BSCCS2003: Week-5 Lab Assignment

In this assignment, you have to create a web application and a database model, using flask and flask-SQLAlchemy. We list below instructions to be followed in preparing and submitting the solution.
General instructions:

- Submit a single .zip file containing all your submission files and folders, the name of which should be "<roll_number>.zip". E.g.: *21f1000000.zip*
- The folder structure inside the zip file should be as follows:
  - The Python program must be written inside a file named "app.py". This file must reside inside the root folder.
  - All the HTML files should be kept inside a folder named "templates" and the images and CSS files (if any) should be kept in "static" folder. Both the "static" and "templates" folder must reside in the root directory.
  - The database file named "database.sqlite3". You are not required to submit this database file with your submission.
- All the endpoints must return 200 as the status code in the case of success.
- You should not keep any code inside the scope of the condition " if __name__ == '__main__' " except run() call.
- Allowed Python packages: jinja2, flask, flask-sqlalchemy, or any standard Python3 package.
- The output pages should be HTML5 compliant, e.g., the file should begin with the declaration <!DOCTYPE html>.
- Please do not use "create_all()" method call in your Python program, and neither add any entry into the "course" table in your program. You can add those entries outside the Python program (using DB Browser, terminal etc.).

  This is because we are using a different database in the backend with the required course entries preloaded in it, and using any of the things mentioned above can cause issues during evaluation.

Problem Statement:

- You have to create a database model and must name the database file as database.sqlite3. The database model must have 3 tables, for which the schema is given below.

- The database URI must be the same as given below:
  app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.sqlite3'
  Note: The table names and column names must be the same as given below.

Table 1: student

Column Name	**Column Type**	**Constraints**
student_id	Integer	Primary Key, Auto Increment
roll_number	String	Unique, Not Null
first_name	String	Not Null
last_name	String	

Table 2: course

Column Name	Column Type	Constraints
course_id	Integer	Primary Key, Auto Increment
course_code	String	Unique, Not Null
course_name	String	Not Null
course_description	String	

Table 3: enrollments

Column Name	Column Type	Constraints
enrollment_id	Integer	Primary Key, Auto Increment
estudent_id	Integer	Foreign Key (student.student_id), Not Null
ecourse_id	Integer	Foreign Key (course.course_id), Not Null

- After creating the model, you have to populate the course table with the following entries (you can DB Browser or terminal for doing this):

Table: Course

course_id	course_code	course_name	course_description
1	CSE01	MAD I	Modern Application Development - I
2	CSE02	DBMS	Database management Systems
3	CSE03	PDSA	Programming, Data Structures and Algorithms using Python
4	BST13	BDM	Business Data Management

Using a standard flask template, create an application that:

1. On the home page (URI = '/'), (when we open it via the browser) displays an index page. The index page must display a table with the list of students currently enrolled. The HTML table should be the same as given below (Its id must be "all-students"). It should display an appropriate message if no student is enrolled. It should also have a button labeled "Add student", as shown in the Figure 1.

```
<table id = "all-students">
    <tr>
        <th>SNo</th>
        <th>Roll Number</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Actions</th>
    </tr>
    <tr>
        <td>s_no_1</td>
        <td><a href="/student/<int:student_id>">roll_number_1</a></td>
        <td>first_name_1</td>
        <td>last_name_1</td>
        <td>
          <a href="/student/<int:student_id>/update" type="button">Update</a>
          <a href="/student/<int:student_id>/delete" type="button">Delete</a>
        </td>
    </tr>
        ......
        ......
        ......
    <tr>
        <td>s_no_n</td>
        <td><a href="/student/<int:student_id>">roll_number_n</a></td>
        <td>first_name_n</td>
        <td>last_name_n</td>
```

```
          <td>
            <a href="/student/<int:student_id>/update" type="button">Update</a>
            <a href="/student/<int:student_id>/delete" type="button">Delete</a>
          </td>
      </tr>

  </table>
```

Note: s_no_1, roll_number_1, first_name_1, last_name_1 .... s_no_n, roll_number_n, first_name_n, last_name_n, $< int : student\_id >$ **should be populated accordingly.**

