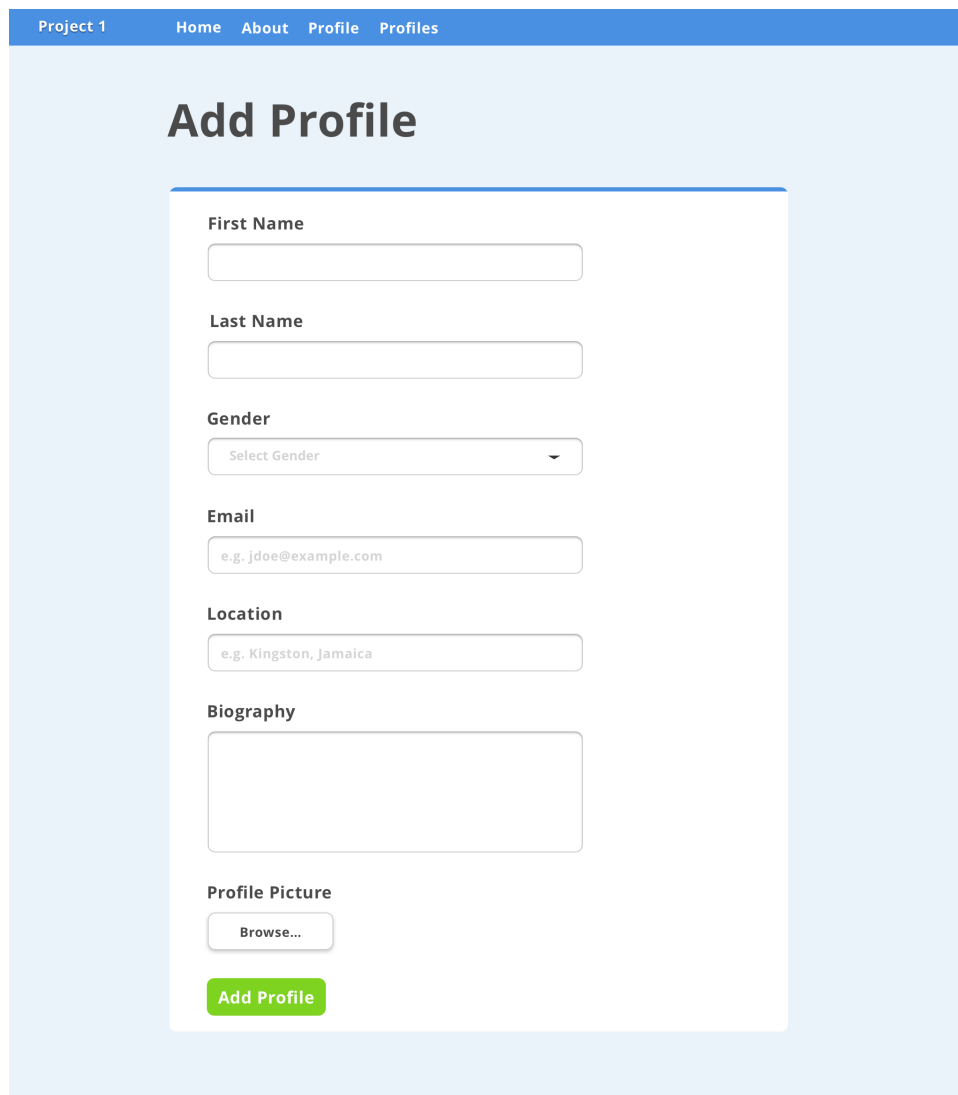


# INFO3180 Project 1 (30 marks)

Due: March 15, 2020 at 11:55 PM

At the end of this project you will have a Flask based application that can accept and display profile information. The profile information will be stored in a PostgreSQL database.



The screenshot shows a web application interface with a blue header bar containing the text "Project 1" and navigation links "Home", "About", "Profile", and "Profiles". The main content area has a light blue background and features a white form titled "Add Profile". The form contains the following fields: "First Name" (text input), "Last Name" (text input), "Gender" (dropdown menu with "Select Gender" as the placeholder), "Email" (text input with placeholder "e.g. jdoe@example.com"), "Location" (text input with placeholder "e.g. Kingston, Jamaica"), "Biography" (text area), "Profile Picture" (file upload button labeled "Browse..."), and a green "Add Profile" button at the bottom.

