Made by Marin Nikolic, Nikola Igic and Marko Radulovic

Kaptcha Game

Professor: Dr. Sc. Martin Žagar
Course: ISTE- 442 Secure Web App Development

Table of Content

Game Information

The Kaptcha game that we develop is a quiz type game. Where two players go against each other in general knowledge questions.

Game Objective:
The objective of the game is to answer 10 questions correctly before the other player.

Gameplay:
The game is played 1 vs 1 in a room type environment. Every player sees the current question but only whose turn it is can answer the question. After correctly answering the question. Your player character is moved on the map by one tile. After you answer 10 questions correctly before your opponent you will receive 30 trophies for winning. After the game you will be redirected to the lobby where you can create your room again and play the game. The UI of the game is a representation of the captcha box.

Scoring:
Players earn points for each correct answer they give. In case they give an incorrect answer, they stay in the same position and the turns change.

Winning:
The game ends after answering 10 questions correctly.. The player who answers 10 questions first wins, and is given 30 trophies as a reward, while the loser is given a penalty of 20 trophies.

Game Features:

- The game can be played online, with players from all over the world.
- Players can create their own profile and track their trophies.
- The game has a chat feature that allows players to communicate with each other during the game.

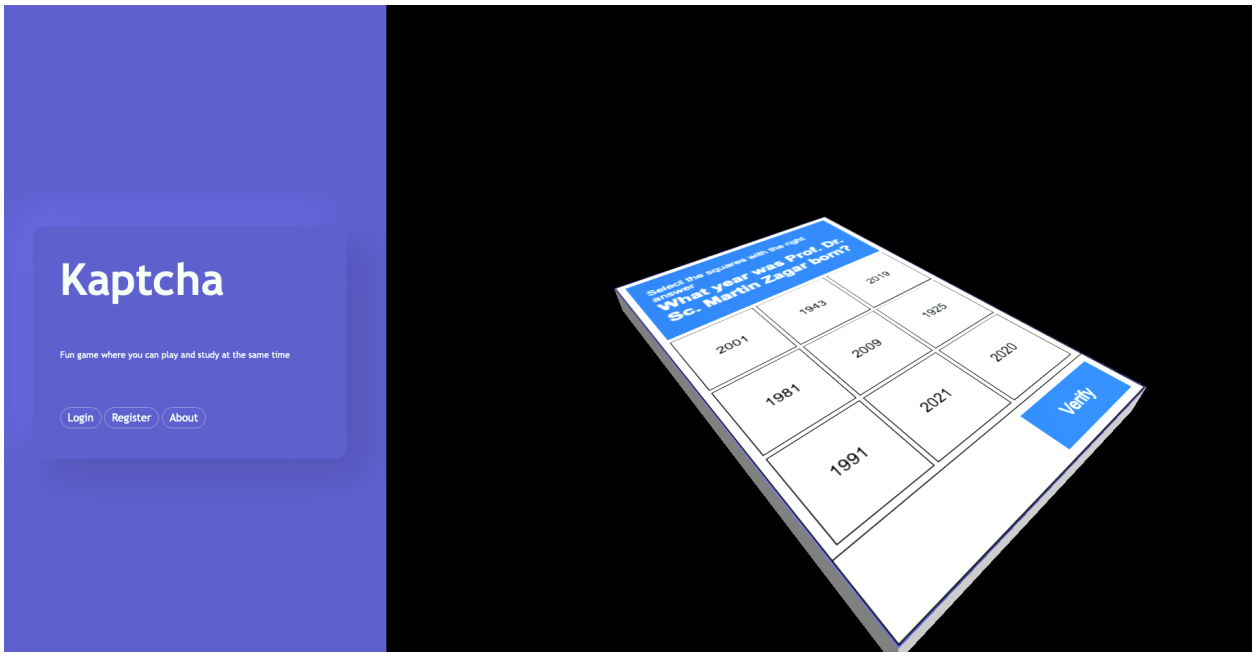
Game Features that could be added with future updates.

- Currently the game only supports text based questions and answers, with the future updates we plan on adding pictures as answers.
- Multiple category questions
- Picture Puzzles

- User being able to design his own game board and character customization.

