Personal Blog on IBM Cloud Static Web App

Arssam Basha M

au110121104014

arssambasha82@gmail.com

Project Title: Personal Travel blog

Phase 4: Development Part 2

- Building the travel blog by setting up the IBM Cloud Static Web App and deploying the website.
- Create a new Static Web App and follow the prompts to set up the repository, build pipeline, and deployment options.
- Converting the HTML content into template files that can be easily updated.

Objective:

The primary objective of the personal blog, hosted on the IBM Cloud as a static web app, is to share the passion for the travel and adventure while demonstrating the capabilities of IBM Cloud for hosting static websites.

Description:

I have created the Personal Travel Blog Website by integrating Flask to specifically handle the Routes, Templates, Static files and to connect with the DB2 Database from the IBM Cloud then used Docker to containerize the application then Kubernetes was used to manage the container.

Overview:

Content Planning and Creation:

Before diving into technical aspects, it is essential to plan the content for the travel blog, then need to create engaging travel stories, share useful travel tips, and curate captivating photos from the journey. These are the heart of your blog and what will inspire others to explore the world.

Website Design:

For the blog's layout, need to design an aesthetically pleasing and user-friendly website. This involves creating web pages using HTML for content structure, CSS for styling, and possibly JavaScript for interactive elements like image galleries or maps showing the places they have visited. This design thinking aspect ensures that your blog is visually appealing and provides a great user experience.

IBM Cloud Setup:

To host the travel blog, need to set up an account on IBM Cloud. IBM Cloud offers a variety of services, including IBM Db2 Cloud. This service allows to store the data for the websites, which is suitable for a blog. Using that can deploy the HTML, CSS, and JavaScript files to the IBM Cloud's Static Web Apps service.

Content Management:

For efficient content management, I have chosen Flask and IBM Db2 Cloud.

Flask: Flask is a micro web framework for Python. It can use it to create dynamic and interactive web pages. In this case, Flask can be used to manage the blog content. It can create templates for the blog posts and use Flask to render them with dynamic content. Flask can also handle routing, making it easy to organize the content and create a structured blog.

IBM Db2 Cloud: IBM Db2 is a database management system. By integrating it into the Flask application, it can store and manage data related to the travel blog. This can include information about the visited places, tips, user comments, and more. With Db2 Cloud, the data is stored securely and can be easily retrieved and updated through your Flask application.

The content management aspect involves using Flask to create and serve your blog content dynamically. You can use templates to ensure a consistent design, and Flask can communicate with the IBM Db2 Cloud to fetch and update data as needed. This setup enables you to easily add, edit, or delete blog posts and manage your blog's data

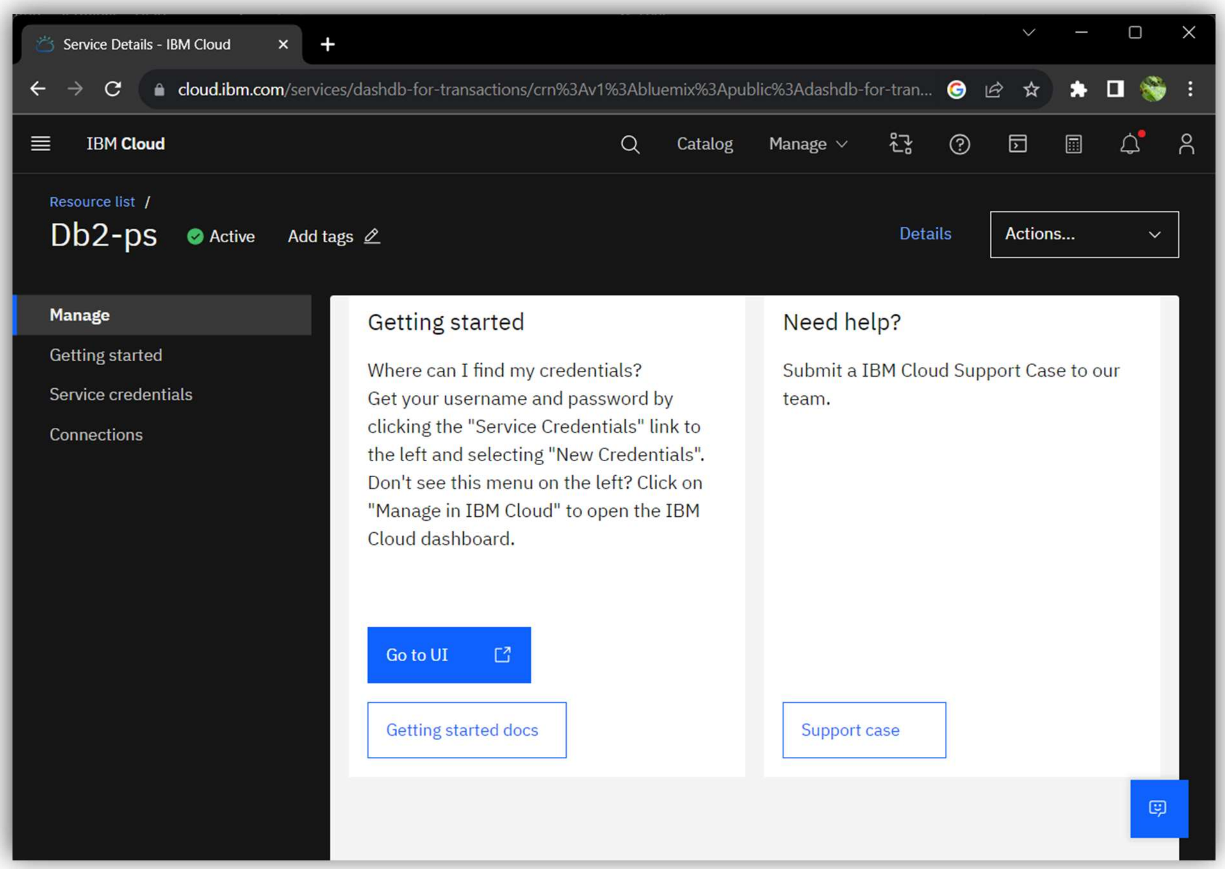
IBM Cloud:

IBM Cloud is a cloud computing platform offered by IBM. It provides a range of cloud services, including infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) offerings.

After signing up and logging in, create a Db2 service instance. Db2 is IBM's database management system that offers relational database capabilities.

In the IBM Cloud dashboard, go to the “Catalog” and search for “Db2.” Choose the plan that suits your needs.

Follow the on-screen instructions to create the Db2 service.



Getting started

Where can I find my credentials? Get your username and password by clicking the "Service Credentials" link to the left and selecting "New Credentials". Don't see this menu on the left? Click on "Manage in IBM Cloud" to open the IBM Cloud dashboard.

Need help?

Submit a IBM Cloud Support Case to our team.