2. If the user clicks the "Add student", your flask application should send a GET request to an endpoint "/student/create", which should display an HTML form as shown in the Figure 2. The HTML form should be the same as given below. Its id must be "create-form".

```
<form action="/student/create" method="POST" id="create-form">
    <div>
      <label>Roll Number:</label>
      <input type="text" name="roll" required />
    </div>

    <div>
      <label>First Name:</label>
      <input type="text" name="f_name" required />
    </div>

    <div>
      <label>Last Name:</label>
      <input type="text" name="l_name" />
    </div>

    <div>
      <label>Select Courses: </label>
      <input type="checkbox" name="courses" value="course_1" />
      <label>MAD I</label>
      <input type="checkbox" name="courses" value="course_2" />
      <label>DBMS</label>
      <input type="checkbox" name="courses" value="course_3" />
      <label>PDSA</label>
      <input type="checkbox" name="courses" value="course_4" />
      <label>BDM</label>
    </div>

    <div>
      <input type="submit" value = "Submit">
    </div>
</form>
```

- The HTML form should not have any other input elements.
- If the user clicks the submit button, the browser should send a POST request to your flask application's "/student/create" URI. The flask application should then create a student object (with attributes roll_number, first_name and last_name) and enrollments objects(s) (depending on the number of courses user selects) and add them into the database and, it should redirect to the

home page (URI = ‘/’) and the student should be added into the table as shown in the Figure 3. Note that each row of the table should be clickable

- If the roll number already exists, then, the user should be redirected to an HTML page, which should display an appropriate message and have a button to navigate back to the home page (URI = ‘/’), as shown in the Figure 4.

3. If the user clicks the “Update” button, your flask application should send a GET request to an endpoint “/student/<int:student_id>/update”, which should display an HTML form as shown in the Figure 5. The HTML form should be the same as given below. Its id must be “update-form”.

```
<form action="/student/<int:student_id>/update" method="POST" id="update-form">
    <div>
      <label>Roll Number:</label>
      <input type="text" name="roll" value="current_roll" disabled />
    </div>

    <div>
      <label>First Name:</label>
      <input type="text" name="f_name" value="current_f_name" required />
    </div>

    <div>
      <label>Last Name:</label>
      <input type="text" name="l_name" value="current_l_name"/>
    </div>

    <div>
      <label>Select Courses: </label>
      <input type="checkbox" name="courses" value="course_1" />
      <label>MAD I</label>
      <input type="checkbox" name="courses" value="course_2" />
      <label>DBMS</label>
      <input type="checkbox" name="courses" value="course_3" />
      <label>PDSA</label>
      <input type="checkbox" name="courses" value="course_4" />
      <label>BDM</label>
    </div>

    <div>
      <input type="submit" value = "Submit">
    </div>
</form>
```

Note: $< int : student\_id >$**, current_f_name, current_l_name and current_roll should be populated accordingly.**

- The HTML form should not have any other input elements.
- If the user clicks the submit button, the browser should send a POST request to your flask application’s “/student/<int:student_id>/update” URI.
- The flask application should then update the student and corresponding enrollments into the database and redirect to the home page (URI = ‘/’).

4. If the user clicks the "Delete" button, your flask application should send a GET request to an endpoint "/student/<int:student_id>/delete", which should delete the student and all the corresponding enrollments from the database and redirect to the home page (URI = '/').

5. If the user clicks on the roll number of any row in the table in the home page of the flask application, the application should send a GET request to an endpoint "/student/<int:student_id>", which should show all the information (student details and enrollment details) in an HTML page. The HTML page should also have a button labelled "Go Back" to navigate back to the home page (URI = '/'), as shown in the Figure 6. There must be 2 HTML tables in this page, one for showing the personal details and the other for displaying the enrollment details. The HTML for showing personal details should be the same as given below. Its id must be "personal-details".

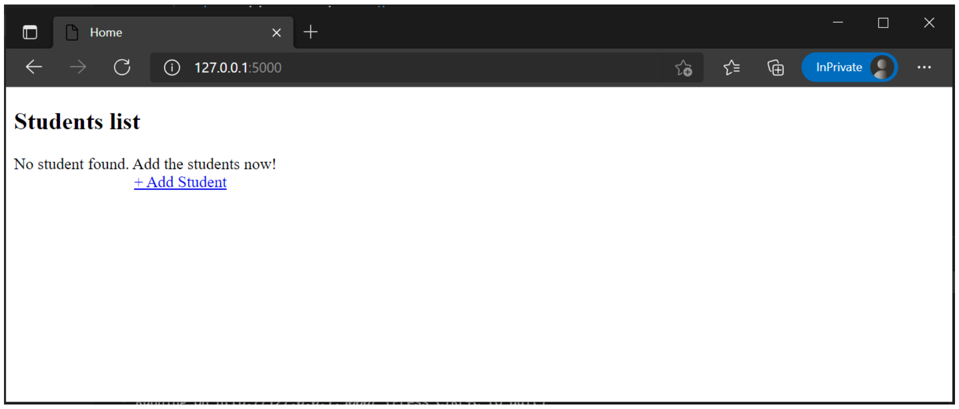
```
<table id = "personal-details">
        <tr>
            <th>Roll Number</th>
            <th>First Name</th>
            <th>Last Name</th>
        </tr>
        <tr>
            <td>roll-no-of-student</td>
            <td>first-name-of-student</td>
            <td>last-name-of-student</td>
        </tr>
</table>
```

Note: All the <td> should be populated accordingly.

