236369

Winter 2019-2020

Managing Data on the World-Wide Web

Final Project

Introduction

In your final project you will implement a **Traveling Social Network** web application.
Your app will include user management (enabling registration, login and profile management). Users may "follow" each other and search other users' profiles. In addition, users may post about their trips (i.e. **posts**) abroad and view their friends' posts in a "feed-like" page. Finally, users can also search other user's trips using a map.

We divided the project to two parts, both done in **pairs** –

1. In the first part you will implement the backbones (/base) of your application. The submission deadline for this part is **Wednesday, 1.1.2020, 23:55.** This part has weight of **10%** of your **final grade**.
2. The rest of your application, the submission deadline is **Sunday, 19.1.2020**, **12:00**, this part **(including the interviews)** has weight of **40%** of your **final grade**. Note the deadline is **12pm.**

Finally, on the week of **19-23 of January**, we will interview you about your project. In the interview we will ask you questions on the project as well as the relevant course material. The exact schedule and further information will be published in the coming weeks.

Technical Background

Your project will include a **Flask** backend and a **React** frontend, as well as REST communication as taught in class. You are free to use any extending-packages you want for both.
In addition you will use **PostgreSQL** for your database.

You should use **Anaconda** for managing your Python packages, and **npm** to manage your JavaScript packages.

You are provided with an Anaconda environment file, and a npm config file, containing the basic, necessary packages.

Part 1 Requirements

In the first part of the project you will decide your database design and create a simple login page. The goal is to familiarize with the different parts of the project, such as connecting your Flask backend to the database and the communication between frontend and backend.

Database Design

You are required to define the tables you will use in the rest of the project and write the corresponding ***models*** and ***relationships*** in your Flask application.

Login Page

In addition, you should implement a **login** page for users. After logging in the user should be redirected to a page which specifies some user-specific information.

The information regarding the **submission** of part 1 can be found below under the submissions section.

Project Guidelines

These are guidelines for the entire project, but they are **not** final. We will add **few small** requirements after the submission of Part1. Note that the general architecture and components will remain the **same**.

Homepage

The application should have a homepage which users will be redirected to when visiting.
For *anonymous* users (e.g. users which aren't logged in), the homepage should present the option to **register** as a new member or **login**.
If the user is already logged-in the page should present the main posts feed (see appropriate section).

Users and Profiles

Each user **must** have a username, password, and email, as well as other informative fields regarding the user (such as image, full name, etc.)
When logging-in and registering you must perform server-side validation of the fields and enforce the database constraints. In addition, *username* must be unique. You must perform user authentication and password security as learned in class. Finally, users may also delete their accounts.

Follow Relationship

Users should be able to follow each other. The follow relationship helps in deciding which posts to display, and effects user interaction. Note that this relationship is **not** symmetric (like follow in Instagram).

Profile Page

Each user should have a personal profile page with their information, a list of their followers, and users they are following. The user should also be able to edit their own profile and information. Users may view other users' profiles, depending on their privileges (as described at the last section).
The profile page is where a user decides to follow another user – when viewing someone's profile the user should have the option to follow that user.

Posts Feed

Users should be able to create **posts** regarding their travel plans. The post **must** include a location and start and end dates, but we recommend adding more information regarding the travel. Users can also edit and delete their posts.

The **post feed** consists of the posts that the user itself created and posts created by other users which the current user follows. It should also include the option to create new post.

Subscribing to Posts

Users can subscribe to other user's posts. The user will get notified if one of his subscribed posts is edited\deleted.
The user should also have the option to view these notifications and delete them.

Searching for Users

Users can search for another user's profile using a search bar. The search should be by username, allowing autocomplete and partial matchings. The search should redirect to the matching user's profile (if the user exists).

Searching for Travel Partners

Users can search for travel partners using a map and dates – you can choose a point on the map, start and end dates and a radius and search for all the travel posts in that area, at that time. The search should also allow autocomplete when searching for a place, and comfortable date choosing. The results should appear as markers on the map, with link to the post. In addition, all the posts of members that the user follows should also appear on the map.

User Privileges

The table below summarizes the actions each user "type" can perform:

Suppose *U* is a user on the site.

Action	Anonymous	Other User	Follower of *U*	*U*
View *U*'s basic info and posts			X	X
Edit *U*'s info and add posts				X
Search for members		X	X	X
View *U*'s travels on the map (using the map search)		X	X	X

Design and Styling

You have full control over the design of your application – but like in previous assignments, it needs to be user-friendly and coherent. You **must** implement some design i.e. you can't use bare HTML.

Handling Errors

First, your server should **never** crash nor stop working unless closed manually or your connection/computer crashes, doing so would be considered a grave error.

If an error happens, you should return a corresponding HTML page describing the error (without exposing information about the server), and the HTTP response should have a matching status code.

Technical Requirements

As said, you must use the specified frameworks – **Flask**, **React,** and **PostgreSQL** for your backend, frontend, and database respectively.

The communication between the frontend and the backend must be **REST**ful, including authentication and data transfer as learned in class.

Tips

Getting Started

- First, download all the libraries
- Try a basic example in both Flask and React to make sure you understand how to run them
- Make sure you can connect to the database

Part 1

- **Read** the whole assignment
- **Plan** your database models and relationships
- **Implement** the models in Flask
- **Implement** a basic **RESTful** example with React and Flask
- **Connect** everything together

General Tips

- Use the web! Someone probably implemented a lot of the features you are looking for.
- In class, we talked about the basic packages (for both React and Flask). Use them but note that there a lot more.

- Plan your workflow, classes, and data structures well – bad planning can make life much harder when implementing.
- We recommend you start working on the project as soon as possible.

General Links

- Anaconda - https://www.anaconda.com/distribution/
- npm - https://www.npmjs.com/
- PostgreSQL - https://www.postgresql.org/
- create-react-app - https://github.com/facebook/create-react-app
- Example for REST with Flask & React - https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

Submission

Part 1

You should submit a zip named ***project_part1.zip***, that contains the following files:

- **backend** – a folder containing all your backend files. We should be able to start the backend using:

  "cd backend"
  "python main.py"

- **frontend** – a folder containing all your frontend files. We should be able to start the frontend using:
  "cd frontend"
  "npm start"
- design.txt – a brief overview of your database design and why you made it the way it is. Please be brief and concise.

- Any other files you used that are required to run your product.

Complete Project

You should submit a zip named ***project.zip***, that contains the following files:

- **environment.yml** – the Python environment you used (using Anaconda).
- **package.json** – the JavaScript environment you used (using npm).
- **backend** – a folder containing all your backend files. We should be able to start the backend using:

  "cd backend"
  "python main.py"

- **frontend** – a folder containing all your frontend files. We should be able to start the frontend using:
  "cd frontend"
  "npm start"
- [Optional] – a file containing anything you'd like to highlight in your work. You can do so in words, pictures, graphs etc. Use this opportunity to show us what you've done exceptionally well.
- Any other files you used that are required to run your product.

Good Luck!