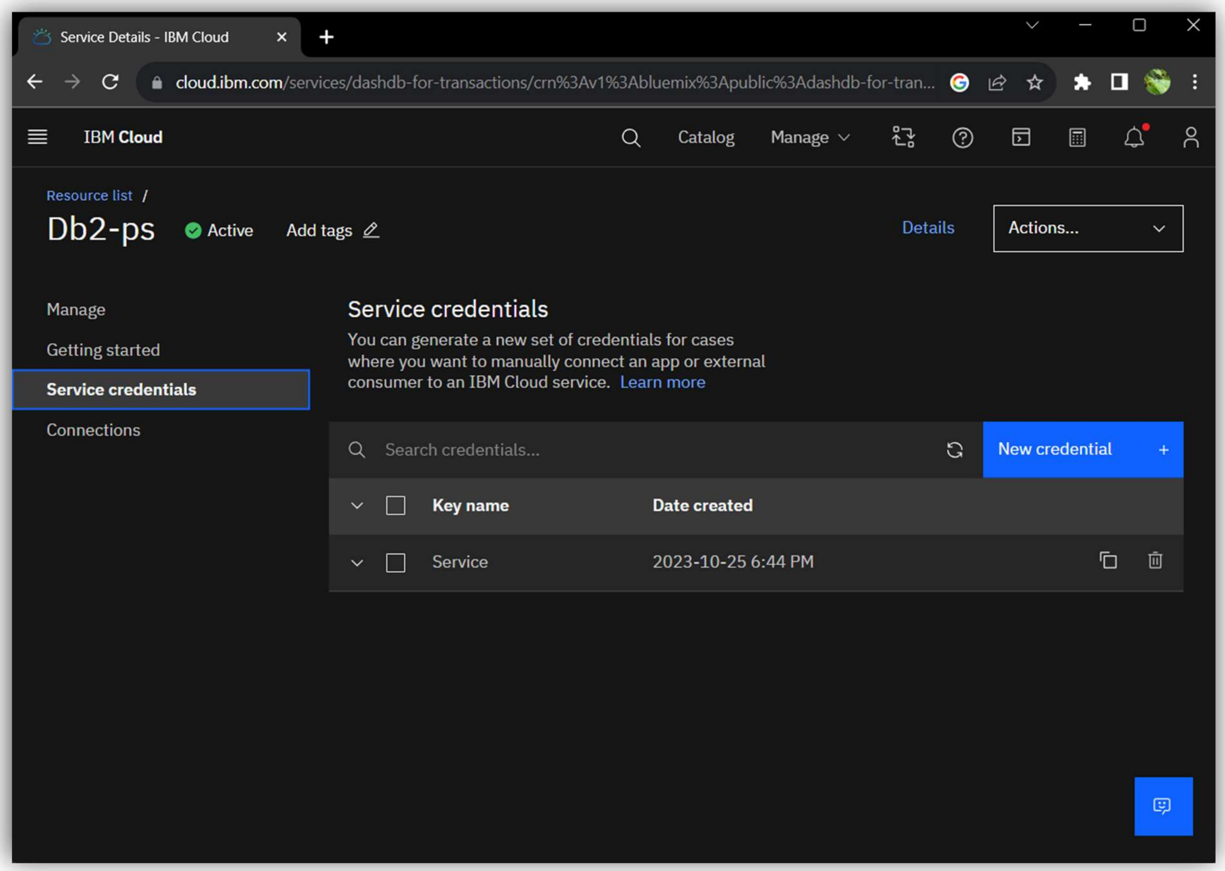
Fig-1: DB2 in IBM Cloud

Connecting to Db2:

Once your Db2 service is created, you can connect to it using various methods. IBM provides drivers and libraries for various programming languages to interact with Db2. You can use these drivers in your Flask application to connect to your Db2 service hosted in IBM Cloud.

Install Required Libraries:

Use the ibm_db Python library to connect to IBM Db2 from your Flask application. You can install it with pip as mentioned in a previous response.



Service credentials

You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud service. Learn more

Key name	Date created
Service	2023-10-25 6:44 PM

Fig-2: Service Credentials in IBM Cloud

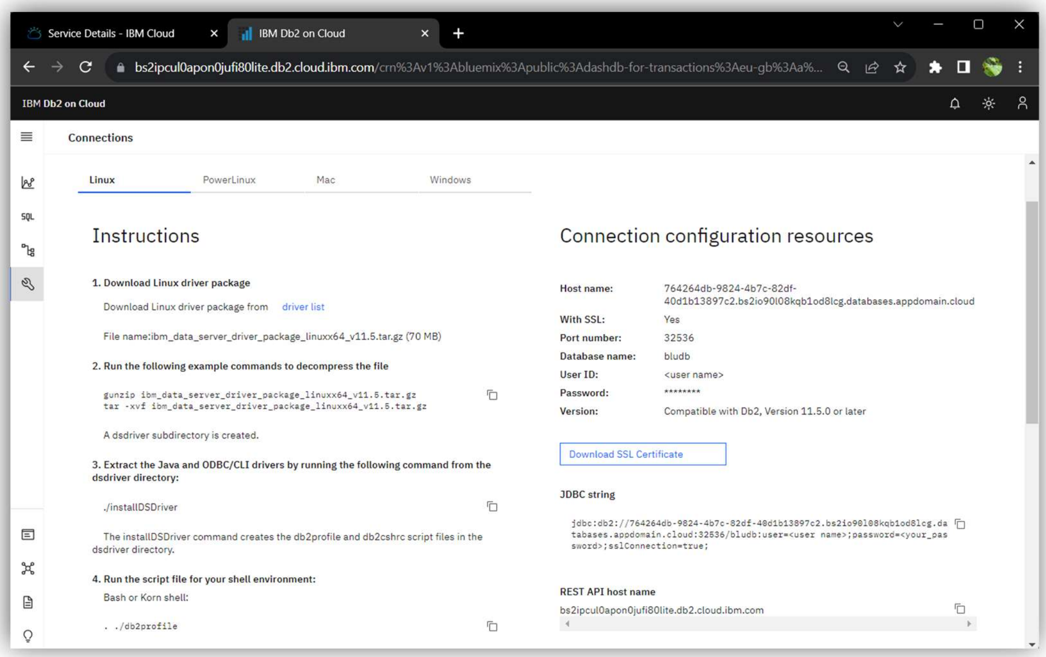
Set Up Db2 Connection:

In the Flask application (app.py), set up a connection to the Db2 database using the credentials provided by your Db2 service on IBM Cloud.

The connection details include the hostname, port, database name, username, and password. You typically fetch these details from environment variables or configuration files.

Perform Database Operations:

With a successful Db2 connection, you can execute SQL queries and perform database operations in your Flask application. For example, you can retrieve data, insert records, and update the database.


Service Details - IBM Cloud
IBM Db2 on Cloud
bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%...
IBM Db2 on Cloud
Connections
Linux
PowerLinux
Mac
Windows

Instructions

1. Download Linux driver package

Download Linux driver package from driver list

File name:ibm_data_server_driver_package_linuxx64_v11.5.tar.gz (70 MB)

2. Run the following example commands to decompress the file

```
gunzip ibm_data_server_driver_package_linuxx64_v11.5.tar.gz
tar -xvf ibm_data_server_driver_package_linuxx64_v11.5.tar.gz
```

A dsdriver subdirectory is created.

3. Extract the Java and ODBC/CLI drivers by running the following command from the dsdriver directory:

```
./installDSDriver
```

The installDSDriver command creates the db2profile and db2cshrc script files in the dsdriver directory.

