National University of Computer and Emerging Sciences
Islamabad Campus

Lab 09
Secure Coding Report

SSD-Lab

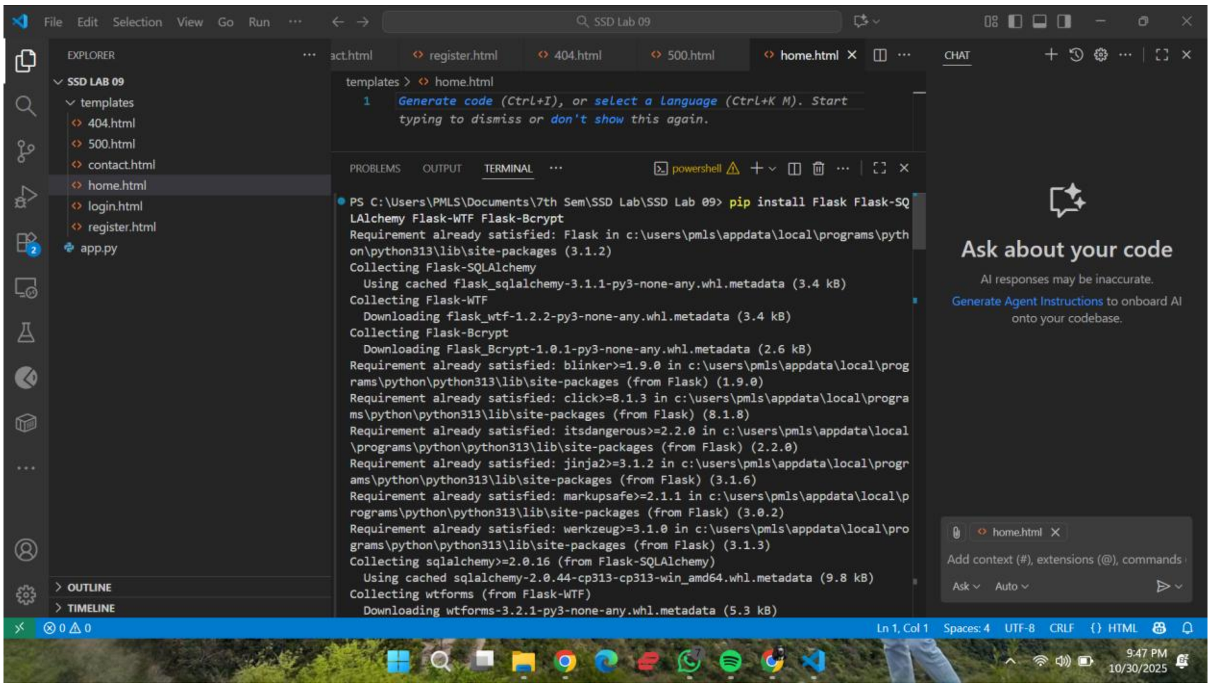
Submitted by: Abdullah Nadeem
Roll number: 22i-1597
Date: 30th October 2025

Table of Contents

1. Setup & Libraries

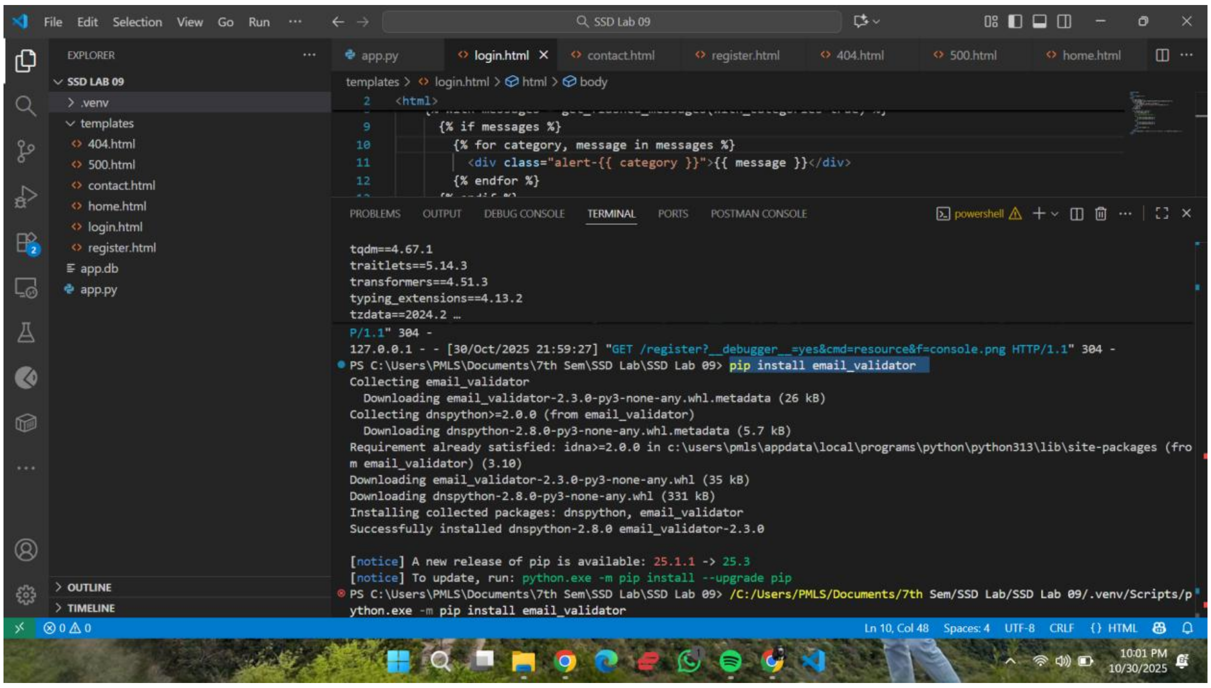
Task: Install required libraries.

Explanation: Ensured all necessary libraries for the application and security features (Flask, SQLAlchemy, Bcrypt, Flask-WTF, email_validator) were installed.


