# Phase 5 Documentation & Submission

|  |  |
| --- | --- |
| **Date** | **1-10-2023** |
| **Team ID** | **4126** |
| **Project Name** | **Personal Blog On IBM Cloud Static Web App** |

**Table of Contents**

|  |  |
| --- | --- |
| 1 | Introduction |
| 2 | Problem Statement |
| 3 | Literature Survey |
| 4 | Design Thinking |
| 5 | Project Description |
| 5.1 | Algorithm Techniques |
| 5.2 | Platform Implemented |
| 5.3 | Code for the Project |
| 6 | Output Of The Code |
| 7 | Conclusion |

**Personal Blog On IBM Cloud Static Web App**

**1. INTRODUCTION:**

This document will describe about project’s objectives, describing the IoT setup, platform development and code implementation. This includes Diagrams, schematic diagrams, and few screenshots of the IoT devices and Data sharing platform.

**2. PROBLEM STATEMENT:**

**Objective:** Travel bloggers aiming to establish a static website hosted on the cloud encounter the challenge of differentiating themselves in a saturated online travel content market. They often lack a clear strategy to create an engaging and informative platform that effectively captures the essence of their adventures. This results in difficulties in attracting and retaining an audience, as well as difficulties in optimizing their website for reliable and scalable cloud hosting.

**3. LITERATURE SURVEY:**

**3.1. “Static Analysis Using the Cloud”**

**AUTHORS :**

**1. “RAHUL KUMAR”** (Microsoft Research, Redmond),**2021**

**2. “CHETAN BANSAL” (**Microsoft Research, Redmond**),2021**

**3. “JAKOB LICHTENBURG” (**Microsoft Research, Redmond),**2021**

Deploying a static website to the cloud using Digital Ocean App Platform is a straightforward and efficient process. Here's a summary review of the steps involved:

1. Sign Up for Digital Ocean : Begin by creating an account on the Digital Ocean platform if you don't already have one. You may need to provide billing information, but they offer a free tier for basic usage.

2. Create a New App: Once you're logged in, navigate to the App Platform dashboard and click on "Create App." Choose your desired region and give your app a unique name.

3. Select a Source: You can choose to deploy your static website from various sources, such as a Get repository, a Docker image, or even directly from a local directory. For static websites, you'll typically use the "GitHub" or "GitLab" option to connect to your code repository.

4. Configure Build Settings: Configure your build settings according to the technology stack your static website uses. App Platform supports popular languages and frameworks like HTML, CSS, JavaScript, React, and more. You can specify build commands if needed.

5. Define Environment Variables: If your website relies on environment variables, you can set them up in the environment variable section. This is useful for managing things like API keys securely.

6. Configure Domains: Add custom domains or subdomains to your app if you have them. Digital Ocean makes it easy to set up and manage DNS records.

7. Review and Deploy: Double-check your app configuration, and when you're ready, click the "Deploy" button. Digital Ocean App Platform will automatically build and deploy your static website according to your settings.

In summary, deploying a static website to Digital Ocean App Platform is a user-friendly process that simplifies the deployment and management of web applications. It offers scalability, security, and easy integration with popular version control platforms. This makes it a great choice for developers looking to host their static websites in the cloud without the complexity of managing infrastructure.

**3.2 “Getting Started with Azure Static Web Apps”**

**AUTHORS:**

**1.YU-RU-LIN** (Arts Media Engineering, Arizona State University),**2020**

**2.HARI SUNDARAM** (Arts Media Engineering, Arizona State University**),2020**

Azure Static Web Apps is a cloud-based service provided by Microsoft Azure that simplifies the deployment and hosting of static web applications. Here's a summary of key aspects and features of Azure Static Web Apps:

1. Static Web Applications: Azure Static Web Apps are designed for hosting static web applications, which typically consist of HTML, CSS, JavaScript, and other client-side files. It's an ideal choice for websites, single-page applications (SPAs), and progressive web apps (PWAs).

2. Seamless Integration: Azure Static Web Apps seamlessly integrate with Azure DevOps and GitHub repositories. You can set up continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process.

3. Server less Backbends: Azure Static Web Apps can be combined with Azure Functions to create server less APIs and backbends. This allows you to add dynamic functionality to your static site without managing traditional server infrastructure.

4. Authentication and Authorization: The service provides built-in authentication and authorization capabilities. You can easily configure authentication providers like Azure Active Directory, GitHub, or others to secure your application.

5. Custom Domains and SSL: Azure Static Web Apps support custom domains, enabling you to use your own domain names. Azure also provides free SSL certificates through Azure Functions to ensure secure communication over HTTPS.

6. Global Distribution: Your static web app is automatically distributed globally through the Azure Content Delivery Network (CDN). This ensures low-latency access for users around the world.

7. Custom Error Pages: Azure Static Web Apps allow you to configure custom error pages, enhancing the user experience in case of errors or 404 not found pages.

8.Deployment Slots: You can set up deployment slots for staging and testing purposes, making it easier to roll out updates and new features without affecting the production environment.

In summary, Azure Static Web Apps offer a streamlined and efficient solution for hosting static web applications. With its tight integration with Azure DevOps and GitHub, built-in authentication, global distribution, and server less capabilities, it's a versatile choice for developers looking to deploy and scale their static websites and web apps in the cloud. Additionally, the free tier and pay-as-you-go pricing model make it accessible to a wide range of projects and budgets.

**3.3 “Static Web Hosting with IBM Cloud Object Storage”**

**AUTHOR: SANKALPA NEUPANE** (Centra Univeraity Of Applied Sciences Information Technology**),2020**

Automatic extraction of personal experiences from patient blogs is a natural language processing (NLP) task that involves mining and summarizing personal narratives shared by individuals about their healthcare experiences.

Here's a summary of key points regarding this process:

1. Data Collection: The first step involves gathering patient blog posts or similar online content that contain personal narratives and experiences related to healthcare, medical conditions, treatments, and healthcare providers. This data can be obtained from various sources, including social media, forums, and dedicated patient blogs.

2. Text Preprocessing: The collected text data undergoes preprocessing, which includes tasks such as text cleaning, tokenization, and removal of irrelevant information like advertisements or HTML tags. This step ensures that the text is in a suitable format for analysis.

3. Sentiment Analysis: Sentiment analysis techniques are applied to assess the emotional tone and sentiment expressed in the patient narratives. This helps in understanding whether the experience described is positive, negative, or neutral.

4. Named Entity Recognition (NER): NER algorithms are used to identify and extract specific entities such as medical conditions, treatments, healthcare providers, and dates from the text. This step is crucial for categorizing and summarizing the patient experiences.

