5.2. Student Handout

Student Handout: User Authentication and Deploying Flask Applications

Welcome to this session on User Authentication and Deploying Flask Applications. This handout provides a concise overview of the key concepts and examples to help you understand and apply these skills in your web development projects.

Part 1: User Authentication in Flask

What is Authentication?

Authentication is the process of verifying the identity of a user trying to access a web application. It ensures that only authorized users can access certain parts of the website.

Why is Authentication Important?

Authentication is crucial for protecting sensitive information and ensuring that only authorized users can access personal data.

Setting Up User Registration and Login Forms in Flask

1. **User Registration Form**: Collects user information like username, email, and password. The data is sent to the server and stored in a database.

```
from flask import Flask, render_template, request, redirect, url_for

from werkzeug.security import generate_password_hash



app = Flask(__name__)



@app.route('/register', methods=['GET', 'POST'])
```

```
def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

hashed_password = generate_password_hash(password)

# Save the username and hashed password to the database

return redirect(url_for('login'))

return render_template('register.html')
```

2. **Login Form**: Asks for the user's username and password. The server checks if the credentials match the database.

```
@app.route('/login', methods=['POST'])

def login():

user = User.query.filter_by(username=request.form['username']).first()

if user and check_password_hash(user.password, request.form['password']):

login_user(user)

return redirect(url_for('dashboard'))

return 'Invalid credentials'
```

3. **Securing Routes with Login-Required Decorators**: Restrict access to certain parts of the website to logged-in users.

```
from flask_login import login_required
```

```
@app.route('/dashboard')

@login_required

def dashboard():

return "Welcome to your dashboard!"
```

Password Hashing and Security Best Practices

- **Password Hashing**: Use hashing to store passwords securely. This prevents exposure of plain text passwords if the database is compromised.

```
from werkzeug.security import generate_password_hash, check_password_hash



# Hashing a password

hashed_password = generate_password_hash('mysecretpassword')



# Checking a password

check_password_hash(hashed_password, 'mysecretpassword') # Returns True if
the password matches
```

Introduction to Flask-Login for Session Management

- **Session Management**: Manage user sessions to keep track of logged-in users.

```
from flask_login import LoginManager, login_user, logout_user,
login_required



login_manager = LoginManager()
```

```
login_manager.init_app(app)



@login_manager.user_loader

def load_user(user_id):

return User.get(user_id)



@app.route('/logout')

@login_required

def logout():

logout_user()

return redirect(url_for('login'))
```

Part 2: Deploying Flask Applications

What is Deployment?

Deployment is the process of making your Flask app available on the internet by hosting it on a server.

Hosting Options

1. **Heroku**: A cloud platform that simplifies deploying web applications with a free tier.
2. **DigitalOcean**: Provides more control over your server, suitable for more advanced users.
3. **AWS (Amazon Web Services)**: Offers a wide range of services for deploying applications, suitable for large-scale projects.

Setting Up a Production Environment

- **Gunicorn**: A Python web server for running Flask apps.
- **Nginx**: A web server that handles incoming requests and forwards them to Gunicorn.

Managing Environment Variables and Secrets

- Use environment variables to store sensitive information like database credentials and API keys securely.

```
import os


DATABASE_URL = os.getenv('DATABASE_URL')

SECRET_KEY = os.getenv('SECRET_KEY')
```

Conclusion

In this session, we covered:

1. **User Authentication**: Setting up user registration and login forms, securing routes, and using password hashing for security. We also introduced Flask-Login for session management.
2. **Deploying Flask Applications**: The process of deploying a Flask app, exploring hosting options, setting up a production environment, and managing environment variables.

These skills are essential for building secure and accessible web applications. Practice implementing these concepts in your projects to gain proficiency.

If you have any questions or need further clarification, feel free to ask!