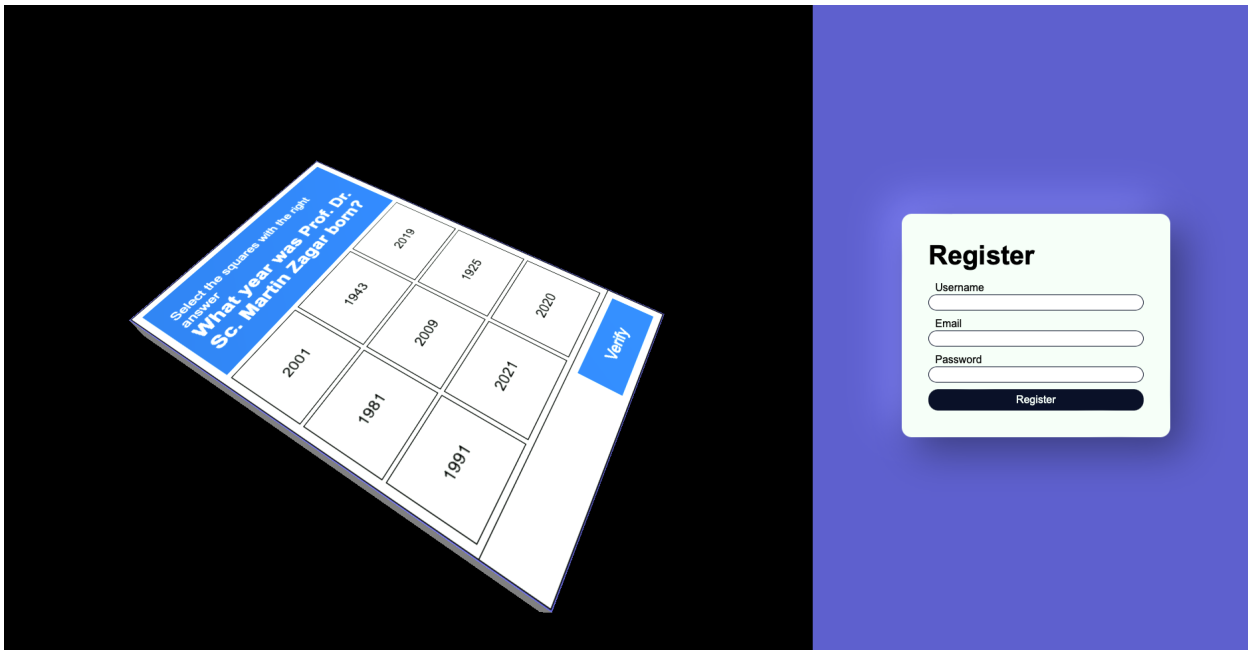
UI Explanation

Home Page



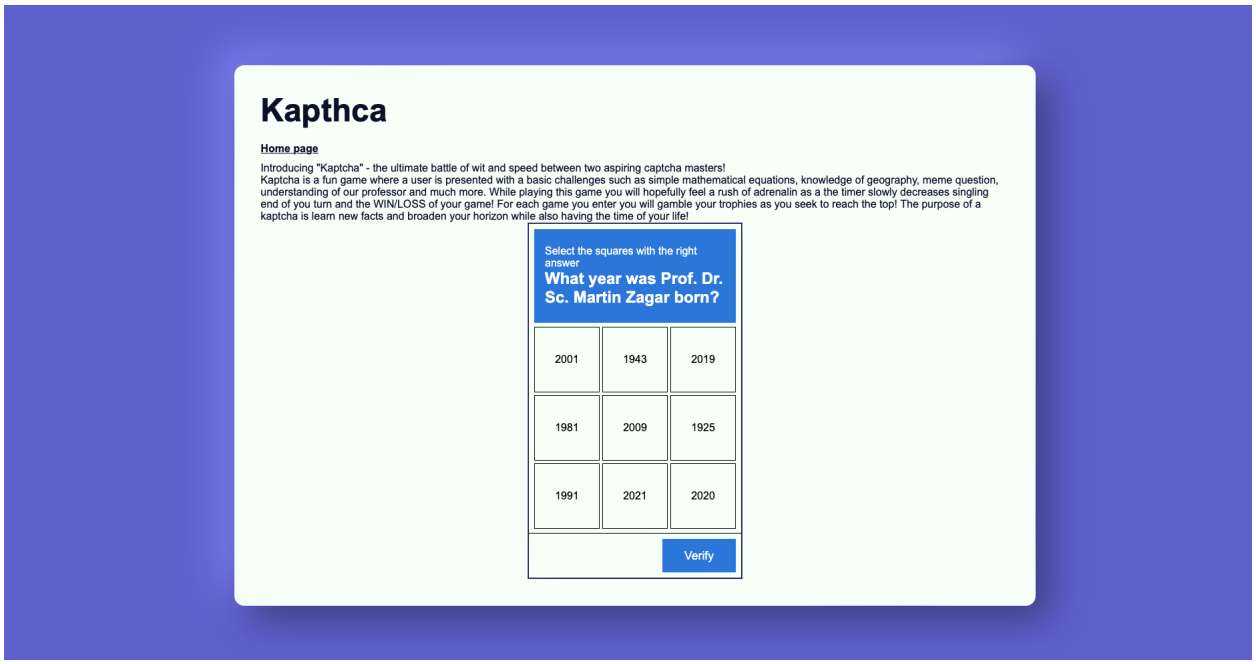
This is the first page that user/client will see upon entering our website. There is a container on the left side of the screen where we can find the title of the game, a small description and three links. There is also a 3D interactive image that should draw the attention of possibly new users.

