BIGINT(20)
- ◇ description TEXT
- ◇ due_date DATETIME(6)
- ◇ task_type ENUM(...)
- ◇ title VARCHAR(255)
- ◇ schedule_id BIGINT(2...
- Indexes ▶

announcement
- 🔑 id BIGINT(20)
- ◇ content TEXT
- ◇ timestamp DATETIME(...
- ◆ faculty_id BIGINT(20)
- ◆ schedule_id BIGINT(20)
- Indexes ▶

schedule
- 🔑 id BIGINT(20)
- ◇ end_time TIME
- ◇ recurring_pattern VARCHAR(255)
- ◇ section VARCHAR(255)
- ◇ start_time TIME
- ◇ course_id BIGINT(20)
- Indexes ▶

course
- 🔑 id BIGINT(20)
- ◇ code VARCHAR(255)
- ◇ credits INT(11)
- ◇ description TEXT
- ◇ title VARCHAR(255)
- ◇ department_id BIGINT(2...
- Indexes ▶

teach
- 🔑 id BIGINT(20)
- ◇ faculty_id BIGINT(20)
- ◇ schedule_id BIGINT(2...
- Indexes ▶

faculty
- 🔑 id BIGINT(20)
- ◇ title VARCHAR(255)
- ◇ department_id BIGINT(2...
- ◇ user_id BIGINT(20)
- Indexes ▶

department
- 🔑 id BIGINT(20)
- ◇ location VARCHAR(25...
- ◇ name VARCHAR(255)
- Indexes ▶