Team Contribution

Yi Meng (andrew id: yim1, Sprint 1 Product Owner):

----------Backend----------

1. User list refresh by AJAX and Score refresh by AJAX in Sprint 1.
2. Word model building, word database importing script, hints selection, hints display format and Word selection function.
3. Websocket framework building, containing consumer, routing in backend and websocket connection building & listening in front end. Define how to route websocket request and group. Specifically, for drawing canvas real time showing in different screen. Websocket delay optimization.
4. Information searching backend for room and user searching API, send JSON format response and frontend parses that.

----------Frontend----------

5. Implement some tools for the drawing canvas: Size choose, color choose, clear button.
6. Websocket for Canvas real time interaction, showing in different screen in real time.
7. Websocket for leaving room status dynamically showing in different screen.
8. Word getting and showing same hints in different guesser pages in new turn and hints display format.
9. Information searching frontend UI.
10. Merge drawer js and guesser js and reformat that, define how to jump to drawer / guesser page.

----------Deploy----------

11. Using Apache as proxy for websocket request, Daphne as interface server and deploying that in background. Run webapp as system service and restart automatically when server error occurs.

----------Others----------

12. Write Unit Test for room model test, user profile building test and routing url test.

Yiran Zhou (andrew id:yiranz1, Sprint 2 Product Owner):

----------Backend----------

1. Room Model: Responsible for all of the game room logic, including create new rooms, join an existing room, and leave the room. Use Haikunator API to generate a random name for the newly created room. Each game room can have a maximum of 3 players, if the room is full or the game has already started, the user will be redirected to homepage if he tries to enter that room. The origin host will be the one who creates the room, but when the host leaves the room, a player in the room will be picked to be the new host based on his join-room-time. When all of the users in a room has left, that game room will be deleted.
2. Get all of the current changes in the homepage (e.g. user creates a new room) and in a specific game room (e.g. the host of the room changes) and record them using json.
3. Check the player's guessing result. Use Django Forms to validate the player's guessing input; make sure the player's score will be 0 when starts a new game; the player cannot accumulate his score by submitting the correct answer multiple times.

4. Responsible for game result logic. Record the winner of the game and each player's score using json.
5. Responsible for login and signup logic.
6. Responsible for validations in view.py using either form validation or get_object_or_404.
7. Responsible for websocket disconnect logic.

----------Frontend----------

8. Implement a fully working canvas. Implement a time counter. When user resizes the browser window, relocate the draw area.
9. Websocket for adding new room. Write home.js and use websocket to add newly created rooms to the homepage in real-time.
10. Websocket for adding new player. Write room.js and use websocket to add new player to the player list in real-time when a player enters the room; use websocket to pass the "begin" button to the new host if the old host leaves the room.
11. Websocket for displaying scores. Use websocket to display the addition of a player's score in real-time.
12. User Interface design and implementation. Use Materialize and Bootstrap to embellish our application's UI.
13. Write all of the html templates, refactor our application to eliminate code duplication in our Django templates, using Django template inheritance.

----------Others----------

14. Responsible for code cleanup (e.g. eliminate useless code and print statement).

Can Liu (andrew id : canl2, Sprint 3 Product Owner):

My contribution to the final version includes:

----------Backend----------

1. Controller Model: When the host of a room click "begin game" button, the controller is created. Each room has only one controller which is responsible for the modularization and abstraction of the game logic, including setters and getters. Specifically, it provides APIs to (a) Get the list of current users in that room ordered by the time when they joined the room.(b) Change the current painter to the next user. (c) Return who is the current painter (d) Return "Game Over" when the game ends with certain rounds.
2. All views that needs to interact with Controller Model (i.e. will be called after the host click "begin game" till all user leave the room)
3. Build the scheme to separate different websockets with different listeners to avoid conflicts that might result in delivery failure.

---------- Frontend----------

4. Implement different tools for the drawing canvas: eraser, pencil brush effect, spray brush effect.
5. Use websocket to synchronize the hints for guessing & count down.
6. Based on established design to choose to show different page contents for users when : he is the painter/he is the guesser/the game is over.(i.e. JS logic that needs to interact with Controller Model and corresponding views.)

----------Others----------

7. Write Unit Test for Controller Model.
8. Optimize the GET request to be POST request.