Multi-player Snake Game

Project Specification

Rosalie Cai*(ruixic)*, Chaoqi Li*(chaoqil)*, Ibrahim Mubarek*(imubarek)*

1 Product Backlog

- Authorization
  - Set up OAuth (Django OAuth Tool-kit) - Implement user profile, log-in, and log-out using Google Accounts
  - After user is logged in they are connected to the game home page
  - Make login required for all game HTML pages
- Game Logic & Visualization
  - Implement snake movement on an HTML canvas
    - Determine the data structure for snake body segments
    - Determine the logic for computing the direction to move for the snake
    - Make snake slide across canvas
    - Make snake change directions based on mouse position
  - Add food for snakes on the canvas to eat and become longer
  - Enable customization of snakes for users
    - Color and shape selection
    - User-defined/created snake skin pattern
    - Size
  - Optimizing canvas and snake parameters for better user experience
  - Deploy multi-player snake movement with Websockets
  - Ensure concurrency correctness for single-user locally
  - Deploy to cloud for an online concurrent multi-user game experience
- HTML & CSS Page set-up & Styling
  - Game front page
  - Authorization page
  - Leaderboard page
  - Snake Customization page
  - In-app store
- Databases
  - Construct backend storage for user profiles and statistics of previous games played
  - Potential usages include building leaderboards, displaying crashers and crashees, and displaying a user's statistics

  - Deploying with SQL database on server

- Performance Optimization
  - Cap the number of players in a room and set up a queue system. Whoever dies is placed at the end of the queue, and the player at the front joins in the room.
  - Optimize snake body segments updates during movements
  - Optimize snake collision and snake eating apple checking

2 First Sprint Backlog

- Game Logic & Visualization
  - Rosalie
    * Implement snake movement on an HTML canvas
    * Make snake change directions based on mouse position
    * Use Websockets for user communication with server to update snake position
    * Ensure concurrency correctness for single-user locally
  - Ibrahim
    * Add randomly placed food to game state for snakes on the canvas to eat and lengthen
    * Add collision logic for when multiplayer snakes run into each other
    * Add logic for how to handle collisions between the multiple snakes currently playing
    * Optimize canvas and snake parameters for better user experience

- HTML & CSS Page set-up & Styling
  - George
    * Game front page
    * Implement single page app navigation
    * Implement profile and leaderboard layout
    * Integrate canvas into game page

- Databases
  - Ibrahim
    * Construct backend storage for user profiles and game statistics

- Performance Optimization
  - Ibrahim
    * Optimize snake body segments updates during movements
    * Optimize snake collision checking with other snakes and food

3 First Sprint Product Owner

Ibrahim Mubarek, Andrew ID: imubarek

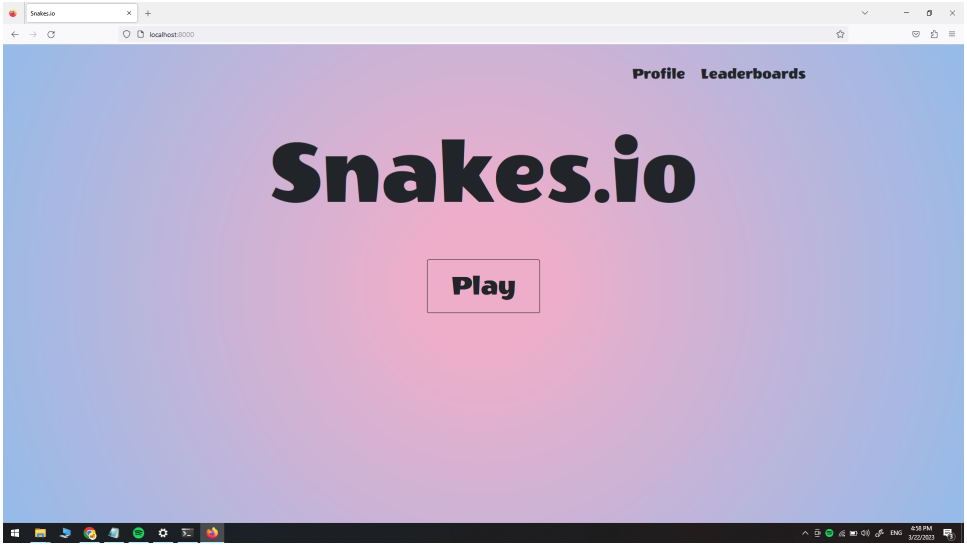
4 Data Model Implementation

```
1 class GamePlayed(models.Model):
2     user_who_played_game = models.ForeignKey(User, on_delete=models.PROTECT,
          related_name="game_user")
3     game_end_time = models.DateTimeField()
4     score = models.IntegerField(blank=True, null=True)
5
6     #Potential Fields to Have Possibly:
7     game_length = models.IntegerField(blank=True, null=True)
8     stopped_by = models.ForeignKey(User, on_delete=models.PROTECT, related_name
          ="game_winning_opponent")
9     snakes_stopped = models.ForeignKey(User, on_delete=models.PROTECT,
          related_name="snakes_stopped")
```

