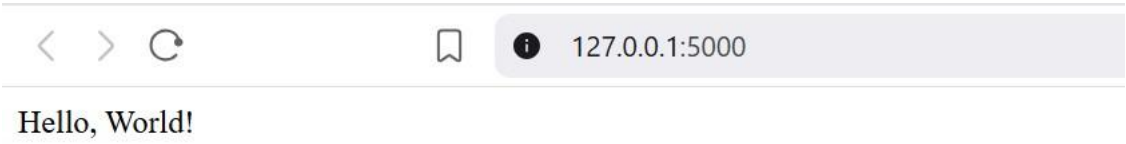
1. Create a simple view that returns "Hello, World!" and map it to a URL using Python Flask.

app.py

```
from flask import Flask

# Create Flask application
app = Flask(__name__)

# Define a route and view function
@app.route('/') def
hello():
   return "Hello, World!"

# Run the application if
__name__ == '__main__':
   app.run(debug=True)
```



2. Create a Flask view that displays a list of hyperlinks to various social media websites using a Jinja template, and map it to a URL route.

1. Project Structure

```
flask_app/
|— app.py
|— templates/
    └— social.html
```

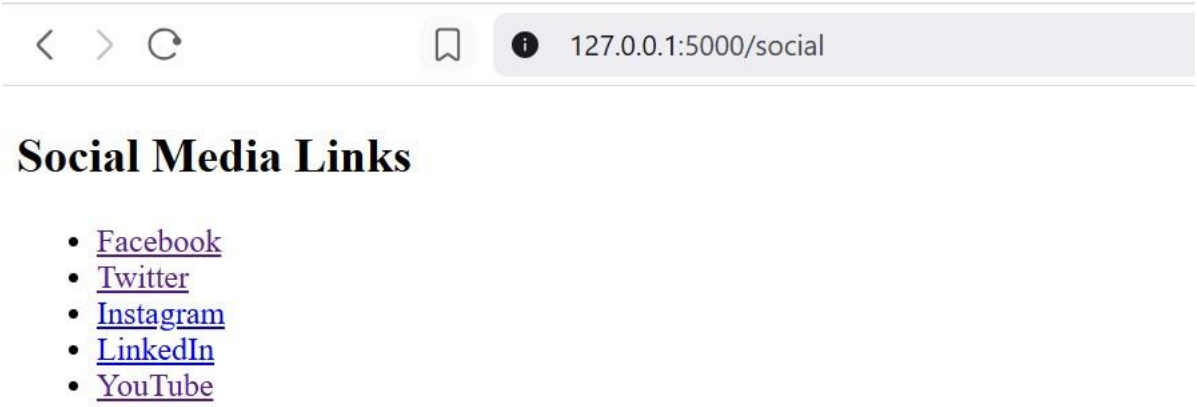
app.py

```
from flask import Flask, render_template

app = Flask(__name__)
```

```
@app.route('/social') def
social_links():
    # List of social media sites
    links = {
        "Facebook": "https://www.facebook.com",
        "Twitter": "https://www.twitter.com",
        "Instagram": "https://www.instagram.com",
        "LinkedIn": "https://www.linkedin.com",
        "YouTube": "https://www.youtube.com"
    }
    return render_template("social.html", links=links)

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

templates/social.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Social Media Links</title>
</head>
<body>
    <h2>Social Media Links</h2>
    <ul>
        {% for name, url in links.items() %}
            <li><a href="{{ url }}" target="_blank">{{ name }}</a></li>
        {% endfor %}
    </ul>
</body>
</html>
```



3. Write a Flask application that:

   1. Displays a message `"Please add a number to the URL, like /5 or /10"` when a user visits the home page (`"/"`).
   2. Accepts an integer from the URL (e.g., `/10`).
   3. Generates and returns all prime numbers up to the given integer as a string.

