COP4521 Programming Assignment 6: Enhanced and Hardened Flask Website

(Due: Dec. 4, 2025, 11:59pm)

Objectives:

- Apply the knowledge of computer security to enhance the security of the basic website developed in Programming Assignment 5
- Experience with role-based access control
- Experience with using encryption to partially support data confidentiality in the data stored in the database

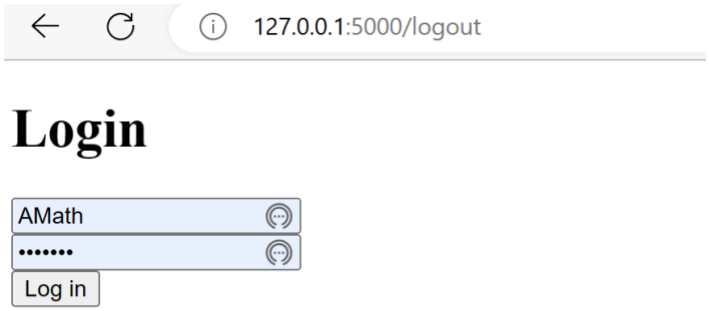
Description:

This assignment is an extension of Programming Assignment 5 and can be done in a group of two. In this assignment, you will use Python, SQLite3, the Flask library, and a cryptography library to create an enhanced and hardened Flask website with role-based access control and (partial) data confidentiality. You will first enhance the website as described in the following and then add support for data confidentiality by encrypting sensitive fields in the database.

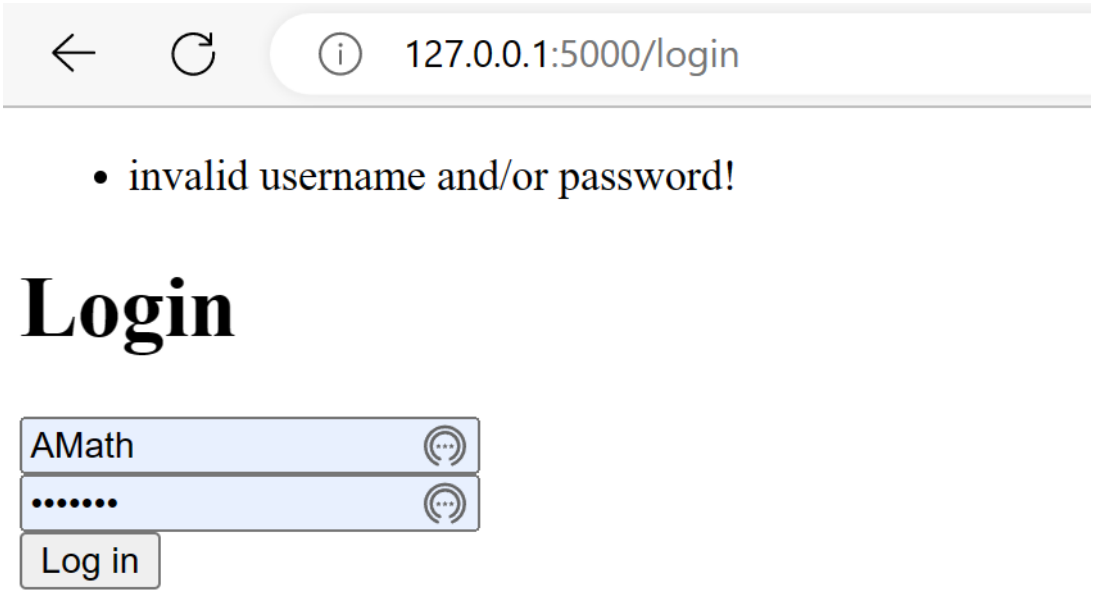
Start with the website that you created in Assignment 5. You will first add a login page (‘/login’). The login page contains two input boxes for username and password and a login button:

- Textboxes for username and password
- Button for Log In. When this button is clicked, the system validates username and password (against the information stored in the user table). If the username and password combination is valid, the system opens the home page for that user. Otherwise, the system notifies user "invalid username and/or password!" and stays on login page.

An example of the login page is as follows:



The following is an example of the page when the login is unsuccessful.