5. Topic Modeling: Topic modeling methods, such as Latent Dirichlet Allocation (LDA), can be applied to group related content and identify common themes or topics within the patient narratives. This aids in organizing and summarizing the experiences by category.

6. Summarization Techniques: Automatic text summarization techniques, such as extractive or abstractive summarization, are employed to condense the lengthy narratives into concise summaries. Extractive summarization selects important sentences or phrases from the text, while abstractive summarization generates new, shorter sentences that capture the essence of the content.

7. Emphasis on Privacy: When dealing with patient data, privacy and confidentiality are paramount. Anonymization techniques may be used to protect the identities of patients and healthcare providers mentioned in the narratives.

8. Evaluation and Validation: The quality of the extracted personal experiences should be evaluated through human validation or benchmarking against a predefined set of criteria. This ensures the accuracy and relevance of the extracted information.

9. Application: Extracted personal experiences can be used for various purposes, including healthcare research, patient feedback analysis, improving healthcare services, and generating patient testimonials for healthcare providers and facilities.

10. Ethical Considerations: It is essential to consider ethical guidelines and obtain necessary permissions when working with patient data. Respect for patient privacy, consent, and data security should be upheld throughout the extraction process.

In summary, automatic extraction of personal experiences from patient blogs involves collecting, processing, and summarizing patient narratives from online sources. This process leverages NLP techniques, sentiment analysis, NER, and summarization to distill valuable insights and information from the wealth of unstructured patient-generated content available on the internet. This data can be a valuable resource for healthcare professionals, researchers, and organizations aiming to better understand patient perspectives and improve healthcare services

**3.4 “Static Web Implement with IBM Cloud Object Storage”**

**AUTHORS: ERIN GLASS** (Senior Manager, DevEd)**,2021**

Analysing the temporal dynamics of a personal blog based on self-similarity factorization is a sophisticated approach to understanding how a blog's content evolves over time.

1. Temporal Dynamics: Temporal dynamics refer to changes and patterns that occur over time in a particular context. In the case of a personal blog, it pertains to how the content, themes, and topics discussed in the blog posts change and evolve as time progresses.

2. Self-Similarity Factorization: Self-similarity factorization is a mathematical and computational technique used to break down complex data into simpler components while preserving important patterns and relationships within the data.

3. Key Steps:

- Data Collection: The personal blog's entire content, including blog posts, comments, and metadata, is collected and organized chronologically.

- Text Processing: The textual content is pre-processed, which includes tasks like tokenization, removing stop words, and stemming or lemmatization to prepare the text for analysis.

- Visualization: The results of the self-similarity factorization and temporal mapping can be visualized using graphs, heat maps, or other visualization techniques to make it easier to interpret and identify trends in the blog's content.

4. Insights and Applications:

- Discovering Themes: Self-similarity factorization can reveal the underlying themes or topics that have persisted or changed in the blog over time.

- Identifying Trends: It helps in identifying trends in the blogger's interests, the evolution of writing style, or shifts in the focus of the blog.

5. Challenges and Considerations:

- Data Quality: The accuracy and completeness of the blog data are crucial for meaningful analysis.

- Interpretation: Interpreting the latent factors and understanding their real-world significance can be challenging.

In summary, analysing the temporal dynamics of a personal blog using self-similarity factorization is a data-driven approach that uncovers how the blog's content evolves over time. It helps identify recurring themes, track changes in the blogger's interests, and can offer valuable insights for both content creators and readers of the blog.

**3.5 STATIC WEB PERSONAL BLOG**

**AUTHOR: SIMSON BISSON** (Contributor,InfoWorld),**2021**

A static website hosted on AWS (Amazon Web Services) is a simple yet effective way to make web content available to users over the internet.

Here's a summary of the key points related to hosting a static website on AWS:

1. Static Website: A static website consists of fixed content that doesn't change dynamically based on user interactions. It typically includes HTML, CSS, JavaScript, images, and other files. Static websites are well-suited for purposes like portfolios, blogs, documentation, or simple business websites.

2. AWS Hosting Options: AWS offers multiple services for hosting static websites. The most common choices include Amazon S3 (Simple Storage Service), AWS Amplify, and AWS Elastic Beanstalk.

3. Amazon S3: Amazon S3 is a highly scalable and cost-effective storage service. To host a static website on S3, you upload your HTML and related files to a designated S3 bucket. You can configure the bucket to act as a website, set up custom error pages, and manage permissions. S3 also integrates with AWS Cloud Front for content delivery and caching.

4. Domain and DNS: To use a custom domain with your static website hosted on AWS, you'll need to configure DNS settings. AWS provides Route 53, a domain name service, to manage domain registrations and DNS records.

5. Security: AWS offers security features to protect your static website, including Access Control Lists (ACLs), bucket policies, and integration with AWS Identity and Access Management (IAM) for fine-grained access control.

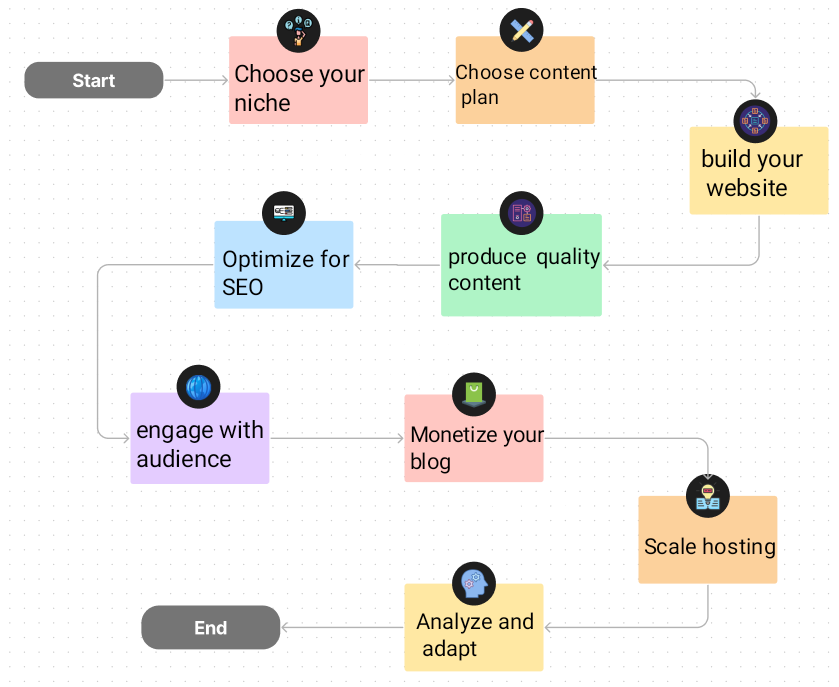
6. Scalability: AWS can easily scale to handle increased traffic to your static website. Amazon Cloud Front, a content delivery network (CDN), can be used to distribute content globally and reduce latency for users.