File Edit Selection View Go Run
SSD Lab 09
EXPLORER
SSD LAB 09
templates
404.html
500.html
contact.html
home.html
login.html
register.html
app.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT TERMINAL
powershell
CHAT
Ask about your code
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Add context (#), extensions (@), commands
Ask Auto
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} HTML
9:47 PM 10/30/2025

```
Generate code (Ctrl+I), or select a language (Ctrl+K M). Start
typing to dismiss or don't show this again.
```

```
PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> pip install Flask Flask-SQ
LAlchemy Flask-WTF Flask-Bcrypt
Requirement already satisfied: Flask in c:\users\pmls\appdata\local\programs\pyth
on\python313\lib\site-packages (3.1.2)
Collecting Flask-SQLAlchemy
  Using cached flask_sqlalchemy-3.1.1-py3-none-any.whl.metadata (3.4 kB)
Collecting Flask-WTF
  Downloading flask_wtf-1.2.2-py3-none-any.whl.metadata (3.4 kB)
Collecting Flask-Bcrypt
  Downloading Flask_Bcrypt-1.0.1-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: blinker>=1.9.0 in c:\users\pmls\appdata\local\prog
rams\python\python313\lib\site-packages (from Flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in c:\users\pmls\appdata\local\progra
ms\python\python313\lib\site-packages (from Flask) (8.1.8)
Requirement already satisfied: itsdangerous>=2.2.0 in c:\users\pmls\appdata\local
\programs\python\python313\lib\site-packages (from Flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in c:\users\pmls\appdata\local\progr
ams\python\python313\lib\site-packages (from Flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in c:\users\pmls\appdata\local\p
rograms\python\python313\lib\site-packages (from Flask) (3.0.2)
Requirement already satisfied: werkzeug>=3.1.0 in c:\users\pmls\appdata\local\pro
grams\python\python313\lib\site-packages (from Flask) (3.1.3)
Collecting sqlalchemy>=2.0.16 (from Flask-SQLAlchemy)
  Using cached sqlalchemy-2.0.44-cp313-cp313-win_amd64.whl.metadata (9.8 kB)
Collecting wtforms (from Flask-WTF)
  Downloading wtforms-3.2.1-py3-none-any.whl.metadata (5.3 kB)
```



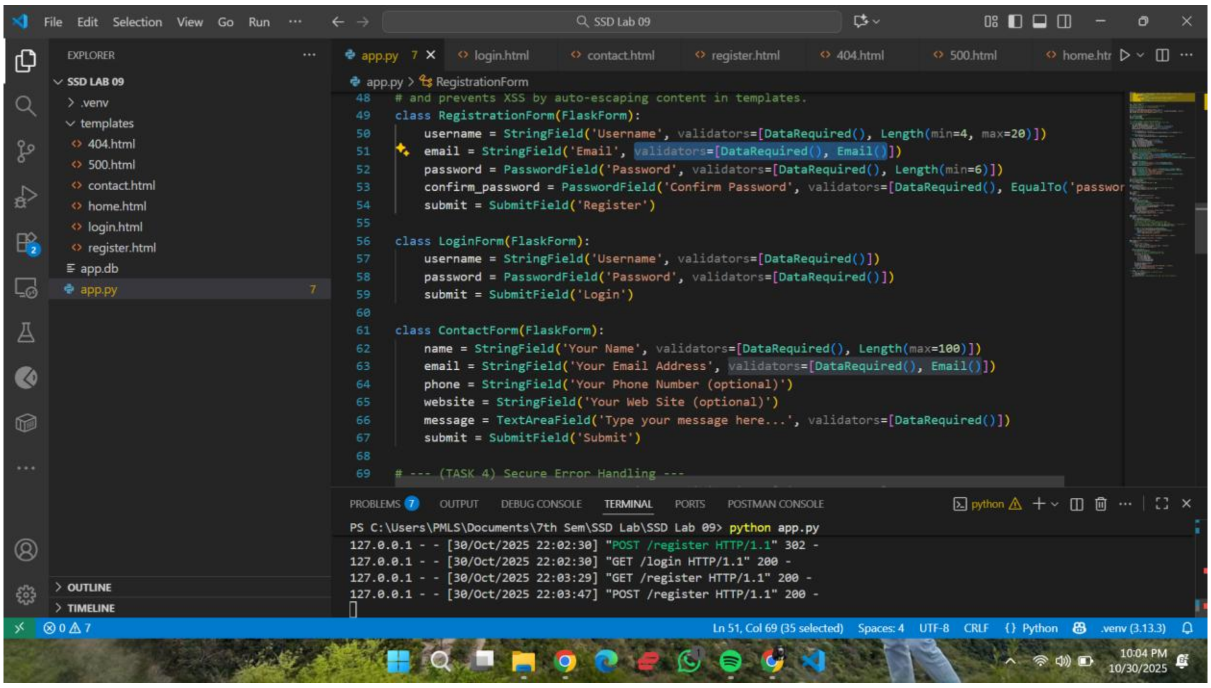
```
<html>
    {% if messages %}
      {% for category, message in messages %}
        <div class="alert-{{ category }}">{{ message }}</div>
      {% endfor %}
```

```
tqdm==4.67.1
traitlets==5.14.3
transformers==4.51.3
typing_extensions==4.13.2
tzdata==2024.2 …
P/1.1" 304 -
127.0.0.1 - - [30/Oct/2025 21:59:27] "GET /register?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> pip install email_validator
Collecting email_validator
  Downloading email_validator-2.3.0-py3-none-any.whl.metadata (26 kB)
Collecting dnspython>=2.0.0 (from email_validator)
  Downloading dnspython-2.8.0-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: idna>=2.0.0 in c:\users\pmls\appdata\local\programs\python\python313\lib\site-packages (fro
m email_validator) (3.10)
Downloading email_validator-2.3.0-py3-none-any.whl (35 kB)
Downloading dnspython-2.8.0-py3-none-any.whl (331 kB)
Installing collected packages: dnspython, email_validator
Successfully installed dnspython-2.8.0 email_validator-2.3.0

[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> /C:/Users/PMLS/Documents/7th Sem/SSD Lab/SSD Lab 09/.venv/Scripts/p
ython.exe -m pip install email_validator
```

2. Secure Input Handling (Task 1)

Task: Validate and sanitize all user input.

