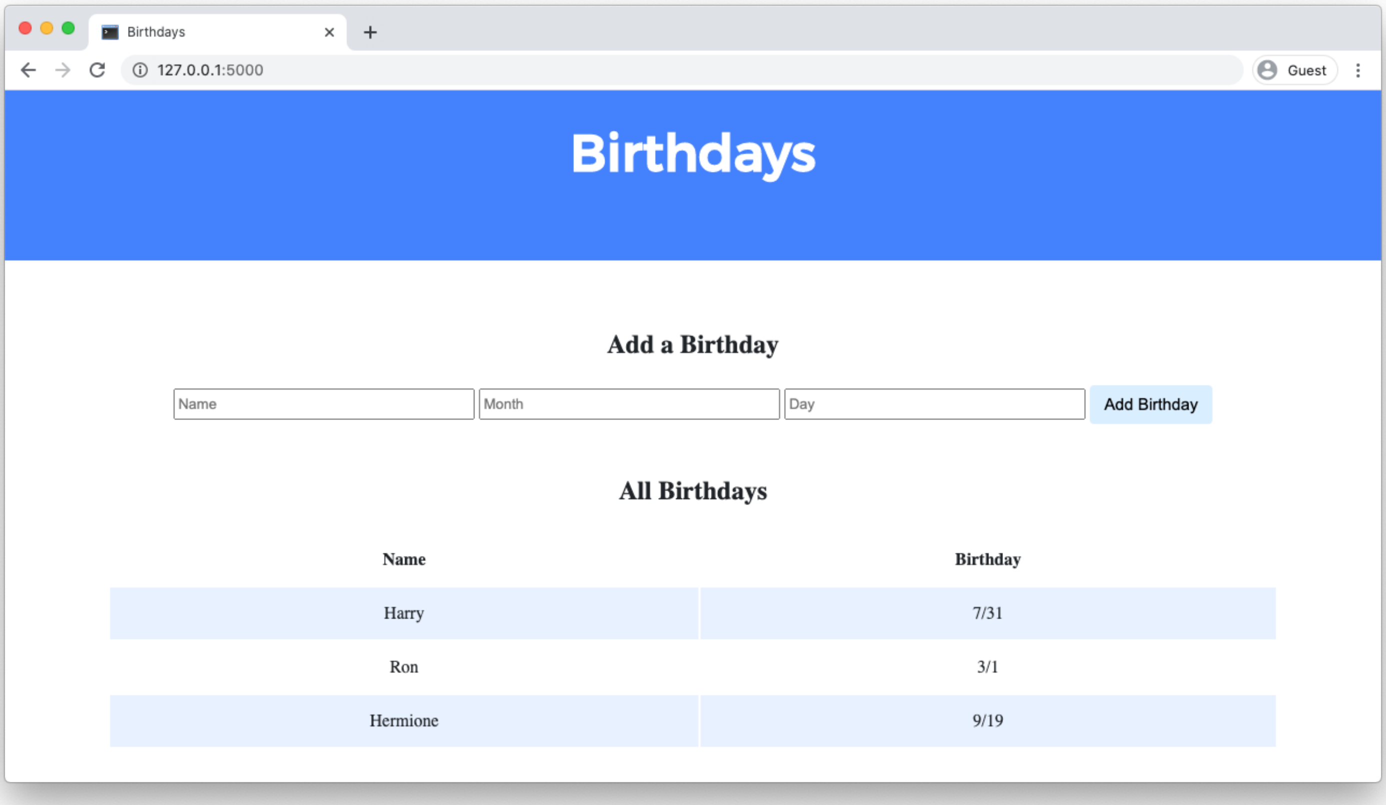
Lab 9: Birthdays

You are welcome to collaborate with one or two classmates on this lab, though it is expected that every student in any such group contribute equally to the lab.

Create a web application to keep track of friends' birthdays.



Birthdays

Add a Birthday

All Birthdays

Name	Birthday
Harry	7/31
Ron	3/1
Hermione	9/19

Getting Started

Here's how to download this lab into your own CS50 IDE. Log into CS50 IDE (https://ide.cs50.io/) and then, in a terminal window, execute each of the below.

