Design Document for CyStudy

Group MK1_1

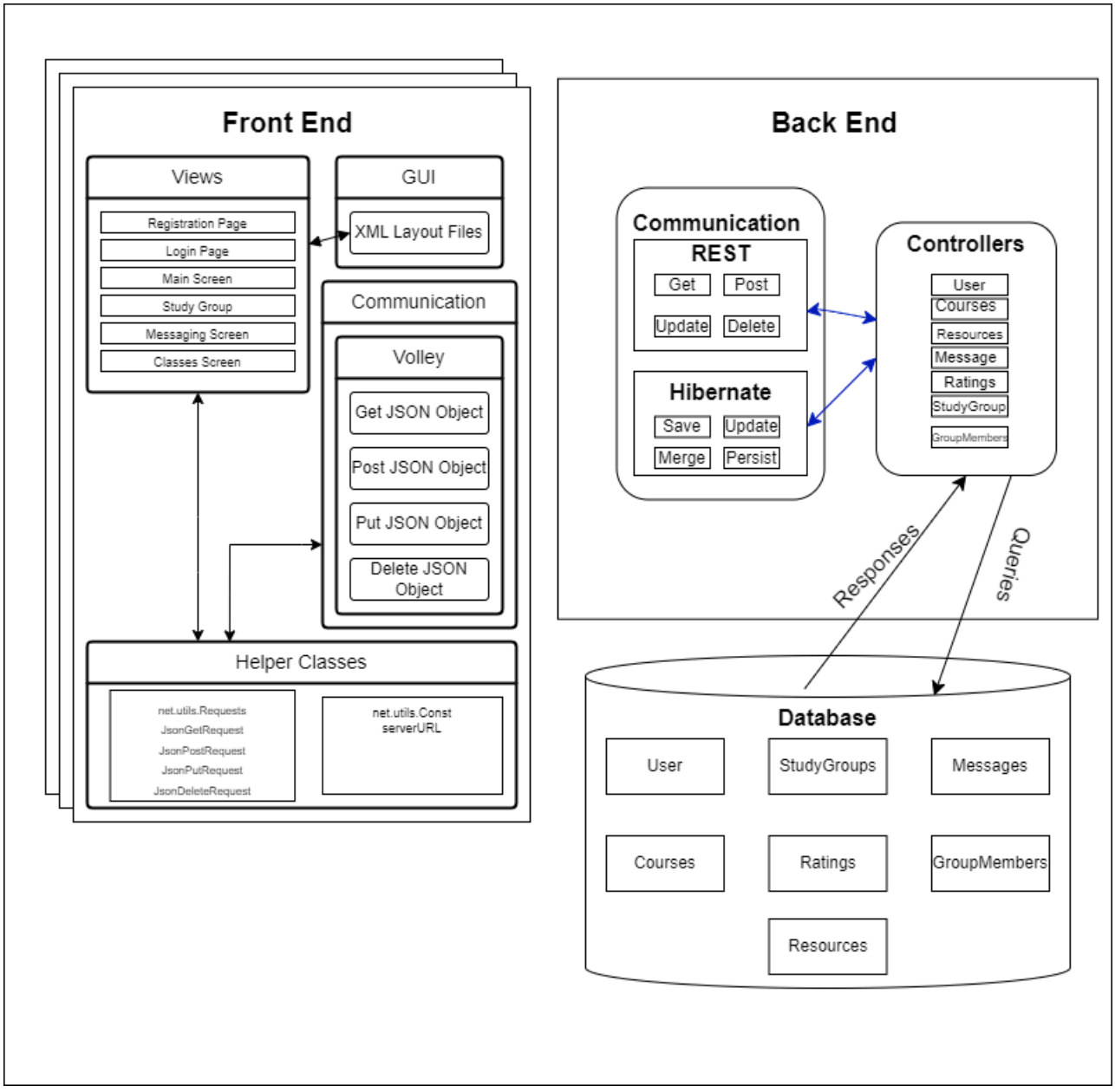
Gavin McClure: 25 % contribution

Rahul Sudev: 25% contribution

Tanish Visanagiri: 25% contribution

Saeshu Karthika Murugan Indumathi: 25% contribution

Block Diagram



Frontend (currently implemented)

