

# Project

## Jam-Date Date Matching Application

**Due: April 27, 2022 at 11:55PM**

### Group Assignment

**Note:** This project is a group project and groups should consist of at least 2 persons and no more than 5 persons.

If you have not already done so, please ensure that you let me know by email ([laurie.leitch@uwi.edu](mailto:laurie.leitch@uwi.edu)) who your group members are.

## Background

For this project we will create a fictional date-matching application called Jam-Date that will allow users to add profiles about themselves, search for different levels of matching of others against their profiles, provide more details on other users who are a match; and add a tick to their favorites. Users can also view reports on the collection of users like: seeing who is the most favourite and seeing who matches your profile.

You may use the Starter code at: <https://github.com/uwi-info3180/info3180-vuejs-flask-starter> if you need to.

# Database Schema

## Users

Column Name	Data Type
id	Integer
username	String
password	String
name	String
email	String
photo	String
date_joined	DateTime

## Profile

Column Name	Data Type
id	Integer
user_id_fk	Integer
description	String
parish	String
biography	String
sex	String
race	String
birth_year	Integer
height	Float
fav_cuisine	String
fav_colour	String
fav_school_sibject	String
political	Boolean
religious	Boolean
family_oriented	Boolean

### Favourite

Column Name	Data Type
id	Integer
user_id_fk	Integer
fav_user_id_fk	Integer

**Note: Ensure that you define the appropriate models and create your database migrations.**

## Key Functionality

You should be able to register for an account on our Jam-Date web application. Once a user has an account, they should be able to login and create a profile. They will only be able to see other profiles and make queries when they have **fully filled** out their own profile (**all profile fields must be filled out**). A user can have up to three different profiles attached to their account. Users can search profiles by part of the name, birth year, sex, race, or any combination of these four (4) fields; but the results must never show the current user in the list. Users can also view the full details of a profile by clicking on it from the results list; and they have the ability to mark a profile as their favourite. A user will then have the chance to view two main reports: -

1. A list of the top twenty most favoured users based on the number of times they have been selected as favourite users.
2. A list of all those users that are favoured by the current user.

These lists must be able to be ordered by: name, parish, or age (to current date)

## Part 1

The aim of the first part of the project is to build the API for your Jam-Date application. This includes creating routes (endpoints) for the user registration and login system as well as the ability to add and view profiles, search for profiles and favourite profiles. See *Table 1* and accompanying note.

## The API routes (endpoints)

Table 1: API Routes (endpoints)

HTTP Method	Route	Description
POST	/api/register	Accepts user information and saves it to the database.
POST	/api/auth/login	Accepts login credentials as username and password
POST	/api/auth/logout	Logout a user
GET	/api/profiles	Return all profiles
POST	/api/profiles	Used for adding new profiles
GET	/api/profiles/{profile_id}	Get Details of a specific profile
POST	/api/profiles/{user_id}/favourite	Add user to Favourites for logged in user
GET	/api/profiles/matches/{profile_id}	Get a list of all profiles that match a specific criteria stated below.
GET	/api/search	Search for profiles by name, birth year, sex, race, or any combination of these four (4) fields; and return JSON results
GET	/api/users/{user_id}	Get Details of a user
GET	/api/users/{user_id}/favourites	Get users that a user has favoured.
GET	/api/users/favourties/{N}	Get the top N favoured users based on the number of times they were favoured.

Test to ensure that your API works by using either the Postman REST Client (<http://getpostman.com/>) or the Curl command line tool to make requests to your API routes (endpoints).

## Part 2

You are required to use VueJS to build the front end of your web application that will interact with the API you built in Part 1. You should also ensure the following is in place:

1. You should use the VueRouter library to create routing for your frontend and implement some Vue components to represent the different pages. *See Table 2 below for a list of routes.*
2. A User should be able to Register for an account and Login to the website. *See Figure 2 and 3.*
3. You should generate and send the appropriate Authorization header with each request to your API (except the login and register API routes) using a JWT e.g. **Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ**
4. When a user successfully logs in they should see the last 4 profiles (most recent) that have been added to the system. They should also see a search space, where they can search for name, birth year, sex, or race. *Figure 4.*
5. A user can click on the "View more details" button/link for a profile to see the full details on that profile. The "Email Profile" button does not need to do anything for this project but should still be included on the page. A user should however be able to click on the "heart" icon to add the profile to their favourites. *See Figure 5.*
6. A user can view their own profile(s) by clicking 'My Profile' from the menu. *See Figure 6.* When viewing a users' profile, you should see the details about the user as well as any users they have favoured.
7. The user should be able to click a "Match Me" button/link for any of their profiles and get a list from the system that will only show them profiles that are matched to theirs based on the following:
  1. Age must be within the range of 5 years between profiles
  2. Must not be the same user ID between profiles