Register



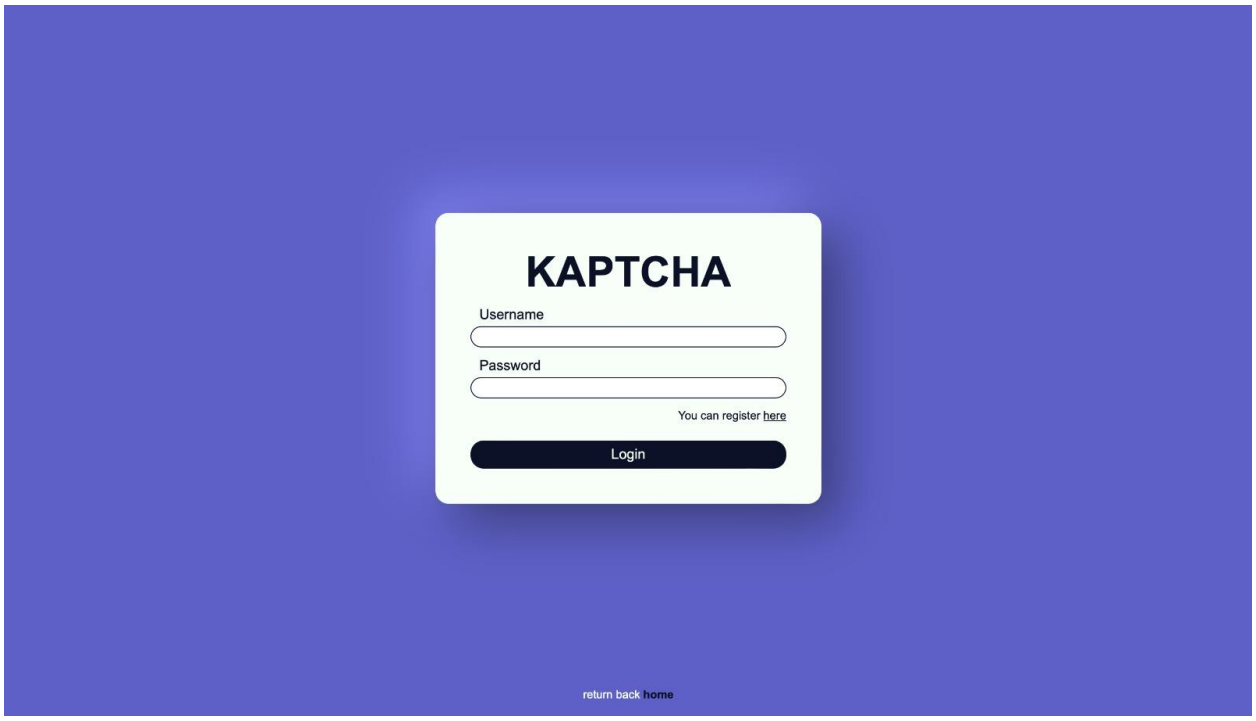
This is the page that would use new players/ clients/ users to create their account and after that play a game. It has a similar design and home page.

About



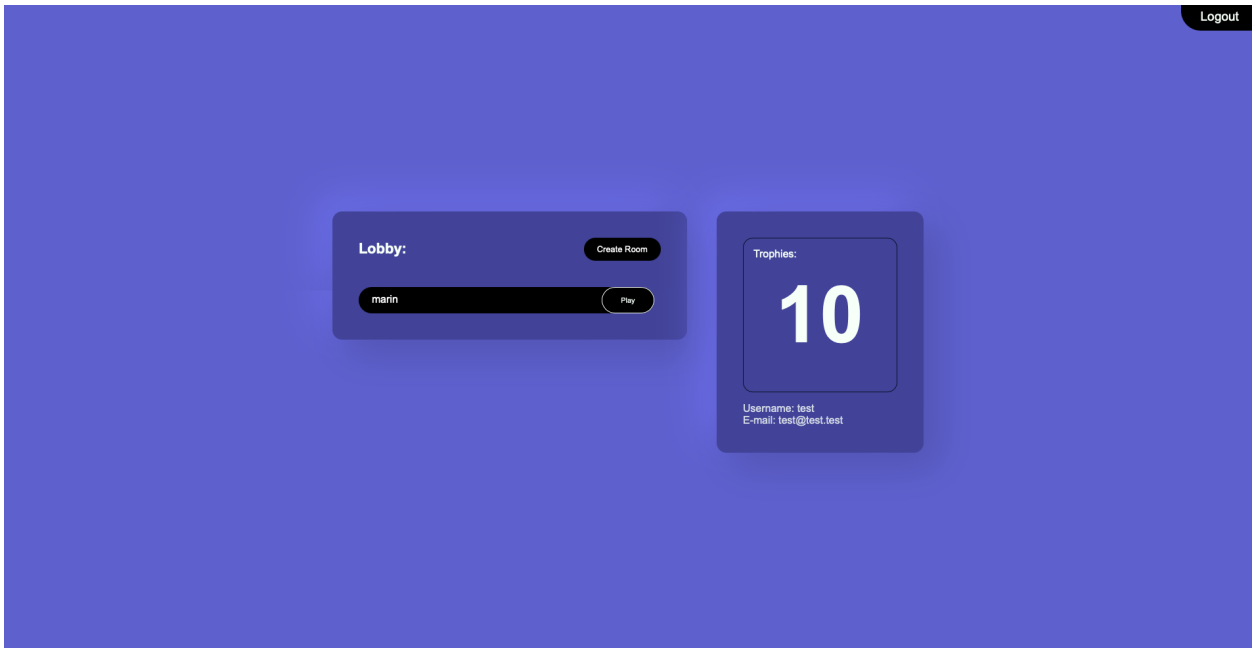
About page is a page with small description of what is our game about and what sort of challenges new players can expect.

