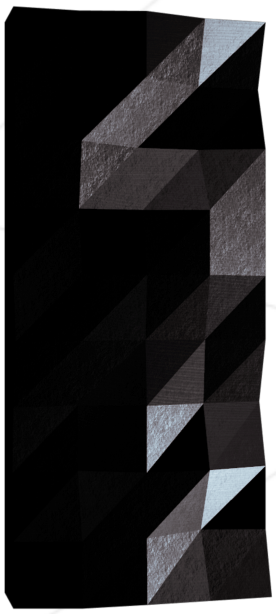

Web

Matcha

Summary: Because love, too, can be industrialized.

Version: 6.0

Contents

Chapter I

Foreword

This second millennium has forever changed and reinforced the habits and customs of the Internet. Choices are now driven by technology, leaving less and less room for chance. Human relationships, the foundation of any modern society, are increasingly formed artificially through dating site algorithms and social networks, connecting people based on highly specific criteria.

Yes, romanticism is dead, and Victor Hugo is probably rolling in his grave.

Chapter II

Introduction

This project aims to create a dating website.

You need to develop an application that facilitates connections between two potential partners, covering the entire process from registration to the final meeting.

Users should be able to register, log in, complete their profile, search for and view other users' profiles, and express interest in them with a "like"[1]. They should also be able to chat with those who have reciprocated their interest.

[1]Since "like" is not an ideal term, you are encouraged to find a more explicit alternative.

Chapter III

General Instructions

- Your application must be free of errors, warnings, or notices, both server-side and client-side.
- For this project, you are free to use any programming language of your choice.
- You may use micro-frameworks and any necessary libraries for this project.
- You are free to use UI libraries such as React, Angular, Vue, Bootstrap, Semantic, or any combination of them.
- No security vulnerabilities are allowed. You must address at least the mandatory security requirements, but we strongly encourage you to go beyond them—everything depends on it.
- We define a "micro-framework" as one that includes a router and possibly templating but does not include an ORM, validators, or a User Account Manager.[1] As long as you adhere to these constraints, you are free to use the tools of your choice.
- If you need inspiration, we suggest using the following languages as your primary choice:
    - Sinatra for Ruby.
    - Express for Node (yes, we consider this a micro-framework).
    - Flask for Python.
    - Scalatra for Scala.
    - Slim for PHP (Silex is not allowed due to its integration with Doctrine).
    - Nickel for Rust.
    - Goji for Golang.
    - Spark for Java.
    - Crow for C++.
- You should use a relational or graph-oriented database. The database should be a free one, such as MySQL, MariaDB, PostgreSQL, Cassandra, InfluxDB, Neo4j, etc. You must manually create your queries, like experienced developers do. However, if you're clever, you can build your own library to simplify query management.

[1]This definition will be authoritative during the defense, regardless of alternative definitions found online.

- For the evaluation of this project, your database must contain a minimum of 500 distinct profiles.
- You are free to choose the web server that best suits your needs, whether it is `Apache`, `Nginx`, or a `built-in web server`.
- Your entire application must be compatible with at least the latest versions of `Firefox` and `Chrome`.
- Your website must have a well-structured layout, including at least a header, a main section, and a footer.
- Your website must be mobile-friendly and maintain an acceptable layout on smaller screens.
- All forms must include proper validation, and the entire website must be secure. This is a mandatory requirement and will be extensively evaluated during the defense. To give you an idea, here are a few examples of security vulnerabilities that will not be tolerated:
  - Storing plain-text passwords in your database.
  - Allowing HTML or JavaScript injection in unprotected variables.
  - Permitting the upload of unauthorized content.
  - Allowing SQL injection attacks.

Chapter IV

Mandatory part

You must develop a web application with the following features:

IV.1 Registration and Signing-in

The app must allow a user to register by providing at least their email address, username, last name, first name, and a securely protected password. Commonly used English words should not be accepted as passwords.

After registration, the user must receive an email with a unique link to verify their account.

Users must be able to log in using their username and password. They should also have the option to request a password reset email if they forget their password. Additionally, users must be able to log out with a single click from any page on the site.

IV.2 User profile

- Once logged in, users must complete their profile by providing the following information:
    - Gender.
    - Sexual preferences.
    - A biography.
    - A list of interests using tags (e.g., #vegan, #geek, #piercing, etc.), which must be reusable
    - Up to 5 pictures, including one designated as the profile picture.
- Users must be able to modify this information at any time, as well as update their last name, first name, and email address.
- Users must be able to see who has viewed their profile.
- Users must also be able to see who has “liked” them.

