

Dating Website Report

Shyam Sri Vardhan Varri

April 28, 2024

1 Introduction

This report outlines the design and functionality of a dating website aimed at helping users find their perfect match.

Note

I have written the steps to launch this website at the very end of this report.

2 Requirements

2.1 Languages

The development of the dating website will require proficiency in the following languages and technologies:

- **HTML/CSS:** For building the front-end user interface and styling.
- **JavaScript:** For implementing client-side interactivity and dynamic features.
- **Node.js:** For building the back-end server and handling server-side logic.
- **Express.js:** A Node.js web application framework for routing and middleware.

Note that both the **EJS** and **Nodejs** work together.

3 Functionality

The dating website will function as follows(includes both the basic task and customisations):

1. **User Registration:** Users will sign up for the website by providing basic information and creating a profile.

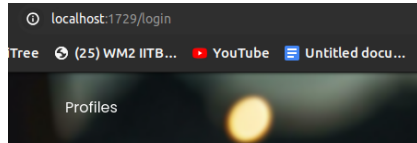
2. **Profile Creation:** Users will fill out their profiles with details about themselves, such as their interests and hobbies.
3. **Matching:** The website's matching algorithm will analyze user profiles and suggest potential matches based on compatibility.
4. **Communication:** Matched users will be able to communicate with each other through the website's messaging system.
5. **Finding a Perfect Match:** The ultimate goal of the website is to help users find their perfect match and form meaningful connections.

4 Implementation Details

The customizations were implemented using Node.js for the server-side logic and EJS (Embedded JavaScript) for server-side templating. The following details highlight key aspects of the implementation:

4.1 Node.js Integration

Node.js was utilized to develop the server-side components of the project. It provided a better runtime environment for executing JavaScript code on the server, allowing for efficient handling of HTTP requests and responses.



(a) Local Host



(b) Code snippet

Figure 1

4.2 EJS Templating

EJS was employed for server-side templating, enabling dynamic generation of HTML content based on data passed from the server. This facilitated the creation of dynamic web pages with reusable components and layouts.

4.3 Sorting Functionality

A custom sorting function was implemented to arrange user profiles based on rating and gender classification. The sorting algorithm arranges profiles in descending order(due to insufficient time, only descending order is embedded),

```

1 const express=require('express');//to create server
2 const fs=require('fs');//to read and write files
3
4 const app=express();//creating server
5
6 app.use(express.urlencoded({ extended: true }));//to parse the data from the form
7 app.set('view-engine','ejs');//setting view engine to ejs
8 app.use(express.static('public'))//to use the public folder
9

```

Figure 2: Ejs

prioritizing profiles with higher ratings and ensuring gender classification alignment as per user preferences.

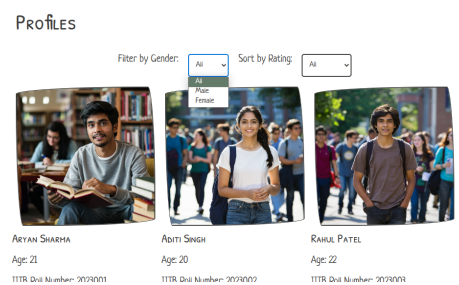


Figure 3: filter

5 User Authentication and Registration

In this section, we discuss the processes involved in user authentication, forgot password functionality, and user registration for the dating website project.

5.1 Login Authentication

The login authentication process ensures that only registered users with valid credentials can access the website. Here's how it works:

1. The user navigates to the login page of the website.
2. The user enters their username and password.
3. The system validates the user's credentials against the database.
4. If the credentials are correct, the user is granted access to their account and redirected to their dashboard. Otherwise, an error message is displayed.

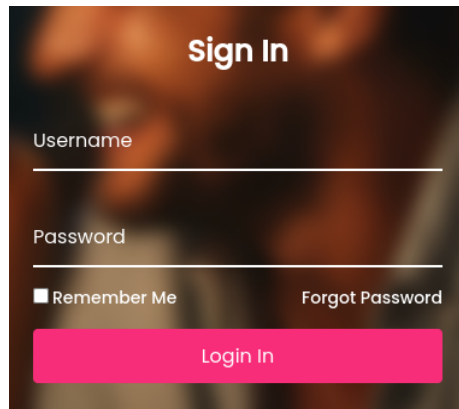
A login form titled "Sign In" with a dark background. It contains two input fields: "Username" and "Password". Below the password field is a checkbox labeled "Remember Me" and a link labeled "Forgot Password". At the bottom is a large pink button labeled "Login In".