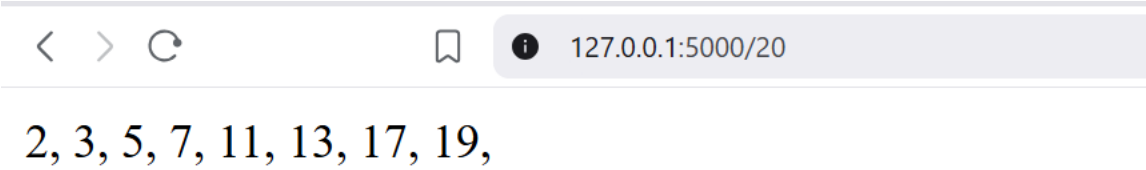
Example:

- Visiting `http://127.0.0.1:5000/10` should return:
  `2, 3, 5, 7,`

app.py

```
from flask import Flask

# Create a Flask application instance
app = Flask(__name__)

# Route for the home page
@app.route("/")
def home():
    # Message asking user to enter a number in the URL
    return "Please add a number to the URL, like /5 or /10"

# Route that accepts an integer from the URL
@app.route("/<int:number>")
def prime(number):
    primes = ""  # String to hold prime numbers

    # Loop through all numbers from 2 to 'number'
    for i in range(2, number + 1):
```

```
        # Check if 'i' is prime
        for n in range(2, (i // 2) + 1):
            if i % n == 0:   # If divisible, not a prime
                break
        else:
            # If no divisor found, it is prime → add to result string
            primes += str(i) + ", "

    # Return all prime numbers as a string
    return primes

# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```



2, 3, 5, 7, 11, 13, 17, 19,

4. Create a Flask application that:

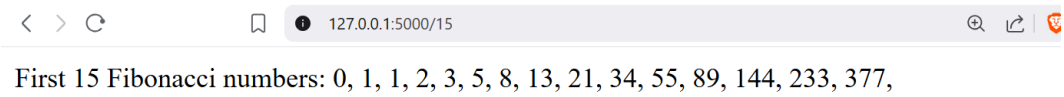
1. Displays a message `"Please add a number to the URL, like /5 or /10"` when a user visits the home page (`"/"`).
2. Accepts an integer from the URL (e.g., `/7`).
3. Generates and returns the first *N* Fibonacci numbers, where *N* is the integer passed in the URL.

Example:

- Visiting `http://127.0.0.1:5000/7` should return:
  `First 7 Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8,`

app.py

```
from flask import Flask

# Create a Flask application instance
app = Flask(__name__)

# Route for the home page
@app.route("/")
def home():
    # Message asking user to enter a number in the URL
    return "Please add a number to the URL, like /5 or /10"

# Route that accepts an integer from the URL
@app.route("/<int:number>")
def fibonacci(number):
    # String to hold Fibonacci numbers
    fibs = "First " + str(number) + " Fibonacci numbers: "

    # Initialize first two Fibonacci numbers
    fib1, fib2 = 0, 1

    # Generate Fibonacci sequence
    for i in range(number):
        fibs += str(fib1) + ", "
        fib1, fib2 = fib2, fib1 + fib2

    # Return the Fibonacci sequence as a string
    return fibs

# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```



5. Create a Flask application that:

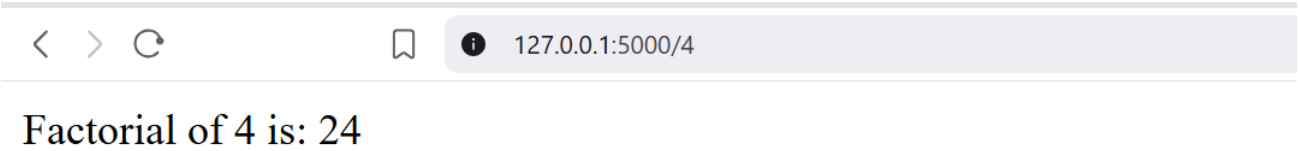
1. Displays a message **"Please add a number to the URL, like /5 or /10"** when a user visits the home page (/).
2. Accepts an integer from the URL (e.g., /6).
3. Calculates the **factorial** of the given number and displays the result in the browser.

