CSCB20 Final Exam

Duration: 3 hours
Total Points: 95
Instructions:

- This test contains 15 pages including THIS page
- Please write legibly
- DO NOT DETACH ANY PAGES
- If you need extra space to write your answers, you can write on the backside of the page and mention continuation of your answer in the allocated box
- NO aids are allowed except a pen/pencil
- SCRATCH work should go on the BACK of each page
- Write your FINAL answers on the boxes
- Budget your time carefully
- Please read all questions carefully before answering them
- Some questions are easier, others harder; if a question sounds hard, skip it, and return later
- If you have any doubts, write your assumptions, and complete your answer
- Good luck!

Student Number: _____ Signature: _____

Question 1 (10 points)

Description: (all topics)

Answer following multiple choice questions. Circle the correct answers. If you think multiple options are correct, explain your answer.

1.1 Which one of the following HTTP request methods sends the data from browser to server to update data on the server?

(a) HTTP GET

(b) HTTP POST

(c) HTTP HEAD

(d) HTTP DELETE

1.2 Which one of the selectors in CSS is used to select only one element using unique identifier?

(a) Type Selector

(b) Class Selector

(c) ID Selector

(d) Attribute Selector

1.3 SQLite is a _____

(a) NoSQL Database

(b) Relational Database

(c) Operational Database

(d) Distributed Database

1.4 _____Key is s minimal super key

(a) Primary

(b) Foreign

(c) Candidate

1.5 SQLite doesn't directly support _____ join

(a) INNER JOIN

(b) NATURAL JOIN

(c) LEFT JOIN

(d) FULL OUTER JOIN

1.6 _____allows to create responsive websites across all screen sizes, ranging from desktop to mobile.

(a) Media Queries

(b) JavaScript

(c) Events

(d) HTML and CSS

1.7 A session Data is stored on the _____

(a) Server

(b) Browser

(c) None of the above

1.8 In order to load web pages with JavaScript faster, where do we embed the <script> tag?

(a) At the bottom of HTML before <body> is closed

(b) At the beginning of HTML inside <head> tag

(c) Doesn't matter

1.9 What is variable hoisting?

(a) Using a variable before declaring it

(b) Declaring a variable before using it

(c) Assuming random value for undeclared variables

1.10 Which one of the following is incorrect statement for JavaScript using Document Object Model?

(a) JavaScript can change HTML elements, attributes, CSS styles in page

(b) JavaScript can remove existing HTML elements and attributes

(c) JavaScript can react to and create HTML elements

(d) JavaScript can connect HTML pages and database

Question 2 (15 points)

Description: (all topics)
Answer following questions

2.1 Explain the difference between authentication and authorization of a user of a website. Include an example of any website page that authenticates vs authorizes its user.

2.2 Consider following code that defines the schema for tables ‘User’, ‘Role’, ‘Post’, ‘Comment’ for a social blogging website.

Users of this social blogging can have one of two roles. A user can be a ‘moderator’ or ‘user’. If user is a ‘moderator’, then user has permission to moderate comments made by other users. If user is a ‘user’, then he/she has basic permissions to write posts, and comments.

You may assume that the code below is complete, and syntax is correct.

- Suppose you open a terminal to create these two tables. What commands will you use to create these tables?
- Next, draw both the tables by filling some random values for each attribute. Create at least two rows for each table. No need to write the code for inserting rows.

```
class Role(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        name = db.Column(db.String(64), unique=True)
        users = db.relationship('User', backref='role', lazy='dynamic')
class User(db.Model):
        id = db.Column(db.Integer, primary_key = True)
        email = db.Column(db.String(64), unique=True)
        username = db.Column(db.String(64), unique=True)
        role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
        comments = db.relationship('Comment', backref='author', lazy='dynamic')
        posts = db.relationship('Post', backref='author', lazy='dynamic')
class Post(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        body = db.Column(db.Text)
        author_id = db.Column(db.Integer, db.ForeignKey('users.id'))
        comments = db.relationship('Comment', backref='post', lazy='dynamic')
class Comment(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        body = db.Column(db.Text)
        author_id = db.Column(db.Integer, db.ForeignKey('users.id'))
        post_id = db.Column(db.Integer, db.ForeignKey('posts.id'))
```

Python Terminal Commands:

Sample tables:

Question 3 (30)

Description: (Web Dev and Databases – Flask and SQLite)

3.1 Suppose you are the moderator of this social blogging website. Assume you have Write a query in SQLite such that for a given user with role 'user' (assume user id – 55 with username 'ABC'), all the posts by that user and comments made by this user on their own post are returned. You must show back the following columns in your result, `UserId, UserName, UserPost, UserPostComment`.

DO NOT FORGET to rename the columns as required.

Write the SQLite query only (no need to write it using SQLAlchemy).

3.2 When the moderator of this social blogging site goes to `http://localhost:5000/viewPosts` using `HTTP GET` request, you must show back a very simple HTML form that allows the moderator to enter the user's name.

When the moderator enters the username into this form, such that HTML form sends the data from client via `HTTP POST` back to flask server on `http://localhost:5000/viewPosts`. The moderator should be able to see following result format for user 'ABC' (Assuming ABC has made 2 blog post with following content):

> Posts by User ABC:
> Post 1: "This is my first blog post…."
> Post 2: "This is my second blog post…."

