CS108 Project

23B1060 [Madhav R Babu]

April 2024

Contents

1 Introduction **2**

2 How to Run the Website? **4**

3 Achieving Basic Tasks **5**
3.1 Login Page 5
3.1.1 validateForm() 5
3.1.2 Login Page Styling 5
3.2 Forgot Password Page 5
3.2.1 checkLogin() 6
3.2.2 Forgot Password Page Styling 6
3.3 Personal Data and Preference Input Page 6
3.3.1 findMatch() 6
3.3.2 Personal Data and Preferences Input Page Styling 7
3.4 Match Page 7
3.4.1 foundMatch() 7
3.4.2 Match Page Styling 7
3.5 Scroll or Swipe Page 7
3.5.1 loadStudentsData() & showStudentDetails() 7
3.5.2 Scroll or Swipe Page Styling 8

4 Customisation **8**
4.1 New User Registration 8
4.1.1 /register endpoint 8
4.2 Filtering Option 8
4.3 Appending the User to student.json 9
4.4 /studentjsonreg 9

5 Miscellaneous **9**

1 Introduction

As part of the CS108 project, I chose the problem statement to create a dating website under the guidance of UG TA Kavya Gupta. Mark distribution for the project was divided into achieving basic tasks, customization, documenting the code, and writing a latex report for the project.
Basic tasks include:

- A login page that validates the user ID and password based on a pre-existing JSON file
- A forgot password page where the user can get their password by answering the secret question associated with their user-id correctly
- An input interface where the user can enter their details and find a match according to the details they entered
- A scrolling/swipe page where the user may scroll through the existing user's profile given in a JSON file
- An output interface that gives the right match to the user based on some algorithm to find the perfect match

Basic tasks mentioned in the problem statement can be achieved by HTML and JavaScript, and the interface can be made attractive with the help of CSS.
Apart from all basic tasks, I've added customization some of which required node.js for creating, and editing the JSON files.
Customizations include:

- A register new user page which creates a new element in the login.json file
- Redirecting previous users directly to their match
- A filter option for each user to get a match according to their sexual orientation and an option to add their preferred age group
- Adding the details of the user to the student.json so that the next user might get all previous users as potential matches
- Dynamic styling based on the match a person gets (according to their sexual orientation) to make the website interface more inclusive.
- A mail your match button that auto generates a customized mail based on the interests and hobbies of the user and their match

The website also has attractive styling based on the common theme of movies. I've included some romantic movies as background and styled the page inspired by their color gradients.All pages uses the font family Verdana.
The entire code is heavily documented (including the CSS) , and posses good indentation along with appropriate naming conventions for variables so that

every part of it is self-explanatory for anyone going through the code. This latex report gives detailed information about how to run the code, each HTML, JavaScript, and CSS file used for creating the website, and the use of node.js for handling the back-end data modification wherever required. I've created a private GitHub repository (which will be made public once the deadline is over) for version control purposes.

2 How to Run the Website?

Make sure all the files required for the website are in the same directory and install node js and express. run the command 'node server.js' in the terminal to launch the website which can accessed using this link in any browser of your choice. It will automatically redirect you to login.html
The website performs all basic tasks based on the JSON files similar to those provided along with the problem statement, however for using some customization feature a modified version of login.json that keeps tracks of whether a user have already visited the site has been submitted along with the files. I've used the name student.json and login.json to refer to both these version throughout the code hence you might have to move the unmodified version of these files from the directory before using the modified version to prevent name conflicts. The 'mail your match' functionality may not work if your device is not connected to internet.

3 Achieving Basic Tasks

Basic tasks are achieved using only HTML, CSS, and JS. I've divided the basic tasks into the HTML pages titled 'login.html', 'forgot.html', 'dating.html', 'scroll_or_swipe.html', and 'match.html', along with their corresponding stylesheets, and a dating.js containing a javascript function. Using these a user can login with valid credentials, retrieve their password in case they forgot their passwords, enter their details, scroll/swipe through all profiles, and find a suitable match.
Details of implementing these functionalities are added below:

3.1 Login Page

The login page is automatically loaded once the localhost:3000 is launched. It asks for the user's user id and password and once the Login button is clicked it calls the js function validateForm(). The page also features links to register a new account page and forgot password page.

3.1.1 validateForm()

It takes the user's username and password by accessing them with their respective ID. Then it fetches the login.json file and iterates through all iterables in it and changes the value of the variable $isValidUser$ if $data[i].username = usernameANDdata[i].password = password$ is found to be true. If the JSON contain the property match the function recognises it to be the modified version required for customization else it redirects valid users to dating.html without any url parameters. In case of customized files it checks if the user has already used the site and received a match, if yes it sends them directly to the match page as the student.json will automatically contain their json data, along with their student.json index and their match's index as url parameters. In case the username and password are found to be incorrect it gives a alert message to the user.

3.1.2 Login Page Styling

Login page uses a still from the movie 'La La Land' as background and use colors from the background throughout the page for style. It also includes a frozen glass effect for the login form. I've also added animation for the text 'Swipe, Match, Love'.

3.2 Forgot Password Page

On clicking forgot password in login.html it redirects to forgot.html. Input element in forgot.html asks the user for their username. On submitting it calls the javascript function checkLogin(). The page also has a link to go back to login.html.

