**Project Report- the FaceMergeQuiz App**

Gur Keinan 213635899 Idan Pipano XXXXXXXXX Itamar Reinman XXXXXXXXX

**Introduction**

**Details on the Utility of the Server**

**Technical Implementation Details**

**Appendix 1 - Reflection on the Most Challenging Aspects of the Project**

**Appendix 2 – Images from the FaceMergeQuiz App**

**Appendix 3 – Endpoints Description**

In this section, we will go through all the endpoints available in the app, describing their functionality and role in the overall app workflow.

Add the classification API endpoint.

1. `/` - Home

- Method(s): GET

- Authentication Required: Yes

- Description: This endpoint renders the home page for logged-in users. It displays general information about the user and the current game status.

- Role: This is the main landing page after the user logs in.

2. `/sign-up` - Sign Up

- Method(s): GET, POST

- Authentication Required: No

- Description: Users can sign up for an account by providing a username and password. On form submission (POST), the app checks if the username already exists and, if not, creates a new user with encrypted credentials and logs it into the database.

- Role: Facilitates user registration and redirects the user to the home page upon successful sign-up.

3. `/login` - Login

- Method(s): GET, POST

- Authentication Required: No

- Description: The login page allows users to enter their username and password to authenticate. On a successful login, the user is redirected to the home page.

- Role: Handles user login, verifies credentials, and initiates a user session.

4. `/logout` - Logout

- Method(s): GET

- Authentication Required: Yes

- Description: Logs the user out of the app and redirects them to the home page.

- Role: Ends the user’s session and ensures proper logout behavior.

5. `/join-game` - Join Game

- Method(s): GET

- Authentication Required: Yes

- Description: Allows the logged-in user to create a new game or join an ongoing game. At this age, the user chooses one of 3 options – create a new game, join a game using code (a game someone else has already created), and join a random game – wait until someone else wants to join a random match and be paired up.

- Role: Handles game joining for users, directing them to the game lobby.

6. `/start-game` - Start Game

- Method(s): GET

- Authentication Required: Yes

- Description: Enables the user to create a new game with a unique game code. It checks for any existing games the user may have initiated and deletes them before creating a new one to avoid duplication, as it does not hurt any other user.

- Role: Responsible for game creation, storing the game details in the database, and redirecting the user to the waiting room for the new game.

7. `/waiting-room-created-game` - Waiting Room for Created Game

- Method(s): GET

- Authentication Required: Yes

- Description: Renders the waiting room for a user who has created a game and is waiting for another player to join.

- Role: Provides an interface for the user to wait for a second player to join the created game.

8. `/check-created-game` - Check Created Game Status

- Method(s): GET

- Authentication Required: Yes

- Description: Periodically checks if another player has joined the game created by the user.

- Role: Used for polling the server to check the status of the created game.

9. `/leave-created-game-waiting-room` - Leave Waiting Room for Created Game

- Method(s): POST

- Authentication Required: Yes

- Description: Allows the user to leave the waiting room and cancel the game if no other player has joined. It is used in cases where a player leaves the game, and we don’t want another player to join its already created game accidentally. To that end, if we recognize an inactive player (one that left the waiting room), we delete the created game from the database.

- Role: Manages game cancellation and session cleanup when the user decides to leave the created game's waiting room.

10. `/join-random-game` - Join Random Game

- Method(s): GET

- Authentication Required: Yes

- Description: Puts the user in a queue for a random game. If a match is found, it pairs the user with another player and starts a game. A game starts if there are two or more players waiting in the queue. After a pairing has been found, a new game object is created to contain all the information that is important for the workflow of the game, and both players are removed from the waiting queue. One player is also transferred from the waiting room to the page at the beginning of the game.

- Role: Handles joining random games, placing the user in a waiting pool, and pairing them with other users.

11. `/leave-random-waiting-room` - Leave Random Waiting Room

- Method(s): POST

- Authentication Required: Yes

- Description: Activated if a user in the waiting room for the random game is recognized to be inactive. We remove such players from the waiting queue, ensuring that no active player will later be paired up with an inactive player.

- Role: Cancels any random game match attempts and cleans up session data.

12. `/check-random-game` - Check Random Game Status

- Method(s): GET

- Authentication Required: Yes

- Description: Periodically checks if the user has been paired with another player for a random game. A player in the waiting room occasionally pools the server and checks if it has been paired up.

- Role: Provides the user with real-time status updates about their random game matchmaking.

13. `/waiting-room-random-game` - Waiting Room for Random Game

- Method(s): GET

- Authentication Required: Yes

- Description: Displays a waiting room for users who have joined a random game and are waiting for another player to be matched.

- Role: Manages the user interface for waiting in a random game matchmaking process.

14. `/enter-code` - Enter Game Code

- Method(s): GET, POST

- Authentication Required: Yes

- Description: Allows users to enter a specific game code to join a private game. Check if the entered code matches any existing games.

- Role: Facilitates joining private games using a game code.

15. `/game-ready` - Game Ready

- Method(s): GET

- Authentication Required: Yes

- Description: Displays the page when both players are ready and the game has been initiated.

- Role: Finalizes the game setup and ensures that both players are ready to proceed with the game.

16. `/load\_image` - Load Image

- Method(s): GET

- Authentication Required: Yes

- Description: Allows the user to load an image for the game. The users have two choices – either uploading an image locally from their device or searching for an image in our built-in search engine, which contains many images of celebrities.

- Role: Responsible for handling the image upload or selection process for the game.

17. `/upload\_image` - Upload Image

- Method(s): POST

- Authentication Required: Yes

- Description: Handles the image upload process from the user, storing it in the database.

- Role: Manages user image uploads and updates the game data with the provided images.

18. `/check\_merge\_ready` - Check If Images Are Merged

- Method(s): GET

- Authentication Required: Yes

- Description: Periodically checks if both users have uploaded their images and if the images have been successfully merged.

- Role: Used to poll the server for the status of the image merging process.

19. `/show\_merged\_image` - Show Merged Image

- Method(s): GET

- Authentication Required: Yes

- Description: Displays the merged image from both users and provides a multiple-choice quiz for guessing the correct answer.

- Role: Presents the merged image to the user and allows them to make a guess based on the provided options.

20. `/submit\_guess` - Submit Guess

- Method(s): POST

- Authentication Required: Yes

- Description: Handles the guess submission from the user and determines if the guess is correct or incorrect.

- Role: Processes user guesses and determines game outcomes.

21. `/game\_result/<result>` - Game Result

- Method(s): GET

- Authentication Required: Yes

- Description: Displays the game result (win/lose) after the guess has been submitted.

- Role: Provides feedback on the result of the game, either congratulating the user or encouraging them to try again.

22. `/cancel\_game` - Cancel Game

- Method(s): POST

- Authentication Required: Yes

- Description: Cancels the game initiated by the user and updates the game status to `canceled`. This function is used if one player is recognized as inactive during the game (after a pairing was found) and before the inactive player has uploaded their image. In this case, we don’t want the active player to wait forever for an image that isn’t going to be sent, so we inform them that the game has been canceled and transfer them to the home page. Then, we clean up the remains of the canceled game to avoid any accidents later on.

- Role: Allows users to cancel their active game if they wish to exit early.

23. `/check\_game\_status` - Check Game Status

- Method(s): GET

- Authentication Required: Yes

- Description: Checks if the game has been canceled and cleans up any associated game data.

- Role: Polls the server for the current game status to determine if it is still active or canceled.

24. `/game\_cancelled` - Game Cancelled

- Method(s): GET

- Authentication Required: Yes

- Description: Displays a message to the user if the game has been canceled.

- Role: Informs the user that their game was canceled either by them or their opponent.