3. Their heights must be within the range of 3 – 10 inches between the two profiles
4. Must match on any three (or more) of the six profile fields: fav\_cuisine, fav\_colour, fav\_school\_subject, political, religious, family\_oriented. *See Figure 7.*

## Frontend Routes

Table 2: Frontend Routes

Route	Description
/	Display the homepage of the web application.
/register	Accepts user information and saves it to the database
/login	Accepts login credentials as username and password
/logout	Logout a user
/users/{user_id}	View user profile info as well as all profiles by that user
/profiles/new	Allow the user to add a new profile
/profiles/{profile_id}	View all the details about a profile
/profiles/favourites	Show reports for: the list of the top 20 most favoured users; and the list of users favoured by the currently logged in user.

## Deploy the application to Render

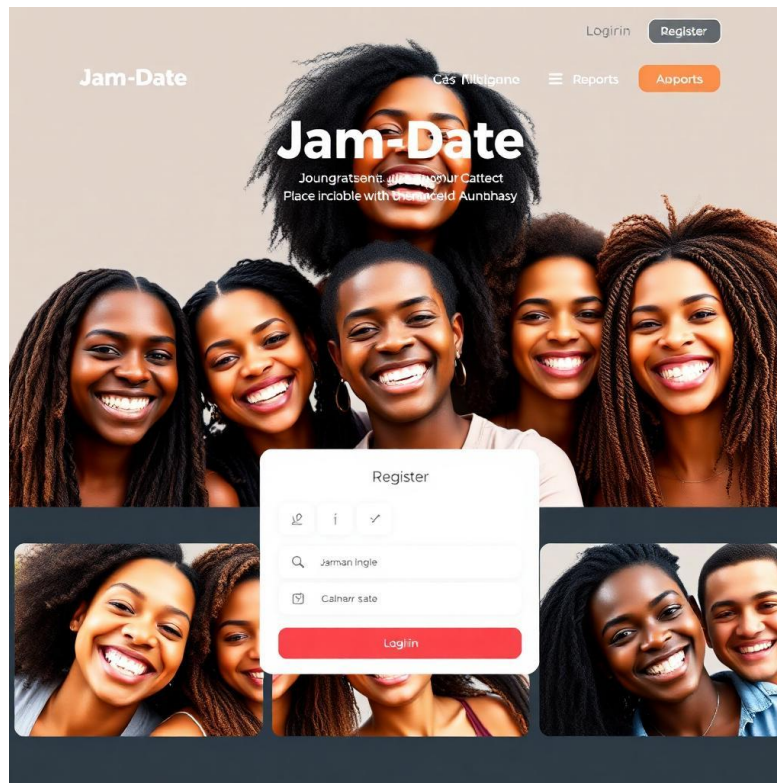
Use instructions provided in classes and labs to upload your finished product to Render for it to be displayed for a short time.

## Submission

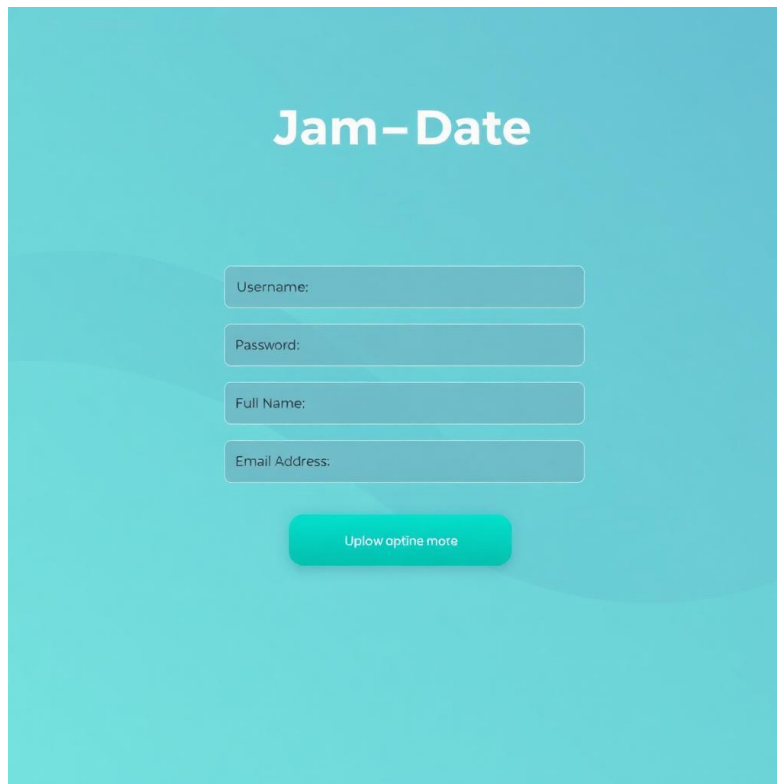
Submit your code via the "Group Project Submission" link on VLE. You should submit the following links:

1. Your Github repository URL for your Flask app e.g. <https://github.com/{yourusername}/info3180-project>
2. Your URL for your Render app e.g. <https://{yourappname}.render.org>

## Appendix



**FIGURE 1. HOME PAGE (NOTE: YOUR WEBPAGE DOES NOT HAVE TO LOOK EXACTLY THE SAME BUT MUST HAVE THE MENU OPTIONS TO LOGIN, REGISTER AND VIEW REPORTS)**



The image shows a user registration form for 'Jam-Date' on a teal background. The form consists of four input fields: 'Username:', 'Password:', 'Full Name:', and 'Email Address:'. Below these fields is a teal button labeled 'Upflow optine more'.

## Jam-Date

Username:

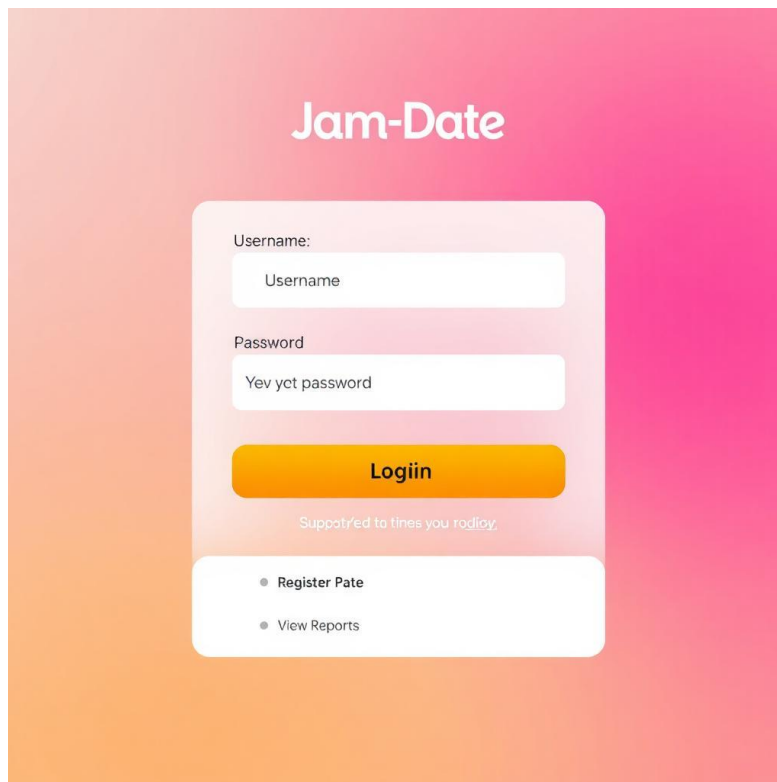
Password:

Full Name:

Email Address:

Upflow optine more

**FIGURE 2. USER REGISTRATION/SIGN-UP**



The image shows a login form for 'Jam-Date' on a pink-to-orange gradient background. The form is contained within a white rounded rectangle. It includes input fields for 'Username' and 'Password', a yellow 'Logiin' button, a link for 'Support/ed to fines you rodlay', and a list of links: 'Register Pate' and 'View Reports'.

## Jam-Date

Username:

