Python Programming with Sqlite and Flask

1. **Objective**

   The objective is to make sure
   You are familiar with the principles of Information Management.
   You can set up a simple Database and interact with it using SQL.
   You can set up a flask server in a Python application to work with the Database.

2. **Specifications**

   For this assignment you will be using the sqlite3 and flask libraries to create a database for a text-only ratings and reviews website, to be used by local restaurants. Samples for the different pages of the web application are shown on Section 3. Please make sure you conform to the following specifications:

1. Create a sqlite3 database for your application. Call it reviewData.db (5 points)
2. Write a Python script called setup.py that connects to the database and creates the tables specified in Steps 3 and 4. (5 points)
3. Create a table called Reviews with the following fields (5 points):
   - Username - 40 character text field
   - Restaurant - 50 character text field
   - ReviewTime - SQL Datetime
   - Rating - float
   - Review - 500 character text field
4. Create a table called Ratings with the following fields (5 points):
   - Restaurant - 50 character text field
   - Food - float
   - Service - float
   - Ambience - float
   - Price - float
   - Overall - float
5. Create an HTML page called index.html. This page will serve as the homepage for the website. It should include a Welcome Message and links to take users to the other pages in the website. (5 points)
6. Create an HTML page called addReview.html. This page will display a form for a user to submit a review for a local restaurant. This will include a “Submit” button. Upon clicking the submit button, the application should insert a row in the Reviews and Ratings tables. (5 points)

7. Create an HTML page called getReviews.html. This page will dis- play a form for a user to get all the reviews for a particular restaurant. This will include a “Submit” button that will fetch all the reviews for the given restaurant from the Reviews table and show them in the showReviews.html page. (5 points)
8. Create an HTML page called showReviews.html. This page will display the Username, Review and Overall ratings for a particular restaurant in a table format. This page will be rendered as a result of clicking on “Submit” in the getReviews.html page. (5 points)
9. Create an HTML page called showReport.html. This page will fetch and display all the ratings of the top - 10 restaurants by overall rating. If dif- ferent restaurants have the same overall rating, order them alphabetically. If there are fewer than 10 restaurants in the database, show all of them. (5 points)
10. All pages except the homepage should have a “back to home” link. (5 points)
11. Write a Python script called app.py that is going to essentially run the application. This script should contain functions to:
    - A function to render the homepage for the root directory of the ap- plication. (5 points)
    - A function to add rows to the 2 tables triggered by submitting the form on addReview.html. The data from a single form should be separated into data pertinent to each table and then run 2 SQL insert queries. Make sure you handle exceptional situations by rolling back. (10 points)
    - A function to fetch all reviews for a restaurant, triggered by submit- ting the form on getReviews.html and rendering showReviews.html. (10 points)
    - A function to run the query to get the top-10 restaurants and render the showReport.html page. (15 points)
    - If this were the main module, run the application. (5 points)
12. Please make sure your submission is documented(5 points)
13. Your submission should contain the following files:
    - setup.py
    - app.py
    - index.html
    - showReviews.html
    - showReport.html
    - addReview.html
    - getReview.html

1. Please note that the SQL commands should handle ALL of the database interactions. You should not be using Python functions to filter the results returned by a select all or similar SQL command.

