# Experiment 3

**Aim: Study and learn basics of TypeScript by writing small code snippets for programs like Hello World, Calculator using TypeScript.**

JavaScript

```
class Calculator {
 private currentResult: number = 0;
  add(num: number): void {
   this.currentResult += num;
 }
  subtract(num: number): void {
   this.currentResult -= num;
 }
  multiply(num: number): void {
   this.currentResult *= num;
 }
  divide(num: number): void {
   if (num === 0) {
     throw new Error("Cannot divide by zero");
   }
   this.currentResult /= num;
 }
  getCurrentResult(): number {
   return this.currentResult;
 }
  clear(): void {
   this.currentResult = 0;
 }
}

const calculator = new Calculator();

// Example usage with user input
const num1 = parseFloat(prompt("Enter first number"));
const num2 = parseFloat(prompt("Enter second number"));
const operation = prompt("Enter operation (+, -, *, /)");

if (operation === "+") {
 calculator.add(num1 + num2);
} else if (operation === "-") {
 calculator.subtract(num1 - num2);
} else if (operation === "*") {
 calculator.multiply(num1 * num2);
} else if (operation === "/") {
 calculator.divide(num1 / num2);
}

console.log(calculator.getCurrentResult()); // prints the result of the calculation
```

1. **Open a terminal and run the following command to install the TypeScript compiler globally:**

```
npm install -g typescript
```

2. **Create a new file with a `.ts` extension and paste the TypeScript code into the file.**
3. **Run the following command to compile the TypeScript code:**

```
tsc your-file-name.ts
```

6. **To run the JavaScript file, use the `node` command followed by the name of the generated file:**

```
node your-file-name.js
```

## Experiment 4

## Aim : study of different types of inheritance in typescript.

JavaScript

```
SINGLE INHERITANCE

class Animal {
  name: string;

  constructor(name: string) {
        this.name = name;
  }

  eat() {
        console.log(`${this.name} is eating.`);
  }
}

class Dog extends Animal {
  bark() {
        console.log(`${this.name} is barking.`);
  }
}

// Create a new instance of Dog
const myDog = new Dog("Buddy");

// Call methods from both classes
myDog.eat(); // Output: Buddy is eating.
myDog.bark(); // Output: Buddy is barking.
```

## MULTILEVEL INHERITANCE:

```
class Animal {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  eat() {
    console.log(`${this.name} is eating.`);
  }
}

class Dog extends Animal {
  bark() {
    console.log(`${this.name} is barking.`);
  }
}

class Bulldog extends Dog {
  growl() {
    console.log(`${this.name} is growling.`);
  }
}

// Create a new instance of Bulldog
const myBulldog = new Bulldog("Spike");

// Call methods from all three classes
myBulldog.eat(); // Output: Spike is eating.
myBulldog.bark(); // Output: Spike is barking.
myBulldog.growl(); // Output: Spike is growling.
```

## HIERARCHIAL INHERITANCE

```
class Animal {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  eat() {
    console.log(`${this.name} is eating.`);
  }
}
```

```
class Dog extends Animal {
  bark() {
    console.log(`${this.name} is barking.`);
  }
}

class Cat extends Animal {
  meow() {
    console.log(`${this.name} is meowing.`);
  }
}

// Create a new instance of Dog and Cat
const myDog = new Dog("Buddy");
const myCat = new Cat("Whiskers");

// Call methods from both classes
myDog.eat(); // Output: Buddy is eating.
myDog.bark(); // Output: Buddy is barking.

myCat.eat(); // Output: Whiskers is eating.
myCat.meow(); // Output: Whiskers is meowing.
```

## MULTIPLE INHERITANCE(INTERFACE)

```
interface Animal {
  name: string;
  eat(): void;
}

interface Mammal {
  run(): void;
}

interface Bird {
  fly(): void;
}

class Bat implements Animal, Mammal, Bird {
  name: string;

  constructor(name: string) {
    this.name = name;
  }
  eat() {
    console.log(`${this.name} is eating.`);
  }

  run() {
    console.log(`${this.name} is running.`);
  }

  fly() {
    console.log(`${this.name} is flying.`);
  }
}

// Create a new instance of Bat
const myBat = new Bat("Batty");

// Call methods from all three interfaces
myBat.eat(); // Output: Batty is eating.
myBat.run(); // Output: Batty is running.
myBat.fly(); // Output: Batty is flying.
```