CreateAccount (user)

- Creates an account that generates the following elements
    - EditText: username
    - EditText: user
    - EditText: password
    - EditText: email
- Upon clicking the signupButton, the user is redirected to the login screen if the account given the credentials does not exist, and the information is sent to the backend server using POST request

Login (user)

- Create Account generates a page with the following elements
    - EditText: username
    - EditText: password
    - Button: login
- The login screen takes the user's username and password input and upon pressing the login button, generates a GET request to the backend to check if an account exists with such credentials

CreateStudyGroup

- Creates a study group that generates a page with messaging capabilities
    - EditText: groupname
    - Button: creategroupbutton
- The study group screen takes the user's group name from the text box and upon pressing the create study group button, generates a POST request to the backend to add a study group to the database.

SendMessage

- The messaging screen generates a page with the following elements
    - EditText: sendtext
    - Button: sendtextbutton
- The messaging screen takes the user's message input from the text box and upon pressing the send button, generates a POST request to the backend to add the message to the database.

Backend

The backend system is responsible for executing CRUD (Create, Read, Update, Delete) operations on the database by leveraging mappings between different tables. Below is the breakdown of each operation:

POST: This operation involves creating a new row in the respective table within the database. It typically involves inserting data into the appropriate table.

GET: This operation is used to request information from the database, often utilizing an identifier such as an ID to retrieve specific data.

UPDATE: This operation involves updating existing information in the database, often provided in the form of a JSON object containing the updated data.

DELETE: Using Java Persistence API (JPA), this operation deletes information from the database, typically based on specified criteria.

Entities:

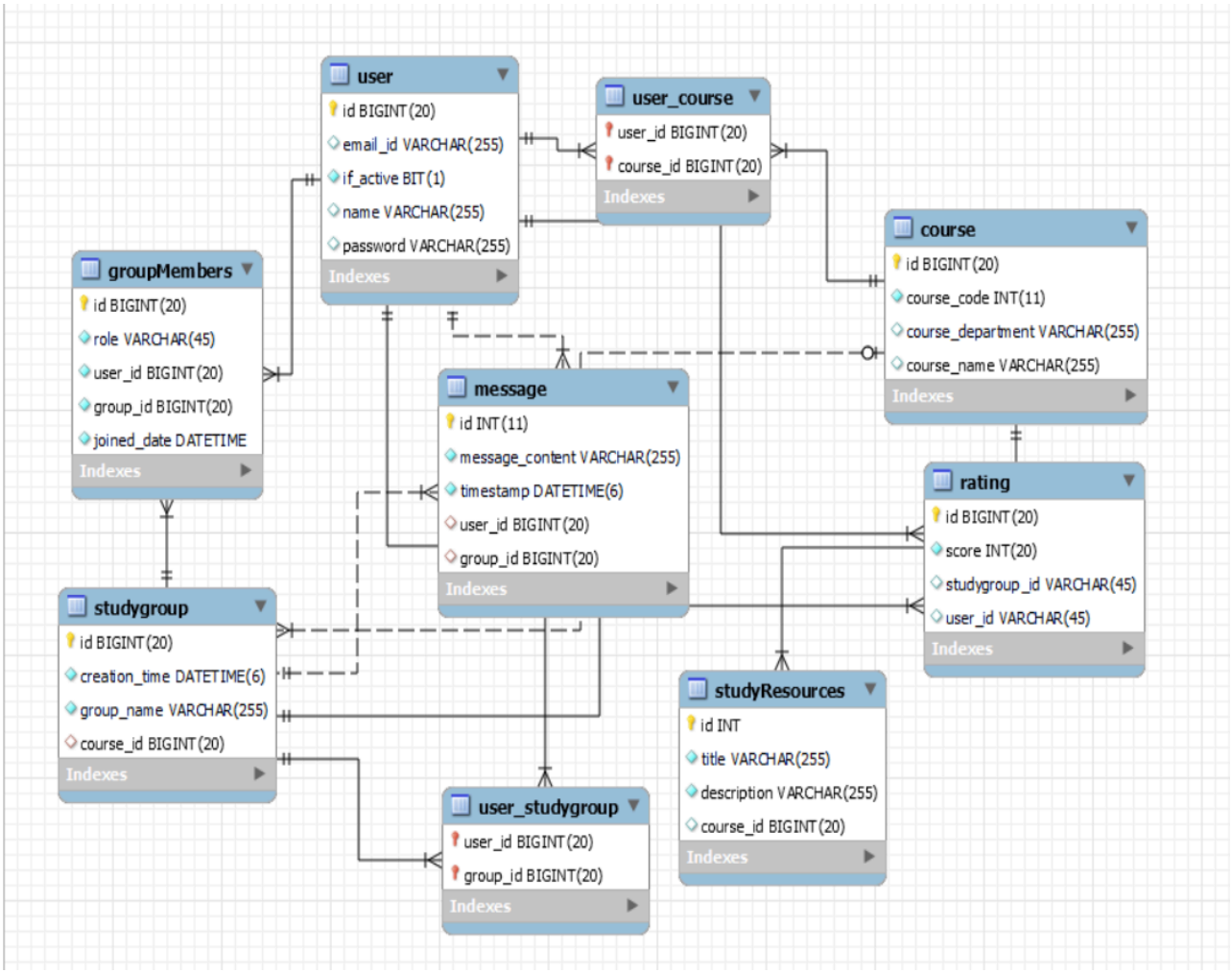
1. **User:**
    - Mapped to many-to-many relationships with Course and StudyGroups.
    - Mapped to one-to-many relationship with Message.
2. **Course:**
    - Has a many-to-many relationship with User.
    - Has a one-to-many relationship with Groups.
3. **Message:**
    - Has a many-to-one relationship with StudyGroup.
    - Has a many-to-one relationship with User.
4. **Groups:**
    - Has a many-to-one relationship with Course.
    - Has a one-to-many relationship with Message.
    - Has a many-to-many relationship with StudyGroup.
5. **GroupMembers:**
    - Has a one-to-one relationship with User.
    - Has a one-to-many relationship with Groups.
6. **StudyResources:**
    - Has a many-to-one relationship with Course.
7. **Rating**