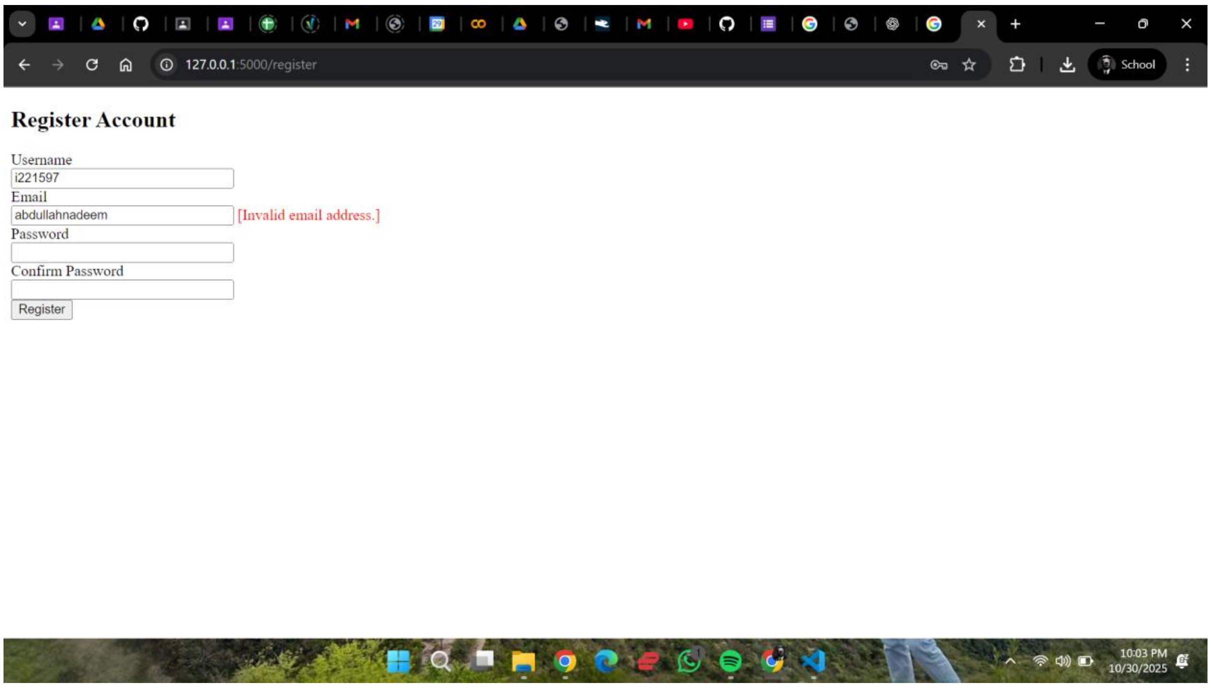
Explanation: Used Flask-WTF and wtforms.validators to enforce rules (like DataRequired, Email, Length) on all forms, preventing invalid data and XSS.



```python
# and prevents XSS by auto-escaping content in templates.
class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=4, max=20)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('passwor
    submit = SubmitField('Register')

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Login')

class ContactForm(FlaskForm):
    name = StringField('Your Name', validators=[DataRequired(), Length(max=100)])
    email = StringField('Your Email Address', validators=[DataRequired(), Email()])
    phone = StringField('Your Phone Number (optional)')
    website = StringField('Your Web Site (optional)')
    message = TextAreaField('Type your message here...', validators=[DataRequired()])
    submit = SubmitField('Submit')

# --- (TASK 4) Secure Error Handling ---
```

```
PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> python app.py
127.0.0.1 - - [30/Oct/2025 22:02:30] "POST /register HTTP/1.1" 302 -
127.0.0.1 - - [30/Oct/2025 22:02:30] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [30/Oct/2025 22:03:29] "GET /register HTTP/1.1" 200 -
127.0.0.1 - - [30/Oct/2025 22:03:47] "POST /register HTTP/1.1" 200 -
```

(Code for 'RegistrationForm' and 'ContactForm' in app.py, highlighting the validators= parts)



127.0.0.1:5000/register

Register Account

Username
i221597
Email
abdullahnadeem [Invalid email address.]
Password

Confirm Password

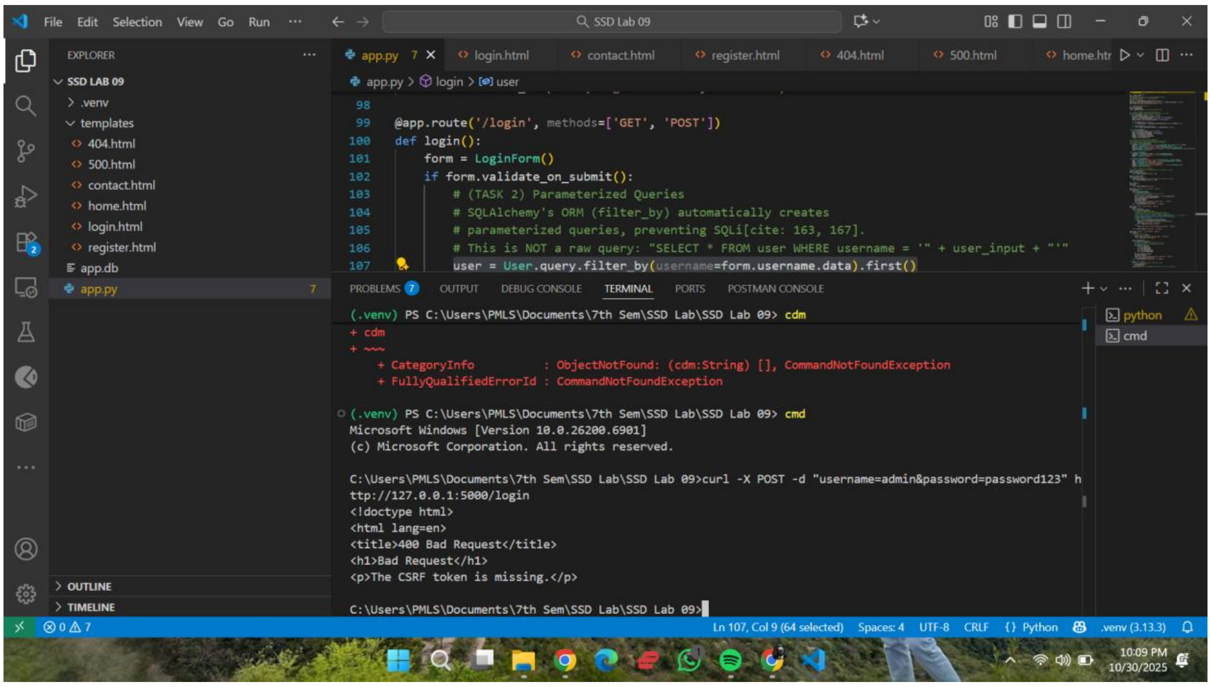
Register

(The registration page showing the 'Invalid email address' error after I tried to submit bad data)

3. Parameterized Queries (Task 2)

Task: Prevent SQL Injection.

