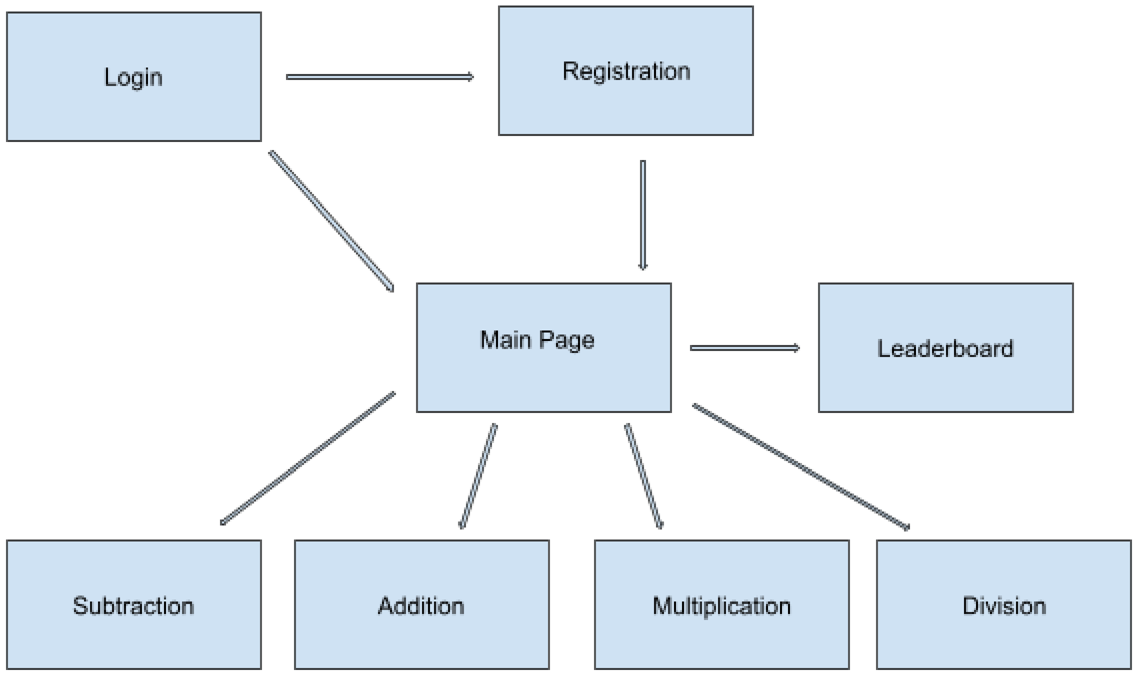
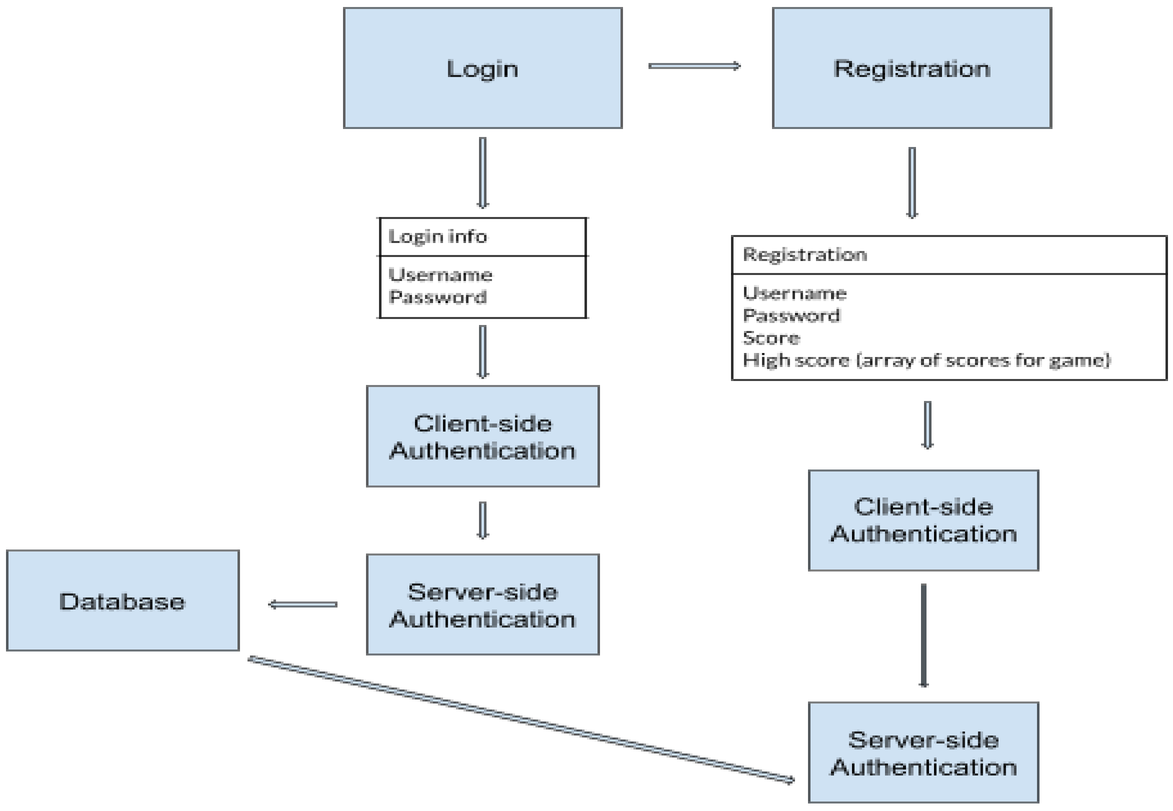
Detailed Design

