Looking for a date ?

Important Update (9 Apr) → Customisations (5 marks) of everyone will be ***relatively graded*** to some extent. This means that the more your customisation stands out or the better you make it, the better is the chance of scoring higher "Customisation" marks than others.
Update (20 Apr) → Submission format has been updated.

This project is intended to create a simple dating website using HTML, CSS, JavaScript. Broadly, you will be creating an interface, using HTML, CSS, where you can enter some of your personal details and **"find"** the "*right match*" for you among a set of students, using JavaScript!

Certain details of those students, among whom you have to find the "*right match*" , will be imported into the JavaScript file using a **.json** file, called **students.json**. This file will contain an array of dictionaries (like Python dictionaries), each dictionary having all the required details about a particular student. These **and only these** fields will be present in each dictionary of **students.json**:- "IITB Roll Number", "Name", "Year of Study", "Age", "Gender", "Interests", "Hobbies", "Email" and "Photo".

- The only possible options for the "Gender" field are going to be "Male", "Female" and "Other". Only a single option can be chosen for this field.
- For the "Interests" field, the only possible options are "Traveling", "Sports", "Movies", "Music", "Literature", "Technology", "Fashion" and "Art". One or more options can be chosen for this field.
- For the "Hobbies" field, the only possible options are "Reading", "Cooking", "Coding", "Gardening", "Painting", "Watching YouTube/Instagram", "Playing musical instruments" and "Photography". One or more options can be chosen for this field.
- For simplicity, you can assume that in students.json, all fields will be non-empty, with valid inputs. For "Photo", a valid file path to the photo will be given as input.

A **sample students.json and login.json** have been given here, along with sample photos. Similar templates of these .json files will be followed during evaluation (of Basic Tasks).
Do whatever you want to do in customisations but this JSON format is the input format for profiles for Basic Tasks.

Important Note:- ~~We will use a "local server" to view our .html files. This is to avoid issues fetching content from .json files. Read this doc to see how to open your .html in a local server. This is also how your submission will be evaluated during Viva..~~
If you feel the need to use Node Js, you can use it and you can show the working of your script using "node script.js". This applies for both Basic Tasks and Customisations. But **the method of execution of script for Basic Tasks and Customisations should be the same**, (i.e. you can't choose to use "python3 -m http.server" for Basic Tasks AND "node script.js" for Customisations, that won't be possible anyways. These two methods are completely different ways of execution.)
You can still use the method of this doc to open your .html file, if you are not using Node Js.
Using Node Js **isn't compulsory** and rest assured, the "Basic Tasks" are doable without the use of Node Js.

Basic Tasks:

You have to achieve the following **basic** tasks (i.e. the minimal tasks to be done for this project):-

1) **Login Page:-** To access the input interface, there should be a login screen before that, which will require you to enter username and password of some sort. If you are "registered" and you fill correct entries, then you login, otherwise give appropriate error messages. Any existing/registered user must have their username and password in login.json. Hence any registered user can access this website (on the same machine) at a much later time in future. You can write the code for the login page in a .html file named **login.html**.
    - You don't need to create a feature (as a Basic Task) to allow a new user to register to the website (i.e. appending a new username, password to login.json).
2) **"Forgot Password?" Button**:- Provide some kind of a password recovery system if a previously registered user forgets their password. A secret question-answer phrase will be there for every username-password in login.json. Whenever a person has to recover a password, they will click some "Forgot Password?" button in the login page. Then a new page (or in login.html itself) opens up where they will provide their username as input and then the secret question corresponding to the username should be asked. If the person enters the correct answer to the question then print the password on the screen otherwise print appropriate error message.
    - Do this in **forgot.html**. But in case you want to integrate it in login.html itself, that's also allowed. Just mention it in the report.pdf.
3) **Input Interface:-** Using HTML, CSS, create a decent input interface with proper labels and input boxes for a person to fill in his/her/their personal details. Those details should correspond to the fields in students.json, i.e. the person's "IITB Roll Number", "Name", "Year of Study", "Age", "Gender", "Interests", "Hobbies", "Email" and "Photo".
    - For "Gender", create a set of radio buttons to choose a single option.
    - For "Interests" and "Hobbies", create a set of checkboxes to choose multiple options for each.
    - There should be some sort of a "Submit" button which on clicking should **"find"** the person's "*right match*".
    - Provide a "Logout" button as well, that takes back to the login screen.
    - Do this work in files named "dating.html" and "style.css". You can assume that the person will always give valid non-empty responses.
    - Instead of radio buttons/checkboxes, drop-down lists work too.
4) **Scrolling/Swiping:-** Provide a feature in the input interface (dating.html) to allow the person to scroll/swipe through all the students (their details and photo) present in students.json file. The person clicks on some "Scroll/Swipe" button in the input interface and a new page opens up where you can freely scroll/swipe through profiles. Do this in **scroll_or_swipe.html**.
5) **Finding the "right match":-** Clicking "Submit" should also **find** the person's "*right match*" among the profiles present in students.json. The "*right match*" of a person should have **significant (not necessarily the maximum) intersections in "Interests" and/or "Hobbies" with that person**. Alongside this, you can create any criteria (of your own