Example:

- Visiting http://127.0.0.1:5000/6 should return:
  Factorial of 6 is: 720

app.py

```
from flask import Flask

# Create a Flask application instance
app = Flask(__name__)

# Route for the home page
@app.route("/")
def home():
    # Message asking user to enter a number in the URL
    return "Please add a number to the URL, like /5 or /10"

# Route that accepts an integer from the URL
@app.route("/<int:number>")
def factorial(number):
    # Calculate factorial
    fact = 1
    for i in range(1, number + 1):
        fact *= i

    # Return the factorial result as a string
    return f"Factorial of {number} is: {fact}"

# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```



Factorial of 4 is: 24

6. Create a Flask application to navigate between multiple links in a webpage.

1.Program structure

```
flask_app/
│
├── app.py
│
└── templates/
    ├── index.html
    ├── about.html
    └── social.html
```

app.py

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

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

```
def social():
    return render_template('social.html')

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

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Home Page</title>
</head>
<body>
  <div >
     <nav>
       <a href="/">Home</a> |
       <a href="/about">About</a> |
       <a href="/social">Social</a>
      </nav>
    <h1>Hello...</h1>
    <p>This is the Home Page.</p>
  </div>
</body>
</html>
```

About.html

```
<!DOCTYPE html>
<html>
<head>
   <title>About Page</title>
</head>
<body>

      <nav>
         <a href="/">Home</a> |
         <a href="/about">About</a> |
         <a href="/social">Social</a>
        </nav>

      <h1>About Us</h1>
      <p>This is the About Page.</p>
</body>
</html>
```

Social.html

```
<!DOCTYPE html>
<html>
<head>
  <title>links</title>
</head>
<body>
  <div class="box">
    <nav>
      <a href="/">Home</a> |
      <a href="/about">About</a> |
      <a href="/social">Social</a>
     </nav>
    <h1>Our Logo:</h1>
    <img src="image_path.jpg " target="_blank" >
     </div>
</body>
</html>
```

7. Write a Flask app with a /contact page containing a form (Name, Message) and a /submit route that displays the submitted data using both **POST** and **GET** methods.

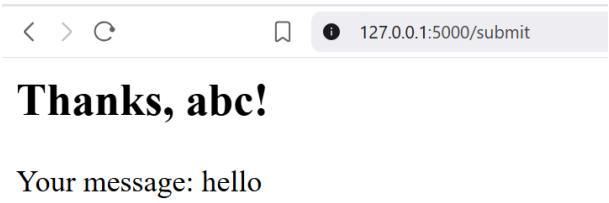
app.py

```
from flask import Flask, render_template,request

app = Flask(__name__)

@app.route('/contact')
def contact():
  return render_template('contacts.html')

# Route to handle form submission
@app.route('/submit', methods=['POST', 'GET'])
def submit():
  if request.method == 'POST':
    name = request.form['username']
    msg = request.form['message']
```

```
    else:
        name = request.args.get('username')
        msg = request.args.get('message')
    return f"<h2>Thanks, {name}!</h2><p>Your message: {msg}</p>"

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

contacts.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Contact Page</title>
</head>
<body>
    <h1>Contact Us</h1>
  <!-- <form action="/submit" method="GET"> -->
    <form action="/submit" method="POST">
        <label>Name:</label>
        <input type="text" name="username"><br><br>