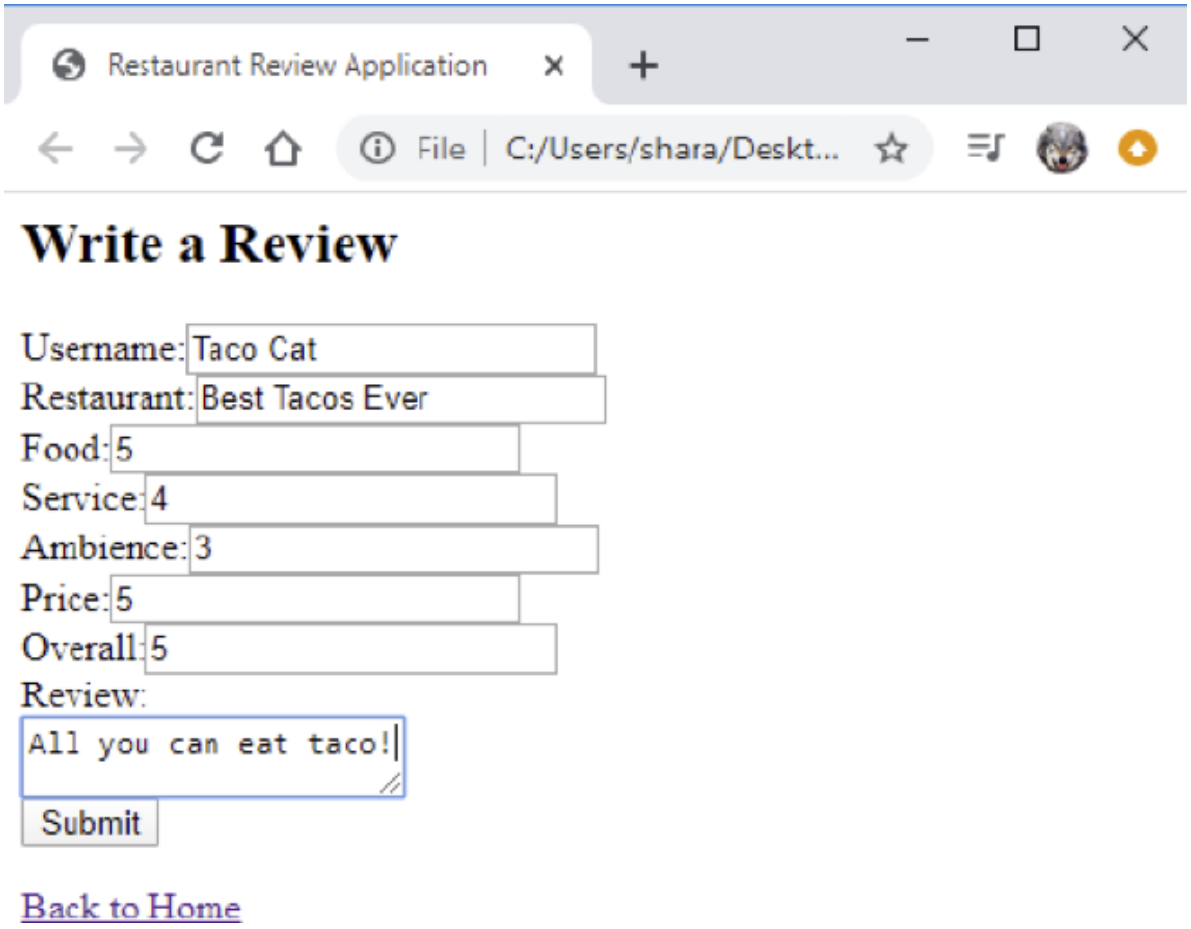
2. Flask requires that the templated HTML files will be in a folder called “templates” in the same folder as the python file. We will use the same setup for tests, even if folder organization is not required for submission.

3. **Sample**

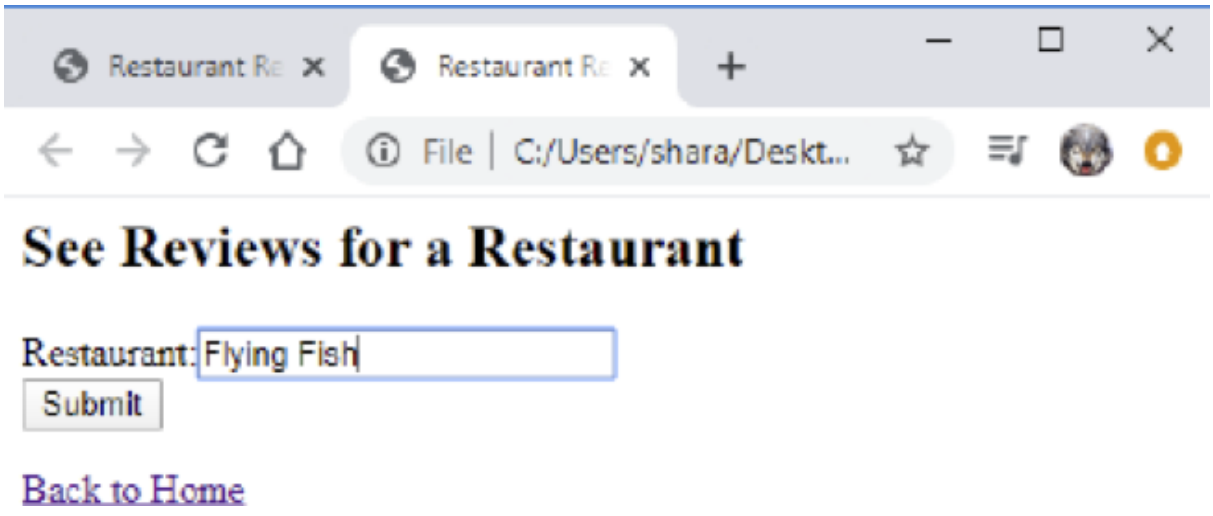
   You do not have to worry about verifying if the data entered by the users would fit in the text fields. The user ratings will also definitely be a number between 1 and 5 (inclusive). The samples below are strictly functional and are not very aesthetic, again because that would be handled by client-side CSS, which is not one of the objectives of the assignment.