- Each user must have a public “fame rating” [1].
- Users must be located via GPS positioning down to their neighborhood, with their explicit consent. If a user opts out of GPS location tracking, they must manually provide their approximate location (city or neighborhood) to use the matching features. This manual location entry is required for the application to function properly. [2] Users must also have the option to modify their location in their profile at any time.

IV.3 Browsing

Users must be able to easily access a list of suggested profiles that match their preferences.

- You should suggest “interesting” profiles. For example, a heterosexual woman should only see male profiles. You must also handle bisexuality. If a user has not specified their orientation, they should be considered bisexual by default.
- Matches must be intelligently determined[3] based on:
  - Proximity to the user’s geographical location.
  - The highest number of shared tags.
  - The highest “fame rating”.
- Priority should be given to users within the same geographical area.
- The list of suggested profiles must be sortable by age, location, “fame rating”, and common tags.
- Users must be able to filter the list based on age, location, “fame rating”, and common tags.

IV.4 Research

Users must be able to perform an advanced search by selecting one or more criteria, such as:

- A specific age range.
- A “fame rating” range.
- A location.
- One or multiple interest tags.

Similar to the suggested list, the search results must be sortable and filterable by age, location, “fame rating”, and interest tags.

[1] You are responsible for defining what “fame rating” means, as long as your criteria are consistent.

[2] Note: This approach respects GDPR requirements for explicit consent in data processing. While some dating websites may use alternative tracking methods, this project emphasizes privacy-compliant development practices.

[3] Take multiple criteria into account.

IV.5 Profile view

Users must be able to view other users' profiles.
Profiles should display all available information except for the email address and password.
When a user views a profile, it must be recorded in their visit history.

The user must also be able to:

- "Like" another user's profile picture. When two users mutually "like" each other's profiles, they will be considered "connected" and can start chatting. If the current user does not have a profile picture, they cannot perform this action.
- Remove a previously given "like". This will prevent further notifications from that user, and the chat function between them will be disabled.
- Check another user's "fame rating".
- See whether a user is currently online, and if not, view the date and time of their last connection.
- Report a user as a "fake account".
- Block a user. A blocked user will no longer appear in search results or generate notifications. Additionally, chatting with them will no longer be possible.

Users must clearly see if the profile they are viewing has "liked" them or if they are already "connected". They must also have the option to "unlike" or disconnect from that profile.

IV.6 Chat

When two users are connected[4], they must be able to "chat" in real-time.[5]
The implementation of the chat feature is up to you. However, users must be able to see, from any page, when they receive a new message.

IV.7 Notifications

Users must receive real-time notifications[6] for the following events:

- When they receive a "like".
- When their profile has been viewed.
- When they receive a message.
- When a user they "liked" also "likes" them back.
- When a connected user "unlikes" them.

[4]Meaning they have mutually "liked" each other.
[5]with a maximum delay of 10 seconds.
[6]With a maximum delay of 10 seconds.

Users must be able to see, from any page, when they have unread notifications.

⚠ For security reasons, all credentials, API keys, environment variables, etc., must be stored locally in a .env file and excluded from Git. Storing credentials publicly may result in project failure.

Chapter V

Bonus part

Below are possible bonus features you can implement to earn extra points:

- Add OmniAuth strategies for user authentication.
- Allow users to create a personal photo gallery with drag-and-drop upload and basic image editing (e.g., crop, rotate, apply filters).
- Develop an interactive map of users, requiring more precise GPS localization via JavaScript.
- Integrate video or audio chat for connected users.
- Implement a feature to schedule and organize real-life dates or events for matched users.

`The bonus part will only be assessed if the mandatory part is perfect. "Perfect" means that the mandatory features have been fully implemented and function without any malfunctions. If you have not met` **`ALL`** `of the mandatory requirements, your bonus features will not be evaluated.`

Chapter VI

Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Make sure to double-check the names of your folders and files to ensure they are correct.

VI.1 Peer-evaluation

- Your code must not produce any errors, warnings, or notices, either server-side or client-side (in the web console).
- Anything not explicitly authorized is strictly forbidden.
- Any security breach will result in a score of 0. At a minimum, you must implement the security measures outlined in the general instructions. This includes:
  - Ensuring that passwords are not stored in plain text in the database.
  - Protecting against SQL injection attacks.
  - Validating all form inputs and file uploads.