The HTML for displaying the enrollment details should be the same as given below. Its id should be "enroll-table".

```
<table id = "enroll-table">
        <tr>
            <th>S. No.</th>
            <th>Course Code</th>
            <th>Course Name</th>
            <th>Course Description</th>
        </tr>
        <tr>
            <td>1</td>
            <td>course-code-for-course_1</td>
            <td>course-name-for-course_1</td>
            <td>course-description-for-course_1</td>
        </tr>
        ......
        ......
        ......
        <tr>
            <td>2</td>
            <td>course-code-for-course_n</td>
            <td>course-name-for-course_n</td>
            <td>course-description-for-course_n</td>
        </tr>
</table>
```

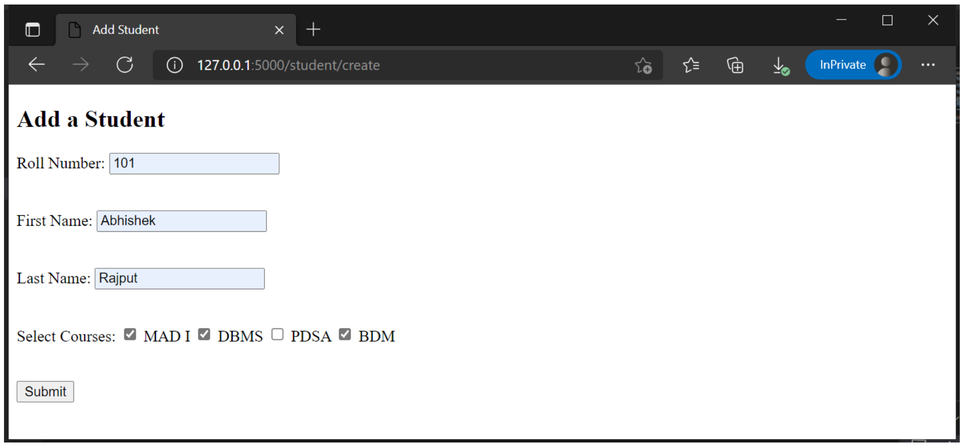
Note: All the <td> should be populated accordingly.



Students list

No student found. Add the students now!
+ Add Student

Figure 1: Home



Add a Student

Roll Number: 101

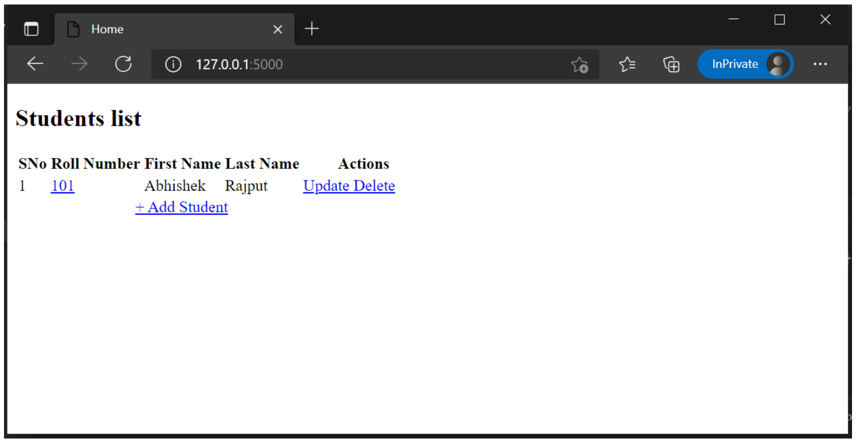
First Name: Abhishek

Last Name: Rajput

Select Courses: [✓ MAD I] [✓ DBMS] PDSA [✓ BDM]