## To run -

Unset
```
npm install -g typescript
```

Unset
```
tsc filename.ts
```

Unset
```
node filename.js
```

## Experiment 5

**Aim : Study of Access Modifiers in typeScript with example.**

JavaScript

```
class Car {
  public make: string; // Public property
  private model: string; // Private property
  protected year: number; // Protected property

  constructor(make: string, model: string, year: number) {
        this.make = make;
        this.model = model;
        this.year = year;
  }

  public startEngine() {
        console.log(`Starting the engine of a ${this.year} ${this.make} ${this.model}.`);
  }

  private stopEngine() {
        console.log(`Stopping the engine of a ${this.year} ${this.make} ${this.model}.`);
  }

  protected honk() {
        console.log(`Honking the horn of a ${this.year} ${this.make} ${this.model}.`);
  }
}

class SportsCar extends Car {
  constructor(make: string, model: string, year: number) {
        super(make, model, year);
  }

  public race() {
        console.log(`Racing in a ${this.year} ${this.make} ${this.model}.`);
  }

  // Uncommenting this line will result in a compile-time error, as the "model" property
is private to the "Car" class.
  //public getModel() {
  //  return this.model;
  //}

  public honk() {
        super.honk();
  }
}

// Create a new instance of Car
const myCar = new Car("Toyota", "Corolla", 2022);

// Access the public property
```

```
console.log(`My car is a ${myCar.make} ${myCar.model} from ${myCar.year}.`);

// Call the public method
myCar.startEngine();

// Uncommenting this line will result in a compile-time error, as the "model" property is
private to the "Car" class.
//console.log(myCar.model);

// Uncommenting this line will result in a compile-time error, as the "stopEngine" method
is private to the "Car" class.
//myCar.stopEngine();

// Uncommenting this line will result in a compile-time error, as the "honk" method is
protected to the "Car" class.
//myCar.honk();

// Create a new instance of SportsCar
const mySportsCar = new SportsCar("Ferrari", "F430", 2023);

// Call the public method from the base class
mySportsCar.startEngine();

// Call the public method from the derived class
mySportsCar.race();

// Uncommenting this line will result in a compile-time error, as the "model" property is
private to the "Car" class.
//console.log(mySportsCar.model);

// Uncommenting this line will result in a compile-time error, as the "stopEngine" method
is private to the "Car" class.
//mySportsCar.stopEngine();

// Call the protected method from the derived class
mySportsCar.honk();
```

## To run -

Unset
```
npm install -g typescript
```

```
Unset
tsc filename.ts
```

```
Unset
node filename.js
```

## Experiment 6

Aim: Create a simple HTML page project using Angular framework and apply ng-controller, ng-model and expressions.

ng new project_name
cd  project_name
ng serve --open

```
src/
  app/
    app.component.ts
    app.component.html
    app.component.css
    app.module.ts
  assets/
    ...
  environments/
    environment.ts
    environment.prod.ts
  index.html
  main.ts
  styles.css
angular.json
package.json
tsconfig.json
```

```
JavaScript
app.component.ts
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
template: `
<div>
<h3>{{title}}</h3>
<input [(ngModel)]="name" placeholder="Enter your name">
<p>Hello {{name}}!</p>
<input [(ngModel)]="exp" placeholder="Experiment number">
<p>This is experiment number {{exp}}.</p>
</div>
`,
styles: [`
div {
padding: 30px;
background-color: #e9e2b6;
width: 200px;
margin-left:30%
}
`]
})
export class AppComponent {

title = 'Experiment 6-Angular';
name = '';
exp= '';
}
```

JavaScript

**app.module.ts**

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
imports: [BrowserModule, FormsModule],
declarations: [AppComponent],
bootstrap: [AppComponent]
})
export class AppModule { }
```

To Run

Unset

```
npm install
```

Unset

```
ng serve
```

**Experiment 7**

Aim: Events and Validations in Angular. (Create functions and add events, adding HTML validators, using the $valid property of Angular, etc.)

Python

**index.html**

```
<!DOCTYPE html>
<html>
<head>
<title>Form Validation Example</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<form id="my-form">
<label for="name">Name:</label>
<input type="text" id="name" name="name">
<label for="email">Email:</label>
<input type="email" id="email" name="email">
<label for="password">Password:</label>
<input type="password" id="password" name="password">
<button type="submit">Submit</button>
</form>
<script src="script.js"></script> </body>
</html>
```

JavaScript

**main.ts**