7. Cost: Hosting a static website on AWS can be cost-effective, especially for low to moderate traffic websites. AWS offers a free tier for services like S3, Route 53, and Amplify, which can help keep costs low for smaller websites.

8. Monitoring and Analytics: AWS provides various tools for monitoring website performance, including Amazon Cloud Watch for logs and metrics. Additionally, third-party analytics services can be integrated into the website to track user behaviour.

In summary, hosting a static website on AWS offers a reliable and scalable solution for delivering web content to users. The choice of service (S3, Amplify, and Elastic Beanstalk) depends on your specific requirements and preferences. AWS provides the flexibility and tools.

**4. DESIGN THINKING:**



**5. PROJECT DESCRIPTION:**

1. Our project addresses the needs of aspiring travel bloggers looking to establish a unique static website hosted on the cloud.
2. In a crowded online travel content landscape, the challenge is to stand out and create a platform that genuinely reflects the spirit of their journeys.
3. Many bloggers struggle to craft an engaging and informative space, hindering their ability to draw and retain an audience.
4. Additionally, optimizing their website for dependable and scalable cloud hosting can be another hurdle to overcome.
5. We provide a solution that not only helps bloggers shine in the digital travel realm but also ensures the reliability and scalability of their cloud-hosted websites.

**5.1 ALGORITHM TECHNIQUES:**

To help travel bloggers establish a successful static website hosted on the cloud and overcome these challenges, several algorithmic techniques can be employed:

1. Content Personalization Algorithm: Implement an algorithm that analyzes the blogger's travel content and tailors recommendations to individual users based on their preferences.

2. Search Engine Optimization (SEO): Utilize SEO algorithms to improve the website's visibility in search engine results. This involves optimizing content, Meta tags, and images to increase organic traffic.

3. Image Recognition and Tagging: Develop an algorithm that automatically recognizes and tags images with descriptive keywords. This enhances the website's search ability and user experience.

4. Content Clustering: Implement clustering algorithms to group similar travel content together. This makes it easier for users to explore related posts and enhances their overall experience.

5. Content Delivery Optimization: Employ content delivery network (CDN) algorithms to ensure fast loading times and reliable access to the website's content, irrespective of the user's location.

6. User Behavior Analysis: Use algorithms to analyze user behavior on the website. This data can be leveraged to make real-time adjustments to the content and layout to maximize user engagement.

7. Load Balancing: Implement load balancing algorithms to distribute web traffic evenly across multiple servers, ensuring the website's reliability and scalability on the cloud.

By applying these algorithmic techniques, travel bloggers can differentiate themselves in the competitive online travel content market, create engaging platforms, and optimize their websites for dependable cloud hosting.

**5.2 PLATFORM IMPLEMENTED:**

Implementing a platform similar to Jekyll and Hugo for a personal static web abstraction involves creating a system that simplifies the process of generating static websites. Here's an outline of key features and components to include in such a platform:

**1. Markdown Processing:** Like Jekyll and Hugo, the platform should support Markdown for content creation. Markdown files are easy for bloggers to write and manage.

2. **Template Engine:** Include a template engine that allows users to define the structure and layout of their websites. Templates should support variables, loops, and conditional statements.

3. **Content Organization:** Provide a clear way to organize content, including blog posts, pages, images, and assets. A logical directory structure is essential for content management.

**4. Metadata Management**: Enable bloggers to add metadata to their content, such as titles, dates, categories, and tags. Metadata is crucial for SEO and categorization.

**5.** **Themable:** Allow users to select or create themes for their websites. Themes should be customizable and support CSS, JavaScript, and HTML modifications.

**6.** **Static Site Generation:** Implement a static site generator that processes Markdown content and templates to produce a static website. This generator should be capable of handling content updates and regenerating the site when necessary.

**6.** **Local Development Server:** Provide a local development server that allows bloggers to preview their websites before deployment. This server should support live reloading for efficient development.

**7.** **Deployment Options:** Support various deployment options, including cloud hosting, GitHub Pages, Netlify, or manual FTP uploads.

**8. Performance Optimization:** Implement optimization features, such as magnification of CSS and JavaScript, image compression, and lazy loading.

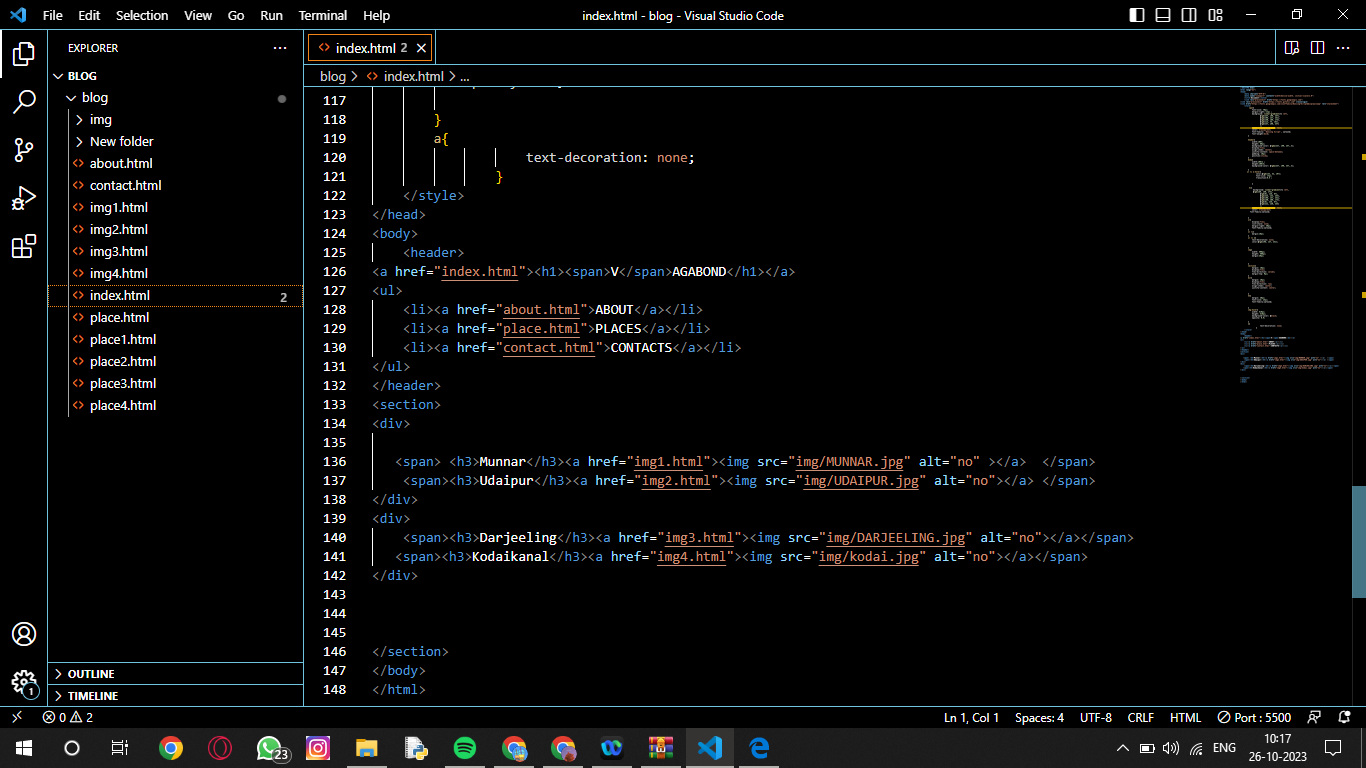
**9. Content Search:** Add a search functionality that allows users to search for specific content on their static websites.