Submit

Figure 2: Add Student Form

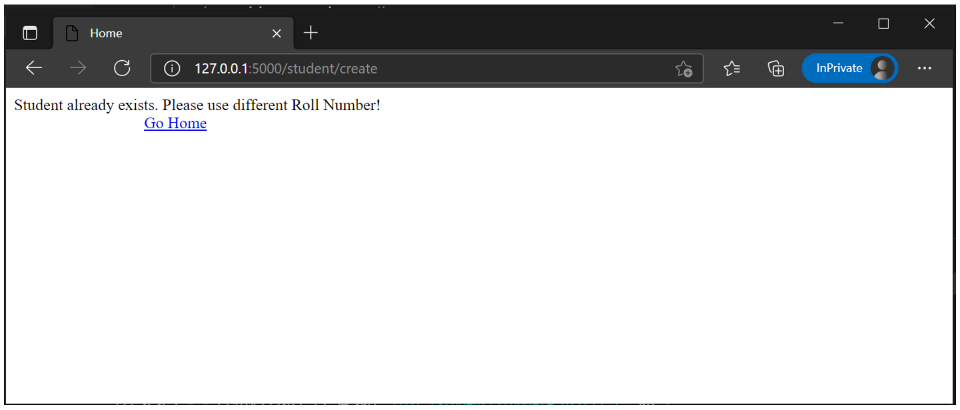


Students list

SNo	Roll Number	First Name	Last Name	Actions
1	101	Abhishek	Rajput	Update Delete

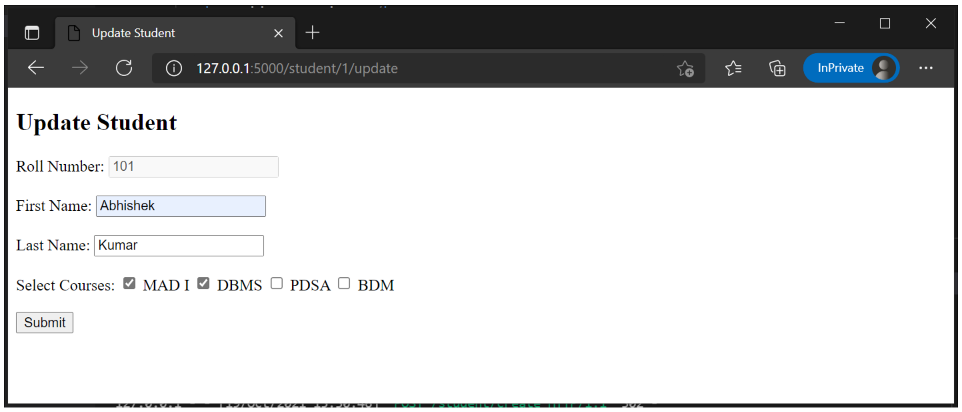
+ Add Student

Figure 3: Home



Student already exists. Please use different Roll Number!
Go Home

Figure 4: Already Exists Page



Update Student

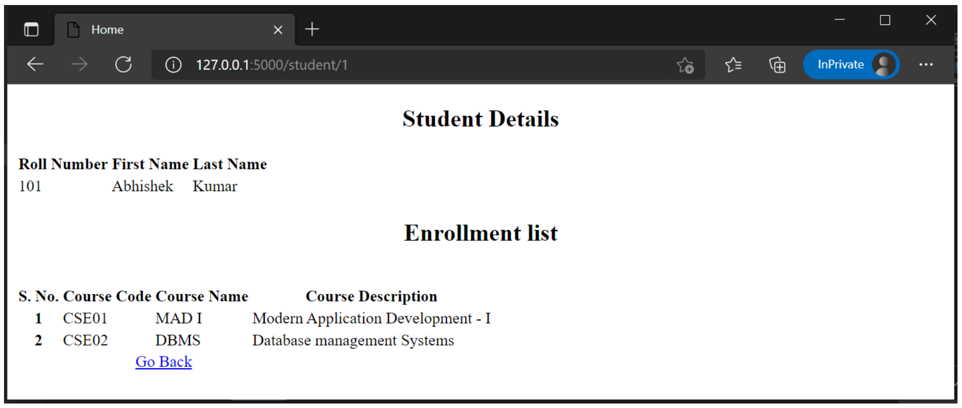
Roll Number: 101

First Name: Abhishek

Last Name: Kumar

Select Courses: [✓ MAD I] [✓ DBMS] PDSA BDM

Figure 5: Update Student Form



Student Details

Roll Number	First Name	Last Name
101	Abhishek	Kumar

Enrollment list

S. No.	Course Code	Course Name	Course Description
1	CSE01	MAD I	Modern Application Development - I
2	CSE02	DBMS	Database management Systems

Go Back

Figure 6: Student Details Page