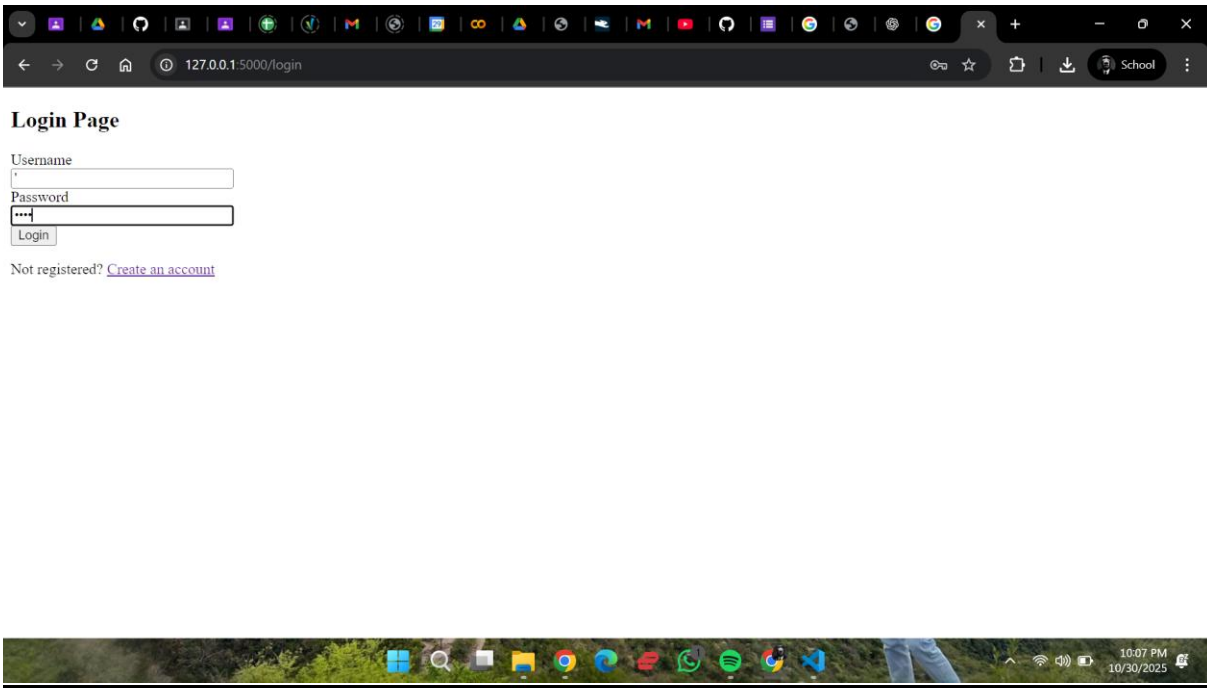
Explanation: Used the SQLAlchemy ORM (.filter_by()) which automatically creates parameterized queries, treating all user input as text, not as SQL commands.


File Edit Selection View Go Run
SSD Lab 09
EXPLORER
SSD LAB 09
.venv
templates
404.html
500.html
contact.html
home.html
login.html
register.html
app.db
app.py
app.py > login > user
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
OUTLINE
TIMELINE
Ln 107, Col 9 (64 selected) Spaces: 4 UTF-8 CRLF {} Python .venv (3.13.3)

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        # (TASK 2) Parameterized Queries
        # SQLAlchemy's ORM (filter_by) automatically creates
        # parameterized queries, preventing SQLi[cite: 163, 167].
        # This is NOT a raw query: "SELECT * FROM user WHERE username = '" + user_input + "'"
        user = User.query.filter_by(username=form.username.data).first()
```

```
(.venv) PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> cdm
+ cdm
+ ~~~
    + CategoryInfo          : ObjectNotFound: (cdm:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

(.venv) PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> cmd
Microsoft Windows [Version 10.0.26200.6901]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09>curl -X POST -d "username=admin&password=password123" h
ttp://127.0.0.1:5000/login
<!doctype html>
<html lang=en>
<title>400 Bad Request</title>
<h1>Bad Request</h1>
<p>The CSRF token is missing.</p>

C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09>
```

(Login route code in app.py, highlighting the User.query.filter_by(...) line)


127.0.0.1:5000/login

Login Page

Username

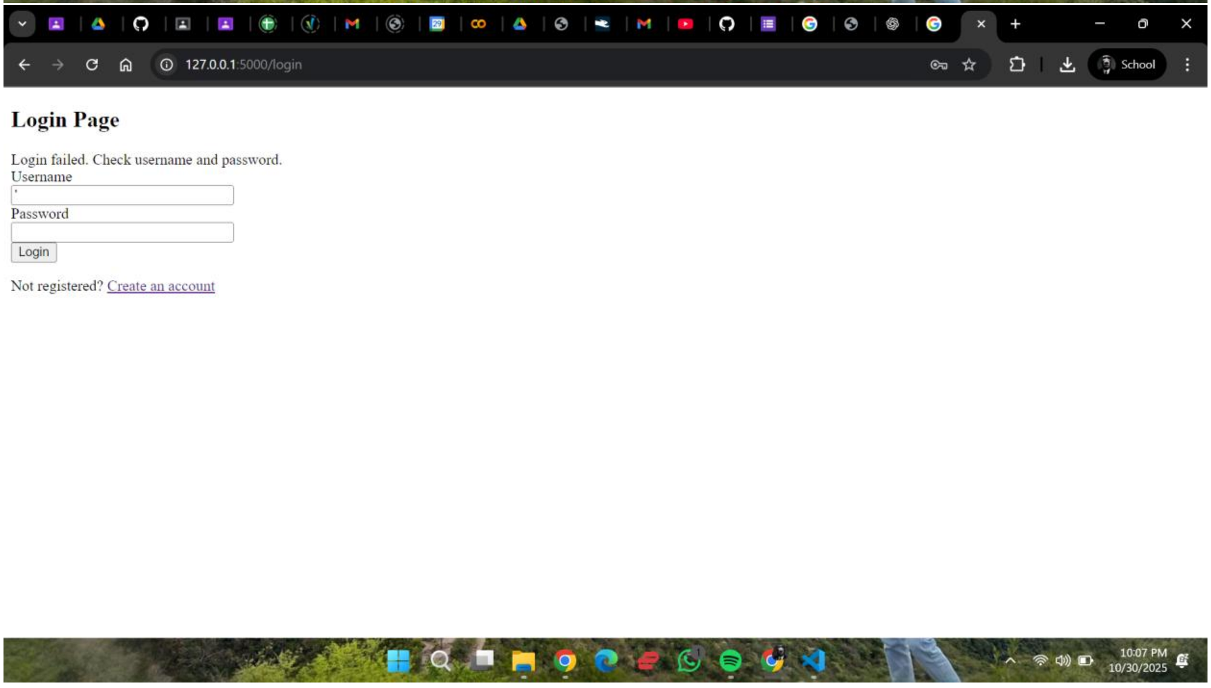
'

Password

••••

Login

Not registered? Create an account


127.0.0.1:5000/login

Login Page

Login failed. Check username and password.