```
interface FormValues {
 name: string;
 email: string;
 password: string;
}
const form = document.querySelector("#my-form") as HTMLFormElement;
const nameInput = document.querySelector("#name") as HTMLInputElement;
const emailInput = document.querySelector("#email") as HTMLInputElement;
const passwordInput = document.querySelector("#password") as HTMLInputElement;
form.addEventListener("submit", (e) => {
 e.preventDefault();
 const values: FormValues = {
 name: nameInput.value,
 email: emailInput.value,
 password: passwordInput.value
 };
 const errorMessage = validateForm(values);
 if (errorMessage) {
 displayError(errorMessage);
 } else {
 alert("Form submitted successfully!");
 }
});
function validateForm(values: FormValues): string | null {
 if (!values.name) {
 return "Name is required";
 }
 if (!values.email) {
 return "Email is required";
 }
 if (!isValidEmail(values.email)) {
 return "Invalid email address";
 }
 if (!values.password) {
 return "Password is required";
 }
 return null;
}
function isValidEmail(email: string): boolean {
 const emailRegex = /^\S+@\S+\.\S+$/;
 return emailRegex.test(email);
}
function displayError(errorMessage: string) {
 const errorElement = document.createElement("p");
 errorElement.classList.add("error");
 errorElement.innerText = errorMessage;
 const form = document.querySelector("#my-form") as HTMLFormElement;
 form.insertBefore(errorElement, form.firstChild);
}
```

JavaScript

**styles.css**

```
form {
 display: flex;
 flex-direction: column;
 max-width: 400px;
 margin: 0 auto;
 }
 label {
 margin-bottom: 0.5rem;
 }
 input[type="text"],
 input[type="email"],
 input[type="password"] {
```

```css
padding: 0.5rem;
margin-bottom: 1rem;
border: 1px solid #ccc;
border-radius: 3px;
font-size: 1rem;
}
input[type="submit"] {
padding: 0.5rem;
border-radius: 3px;
background-color: #007bff; color: #fff;
font-size: 1rem;
border: none;
cursor: pointer;
}
input[type="submit"]:hover { background-color: #0069d9; }
.error {
color: red;
margin-bottom: 1rem;
}
```

To run

```
npm install -g @angular/cli
ng serve
```

## Experiment 8(AJAX)

Aim : Write a program to use AJAX for user validation using and to show the result on the same page below the submit button.

JavaScript

**form.js**

```javascript
$(document).ready(function () {
 $("form").submit(function (event) {
 var formData = {
 name: $("#name").val(),
 email: $("#email").val(),
 superheroAlias: $("#superheroAlias").val(),
 };

 $.ajax({
 type: "POST",
 url: "process.php",
 data: formData,
 dataType: "json",
 encode: true,
 }).done(function (data) {
 console.log(data);
 if (!data.success) {
 if (data.errors.name) {
 $("#name-group").addClass("has-error");
 $("#name-group").append(
 '<div class="help-block">' + data.errors.name + "</div>"
 );
 }
 if (data.errors.email) {
 $("#email-group").addClass("has-error");
 $("#email-group").append(
 '<div class="help-block">' + data.errors.email + "</div>"
 );
 }
 if (data.errors.superheroAlias) {
 $("#superhero-group").addClass("has-error");
 $("#superhero-group").append(
 '<div class="help-block">' + data.errors.superheroAlias + "</div>"
 );
 }
 } else {
 $("#message").html('<div class="alert alert-success">' + data.message + "</div>");
 }
 });

 event.preventDefault();
 });
 });
```

JavaScript

**index.html**

```
<!DOCTYPE html>
<html>
 <head>
 <title>Ajax Form </title>
 <link
 rel="stylesheet"
 href="//netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css"
 />
 <script src="//ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>
</head>
 <body>
 <script src="form.js"></script>
 <div class="col-sm-6 col-sm-offset-3">
 <h1>AJAX Form</h1>
 <form action="process.php" method="POST">
 <div id="name-group" class="form-group">
 <label for="name">Name</label>
 <input
 type="text"
 class="form-control"
 id="name"
 name="name"
 />
 </div>
 <div id="email-group" class="form-group">
 <label for="email">Email</label>
 <input
 type="text"
 class="form-control"
 id="email"
 name="email"
 />
 </div>
 <div id="superhero-group" class="form-group">
 <label for="superheroAlias">Superhero Alias</label>
 <input
 type="text"
 class="form-control"
 id="superheroAlias"
 name="superheroAlias"
 />
 </div>
 <button type="submit" class="btn btn-success">
 Submit
 </button>
 </form>
 <div id="message"></div>
 </div>
 </body>
</html>
```

JavaScript

**process.php**

