CSCB20 Final Exam
Winter 2022

Duration: 3 hours
Total Points: 80
Instructions:

- This test contains 12 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: _____

	Number of Questions	Points
Question 1 (multiple choice)	10	10
Question 2 (Flask and SQLite)	2	30
Question 3 (short answers)	4	20
Question 4 (SQL and RA)	3	20

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 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

1.2 SQLite is a _____

(a) NoSQL Database

(b) Relational Database

(c) Operational Database

(d) Distributed Database

1.3 _____key is a minimal super key

(a) Primary

(b) Foreign

(c) Candidate

1.4 SQLite doesn't directly support _____ join

(a) INNER JOIN

(b) NATURAL JOIN

(c) LEFT JOIN

(d) FULL OUTER JOIN

1.5 _____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

1.6 A session Data is stored on the _____

(a) Server

(b) Browser

(c) None of the above

1.7 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

1.8 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

1.9 Which of the following methods is used to access HTML element using JavaScript?

(a) getElementById()

(b) getElementByClassName()

(c) Both (a) and (b)

(d) None of the above

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 (30)

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

2.1 [12 points] Consider following code that defines the schema for tables 'Role', 'User', '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.
- Posts are the posts made by users. One user can make multiple posts.
- Comments are the comments that user can add on their own posts as well as other users' posts. One user can add multiple posts.

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

To complete:

- Draw all the four 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)
        username = db.Column(db.String(64), unique=True)
        password = db.Column(db.String(20), unique=True)
        role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
        comments = db.relationship('Comment', backref='author', lazy='dynamic')
        post = 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'))
```

Sample tables:

id	name
1	user
2	moderator

id	username	password	role_id
1004	user1	user1p	1
2004	m1	m1p	2
1005	user2	user2p	1

id	body	author_id
3001	post made by user1	1004
3002	posts made by user2	1005

id	body	author_id	post_id
4001	good post	1004	3002
4002	informative post	1005	3001

2.2 [18 points] Complete the following login method in Python Flask, for users with the role 'moderator'. The login page takes the username and password of the moderator as input in the form and logs the moderator in if both matches with the User's table in database.

To complete:

1) In case the username of the moderator entered in the form doesn't exist in the table `User`, or if the passwords don't match, then it flashes a message "check the login details and try again". Otherwise, it redirects the moderator to "home" page. With the correct login, `login.html` is rendered.
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 entered by the moderator 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. Also assume that all the imports are complete. Work only with the `User` table. Don't work with the `Role` table.

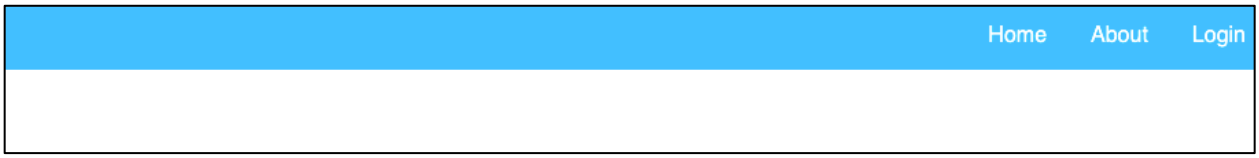
```
from flask import Flask, render_template, url_for, flash, redirect, request,
session
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt

# complete the configurations
app = Flask(__name__)

app.config['SECRET_KEY'] = '84Br5667bb0b13ce0c676dfde280ba245'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///notes.db'
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
```

```
@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'GET':
      if 'name' in session:
                  flash('already logged in!!')
                  return redirect(url_for('home'))
              else:
                  return render_template('login.html')



    else:
        username = request.form['Username']
        password = request.form['Password']
        user = User.query.filter_by(username = username).first()

        if not user or not bcrypt.check_password_hash(user.password,
        password):
                    flash('Please check your login details and try
        again.', 'error')
                    return render_template('login.html')
        else:
        session['name'] = username
                    session.permanent = True
                    return redirect(url_for('home'))
```

Question 3 (20)

Description: (Short answers)

3.1 For the social blogging website, we want to create a navigation bar with all the links (home, about, login) to the right-end using CSS Flexbox.



The HTML and CSS code is given:

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

HTML

```
<ul class="navigation">
  <li><a href=" {{ url_for('home')}}">Home</a></li>
  <li><a href={{ url_for('about') }}>About</a></li>
  <li><a href= {{ url_for('login') }}> login</a></li>
</ul>
```

CSS

```
.navigation {
  display: flex;
  flex-flow: row wrap;
  justify-content: flex-start;
  list-style: none;
  margin: 0;
  background: deepskyblue;
}
```

Currently, CSS code compiles and gives me the output by displaying the navigation bar. But the navigation bar is to the left-end. What should we **change in the given CSS** code so that all the links in my navigation bar are to the right end?