Username

'

Password

Login

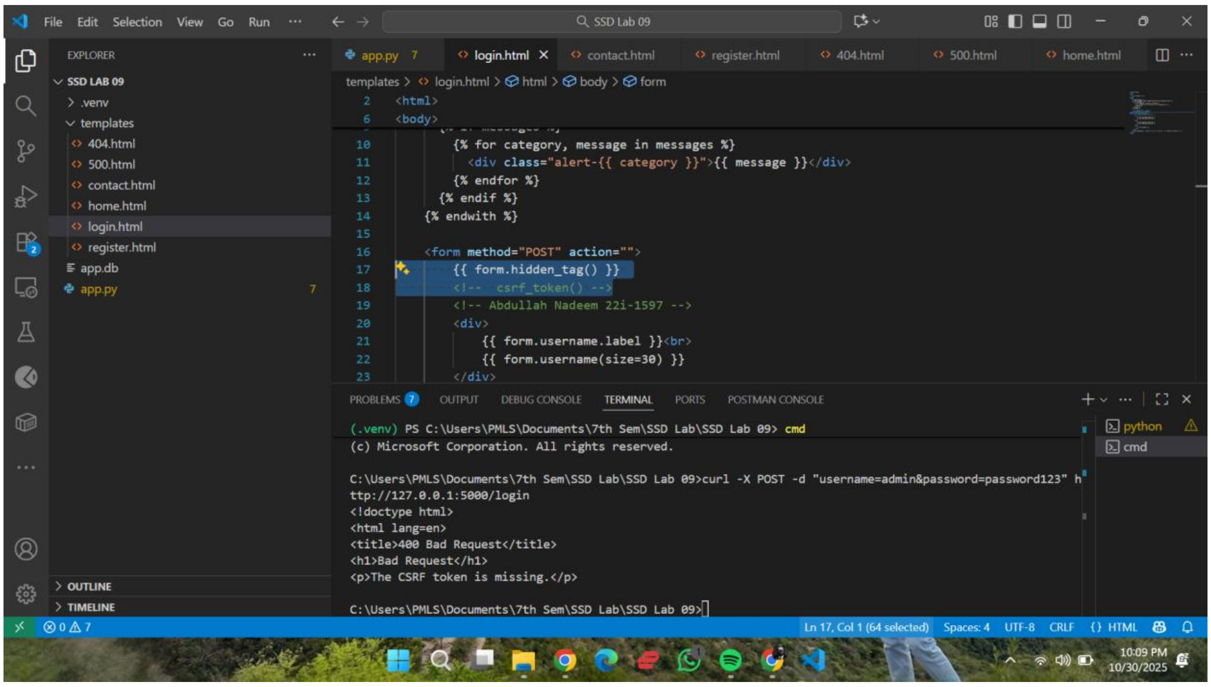
Not registered? Create an account

(The login page showing SQLi attack (' OR '1'='1) failing with the "Login failed" error message)

4. Session Management & CSRF Protection (Task 3)

Task: Implement CSRF tokens to protect all forms.

Explanation: Used Flask-WTF to automatically generate a unique, hidden csrf_token for every form, which is validated on submission to prevent CSRF attacks.


File Edit Selection View Go Run
SSD Lab 09
EXPLORER
SSD LAB 09
.venv
templates
404.html
500.html
contact.html
home.html
login.html
register.html
app.db
app.py
OUTLINE
TIMELINE
app.py login.html contact.html register.html 404.html 500.html home.html
templates > login.html > html > body > form
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
python
cmd
Ln 17, Col 1 (64 selected) Spaces: 4 UTF-8 CRLF {} HTML

```
<html>
<body>
        {% for category, message in messages %}
          <div class="alert-{{ category }}">{{ message }}</div>
        {% endfor %}
      {% endif %}
    {% endwith %}

    <form method="POST" action="">
        {{ form.hidden_tag() }}
        <!--  csrf_token() -->
        <!-- Abdullah Nadeem 22i-1597 -->
        <div>
            {{ form.username.label }}<br>
            {{ form.username(size=30) }}
        </div>
```

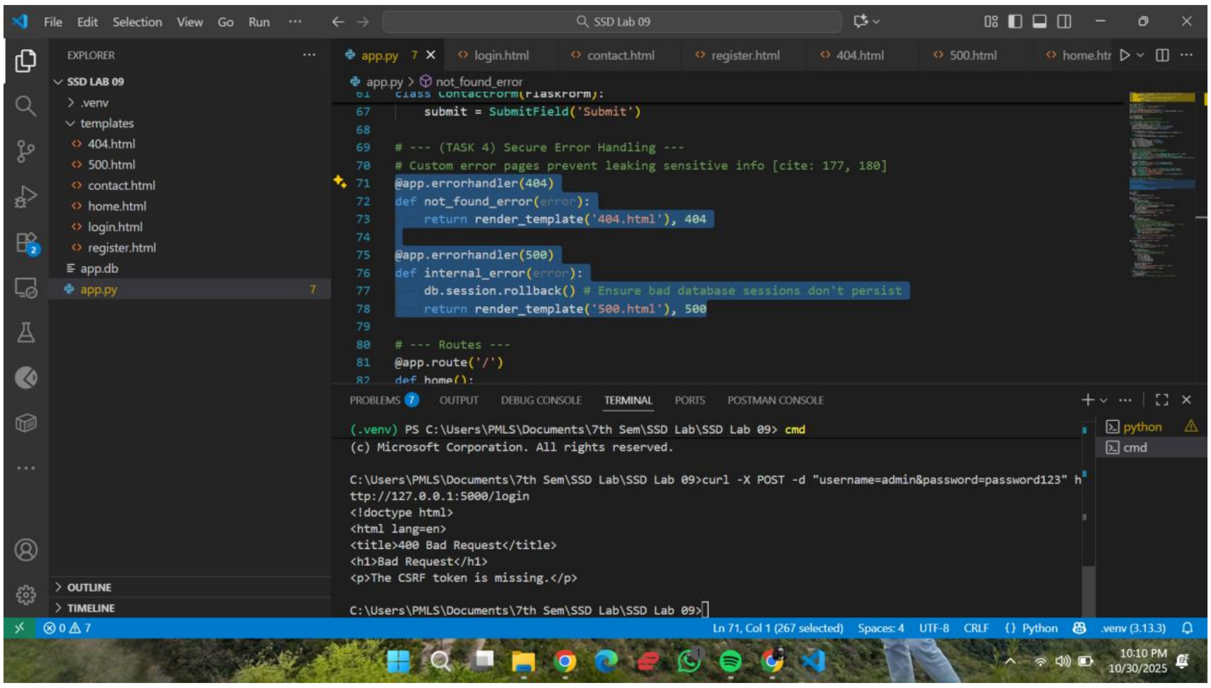
```
(.venv) PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> cmd
(c) Microsoft Corporation. All rights reserved.

C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09>curl -X POST -d "username=admin&password=password123" h
ttp://127.0.0.1:5000/login
<!doctype html>
<html lang=en>
<title>400 Bad Request</title>
<h1>Bad Request</h1>
<p>The CSRF token is missing.</p>

C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09>
```