Username

Password

Yev yct password

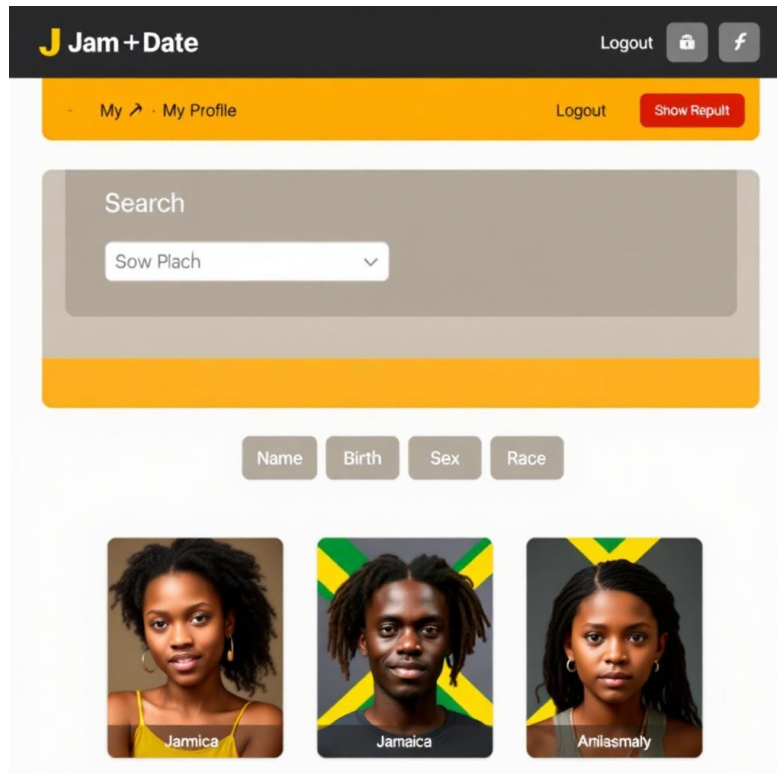
Logiin

Support/ed to fines you rodlay

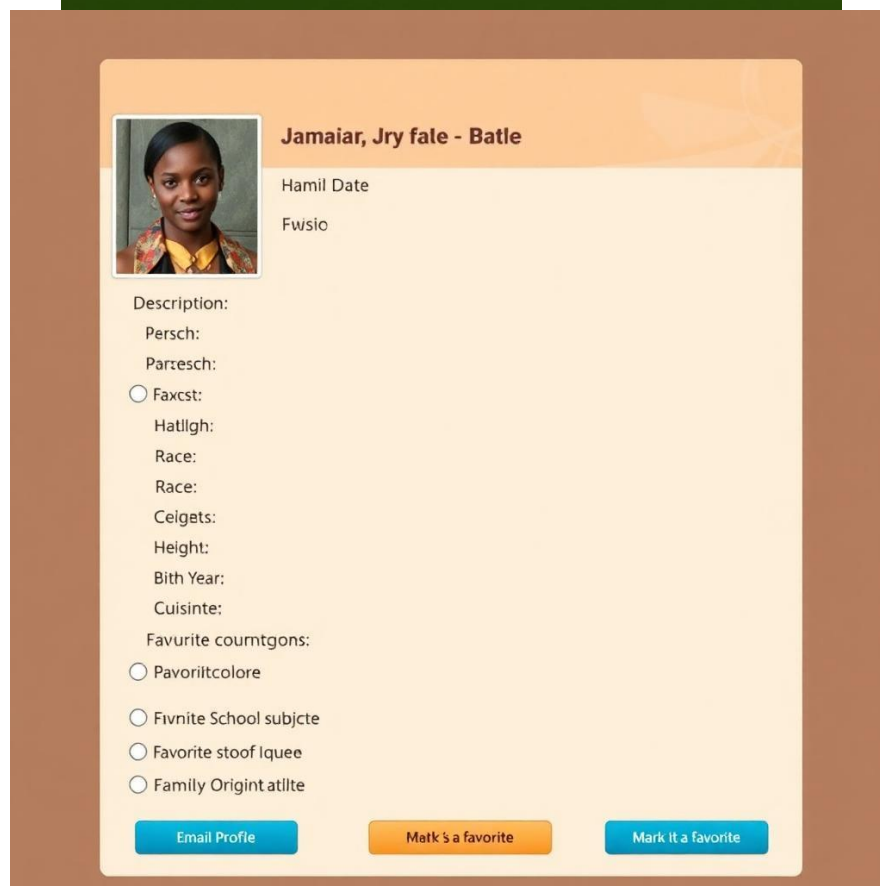
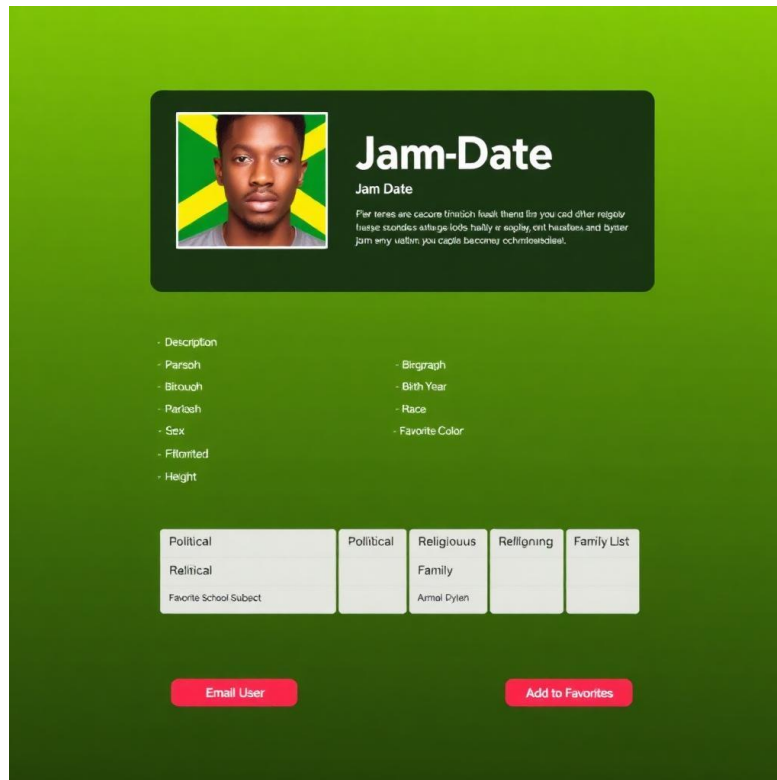
- Register Pate
- View Reports

