PYTHON - FLASK

Presented by Group 10

CONTENTS

INTRODUCTION

INTRODUCTION

- Flask is used for building web applications in Python. It offers simplicity and a small core, allowing developers to quickly create web services and APIs.
- With Flask, you can define routes, render dynamic content using templates, and take advantage of a wide range of extensions for additional functionality.

WEB APPLICATIN FRAMEWORK

Web Application Framework represents a collection of libraries and modules.

WERKZEUG WSGI TOOLKIT

Werkzeug is a collection of libraries that can be used to create a WSGI (Web Server Gateway Interface) compatible web application in Python.

JINJA2 TEMPLATE ENGINE

Jinja2 is a popular templating engine for Python. It is used to create HTML or other markup formats that are returned to the user via an HTTP request.

FEATURES

- Contains development server and debugger
- Uses Jinja2 templating
- Support for secure cookies (client side sessions)
- Unicode-based
- Extensive documentation
- Google App Engine Compatibility
- Extensions available to enhance desired features

INSTALLATION

INSTALLATION

Prerequisites: VS Code, Python, MySQL.

Steps:

1. Install Python from python.org
2. On your system, create a folder (eg. hello_flask) and open this folder in VS Code
3. Open the terminal and type- pip install virtualenv
4. To create a virtual environment- virtualenv env
   (Note: 'env' is the name of the environment)
5. To activate the environment- env\Scripts\activate.bat
   (Note: for MacOS- source env/bin/activate)
6. The enter the command- pip install flask
7. Create the app.py file

In order to test Flask installation run this code In the editor

```
from flask import Flask
app = Flask(__name__)


@app.route('/')
def hello_world():
    return 'Hello World’
if __name__ == '__main__':
    app.run()
```

CONCEPTS

DEBUG MODE

Allows the developers to locate any possible error

Code for enabling the debug mode:

```
app.debug=True
app.run()
app.run(debug=True)
```

FLASK TEMPLATING

- Flask supports Jinja2 templating, which enables you to generate dynamic HTML content by combining static HTML templates with Python code.
- You can insert dynamic content into templates using variables provided by the server-side code.
- Error handling mechanisms are available to handle exceptions or errors that may occur during template rendering.

