Experiment No. 10

Title: Develop a Guestbook Application using Google App Engine

Course / Lab: Web Applications / Cloud Lab

Author: (Your Name)

Date: (DD-MM-YYYY)

Aim

10. Develop a Guestbook Application using Google App Engine.

Description

In this experiment you will build a simple Guestbook web application where visitors can sign (name + message). The app is implemented using Flask (Python 3) and can be run locally or deployed to Google App Engine (Standard Environment). The Guestbook demonstrates routing, templates, form handling, and basic deployment to Google Cloud. For persistence you may use in-memory storage for local testing or configure Google Datastore / Firestore for production.

Prerequisites

1. A computer with internet connection.\n
2. Python 3.7+ installed (you have Python 3.13). Ensure `python --version` works.\n
3. pip (Python package manager).\n
4. (Optional but recommended) a virtual environment tool: `venv`.\n
5. Google Cloud SDK (gcloud) installed and authenticated for deployment.\n
6. A Google Cloud project (create one via Console or `gcloud`).\n
7. Basic familiarity with terminal/command prompt and text editor.\n

Procedure (Step-by-step)

Step 1: Create project folder

```
Create a project folder and change into it. Example commands:

Windows / PowerShell:
    mkdir guestbook_flask_app
    cd guestbook_flask_app

Linux / macOS:
    mkdir -p guestbook_flask_app
    cd guestbook_flask_app
```

Step 2: Create a virtual environment (recommended)

```
Create and activate a venv to isolate dependencies:

Windows:
    python -m venv venv
    venv\Scripts\activate

macOS / Linux:
    python3 -m venv venv
    source venv/bin/activate
```

Step 3: Install Flask

Install Flask (and gunicorn for App Engine deployment):

```
    pip install Flask gunicorn
```

Alternatively, create a requirements.txt file with:

```
    Flask==2.3.3
    gunicorn==21.2.0
```

and run:

```
    pip install -r requirements.txt
```

Step 4: Create application files

Create `main.py`, templates and static folders. Example structure:

```
guestbook_flask_app/
■■ app.yaml
■■ main.py
■■ requirements.txt
■■ templates/
■   ■■ index.html
■   ■■ guestbook.html
■■ static/
   ■■ css/style.css
   ■■ js/bubbles.js
```

Step 5: main.py (Flask application)

Create `main.py` with the Flask routes for '/', '/sign' and '/guestbook'.

Example (minimal):

Step 6: Templates and design

Create `templates/index.html` (sign-in page) and `templates/guestbook.html` (entries page). Include shared CSS/J

Step 7: Run locally and test

Run the application locally for testing:

```
    python main.py
```

Open browser at http://127.0.0.1:8080 (or http://127.0.0.1:5000 depending on your code) and test signing the gue

Step 8: Prepare for App Engine deployment

Create `app.yaml` and `requirements.txt`. Example `app.yaml` for App Engine Standard (Python 3):

```
runtime: python310
entrypoint: gunicorn -b :$PORT main:app

handlers:
- url: /static
  static_dir: static

- url: /.*
  script: auto
```

Step 9: Create Google Cloud project and enable APIs

Login with gcloud and set the project. Example commands:

```
    gcloud auth login
    gcloud projects create <PROJECT_ID> --name="Guestbook Project"  # if needed
    gcloud config set project <PROJECT_ID>
```

Enable required services:

```
    gcloud services enable appengine.googleapis.com datastore.googleapis.com cloudbuild.googleapis.com
```

Create App Engine app (choose region):

```
    gcloud app create --project=<PROJECT_ID> --region=us-central
```

Step 10: Deploy to App Engine

Deploy the app using gcloud:

```
    gcloud app deploy app.yaml --project=<PROJECT_ID>
```

Open the app in browser:

```
    gcloud app browse --project=<PROJECT_ID>
```

Step 11: Verification and logs

Check that the app loads at https://<PROJECT_ID>.appspot.com and that newly signed entries appear on /guestbook.

```
    gcloud app logs tail -s default
```

or view logs in the Google Cloud Console.

