PICTIONARY by Fast Byte

Team Member	Roll Number
Abhijit Amrendra Kumar	210050002
Harsh Poonia	210050063
Hrishikesh Jedhe Deshmukh	210050073

We are currently using the following languages/libraries/frameworks/softwares :

- HTML, CSS, *Tailwind* and *Bootstrap* CSS frameworks
- *ReactJS* framework
- *Vite* Javascript build tool
- *npm* modules used:
  - *@fortawesome* icon library.
  - *react-toastify* for app notifications.
  - *roughjs* for the drawing canvas.
  - *socket.io* library to facilitate bi-directional and simultaneous data transfer between server and clients.
  - *uuidv4* to generate unique IDs for rooms as well as players.
  - *axios* to make HTTP requests to the server.
  - *express* web framework to make APIs.

Features and gameplay

- A page for players to create private lobbies, where they can share invite CODES to other players. After players join into a lobby, the game starts.
- Each player is given a word on their turn, which is only visible to them. They have to draw on a given canvas to describe the word without actually mentioning it directly, and the other players will receive a live rendering of the drawing made by that player as an image, using which they have to guess the word. The order of guessing yields the players different points.

- For each round of the game, every player is given a turn to draw. On completion of all 5 rounds, the game ends, and a leaderboard screen pops up showing the top 3 players of the game.

The Game

Server port is mentioned in the `env` file, which is not included in the github repo for security reasons.

Creating and Joining Rooms

- We created a room for each game instance, and generated a corresponding unique room ID.
- Players can join the room using the generated room ID
- There is a button on the top bar to copy the current room's ID whenever needed.

Game Dashboard

- The ongoing round is displayed on the top bar.
- On the left sidebar, the player list is rendered. On the right sidebar, a room chat is rendered which is synced across all clients.

The Canvas

- On the center of the screen, we used roughjs to render an HTML canvas, on which we can draw using our mouse. Using mouse click triggers in the returned div tag, we can capture mouse path and draw accordingly.
- We used a useRef to generate an image, which is sent via socket to all other clients to display.
- We used props to send a useState for changing the pencil color for drawing.
- We created the Undo, Redo and Clear buttons and linked them up with appropriate onClick functions.

Leaderboard

- For the final leaderboard, we used props and useState to update the list of top 3 players according to score, and display the final winners page.

Implementation and Working

Sockets

- Sockets were used to first communicate and synchronize user list with all clients in a particular room, and display welcome messages to the newly joined users, and the names of newly joined players to existing ones.
- In the chat area, sockets were used to relay messages from the client to the server, and from the server to all the other clients for seamless communication between clients
- We used sockets to send the state of the canvas to connected clients in a room and keep the canvases in sync.
- Answer validation and sending of drawing prompts was done on the server side and sent to clients through sockets.

React Hooks

- To keep certain elements synchronized between parent and child components, and to make re-rendering things easier, we used React Hooks useState, useEffect and useRef.
- Hooks were used to set up socket listeners in component constructors, and to update the canvas.