4. Run the script file for your shell environment:

Bash or Korn shell:

```
. ./db2profile
```

Connection configuration resources

Host name:	764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
With SSL:	Yes
Port number:	32536
Database name:	bludb
User ID:	<user name>
Password:	********
Version:	Compatible with Db2, Version 11.5.0 or later

Download SSL Certificate

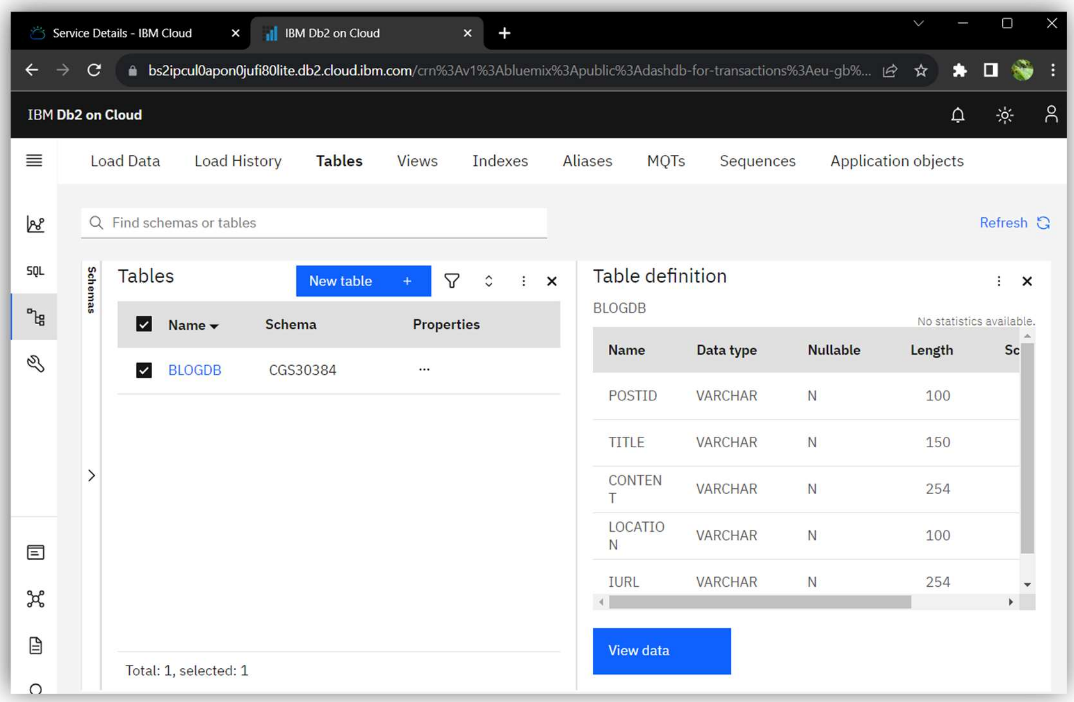
JDBC string

```
jdbc:db2://764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.da
tabases.appdomain.cloud:32536/bludb:user=<user name>;password=<your_pas
sword>;sslConnection=true;
```

REST API host name

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com

Fig-3: Connection Details for the DB2



Name	Schema	Properties
☑ BLOGDB	CGS30384	···

Total: 1, selected: 1

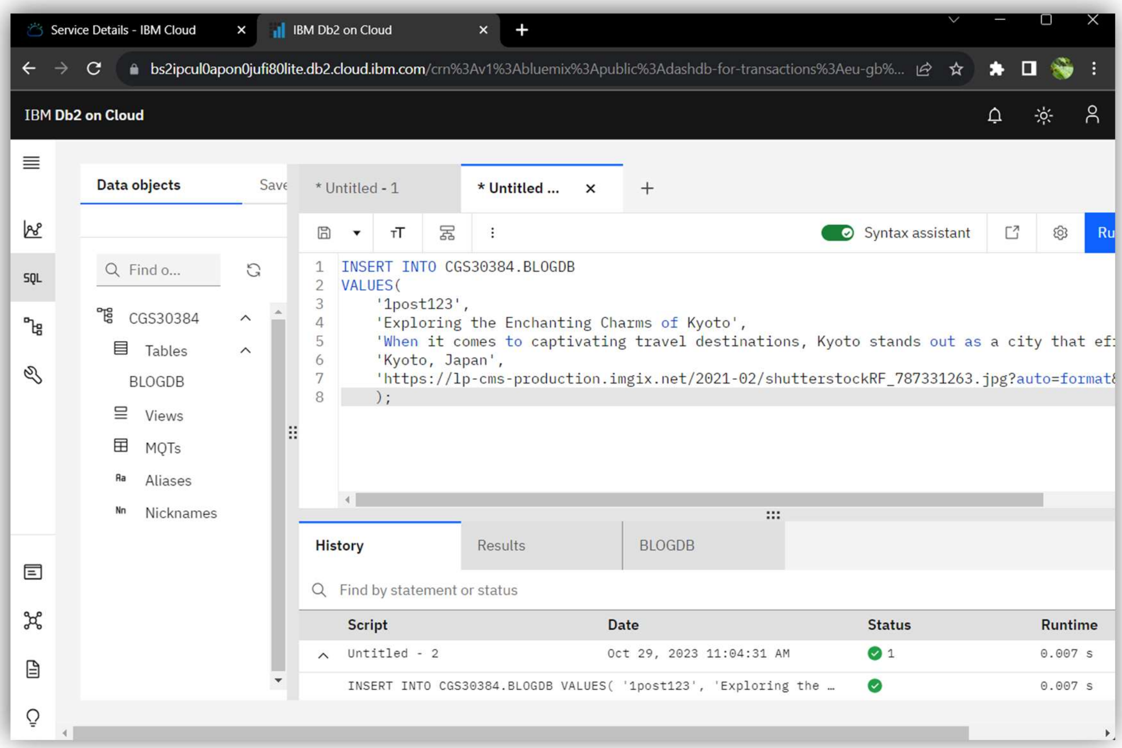
Table definition

BLOGDB

No statistics available.

Name	Data type	Nullable	Length	Sc
POSTID	VARCHAR	N	100	
TITLE	VARCHAR	N	150	
CONTEN T	VARCHAR	N	254	
LOCATIO N	VARCHAR	N	100	
IURL	VARCHAR	N	254	

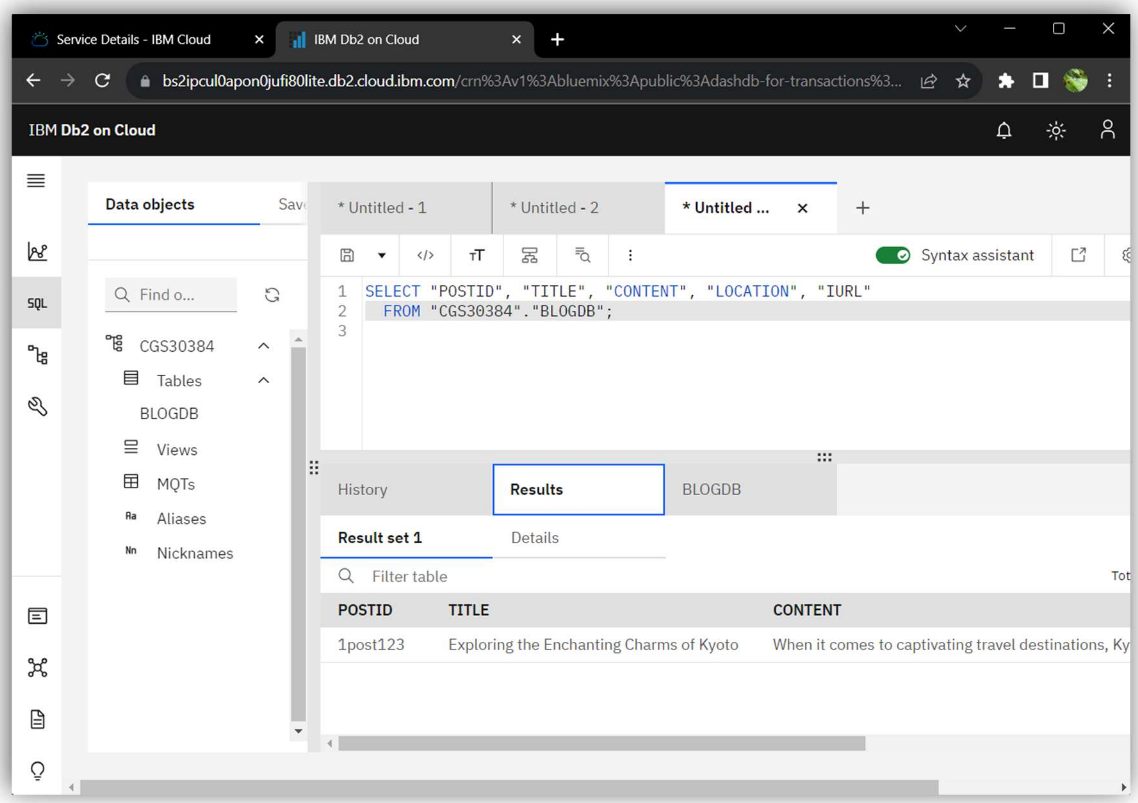
Fig-4: Table for the DB storage



```sql
INSERT INTO CGS30384.BLOGDB
VALUES(
    '1post123',
    'Exploring the Enchanting Charms of Kyoto',
    'When it comes to captivating travel destinations, Kyoto stands out as a city that ef
    'Kyoto, Japan',
    'https://lp-cms-production.imgix.net/2021-02/shutterstockRF_787331263.jpg?auto=format&
    );
```

```
Script                                              Date                        Status    Runtime
Untitled - 2                                        Oct 29, 2023 11:04:31 AM    ✓ 1       0.007 s
INSERT INTO CGS30384.BLOGDB VALUES( '1post123', 'Exploring the …   ✓                     0.007 s
```