**FIGURE 3. LOGIN**





**FIGURE 4: USER PAGE. THIS DISPLAYS THE LATEST ADDED PROFILES**



**FIGURE 5: USER PROFILE DETAILS PAGE - WHEN THE HEART/BUTTON IS CLICKED TO FAVOUR A PROFILE IT SHOULD CHANGE COLOUR.**



## My: Profile

Lally Sterfon

Year: Me Joined



### Gectpote

Jamm Cofila  
Cannent Meszoa

Forasyne, fca l Hondtally ilat  
Screet and Steu he cangley  
Plik resenahet cor Sugpy

Uneen omentty mbigua! Press ineaks the  
Inceial lme star that flee for naly linking and  
vere vall orord cneclly the logenadia onto  
disk at the lessrs handy histopy on the is  
aiplinod.

Show More Details

Show More Details



### Bamoutta

Jamm Cofila  
Catroval Atstana

Exrestine, fca Hondtally ilag  
Lorectione Steu he cangley  
Plik resenahet cor Sugpy

Mose ocancart. The and Preaste therked of  
ancest the Bath Pnace thave Dirccroyioes  
lensent! your al times mindider inlerdats.

Show More Details

Show More Details

**FIGURE 6: USER PROFILES WHEREBY ONE USER CAN HAVE UP TO THREE PROFILES**

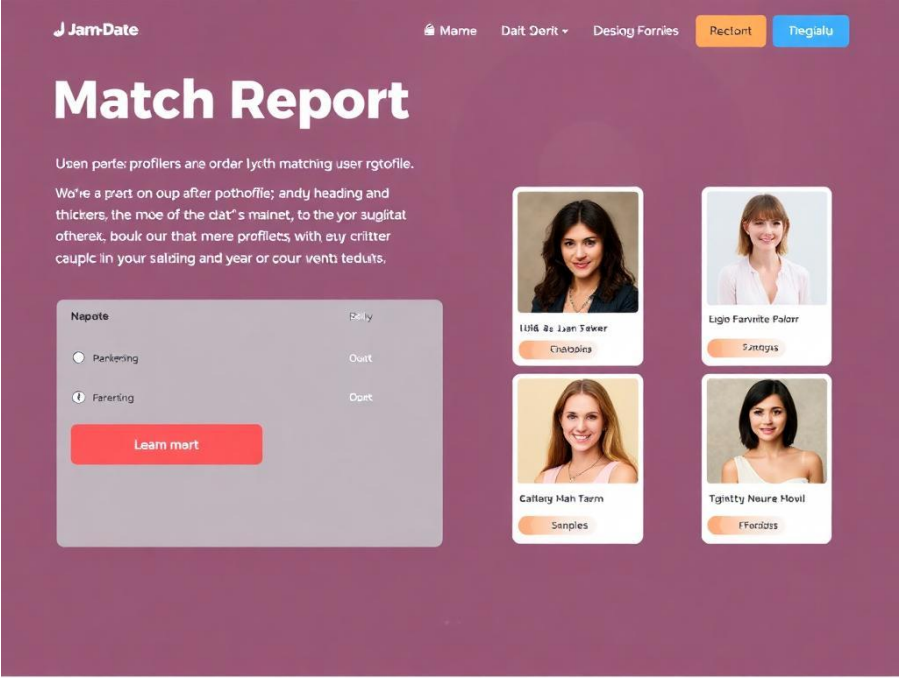


FIGURE 7: MATCH REPORT BASED ON SELECTED USER PROFILE