Indian Institute of Technology
Bombay

CS108 Project

Looking For A Date?

By:
Mradul Sonkar

Teaching Assistant:
Kavya Gupta

Instructor: Prof. Kameswari Chebrolu

May 14, 2024

Contents

1 Introduction

1.1 Background

The concept of dating has changed dramatically in the digital age, thanks to the introduction of online dating sites. As part of our course project, I attempted to create a dating website that would provide users with a simple and quick way to interact with potential mates. This report serves as a documentation of my efforts in conceptualizing, designing, and implementing this dating website.

1.2 Objectives

The primary objective of this project was to create a user-friendly dating website that facilitates meaningful connections between individuals based on their preferences, interests, and compatibility. Specifically, our goals were to:

- Develop a matchmaking algorithm that suggests potential matches based on user-provided information.
- Design an attractive user interface that enhances the browsing and interaction experience.
- Test the functionality and usability of the website to ensure a seamless user experience.

1.3 Importance

This project serves as a practical application of the theoretical concepts learned throughout our course. It enables us to gain hands-on experience in web development, database management, algorithmic design, and user interface design, thereby enhancing our skills and competencies in the field of software engineering.

2 Project Overview

2.1 Technologies Used

I developed the dating website using front-end technologies - *HTML, CSS, JavaScript* for basic tasks and back-end technologies - *NodeJs* for customisation tasks.

2.2 Description of Files

My dating website project consists of several files, each serving a specific purpose in the functionality and presentation of the website. Below is an overview of the main files and their roles:

- **index.html:** This file serves as the homepage of our dating website. It provides an introduction to the platform and includes navigation links to other sections of the website.
- **dating.html:** The "dating.html" file contains the interface for the matchmaking feature of the website. Users are prompted to answer a series of questions about their interests, preferences, and personality traits. Based on their responses, the matchmaking algorithm suggests potential matches from the website's database (student.json).
- **scroll_or_swipe.html:**This file displays a page where users can view profiles of other members. It provides a snapshot of each user's profile information, allowing users to filter people based on their interest.
- **login.html:** This file includes the login form for existing users to access their accounts. Users are prompted to enter their email address and password to authenticate and gain access to their profile dashboard.
- **signup.html:** The "signup.html" file contains the signup form where new users can create an account on the dating website. Users are required to provide information such as username, password, a secret question and answer to the secret question. This page is part of customisation and uses NodeJs.
- Other than these files we have styling files (.css) and script files (.js) to make the website fully working.

3 Working of the Code

3.1 Login Page

The login page of the dating website serves as the gateway for registered users to access their accounts. Below is an overview of the code implemented for the login functionality:

3.1.1 HTML Structure (login.html):

The login page is structured using HTML markup, comprising input fields for email address and password, along with a submit button to initiate the login process.

Instead of making a new file - forgot.html, I have tried to keep the user on the same page, that is, I have asked the secret question and shown the password on successful answering of that question on the same page.

3.1.2 JavaScript (script.js):

- Within the JavaScript file, I have defined a function (login()) responsible for handling the login process.
- This function is triggered when the user submits the login form.
- It retrieves the username and password entered by the user from the input fields.
- The function then makes an asynchronous request to fetch the user data from the "student.json" file
- Upon successful retrieval of the JSON data, the function iterates through each object in the JSON array to compare the username and password provided by the user with the corresponding values in the student.json
- If the user clicks on the "Forget Password" button, he/she is prompted to answer the secret question. On successful answering of that question, the password is shown to the user

3.2 The Matching Algorithm

Algorithm 1: Algorithm for Calculating Compatibility Score

Data: user details from data, JSON data
Result: highest scoring student stored in local storage
Extract user details from the input data;
for *each element in the JSON data* **do**
  Extract age, interests, hobbies, and gender of the element;
  Create sets from interests and hobbies;
  Calculate the union and intersection of user interests with the current element's interests;
  Calculate the union and intersection of user hobbies with the current element's hobbies;
  if *the gender of the current element is different from gender_of_user* **then**
    Calculate *age_factor*: $1 - \frac{\text{absolute difference between ages}}{\text{maximum of ages}}$;
    Calculate *interest_factor*: $\frac{\text{size of intersection of interests}}{\text{size of union of interests}}$;
    Calculate *hobbies_factor*: $\frac{\text{size of intersection of hobbies}}{\text{size of union of hobbies}}$;
    Calculate compatibility score: $0.3 \times age\_factor + 0.3 \times interest\_factor + 0.4 \times hobbies\_factor$;
    Update max_comp_score and max_comp_score_rollno if the current score is higher;
  end
end
Find the student with the highest compatibility score;
Store the details of the highest scoring student, user email, and user name in the local storage;
Call the function to open the output page with the stored data;

4 Customisations

4.1 Sign Up Page

The signup page of our dating website allows new users to register for an account by providing their personal details. Upon submission of the signup form, the entered information is sent to a Node.js server, which processes the data and appends it to the "login.json" file, thereby creating a new user account.