Detailed Game Design:



User Authentication Design


Login
Registration
Login info
Username
Password
Registration
Username
Password
Score
High score (array of scores for game)
Client-side Authentication
Client-side Authentication
Database
Server-side Authentication
Server-side Authentication

Authentication Screens:

UI Design (Front-end, size not to scale)

Sign-In:

NavBar

Username:	<Text field>	
Password:	<Text field>	
		Sign In
		Register

Register:

NavBar

Username:	<Text field>	
Password:	<Text field>	
		Register
		Sign In

Detailed Implementation:

A new user can create an account with a unique username and password combination. This data will be stored in the database.

To log-in to an existing account, the username and password combination inputted by a user on the client-side must match data extracted in the database.

Data Members:

(Database)

- $username (string)
- $password (string)

(Local React State)

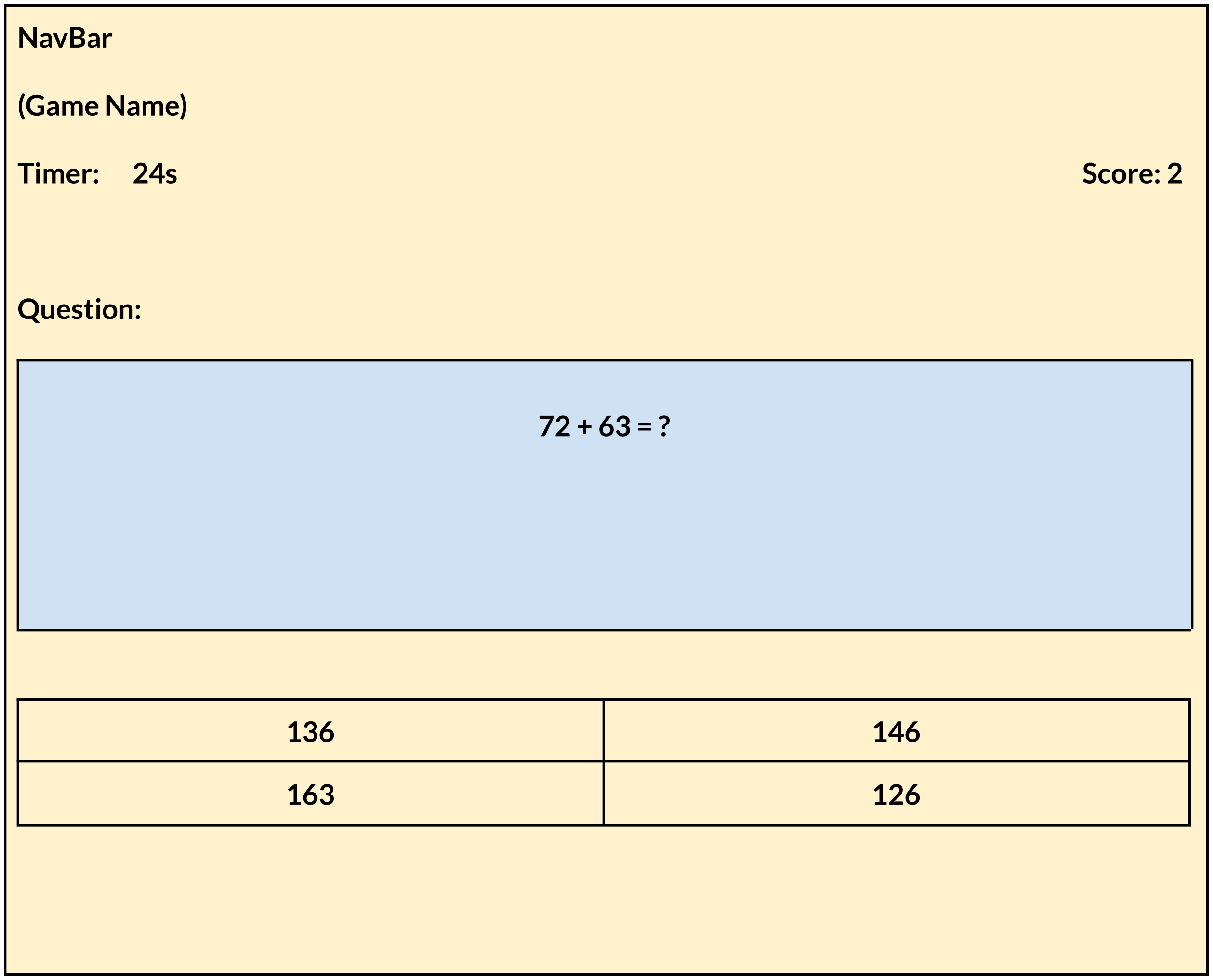
- $username (string)
- $password (string)
- $confirm_password (string)
- $is_logged_in (boolean)
- $error (boolean)