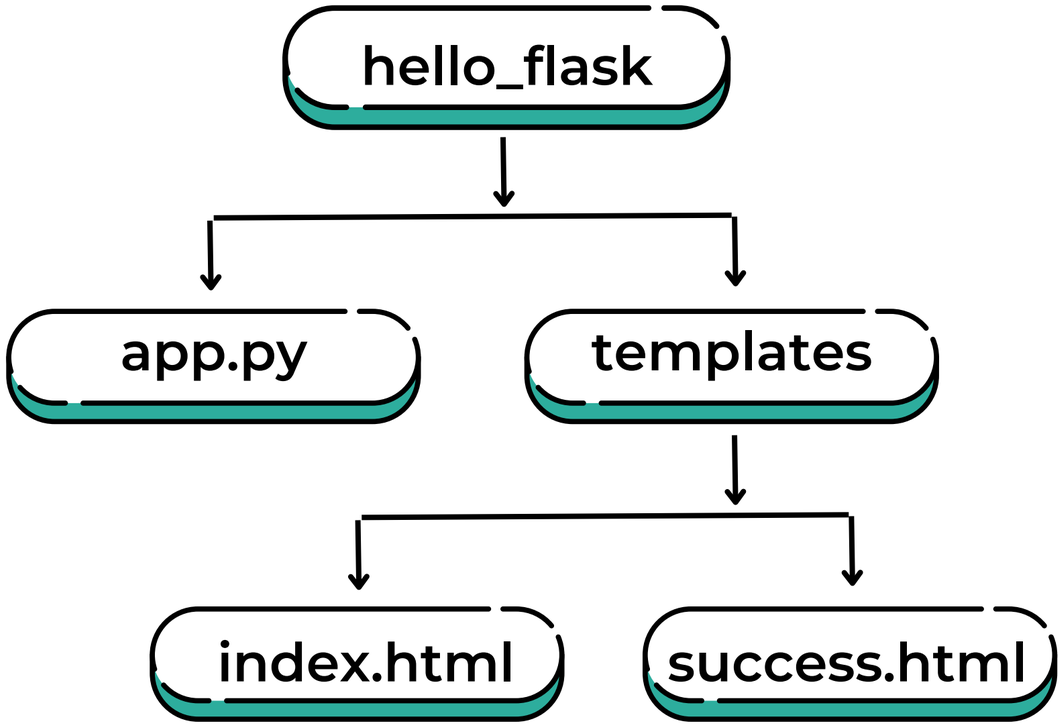
FLASK ROUTING

- The route() decorater is used to bind an url to a function.
- Routing means mapping the URLs to a specific function that will handle the logic for that URL.
- As a result when the url is mentioned in the browser , the function is executed to give the results. Here , URL '/hello' is bound to the hello_world() function.

```
@app.route(‘/hello’)
def hello_world():
    return ‘hello world’
```

EXAMPLE

EXAMPLE



index.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Flask MySQL Example</title>
</head>
<body>
 <h1>Flask MySQL Example</h1>
 <form action="/submit" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
<br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
<br>
  <input type="submit" value="Submit">
 </form>
</body>
</html>
```

EXAMPLE

hello.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Flask MySQL Example</title>
</head>
<body>
  <h1>Success!</h1>
  <p>Data has been submitted
successfully.</p>
</body>
</html>
```

To create a database:

- CREATE DATABASE your_database;
- USE your_database;
  CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL
  );

To display the databasse:

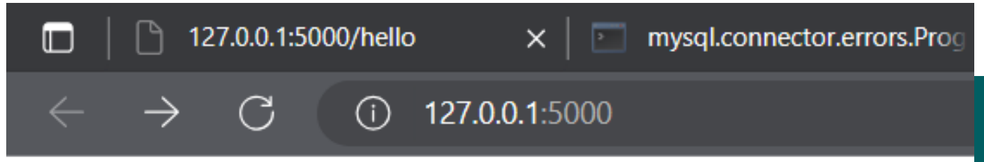
- USE your_database;
- SHOW TABLES;
- SELECT * FROM users;

pip install flask mysql-connector-python

EXAMPLE

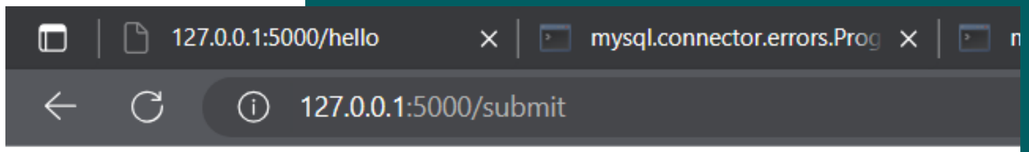
app.py

```
from flask import Flask, render_template, request
import mysql.connector
app = Flask(__name__)
# MySQL configuration
mysql_host = 'localhost'
mysql_user = 'your_username'
mysql_password = 'your_password'
mysql_database = 'your_database'
# Connect to MySQL database
db = mysql.connector.connect(
  host=mysql_host,
  user=mysql_user,
  password=mysql_password,
  database=mysql_database
)
# Define a route for the home page
@app.route('/')
def home():
  return render_template('index.html')

# Define a route for the form submission
@app.route('/submit', methods=['POST'])
def submit():
  # Get form data
  name = request.form['name']
  email = request.form['email']

  # Insert data into the database
  cursor = db.cursor()
  query = "INSERT INTO users (name, email) VALUES (%s, %s)"
  values = (name, email)
  cursor.execute(query, values)
  db.commit()

  # Close the database connection
  cursor.close()

  return render_template('success.html')

if __name__ == '__main__':
  app.run(debug=True)
```