Login



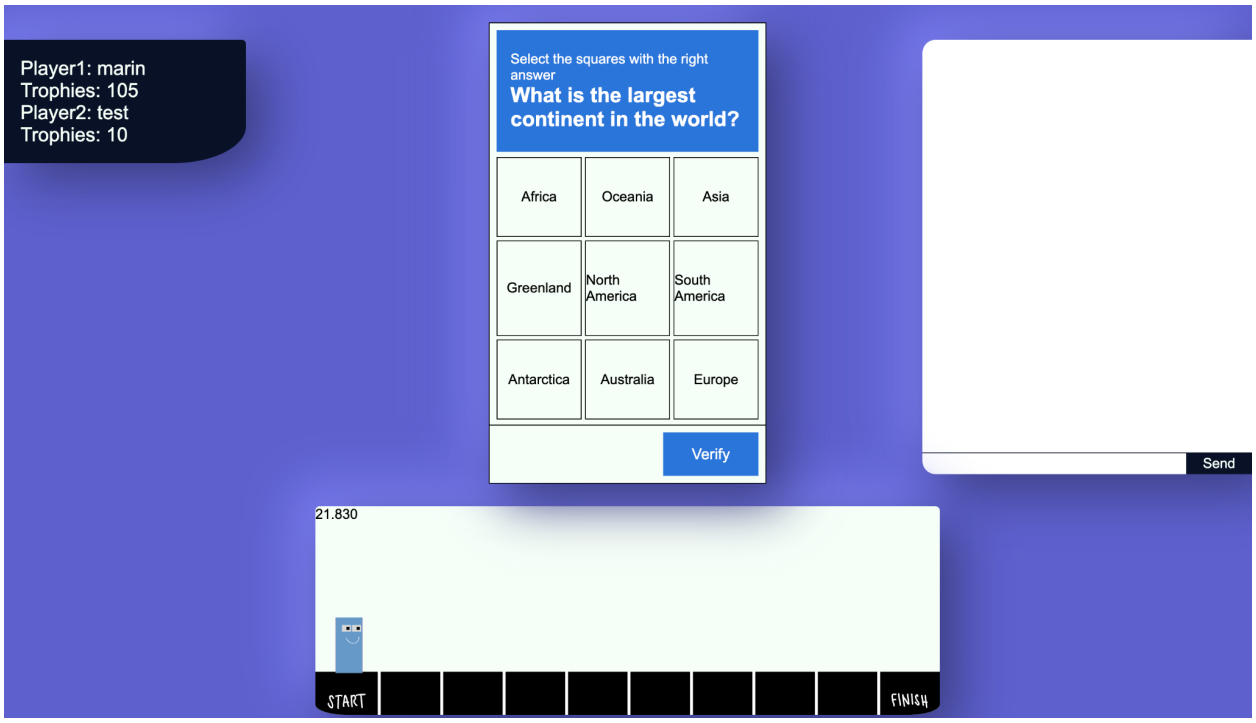
On this page this is a representation of the login page where the user can insert his username and password to join our Kaptcha game. If the user does not have an account we can redirect him back to the home page or he can click and you **can register here.**

Lobby



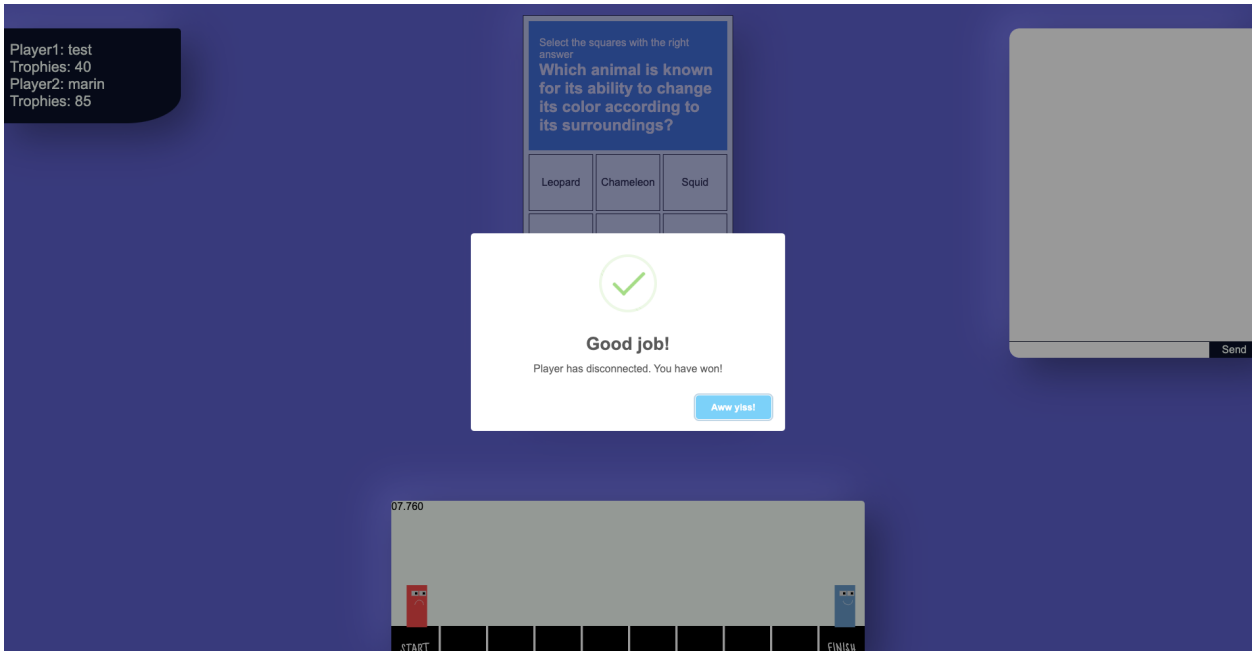
This is the page that the user sees when he/she logins in their account. We have a container ‘lobby’ where users can see rooms that are currently available to enter and challenge other players. There is also a button ‘create room’ that is used for creating your own room. Upon clicking on that button the user will immediately enter their room. There is also container user information, their username, current amount of points and their email. In the right top corner we have a logout button to leave the room.

