Final Project - Advanced Programming and Web Development

Student: Elad Jacobovitz (305285041)

Email: elad.jacobovitz@post.idc.ac.il

Name of the app: "The RGB color game"

1. **Overview:**

   The RGB Color Game is a simple project developed using JavaScript, CSS, and HTML. This project is an interesting color guessing game. The user/player has to guess the result of the given RGB color combination and find the correct output of the color mixture. The user can guess the color until the option finishes.

2. **So how can you play?**

   Dirst, go to this link: **https://the-color-game-nu.vercel.app**
   than go to the server directory and open the terminal and run "node ./index.js".
   After you finish to upload the sever you can start.

   Go to the game menu and press "let's play". then you'll see 3 different levels on the mid bar (easy, hard and very hard).

   Pick your level and guess which number is represented by the RGB number on the top.

   If you know what color it is. Simply press on the square. Don't worry if you don't succeed in guessing, you will have as many chances until the squares are over.

   Want to start over? just press on the button "new color".

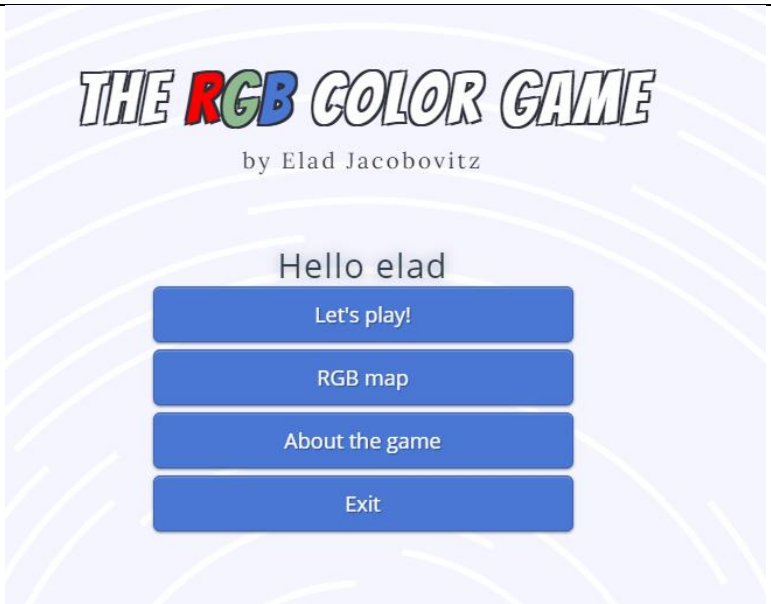
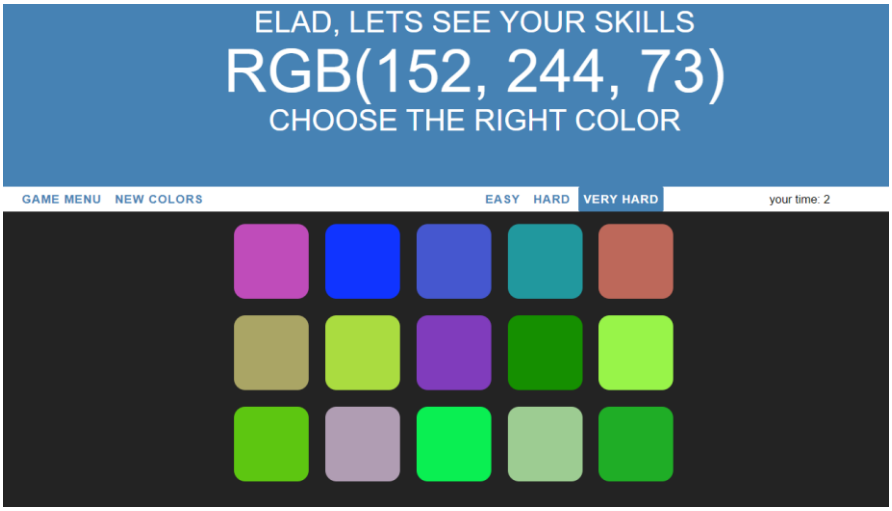
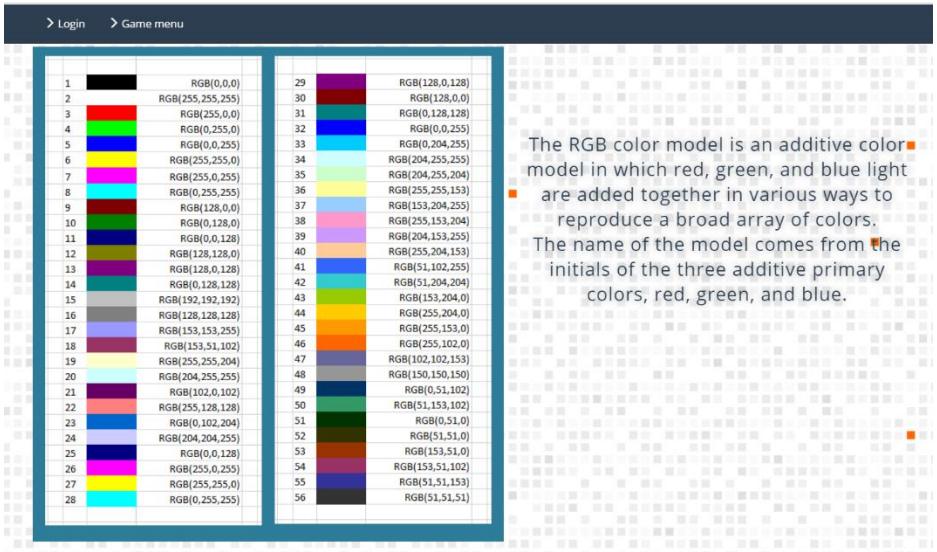
   Got it right the first time? Wanna try it again? just press on the button "new color" or "play again".