Fig-5: Testing – Inserting a post to the table using SQL



```sql
SELECT "POSTID", "TITLE", "CONTENT", "LOCATION", "IURL"
  FROM "CGS30384"."BLOGDB";
```

POSTID	TITLE	CONTENT
1post123	Exploring the Enchanting Charms of Kyoto	When it comes to captivating travel destinations, Ky

Fig-6: Testing – Querying posts from the table using SQL

Error Handling and Security:

Ensure you have proper error handling for database operations, as well as security practices like parameterized queries to prevent SQL injection.

Deploy the Site:

Once your Flask application is ready and configured to connect to the Db2 service on IBM Cloud, you can deploy it to your preferred hosting environment.

By integrating IBM Cloud's Db2 service with your Flask application, can able to build a secure and scalable web application with a reliable database backend. This combination provides a powerful and flexible solution for web application development and data management.

Program Section:

File Structure:

It contains "static," "templates," and "app.py" sections.

- static/
    - CSS and JavaScript files
- templates/
    - HTML templates
- app.py
    - Main Flask application file
- Dockerfile
    - Docker configuration
- requirements.txt
    - List of Python package dependencies
- DigiCertGlobalRootCA.crt
    - Certificate authority file for secure database connections

Templates Directory:

The "templates" directory is where it stores the HTML templates. Flask uses a template engine, typically Jinja2, to render these templates dynamically with data and variables, allowing it to create dynamic web pages with consistent structures and layouts.

It contains, four files namely base.html, index.html, create.html and post.html.

<u>base.html</u>

```
<!DOCTYPE html>

<html lang="en">
```

```
<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link
href="https://fonts.googleapis.com/css2?family=Cinzel+Decorative:wght@700&family=Roboto:wgh
t@300;400;500;700&display=swap" rel="stylesheet">

    {% block head %}{% endblock %}

</head>

<body>

    {% block body %}{% endblock %}

</body>

</html>
```

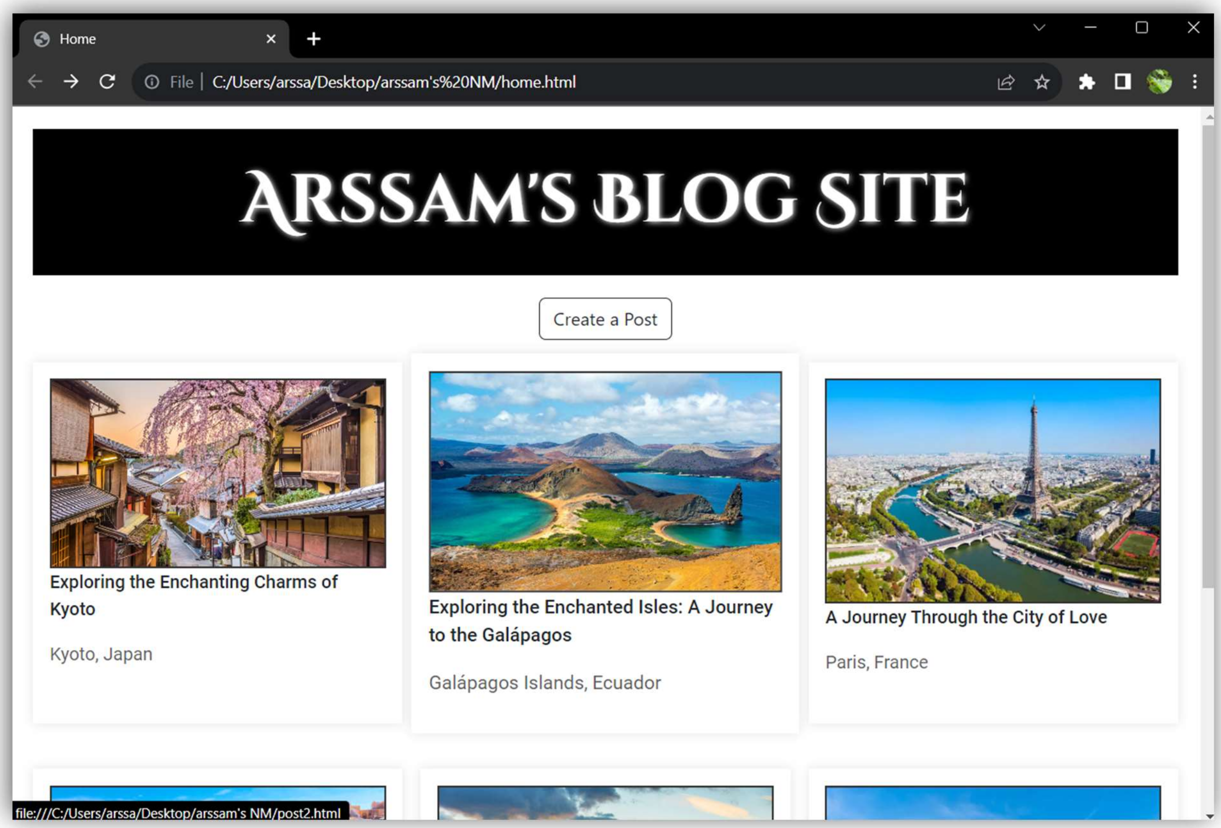

Home
File | C:/Users/arssa/Desktop/arssam's%20NM/home.html
ARSSAM'S BLOG SITE
Create a Post
Exploring the Enchanting Charms of Kyoto
Kyoto, Japan
Exploring the Enchanted Isles: A Journey to the Galápagos
Galápagos Islands, Ecuador
A Journey Through the City of Love
Paris, France
file:///C:/Users/arssa/Desktop/arssam's NM/post2.html