To complete:

Python Flask:

Complete the `viewPosts` method for both HTTP GET and POST request, in Python using Flask. You can safely assume that all the import statements are in place. Pay attention to any routing that may be required. You can write additional helper functions in Flask if required.

HTML:

1) Form: The form you need to render should be written in HTML named `viewPosts.html`. Complete the form below Python Flasks method `viewPosts` method.

2) View the posts for entered user: The result that should be rendered in the format mentioned above in html page `view_posts_success.html`

Here is a **helper function** in Python Flask that will help you write queries to get the users based on their username and select the related posts for those users:

```
def get_posts(user_details):
  userid = user_details[0] #returns the id based on the user
   sql = text('select * from Posts JOIN User
              where User.id = Posts.author_id AND User.id = :val')
   result = db.engine.execute(sql, val = userid)
   return result;
```

This function receives a tuple based on the form and returns a dictionary r that contains all the information about given user including the posts by that user. The posts for user can be accessed as follows:

```
for r in result:
    print(r['body'])
```

The output will be as follows:

```
"This is my first blog post…."
"This is my second blog post…."
```

Assumption: You can safely assume that moderator has already logged in and can view other users' posts and comments.

```
from flask, import Flask, session, redirect, url_for, escape, request,
render_template
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import text


app = Flask(__name__)
db = SQLAlchemy(app)



def viewPosts():
```

```
 HTML form for entering the user to view their posts
 <html>
```

```
 </html>
```

```
HTML form for viewing the posts by users
<html>









</html>
```

3.3 Complete the following login method in Python Flask, for users with the role 'moderator'. The login page takes the email and username of the moderator as input in the form and logs the moderator in if both matches.

To complete:

1) In case the username of the moderator entered the form doesn't exist, then it flashes a message "check the login details and try again". Otherwise, it redirects the moderator to some "home" page.

2) Create a session to store the username of the moderator. Pay attention to configurations that need to be added for creating a session.

3) Passwords should be matched using hashing method `check_password_hash('hashed_pw', 'password')`

Assume that "home.html" and "login.html" HTML pages exist, and you do not need to create them.

```
from flask import Flask, render_template, url_for, flash, redirect, request,
session
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt

# complete the configurations
app = Flask(__name__)



@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'GET':







    else:
        username = request.form['Username']
        password = request.form['Password']
```

Question 4 (20)

Description: (CSS and JavaScript)

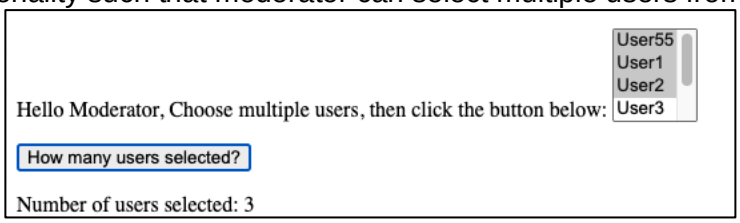
4.1 For the social blogging website, create a navigation bar with all the options to the right-end using CSS Flexbox.

The HTML code is given:

Assumptions: All the HTML pages are already created. Syntax for given HTML is correct.

```
<ul class="navigation">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Moderator login</a></li>
  <li><a href="#">User login</a></li>
</ul>
```

4.2 For the social blogging website, suppose you want to create HTML page viewPosts.html with added functionality such that moderator can select multiple users from given options.



HTML code is given below:

```
<form name="selectForm">
  <p>
    <label for="Users"> Hello Moderator, choose multiple users,..:</label>
    <select id="Users" name="Users" multiple="multiple">
      <option selected="selected">User55</option>
      <option>User1</option>
      <option>User2</option>
      <option>User3</option>
      <option>User4</option>
    </select>
  </p>
  <p><input id="btn" type="button" value="How many users selected?"/></p>
  <p id="output">
  </p>
</form>
```

To complete: Complete the JavaScript code given below.
The code should add the line: “Number of users selected” in the `<p id="output"> </p>`

```
let outputBox = document.getElementById("output");
// a function that loops over all the options using selectObject.options.length
// to get the number of users selected and return that number
function howMany(selectObject) {









return numberSelected;
}

let btn = document.getElementById('btn');
// event listener that calls howMany function and adds the line
// Number of users selected
btn.addEventListener('click', function() {









});
```

Question 5 (20)

Description: (RA and SQL queries)

An online image sharing company uses a database with the following schema:

`Users(uid, uname, city)`
`Image(iid, creator, size, pdf)`

- `Users` stores all users; `uid` is the key.
- `Image` stores their images; iid is the key; `creator` is the `uid` of the images's creator; `size` represents the size of the image in bytes; `pdf` is the actual pdf content of the image.
- `uid, iid, author, size` are integers; `uname, city, pdf` are text.

5.1 Write a SQL query that retrieves all users who do not have any image greater than 1MB (`size` > 1000000). Your query should return the users' `uid` and `uname`

5.2 Write a SQL query that returns all users that have posted both an image larger than 1MB (size > 1000000) and an image smaller than 1MB. Your query should return the users' `uid` and `uname`

5.3 Write a Relational Algebra expression that is equivalent to the following SQL query

```
select x.uid, x.uname, avg(y.size) as s
from Users x, Image y
where x.uid = y.creator and x.city = 'Toronto' and y.size > 20000
group by x.uid, x.uname
having count(*) > 10;
```

You may either write a Relational Algebra expression, or draw a query plan.

That's it! Have a great Summer!!