Design Document for CyEvents

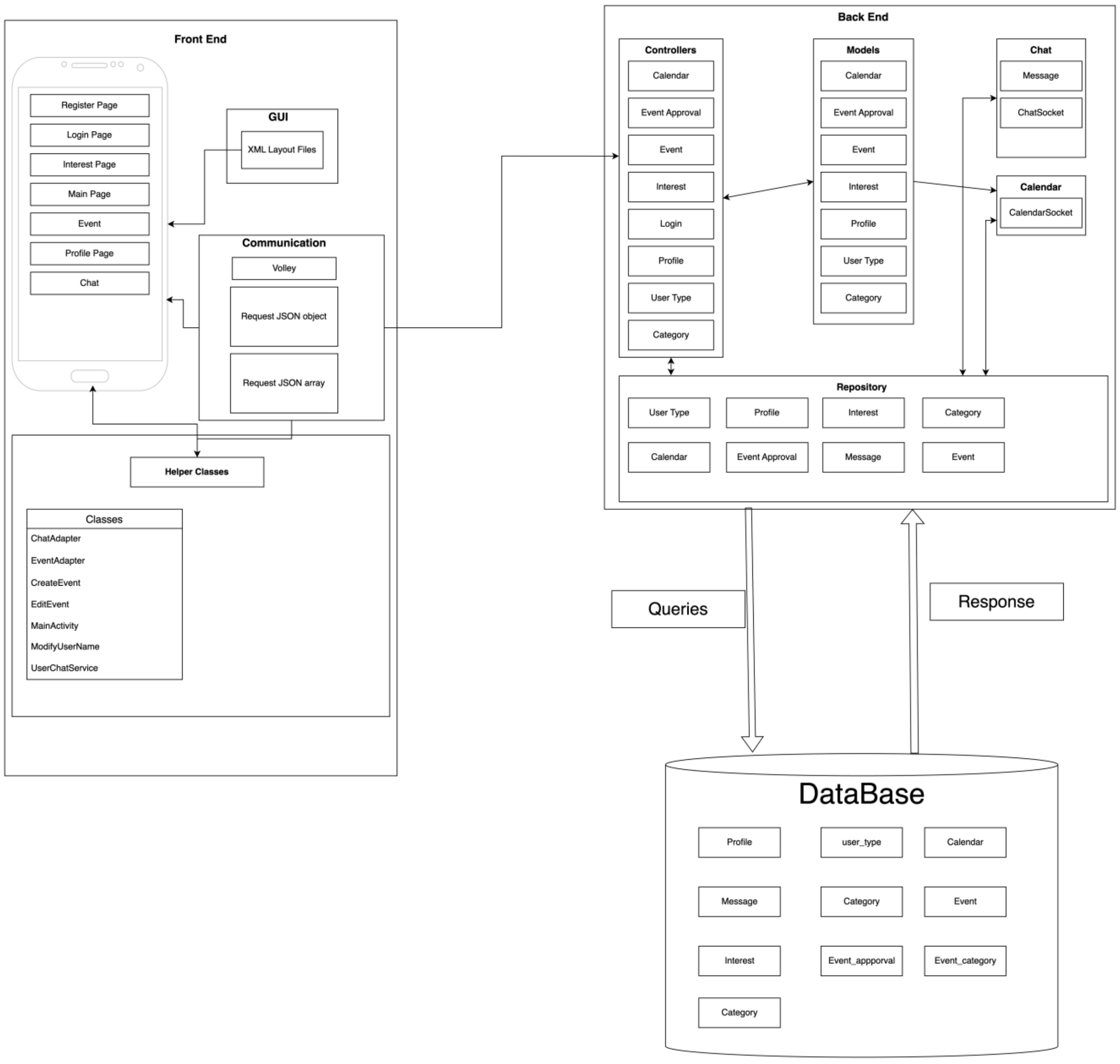
Group <3_Roy_1>

Member1 Name: Kaung Son %1/3 contribution

Member2 Name: Nathan Church %1/3 contribution

Member3 Muhammed Fadel %1/3 contribution

THE BLOCK DIAGRAM PICTURE



Frontend:

Login Page

- **Login Page** generates a page with the following elements:
  - **EditText**: UserID
  - **EditText**: Password
  - **Button**: Login
- Upon clicking the **'Login'** button, the UserID and Password are validated. If valid, the user is taken to the **Main Page**.

