Design Document for Cymon

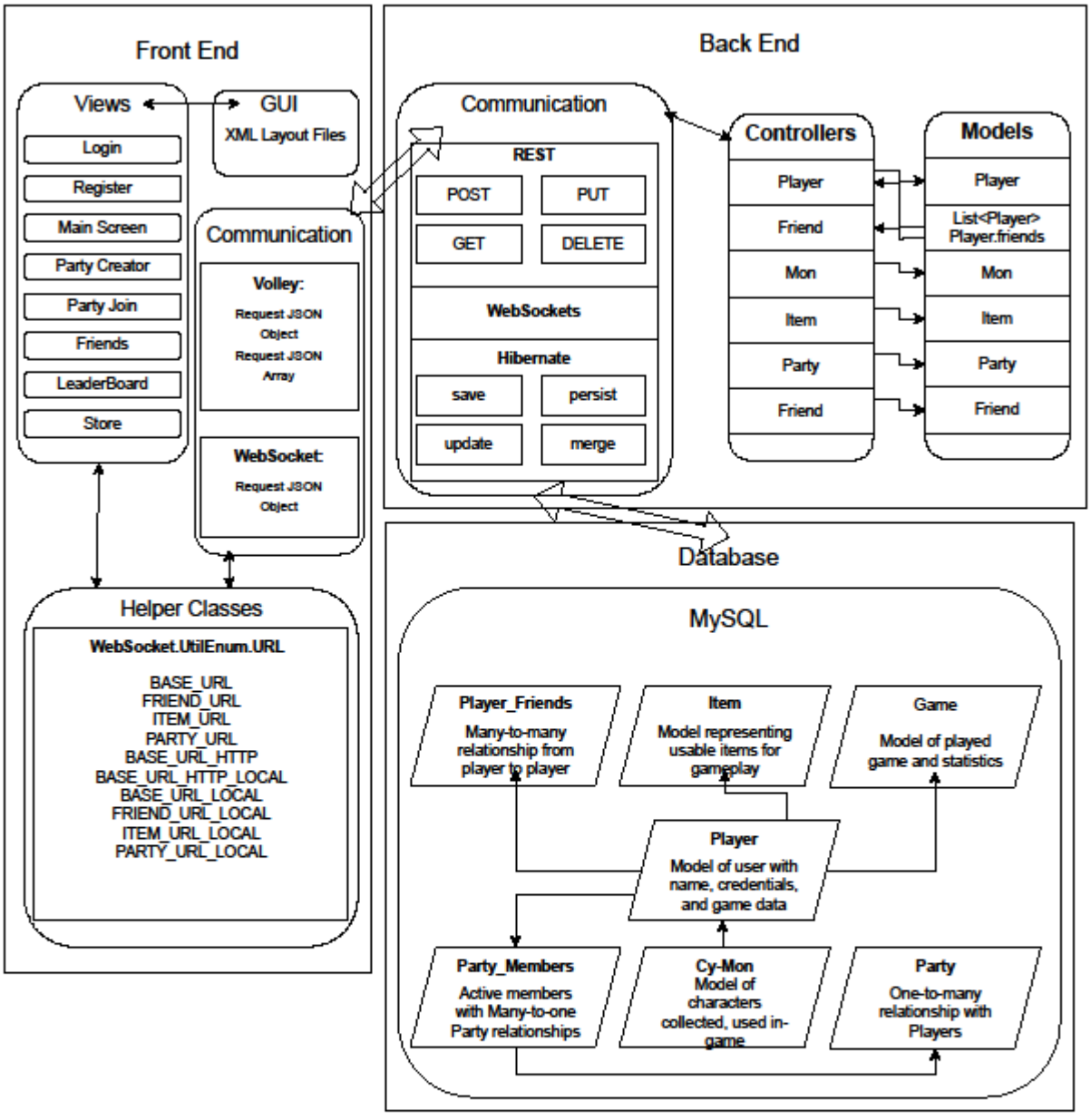
Group ms-318

Grant Pierce

Jack Krause

Danny Rosenthal

BLOCK DIAGRAM:



Design Description:

Frontend (currently implemented):

Registration Screen (User)

- Create account that generates a page with the following elements
    - EditText: UserID
    - EditText: Password
    - EditText: Email
    - Buttom: Create Account
- Upon clicking the button ‘CreateAccount’ the values of the UserID, Password, and Email are sent as a POST request to the server.