Figure 4: Login

5.2 Forgot Password

The forgot password functionality allows users to reset their passwords if they forget them. Here's the procedure:

1. The user clicks on the "Forgot Password" link on the login page.
2. The system prompts the user to enter their username.
3. The system retrieves a secret question from the database and sends it to the html page and renders on the page.
4. The user enters the secret answer of that question.
5. The system validates the answer and prints the password for the corresponding username.

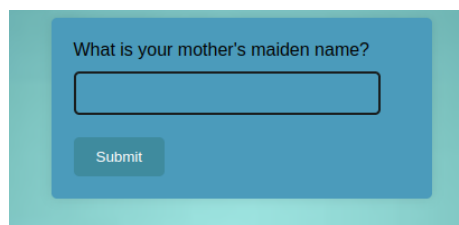
A forgot password form with a light blue background. It displays the question "What is your mother's maiden name?" above a text input field. Below the input field is a button labeled "Submit".

Figure 5: Forgot-password

5.3 User Registration

The user registration process allows new users to create accounts on the dating website. Here's how it works:

1. The user navigates to the registration page of the website.
2. The user fills out the registration form, providing details such as their username, password, secret question, secret answer.
3. The system validates the user's input, ensuring that all required fields are filled and username is unique.
4. If the validation is successful, the user's account is created, and they are logged in automatically. Otherwise, error messages are displayed to guide the user in correcting their input.

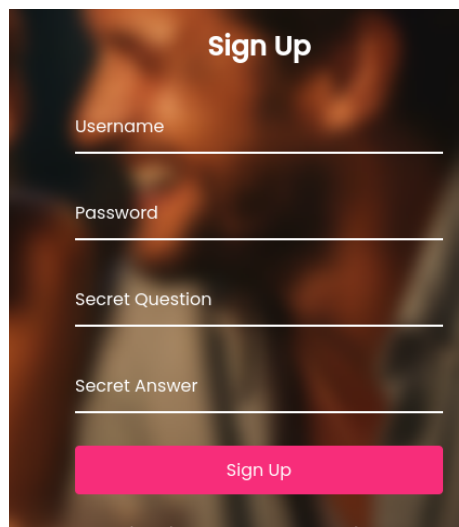
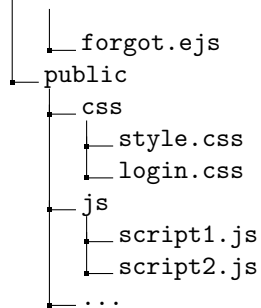


Figure 6: SignUp

6 File Structure

The following is the file structure of the project folder:

```
/ (Root Directory)
├── server.js
├── package.json
├── package-lock.json
├── node_modules
├── views
│   ├── dating.ejs
│   ├── profiles.ejs
│   └── login.ejs
```



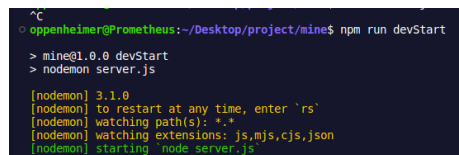
7 File Handling

To handle files in the project, follow these steps:

1. **Open the Zip File:** First, unzip the provided project file.
2. **Navigate to the Project Directory:** Open a terminal window and navigate to the project directory using the `cd` command.
3. **Start the Server:** Run the following command in the terminal to start the server:

```
npm run devStart
```

This command will start the server, and it will be accessible at `localhost:1729`.



```
^C
○ oppenheimer@Prometheus:~/Desktop/project/mine$ npm run devStart
> mine@1.0.0 devStart
> nodemon server.js

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
```

Figure 7: server

4. **Replace students.json and login.json:** Locate the `students.json` and `login.json` files in the project directory. Replace these files with the real data files you want to use.
5. **Update File References:** Check for all locations where the `login.json` and `students.json` files are used in your project. Update these references to point to the new files you replaced in the previous step.

Make sure to save all changes and restart the server (**this will be automatically done as we are using an special extension devStart**). Now you can login and explore the website!!