3.2.1 checkLogin()

It accesses the username submitted by the user using the id. It fetches the login.json file and checks if the username exists if the username is invalid it creates a div inside the html page and sends invalid credentials message. If the username is valid it creates an element to ask the corresponding secret question and to retrieve the user's input for that question. If the input answer matches the one in JSON file user's password is displayed.

3.2.2 Forgot Password Page Styling

Styling is similar to login page. It uses the 'La La Land' background along with frozen glass feature for the input receiving section.

3.3 Personal Data and Preference Input Page

Once the user submits correct credentials the page is redirected to dating.html. The page asks the user to enter their personal details such as name, IITB roll number, year of study, gender, interests, hobbies, upload a photo, preferred gender, and preferred age group. On submitting the action attribute of the form redirects the data to the /studentjsonreg endpoint in server.js which is part of customization. It also calls the function findMatch() present in dating.js file. The page also contain links to scroll_or_swipe.html and login.html.

3.3.1 findMatch()

The function takes the user's roll number, interest, hobbies, age preference and gender preference from the form. It splits the string and set appropriate lower and upper bounds for the age preference as it received as a string. It then fetches the student.json data and on finding m common hobbies (it also penalises 0.05 if a certain hobby doesn't match to prevent an existing user in studnet.json who ticked all the hobbies from gaining an unfair advantage) for $student_i$ pushes m to hobby_count and it's i^{th} element, and similarly for interests. If $student_i$ falls under the preferred age group it pushes 1 to the i^{th} element of age_preference. At the end weight is calculated as

$$weight_i = 10 \times interest\_count_i + 5 \times hobby_count\_i + 15 \times age\_preference_i$$

if $student_i$ comes under the preferred gender else it is taken as 0. At the end the person with maximum weight is chosen as the match. In case of tie preference is given to the one under age preference, then to interest count, if tie still persists the person appearing first in student.json is given preference. The function then redirects to match.html with url parameters containing the index of the match and the user in student.json.

3.3.2 Personal Data and Preferences Input Page Styling

The page uses a still from the anime movie 'Kimi No Na Wa' as background. The input form has frozen glass effect and each individual input section is appropriately styled.

3.4 Match Page

Match page is loaded by the findMatch() function present in dating.js. A match div is present in body, in which the details of the match is dynamically added by the JavaScript functions. The page also contains a mail your match button as part of customization.

3.4.1 foundMatch()

On loading the page foundMatch() function is called with parameters index of user and match which is obtained from url parameters.
If the match's index is -1 it implies no match was found hence it displays an appropriate response inside the div match. If match is found the function fetches the json data of match using the index and creates an html element that contain details of the match and appends it to the div match. It also adds a mail you match button which allows you to mail your match. The function also checks for the gender of both match and user and identifies the type of match and hence dynamically changes the css of the page to display appropriate styling, by appending a suitable class to elements it creates.

3.4.2 Match Page Styling

The page three different background images, depending on the kind of match class appended. Background images have been chosen from the movies 'Brokeback Mountain', 'Past Lives', and 'Portrait of a Lady on Fire'. Styling for respective classes have been done separately for comparability with the background.

3.5 Scroll or Swipe Page

The page is reached on redirection from dating.html on clicking 'Go to Smash or Pass' The page contains the div container and student-list to receive dynamically created content from the javascript. The page displays profiles stored in student.json with Smash or Pass button on two sides. The next profile is loaded if the user clicks on any one of them.

3.5.1 loadStudentsData() & showStudentDetails()

On loading the page the details of the student with index zero in student.json is displayed and when the user clicks on either 'Smash' or 'Pass' button, next student's details are displayed.

3.5.2 Scroll or Swipe Page Styling

The background image is taken from the movie 'A Star is Born'. All elements inside the container are styled separately.

4 Customisation

4.1 New User Registration

New users can register their account by clicking on the 'Register a New User' link in the login page. It redirects to register.html which asks the user for their username, password, secret question, and answer to their secret question. The page is styled similar to login.html and uses 'La La Land' background along with frozen glass for the container. The form on submission calls the /register endpoint in server.js. On successful submission, a new element is appended to login.json.

4.1.1 /register endpoint

It listens for POST requests containing data such as username, password, secret_question, and secret_answer in the request body.

1. It begins by reading a file named login.json to check for existing users. If an error occurs during this file read operation, it logs the error and returns a 500 Internal Server Error response.
2. After successfully reading the file, it parses the JSON data to obtain an array of existing users. It then checks if the requested username already exists among these users. If a match is found, the code redirects the user back to the registration page (/register.html) with an error message indicating that the username already exists.
3. If the username is unique, a new user object (newUser) is created using the provided information, including the username, password, secret_question, secret_answer, and a default "match" property set to "none".
4. The new user object is then added to the existing array of users (users), and the updated user list is written back to the login.json file by overwriting the previous data.
5. If writing to the file is successful, a success message is logged to the console, and the user is redirected to the login page (/login.html).

4.2 Filtering Option

I've added an option for the user to choose their match according to their sexual orientation making the site more inclusive. The javascript function gives a match only if it aligns with the user's preference. The user's age group preference is also taken into account by giving it weightage in the matching algorithm.

4.3 Appending the User to student.json

On filling the form in dating.html the /studentjsonreg endpoint in server.js is called upon submitting the form. It adds the user's details in student.json to ensure new users can get all previous user's as potential matches.

4.4 /studentjsonreg

5 Miscellaneous