Room



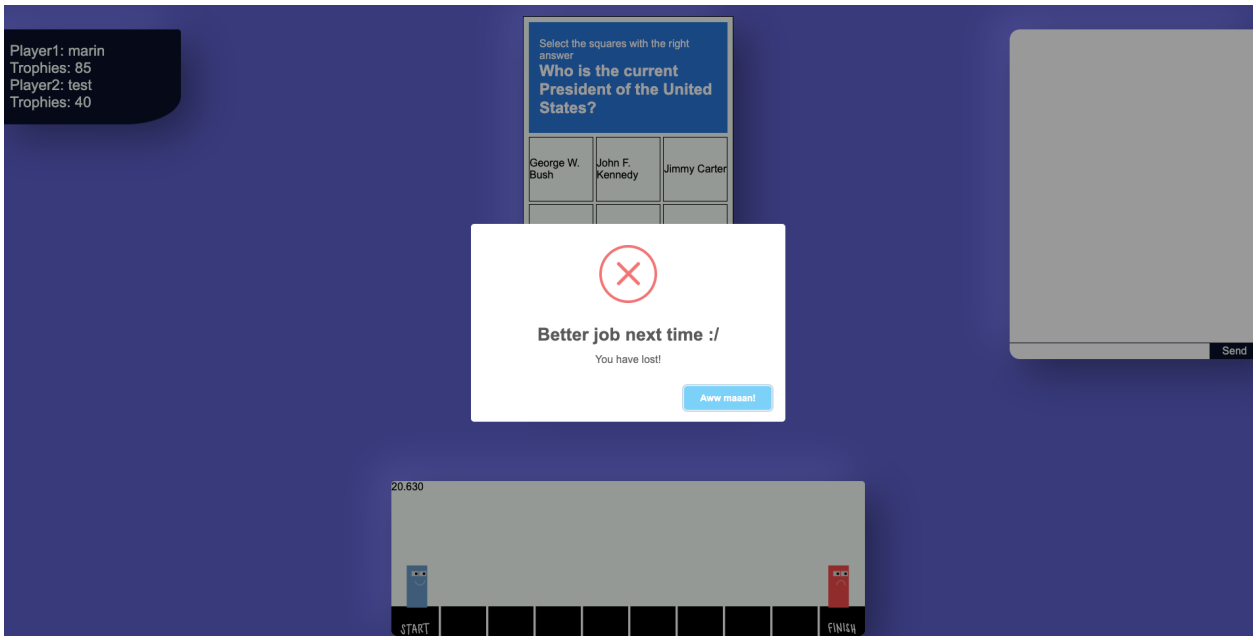
Upon entering a room, you can see other players' names and their trophies. In the middle is the game. On the left is a chat that you can use to chat with an opponent player. And on the bottom there is gamebord where you can see how many true answers you should give to win the game. There is also a small clock that shows how much time you have to answer a question. Based on the color of captcha you will know if it is your turn or enemies.

Won game



Here is the display the user is given when he wins the game. We can see two characters at the bottom. The red one signifies the opponent, while the blue one signifies the user. Once the user clicks the alert button, he is redirected to the lobby and automatically is granted 30 trophies.