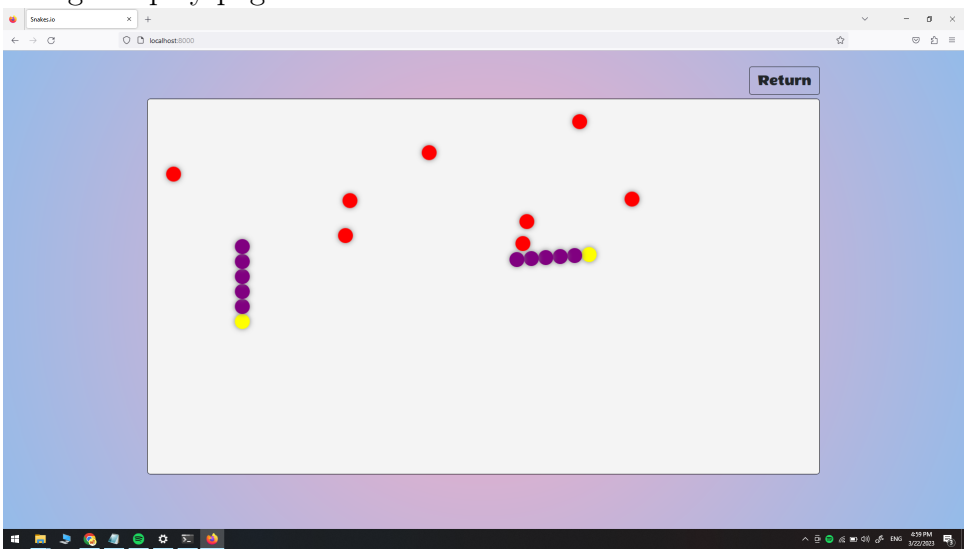
We will also be using a User model that will automatically be provided after setting up OAuth. As of now we envision these to be the two models we need but will add more models as needed.

5 Application Mock-up

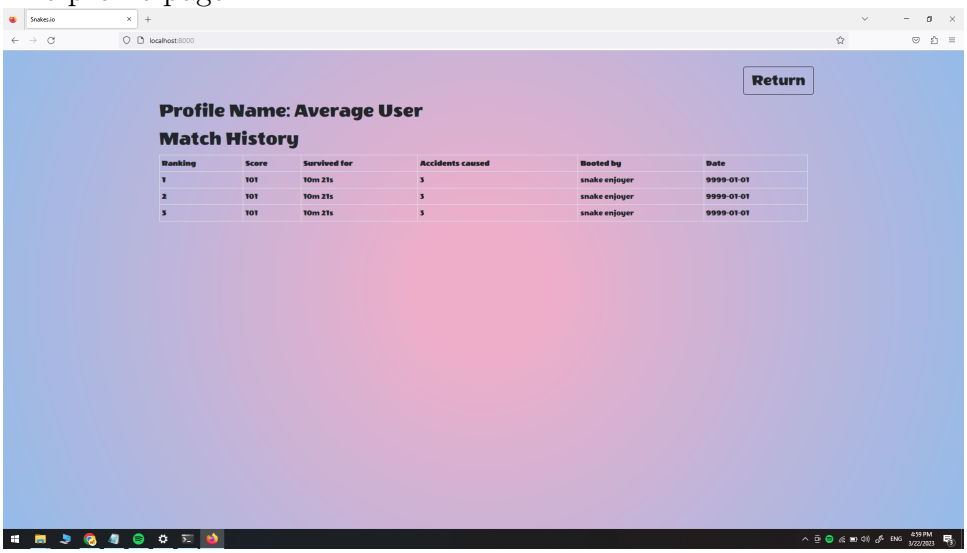
- The front page of the game



- The game play page



- The profile page



- The leaderboard page