(login.html template code, highlighting the {{ form.hidden_tag() }} line plus the terminal showing the curl command failing with a "CSRF token missing" or "Bad Request" error)

5. Secure Error Handling (Task 4)

Task: Create custom error pages to avoid information disclosure.

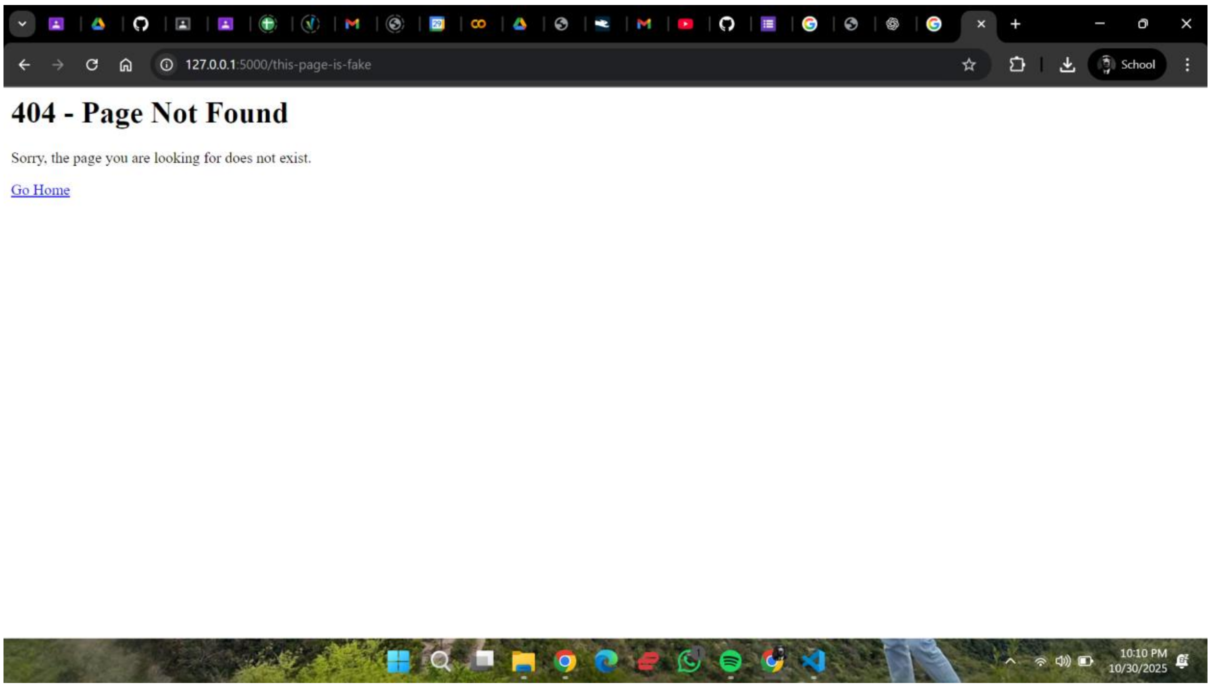
Explanation: Implemented custom @errorhandler functions for 404 and 500 errors to show a user-friendly page instead of leaking stack traces or server info.



```python
class ContactForm(FlaskForm):
    submit = SubmitField('Submit')

# --- (TASK 4) Secure Error Handling ---
# Custom error pages prevent leaking sensitive info [cite: 177, 180]
@app.errorhandler(404)
def not_found_error(error):
    return render_template('404.html'), 404

@app.errorhandler(500)
def internal_error(error):
    db.session.rollback() # Ensure bad database sessions don't persist
    return render_template('500.html'), 500

# --- Routes ---
@app.route('/')
def home():
```

```
(.venv) PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> cmd
(c) Microsoft Corporation. All rights reserved.

C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09>curl -X POST -d "username=admin&password=password123" h
ttp://127.0.0.1:5000/login
<!doctype html>
<html lang=en>
<title>400 Bad Request</title>
<h1>Bad Request</h1>
<p>The CSRF token is missing.</p>

C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09>
```

(@app.errorhandler(404) and @app.errorhandler(500) functions in app.py)



127.0.0.1:5000/this-page-is-fake

404 - Page Not Found

Sorry, the page you are looking for does not exist.