Login (Admin, User):

- Create account with account that generates a page with the following elements
    - EditText: Email
    - EditText: Password
    - EditText: Registration
    - Button: Sign In
- Login Screen takes user input of their username and password and completes a GET request from the server into the dynamic table ‘Sign In’ and if that username and password are valid, it will take the user to the main menu, where they are logged in as that user.

Main Screen (User):

- Generates page with the following elements:
    - Button: Friends
    - Button: Store
    - Button: Battle
    - Button: Party Chat
    - Button: Leaderboards
    - Button: Party Creator
    - Button: Join Existing Party
- Main Screen acts as a primary interface and navigational hub within the application to offer users a comprehensive view and access to multiple features.

Friends Screen (User):

- Generates a page with the following elements
    - ChipGroup: Group of Chip buttons to select a player who is you friend.
    - Chips: Buttons representing players the user has added as a friend
    - Button: Sends a post request to add the respective player to your friend list
    - Button: Gets the list of friends for a respective player
    - EditText: User can enter a player's name to add them as a friend

The Friends Activity allows users to enter input and add friends to their list. Once a player is added, a GET request can be sent to show the current player's friends.

Store Screen (User):

- Generates a page with the following elements
    - Button: Pressable button representing an item users can purchase
    - Button: Button that sends GET method to retrieve leftover balance for a user
    - TextView: Data container shows information regarding all items

The Store activity shows instances of Items in a scrollable list. Users can choose any items to purchase. When the purchase button is pressed, the user account is debited the price of the item. The leftover amount is then shown on the user's screen.

Battle Screen (User):

- Generates a NewGameFragment and author page for the user to access the game portion of the application
    - Button: New Game
    - Table: Authors
- Sends post request to server to set up new game for user signed in using userID value

Trainer Screen (User):

- Generates page with the following elements
    - EditText: Enter Trainer Name
    - Button: 3 start Cymons (Squirtle, Bulbausaur, Charmander)
    - EditText: Enter your Cymon nickname
    - Button: Create Trainer
- Upon clicking the "Create Trainer" button, the user-submitted information is sent to the server through a POST request, where the new Trainer profile is generated and associated with the specified Trainer Name, selected starter Cymon, and their personalized nickname.

Cymon Battle Main Menu (User):

- Generates a page with the following elements
    - Button: Wild Cymon
    - Button: Trainer Battle
    - Button: Cymon center
    - Button: Change team
    - Button: Save
- Cymon Battle Main Menu acts as a pivotal point for users engaged in Cymon battles, offering a convenient and comprehensive interface where they can initiate various battle-related activities, manage their Cymon teams, engage in battles against wild Cymons or other trainers, and ensure the preservation of their progress

Wild Cymon (User):

- Generates a page with the following elements
    - Button: Fight
    - Button: Cymon
    - Button: Items
    - Button: Run
- Each button represents a specific action or choice available during a sers encounter with a wild Cymon. Uses websockets to live update healthbars between the trainers Cymon and the wild Cymon. Buttons give options to cater to different strategies and preferences, offering users the flexibility to engage in battle, manage their team, use items, or strategically retreat if necessary, enhancing the overall interactive experience.

Trainer Battle (User):

- Generates a page with the following elements
    - Button: Fight
    - Button: Team
    - Button: Items
    - Button: Run
- Each button represents a specific action or choice available during a sers encounter with an opponent (another trainer). Uses websockets to live update healthbars between the trainers Cymon and opponent trainer. Buttons give options to cater to different strategies and

preferences, offering users the flexibility to engage in battle, manage their team, use items, or strategically retreat if necessary, enhancing the overall interactive experience.

Cymon Center (User):

- Generates a page with the following elements
    - Button: Restore Cymon
- The “Restore Cymon” button provides users with a convenient means to ensure their Cymons are in prime condition for battles or further engagements. Sends POST requests to update Cymons current health on the server.