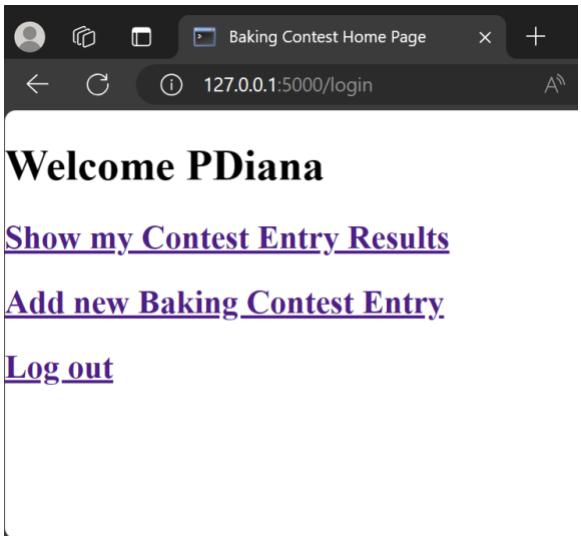
When the user logs in successfully, the system goes to the home page of the user, which will be extended from the home page in Assignment 5 as follows:

- Add a logout link (logs the user out and goes to the login page)
- Display username at top
- Only display links available to the user based on his/her role in the system, which is indicated by attribute SecurityLevel (SecurityLevel is stored in the user table). Depending on the value (1, 2 or 3) of SecurityLevel, a user can have different roles in the system and thus will be given different options of operations for them to perform in the system.

For users whose SecurityLevel = 1, the home page should have four items:

- Name on the top
- A link to Show my Contest Entry Results page
- A link to Add new Baking Contest Entry page
- A logout link

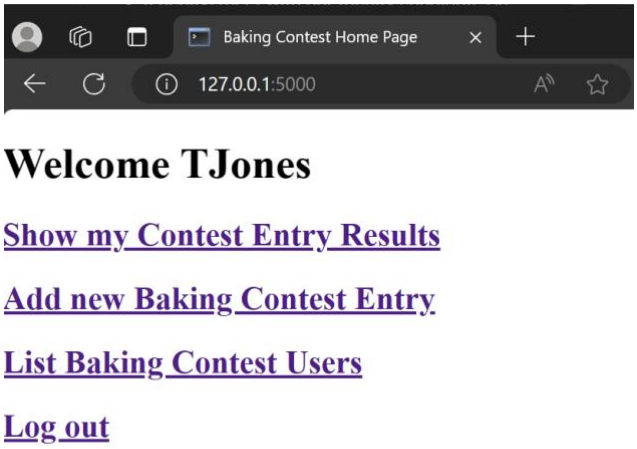
An example is shown below:



For users whose SecurityLevel = 2, the home page should have five items:

- Name on the top
- A link to Show my Contest Entry Results page
- A link to Add new Baking Contest Entry page
- A link to List Baking Contest Users page
- A logout link

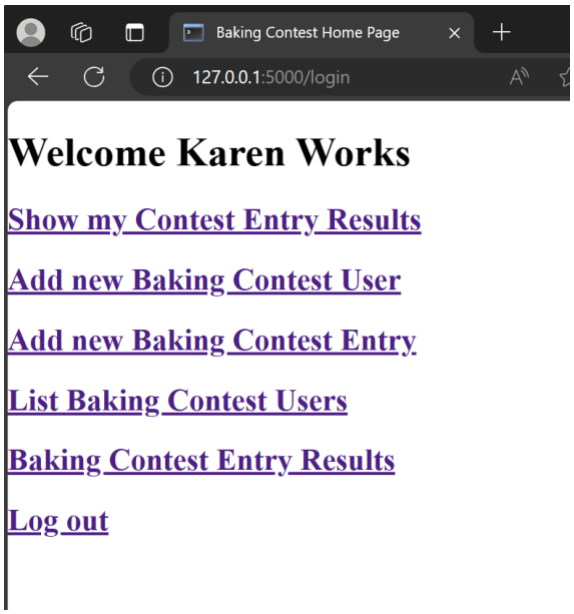
An example is shown below:



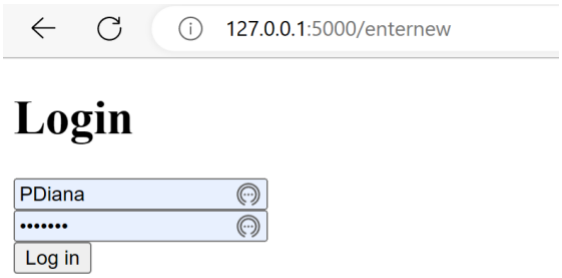
For users whose SecurityLevel = 3, the home page should have seven items:

- Name on the top
- A link to Show my Contest Entry Results page
- A link to Add new Baking Contest User page
- A link to Add new Baking Contest Entry page
- A link to List Baking Contest Users page
- A link to Baking Contest Entry Results page
- A logout link