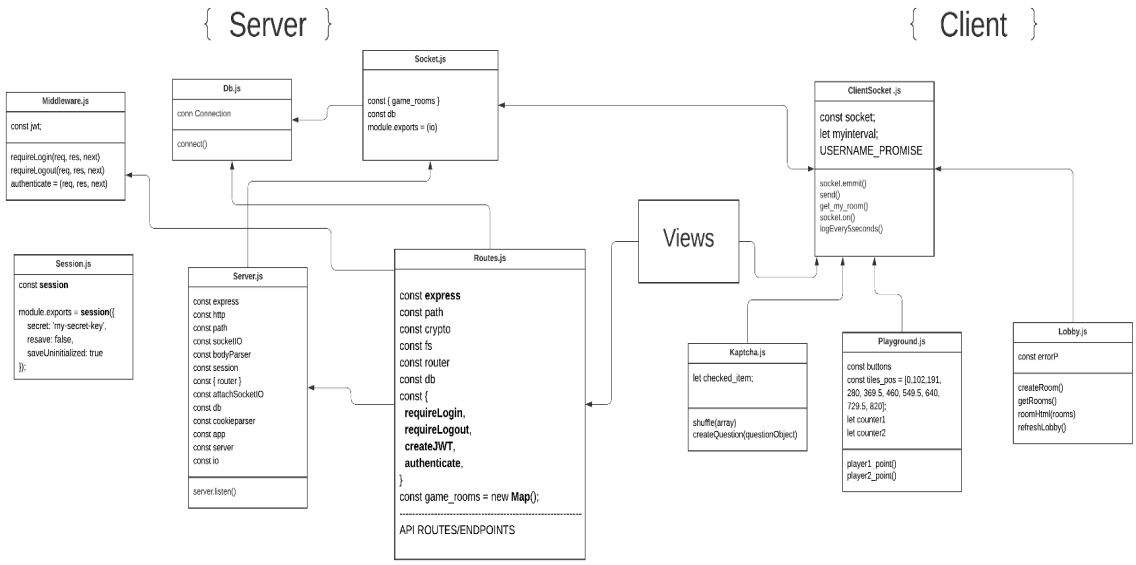
Lost game



Here is the display the user is given when he loses the game. We can see two characters at the bottom. The red one signifies the opponent, while the blue one signifies the user. Once the user clicks the alert button, he is redirected to the lobby and automatically is deducted 20 trophies.

System Architecture

Key Files and their relationship overview diagram



Key Files and their relationship

ClientSocket.js and ServerSocket.js

The ClientSocket.js and ServerSocket.js classes are highly dependent on each other, as they are responsible for facilitating communication between the game client and server. The ClientSocket.js class is responsible for establishing a connection to the game server, and for sending and receiving messages from the server. The ServerSocket.js class, on the other hand,

is responsible for receiving messages from the client, processing them, and sending responses back to the client.

The two classes are dependent on each other because they need to constantly communicate in order to keep the game running smoothly. For example, the ClientSocket.js class may send messages to the ServerSocket.js class indicating that a player has joined the game, or that a player has answered a question correctly. The ServerSocket.js class, in turn, will need to process these messages and take appropriate action based on the content of the message.

Likewise, the ServerSocket.js class may send messages to the ClientSocket.js class indicating that a player has disconnected, or that it is now the player's turn. The ClientSocket.js class will need to process these messages and update the game interface accordingly.

In short, the ClientSocket.js and ServerSocket.js classes are highly dependent on each other because they need to constantly communicate in order to keep the game running smoothly. Without one class, the other would not be able to function properly, and the game would be unable to function.

Routes.js with the ServerSocket.js and ClientSocket.js(GAME)

The routes.js class is a key component of the system architecture because it is responsible for mapping HTTP requests to the appropriate API endpoint. When a user interacts with the game interface, the routes.js class receives the HTTP request and uses it to determine which API endpoint should be triggered.

The routes.js class is dependent on other classes in the system architecture because it needs to be aware of the different API endpoints that are available and the actions that they perform. For example, if a user submits a login form, the routes.js class will need to trigger the appropriate API endpoint to authenticate the user's credentials and log them into the game.

In order for the routes.js class to function properly, it must be aware of the API endpoints that are available and how they interact with other components of the system architecture. For example, if the API endpoint for adding a player to a game room is triggered, the routes.js class

must ensure that the appropriate data is sent to the ServerSocket.js class to update the game state.

Without the routes.js class, the system architecture would be unable to map HTTP requests to the appropriate API endpoint, and users would not be able to interact with the game interface. Therefore, the routes.js class is a critical component of the system architecture, and its proper functioning is essential for the game to run smoothly.

DB.js

The db.js class is a crucial component of the system architecture because it is responsible for managing the connection to the database where the game data is stored. This includes information about players, trophies, quiz information, and other relevant data.

The db.js class is dependent on other classes in the system architecture because it needs to be aware of the data models and schema of the database in order to interact with it properly. For example, if the ServerSocket.js class needs to update a player's score, it will need to send a request to routes.js which then will forward the data to the db.js class to update the appropriate record in the database.

Likewise, if a user submits a registration form, the routes.js class will need to trigger the appropriate API endpoint to validate the user's information and create a new record in the database using the db.js class.

Without the db.js class, the system architecture would be unable to interact with the database, and the game data would be unavailable or inconsistent. Therefore, the db.js class is a critical component of the system architecture, and its proper functioning is essential for the game to run smoothly.

Server.js

The server.js class is the most important component of the system architecture because it is responsible for initializing and starting the server and the game application. This includes setting up the necessary connections and dependencies to ensure that the application runs smoothly and efficiently.

