Design Document for 2v2 Chess App

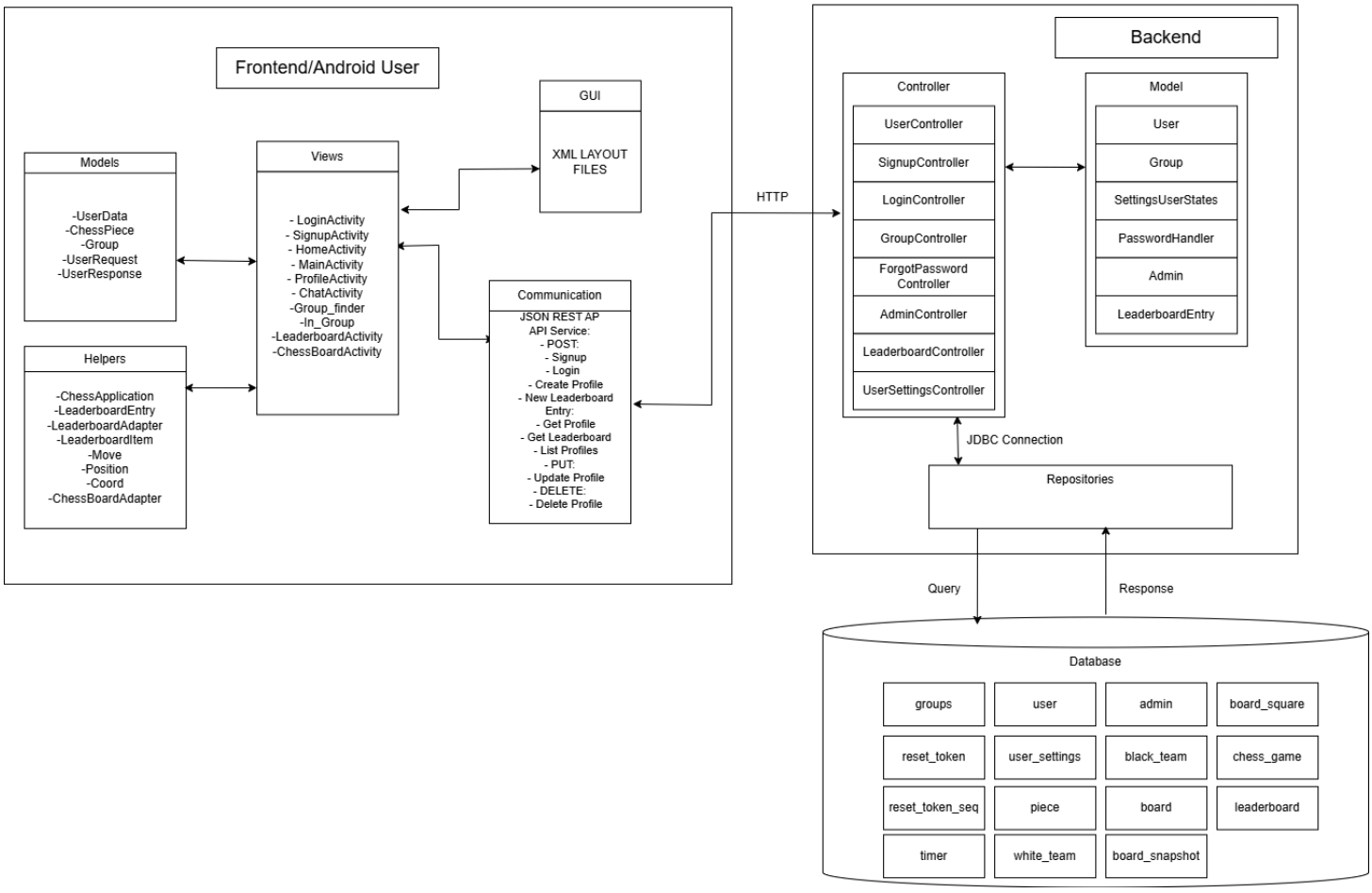
Group 3_Swarna_3

Phillip Giametta: 25% contribution

Ethan Roe: 25% contribution

Noah Busack: 25% contribution

Alec Moore: 25% contribution


Frontend/Android User
GUI
XML LAYOUT FILES
Models
-UserData
-ChessPiece
-Group
-UserRequest
-UserResponse
Views
- LoginActivity
- SignupActivity
- HomeActivity
- MainActivity
- ProfileActivity
- ChatActivity
-Group_finder
-In_Group
-LeaderboardActivity
-ChessBoardActivity
Helpers
-ChessApplication
-LeaderboardEntry
-LeaderboardAdapter
-LeaderboardItem
-Move
-Position
-Coord
-ChessBoardAdapter
Communication
JSON REST AP
API Service:
- POST:
- Signup
- Login
- Create Profile
- New Leaderboard Entry:
- Get Profile
- Get Leaderboard
- List Profiles
- PUT:
- Update Profile
- DELETE:
- Delete Profile
HTTP
Backend
Controller
UserController
SignupController
LoginController
GroupController
ForgotPassword Controller
AdminController
LeaderboardController
UserSettingsController
Model
User
Group
SettingsUserStates
PasswordHandler
Admin
LeaderboardEntry
JDBC Connection
Repositories
Query
Response
Database
groups
user
admin
board_square
reset_token
user_settings
black_team
chess_game
reset_token_seq
piece
board
leaderboard
timer
white_team
board_snapshot

Block Diagram Description

Frontend Functionality

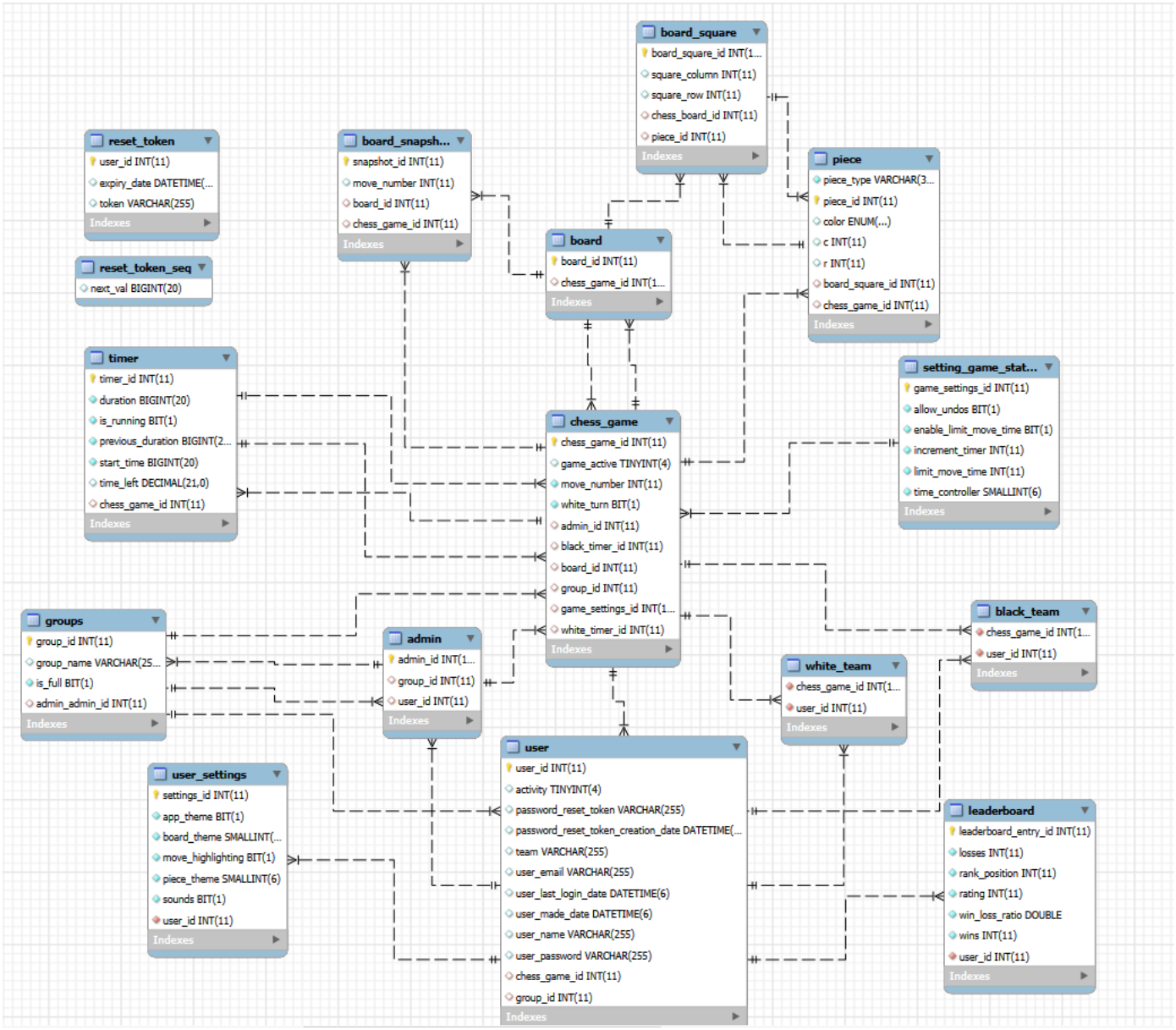
The ChessApp Android front end is organized into activities, adapters, models, and helper classes. At the core, the app uses a JSON REST API with Retrofit to handle network communication, enabling API calls for user signup, login, profile updates, and leaderboard management. This integration with Retrofit allows activities to respond based on server responses, effectively transitioning between screens. The main screens, represented as activities, include LoginActivity, SignupActivity, HomeActivity, ProfileActivity, ChatActivity, GroupFinder, In_Group and LeaderboardActivity, and ChessboardActivity. Each activity handles specific user interactions: LoginActivity validates user credentials and, upon success, navigates to HomeActivity, while ProfileActivity displays and allows modifications to user profiles. The chess functionality resides in ChessBoardActivity, which displays the board and manages moves through WebSockets. ChessBoardAdapter tracks the chess pieces, while ChessPiece defines individual piece attributes. The leaderboard feature is managed by LeaderboardActivity, with LeaderboardAdapter and LeaderboardItem organizing and presenting player rankings based on Elo scores. UserData and ChessPiece store essential data, with UserData holding user-specific information. Helper classes like ChessApplication maintain app-wide settings and information for the chess game, while LeaderboardEntry represents leaderboard data.

