# Design Document for CyGo

**3_swarna_4**

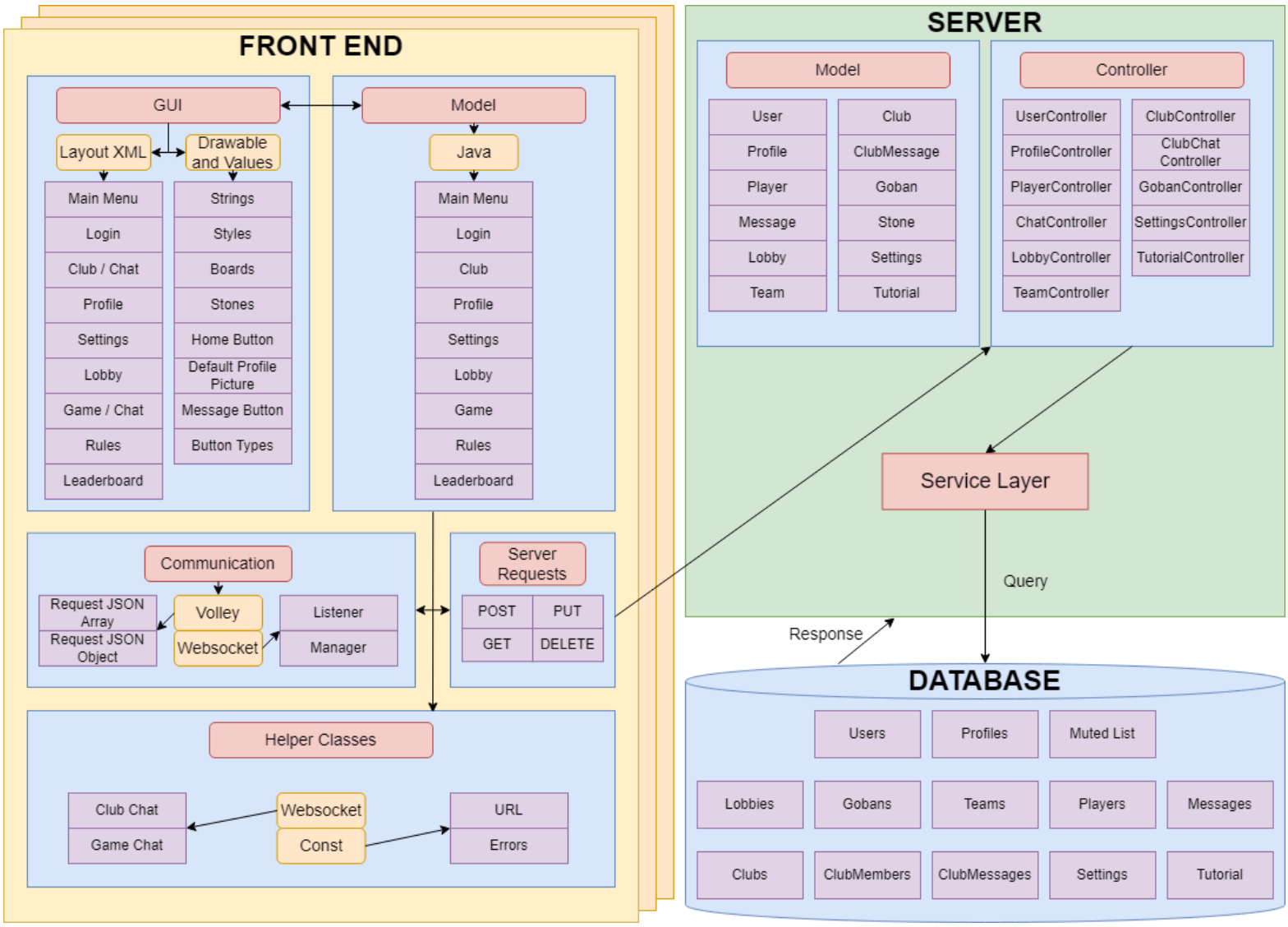
Octavio Munoz:  25%

Eden Basnet: 25%

Devon Keuning: 25%

Matthew Hill: 25%

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE!



