Instruction manual

Flask app

Using python 3.9

Dependencies

List of dependencies

```
flask
flask-wtf
flask-sqlalchemy
```

Install deps (with pip)

```
pip3 install {depname}
```

Run the app

In a terminal

```
python3 run.py
```

Useful exemples

Add a new page

Create a HTML view

Start by adding a new HTML file in `"/app/templates/public/"` and call it whatever you want, in this exemple, I will call it `"about.html"`

Edit your HTML file

It has to look like this because we are using jinja's templates brought by flask.

You can edit your HTML in the "main" block in the middle.

```
{% extends "public/templates/public_template.html" %}
{% block title %}About{% endblock %}

{% block main %}
<div class="container">
      <h2>About page</h2>
</div>
{% endblock %}
{% block script %}

{% endblock %}
```

Since we use a template, let's take a look at ours in
`/app/public/templates/public_template.html`

```
<!doctype html>
<html lang="en">

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

  <!-- Import our custom stylesheet -->
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">

  <title>{% block title %}{% endblock %}</title>
</head>

<body>

  <main>
    {% block main %}{% endblock %}
  </main>
```

```
  <!-- Import our custom JavaScript -->
  <script src="{{ url_for('static', filename='js/app.js') }}"></script>
  {% block script %}{% endblock %}
</body>

</html>
```

In the <main> tag you can see the “main” block content is included from the view you just created.

Create a route to access the view

Routes look like this in flask.

```
@app.route("/about")
def about():
    return render_template("public/about.html")
```

The function about returns the template /public/about.html from the app.

Text decorator “@app.route” allows you to create a custom URL for your view.

SQLAlchemy

Create a new model

Models are defined as a table of your database, it works like a class with attributes

Go to `/app/models.py` and let’s see how models work.

```
class users(db.Model):
    id = db.Column('student_id', db.Integer, primary_key = True)
    name = db.Column(db.String(100))
    email = db.Column(db.String(100))
    password = db.Column(db.String(100))

    def __init__(self, name, email, password):
        self.name = name
        self.email = email
        self.password = password
```

The name of the class is the table name. Here its “users”.

For each attribute in a table, SQLAlchemy creates a new column in the database.

Table attributes are named after the class attributes but you can specify it in the first parameter of the function `db.Column()`

The previous function always takes a type of data as a parameter. In this case, every attribute is a string of 100.

You can also specify various params like the primary key.

Access/edit the database

Inserts records into a mapping table

```
db.session.add (model object)
```

Delete records from a table

```
db.session.delete (model object)
```

Retrieves all records (corresponding to SELECT queries) from the table

```
model.query.all ()
```

Send queries data through routes

Simple query

```
students.query.filter_by(city = 'Tokyo').all()
```

Route with query

```
@app.route('/')
def show_all():
   return render_template('show_all.html', students = students.query.all()
)
```

More to come soon..