- Has one to one relationship with Group.

Controllers:

1. **UserController:** Manages user-related operations such as authentication, registration, and profile management.
2. **CourseController:** Handles actions related to courses which will automatically be uploaded including creation, retrieval, updating, and deletion.
3. **MessageController:** Manages messages and WebSocket connections for real-time messaging functionality for each group.
4. **GroupController:** Manages group-related operations such as creation, retrieval, updating, and deletion.
5. **GroupMembersController:** Handles operations related to group memberships and user associations with groups.
6. **StudyResourcesController:** Manages study resource operations including creation, retrieval, updating, and deletion.
7. **RatingController:** Handles operations related to ratings for various entities and ensures data integrity.

This structure outlines the entities, their relationships, and the controllers responsible for handling operations related to each entity.


user
id BIGINT(20)
email_id VARCHAR(255)
if_active BIT(1)
name VARCHAR(255)
password VARCHAR(255)
Indexes
user_course
user_id BIGINT(20)
course_id BIGINT(20)
Indexes
course
id BIGINT(20)
course_code INT(11)
course_department VARCHAR(255)
course_name VARCHAR(255)
Indexes
groupMembers
id BIGINT(20)
role VARCHAR(45)
user_id BIGINT(20)
group_id BIGINT(20)
joined_date DATETIME
Indexes
message
id INT(11)
message_content VARCHAR(255)
timestamp DATETIME(6)
user_id BIGINT(20)
group_id BIGINT(20)
Indexes
rating
id BIGINT(20)
score INT(20)
studygroup_id VARCHAR(45)
user_id VARCHAR(45)
Indexes
studygroup
id BIGINT(20)
creation_time DATETIME(6)
group_name VARCHAR(255)
course_id BIGINT(20)
Indexes
studyResources
id INT
title VARCHAR(255)
description VARCHAR(255)
course_id BIGINT(20)
Indexes
user_studygroup
user_id BIGINT(20)
group_id BIGINT(20)
Indexes