Design Document for Cymind

Group 4_Shrestha_4

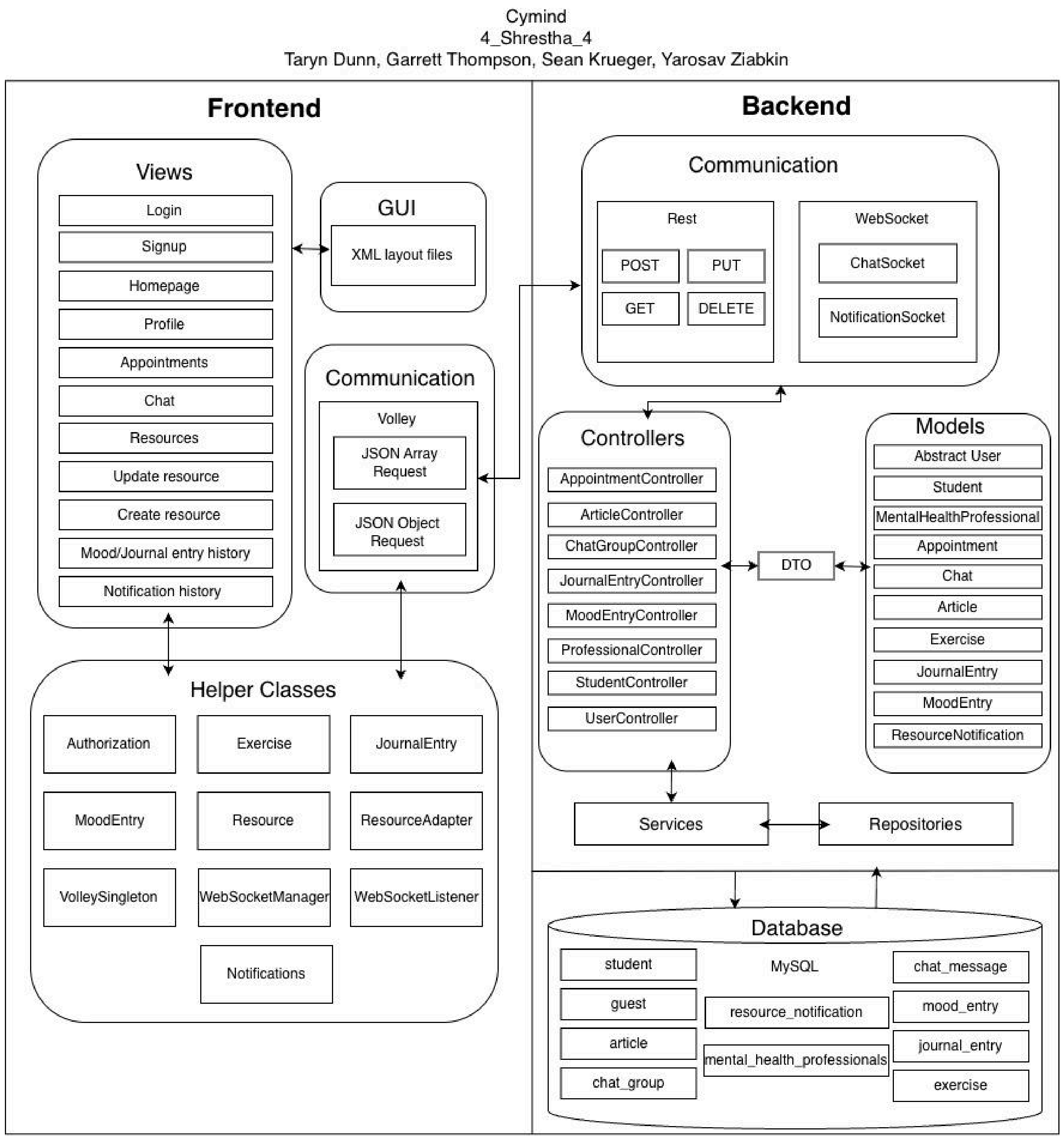
Taryn Dunn: 25% contribution

Garrett Thompson: 25% contribution

Sean Krueger: 25% contribution

Yarosav Ziabkin: 25% contribution

Block Diagram



Block Diagram Description

Actors:

Our application has 3 different types of users: students, mental health professionals, and guests. Actors use our Android application, where they interact with different views, which provide input, output, and UI functionality. Examples of input types are: registration/login information, students' mood/journal entries, typed messages in chat, and appointment time selection. Examples of output types are: appointment confirmation, requested mental health article/exercise, and new resource/appointment notifications. Depending on the user type, most screens offer different functionality. For instance, all students, mental health professionals, and guest users have a home page, however its content is different for every type of user. Key components of the client application are:

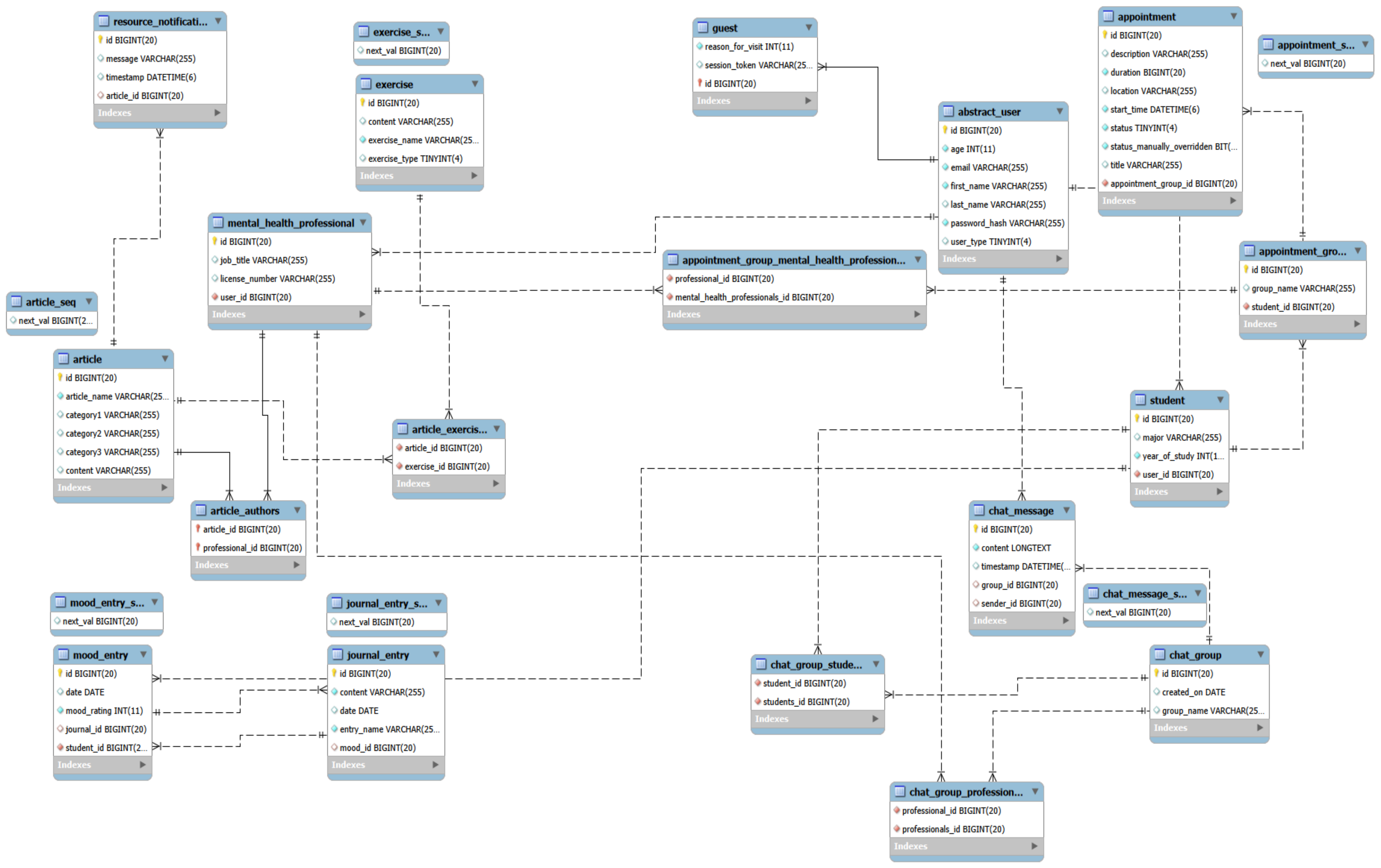
- **UI Layer:** Uses Android Views to handle input and display data.
- **Logic Layer:** Implements the MVP pattern. Views pass actions to Presenters.
- **Networking:** The Presenter delegates API calls to a ServerRequest handler (using Volley).
- **Callback Loop:** The Presenter implements a VolleyListener. When the server responds, the listener triggers, and the Presenter updates the View with the new data.