Data Methods:

- SignUserIn()
    - Checks if username and password inputted match values in database
    - Logs user in / displays error messages
- RegisterUser() :
    - Checks if username and password regex is correct
    - Checks if duplicate username found
    - Adds user to database
- IsLoggedIn() :
    - Helper function to use on all screens
    - Checks if a user is logged in or not
    - Used to differentiate between guest and non-guest user functionality
- HashPassword() :
    - Hash function to encrypt password

Game Screens

UI Design (Front-end, size not to scale):



NavBar

(Game Name)

Timer: 24s Score: 2

Question:

72 + 63 = ?

136	146
163	126

Detailed Implementation:

There are four games: Addition, Subtraction, Multiplication, and Division. A game mode object will store what type of game will be played. Questions will be generated randomly via backend and then sent back into frontend. The games are styled in a multiple choice format. When a player chooses one of the four answer options the question will change regardless of whether it is right. If the question is right the score will increment by 1 and the grid box will turn green. If the answer is wrong, the score will decrease by 1 and the grid box will turn red. The timer and answer calculations and answer checks are all done in the frontend. Only the random number generator is part of the backend. There are two modes: single player and multiplayer. Single player mode can be played by anyone (registered or unregistered players). The multiplayer mode can only be played by registered users and is between two people. The person who scores the highest wins the multiplayer mode. Only scores from the multiplayer mode can make it to the leaderboard. Instead of using multithreading to implement the multiplayer mode for the game, an alternate approach was used. Two users connect to a game object with a unique ID to connect to the same game lobby. Data is exchanged and updated on MongoDB which allows the players to play together.

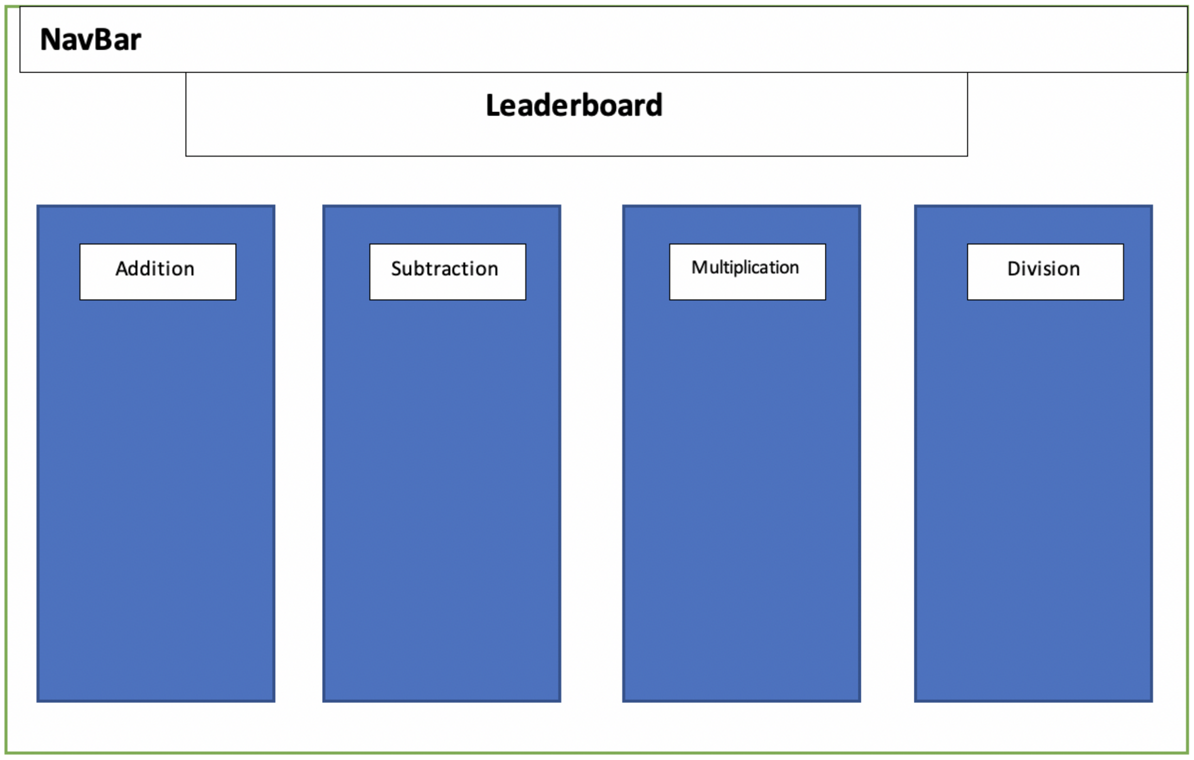
Data Members:
(Database):
- $prev_high_score

