# Looking for a Date?

Ritwik Bavurupudi

April 2024

## 1 How to launch website?

This is the process to be done in Linux.

1. Open a terminal at the directory where your .html file is located. We have default terminal in Linux.
2. Then type the following:
   - If you have "python3" command then do:- python3 -m http.server
   - If you have "python" command then do:- python -m http.server
3. Then we get Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ... on terminal. Don't interrupt/close this command while you are viewing your .html file.
4. Go to a web browser and type:- http://localhost:8000/name of your html file
5. Now you can work on your .html file.
6. Close the command from step 2 (by Ctrl+C) after you are done viewing your files.

## 2 Introduction

This documentation provides an overview of my code written in HTML,CSS,JavaScript in order of the files which we proceed in website. It also contains some algorithms used by me to match the details of students to date.

## 3 Basic Tasks

### 3.1 Login Page

The provided code constitutes a login page designed for user authentication. It consists of a login.html file having script in html file itself and styling is done

in style.css file. Information about students username,password,secret question, secret answer can be given to login.json file.

The HTML structure defines the layout of the login page. The elements are encapsulated with appropriate tags on of which is ¡fieldset¿. This tag ensures that the content will be in a box and can be positioned easily and have a good accessibility.

In CSS, the background image spans the entire view-port, while form elements are styled with rounded borders and subtle shadows for a modern look. The login functionality is implemented using JavaScript to validate user credentials against data stored in a JSON file (login.json).Upon submission, the script fetches the JSON data asynchronously, iterates through user records to find a match, and redirects authenticated users to a designated landing page (dating.html). Additionally, error handling is incorporated to notify users of incorrect passwords through an alert mechanism, ensuring a seamless login experience.

### 3.2 "Forgot Password?" Button

The provided code constitutes a "Forgot Password" page designed to assist users in recovering their passwords.

The HTML structure defines the layout of the forgot password page, featuring input fields for entering the username and secret answer, along with buttons for submitting the username, checking the secret answer, and returning to the login page.

The JavaScript functions enable the validation of the entered username and secret answer against data stored in a JSON file (login.json). Upon submission, the script fetches the JSON data asynchronously, checks for the existence of the username, and retrieves the corresponding secret question. Subsequently, it compares the provided secret answer with the stored answer to verify the user's identity. If the verification is successful, the user's password is displayed, allowing them to regain access to their account.

```
<div id="qnbox" style="display: none;">
    <b><p id="secretqn"></p></b>
        <label class="label" for="secretAnswer">Your Answer:</label>
        <input class="input" type="text" id="secretAnswer" name="secretAnswer"><br><br>
        <button type="button" class="s2" onclick="checkAnswer()">Submit</button><br>
    </div>
    <div id="result" style="display: none;">
        <b><p id="password"></p></b>
    </div>
```

The above code contains the div elements which are not visible until some condition takes place.The use of error alerts ensures prompt feedback in case of incorrect inputs, while the option to return to the login page enhances user navigation and convenience.

### 3.3 Input Interface

The provided code constitutes a "Fill Details...Find your Match!!!" page designed to facilitate users in finding potential matches based on specified criteria.

```
function findMatch() {
    var UsName = document.getElementById('name').value;
    localStorage.setItem('Name',UsName);

    const inputGender = getGender();
    const inputInterestsArray = inputInterests();
    const inputHobbiesArray = inputHobbies();

    fetch('students.json')
        .then(function(response) {
            return response.json();
        })
        .then(function(studentsData){
            const matchScores = {};
            var userfound = false;

            for (var i = 0; i < studentsData.length; i++) {
                const student = studentsData[i];

                if (
                    (inputGender == 'Male' && student.Gender == 'Female') ||
                    (inputGender == 'Female' && student.Gender == 'Male') ||
                    (inputGender == 'Others' && student.Gender == 'Others')
                ) {
                    userfound = true;
                    const matchScore = calculateMatchScore(student, inputInterestsArray, inp
                    matchScores[student.Name] = matchScore;
                }
            }
             if (userfound == false) {
                var result = "NotFound";
                var storedname = localStorage.setItem('name',result);
             }
             else{
```

```
            const sortedStudents = Object.entries(matchScores).sort(function(a, b) {
                return b[1] - a[1];
            })

            const topMatch = sortedStudents[0];
            const topMatchName = topMatch[0];
            const topMatchScore = topMatch[1];
            if(topMatchScore>0){
            var result = topMatchName;
            }
            else{
                var result = "NotFound";
            }

            var storedname = localStorage.setItem('name',result);
        }
        })

        .catch(function(error) {
            console.error('Error fetching data:', error);
        });

    setTimeout(function(){window.location.href='match.html';}
    ,2000);
}
```