- index.html

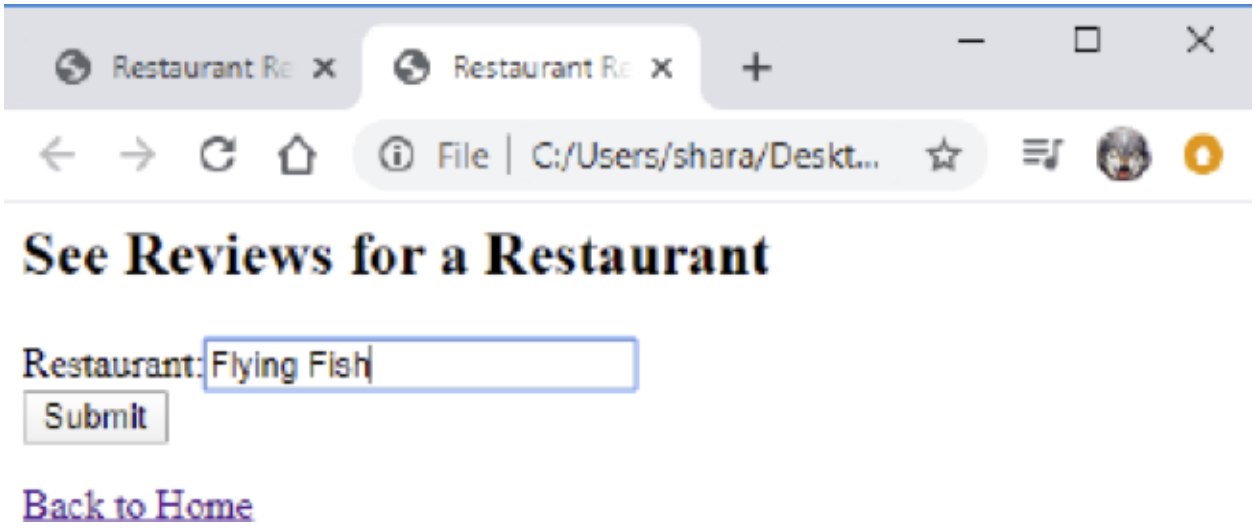
- addReview.html



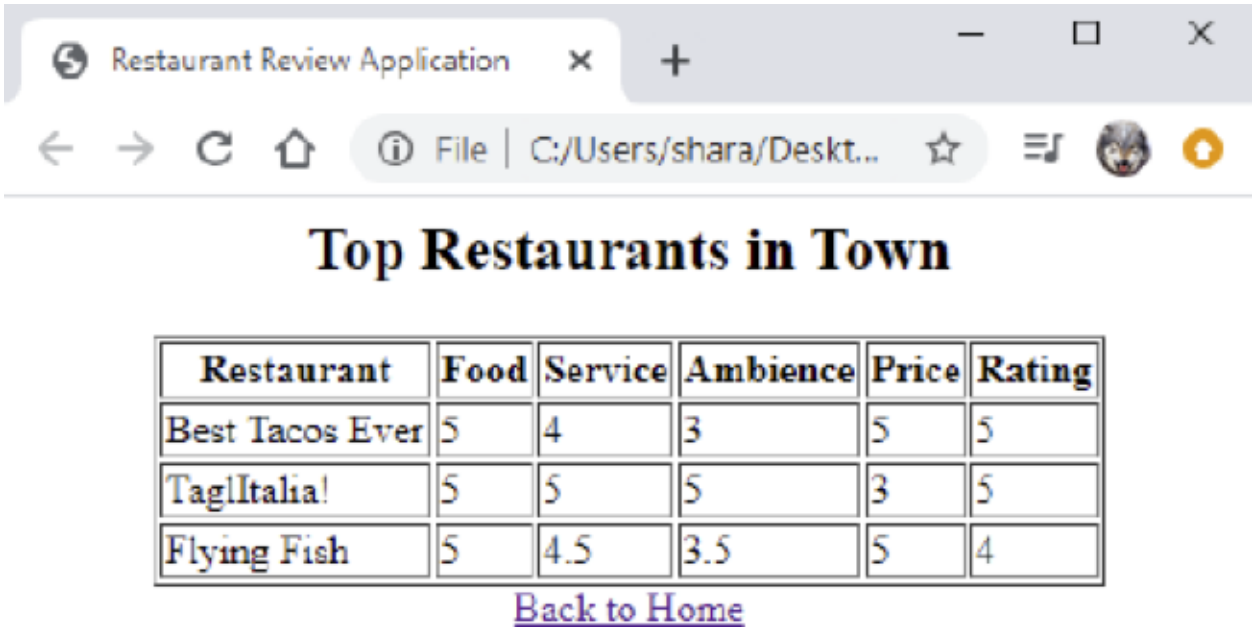
- getReviews.html



- showReviews.html



- showReport.html



Generic Guidelines

- If we have listed a specification and allocated points for it, you will lose points if that particular item is missing from your code, even if it is trivial.
- Your program should load and run without issues. Every interpretation error will result in a loss of 5 points.