- Execute `cd` to ensure that you're in `~/` (i.e., your home directory, aka `~`).
- Execute `wget https://cdn.cs50.net/2020/fall/labs/9/lab9.zip` to download a (compressed) ZIP file with this problem's distribution.
- Execute `unzip lab9.zip` to uncompress that file.
- Execute `rm lab9.zip` followed by `yes` or `y` to delete that ZIP file.
- Execute `ls`. You should see a directory called `lab9`, which was inside of that ZIP file.
- Execute `cd lab9` to change into that directory.
- Execute `ls`. You should see an `application.py` file, a `birthdays.db` file, a `static` directory, and a `templates` directory.

Understanding

In `application.py`, you'll find the start of a Flask web application. The application has one route (`/`) that accepts both `POST` requests (after the `if`) and `GET` requests (after the `else`). Currently, when the `/` route is requested via `GET`, the `index.html` template is rendered. When the `/` route is requested via `POST`, the user is redirected back to `/` via `GET`.

`birthdays.db` is a SQLite database with one table, `birthdays`, that has four columns: `id`, `name`, `month`, and `day`. There are a few rows already in this table, though ultimately your web application will support the ability to insert rows into this table!

In the `static` directory is a `styles.css` file containing the CSS code for this web application. No need to edit this file, though you're welcome to if you'd like!

In the `templates` directory is an `index.html` file that will be rendered when the user views your web application.

Implementation Details

Complete the implementation of a web application to let users store and keep track of birthdays.

- When the `/` route is requested via `GET`, your web application should display, in a table, all of the people in your database along with their birthdays.
  - First, in `application.py`, add logic in your `GET` request handling to query the `birthdays.db` database for all birthdays. Pass all of that data to your `index.html` template.
  - Then, in `index.html`, add logic to render each birthday as a row in the table. Each row should have two columns: one column for the person's name and another column for the person's birthday.
- When the `/` route is requested via `POST`, your web application should add a new birthday to your database and then re-render the index page.
  - First, in `index.html`, add an HTML form. The form should let users type in a name, a birthday month, and a birthday day. Be sure the form submits to `/` (its "action") with a method of `post`.
  - Then, in `application.py`, add logic in your `POST` request handling to `INSERT` a new row into the `birthdays` table based on the data supplied by the user.

Optionally, you may also:

- Add the ability to delete and/or edit birthday entries.
- Add any additional features of your choosing!

Hints

- Recall that you can call `db.execute` to execute SQL queries within `application.py`.
  - If you call `db.execute` to run a `SELECT` query, recall that the function will return to you a list of dictionaries, where each dictionary represents one row returned by your query.
- You'll likely find it helpful to pass in additional data to `render_template()` in your `index` function so that access birthday data inside of your `index.html` template.
- Recall that the `tr` tag can be used to create a table row and the `td` tag can be used to create a table data cell.
- Recall that, with Jinja, you can create a `for` loop (https://jinja.palletsprojects.com/en/2.11.x/templates/#for) inside your `index.html` file.
- In `application.py`, you can obtain the data `POST`ed by the user's form submission via `request.form.get(field)` where `field` is a string representing the `name` attribute of an `input` from your form.
  - For example, if in `index.html`, you had an `<input name="foo" type="text">`, you could use `request.form.get("foo")` in `application.py` to extract the user's input.

Testing

No `check50` for this lab! But be sure to test your web application by adding some birthdays and ensuring that the data appears in your table as expected.

Run `flask run` in your terminal while in your `lab9` directory to start a web server that serves your Flask application.

How to Submit

1. Download a ZIP file of your Flask application by control-clicking on your `lab9` folder in CS50 IDE's file browser and choosing **Download**.
2. Go to CS50's Gradescope page (https://www.gradescope.com/courses/157004).
3. Click "Lab 9: Birthdays".
4. Drag and drop your `lab9.zip` file to the area that says "Drag & Drop".
5. Click "Upload".

You should see a message that says "Lab 9: Birthdays submitted successfully!"