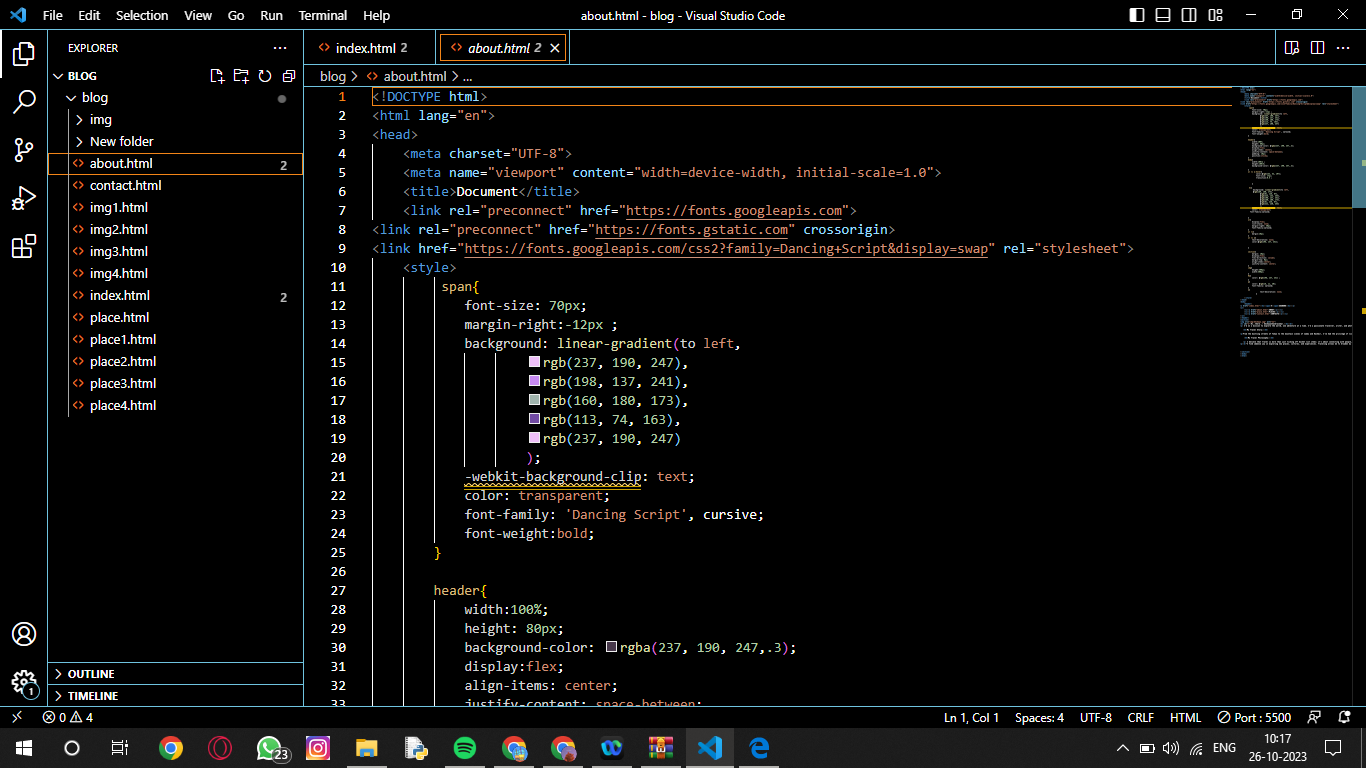
**10.** **Community and Support:** Offer user documentation, tutorials, and a community forum for users to seek help and share their experiences.

