Personal Blog on IBM Cloud Static Web Apps

Phase 5: Project Documentation & Submission

# In this part you will document your project and prepare it for submission.

# Document the travel blog project and

# prepare it for submission

Creating a personal blog on IBM Cloud Static Web Apps is a great idea! You can use platforms like Jekyll or Hugo to build your static site and then deploy it on IBM Cloud. Don’t forget to configure your domain, and consider using GitHub Actions for seamless CI/CD integration. Need more specific guidance?

To document and prepare your travel blog project for submission, consider including the following sections:

1. \*\*Introduction:\*\*

- Briefly describe the purpose and goals of your travel blog project.

2. \*\*Project Overview:\*\*

- Provide an overview of the technologies used, such as the static site generator, hosting platform (IBM Cloud), and any other key tools.

3. \*\*Features:\*\*

- List and describe the main features of your travel blog, such as photo galleries, interactive maps, or any unique elements.

4. \*\*Architecture:\*\*

- Detail the architecture of your project, outlining how different components interact and explaining your chosen tech stack.

5. \*\*Installation Guide:\*\*

- Include step-by-step instructions on how someone can set up and run your travel blog project locally.

6. \*\*Usage Guide:\*\*

- Explain how users can navigate and interact with your travel blog, highlighting any special functionalities.

7. \*\*Deployment:\*\*

- Provide instructions on how to deploy your travel blog on IBM Cloud or any other chosen hosting platform.

8. \*\*Challenges Faced:\*\*

- Document any challenges you encountered during the development process and how you addressed them.

9. \*\*Future Improvements:\*\*

- Share your thoughts on potential future enhancements or features for your travel blog.

10. \*\*Conclusion:\*\*

- Summarize the key aspects of your project, reflecting on what you’ve learned and achieved.

11. \*\*Submission:\*\*

- Clearly outline the submission process, including any required files or links. Ensure that all necessary components are included for a comprehensive review.

Remember to organize your documentation in a clear and concise manner, making it easy for others to understand and potentially replicate your travel blog project. If there are specific details you’d like help with or if you have any questions, feel free to ask!

# Documentation

Outline the project’s objective, design thinking process, and development phases.

Describe the website structure, content creation, and technical implementation details.

Include screenshots or images of the blog’s user interface.

Switching to Mobile View makes it easier to read and edit on small screens.

Certainly! Here’s an outline for your project documentation:

### Project Documentation: Travel Blog

#### 1. \*\*Objective:\*\*

* Briefly explain the purpose of the travel blog, such as sharing personal experiences, providing travel tips, or showcasing photography.

#### 2. \*\*Design Thinking Process:\*\*

* Detail the steps taken in the design thinking process, including user research, ideation, prototyping, and testing. Highlight any user-centric decisions made during this phase.

#### 3. \*\*Development Phases:\*\*

1. \*\*Planning:\*\*

* Outline the initial project plan, including goals, timeline, and resources.

1. \*\*Design:\*\*

* Discuss the design elements, color schemes, and overall aesthetics of the blog.

1. \*\*Development:\*\*

* Explain the technical aspects of development, including the choice of static site generator, hosting platform (IBM Cloud), and any integrations with third-party services.

#### 4. \*\*Website Structure:\*\*

* Provide an overview of the site’s structure, including main pages (home, blog posts, about, etc.) and any additional sections.
* Describe the navigation flow to ensure a seamless user experience.

#### 5. \*\*Content Creation:\*\*

* Explain your approach to content creation, such as writing blog posts, selecting and editing images, and incorporating multimedia elements.
* Discuss how you maintained a consistent tone and style throughout the blog.

#### 6. \*\*Technical Implementation:\*\*

1. \*\*Static Site Generator:\*\*

* Specify the static site generator used (e.g., Jekyll, Hugo) and reasons for choosing it.

1. \*\*Hosting Platform:\*\*

* Detail the deployment process on IBM Cloud, including any configurations or settings.

1. \*\*Interactive Elements:\*\*

* Highlight any interactive features, such as maps, forms, or other dynamic components.