Change Team (User):

- Generates a page with the following elements
    - Button: displayed any current Cymon a trainer has
- The “Change Team” feature provides a user-friendly interface for trainers to manage their Cymon team composition and allow them to rearrange their team or substitute team members with others from the collection.

Party Chat (User):

- Generates a page with the following elements
    - EditText: Username
    - Button: Chat Room
    - TextView: Chat room messages
    - EditText: Type your Message
    - Button: Connect
    - Button: Send
    - Button: Main Menu
- Chat room interface facilities real-time communication using websockets and allows users the ability to join specific chat rooms using the chat room button, view ongoing conversations, type and send messages, and navigate back to the main menu to select other sections of the application.

Leaderboards (User):

- Generates a page with the following elements
    - Button: Load scores for the respective game
    - Button: Enter a new score to be saved to the arraylist of scores

    - Button: Create a new player with the score to be saved
    - EditText: Score
    - EditTest: Player name
    - TextView: Leaderboard list of scores from

The leaderboard activity allows users to save an instance of a game played with a score. Users can also use a GET request to pull all scores from the database. When this method is called, the best score from every user is retrieved from the database and shown on the leaderboard.

Party Creator (User):

- Generates a page with the following elements
    - Button: select amount of players (2, 4, 8, 16)
    - EditText: Party Name
    - Button: Back
    - Button: Continue
- The Party Creator interface establishes a party within the application by specifying the number of participants and assigning a unique name to the group. When the user hits the continue button they sent a POST request to the server into the dynamic table ‘Party’.

Party Player IDs (User):

- Generates a page with the following elements
    - EditText: Enter the name of the players
    - Button: Create Party
- Party Player ID interface assists users in assembling the list of player names required to create a party within the application. Once the necessary player information is entered, clicking the “Create Party” button triggers the formation of the party sending a POST request to the server into the dynamic table ‘Party’.

Join Existing Party (User):

- Generates a page with the following elements
    - TextView: A list containing active parties
    - Button: Loads existing active parties
    - Button: Join an existing party that has room for another member
    - RadioGroup: A group of radio buttons that allows the user to join only one available party

    - RadioButtons: Buttons with labels of active parties

This activity allows users to join parties that are active and have room for another member. When a user sends the GET request, buttons are loaded with data for the parties. Users can select a party and join. When the user joins, the following screen will show current members in that party.

Backend Communication:

The backend utilizes REST mappings to update the database, determining operations based on the information sent to the mappings' URLs. The mappings used in the project consist of:

- GET: Retrieves information from the server to return to the user regarding the state of objects in the database.
- POST: Edits the state of objects in the database given specific input from the user.
- DELETE: Deletes certain objects or fields of objects given specific input from the user.

Additionally, we extensively employ websockets to control various aspects of our app, including the friends list, leaderboard, store, party creation and status, and various elements of the actual battle screen.

Controllers:

We employ controllers to route mappings for communication between the frontend, backend, and the database. The controllers used include:

- FriendController: A websocket-based controller containing operations for adding, removing, and sending the state of a player's friends list.
- GameController: A websocket-based controller responsible for keeping track of the games that a player has played and other statistics related to specific games that the player has completed (yet to be implemented).
- ItemController: A websocket-based controller used to add and subtract items from a player's inventory. The controller can also retrieve information regarding the items in the player's inventory.
- MonController: This controller is responsible for updating and sending the current state of the CyMon in the player's party, from the monster's current HP to whether they are the battling monster or not.

- PartyController: A websocket-based controller responsible for updating and retrieving a player's party and the different parties that are currently active. It allows the user to join and create parties on the fly.
- PlayerController: A websocket-based controller that deals with the sign-in functionality. It keeps track of the players currently registered in the app's database and updates it when somebody new registers. It also contains the logic for the current player sign-in.

Models:

We use various models (objects) to represent different aspects of the application that need to be updated and tracked over time. These include:

- Player: A user of the application.
- Party: A group of players that can play together.
- Item: An item that can be used by the player in battle or add some sort of cosmetic effect.
- Game: An instance of a game played that has various statistics about what happened when it was played.
- CyMon: A specific CyMon mapped to a specific player for their use in the app.

Backend Schema