Main Page

- **Main Page** serves as the primary navigation hub and includes:
  - **Button**: View Events
  - **Button**: View Profile
  - **Button**: Chat
- This page allows users to access events, their profile, and the chat feature.

Event Page

- **Event Page** generates a page displaying details of selected events and includes:
  - **RecyclerView** for displaying a list of events.
  - **Button**: Register for Event (for individual events)
  - **Button**: Edit Events
  - **Button**: Delete Events
- Loading the page sends a **GET request** to retrieve a list of events. Clicking **'Register for Event'** sends a **POST request** to register the user for that specific event. Clicking **'Edit event'** allows for modification and send a **PUT request**. Similarly, clicking **Delete** sends a **Delete request** to delete the specific event from the list.

Profile Page

- **Profile Page** generates a page with the following elements:
  - **TextView**: Displays User Information
  - **Button**: Modify Username

    - **Button**: Edit Interests
    - **Button**: Logout
    - **Button**: Delete Account
- Clicking **‘Modify Username’** or **‘Edit Interests’** allows the user to update their profile information, sending **PUT requests** to the server with the new data. Clicking **Delete** Account will send a **Delete request** and remove the account from the profile table.
- **Chat Page**
- **Chat Page** generates a chat interface with the following elements:
    - **EditText**: Message Input
    - **Button**: Send
    - **Button**: Reply
    - **Button**: React
    - **Button**: Delete
    - **RecyclerView**: Chat Messages
- The page uses a **ChatAdapter** to display chat messages. Messages are sent and received via a **WebSocket** connection:
    - **Send**: Clicking the **‘Send’** button sends the message input through the WebSocket connection in real time.
    - **Reply**: Clicking **‘Reply’** allows users to respond to a specific message.
    - **React**: Clicking **‘React’** allows users to add an emoji or reaction to a message.
    - **Delete**: Clicking **‘Delete’** removes the selected message from the chat for that user.
- The WebSocket connection maintains a live feed of incoming messages, reactions, and replies, updating the **RecyclerView** as new messages arrive in real time.

GUI

- **XML Layout Files** define the user interface elements for each of these pages, providing the structure and style of the app.

Communication

- **Volley** library is used to handle network requests:

    - **Request JSON Object**: Used for single data entries, such as login or creating an account.
    - **Request JSON Array**: Used for lists of data, such as displaying events or chat history.

Helper Classes

- **ChatAdapter**: Manages displaying chat messages in the chat interface.
- **EventAdapter**: Handles displaying event details in lists.
- **CreateEvent**: Class for creating a new event in the Event Page.
- **EditEvent**: Class for editing event details.
- **MainActivity**: The main controller that manages navigation between pages.
- **ModifyUserName**: Class for handling username modification on the Profile Page.
- **UserChatService**: Manages sending and receiving messages in the Chat Page.

BackEnd:

Communication:

The backend handles requests from the frontend and updates the database using:

POST: To add new data like users, events, or messages.

GET: To retrieve profiles, events, or calendar entries.

PUT: To update data like profiles or event approvals.

DELETE: To remove users, events, or chat messages.

Controllers:

Controllers manage specific tasks by linking the frontend to the database:

ProfileController: Manages user profiles (view, edit, delete).

EventController: Handles creating, editing, and deleting events. Includes approval tracking.

CalendarController: Links users to events via many-to-many relationships.

ChatSocket: Manages real-time messaging with WebSockets.

EventApprovalController: Allows admins to approve or reject events.

Database:

The database stores all app data, including:

Profile: User details like username and interests.

Event: Event info (name, date, approval).

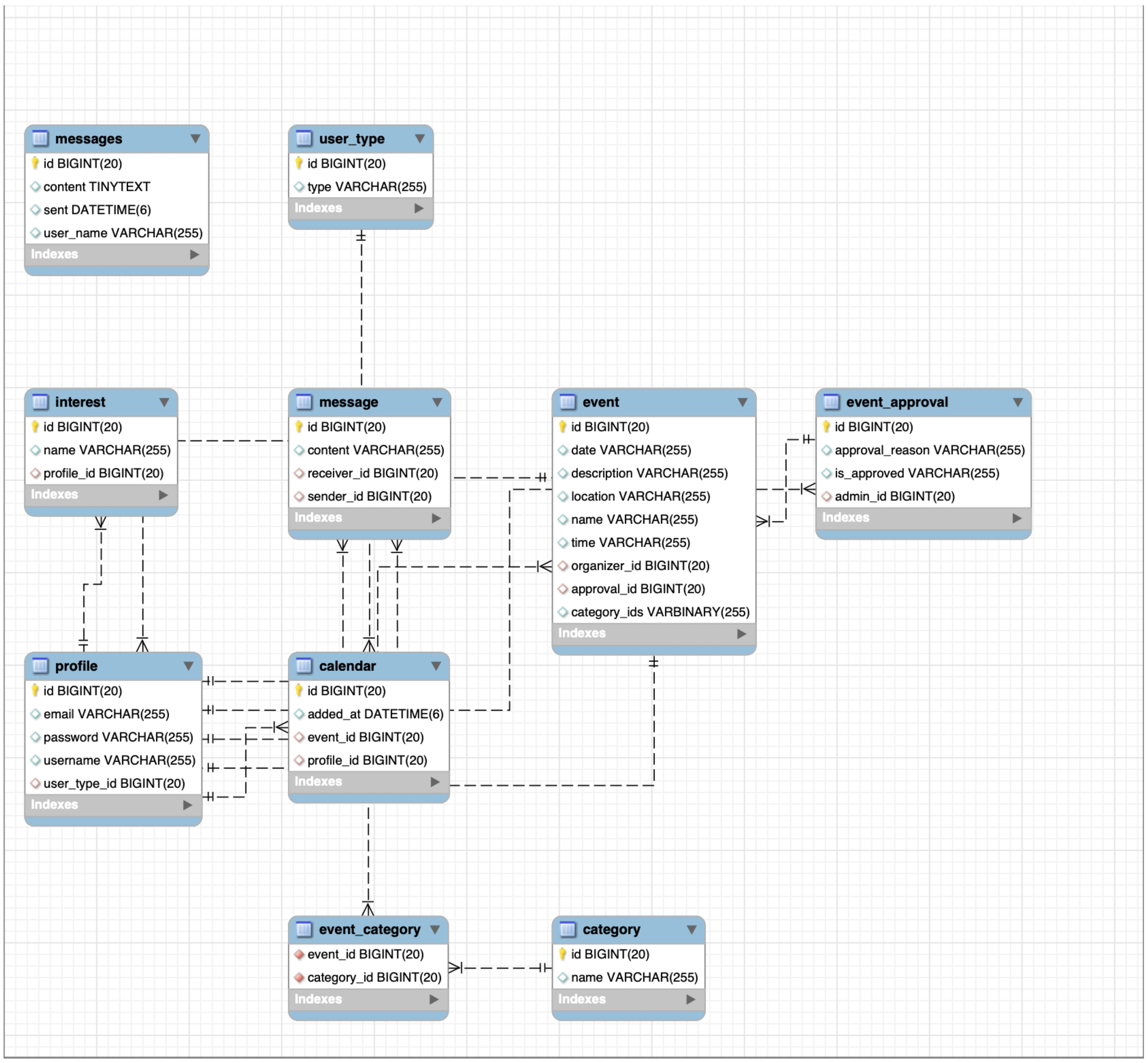
Calendar: Links users and events.

Message: Saves chat history.

EventApproval: Tracks event approvals and reasons.

-

THE TABLE RELATIONSHIPS DIAGRAM


messages
id BIGINT(20)
content TINYTEXT
sent DATETIME(6)
user_name VARCHAR(255)
Indexes
user_type
id BIGINT(20)
type VARCHAR(255)
Indexes
interest
id BIGINT(20)
name VARCHAR(255)
profile_id BIGINT(20)
Indexes
message
id BIGINT(20)
content VARCHAR(255)
receiver_id BIGINT(20)
sender_id BIGINT(20)
Indexes
event
id BIGINT(20)
date VARCHAR(255)
description VARCHAR(255)
location VARCHAR(255)
name VARCHAR(255)
time VARCHAR(255)
organizer_id BIGINT(20)
approval_id BIGINT(20)
category_ids VARBINARY(255)
Indexes
event_approval
id BIGINT(20)
approval_reason VARCHAR(255)
is_approved VARCHAR(255)
admin_id BIGINT(20)
Indexes
profile
id BIGINT(20)
email VARCHAR(255)
password VARCHAR(255)
username VARCHAR(255)
user_type_id BIGINT(20)
Indexes
calendar
id BIGINT(20)
added_at DATETIME(6)
event_id BIGINT(20)
profile_id BIGINT(20)
Indexes
event_category
event_id BIGINT(20)
category_id BIGINT(20)
Indexes
category
id BIGINT(20)
name VARCHAR(255)
Indexes