```
justify-content: flex-end;
```

3.2 What is one major difference between `HTTP GET` request and `HTTP POST` request?

HTTP GET is the most common method. A GET message is send, and the server returns data HTTP POST is Used to send HTML form data to the server. The data received by the POST method is not cached by the server.

3.3 For our social blogging website, suppose that the moderator is already logged in using the code you wrote, and moderator wants to view posts made by each user by clicking on the link **View Posts** on a certain HTML page in browser.
Explain in words (no code) set of steps that are executed starting from the browser when the link is clicked.
Your answer should include following:

- What HTTP request is generated when the link is clicked?
- How is the table in database accessed to get the User's information?
- How is response generated from server?
- What HTTP request returns the data back from server to the client's browser?

Make sure to use at least one of the words somewhere in your answer:
HTTP Protocol, HTTP GET/POST, Web Server, Python/Flask, SQLite Database, SQLite table

When the moderator clicks the view Posts, then HTTP GET request is generated to fetch the form that lets moderator input the user's name.
Once moderator inputs user's name and hits submit, HTTP POST request is generated that takes the user's info from the form and check whether that user exists using the USER table. Based on the code written using Python Flask, response is generated and sent back to the client's browser. If the user exists, it accesses the database and fetches the information from the database.
The database is accessed using either using SQLAlchemy (an extension of flask), or by using import sqlite3 to open a database connection on demand
If sqlite3 is used to open database connection, then get_db function is used to get current open db connection. Next, query function can be used that combines getting the cursor, executing and fetching the results from SQLite Database.
If SQLAlchemy is used, a **query** attribute on **Model** class can be used to get back a new query object over all records.
In order to get a specific user's posts, **join** method is used to join two tables USERS and POSTS based on the userid before the select with **all()** to get all the posts for a particular User.

3.4 Explain briefly difference between user authentication and authorization.

- Authentication:
  - Act of validating that users are whom they claim to be.
- Authorization:
  - Process of giving the user permission to access a specific resource or function.
  - Gives client privilege/access control

Question 4 (20)

Description: (RA and SQL queries)

Consider following modified schema for social blogging site, where users can make multiple posts, as well as can comment on their own post and other people's posts. Users can also give each post a rating (an integer between 1 and 10) for their posts.

```
Users (uid, uname)
Posts (pid, author, body)
Comment (uid, pid, rate, txt)
```

The database has following constraints:

- `Users` stores all users; `uid` is the key.
- `Posts` stores their posts; `pid` is the key; `author` is the `uid` of the post's author; `body` represents the text in the post.
- `Comment` stores comments made by each user with the `rate` attribute that gives rating to the post. `txt` represents the actual comment.
- `uid, pid, author, rate` are integers; `uname, body, txt` are text.
- All attributes are `NOT NULL`.

4.1 [5 points] Write a SQL query that retrieves all users who have given a rating of 7 or higher to 100 posts or more. For each user, your query should return the user id and name.

```
select x.uid, x.uname
from Users x, Comment y
where x.uid = y.uid and y.rate >= 7
group by x.uid, x.uname
having count(*) >= 100
```

4.2 [10 points] A post is considered ***highly rated*** if it received at least one rating of 10, from a user other than its author.

A ***vigilant user*** is a user who commented only on ***highly rated*** posts. (A user who did not comment at all is also vigilant.)

Write a SQL query that returns ***all vigilant users***. Your query should return a list of `uid`, `uname` pairs.

```
select x.uid, x.uname
from users x
where x.uid not in
   (select y.uid  -- these are the non-vigilant users
    from comment y  -- check that y.pid is not highly rated
    where y.pid not in
       (select u.pid  -- the posts that are highly rated
        from comment u, posts v
        where u.pid = v.pid and u.rate = 10 and u.pid != v.author))
```

4.3 [5 points] Write a Relational Algebra expression that is equivalent to the following SQL query.

```
select x.pid
from posts x
where not exists
      (select *
      from comment y
      where x.pid = y.pid and y.rate < 5)
```

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

$\Pi_{pid}(Posts) - \Pi_{pid}(\sigma_{rate<5}(Comment\ y))$

That's it! Have a great Summer!!