   In what ways can you win? If you spend less time guessing and make fewer guesses, your score will be higher!

   Are you curious about your ranking? Go to the home page (game menu -> exit) and then click on the "records" tab.

3. flow chart:

Home page - "home.js" Here you can login to the game with your username and password. you can move to the registration page or to the records page as well.	
Registration page - "registration.js" Here you can register to the app by inserting your details. you can move to the login page or to the records page as well.	
Records page - "registration.js" Here you can find out who has the best results (by difficulty). Results are ordered by best performance (time and clicks - the fewer the better). you can move to the registration page or to the login page as well.	

Game menu page - "pregame.js" You can choose to play a new game, read about the game, and then exit the game. You can only access this page if you are a registered user.	
Game page - "colorgame.js" From here, you can select a level and play the game. Throughout the game, there is a timer that indicates how many seconds you have, and you can choose to either start at the beginning or go to the menu.	
The RGB explanation page - "RGBmap.js" From here, you can read about the RGB map model.	

The about page - "about.js"
Here you'll find an explanation about the game.

4. Function table:

Client-side functions

Function name	Parameters	Returns	What it does
`goto_pregame ()`	-	-	take the user from the login page to the menu of the game (the pregame page).
`goto_colorgame ()`	-	-	take the user from the pregame page (the menu) to the game zone. Only here, it will call the queryselectors and the eventlisteners (only here the page is on and it can perform).
`goto_records ()`	-	-	take the user from the login page/ the registration page to the records page ("top players ranking").
`insertToTableOfRecords (recordObj, row, table)`	**recordObj** - {type, numberOfsquers, user, record, numOfClicks, time, date} **row** **table**	-	helper function - help to load the 3 tables of the records and insert every new row to the tables ("easy", "hard", "very hard").

`OnNavigate (pathname)`	**Pathname –** the path of the page after the server_url	-	Every time that we navigate to different pages in our app, it will run.
`updateName ()`	-	-	Extracts the user name (getElementById) and adds the name to the header.
`window.onload=function()`	-	-	Make sure that the user won't get to the game without login or register. If accidentally it happens, it will take the user back to the home page (login page).
`get_in(userObj)`	**userObj** - {_id, date, type, userE, userN, userP, EasyBestRecord:{}, MediumBestRecord:{}, HardBestRecord:{} }	-	In the case of a match between the username and password, it will take the user to the game menu ('pregame').
`check_login (user, pass)`	**username** **pass**	-	Send a get request for all users in the database and check if there is a match so the user can login to the application.
`send_login_details ()`	-	-	Take the username and password from the login form and send them to the checking process.

`send_registration_details (path)`	**Pathname –** the path of the page after the server_url	-	Register a new user to the application with all the details below:
			{
			type of Jason, date of registration, user Email address, username, user password, records ({easy}, {medium}, {hard})
			}
			if the username and email doesn't exist already in the database, it will send a post request to the server.

`postData (url, data)`	**url –** the post url **data –** the data it send	{Promise<Response>}	post a request to the server.
`getData(url)`	**url –** the get url **data –** the data it send	{Promise<Response>}	Ask for a get request from the server

`send_forgetPassword_request()`	-	-	Send to the database a post with the user name from a type of "forget password"
Color game functions			
`startGameTimer()`	-	-	Starts the timer in the game mode (colorgame)
`play_game()`	-	-	start new color game by running these functions: clearInterval, startGameTimer, setUpModeButtons, setUpSquares, reset
`setUpModeButtons()`	-	-	In the colorgame page (the game mode), set up the mode buttons of the levels.
`setUpSquares()`	-	-	In the colorgame page (the game mode), set up the color squares (15, 10 or 5 squares depending the difficulty)
`updateUser()`	-	-	When a new record is taken, the system will send a post request to update the user object in the database.
`updateRecord(data)`	**Data -** {type, numberOfsquers, userN, record: clicks + sec, numOfClicks, time, date}	-	When a new record is taken, the system will send a post request to add another object from a type of 'new record'.
`reset()`	-	-	Reset the game (new colors, timer, buttons).
`changeColors(color)`	**color -** array of squares (colors)	-	Color the squares according to the parameter it receives.
`pickColor()`	-	A color from the arrays of colors.	random a color and returns it.
`generateRandomColors(num)`	**num -** a random RGB type of color	Array of squares of colors).	Generate and push a random color to the array of the squares (it could be 5, 10 or 15 - according the difficulty).
`randomColor()`	-	String of an RGB color type For example: "(240,56,128)"	Create and return a random RGB type of color. (0 - 255, 0 - 255, 0 - 255)

Server-side functions

Function name	Parameters	type	What it does
`addUser: async (item) =>`	**Item** – user object to add to the database	insert	Check if exist, and if not, insert new user object to the database.
`addItem: (item) =>`	**Item** – any kind	insert	Insert any kind of an object to the database.
`getAllItems: (callback) =>`	**Callback**	Find	Return to the user all the data that exist in the database and convert it to an array of objects.
`updateItem: (oldQuery, newQuery) =>`	**Oldquery** – what should we look for in the database. **Newquery** – what we want to update in the object	updateOne	Search in the database an object that has the filter (oldquery) and update it with the field that inside the newquery.
`getRecordsItems: (callback) =>`	**Callback**	Find	Return to the user all the records data (objects) that exist in the database and convert it to an array of objects.