Fig-7: Home page of the Personal Travel Blog Website

index.html

```
{% extends 'base.html' %}

{% block head %}

    <title>Home</title>

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

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

{% endblock %}

{% block body %}

    <header>Arssam's Blog Site</header>


    <div class="create-btn">

        <a href="/create" class="btn btn-outline-dark" type="button">Create a Post</a>

    </div>

    <div class="all-posts-container">

        {% for list in lists %}

        <div class="post-container">

            <a href="/post/{{ list.postid }}" class="post-preview">

                <div class="picture-space">

                    <img

                        class="picture"

                        src="{{ list.iurl }}"

                        alt="{{ list.postid }} image"

                        width="500px"

                    />

                </div>

                <div class="post-info">

                    <p class="post-title">{{ list.title }}</p>

                    <p class="post-location">{{ list.location }}</p>

                </div>

            </a>

        </div>

        {% endfor %}

    </div>


{% endblock %}
```

create.html

```
{% extends 'base.html' %}


{% block head %}

    <title>New Post</title>

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

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

{% endblock %}


{% block body %}


<header>Arssam's Blog Site</header>


<h1>Create a New Post</h1>


<form action="/create" method="post">

    <div class="form-floating mb-3 mt-3">

        <input name="title" class="form-control" placeholder="Title of the post" required>

        <label for="floatingInput">Title</label>

    </div>


    <div class="form-floating mb-3">

        <input name="postid" class="form-control" placeholder="example123" required>

        <label for="floatingPassword">Post ID</label>

    </div>


    <div class="form-floating mb-3">

        <textarea name="content" class="form-control" placeholder="Describe your
journey..." style="height: 250px" required></textarea>

        <label for="floatingTextarea2">Post Content</label>

    </div>


    <div class="form-floating mb-3">

        <input name="location" class="form-control" placeholder="City, Country" required>

        <label for="floatingPassword">Location</label>
    </div>

    <div class="input-group mb-3">
        <span class="input-group-text">URL</span>
        <div class="form-floating">
            <input name="iurl" type="text" class="form-control"
placeholder="https://imagesource..." required>
            <label for="floatingInputGroup1">Image Link</label>
        </div>
    </div>

    <button type="submit" class="btn btn-outline-primary">Create</button>
</form>
{% endblock %}
```

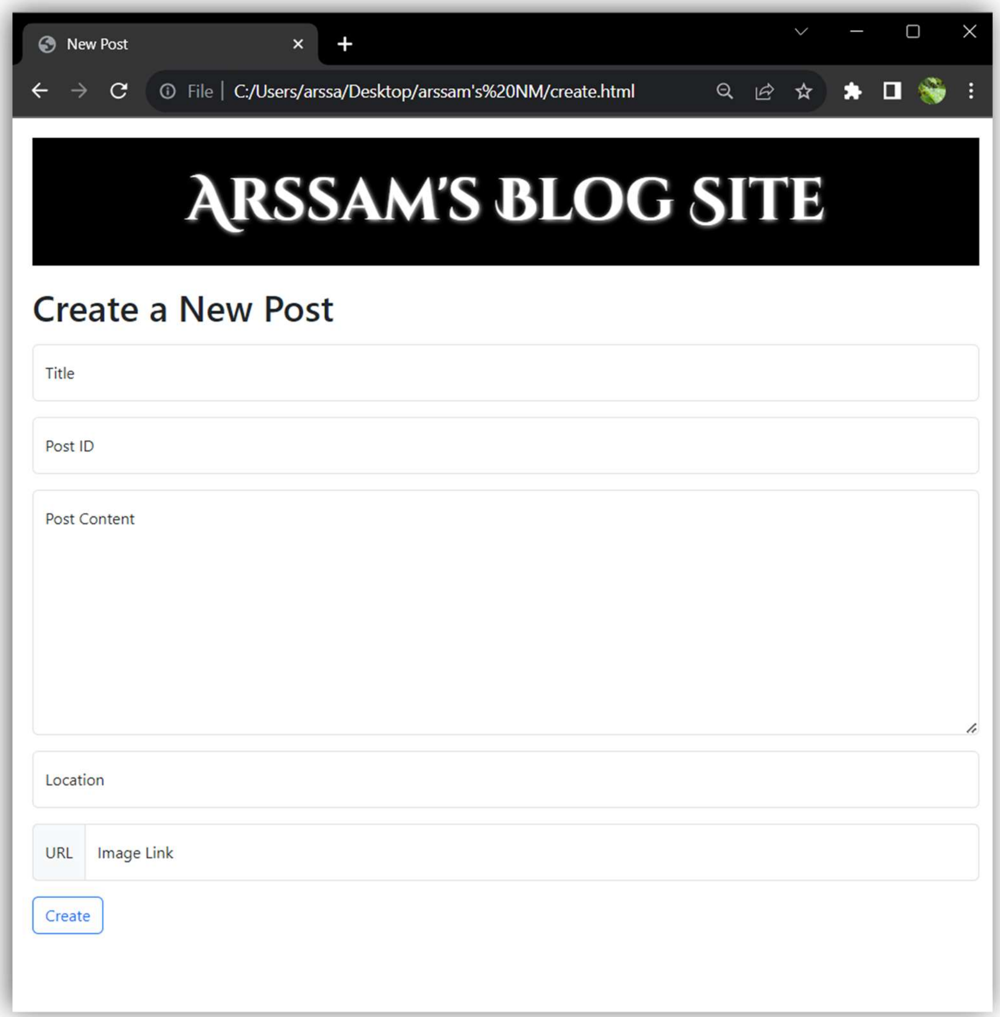
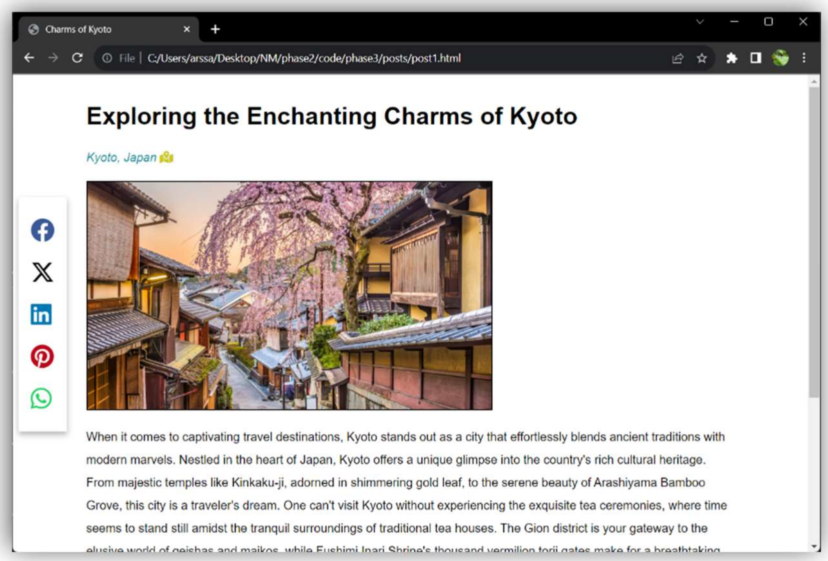