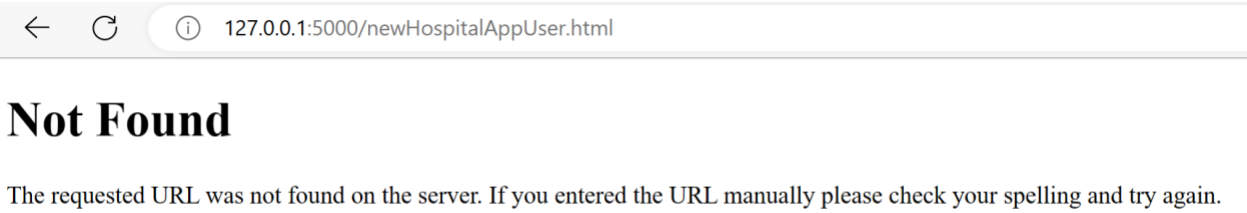
An example is shown below:



If a user is NOT logged in and tries to access a page (which should not be allowed per their SecurityLevel), the system should redirect such request to the login page:



If a user is logged in and tries to access a page that should not be allowed per their SecurityLevel : notify the user “Page not found”.



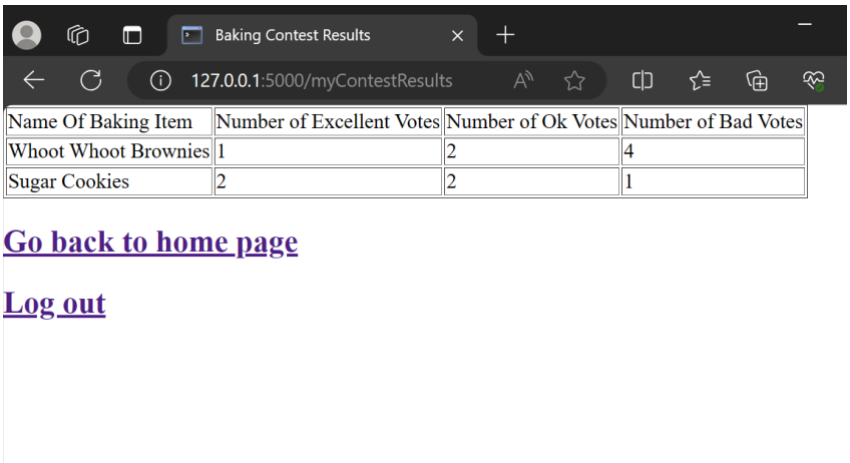
You may need to validate access in both the python script and the html template if necessary.

The Show my Contest Entry Results page **s**hould have

- A table displaying the following information for the current user from the Baking Contest Entry table: NameOfBakingItem, NumExcellentVotes, NumOkVotes, and NumBadVotes

- A Go Back to Home Page link
- A Logout link

Following is an example of this page:



The Add a Contest Entry page **s**hould have

- a label and input text field for each possible attribute other than the ids: Name of Baking Item, Number of Excellent Votes, Number of OK Votes, and Number of Bad Votes.
- Submit button

When the Submit button is clicked: the system should validate the values entered by the user.

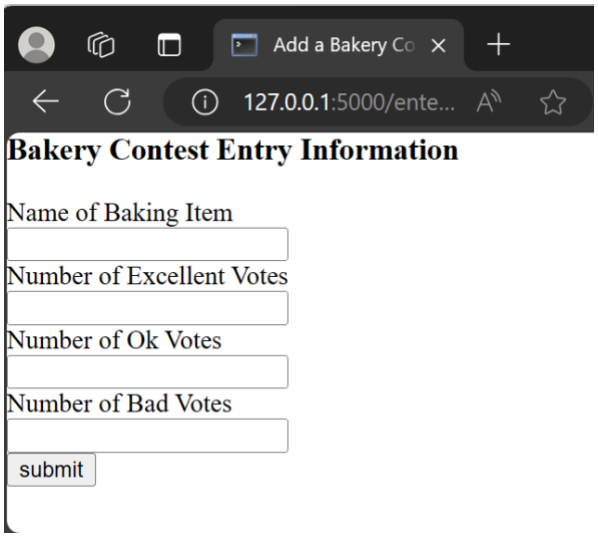
If all values are valid, then a record is added to the Baking Contest Entry table with the values entered by the user and the user id of the user currently logged in then a record added message is sent to the result page to display to the user. If some of the values are invalid, an error message is created indicating all the input errors and displayed to the user in the result page.