4.1.1 HTML Structure (signup.html)

The signup page includes input fields for users to enter their username, password, secret question and answer to that secret question. The data entered by the user is packaged into a JSON object and sent to the Node.js server using an HTTP POST request.

4.1.2 Node.js Server (server.js)

The Node.js server receives the POST request containing the user data. It parses the JSON payload and appends the user details to the "login.json" file, effectively creating a new user account.

4.2 Filter in Profile page

The filter functionality implemented on the profile page enhances the user experience by allowing individuals to narrow down the displayed profiles according to their preferences for gender and year of study. I created the function named applyFilter() which does the following tasks:

- The function retrieves the selected genders and years of study from the checkboxes on the filter screen.
- If no filters are selected, default values are applied to ensure a comprehensive display of profiles. For example, if no gender filters are selected, profiles for all genders (Male, Female, and Others) are included. Similarly, if no year of study filters are selected, profiles for all years (1st, 2nd, 3rd, and 4th) are included.
- The function then fetches data from the "student.json" file, which contains information about the profiles to be displayed.
- For each profile in the data, the function checks if the profile matches the selected filter criteria for gender and year of study. If the profile matches the selected criteria, a new HTML element is dynamically created to display the profile information, including the name, age, year of study, and interests. The dynamically created profile elements are appended to the page's content area.

4.3 Background Music Integration

The background music integration enhances the user experience by providing an auditory element to the webpage. This feature allows users to listen to background music while browsing the website, adding ambiance and entertainment value.

- An Audio object is created using the new Audio() constructor. The constructor takes the path to the audio file as an argument ("audio.mp3").
- The loop property of the Audio object is set to true, enabling the audio to loop continuously.
- A function named toggleMusic() is defined to toggle the playback state of the audio. When the function is called, it checks if the audio is currently playing. If the audio is playing, it pauses the playback using the pause() method. If the audio is paused or stopped, it starts the playback using the play() method.

- A clickable `<div>` element is created in the HTML with the id "musicPlayer".When this `<div>` is clicked, it triggers the toggleMusic() function, toggling the playback of the background music.

5 How to launch Website

5.1 Unzipping the Folder:

After downloading the website files, navigate to the directory where the zip file is located using your file manager or terminal. To unzip the folder via terminal, use the command:

unzip filename.zip

5.2 Backgrounds Directory:

Within the unzipped directory, you'll find a folder named "Backgrounds". Use the terminal to navigate to this directory using the `cd` command:

cd path/to/Backgrounds

This folder contains background images utilized across various pages of the website.

5.3 Images Directory:

Navigate to the "Images" folder located in the main directory using the terminal:

cd path/to/Images

Inside this folder, you'll discover a collection of images used for decorative purposes, icons, or other visual elements on the website.

5.4 Photos Directory:

Open the "photos" folder within the main directory:

cd path/to/photos

Here, you'll encounter individual profile photos associated with user profiles or team members featured on the website. For adding more users, add their profile pictures in this folder only.

5.5 Running Node.js for Sign-up:

Open a terminal window. Navigate to the directory where you've extracted the website files using the `cd` command:

cd path/to/website

Before proceeding, ensure that Node.js is installed on your system. If not, you can download and install it using the following commands:

sudo apt install nodejs

Replace these commands with the appropriate commands for your Linux distribution if you're not using Ubuntu or a Debian-based system. Once Node.js is installed, start the Node.js server by executing the appropriate command:

node server.js

With the server running, the sign-up page will become operational, allowing users to register and interact with the website website.

5.6 Precautions

- All the .html, .css , .js and .json files must be in same parent directory.
- All background images must be in "Backgrounds" folder.
- All images used at different places in website must be in "Images" folder.
- All profile pictures must be in "photos" folder.

6 Conclusion

In conclusion, the dating website project, which incorporates functions like signup, login, profile viewing, and matchmaking, is the pinnacle of web building expertise. With a focus on data security and user experience, the project offers a strong platform for future improvements.