Fig-8: Creation page for the new post

post.html

```
{% extends 'base.html' %}


{% block head %}

    <title>{{ post.title }}</title>

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

    <link rel="stylesheet" href="{{ url_for('static', filename='main.js') }}" />

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.2/css/all.min.css"/>

    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRA
D@24,400,0,0" />

{% endblock %}


{% block body %}

    <div class="share-btn-container">

      <a href="#" class="facebook-btn">

        <i class="fab fa-facebook"></i>

      </a>

      <a href="#" class="twitter-btn">

        <i class="fab fa-x-twitter"></i>

      </a>

      <a href="#" class="linkedin-btn">

        <i class="fab fa-linkedin"></i>

      </a>


      <a href="#" class="pinterest-btn">

        <i class="fab fa-pinterest"></i>

      </a>


      <a href="#" class="whatsapp-btn">

        <i class="fab fa-whatsapp"></i>

      </a>


    </div>
    <div class="content">
      <h1 id="title">{{ post.title }}</h1>

      <p>
        <a href="#" class="maps-btn">
          <i id="location">{{ post.location }}</i>
          <i class="fa-solid fa-map-location-dot"></i>
        </a>

      </p>

      <div class="main-content-container">
        <img
        class="img"
        src="{{ post.iurl }}"
        alt="{{ post.postid }} Image"
        />

        <p>
          {{ post.content }}
        </p>

      </div>
    </div>

    <div class="comment-container">
      <h2>Comments</h2>
      <div class="container">
        <input id="commentInput" type="text" placeholder="Add a comment...">
        <span class="material-symbols-outlined send" onclick="addComment()">send</span>
      </div>
      <div id="comments"></div>
    </div>

    <script src="{{ url_for('static', filename='main.js') }}"></script>
{% endblock %}
```


Charms of Kyoto

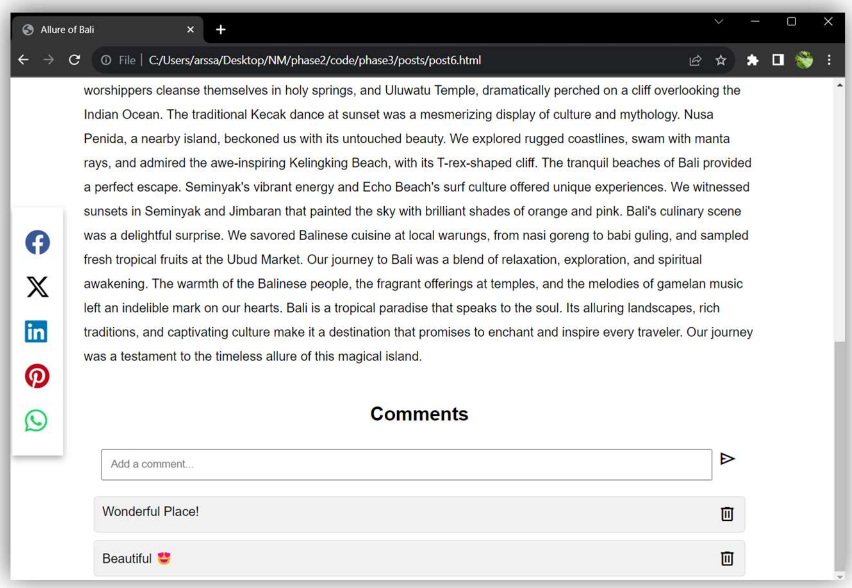
C:/Users/arssa/Desktop/NM/phase2/code/phase3/posts/post1.html

Exploring the Enchanting Charms of Kyoto

Kyoto, Japan

When it comes to captivating travel destinations, Kyoto stands out as a city that effortlessly blends ancient traditions with modern marvels. Nestled in the heart of Japan, Kyoto offers a unique glimpse into the country's rich cultural heritage. From majestic temples like Kinkaku-ji, adorned in shimmering gold leaf, to the serene beauty of Arashiyama Bamboo Grove, this city is a traveler's dream. One can't visit Kyoto without experiencing the exquisite tea ceremonies, where time seems to stand still amidst the tranquil surroundings of traditional tea houses. The Gion district is your gateway to the elusive world of geishas and maikos, while Fushimi Inari Shrine's thousand vermilion torii gates make for a breathtaking

Fig-9: Post page for the travel blog


Allure of Bali

C:/Users/arssa/Desktop/NM/phase2/code/phase3/posts/post6.html

worshippers cleanse themselves in holy springs, and Uluwatu Temple, dramatically perched on a cliff overlooking the Indian Ocean. The traditional Kecak dance at sunset was a mesmerizing display of culture and mythology. Nusa Penida, a nearby island, beckoned us with its untouched beauty. We explored rugged coastlines, swam with manta rays, and admired the awe-inspiring Kelingking Beach, with its T-rex-shaped cliff. The tranquil beaches of Bali provided a perfect escape. Seminyak's vibrant energy and Echo Beach's surf culture offered unique experiences. We witnessed sunsets in Seminyak and Jimbaran that painted the sky with brilliant shades of orange and pink. Bali's culinary scene was a delightful surprise. We savored Balinese cuisine at local warungs, from nasi goreng to babi guling, and sampled fresh tropical fruits at the Ubud Market. Our journey to Bali was a blend of relaxation, exploration, and spiritual awakening. The warmth of the Balinese people, the fragrant offerings at temples, and the melodies of gamelan music left an indelible mark on our hearts. Bali is a tropical paradise that speaks to the soul. Its alluring landscapes, rich traditions, and captivating culture make it a destination that promises to enchant and inspire every traveler. Our journey was a testament to the timeless allure of this magical island.

Comments

Add a comment...

Wonderful Place!

Beautiful 😍

Fig-10: Post page contains share buttons and comment Section

Static Directory:

The "static" directory is where it stores static assets such as CSS (Cascading Style Sheets), JavaScript, images, and other files that don't change dynamically. This directory is used to serve static files to the client's web browser, enhancing the styling and interactivity of your web pages.

It contains, three files namely home.css, style.css, create.css and main.js.

home.css

```
.post-info {
    font-family: Roboto, Arial;
    font-size: 16px;
}
.post-title {
    font-weight: 500;
}
.post-location {
    color: rgb(92, 92, 92);
}

.post-container {
    padding: 15px;
    background-color: rgb(255, 255, 255);
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    transition: transform .2s;
}

.post-container:hover {
    transform: scale(1.05);
}

.all-posts-container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    column-gap: 16px;
    row-gap: 40px;
}

a {
    text-decoration: inherit;
    color: inherit;
}

.picture {
    border: 2px solid rgb(52, 52, 52);
    width: 100%;
}

@media (max-width: 450px) {
    .all-posts-container {
        grid-template-columns: 1fr;
    }
}

@media (min-width: 451px) and (max-width: 750px) {
    .all-posts-container {
        grid-template-columns: 1fr 1fr;
    }
}

@media (min-width: 751px) and (max-width: 1249px) {
    .all-posts-container {
        grid-template-columns: 1fr 1fr 1fr;
    }
}

@media (min-width: 1250px) {
    .all-posts-container {
        grid-template-columns: 1fr 1fr 1fr 1fr;
    }
}

header {
    text-align: center;
    font-family: 'Cinzel Decorative', Arial, Helvetica, sans-serif;
    font-size: 60px;
    font-weight: 700;
    background-color: rgb(0, 0, 0);
    color: rgb(255, 255, 255);
    padding: 20px;
    margin-bottom: 20px;

    text-shadow: 1px 1px 5px rgb(255, 255, 255);


}


.create-btn {

    display: flex;

    justify-content: center;

    margin-bottom: 20px;

}


body {

    background-color: rgb(245, 245, 245);

    padding: 20px;

}
```