        <label>Message:</label>
        <input type="text" name="message"><br><br>

        <button type="submit">Send</button>
    </form>
</body>
</html>
```



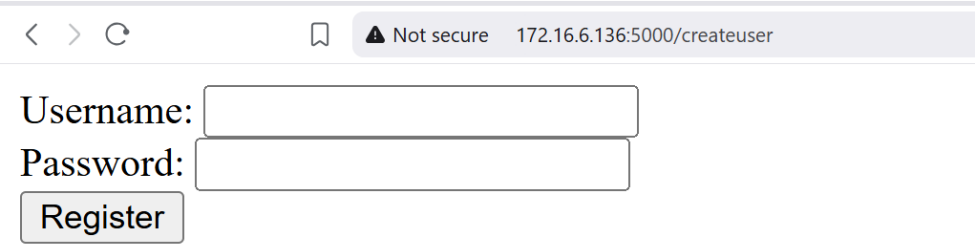
8. Create a Flask application that connects to a MySQL database **flaskdb** containing a table **users(id, name, email)**.
1. Establish a connection to MySQL using **mysql.connector**.
2. Create a route **/**createuser to add a new user.
3. Create a route **/users** to display all users in json format.
4. Return the result of the query in **JSON format** using jsonify.

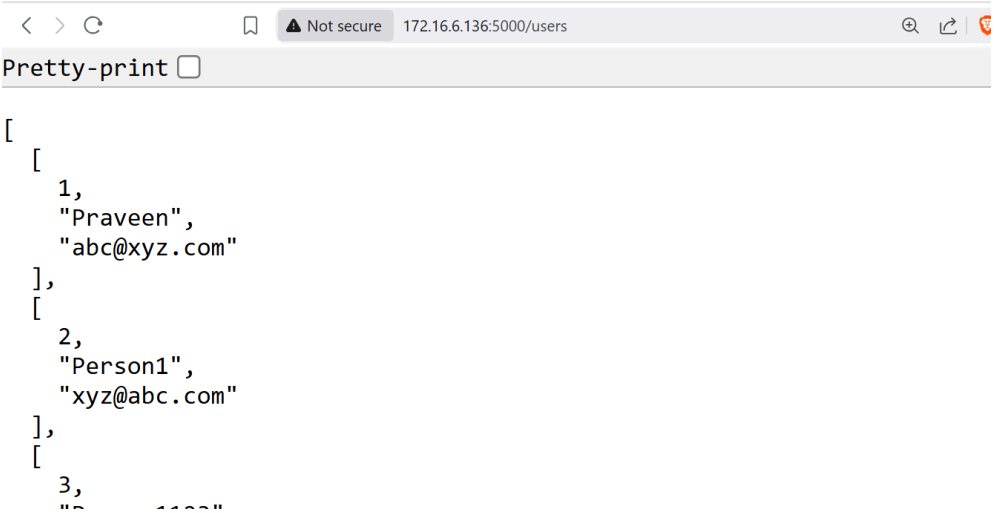
```
CREATE DATABASE flaskdb ;
USE flaskdb ;

CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL
);
```

app.py

```
import mysql.connector
from flask import Flask, request, jsonify

app = Flask(__name__)

db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="flaskdb"
)
cursor = db.cursor()

@app.route('/users', methods=['GET'])
def get_users():
    cursor.execute("SELECT * FROM users")
    return jsonify(cursor.fetchall())

@app.route('/createuser', methods=['POST','GET'])
def add_users():
    if(request.method != 'POST'):
        return '''<form method="post">
        Username: <input type="text" name="name" required><br>
        Password: <input type="email" name="email" required><br>
        <input type="submit" value="Register">
        </form>'''
    else:
        name=request.form["name"]
```

```
    email=request.form["email"]
    cursor.execute("INSERT INTO users (name, email) VALUES (%s, %s)", (name, email))
    db.commit()
    return jsonify({"message": "User added successfully!"})

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=5000, debug=True)
```





9. develop a Flask web application that connects to a MySQL database using the **Flask-MySQLdb** extension.
   i) Create an API endpoint /users that retrieves all rows from the users table.
   ii) Display the query result in a tabular format using a Jinja template (users.html).

pip install flask-mysqldb