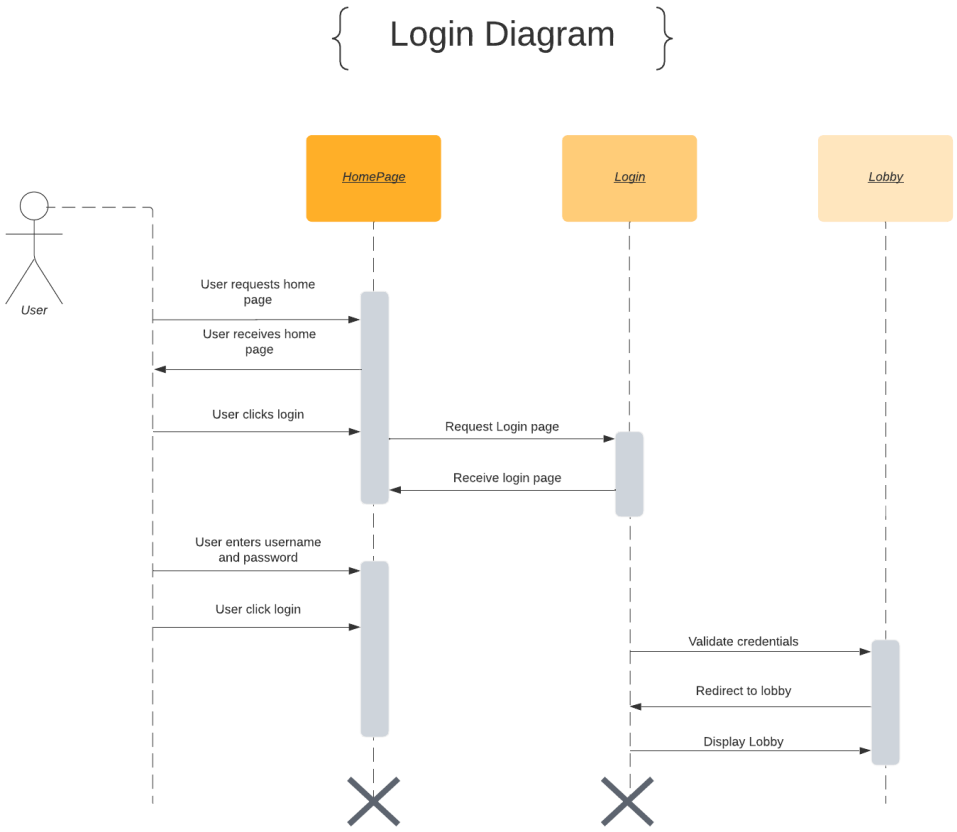
The server.js class is dependent on other classes in the system architecture because it needs to be aware of the different components and modules that are required to start and run the application. This includes dependencies such as Express, Socket.io, and the database connection module.

In order for the server.js class to function properly, it must be able to manage and interact with these dependencies as needed. For example, if a user submits a request to join a game room, the server.js class must be able to use the appropriate Socket.io and database connection modules to create and manage the game room.

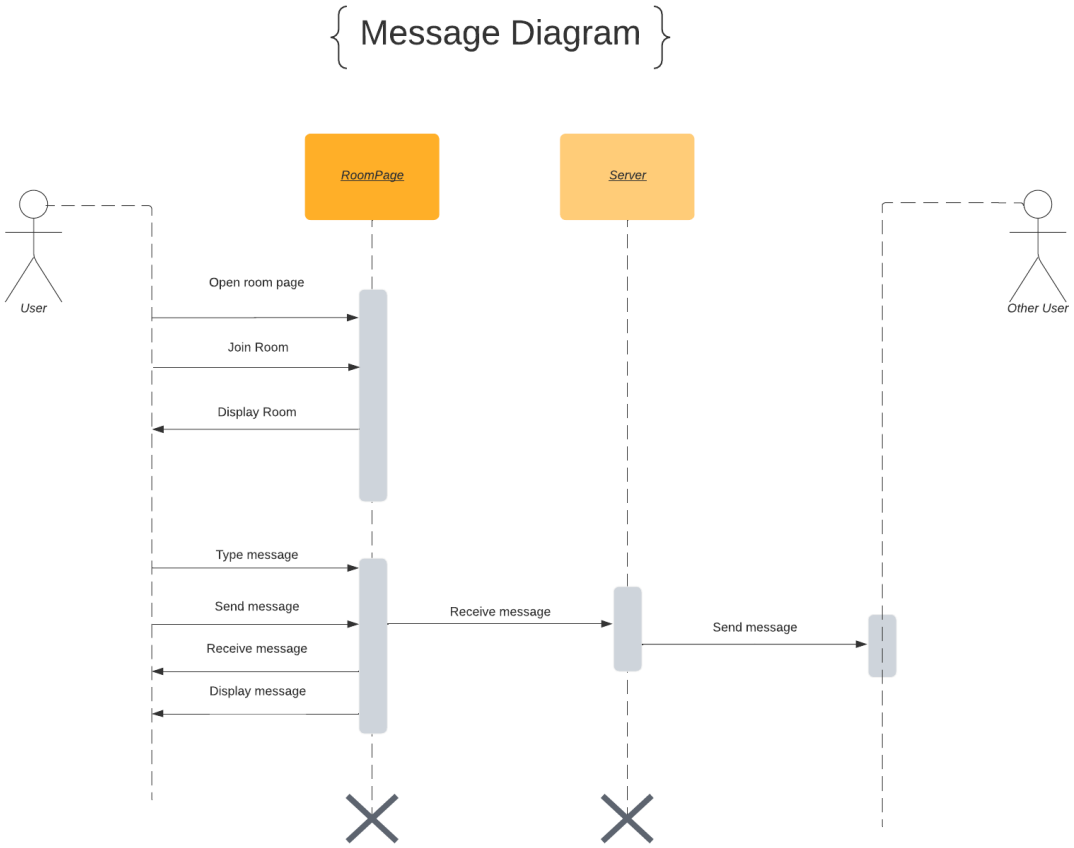
Without the server.js class, the system architecture would be unable to initialize and start the application, and users would be unable to access the game interface. Therefore, the server.js class is the most critical component of the system architecture, and its proper functioning is essential for the game to run smoothly.