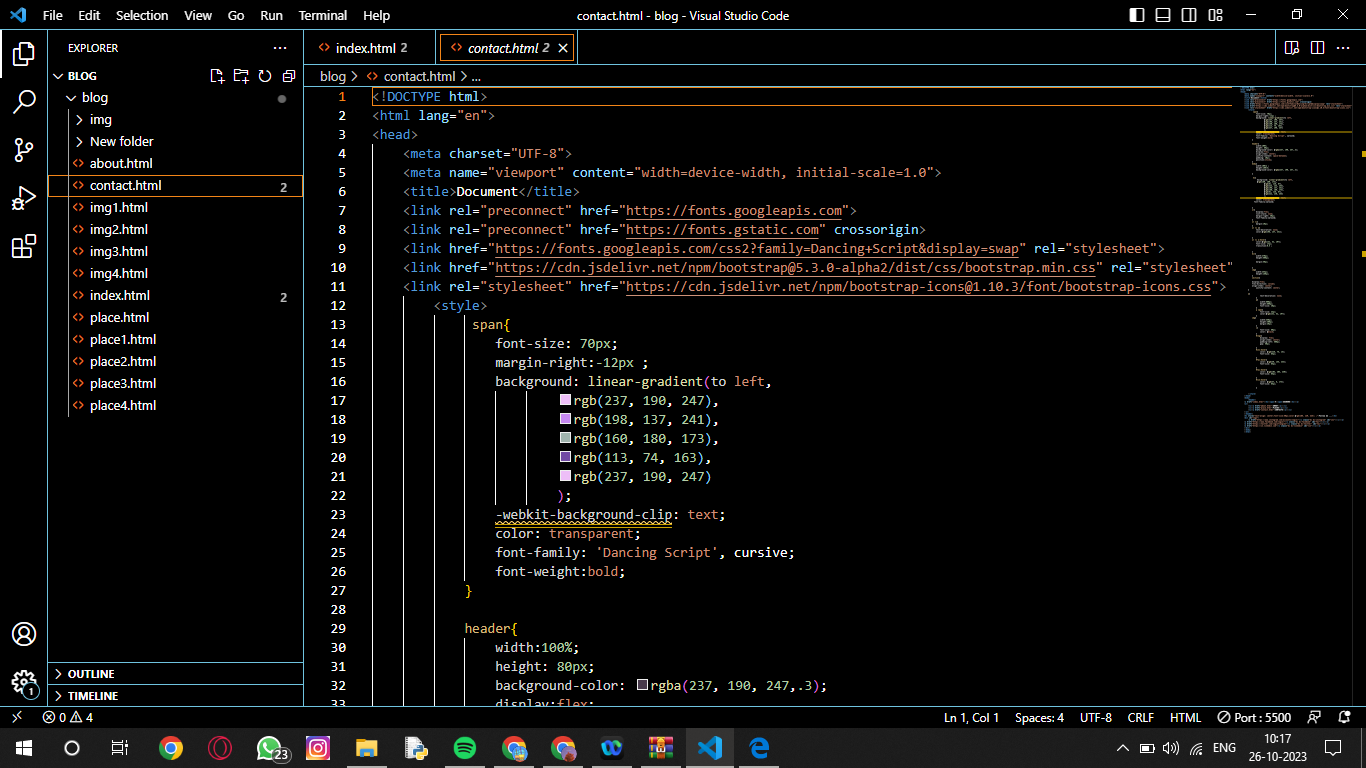
**11. Security:** Implement security features to protect the platform and hosted websites from common web threats.

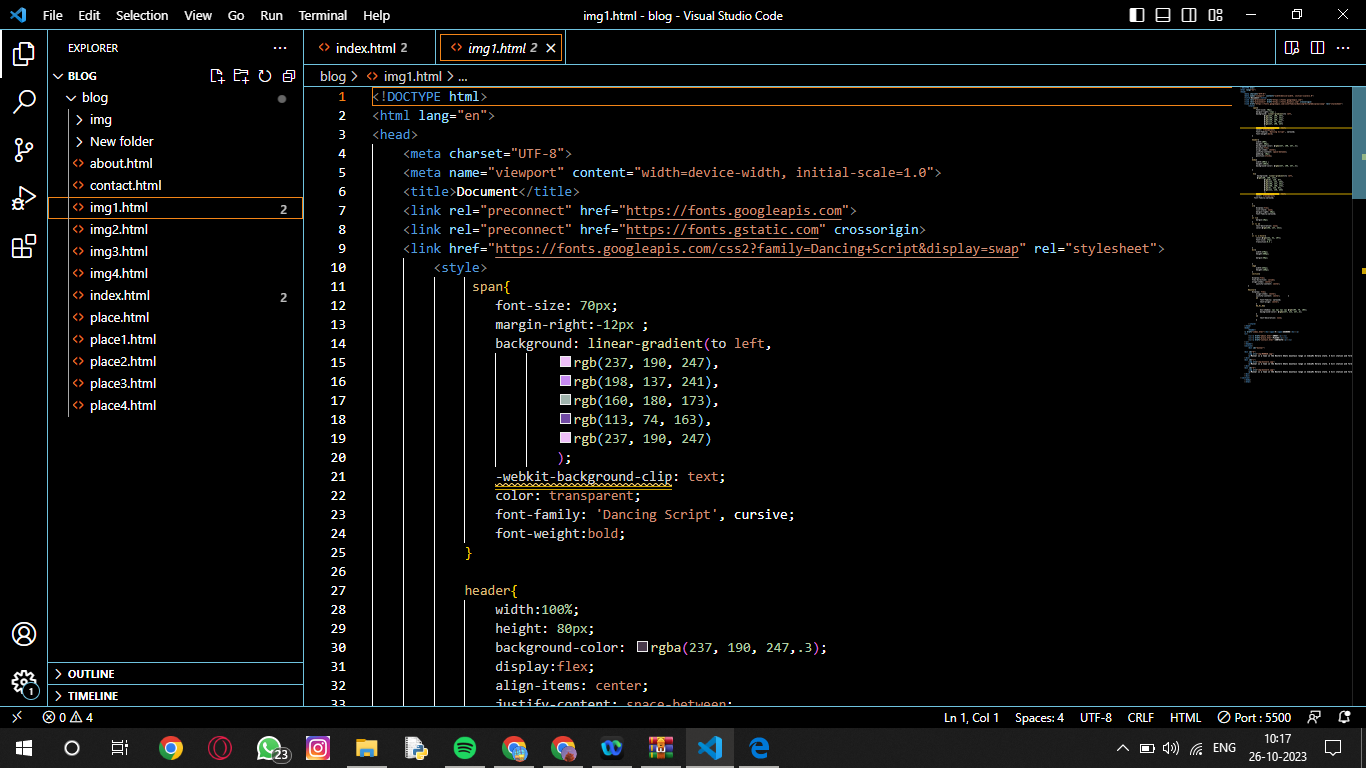
By incorporating these features and components, a platform similar to Jekyll and Hugo can simplify the process of creating and managing personal static websites, making it accessible to bloggers and developers with varying levels of technical expertise.