will) to decide ***a*** "*right match*" of a person. This will be achieved using JavaScript. Code this in "script.js" or Internal Js of .html (if another .js file is needed by you, it is allowed, just mention that in your report.pdf properly).

- Note that the "*right match*" of a person is not the person itself. *That's how dating works!*
- If you fail to find an apt "*right match*" (according to whatever criteria you use), then give an appropriate message in the output interface.
- If there are multiple matches (according to your criteria), then you should break ties arbitrarily and return only one match.
- The "*right match*" isn't necessarily fixed for any person. It may depend on how you are finding it.

6) **Output Interface:-** When the person clicks "Submit", a new tab will appear showing the "*right match*" for that person along with their details (if you succeed in finding the "*right match*" for that person). The structure for this interface will be done in a separate .html file named "match.html", but the same CSS and JavaScript files will be used, i.e. style.css and script.js
   - If you fail in finding the "*right match*", then do as stated in point (5).
   - You can display the "*right match*" in the same page as the input interface, just mention it clearly in the report.

Important Note:- ~~**For Basic Tasks, write code only in the files mentioned below and only in HTML, CSS and JavaScript. No new files or other programming languages expected.**~~ **For Basic Tasks, write code only in the files mentioned above.** ~~**No new files accepted.**~~ **No new files needed if NodeJs is not being used, new helper files are allowed, otherwise.. Node Js is accepted along with HTML, CSS and simple JavaScript, if required, but the need for it should be properly told in Viva if asked. If more .js files are needed, you can use it.**

Customisations:

Apart from the Basic Tasks, adding some extra features (like 2-3 atleast or more if possible) of **your own creativity** is required as a part of this project. The customisations are completely open to your own imagination, but here are some example ideas of customisations for this project **(**You can pick some/all of them, however **these are just suggestions and are not mandatory to be chosen as your customisations)**→

1) You can make the UI more interactive, like adding sound, animations and/or graphics. Be creative for such a customisation.
2) **Login but with Style!:-** Using just a simple username-password pair to login is sorta boring and less secure. So you can come up with an interactive, more secure way to login. For example:- You can provide a "pattern unlock" as in mobile phones or a digital click keypad where you enter your pin by clicking some digits on the screen or any other innovative idea you can think of.
3) **Likes/Like Meter**:- You can add a feature where any logged in user can like/rate any profile while "Scrolling/Swiping" through all of them, in turn some "Like Meter" should be there for a profile which tells/shows how much that profile is liked by everyone.

4) **Filters**:- Add filters like selecting users which have certain “Interests”, “Hobbies”, “Year of study” and/or “Gender” etc, or any sort of creative filter you can imagine. If a person applies those filters then the “*right match*” should be chosen from those selected people only and “Scrolling/Swiping” should also show only their profiles.
5) **Mailing the “*right match*”:-** After the “*right match*” is decided, you can provide an option to contact the “*right match*” by mailing them an email from your mail ID. This customisation is much more complicated than the above ones. You can refer to this link or this link to find out a feasible solution for mailing using a .js file.