Results

After performing the steps above you should observe: 1. The app runs locally and the sign form accepts name + message. After submitting, the app redirects to /guestbook where all entries are listed. 2. When deployed to App Engine, the app is available at your project's URL (https://.appspot.com). Users can sign the guestbook using the live URL. 3. Note: If you use in-memory storage (a Python list), entries are lost after the app restarts or scales. For persistent storage in production, configure Google Cloud Datastore / Firestore (Datastore mode) and use google-cloud-ndb library. 4. Logs show request/response info and can help debug issues such as missing dependencies or runtime errors.

Sample output (entries):

```
John Doe - 2025-09-17 10:45:31
Hello! Great workshop. - Entry displayed under Guestbook.

Jane Smith - 2025-09-17 11:02:05
Nice app — congrats! - Entry displayed under Guestbook.
```

Troubleshooting Tips

```
- Error: ModuleNotFoundError / ImportError: Ensure dependencies are installed and your virtualenv is active. Rur

- Error: App Engine runtime issues: Check `app.yaml` runtime and entrypoint. Use `gcloud app deploy` logs.

- If the app opens the wrong page: ensure Flask route `@app.route('/')` returns index.html, and `/guestbook` ret

- Local dev server for App Engine: use `dev_appserver.py app.yaml` if using Google Cloud SDK's local emulator.

- Persistent storage: create Firestore in Datastore mode in the Cloud Console and update code to use google-clou
```

Appendix: Code Listings

main.py (Flask app)

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

app = Flask(__name__)

# In-memory storage for guestbook entries
entries = []

@app.route("/")
def index():
    return render_template("index.html")  # Homepage with Sign-in form

@app.route("/sign", methods=["POST"])
def sign():
    name = request.form.get("name", "Anonymous")
    message = request.form.get("message", "")
    entries.append({
        "name": name,
        "message": message,
        "date": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    })
    return redirect(url_for("guestbook"))  # Redirect to Guestbook page

@app.route("/guestbook")
def guestbook():
    return render_template("guestbook.html", entries=entries)  # Show all entries

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=8080, debug=True)
```

app.yaml

```
runtime: python310
entrypoint: gunicorn -b :$PORT main:app

handlers:
- url: /static
  static_dir: static

- url: /.*
  script: auto
```

requirements.txt

```
Flask==2.3.3
gunicorn==21.2.0
```

templates/index.html (simplified)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Guestbook - Sign</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
  <canvas id="bubblesCanvas"></canvas>
  <div class="container">
    <h1>Sign the Guestbook</h1>
    <form action="{{ url_for('sign') }}" method="post">
      <input type="text" name="name" placeholder="Your Name" required>
      <textarea name="message" placeholder="Your Message" required></textarea>
      <button type="submit">Sign Guestbook</button>
    </form>
  </div>
  <script src="{{ url_for('static', filename='js/bubbles.js') }}"></script>
</body>
</html>
```

templates/guestbook.html (simplified)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
  <title>Guestbook - Entries</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
  <canvas id="bubblesCanvas"></canvas>
  <div class="container">
    <h1>Guestbook Entries</h1>
    <a href="{{ url_for('index') }}">Sign Another Entry</a>
    {% for entry in entries %}
      <div class="entry">
        <h3>{{ entry.name }}</h3>
        <p>{{ entry.message }}</p>
        <small>{{ entry.date }}</small>
      </div>
    {% endfor %}
  </div>
  <script src="{{ url_for('static', filename='js/bubbles.js') }}"></script>
</body>
</html>
```

Conclusion

You have built a simple Guestbook application, tested it locally, and prepared it for deployment on Google App Engine. For production use, replace in-memory storage with a persistent datastore (google-cloud-ndb + Firestore in Datastore mode), add authentication to prevent spam, and improve scalability via proper App Engine settings.