Figure 1: Add Profile Form

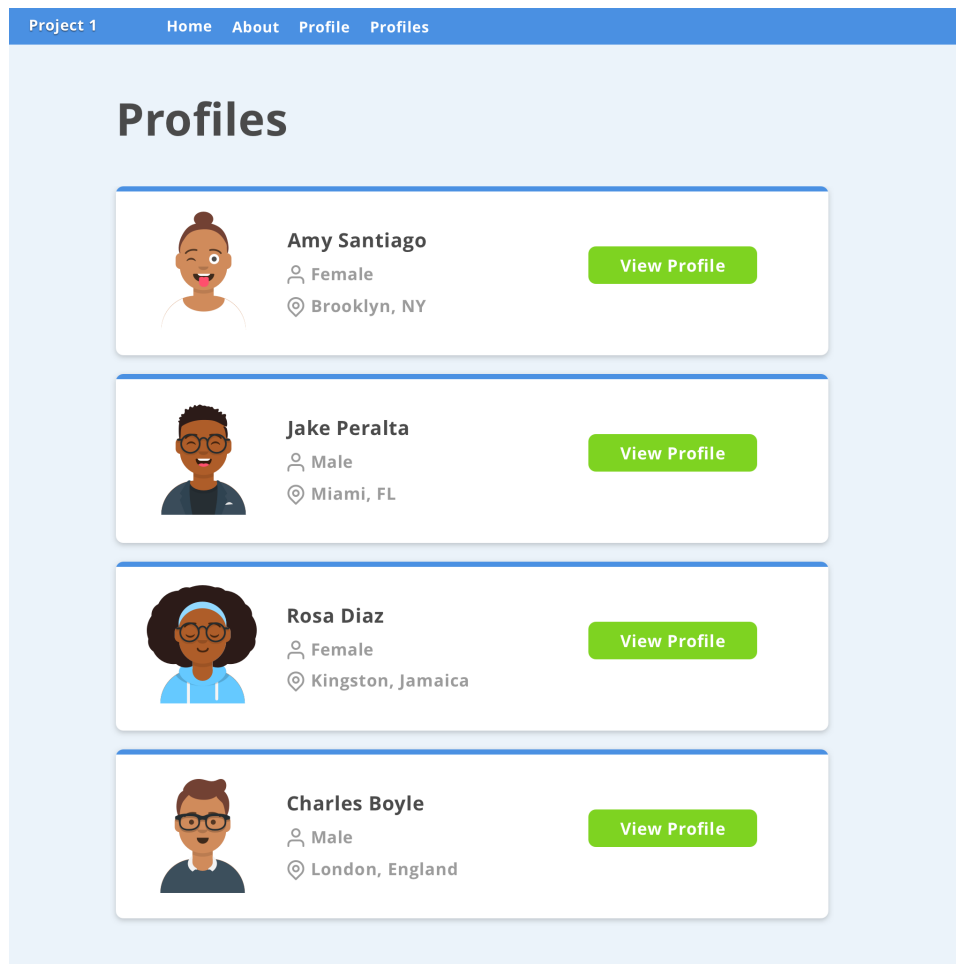


Figure 2: List of User Profiles

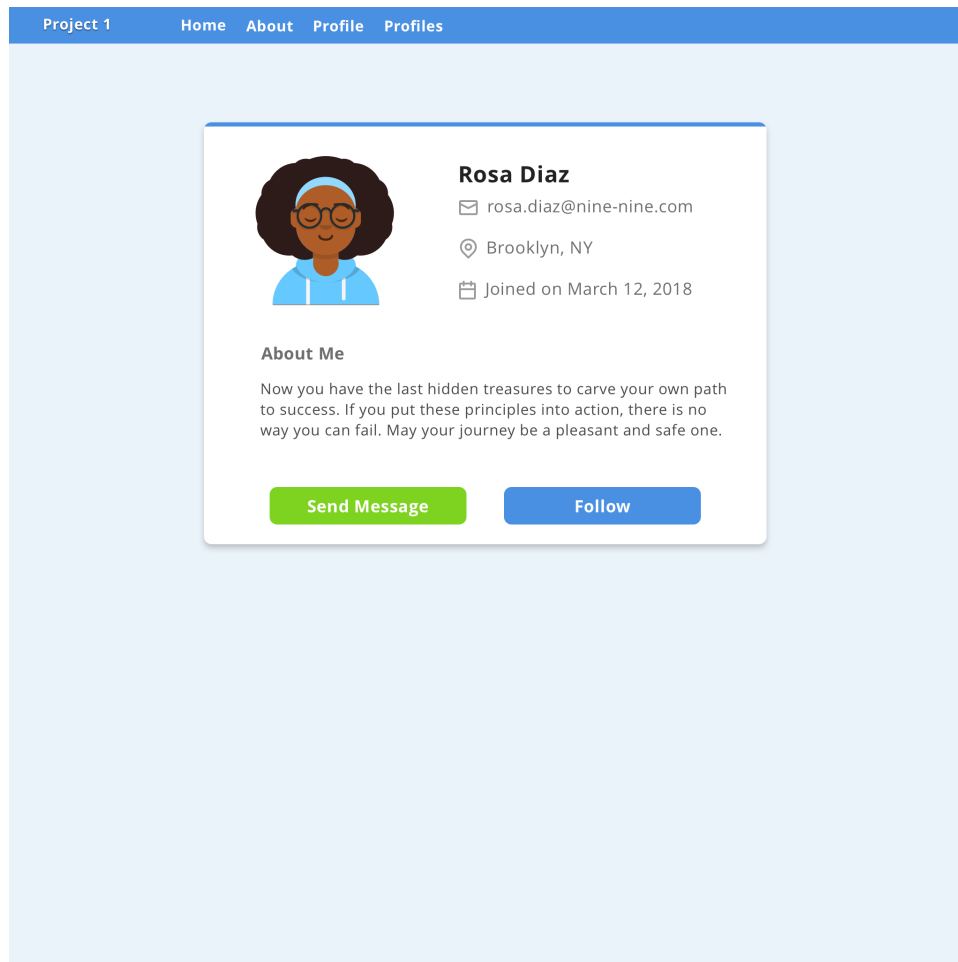


Figure 3: Individual User Profile

## Specifications

### Part 1

Create a Flask App that accepts input for a user to create a new user profile and also display both a list of users and an individual user profile.