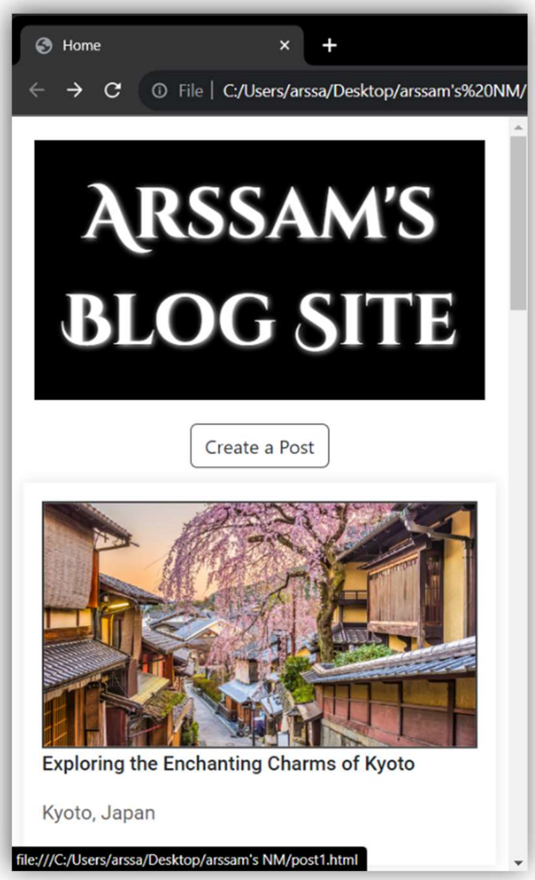

Home
File
C:/Users/arssa/Desktop/arssam's%20NM/
ARSSAM'S BLOG SITE
Create a Post
Exploring the Enchanting Charms of Kyoto
Kyoto, Japan
file:///C:/Users/arssa/Desktop/arssam's NM/post1.html

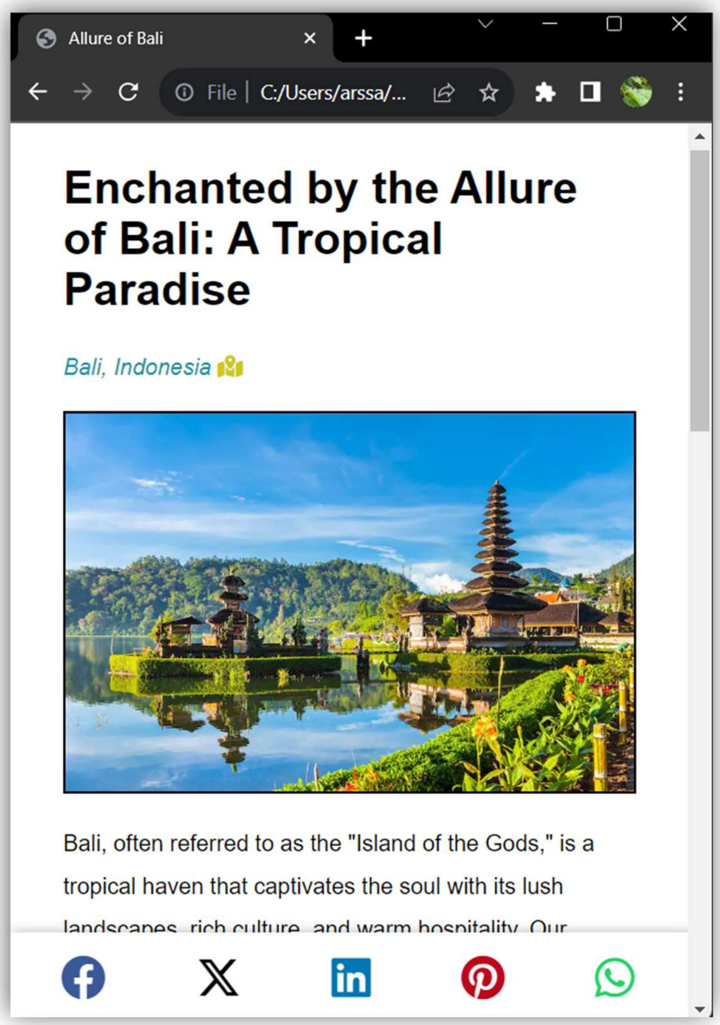
Fig-11: Responsive design for the home page

style.css

```
/* Content */

.content {
  padding: 8px 90px;
  font-family: "Roboto", sans-serif;
}

.content p {
  line-height: 1.9;
}

.content img {
  max-height: 500px;
}

/* Share Buttons */

.share-btn-container {
  background: #fff;
  display: flex;
  flex-direction: column;
  padding: 16px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
  position: fixed;
  top: 50%;
  transform: translateY(-50%);
}

.share-btn-container a i {
  font-size: 32px;
}

.share-btn-container a {
  margin: 12px 0;
  transition: 500ms;
}

.share-btn-container a:hover, .maps-btn :hover{
  transform: scale(1.2);
}

.share-btn-container .fa-facebook {
  color: #3b5998;
}

.share-btn-container .fa-x-twitter {
  color: #000000
}

.share-btn-container .fa-linkedin {
  color: #0077b5;
}

.share-btn-container .fa-pinterest {
  color: #bd081c;
}

.share-btn-container .fa-whatsapp {
  color: #25d366;
}

.maps-btn .fa-map-location-dot {
  color: #cdca15;
  transition: 500ms;
}

.maps-btn {
  text-decoration: none;
  color: #1c95a2;
}
```

```
.img {

  width: 50vw;
  max-width: 100%;
  border: 2px solid rgba(0, 0, 0);
}

/* Media Queries */

@media (max-width: 550px) {
  .content {
    padding: 8px 32px;
  }

  .share-btn-container {
    transform: unset;
    top: unset;
    left: 0;
    bottom: 0;
    width: 100%;
    flex-direction: row;
    box-shadow: 4px 0 8px rgba(0, 0, 0, 0.3);
    padding: 16px 0;
    justify-content: center;
  }

  .share-btn-container a {
    margin: 0 32px;
  }

  .img {
    width: 100vw;
    max-width: 100%;
    border: 2px solid rgba(0, 0, 0);
  }
}
```

```
/* Comment Section */

.container{
  display: flex;
      justify-content: space-between;
      padding: 10px;
      margin: 10px 0;
}
.comment-container {
    width: 80%;
    margin: 0 auto;
    font-family: Arial, sans-serif;
}
.comment-container h2 {
    text-align: center;
}
#commentInput{
    flex: 0.99;
    padding:10px;
}
.comment {
    display: flex;
    justify-content: space-between;
    background-color: #f2f2f2;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ddd;
    border-radius: 5px;
}
.comment-actions {
    display: flex;
}
.delete-button {
    background-color: #ff5757;
    color: white;
    border: none;
    padding: 5px 10px;
    margin-left: 5px;
    cursor: pointer;
}
.material-symbols-outlined: hover{
    cursor: pointer;
}
.send{
    font-size:30px;
}
```



Enchanted by the Allure of Bali: A Tropical Paradise

Bali, Indonesia

Bali, often referred to as the "Island of the Gods," is a tropical haven that captivates the soul with its lush

Fig-12: Responsive design for the post page

create.css

```
body {
    padding: 20px;
}


header {
    text-align: center;
    font-family: 'Cinzel Decorative', Arial, Helvetica, sans-serif;
    font-size: 60px;
    font-weight: 700;
    background-color: rgb(0, 0, 0);
    color: rgb(255, 255, 255);
    padding: 20px;
    margin-bottom: 20px;
    text-shadow: 1px 1px 5px rgb(255, 255, 255);

}
```