Use this third page to describe complex parts of your design.

Complex Tasks include Pre-game, Main-Game, and Club.
**Main Game:**
2-v-2 turn-based game with black and white stones where two players are one set of stones. Players take turns placing stones on the intersection. Interacts with the Goban controller to keep track of stone placements. The main game also has a game chat that players can use to communicate with each other during the gameplay.
**Pre-game Screen**

- **Pre-game Screen** generates a page with the following elements:
    - **Game Configuration Options:** Allows selection of specific game rules and settings.
    - **Ready Status Indicator:** A status near the bottom of the screen indicating player readiness and rule confirmation before starting the game.
    - **Player Icon Display:** Shows player icons or default icons when players join the pre-game lobby, replacing a placeholder (e.g., question mark) with each new participant.
- **Purpose:** Acts as an intermediate screen where players configure game settings, invite friends, or prepare for random matchmaking.
- **Purpose:** Offers a personalized overview of the player's stats, serving as a navigation hub with access to profile details and the home screen.

### Club Screen

- **Club Screen** generates a page with the following elements:
    - **Button:** Club Chat - Takes the user to the club chat area.
    - **Chat Interface:** Contains an EditText for typing messages and a Send Button to send messages.
    - **Conversation Display:** A dynamic area (e.g., a chat window) that shows the ongoing conversation between club members in real-time.
- **Functionality:**

    - **Real-Time Chat:** Allows users to send and receive messages within their club, facilitating communication and interaction among club members.
    - **Message Handling:** Each message sent by a user is added to the chat display and sent to the server to be shared with all members of the club.
- **Purpose:** The **Club Screen** enhances collaboration within the app, allowing users to discuss strategy, plan matches, or just socialize with fellow club members.

**Backend**

*Communication*

The backend uses mappings to manage and interact with the game-related data in the database through specific HTTP methods. These methods ensure proper data flow between the frontend and the backend. The mappings include:

- **Post**: This method is used to add new entities to the database. For example, when a new player joins the game or a new team is formed, a POST request will send this data for insertion.
- **Get**: This method retrieves specific data from the database, often using unique identifiers to query for details such as user profiles, game settings, or team information.
- **Put**: The PUT method is used to update an existing entity. For example, if a player updates their profile settings or a team leader changes the team's structure, PUT requests update these details in the database.
- **Delete**: This method deletes a specific entity from the database based on a provided identifier. It can be used to remove users, teams, or any other entity when no longer needed.

*Controllers*

Controllers are responsible for defining the mappings between the frontend and backend, ensuring that data can be sent, retrieved, updated, or deleted as necessary.

- **UserController**: Manages all user-related operations, including creating and managing user accounts. It links users to their respective profiles.
- **PlayerController**: This controller handles operations related to the players, such as their status, settings, and profile updates, so it mostly serves as a way to temporarily store game data.
- **TeamController**: Handles the creation and management of teams. It includes functionality to assign players to teams, manage team settings, and track team performance.
- **ClubController**: Manages the creation of clubs and the relationships between players within a club. Clubs have internal messaging, and friendly battle systems.
- **GobanController**: This controller governs the game board, including setting up the Go board, handling stoneplacements, and tracking the state of the game.
- **LobbyController**: Handles the lobby area where players can join games, set up matches, and view their status in real time.
- **SettingsController**: Provides functionality for players to modify their personal settings, such as game preferences or account settings.
- **TutorialController**: Provides instructions and guides for new players, helping them understand how to play the game.
- **ChatController**: Manages chat features, enabling real-time communication between players.
- **ClubChatController**: A specialized controller for managing chats within clubs, allowing for team or club-wide discussions.
- **TheProfileController**: Handles user profile-related actions, such as updating player data, viewing player and serves as a helper class for settings to modify profile attributes.

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkber