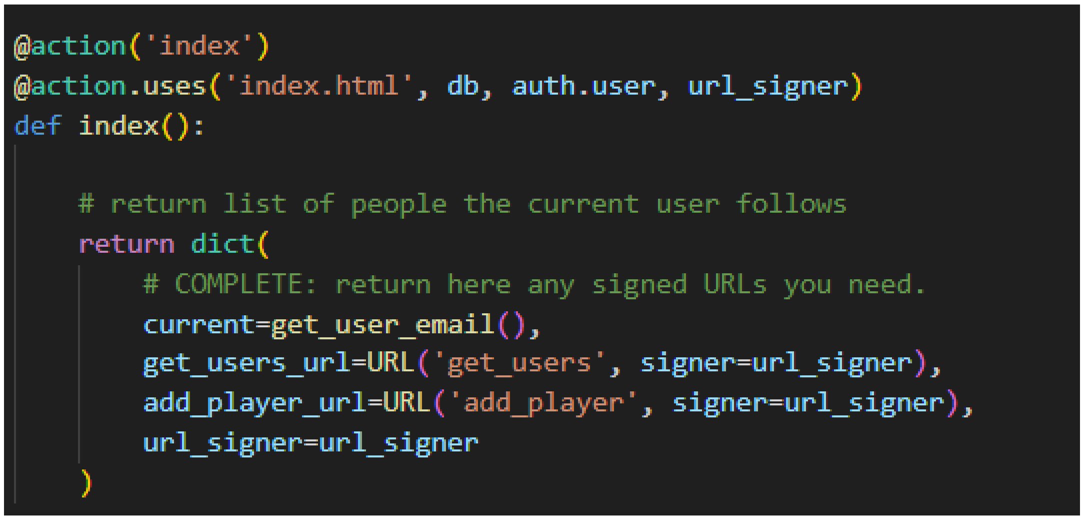
ASSASIN Documentation

BY: Maxim Carbonell-Kiamtiâ Preethika Rangamgari

Section 1: Authentication

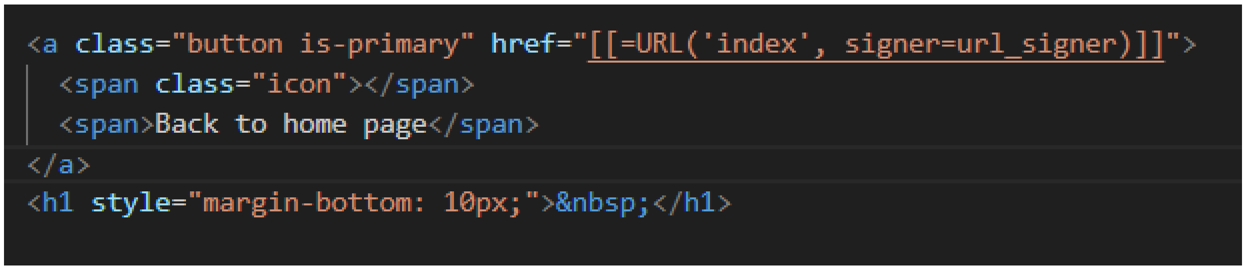
The Assasin project utilizes py4web's built-in authentication system and URL signers for secure user access. Authentication is enforced on the "index," "group_page," "game_page," and "statistics_page" of the app. In the controllers.py file, the @action.uses field includes auth.user and url_signer.verify to ensure authentication and valid URL signatures for each page. Figure 1 can be observed below to show how we use url signer and auth.user to make sure that if a user visits a page and is not logged in, py4web will make sure that they do so.



```python
@action('index')
@action.uses('index.html', db, auth.user, url_signer)
def index():

    # return list of people the current user follows
    return dict(
        # COMPLETE: return here any signed URLs you need.
        current=get_user_email(),
        get_users_url=URL('get_users', signer=url_signer),
        add_player_url=URL('add_player', signer=url_signer),
        url_signer=url_signer
    )
```

Figure 1: Authentication in the index page

In addition, every time we visit a page, we ensure that we include the url signer when a user wishes to be redirected to another page. Figure 2 below demonstrates how this behavior is achieved in html.



```html
<a class="button is-primary" href="[[=URL('index', signer=url_signer)]]">
  <span class="icon"></span>
  <span>Back to home page</span>
</a>
<h1 style="margin-bottom: 10px;">&nbsp;</h1>
```

Figure 2: Using url signers when navigating to pages

Section 2: Backend Database

2.1 Group Database:

The "group" table contains the following fields:

- creator: Represents the user who created the group.
- current_assasin: Stores the username of the current assassin in the group.
- winner: Indicates the winner of the group, either "bystanders" or the username of the assassin.
- players: A list of strings representing the players in the group.
- bot: A boolean field indicating whether the group is hosted by a bot.
- active: A boolean field indicating the active status of the group.

2.2 Player Database:

The "player" table includes the following fields:

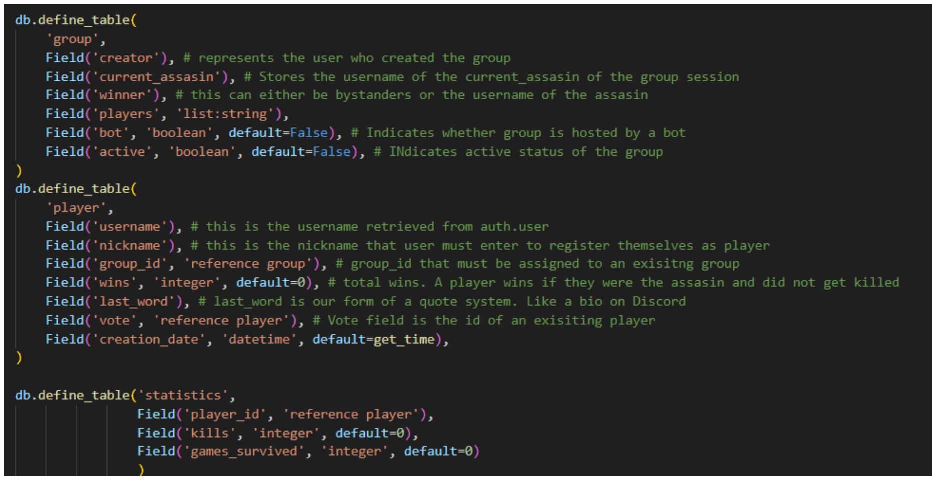
- username: Represents the username of the player.
- nickname: Stores the nickname of the player.
- group_id: A reference to the "group" table, linking the player to a specific group.
- wins: Tracks the number of wins achieved by the player.
- last_word: Stores the last message or communication from the player.
- vote: A reference to another player, representing the player's vote.
- creation_date: Stores the datetime of the player's creation, using the "get_time" function.

2.3 Statistics Database:

The "statistics" table tracks statistical information for each player, including:

- player_id: A reference to the "player" table, associating the statistics with a specific player.
- kills: Stores the number of kills made by the player.
- games_survived: Tracks the number of games survived by the player.

Note: The database structure outlined above represents the foundation of the Assasin project and allows for efficient data management and retrieval within the application.



```
db.define_table(
    'group',
    Field('creator'), # represents the user who created the group
    Field('current_assasin'), # Stores the username of the current_assasin of the group session
    Field('winner'), # this can either be bystanders or the username of the assasin
    Field('players', 'list:string'),
    Field('bot', 'boolean', default=False), # Indicates whether group is hosted by a bot
    Field('active', 'boolean', default=False), # INdicates active status of the group
)
db.define_table(
    'player',
    Field('username'), # this is the username retrieved from auth.user
    Field('nickname'), # this is the nickname that user must enter to register themselves as player
    Field('group_id', 'reference group'), # group_id that must be assigned to an exisitng group
    Field('wins', 'integer', default=0), # total wins. A player wins if they were the assasin and did not get killed
    Field('last_word'), # last_word is our form of a quote system. Like a bio on Discord
    Field('vote', 'reference player'), # Vote field is the id of an exisiting player
    Field('creation_date', 'datetime', default=get_time),
)

db.define_table('statistics',
                Field('player_id', 'reference player'),
                Field('kills', 'integer', default=0),
                Field('games_survived', 'integer', default=0)
                )
```

Figure 3: Definition of Database

Section 3: Controllers/Routing

The controllers.py file in the Assasin project defines various actions, which are functions mapped to specific URLs. These actions handle the logic and processing of requests in the application. Let's explore the key components of the controllers.py file:

3.1 Actions and Decorators:

- The file begins with an explanation of the @action(path) decorator, which exposes the function at a specific URL. It mentions different decorators used within the file, including @action.uses, which specifies the fixtures (e.g., 'generic.html', session, db, T, auth) required by an action.

3.2 Index Action:

- The index() action is mapped to the 'index' URL and renders the index.html template. It utilizes auth.user and url_signer to enforce authentication and provide signed URLs for necessary functionalities.

3.3 Group Page Action:

- The group_page() action handles GET and POST requests for the 'group_page' URL. It uses the 'group_page.html' template and requires authentication and valid URL signatures. The action provides various signed URLs for interacting with the group page's functionalities.

3.4 Game Page Action:

- The game_page() action is responsible for the 'game_page' URL. It uses the 'game_page.html' template and enforces authentication and URL verification. The action provides signed URLs for different game-related operations.

3.5 Statistics Page Action:

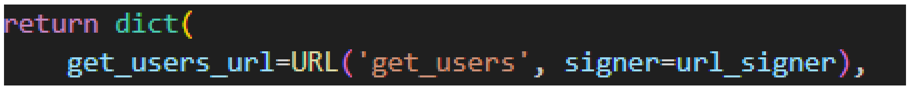
- The statistics_page() action handles the 'statistics_page' URL and renders the 'statistics_page.html' template. It requires authentication and valid URL signatures. The action provides a signed URL for adding last words.

3.6 Backend Functions:

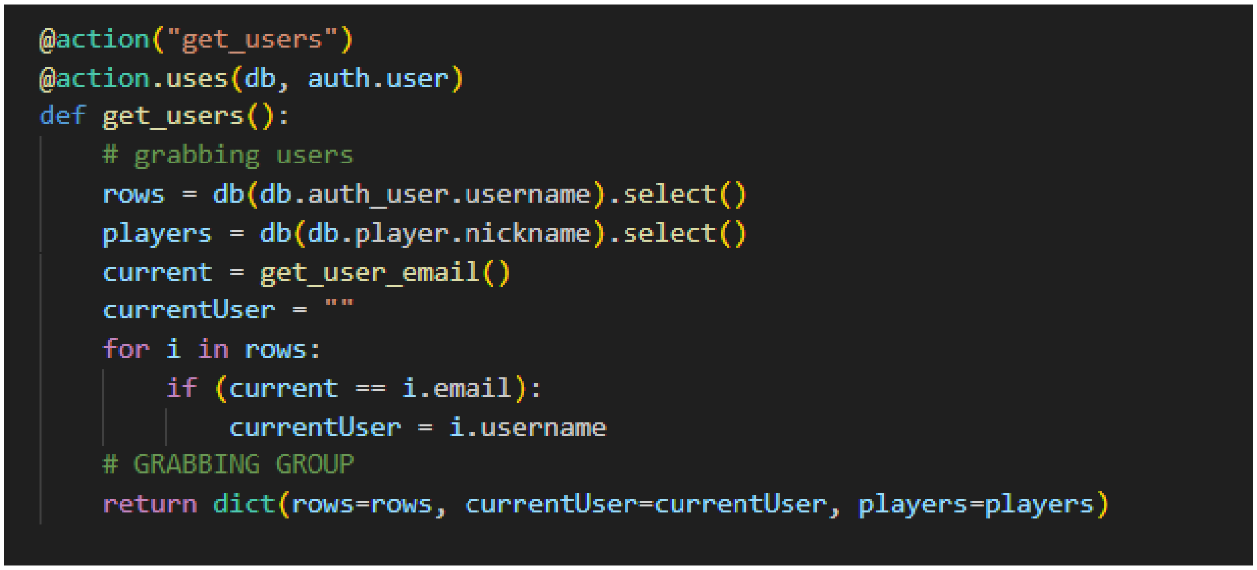
- The controllers.py file also includes several backend functions used by the actions. These functions interact with the database and perform various operations, such as fetching users, creating groups, changing group IDs, adding players, setting group activity, voting for players, adding wins, and adding last words.
- We call these backend functions with the use of axios.get and axios.post in our vue.js scripts.

Notes on Workflow:

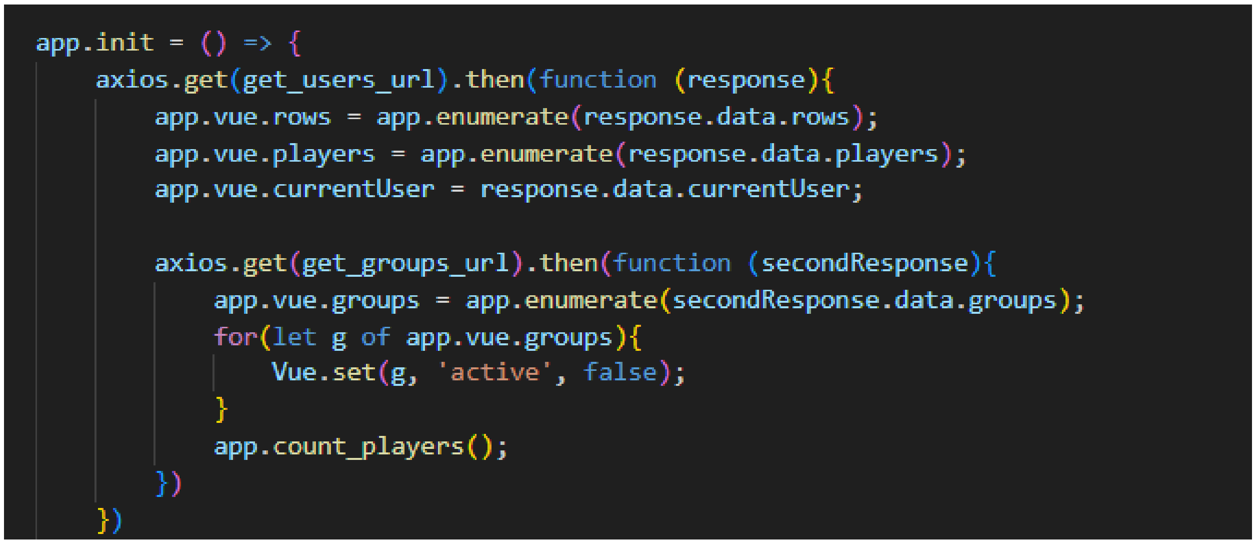
- In every page we return a dict() with essential urls that route to our backend functions. Figure 4 is an example of our most used backend functions get_users_url which routes to our get_users() backend function.
- We can then call these urls in our Vue scripts when we wish to read or write information into the backend.



```
return dict(
    get_users_url=URL('get_users', signer=url_signer),
```



```
@action("get_users")
@action.uses(db, auth.user)
def get_users():
    # grabbing users
    rows = db(db.auth_user.username).select()
    players = db(db.player.nickname).select()
    current = get_user_email()
    currentUser = ""
    for i in rows:
        if (current == i.email):
            currentUser = i.username
    # GRABBING GROUP
    return dict(rows=rows, currentUser=currentUser, players=players)
```



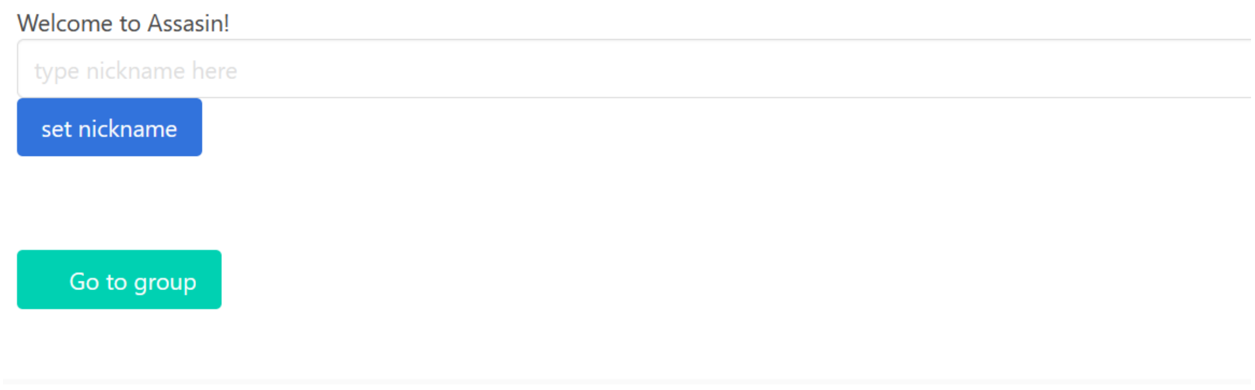
```
app.init = () => {
    axios.get(get_users_url).then(function (response){
        app.vue.rows = app.enumerate(response.data.rows);
        app.vue.players = app.enumerate(response.data.players);
        app.vue.currentUser = response.data.currentUser;

        axios.get(get_groups_url).then(function (secondResponse){
            app.vue.groups = app.enumerate(secondResponse.data.groups);
            for(let g of app.vue.groups){
                Vue.set(g, 'active', false);
            }
            app.count_players();
        })
    })
```

Figure 4: Frontend Reading from Backend

Above is our most prominent frontend-backend interaction which is the retrieval of the players in our database via the get_users_url. Everytime we load a page we update and use axios to send a GET request to the backend and feed our response into our Vue object variables.

User Story 1: Home Page

The home page serves as the entry point for users in the Assasin application. Its purpose is to facilitate user registration by prompting them to enter a nickname if they are new to the game. Existing players are presented with buttons to access the statistics page or the group page directly, providing a streamlined user experience.



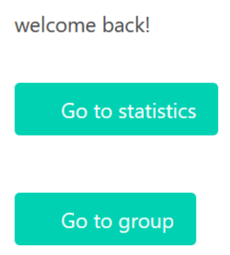


Figure 5: Home page when user is not registered vs when they are registered as a player

User Story 2: Group Page

The group page in the Assasin application provides users with a comprehensive workflow and interactive features for managing game groups. Here's an overview of the key functionalities and user experience:

5.1 Navigation Buttons:

- The group page includes buttons that allow users to easily navigate to the statistics page or return to the home page. These buttons provide quick access to relevant sections for users to view game statistics or explore other features.

5.2 Workflow Explanation:

- The group page offers a clear explanation of how the workflow operates within the Assasin application. It guides users through the following steps:
- Minimum Group Size: A group must have at least three players to start a game session.
- Joining Groups: Users can join groups that do not have another player already in them. This ensures fair gameplay and prevents duplicate entries.
- Guessing the Assassin: Once a user joins a group, they participate in the game session where they need to guess who the assassin is, similar to a "whack-a-mole" concept. This adds an engaging and interactive element to the gameplay experience.
- Creating Own Group: Users have the option to create their own group, which automatically removes them from their previous group. This allows users to take control and initiate their own game sessions.
- Joining Other Groups: Users can join other groups by clicking on them, expanding their opportunities for gameplay and collaboration.
- Total Player Count: The group page continuously counts and updates the total number of players in each group, providing real-time information on group size and activity.

5.3 Group Activeness:

- An important component of the group page is the concept of group activeness. Only one group can be active at a time. Once a group becomes active, a button appears that enables users to access the active group's game session directly. The activation of a group is managed by a recurring function called "check_status()", which runs every two seconds. This function ensures that only one group is active at any given time, providing a focused and organized gameplay experience.

5.4 Group Activity Requirements:

To maintain fair gameplay and ensure an engaging experience, certain requirements are in place regarding group activity:

- Minimum Player Count: A group must have at least two players to be eligible for activation and game sessions.
- Active Group Selection: If a user clicks on another group, that group becomes the active one, as it is the most recently set group. This ensures that users have access to the most relevant and up-to-date game sessions.

Figure 6 below demonstrates what the group page looks like. As shown, you can see that the user can be redirected to the statistics page and the home page. Davidson's group is capable of being active because there are 3 players enrolled in the group. In addition, since the user clicked on "set active", the button at the top appears to go to the game session.

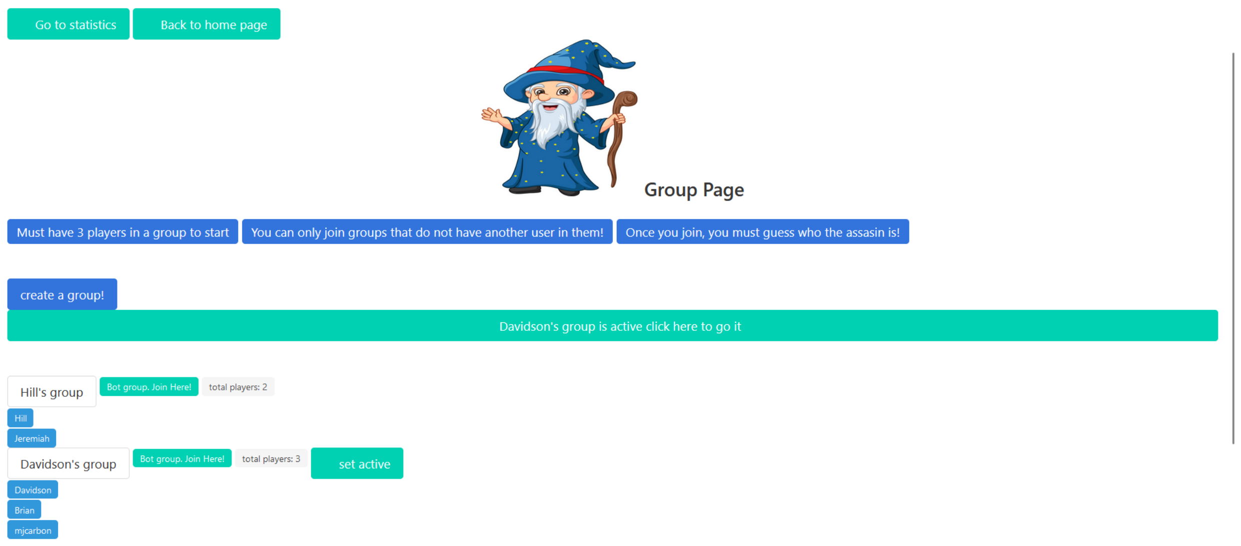


Figure 6: Group Page

User Story 3: Game Page

The game page in the Assasin application provides an engaging and time-sensitive gameplay experience. Here's an overview of the key elements and user flow:

6.1 Timer and Gameplay Mechanics:

- Upon reaching the game page, a 30-second timer is initiated, creating a sense of urgency and excitement. During this interval, users have the opportunity to cast their vote for who they believe the assassin is. They can change their vote as many times as they want before the timer reaches zero.

6.2 Current Assassin and Group Activation:

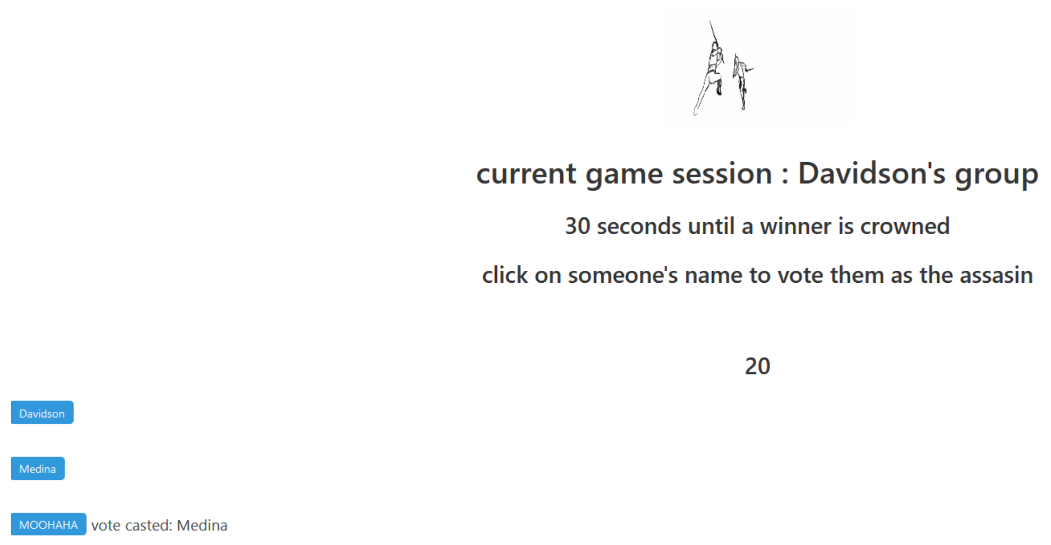
- At the start of the game session, the current assassin is determined, and the group is set to be active. This sets the stage for gameplay and creates suspense as players try to identify the assassin within the given time limit.

6.3 Game Outcome:

- When the timer reaches zero, the group is set to be inactive. The game outcome depends on whether the assassin was killed or not:
- Assassin Not Killed: If the assassin successfully remains unidentified and survives, the current assassin is declared the winner, earning a +1 in their wins database.
- Assassin Killed: If the assassin is correctly identified and killed, the bystanders win, and no player receives a win.

6.4 Returning to Group Page:

Once the game session concludes, users are presented with the option to click "Go back to groups page." This allows them to navigate back to the group page, where they can join other groups, create new ones, or explore different game sessions.
Figure 7 below shows the game page. Since MOOHAHA clicked on Medina, MOOHAHA’s current vote is Medina and when the game ends Medina is killed. In this case MOOHAHA was the assasin and so MOOHAHA gets a win.

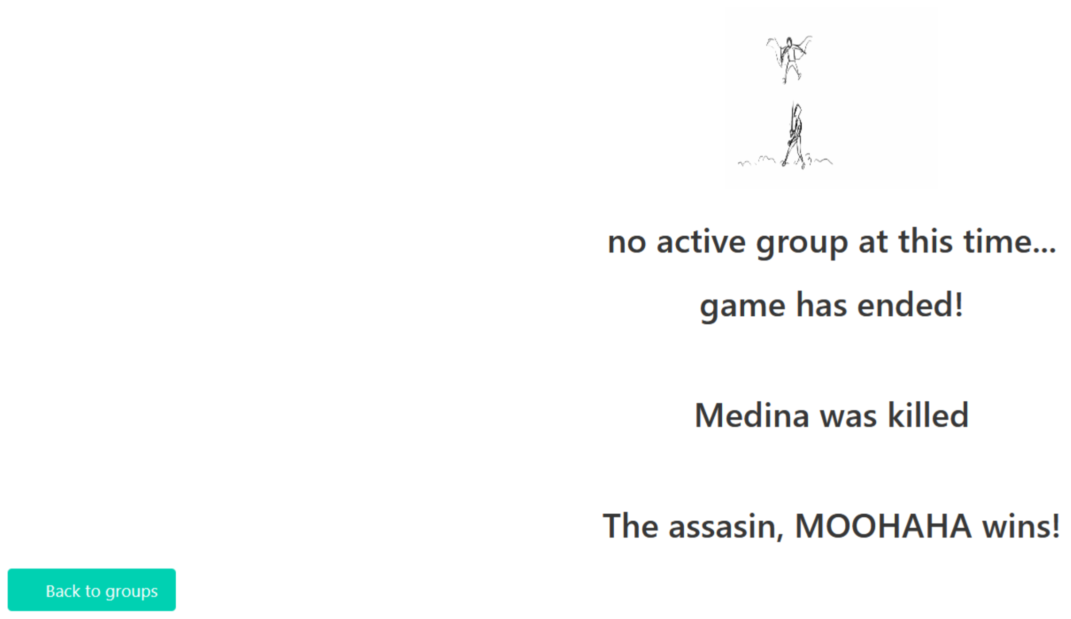

current game session : Davidson's group

30 seconds until a winner is crowned

click on someone's name to vote them as the assasin

20

vote casted: Medina


no active group at this time...

game has ended!

Medina was killed

The assasin, MOOHAHA wins!

Figure 7: Game Page

User Story 4: Statistics Page
The statistics page in the Assasin application provides users with a comprehensive overview of game-related information and features. Here's an overview of the key functionalities and user experience:

7.1 Submitting a "Last Word":

- Upon arriving at the statistics page, users have the option to submit a new "last_word." This feature allows users to share a short message or update, similar to a "bio of the day" concept found in platforms like Discord. Users can provide insights, highlights, or any other information they wish to share with the community.

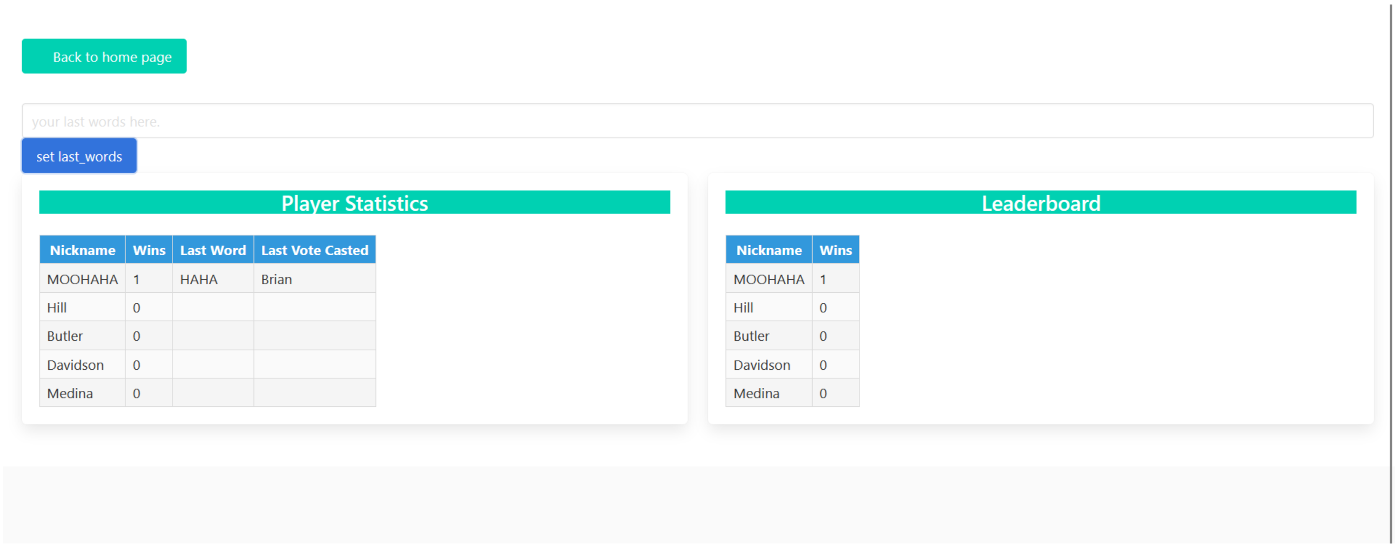
7.2 Last Voted Player and "Last Word" Display:

- On the statistics page, users can view the player they last voted for during gameplay. This information helps them track their voting history and previous interactions within the game. Additionally, the page displays the "last_word" field of other users, allowing users to see updates or messages shared by their fellow players.

7.3 Leaderboard:

- A prominent component of the statistics page is the leaderboard, which showcases the players' wins in descending order, from the highest to the lowest. This feature provides a clear visual representation of player achievements and standings within the Assasin game.

Below is Figure 8, where you can observe the informational nature of the statistics page, as well as our system of submitting an up-to-date “last_word”...



Player Statistics

Nickname	Wins	Last Word	Last Vote Casted
MOOHAHA	1	HAHA	Brian
Hill	0		
Butler	0		
Davidson	0		
Medina	0		

Leaderboard

Nickname	Wins
MOOHAHA	1
Hill	0
Butler	0
Davidson	0
Medina	0

Figure 8: Statistics Page