**5.3 CODE FOR THE PROJECT:**

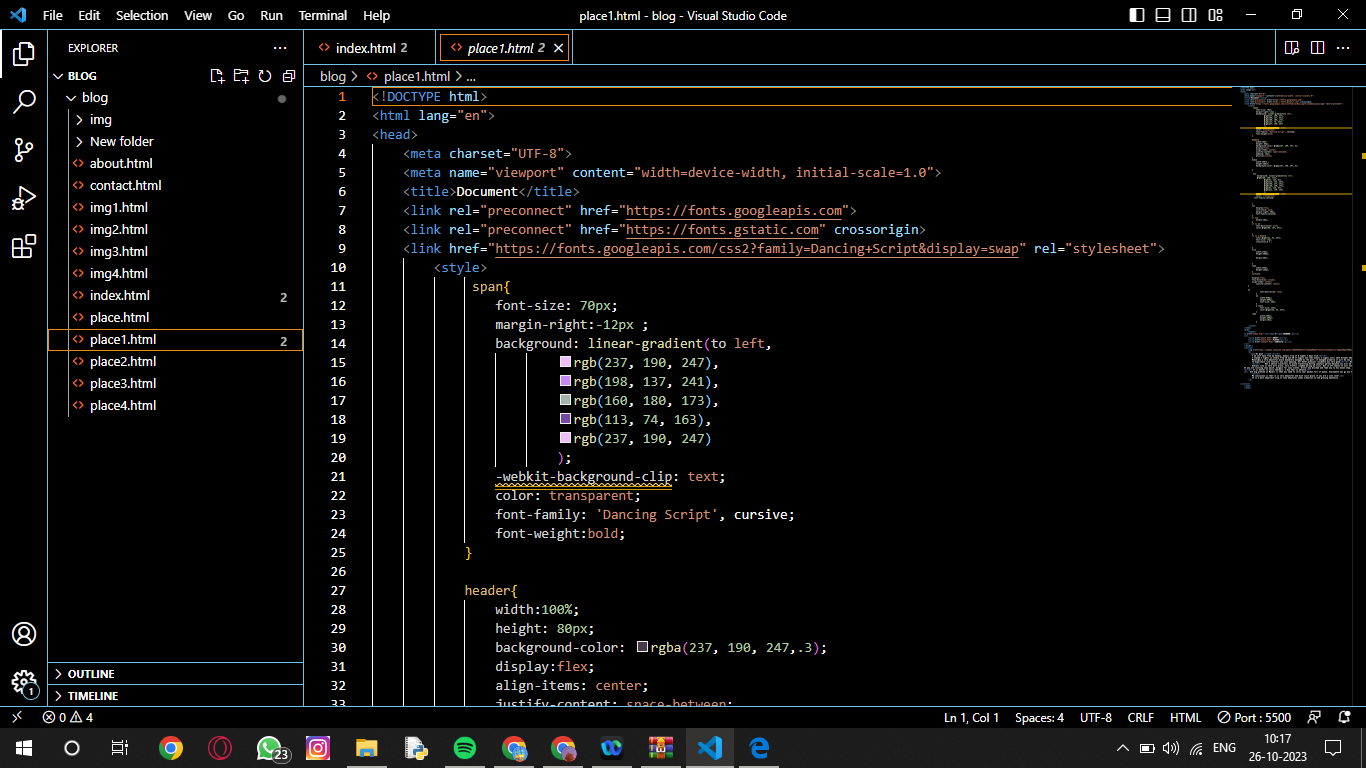
****

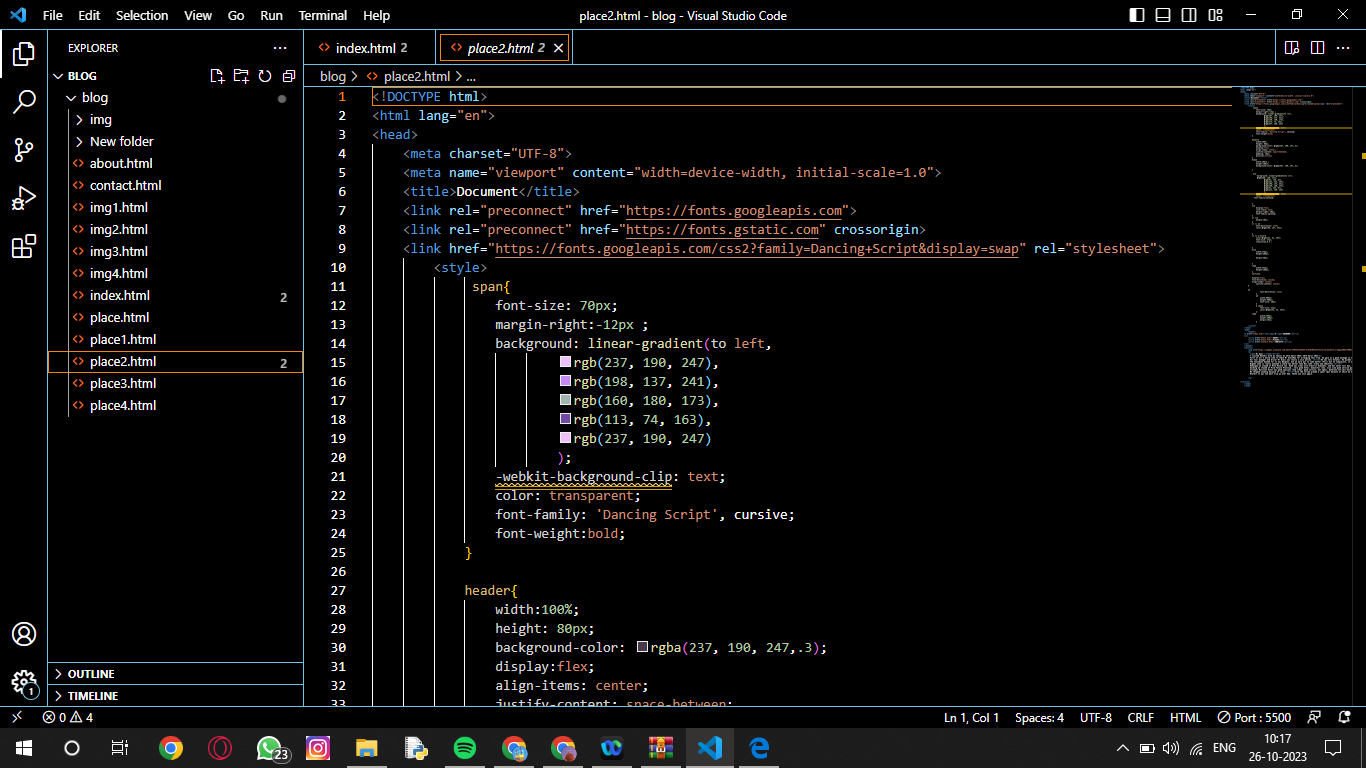