Server:

Our server has three main parts: the controllers, which handle HTTP requests between the front end. The Models serve as the leading implementations of classes with constructors, getters, and setters. The repositories which communicate between the Server and the Database. A typical scenario like signing up a new user executes as follows. First, the front end would send an HTTP request to the respective controller, which would then make a call to the respective model to create a new user. Then, the controller sends the data associated with the newly created user to the Database to be stored and saved.

Database:

Our Database contains several tables used to store and organize data. This data includes user information (username, passwords, emails, etc.), groups, information about the game (piece positions on the board, whose turn it is, game settings, etc), and leaderboard information (user rating, position compared to all other users, wins / losses). Our Database uses MySQL.


board_square
board_square_id INT(1...
square_column INT(11)
square_row INT(11)
chess_board_id INT(11)
piece_id INT(11)
Indexes
reset_token
user_id INT(11)
expiry_date DATETIME(...
token VARCHAR(255)
Indexes
board_snapsh...
snapshot_id INT(11)
move_number INT(11)
board_id INT(11)
chess_game_id INT(11)
Indexes
piece
piece_type VARCHAR(3...
piece_id INT(11)
color ENUM(...)
c INT(11)
r INT(11)
board_square_id INT(11)
chess_game_id INT(11)
Indexes
board
board_id INT(11)
chess_game_id INT(1...
Indexes
reset_token_seq
next_val BIGINT(20)
timer
timer_id INT(11)
duration BIGINT(20)
is_running BIT(1)
previous_duration BIGINT(2...
start_time BIGINT(20)
time_left DECIMAL(21,0)
chess_game_id INT(11)
Indexes
setting_game_stat...
game_settings_id INT(11)
allow_undos BIT(1)
enable_limit_move_time BIT(1)
increment_timer INT(11)
limit_move_time INT(11)
time_controller SMALLINT(6)
Indexes
chess_game
chess_game_id INT(11)
game_active TINYINT(4)
move_number INT(11)
white_turn BIT(1)
admin_id INT(11)
black_timer_id INT(11)
board_id INT(11)
group_id INT(11)
game_settings_id INT(1...
white_timer_id INT(11)
Indexes
groups
group_id INT(11)
group_name VARCHAR(25...
is_full BIT(1)
admin_admin_id INT(11)
Indexes
admin
admin_id INT(1...
group_id INT(11)
user_id INT(11)
Indexes
black_team
chess_game_id INT(1...
user_id INT(11)
Indexes
white_team
chess_game_id INT(1...
user_id INT(11)
Indexes
user
user_id INT(11)
activity TINYINT(4)
password_reset_token VARCHAR(255)
password_reset_token_creation_date DATETIME(...
team VARCHAR(255)
user_email VARCHAR(255)
user_last_login_date DATETIME(6)
user_made_date DATETIME(6)
user_name VARCHAR(255)
user_password VARCHAR(255)
chess_game_id INT(11)
group_id INT(11)
Indexes
user_settings
settings_id INT(11)
app_theme BIT(1)
board_theme SMALLINT(...
move_highlighting BIT(1)
piece_theme SMALLINT(6)
sounds BIT(1)
user_id INT(11)
Indexes
leaderboard
leaderboard_entry_id INT(11)
losses INT(11)
rank_position INT(11)
rating INT(11)
win_loss_ratio DOUBLE
wins INT(11)
user_id INT(11)
Indexes