The validation is similar to that in the *Add a new Baking Contest User page* in Assignment 5. Please refer to the results page in Assignment 5 for potential images.

Input Validation Rules:

- The Name of Baking Item is not empty and does not only contain spaces.
- The Num Excellent Votes is an integer greater than or equal to 0 (Hint: Python - How to validate that a variable stores a numeric value).
- The Num Ok Votes is an integer greater than or equal to 0 (Hint: Python - How to validate that a variable stores a numeric value).
- The Num Bad Votes is an integer greater than or equal to 0 (Hint: Python - How to validate that a variable stores a numeric value).

Following is an example of this page:



We have now described all of the extensions in this assignment over Assignment 5. After you implement the extension, the next step is to partially encrypt some more sensitive fields in the database so as to have some data confidentiality in the system. In particular, the Baking Contest People table must maintain the following attributes for each contest participant: UserId, Name, Age, PhNum, SecurityLevel, and LoginPassword where UserId is the primary key. Among the attributes, Name, PhNum, and LoginPassword are sensitive and will be encrypted in the database in this assignment. You can use the Python module (Encryption.py) that we discussed in the class for this purpose. You may use another cryptography library, but linprog must have the library that you use. Note that besides modifying the Python scripts and html templates for the website, you also need to modify the code for database creation to store the encrypted fields. The requirement is that

1. Three fields, Name, PhNum, and LoginPassword must be encrypted in the SQLite database using an external encryption library.
2. The website must have the same behavior as the site with no encryption of the fields in the database.

The website front end will encrypt and decrypt data to and from the database. The changes to your Flask website to realize this functionality include (but not limited to) the following:

- Login page: Data values for UserName, UserPhNum, and LoginPassword should be encrypted before they are used to query the table.
- Add a new Baking Contest User page: data values for Name, PhNum, and LoginPassword should be encrypted before they are added to the table.
- List Users page: data values for the following fields Name, PhNum, and LoginPassword should be decrypted after they are pulled from the database table.

Due date: Dec. 4, 2025, 11:59pm

Submission: Put all of your programs and files that are needed to run the website as well as supporting documents in a tar file. Name the tar file lastname_firstinitial_hardenedflaskwebsite.tar, and submit through canvas. Make sure that your website works on linprog before you submit your program. If your website with encrypted database is not completely functional, your submission should contain two subdirectories, one for the website without encryption, and one for the website with encryption (which is incomplete).

Grading Policy (total 70 points, built-in 5 extra points):

- Include the basic header for assignments in a README.txt file in the root directory of your website. The README.txt should also include clear instruction about how to initialize your website (e.g. run what scripts to initialize the database), how to start the website, how to login as users of different security levels (username and password), and the status of your website (which part is not completely functional, whether the website is fully functional with encrypted database fields, etc) (10 points).

- Python scripts to initialize each of the tables in your database. In particular, the Python script to initialize the Baking Contest People table must meet the following requirements:
    - You must have a drop table command at the beginning of your script to drop the table in your database.
    - You must have a create table statement.
    - Your table must contain the following attributes: UserId, Name, Age, PhNum, SecurityLevel, and LoginPassword.
    - The fields Name, PhNum, and LoginPassword must be encrypted in the database.
    - The table create script must properly define UserId as the primary key.
    - Your script must run to completion without errors.
    - Your script must create at least 3 users, one for each security level for testing purposes.
    - At the end of the script, you must display all data in your table. In the display, Name, PhNum, and LoginPassword must be in unencrypted.

    Scripts to initialize other tables should add a few entries to the tables for testing purposes. (5 points)

- Login page, logout link, and session function correctly (10 points). You will get 8 points if the page is functional, but does not work with the database with encryption (or your database does not have all of the three encrypted fields).
- 3 different types of homepages (corresponding to SecurityLevel 1, 2, 3) (15 points). 5 points for each page. You will get 4 points if the page is functional, but does not work with the database with encryption (or your database does not have all of the three encrypted fields).
- Login error and page not found error (access error) pages are triggered properly (5 points).
- Show my Contest Entry Results page functions correctly (5 points)
- Add a Contest Entry page functions correctly (5 points)
- Add Baking User page functions correctly with encryption (5 points)
- List Contest Users page functions correctly with encryption (5 points)
- Overall website operating properly (10 points) – If the website is completely functional without encryption, 7 points will be assigned.

Note:

- Make sure you develop and test thoroughly on linprog before you submit the files.
- All pages specified in Programming Assignment 5 will be tested.