****

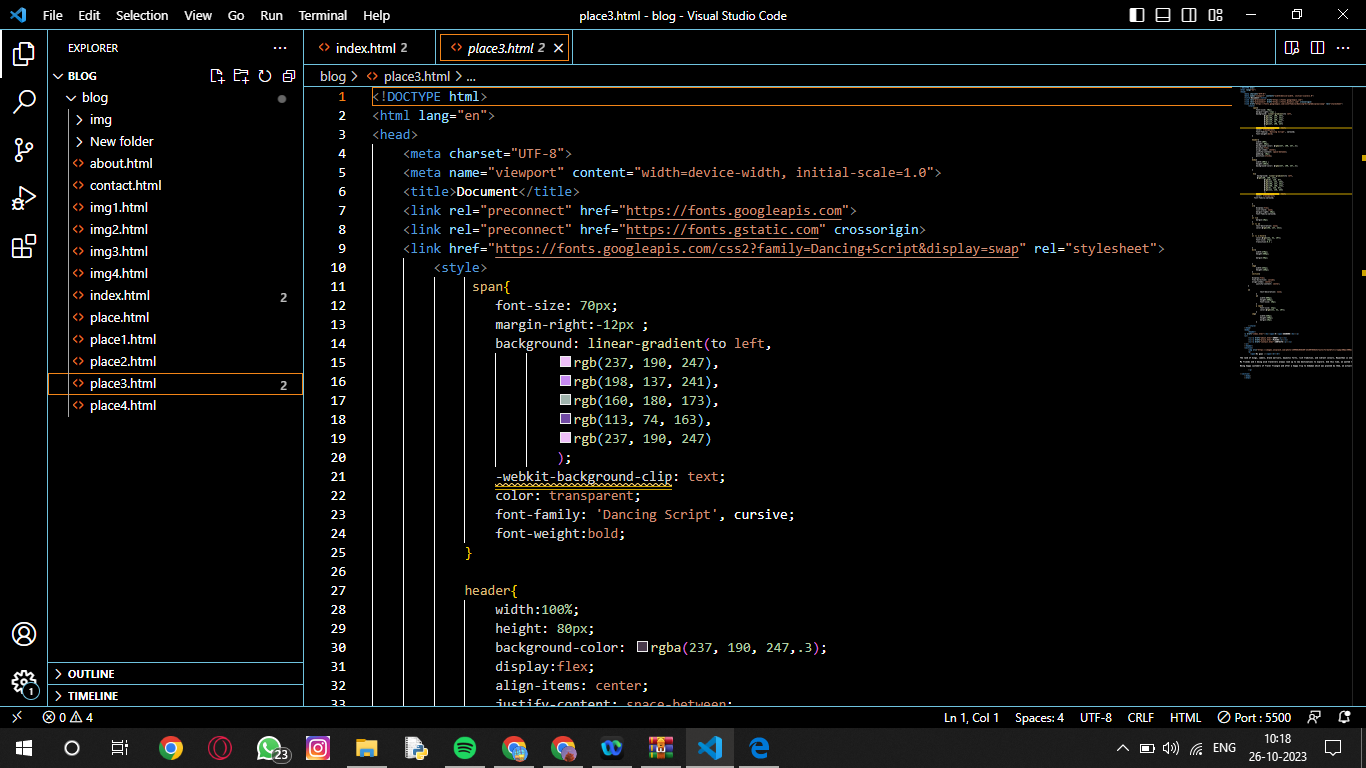
****

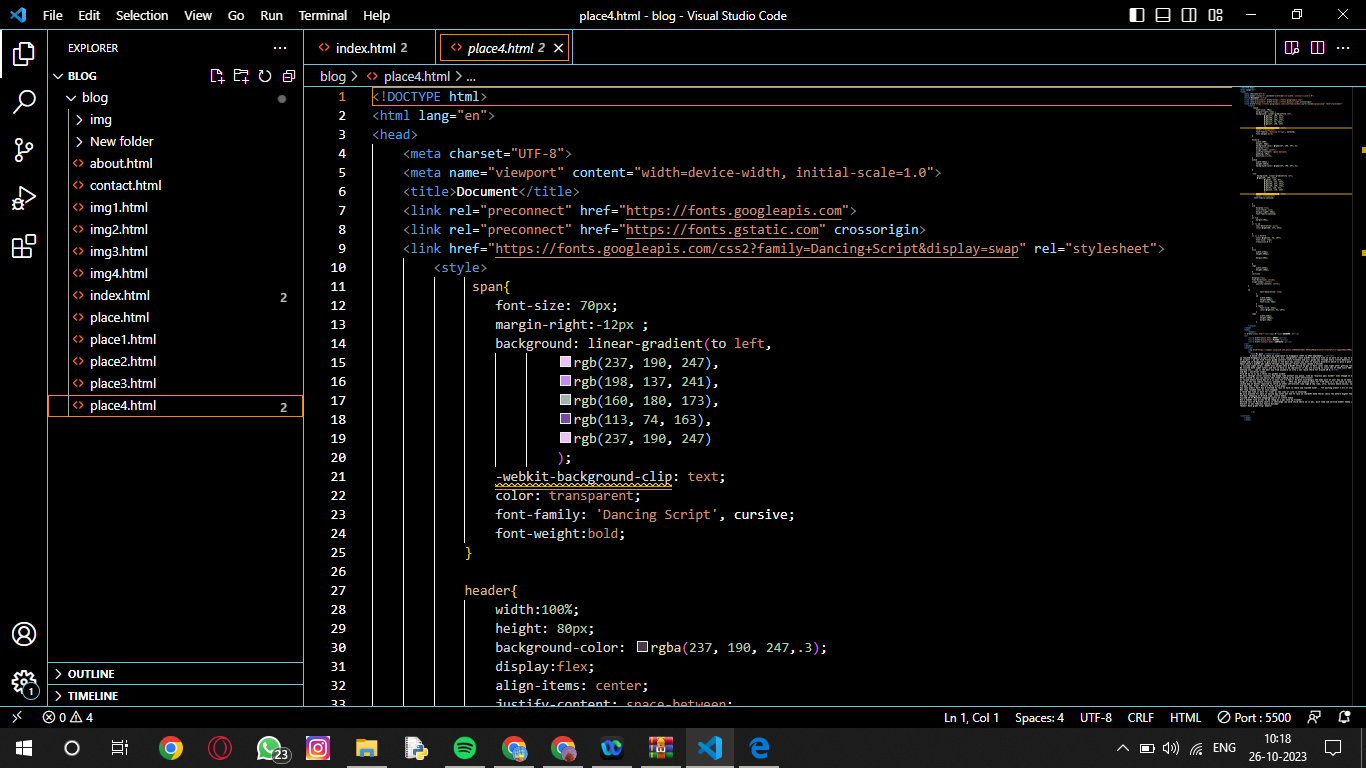