```
CREATE DATABASE flaskdb ;
USE flaskdb ;

CREATE TABLE `users` (
 `id` int(11) NOT NULL,
 `name` varchar(100) DEFAULT NULL,
 `email` varchar(100) DEFAULT NULL
);
```

```
INSERT INTO `users` (`id`, `name`, `email`) VALUES
(1, 'Praveen', 'abc@xyz.com'),
(2, 'Person1', 'xyz@abc.com'),
(3, 'Person1123', 'xyz@abc.com'),
(4, 'ssdasd', 'ssasa@123.com');
```

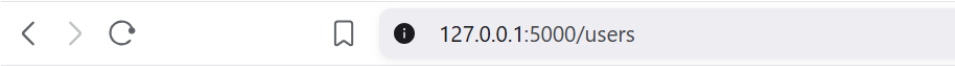
app.py

```
from flask import Flask, render_template
from flask_mysqldb import MySQL

app = Flask(__name__)

# Database configuration
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'flaskdb'
app.config['MYSQL_HOST'] = 'localhost'

mysql = MySQL(app)

@app.route('/users', methods=['GET'])
def get_users():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM users")
    rows = cur.fetchall()
    columns = [desc[0] for desc in cur.description]
    cur.close()

    return render_template("users.html", rows=rows, columns=columns)

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

templates/users.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Users Table</title>
    <style>
        table, th, td {
            border: 1px solid black;
            border-collapse: collapse;
            padding: 8px;
        }
```

```
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <h2>Users Table</h2>
    <table>
        <tr>
            {% for col in columns %}
                <th>{{ col }}</th>
            {% endfor %}
        </tr>
        {% for row in rows %}
            <tr>
                {% for item in row %}
                    <td>{{ item }}</td>
                {% endfor %}
            </tr>
        {% endfor %}
    </table>

</body>
</html>
```



Users Table

id	name	email
1	Praveen	abc@xyz.com
2	Person1	xyz@abc.com
3	Person1123	xyz@abc.com
4	ssdasd	ssasa@123.com

10. Develop a Flask web application that connects to a MySQL database using SQLAlchemy ORM. The application must:

a. Define a model class User representing the users table with fields such as id, name, and email.

b. Configure the database connection using SQLAlchemy in app.py.

c. Create an API endpoint /users that retrieves all user records from the database and renders the results in an HTML table using a Jinja template (users.html).

pip install flask flask_sqlalchemy mysqlclient

pip install pymysql

app.py

```
from flask import Flask, render_template
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

# Configure MySQL database connection
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:@localhost/flaskdb'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# Define ORM model
class User1(db.Model):
    __tablename__ = 'user1'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)

    def __repr__(self):
        return f"<User1 {self.name}>"

# Create tables and insert sample data (runs at startup)
with app.app_context():
    db.create_all()
    if not User1.query.first():  # insert only if empty
        user1 = User1(name="Alice", email="alice@example.com")
        user2 = User1(name="Bob", email="bob@example.com")
        user3 = User1(name="Charlie", email="charlie@example.com")
        db.session.add_all([user1, user2, user3])
        db.session.commit()

# Route to display all users
@app.route("/users")
def show_users():
```

```
    users = User1.query.all()
    return render_template("users.html", users=users)

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

templates/users.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Users Table</title>
    <style>
        table, th, td {
            border: 1px solid black;
            border-collapse: collapse;
            padding: 8px;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <h2>Users Table</h2>
    <table>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Email</th>
        </tr>
        {% for user in users %}
            <tr>
                <td>{{ user.id }}</td>
                <td>{{ user.name }}</td>
                <td>{{ user.email }}</td>
            </tr>
        {% endfor %}
    </table>
</body>
</html>
```

127.0.0.1:5000/users

Users Table

ID	Name	Email
1	Alice	alice@example.com
2	Bob	bob@example.com
3	Charlie	charlie@example.com