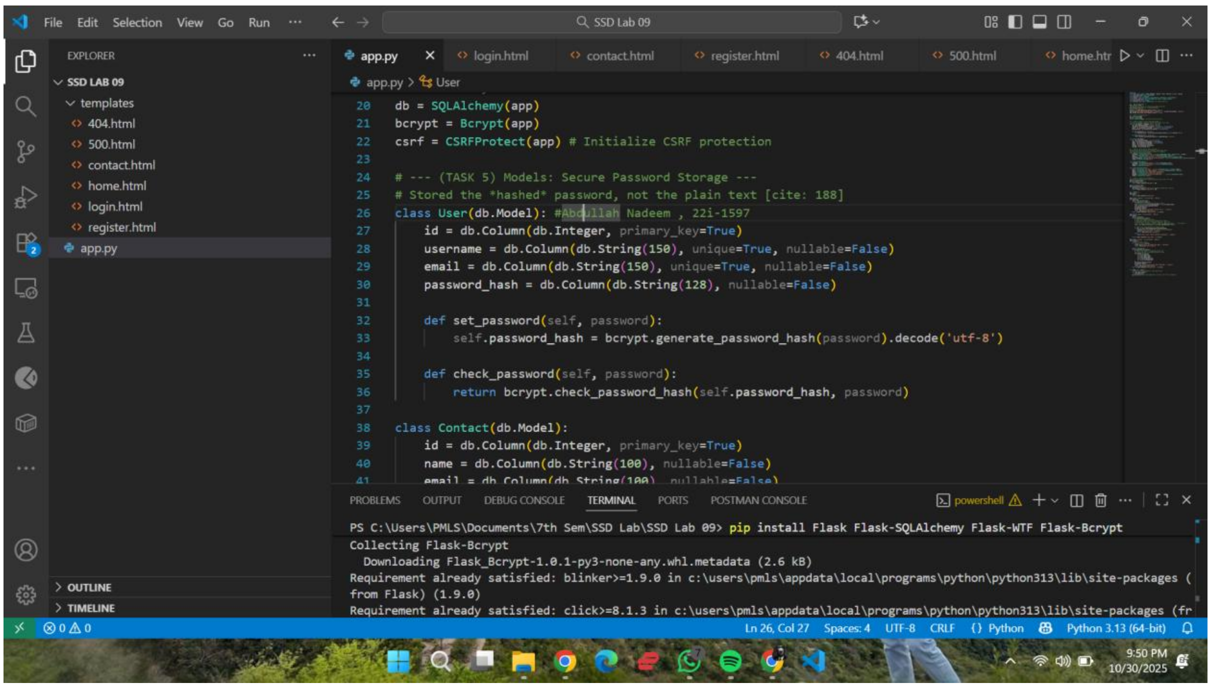
Go Home

(The custom "404 - Page Not Found" page shown in the browser)

6. Secure Password Storage (Task 5)

Task: Securely hash and store user passwords.

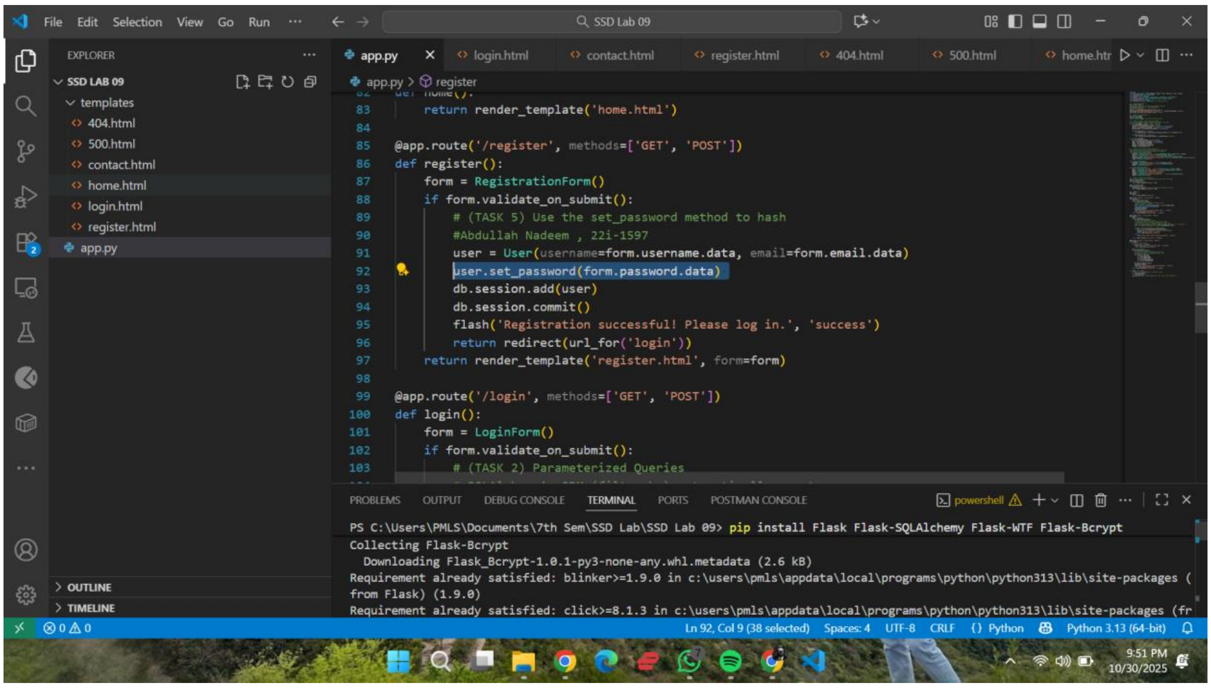
Explanation: Used Flask-Bcrypt to generate a strong, salted hash of the user's password (set_password) and safely compare it during login (check_password).



```python
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
csrf = CSRFProtect(app) # Initialize CSRF protection

# --- (TASK 5) Models: Secure Password Storage ---
# Stored the *hashed* password, not the plain text [cite: 188]
class User(db.Model): #Abdullah Nadeem , 22i-1597
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)

    def set_password(self, password):
        self.password_hash = bcrypt.generate_password_hash(password).decode('utf-8')

    def check_password(self, password):
        return bcrypt.check_password_hash(self.password_hash, password)

class Contact(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), nullable=False)
```

```
PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> pip install Flask Flask-SQLAlchemy Flask-WTF Flask-Bcrypt
Collecting Flask-Bcrypt
  Downloading Flask_Bcrypt-1.0.1-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: blinker>=1.9.0 in c:\users\pmls\appdata\local\programs\python\python313\lib\site-packages (
from Flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in c:\users\pmls\appdata\local\programs\python\python313\lib\site-packages (fr
```