The above code is my algorithm for finding the perfect match among the students in students.json file. First, I calculated score by giving 10 points when each interest or hobby matches with respective interest or hobby. Then, I gave 9 points to the specific Interest and hobby as shown in below matrix where first element is interest and followed by hobby.

```
        ['Travelling', 'Photography'],
        ['Sports', 'Playing'],
        ['Movies', 'Watching YouTube/Instagram'],
        ['Music', 'Playing Musical Instruments'],
        ['Literature', 'Reading'],
        ['Technology', 'Coding'],
        ['Art', 'Painting']
```

Now, I used a function to calculate score of respective students with specific gender and finally calculate the person with maximum score to display name. If, all of them no match at all or if there are no genders matched, then it is redirected to a page which displays $Matchnotfound$. If there is match, upon submission, the script fetches student data from a JSON file (students.json) and

iterates through the dataset to identify potential matches an displays the best match.

### 3.4 Scrolling/Swiping

The provided code creates a scroll/swipe interface for viewing profiles. It fetches data from a JSON file containing student profiles and dynamically generates profile cards with details such as name, age, gender, interests, hobbies, and a photo. Users can interact with each profile card and like it, with a counter keeping track of the total likes. It contains CSS and JS files inline which are written in scrollorswipe.html.

The HTML structure consists of a container (profilesContainer) where profile cards are appended dynamically. Each profile card is composed of a div element with two child elements: a section for the profile photo and an aside for the profile details. Additionally, a like button (likeButton) is appended to each profile card.

```
fetch('students.json')
            .then(function(response) {
                return response.json();
            })
            .then(function(data) {
                const profilesContainer = document.getElementById('profilesContainer');
                 data.forEach(function(student) {
                    const profileDiv = document.createElement('div');
                    profileDiv.classList.add('profile');

                    const aside = document.createElement('aside');
                    aside.classList.add('aside');

                    const name = document.createElement('h2');
                    name.textContent = student.Name;

                    aside.appendChild(name);

                    profileDiv.appendChild(aside);

                     profilesContainer.appendChild(profileDiv);
                });
            })
            .catch(function(error) {
                console.error('Error fetching data:', error);
            });
```

Above code is the main part in this file. It first parses the data from students.json and iterates over it. Then each value is assigned to certain elements. The elements are been created and appended to the web page at the instant when it iterates over profiles.

It also contains an arrow at the bottom of the file which on clicking we reach to the top of page. Here, I added a like button which on clicking gets one like to the person. But, I was unable to store that in a file because it requires NodeJs to make it work properly.

### 3.5 Output interface of "right match"

This code extracts the best match name from the local storage of the server. Then, it iterates over the students.json file to find the name of the extracted data to know whether it exists. Once it is found, we get the display of the details of the student using JS where we append data.

The profile box also contains a button which shows $Wannatryagain?$ to go back to dating.html page.

When the data isn't matched, I gave "notfound" to local storage which is not found in the json file which in turn returns error page.

## 4 Customization

### 4.1 UI

The user interface looks attractive with its background image and profile containers with back ground colour. It also has hover for buttons where button gets highlighted.

### 4.2 Like Button

There is like button which on clicking gets one like to the person. But, I was unable to store that in a file because it requires NodeJs to make it work properly.