The following routes will need to be created and appropriate templates rendered:

1. `"/profile"` For displaying the form to add a new profile. (See Figure 1)

2. `"/profiles"` For displaying a list of all user profiles in the database. (See Figure 2)
3. `"/profile/<userid>"` For viewing an individual user profile by the specific user's id. (See Figure 3)

The add new profile form must be created using Flask-WTF and contain the following fields:

1. Text fields for firstname, lastname, email, location.
2. Select (option) field for gender (whether Male or Female)
3. Textarea field for a short biography.
4. File upload field called photo which accepts the profile image

Upon submission, the form should make a POST request to the `"/profile"` route and validate the user input to prevent bad data. A unique userid should be generated (e.g. an auto incrementing id field in your model) and also the date the user was created\_on should be stored in the database. All of this input must be stored in a PostgreSQL database.

Once a profile is successfully added the user should be redirected to the `"/profiles"` route and a flash message should be displayed notifying the user that the profile was successfully added.

Note: You MUST generate a migration file for your UserProfile model so that the database can be recreated.

For the list of user profiles page, you are to display a list of all user profiles. Each profile should display the photo, name of the user, gender, location and a button/link that when clicked should carry you to the individual user profile where you can view more details. (See Figure 2).

On the Individual user profile page, you should have the user's photo, name, email, location and date they joined (ie. the date the account was created\_on), along with their short bio. You should also have two buttons, one to "send a message" and the other to "follow" the user (See Figure 3). Please note these buttons do not need to do anything for this project.

Note: You must also add navigation links to the "/profile" and "/profiles" routes to your header.html file.

## Part 2

The finished application should be deployed to Heroku. If you haven't already done so, ensure that you sign up for an account on the Heroku website (<https://heroku.com>) and then do the following.

Note: You will also need the Heroku CLI. This will need to be installed on Cloud9 and on your local machine if you haven't already done so. To install the Heroku CLI, see instructions at <https://devcenter.heroku.com/articles/heroku-cli> .

```
heroku login
heroku apps:create
git push heroku master
```

You will also need to ensure that you have a Database setup on Heroku for your application. To create and provision the Postgres database add-on and create a PostgreSQL database on Heroku, follow the instructions at the following link:

<https://devcenter.heroku.com/articles/heroku-postgresql#provisioning-the-add-on>

Note: You will be using the hobby-dev plan when creating your database.

```
heroku addons:create heroku-postgresql:hobby-dev  
heroku config -s
```

You should then see something like the following:

```
postgres://  
yecsapnzlttgcb:8860d3512549e3ebba04aa68ede74496e30041ff458ea1d4  
6d7c4ed7f5402af0@ec2-54-221-244-196.compute-1.amazonaws.com:543  
2/d9d6gbbcjoc71g
```

This is the URI that you will use in your `SQLALCHEMY_DATABASE_URI` config option in your Flask Application. Ensure that you change driver (at the start of the URI) from `postgres` to `postgresql` for Flask SQLAlchemy.

You will also need to ensure you make a modification to your Heroku Procfile. This will allow you to run the migration you created earlier (and any other future migrations) to create/update your Heroku Database. The updated Procfile will look similar to this:

```
release: python flask-migrate.py db upgrade --directory migrations  
web: gunicorn -w 4 -b "0.0.0.0:$PORT" app:app
```

## Submission

Submit your code via the "Project 1 Submission" link on OurVLE. You should submit the following links:

1. Your Github repository URL for your Flask app e.g. <https://github.com/{yourusername}/info3180-project1>

2. Your URL for your Heroku app e.g. <https://{yourappname}.herokuapp.com>

## Grading

1. 3 routes /profile, /profiles, /profile/<userid> should be defined along with their respective view functions. (3 marks)
2. /profile has form with firstname, lastname, email, location, biography, file upload field called photo and select gender field. (6 marks)
3. The Form successfully submits and user profile added to database. (1 mark)
4. File successfully uploads and filename stored in database. (2 marks)
5. Ensure userid is generated and created\_on date added to user profile. (2 marks)
6. You should be able to view the specific user profile at /profile/<userid> (5 marks)
7. You should be able to view a list of users added at /profiles (2 marks)
8. Database Migration(s) should be created and the database should be able to be recreated from that migration. (2 marks)
9. Navigation links for "Profile" and "Profiles" are in the navigation bar at the top. (2 marks)
10. Deployed to Heroku and the application works there. If you are able to create a user then it also means the database was created and connected to properly on Heroku. (2 marks)
11. 'Profile', 'Profiles' and 'Add Profile Form' should look similar to screenshots. (3 marks)