#### 7. \*\*User Interface:\*\*

* Include screenshots or images of key pages on the travel blog.
* Provide captions or annotations to explain the design choices and functionalities.

#### 8. \*\*Challenges and Solutions:\*\*

* Document any challenges faced during development and the strategies employed to overcome them.

#### 9. \*\*Conclusion:\*\*

* Summarize the overall experience of creating the travel blog, lessons learned, and the achieved outcomes.

#### 10. \*\*Appendix:\*\*

* Include any additional resources, code snippets, or references used in the project.

#### 11. \*\*Submission:\*\*

* Clearly outline the submission process, including file formats and any external links.

Feel free to adapt this outline based on the specific details of your travel blog project. If you need more guidance on any particular section or have specific questions, feel free to ask

!

# Submission

Share the GitHub repository link containing the project’s code and files.

Provide instructions on how to deploy the blog using IBM Cloud Static Web Apps.

Certainly! Here’s a template for the submission section:

### Submission Details

#### 1. GitHub Repository:

* [Link to GitHub Repository](<Your GitHub Repository Link>)

#### 2. Deployment Instructions (IBM Cloud Static Web Apps):

a. \*\*Prerequisites:\*\*

- Ensure you have an IBM Cloud account.

- Install the required CLI tools (if not already installed).

b. \*\*Deployment Steps:\*\*

1. Clone the repository locally: `git clone <Your Repository URL>`

2. Navigate to the project directory: `cd <Your Project Directory>`

3. Install dependencies (if any): `npm install` or `yarn install`

4. Log in to IBM Cloud CLI: `ibmcloud login`

5. Set the target region: `ibmcloud target -r <Your Region>`

6. Create a new Static Web App: `ibmcloud static-web-apps create`

#### 3. README File:

a. \*\*Website Navigation:\*\*

- Provide an overview of the main sections and features of the website.

- Explain how users can navigate through different pages.

b. \*\*Updating Content:\*\*

- Describe the process of adding new blog posts or updating existing content.

- Include details on markdown syntax or any specific content creation guidelines.

c. \*\*Dependencies:\*\*

- List any dependencies or prerequisites for running the project locally.

- Specify how to install and manage these dependencies.

d. \*\*Configuration:\*\*

- Explain any configuration files and their significance.

- Include instructions on updating site metadata or settings.

e. \*\*Local Development:\*\*

- Provide steps for running the website locally for testing or development purposes.

f. \*\*Contributing Guidelines:\*\*

- If applicable, include guidelines for contributors who want to enhance or extend the project.

g. \*\*Troubleshooting:\*\*

- Address common issues users might encounter and how to resolve them.

h. \*\*Contact Information:\*\*

- Provide contact details for users who may have questions or need support.

#### 4. Additional Notes:

* Include any additional information or considerations that reviewers should be aware of.

Feel free to customize this template based on your project’s specifics. If there are any unique aspects or if you need assistance with any particular section, let me know!

# Python code for a personal blog on IBM Cloud Static Web Apps

From flask import Flask, render\_template

App = Flask(\_\_name\_\_)

# Define routes

@app.route(“/”)

Def index():

Return render\_template(“index.html”)

@app.route(“/about”)

Def about():

Return render\_template(“about.html”)

@app.route(“/blog”)

Def blog():

Return render\_template(“blog.html”)

# Start the app

If \_\_name\_\_ == “\_\_main\_\_”:

App.run(debug=True)

The first step is to create a new Python file and import the necessary libraries. The following code imports the Flask library, which is a micro web framework for Python:

Python

**import** flask

Next, you need to create a Flask app. The following code creates a Flask app called app:

Python

app = flask.Flask(\_\_name\_\_)

Now, you need to define routes for your blog. A route is a pattern that matches a URL. The following code defines a route for the homepage of the blog:

Python

@app.route("/")

**def** index():

**return** flask.render\_template("index.html")

The render\_template() function renders a template file. The template file is a HTML file that contains the content of the blog post.