Sequence Diagram

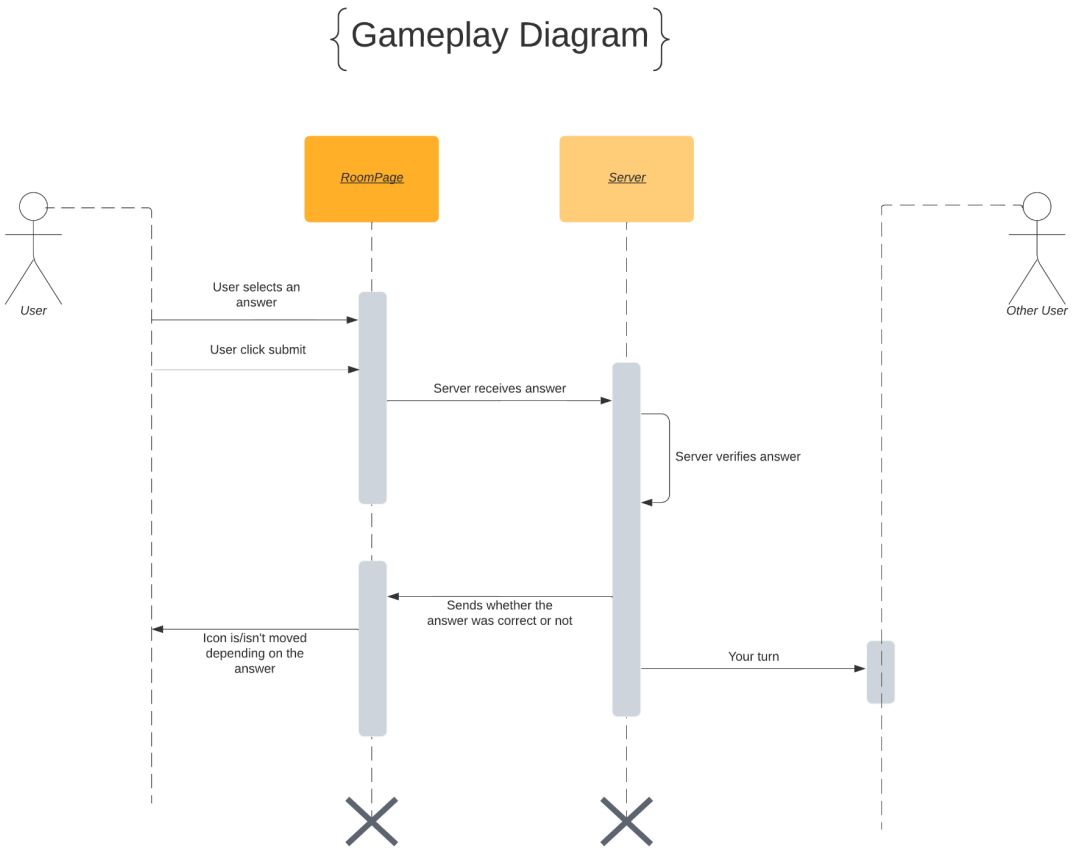
Sequence diagram Login



Sequence diagram Message



Sequence diagram Gameplay



Key decisions for our design

Node.js

Node.js is a popular choice for web development because it is a lightweight and scalable server-side framework that is particularly well-suited for handling small requests and real-time applications. This makes it an ideal choice for our captcha game, which requires real-time communication and frequent updates to the game state.

One of the key advantages of Node.js is its event-driven architecture, which allows it to handle large numbers of simultaneous connections and requests with minimal overhead. This means that our game can handle multiple players and game rooms at once without experiencing significant slowdown or lag.

Another benefit of Node.js is its ability to work seamlessly with JavaScript, which is the primary language used in our frontend user interface. This allows us to maintain consistency and compatibility between the frontend and backend of our application, making it easier to develop, test, and deploy new features and updates.

Socket.io

Socket.io is a popular and reliable library that provides real-time communication capabilities between the server and the client. It is particularly well-suited for real-time applications, such as our captcha game, which require fast and efficient communication between players and the game server.

One of the key advantages of Socket.io is its ability to handle bi-directional communication, which means that data can be sent and received in real-time between the server and the client without the need for traditional HTTP requests. This makes it ideal for our game, which requires frequent updates to the game state and real-time notifications for players.

Another benefit of Socket.io is its ease of use and compatibility with JavaScript, which is the primary language used in our game application. This allows us to easily integrate Socket.io with our frontend and backend code, making it easier to develop, test, and deploy new features and updates.

Extras

The game also features an Augmented Reality (AR) and Virtual Reality (VR) component, which is available on the game's home page, as well as the login and registration pages. The AR/VR component is a 3D model that has been integrated into the game's interface. The model represents the game and provides users with a unique and immersive visual experience.

The 3D model was sourced from the Internet and was chosen to match the theme and aesthetic of the game. The AR/VR feature provides users with a fun and engaging way to interact with the game, and adds an additional level of interactivity to the overall user experience.