Server:

The backend is built using Spring Boot and deployed via Apache Tomcat, comprising four primary components. The first one is a REST Controller, which serves as the entry point to our server for the frontend. This layer contains many URL-mapped endpoints that provide data transfer between the client and the server. All the endpoints are either GET, POST, PUT, or DELETE HTTP methods. Additionally, there are two WebSocket endpoints that provide full-duplex communication for live chat and live notifications. Next is a Service layer, which contains all the business logic, and lies between the database and controller layers, making our architecture more modular. Then we have a Model layer, which contains the "blueprints" for all the entities. DTOs are used to transfer relevant Models data through the controller. Models are managed by the Repositories layer, which are links between the database and the server, and are responsible for data persistence and communication. This is done by utilizing the Hibernate framework and Java Persistence Architecture.

Database:

We use MariaDB, which is an open source fork of MySQL. The database serves as the single source of truth for the application, storing data such as user credentials, profiles, articles, mood entries, chats, and notifications. MariaDB is a relational database that allows us to define structured relationships between these different data entities, so our tables have One-to-One, One-to-Many, Many-to-One, and Many-to-Many relationships, making the app's features and tables comprehensive.


resource_notificati...
id BIGINT(20)
message VARCHAR(255)
timestamp DATETIME(6)
article_id BIGINT(20)
Indexes
exercise_s...
next_val BIGINT(20)
exercise
id BIGINT(20)
content VARCHAR(255)
exercise_name VARCHAR(25...
exercise_type TINYINT(4)
Indexes
guest
reason_for_visit INT(11)
session_token VARCHAR(25...
id BIGINT(20)
Indexes
appointment
id BIGINT(20)
description VARCHAR(255)
duration BIGINT(20)
location VARCHAR(255)
start_time DATETIME(6)
status TINYINT(4)
status_manually_overridden BIT(...
title VARCHAR(255)
appointment_group_id BIGINT(20)
Indexes
appointment_s...
next_val BIGINT(20)
abstract_user
id BIGINT(20)
age INT(11)
email VARCHAR(255)
first_name VARCHAR(255)
last_name VARCHAR(255)
password_hash VARCHAR(255)
user_type TINYINT(4)
Indexes
mental_health_professional
id BIGINT(20)
job_title VARCHAR(255)
license_number VARCHAR(255)
user_id BIGINT(20)
Indexes
appointment_group_mental_health_profession...
professional_id BIGINT(20)
mental_health_professionals_id BIGINT(20)
Indexes
appointment_gro...
id BIGINT(20)
group_name VARCHAR(255)
student_id BIGINT(20)
Indexes
article_seq
next_val BIGINT(2...
article
id BIGINT(20)
article_name VARCHAR(25...
category1 VARCHAR(255)
category2 VARCHAR(255)
category3 VARCHAR(255)
content VARCHAR(255)
Indexes
article_exercis...
article_id BIGINT(20)
exercise_id BIGINT(20)
Indexes
student
id BIGINT(20)
major VARCHAR(255)
year_of_study INT(1...
user_id BIGINT(20)
Indexes
article_authors
article_id BIGINT(20)
professional_id BIGINT(20)
Indexes
chat_message
id BIGINT(20)
content LONGTEXT
timestamp DATETIME(...
group_id BIGINT(20)
sender_id BIGINT(20)
Indexes
mood_entry_s...
next_val BIGINT(20)
journal_entry_s...
next_val BIGINT(20)
chat_message_s...
next_val BIGINT(20)
mood_entry
id BIGINT(20)
date DATE
mood_rating INT(11)
journal_id BIGINT(20)
student_id BIGINT(2...
Indexes
journal_entry
id BIGINT(20)
content VARCHAR(255)
date DATE
entry_name VARCHAR(25...
mood_id BIGINT(20)
Indexes
chat_group_stude...
student_id BIGINT(20)
students_id BIGINT(20)
Indexes
chat_group
id BIGINT(20)
created_on DATE
group_name VARCHAR(25...
Indexes
chat_group_profession...
professional_id BIGINT(20)
professionals_id BIGINT(20)
Indexes