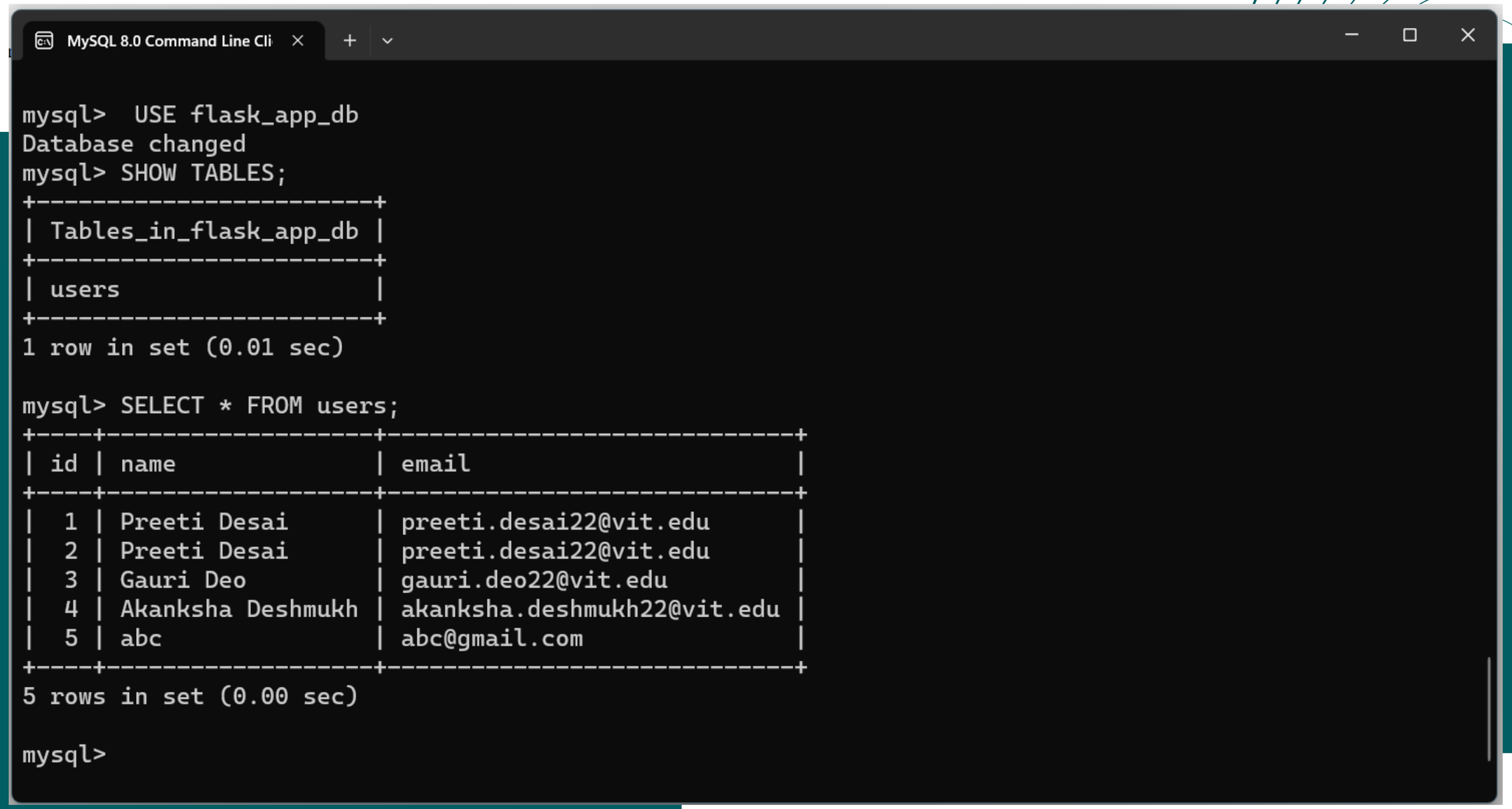
Flask MySQL Example

Name: abc
Email: abc@gmail.com
Submit



Success!

Data has been submitted successfully.


MySQL 8.0 Command Line Cli
mysql> USE flask_app_db
Database changed
mysql> SHOW TABLES;
+------------------------+
| Tables_in_flask_app_db |
+------------------------+
| users |
+------------------------+
1 row in set (0.01 sec)
mysql> SELECT * FROM users;
+----+------------------+----------------------------+
| id | name | email |
+----+------------------+----------------------------+
1	Preeti Desai	preeti.desai22@vit.edu
2	Preeti Desai	preeti.desai22@vit.edu
3	Gauri Deo	gauri.deo22@vit.edu
4	Akanksha Deshmukh	akanksha.deshmukh22@vit.edu
5	abc	abc@gmail.com
+----+------------------+----------------------------+
5 rows in set (0.00 sec)
mysql>

THANK YOU!