Important Note:- For Customisations also, the MAIN functioning should be done in the same files as Basic Tasks. New file(s) are allowed if needed as extra helper file(s) for presenting your customisation(s), but they should also be submitted. Those new files shouldn’t be assisting in functioning of the Basic Tasks in any manner.

Marks Distribution: (Max 15 Marks) (Final)

- Basic Tasks: (Max 8 marks)
  - Login Page + Forgot Password? + Input Interface + Scrolling/Swiping: 3 marks (This will be judged on how well it functions but **more** on how well the UI looks, how much CSS is put into it. Better the UI, better the marks awarded here)
  - Finding the “*right match*”: 3 marks
  - Output Interface: 2 marks (This will also be judged on how well it looks)
- Customisation(s): (Max 5 marks)
  - This will be judged by your creativity in it!
  - Read the “Important Update” in Page 1 as well for this.
- Overall code-quality and a LaTeX report: (Max 2 marks)
  - The LaTeX report should briefly describe your implementation of Basic Tasks but properly describes your Customisations! (Name it as “report.pdf”)
  - Make sure your code is heavily commented!
- Viva:
  - There will be **no separate marks** for Viva.
  - It will serve as an on-spot evaluation of whatever you will achieve in the above sections (i.e. should those marks *really* be awarded or not).

Submission Format (Final):

Submit all these files in submission.zip:-

- login.html, forgot.html, dating.html, scroll_or_swipe.html**,** style.css, script.js, match.html (if you’re using one) and other .css/.js files (which you may have used)
- Other files related with your customisations (any type of file)

Submit report.pdf separately.

Note that even if you submit students.json or login.json from your side, we will use our own version of those files to evaluate your submission **for the Basic Tasks.** For customisations, if you have a customisation that involves a different version of those files, then you should submit an example of those files, for us to check that customisation.

Demo:

This is a simple basic implementation of the input interface:-
(**You are not expected to replicate these**, these are just for demonstration purposes)

- A simple example for input interface, check this image here.
- A simple example for scrolling, check this image here.
- A simple demo video of how some of the basic "Basic Tasks" may look like. Refer to the video here. All of this has been done only using normal HTML, CSS and JavaScript.

Important notes:

- **Plagiarism/copying from one-another will lead to severe penalties.**
- For any further queries related to this project, please contact **Kavya Gupta**, via WhatsApp on 7753932449 number or via email at 22b1053@iitb.ac.in.
- This problem statement may or may not be updated in future. Hence **keep yourselves updated with the latest version of this document.**
- If you have any kind of feedback/doubt, please report to (me) Kavya Gupta!
- ~~**Modifying a .json file from a JavaScript file is tedious hence it is suggested not to look for methods which require changing the .json file from the browser. This applies for Basic Tasks only. For customisations, you can do whatever you like.**~~ **Modifying a .json file from a JavaScript file is tedious without Node Js hence it is suggested not to look for methods which require changing the .json file from the browser, in case you're not using Node Js.**
- Refer to the "Important Note" in Page 1 once without fail.
- The CSS (or UI) design should be **original** and not be made using built-in classes or external references.
- You don't have to make the page "responsive" as a Basic Task. Customisation you may.

References/Help:

- Sample .json files and photos provided. For link, refer to Page 1.
- If you want to get photos for more profiles for your testing, you can use AI-generated images from any website (ideogram.ai is one example).
- Using AI like ChatGPT for coding can be helpful for this project. Although use of AI for coding is not prohibited, you should know in-depth whatever you submit, for Viva purposes.
- Refer to the demo video I made. Link under "Demo" section. This shows that you don't need any PHP, NodeJs, or anything other than JavaScript for getting the "Basic Tasks".
- (15 Apr) If this helps, try googling/ChatGPTing about "localStorage" in JavaScript. It may be useful in Basic Tasks. Ignore if you are able to do Basic Tasks without it.