```
<?php
$errors = [];
$data = [];
if (empty($_POST['name'])) {
 $errors['name'] = 'Name is required.';
}
```

```
if (empty($_POST['email'])) {
 $errors['email'] = 'Email is required.';
}
if (empty($_POST['superheroAlias'])) {
 $errors['superheroAlias'] = 'Superhero alias is required.';
}
if (!empty($errors)) {
 $data['success'] = false;
 $data['errors'] = $errors;
} else {
 $data['success'] = true;
 $data['message'] = 'Success!';
}
echo json_encode($data);
```

To Run

```javascript
php -S localhost:8000
```

## Experiment-9 (Sign In Flaak)

Aim: To develop a Flask Application

```
my_flask_app/
|
css/
|└──main.css
|
templates/
|   └──index.html
main.py
```

```javascript
index.html
```

```
<html lang="en">
    <head>
         <meta charset="UTF-8">
         <meta http-equiv="X-UA-Compatible" content="IE=edge">
         <meta name="viewport" content="width=device-width, initial-scale=1.0">
         <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
         <title>Document</title>
    </head>
    <body>
         <div class="conter">
                 <h1>Login</h1>
                 <form action = "http://localhost:5000/login" method = "post">
                         <div class="txt_field">
                                 <input type="text" name="name" required>
                                 <span></span>
                                 <label >UserName</label>
                         </div>
                         <div class="txt_field">
                                 <input type="password" name = "password" required>
                                 <span></span>
                                 <label >Password</label>
                         </div>
                         <div class="pass">Forget Password?</div>
                         <input type="submit" value="Login">
                         <div class="signup_link">
                                 No a member?
                                 <a href="#">signup</a>
                         </div>
                 </form>
         </div>
    </body>
    </html>
Footer
```

```
JavaScript
```

## main.css

```
@import
url('https://fonts.googleapis.com/css2?family=Montserrat&family=Poppins:wght@500&display=s
wap');
body{
        margin: 0;
        padding: 0;
        font-family: montserrat ;
        background: linear-gradient(120deg,#2980b9, #8e44ad);
        height: 100vh;
        overflow: hidden;
}
.conter{
        position: absolute;
        top:50%;
        left: 50%;
        transform: translate(-50%, -50%);
        width: 400px;
        background: white;
        border-radius: 10px;
}
.conter h1{
        text-align: center;
        padding: 0 0 20px 0;
        border-bottom: 1px solid silver;
}
.conter form{
        padding: 0 40px;
        box-sizing: border-box;
}
form .txt_field{
        position: relative;
        border-bottom: 2px solid #adadad;
        margin: 30px 0;
}
.txt_field input{
        width: 100%;
        padding:  0 5px;
        height: 40px;
        font-size: 16px;
        border: none;
        background: none;
        outline: none;
}

.txt_field label{
        position: absolute;
        top: 50%;
        left: 5px;
        color: #adadad;
        transform: translateY(-50%);
        font-size: 16px;
        pointer-events: none;
        transition: .5s;
}

.txt_field span::before{
        content: '';
        position: absolute;
        top: 40px;
        left: 0;
        width: 0%;
        height: 2px;
        background: #2691d9;
        transition: .5s;

}
.txt_field input:focus ~ label,
.txt_field input:valid ~ label{
        top: -5px;
        color: #2691d9;
}
.txt_field input:focus ~ span::before,
.txt_field input:valid ~ span::before{

        width: 100%;
}
.pass{
        margin: -5px 0 20px 5px;
        color: #a6a6a6;
        cursor: pointer;
}
.pass:hover{
        text-decoration: underline;
}
input[type="submit"]{
        width: 100%;
        height: 50px;
        border: 1px solid;
        background: #2691d9;
        border-radius: 25px;
        font-size: 18px;
        color: #e9f4fb;
        font-weight: 700;
        cursor: pointer;
        outline: none;
}
input[type="submit"]:hover{
        border-color: #2691d9
        transparent 0.5s;
}
.signup_link{
        margin: 30px;
        text-align: center;
        font-size: 16px;
        color: #666666;
}
.signup_link a{
        color: #2691d9;
        text-decoration: none;
}
.signup_link a:hover{
        text-decoration: underline;
}
```

Python

### main.py

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

app = Flask(__name__)
def checkAuth(name,password):
        if(name == 'Elon' and password == '123'):
        return True
        else:
        return False

@app.route('/login', methods=['POST', 'GET'])
def login():
        if request.method == "POST":
        # getting input with name = fname in HTML form
        name = request.form.get("name")
        # getting input with name = lname in HTML form
        password = request.form.get("password")
        valid = checkAuth(name,password)
        if(valid):
                return 'Welcome ' + name
        else:
        return 'Incorrect Username or Password'
        return render_template("index.html")



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

**To run your Flask app,**
open a terminal or command prompt, navigate to your project directory (my_flask_app), and run the following command:

```
python main.py
```

## Experiment 10

https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-20-04

```python
mongo
show dbs
use booksdb
db.createCollection("books")
db.books.insert({
  title: "The Catcher in the Rye",
  author: "J.D. Salinger",
  year: 1951
})
db.books.insertMany([
  {
    title: "To Kill a Mockingbird",
    author: "Harper Lee",
    year: 1960
  },
  {
    title: "Pride and Prejudice",
    author: "Jane Austen",
    year: 1813
  }
])
db.books.find()
db.books.findOne({ title: "The Catcher in the Rye" })
db.books.updateOne(
  { title: "The Catcher in the Rye" },
  { $set: { year: 1952 } }
)
db.books.deleteOne({ title: "The Catcher in the Rye" })
db.books.drop()
db.dropDatabase()
```

## BMI CALCULATOR

```javascript
FLASK- BMI CALCULATOR

.html
<!doctype html>

<head><meta charset="utf-8">
```

```
   <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>BMI Calculator</title>

    <link rel="stylesheet" href="https://unpkg.com/purecss@0.6.2/build/pure-min.css"
integrity="sha384-UQiGfs9ICog+LwheBSRCt1o5cbyKIHbwjWscjemyBMT9YCUMZffs6UqUTd0hObXD"
crossorigin="anonymous">

    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">

</head>

<h1>BMI Calculator</h1>

<body>

<div class="main">
    <form class="pure-form" method="POST" action="/">
    Weight in kgs:<br>
    <input type="text" name="weight"><br>
    Height in cms:<br>
    <input type="text" name="height"><br>
    <button type="submit" class="pure-button pure-button-primary"
value="Submit">Submit</button>
    </form>
</div>

<br>

<div class="main">
    {% if bmi %}
    <p>
        {% print("Your BMI is {}.".format(bmi)) %}
    </p>
    {% endif %}
</div>


</body>

.css

.main {
    padding-top: 50px;
    padding-bottom: 50px;
    /* width: 200px;
    height: 140px; */
    background-color: cadetblue;
    /* background-image: url("image.jpg"); */
    color: black;
    color-adjust: inherit;
    overflow: hidden;
    text-align: center;
}

h1 {
    text-align: center;
    /* padding-left: 0px; */
}

.centered-text {
    text-align: center;
}
th, td , table {
    width: 20%;
    border: 1px solid black;
    border-collapse: collapse;
}


tr:nth-child(even) {
    background-color: #e1e2f7;
}
.border {
    border: 1px solid black;
    border-collapse: collapse;
}
```

Python

```
App.py
#!python3


from flask import Flask, render_template, request
app = Flask(_name_)
@app.route('/', methods=['GET', 'POST'])
def index():
    bmi = ''
    if request.method == 'POST' and 'weight' in request.form:
        weight = float(request.form.get('weight'))
        height = float(request.form.get('height'))
        bmi = calc_bmi(weight, height)
    return render_template("bmi_calc.html",
                                bmi=bmi)

def calc_bmi(weight, height):
    return round((weight / ((height / 100) ** 2)), 2)
if _name_ == '_main_':
    app.run()
```

pip install python
to run: python -m flask run

## WEATHER-APP FLASK

```
pip install Flask
pip install requests
```

```python
app.py
from flask import Flask, render_template, request
import requests
app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    weather_data = {}
    if request.method == 'POST':
        city = request.form['city']
        api_key = 'your_openweathermap_api_key'
        url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric'
        response = requests.get(url)
        data = response.json()

        if data.get('cod') != '404':
            weather_data = {
                'city': data['name'],
                'temperature': data['main']['temp'],
                'description': data['weather'][0]['description'],
                'icon': data['weather'][0]['icon']
            }
        else:
            weather_data = {'error': 'City not found'}

    return render_template('index.html', weather_data=weather_data)