(Local React State)
- $timer (Integer)
- $curr_question (String)
- $curr_answer (Integer)
- $score (Integer)

(Local React Components)
- Game (base class):
    - NavBar
    - Game Name
    - Question
    - Submission Box
    - Correct Answer
    - Timer
- Subtraction (child class)
    - Subtract logic function
- Addition (child class)
    - Addition logic function
- Multiplication (child class)
    - Multiplication logic function
- Division (child class)
    - Division logic function

Data Methods:
- generateQuestion()
    - Prompts backend and generates a question using Math.random for two integers
- incrementScore()
    - Increases user score if question is answered correctly
- checkAnswer()
    - Checks if answer of the user matches the expected answer
- isFinished()
    - Checks if timer is finished
- upDateHighestScore

Leaderboard
UI Design (front-end, size not to scale)



<u>Detailed Implementation:</u>
The leaderboard will display the top 10 positions for each of the games: Addition, Subtraction, Multiplication, and Division.
The leaderboard will display the username, rank, and score of these users. The leaderboard is implemented using multithreading. This ensures efficient updating of the leaderboard. Each player represents a thread and is compared between other player threads. The top players will be updated onto the leaderboard.

<u>Data Members:</u>
(Database):

- $Username
- $Rank
- $Score

(Local):

- $Username: String
- $Rank: int
- $Score: int
- $Game: Game
- $Map of score, usernames, and rank: Tree Map

<u>Data Methods:</u>

- updateLeaderboardAddition()

Updates the leaderboard with the new winner for the Addition game.

- updateLeaderboardSubtraction()
  Updates the leaderboard with the new winner for the Subtraction game.
- updateLeaderboardMultiplication()
  Updates the leaderboard with the new winner for the Multiplication game.
- updateLeaderboardDivision()
  Updates the leaderboard with the new winner for the Division game.
- printLeaderboard()
  prints all four of the leaderboard contents on the screen

Homepage

Detailed Implementation:

The homepage will be the first page of the website. This will feature the rules to the game. The user can navigate to the different pages of the web page through the navigation bar that will be present on top of the page. The navigation will contain links to the games, the login page, and the leaderboard. Furthermore, the user will be introduced to the rules of the games through this page.

Data Members:

(Local React State)

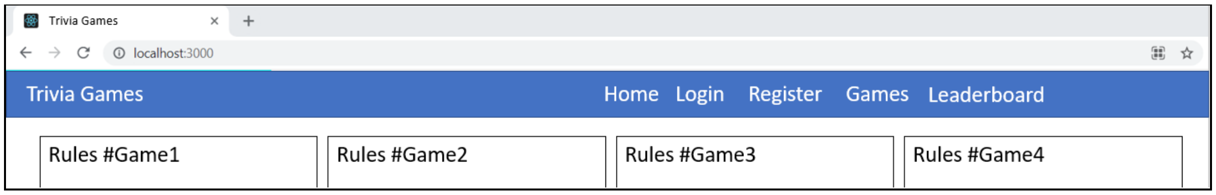
- $game_links (map<Link Text, URL> (string,string))

(Local React Components)

- NavBar

Navigation Bar

UI Design (size not to scale)



Detailed Implementation:

The users will first encounter the navigation bar at the home page. Clicking on the links on this bar will be the only method of exploring the different pages. Hence, the bar will be present throughout the website.

Data Members:

(Local React State)

- $nav_links (map<Link Text, URL> (string,string))
- $is_logged_in (boolean)

Data Methods:

N/A