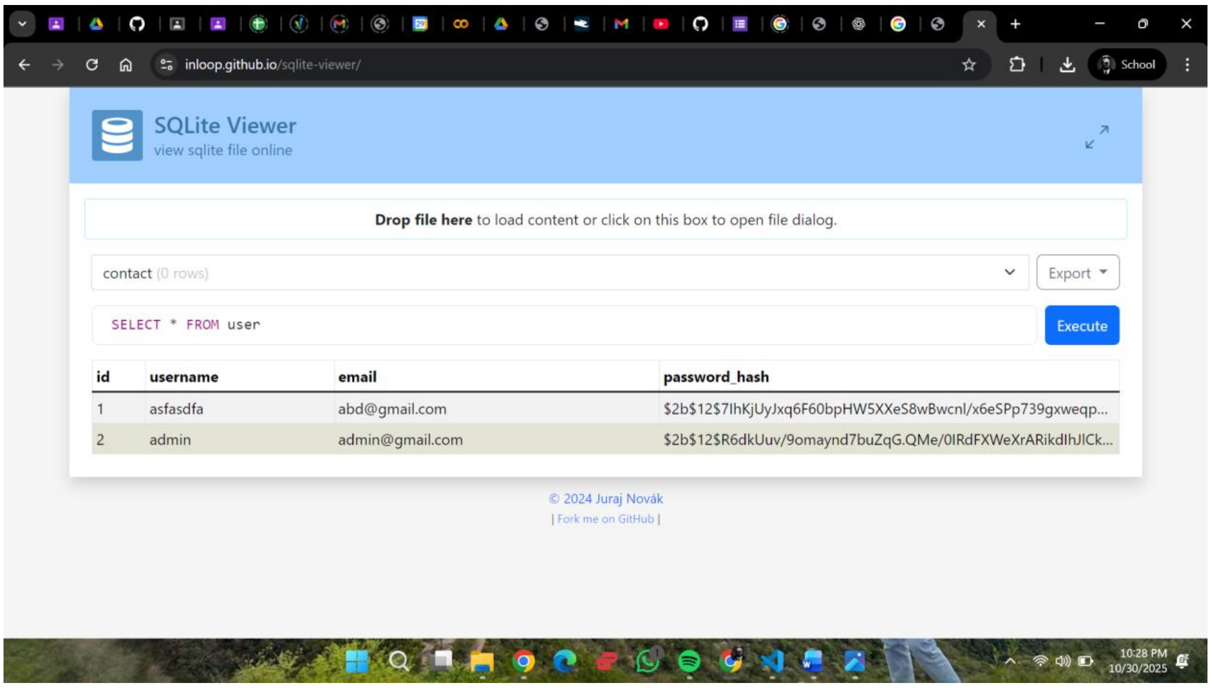
(User model code in app.py, highlighting the password_hash, set_password, and check_password parts)



```python
    return render_template('home.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        # (TASK 5) Use the set_password method to hash
        #Abdullah Nadeem , 22i-1597
        user = User(username=form.username.data, email=form.email.data)
        user.set_password(form.password.data)
        db.session.add(user)
        db.session.commit()
        flash('Registration successful! Please log in.', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', form=form)

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        # (TASK 2) Parameterized Queries
```

```
PS C:\Users\PMLS\Documents\7th Sem\SSD Lab\SSD Lab 09> pip install Flask Flask-SQLAlchemy Flask-WTF Flask-Bcrypt
Collecting Flask-Bcrypt
  Downloading Flask_Bcrypt-1.0.1-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: blinker>=1.9.0 in c:\users\pmls\appdata\local\programs\python\python313\lib\site-packages (
from Flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in c:\users\pmls\appdata\local\programs\python\python313\lib\site-packages (fr
```

(Register route code, highlighting the user.set_password(form.password.data) line)



SQLite Viewer
view sqlite file online

Drop file here to load content or click on this box to open file dialog.

contact (0 rows)

```sql
SELECT * FROM user
```

id	username	email	password_hash
1	asfasdfa	abd@gmail.com	$2b$12$7IhKjUyJxq6F60bpHW5XXeS8wBwcnl/x6eSPp739gxweqp...
2	admin	admin@gmail.com	$2b$12$R6dkUuv/9omaynd7buZqG.QMe/0IRdFXWeXrARikdIhJICk...

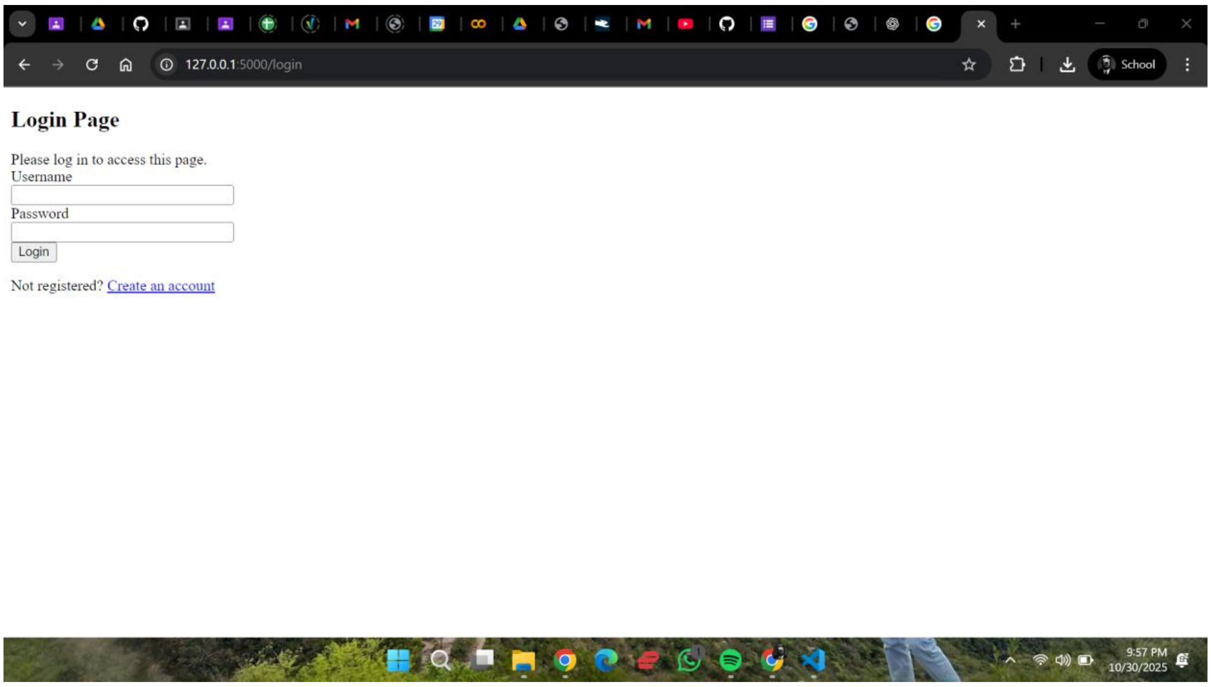
© 2024 Juraj Novák
| Fork me on GitHub |

(The passwords in the database are encrypted)

7. Application Screenshots

Task: Show the functional login page.

Explanation: This is the main login page of the completed Flask application.



Login Page

Please log in to access this page.
Username

Password

Login

Not registered? Create an account

(The login.html page rendered in the browser)