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

Create a templates directory and an index.html file inside it:

```
JavaScript
index.html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Basic Weather App</title>
  </head>
  <body>
    <h1>Basic Weather App</h1>
    <form method="post" action="/">
      <input type="text" name="city" placeholder="Enter city name" required>
      <button type="submit">Get Weather</button>
    </form>
    {% if weather_data %}
      {% if weather_data.error %}
        <p>{{ weather_data.error }}</p>
      {% else %}
        <h2>{{ weather_data.city }}</h2>
        <img src="http://openweathermap.org/img/w/{{ weather_data.icon }}.png" alt="{{
weather_data.description }}">
        <p>{{ weather_data.temperature }}°C</p>
        <p>{{ weather_data.description }}</p>
      {% endif %}
    {% endif %}
  </body>
</html>
```

To run

```
export FLASK_APP=app.py
export FLASK_ENV=development
flask run
```

## TYPESCRIPT WEBSITE

```
JavaScript
index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple TypeScript Website</title>
</head>
<body>
    <h1>Simple TypeScript Website</h1>
    <button id="clickButton">Click me!</button>

    <script src="app.js"></script>
</body>
```

```
</html>
```

JavaScript

```
app.ts
document.addEventListener('DOMContentLoaded', () => {
    const button = document.getElementById('clickButton') as HTMLButtonElement;
    let clickCount = 0;

    button.addEventListener('click', () => {
        clickCount++;
        button.textContent = `Clicked ${clickCount} times`;
    });
});
```

To run - tsc app.ts

## BLOG APP/ PORTFOLIO WEBSITE  FLASK

Python

```
app.py
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def index():
    blog_posts = [
        {
            'title': 'My First Blog Post',
            'content': 'This is the content of my first blog post.'
        },
        {
            'title': 'My Second Blog Post',
            'content': 'This is the content of my second blog post.'
        }
    ]
    return render_template('index.html', blog_posts=blog_posts)
if __name__ == '__main__':
    app.run(debug=True)
```

Create a templates directory and an index.html file inside it:

```
Python
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Simple Blog App</title>
  </head>
  <body>
    <h1>Simple Blog App</h1>
    <div>
      {% for post in blog_posts %}
        <h2>{{ post.title }}</h2>
        <p>{{ post.content }}</p>
      {% endfor %}
    </div>
  </body>
</html>
```

To run -

export FLASK_APP=app.py
export FLASK_ENV=development
flask run

## FEEDBACK FORM FLASK

```
Python
app.py
from flask import Flask, render_template, request, redirect, url_for, flash

app = Flask(__name__)
app.secret_key = 'your_secret_key'

@app.route('/', methods=['GET', 'POST'])
def feedback():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        feedback = request.form['feedback']

        flash(f'Thank you {name}, your feedback has been submitted.', 'success')
        return redirect(url_for('feedback'))

    return render_template('feedback.html')

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

Replace your_secret_key with a secret key for your app, which is used for session handling.

Python

```
templates/feedback.html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Feedback Form</title>
  </head>
  <body>
    <h1>Feedback Form</h1>
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        {% for category, message in messages %}
          <div>{{ message }}</div>
        {% endfor %}
      {% endif %}
    {% endwith %}
    <form method="post" action="/">
      <label for="name">Name:</label>
      <input type="text" name="name" required>
      <br>
      <label for="email">Email:</label>
      <input type="email" name="email" required>
      <br>
      <label for="feedback">Feedback:</label>
      <textarea name="feedback" required></textarea>
      <br>
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

To run
export FLASK_APP=app.py
export FLASK_ENV=development
flask run

## STUDENT RECORD ANGULAR

npm install -g @angular/cli

ng new simple-student-record --minimal --skip-tests --inline-style --inline-template
cd simple-student-record

Replace the content of src/app/app.component.ts with the following code:

JavaScript

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1>Simple Student Record</h1>
    <table>
      <thead>
        <tr>
          <th>Name</th>
          <th>Age</th>
          <th>Grade</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let student of students">
          <td>{{ student.name }}</td>
          <td>{{ student.age }}</td>
          <td>{{ student.grade }}</td>
        </tr>
      </tbody>
    </table>
  `,
  styles: [`
    table {
      width: 100%;
      border-collapse: collapse;
    }
    th, td {
      border: 1px solid black;
      padding: 8px;
      text-align: left;
    }
    th {
      background-color: #f2f2f2;
    }
  `]
})
export class AppComponent {
  students = [
    { name: 'John Doe', age: 18, grade: 'A' },
    { name: 'Jane Smith', age: 17, grade: 'B' },
    { name: 'Alice Brown', age: 19, grade: 'C' },
  ];
}
```

Replace the content of src/index.html with the following code:

JavaScript
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Simple Student Record</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
```

```
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

To run - ng serve