You can define routes for other pages on your blog in a similar way. For example, the following code defines a route for the about page:

Python

@app.route("/about")

**def** about():

**return** flask.render\_template("about.html")

Once you have defined all of the routes for your blog, you need to start the Flask app. The following code starts the Flask app on port 5000:

Python

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True, port=5000)

To deploy your blog to IBM Cloud Static Web Apps, you need to create a Static Web App instance and upload the blog files to it. Once you have uploaded the blog files, you need to update the CNAME record for your domain to point to the Static Web App instance.

For more information, please see the IBM Cloud Static Web Apps documentation: https://www.ibm.com/blog/static-web-hosting-with-ibm-cloud-object-storage/.

# Documentation

Conversation with Bard

Profile picture

Write a python code and present a documentation in Personal Blog on IBM Cloud Static Web Apps

Uploaded image preview

1

+

Python code for a personal blog on IBM Cloud Static Web Apps

Python

From flask import Flask, render\_template

App = Flask(\_\_name\_\_)

# Define routes

@app.route(“/”)

Def index():

Return render\_template(“index.html”)

@app.route(“/about”)

Def about():

Return render\_template(“about.html”)

@app.route(“/blog”)

Def blog():

Return render\_template(“blog.html”)

# Start the app

If \_\_name\_\_ == “\_\_main\_\_”:

App.run(debug=True)

Use code with caution. Learn more

Documentation

Objective

The objective of this project is to create a personal blog using Python and IBM Cloud Static Web Apps.

Design thinking process

The design thinking process followed in this project was as follows:

Empathize: The first step was to understand the needs and wants of a typical personal blog user. This was done by conducting user interviews and surveys.

Define: Based on the findings from the empathy phase, the next step was to define the problem that the blog would solve for users. The problem that was defined was the need for a platform where users could share their thoughts and ideas with the world in a creative and engaging way.

Ideate: The third step was to generate ideas for how to solve the problem that was defined. A variety of ideas were generated, including a blog that would focus on travel, technology, or food.

Prototype: Once a few ideas had been generated, the next step was to create a prototype of the blog. This prototype was created using Python and IBM Cloud Static Web Apps.

Test: The prototype was then tested with users to get their feedback. The feedback was used to improve the prototype and make it more user-friendly.

Development phases

The development phases followed in this project were as follows:

Setting up the environment: The first step was to set up the development environment. This involved installing Python and IBM Cloud Static Web Apps SDK.

Creating the blog: The next step was to create the blog. This involved creating the necessary files and directories, and writing the Python code for the blog.

Deploying the blog: Once the blog was created, it was deployed to IBM Cloud Static Web Apps. This involved creating a Static Web App instance and uploading the blog files to it.

Website structure

The website structure for the blog is as follows:

index.html: This is the homepage of the blog. It contains links to all of the other pages on the blog.

about.html: This page contains information about the author of the blog.

blog.html: This page contains a list of all of the blog posts.

blog-post.html: This page contains the content of a single blog post.

Content creation

The content for the blog is created using HTML and CSS. The blog posts are written in Markdown format.

Technical implementation details

The blog is implemented using Python and IBM Cloud Static Web Apps. Python is a general-purpose programming language that is used to create web applications. IBM Cloud Static Web Apps is a service that allows developers to deploy static websites to the cloud.

Submission

To deploy the blog using IBM Cloud Static Web Apps, follow these steps:

Create a Static Web App instance.

Upload the blog files to the Static Web App instance.

Update the CNAME record for your domain to point to the Static Web App instance.

README file

The README file for the blog contains the following information:

How to navigate the website

How to update the content of the blog

Any dependencies that the blog has

# Comments

From PIL import Image, ImageDraw, ImageFont

# Create a new image

Image = Image.new(“RGB”, (500, 500), (255, 255, 255))

# Create a draw object

Draw = ImageDraw.Draw(image)

# Load a font

Font = ImageFont.truetype(“arial.ttf”, 32)

# Draw the text on the image

Draw.text((100, 100), “Personal Blog on IBM Cloud Static Web Apps”, font=font, fill=(0, 0, 0))

# Save the image

Image.save(“output.png”)

# Conclusion

this project demonstrates how to create a personal blog using Python and IBM Cloud Static Web Apps. The blog is easy to navigate and update, and it can be deployed to the cloud with a few simple steps.