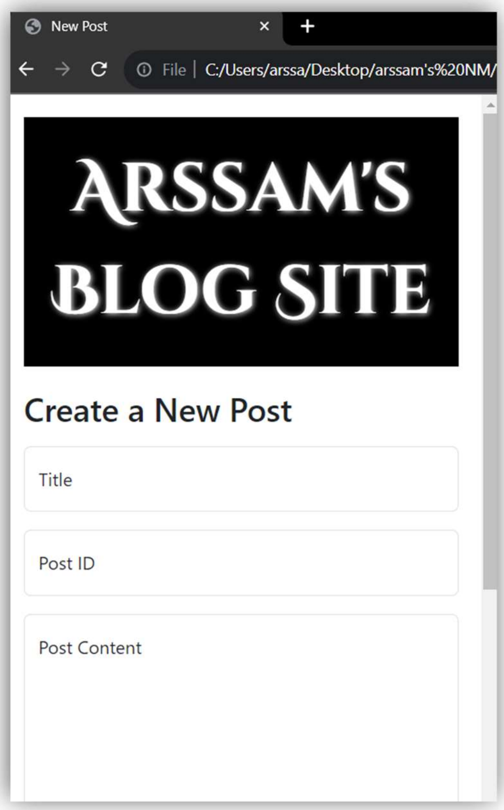


Fig-13: Responsive design for the post creation page

main.js

```
// Share and Maps section
const facebookBtn = document.querySelector(".facebook-btn");
const twitterBtn = document.querySelector(".twitter-btn");
const pinterestBtn = document.querySelector(".pinterest-btn");
const linkedinBtn = document.querySelector(".linkedin-btn");
const whatsappBtn = document.querySelector(".whatsapp-btn");
const mapsBtn = document.querySelector(".maps-btn");

function init() {
  const img = document.querySelector(".img");

  let postUrl = encodeURI(document.location.href);
  let postTitle = encodeURI(document.getElementById("title").textContent);
  let postImg = encodeURI(img.src);
  let location = encodeURI(document.getElementById("location").textContent);

  facebookBtn.setAttribute(
    "href",
    `https://www.facebook.com/sharer.php?u=${postUrl}`
  );

  twitterBtn.setAttribute(
    "href",
    `https://twitter.com/share?url=${postUrl}&text=${postTitle}`
  );

  pinterestBtn.setAttribute(
    "href",
`https://pinterest.com/pin/create/bookmarklet/?media=${postImg}&url=${postUrl}&description=
${postTitle}`
  );

  linkedinBtn.setAttribute(
    "href",
    `https://www.linkedin.com/shareArticle?url=${postUrl}&title=${postTitle}`
  );

  whatsappBtn.setAttribute(
    "href",
    `https://wa.me/?text=${postTitle} ${postUrl}`
  );

  mapsBtn.setAttribute(
    "href",
    `https://www.google.com/maps?q=${location}&ie=UTF8`
  );
}

init();

//Comment Section
commentInput.addEventListener("keyup",   (event)   =>   {if   (event.key   ==   "Enter")
{addComment();}});

function addComment() {
  const commentInput = document.getElementById("commentInput");
  const commentText = commentInput.value.trim();

  if (commentText !== "") {
      const commentsContainer = document.getElementById("comments");
      const commentDiv = document.createElement("div");
      commentDiv.classList.add("comment");
      commentDiv.innerHTML = `
          <div>${commentText}</div>
          <div class="comment-actions">
              <span                                       class="material-symbols-outlined"
onclick="deleteComment(this)">delete</span>
          </div>
      `;
      commentsContainer.appendChild(commentDiv);
      commentInput.value = "";
      window.scrollTo(0, document.body.scrollHeight);
  }
}
function deleteComment(button) {
  const commentDiv = button.closest(".comment");
  commentDiv.remove();
}
```

Manual Installation:

By using the Python package manager, pip, to install Flask. If you do not have Python installed, you should install Python first. Here are the steps:

Create a Virtual Environment (Optional):

It is a good practice to create a virtual environment to manage your project's dependencies. This step is optional but recommended.

# Create a virtual environment

```
python -m venv myenv
```

Activate the virtual environment

```
source myenv/bin/activate
```

# On Windows,

```
myenv\Scripts\activate
```

Install Flask:

After activating your virtual environment, you can install Flask using pip:

```
pip install Flask
```

Install ibm_db for IBM Db2:

To use ibm_db with Flask for IBM Db2 integration, you need to install this package. You will also need the IBM Db2 client, which should be installed on your system. Follow these steps:

Install IBM Db2 Client:

You should have the IBM Db2 client installed on your system. You can download the client from the IBM website and follow the installation instructions for your operating system.

Install ibm_db:

Once you have the IBM Db2 client installed, you can install ibm_db using pip:

```
pip install ibm_db
```

app.py

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

import ibm_db


global conn


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURITY=SSL;SSLServ
erCertificate=DigiCertGlobalRootCA.crt;UID=cgs30384;PWD=pMs2fEZ5sMZzIk6g;",'','')


app = Flask(__name__)


@app.route('/')

def index():


    lists = []


    sql = "SELECT * FROM CGS30384.BLOGDB"

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.execute(stmt)


    while True:

        row = ibm_db.fetch_assoc(stmt)

        if row is None:

            break

        lists.append(row)


    return render_template('index.html', lists=lists)
```

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

    if request.method == 'POST':

        postid = request.form['postid']
        title = request.form['title']
        content = request.form['content']
        location = request.form['location']
        iurl = request.form['iurl']

        sql = "INSERT INTO CGS30384.BLOGDB(postid, title, content, location, iurl)
VALUES(?, ?, ?, ?, ?)"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, postid)
        ibm_db.bind_param(stmt, 2, title)
        ibm_db.bind_param(stmt, 3, content)
        ibm_db.bind_param(stmt, 4, location)
        ibm_db.bind_param(stmt, 5, iurl)

        if ibm_db.execute(stmt):
            print("New Post was inserted into the database successfully")
        else:
            print("Failed to insert the new post!")

    else:
        return render_template('create.html')

@app.route('/post/<string:postid>')
def post_view(postid):

    sql = "SELECT * FROM CGS30384.BLOGDB WHERE POSTID = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, postid)
    ibm_db.execute(stmt)
    post = ibm_db.fetch_assoc(stmt)
    if post:
        return render_template('post.html', post=post)
    else:
        return redirect('/')

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

Dockerfile is used to define the environment and instructions for creating a Docker image. A Docker image is a standalone executable package that contains everything needed to run a piece of software, including the code, runtime, libraries, and system tools.

A requirements.txt file is a text file used in Python projects to specify and document the project's dependencies. This file is crucial for managing the Python packages that your project relies on.

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

```
docker build -t blog-site-app .
```

Dockerfile

```
#syntax=docker/dockerfile:1
FROM python:alphine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5000
ENTRYPOINT [ "python" ]
CMD [ "app.py" ]
```

requirements.txt

```
ibm-db==3.2.0

ibm-db-sa==0.4.0

Flask==3.0.0

Flask-SQLAlchemy==3.1.1

Click==8.1.7

Jinja2==3.1.2

SQLAlchemy==2.0.22

Werkzeug==3.0.1
```

Conclusion:

The creation of the personal travel blog hosted on IBM Cloud as a web app represents a compelling showcase of the capabilities of IBM Cloud. This project has been a journey of integration and innovation, combining various technologies and cloud services to bring this blog to life.

The use of Flask to manage routes, templates, and connect with the DB2 Database demonstrates the flexibility and power of web application frameworks. Containerizing the application with Docker and utilizing Kubernetes for container management reflects a commitment to scalability and robust infrastructure.

The outcome of this project is a dynamic, user-friendly travel blog that showcases a passion for travel and adventure. By converting HTML content into easily updatable template files, the website becomes a platform for sharing experiences and inspiring others to explore the world.

Ultimately, this travel blog is not only a personal venture but also a testament to the capabilities of IBM Cloud, serving as an example of how cloud services can empower individuals to create and share their unique content with the world. It is a testament to the power of technology to enable and amplify our passions, in this case, the love for travel and adventure.