****

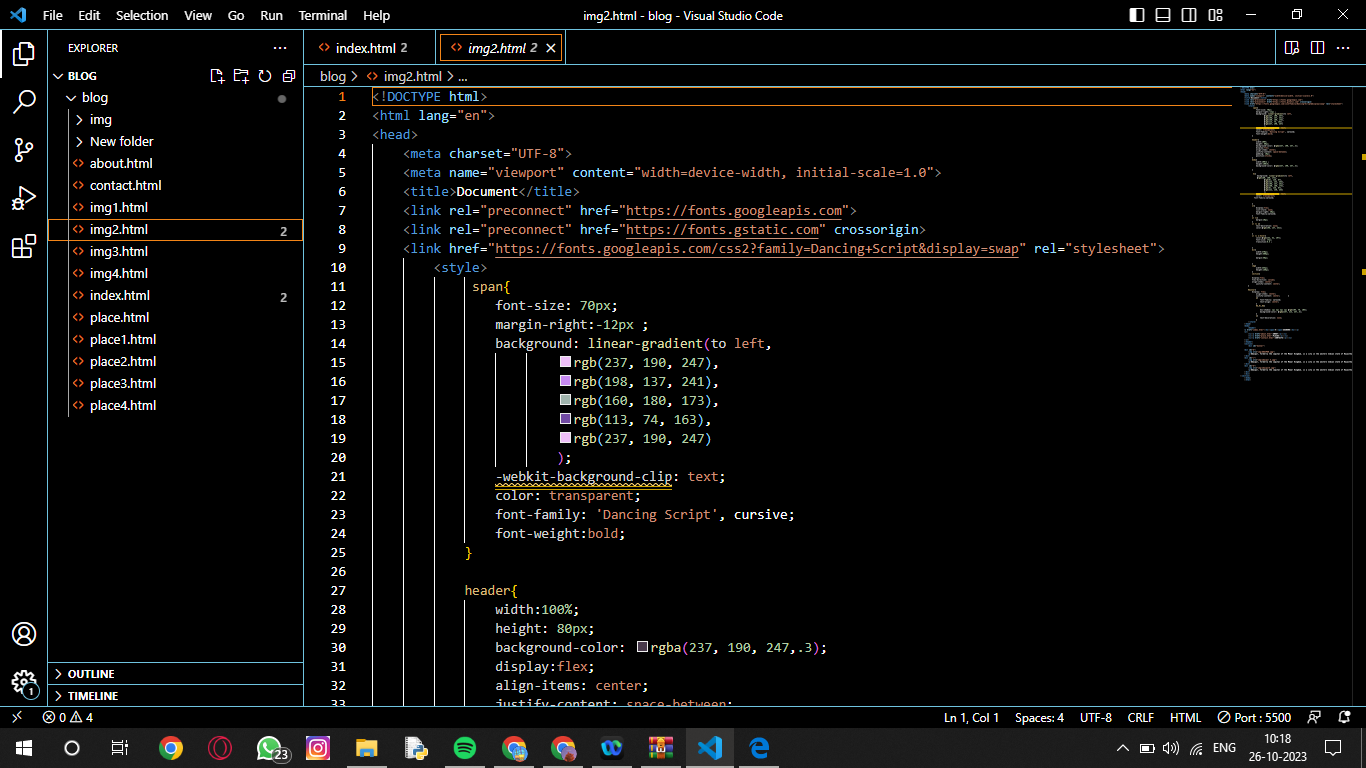
****

****

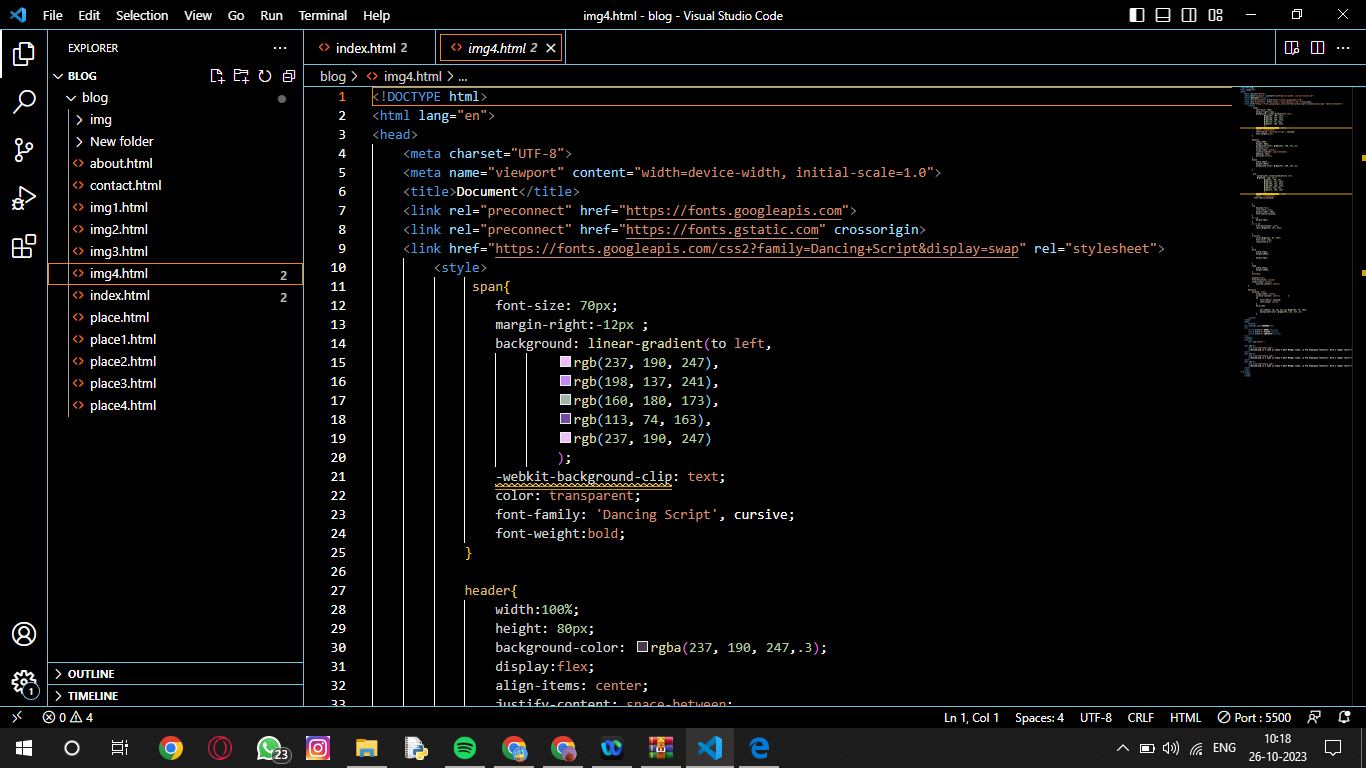
****

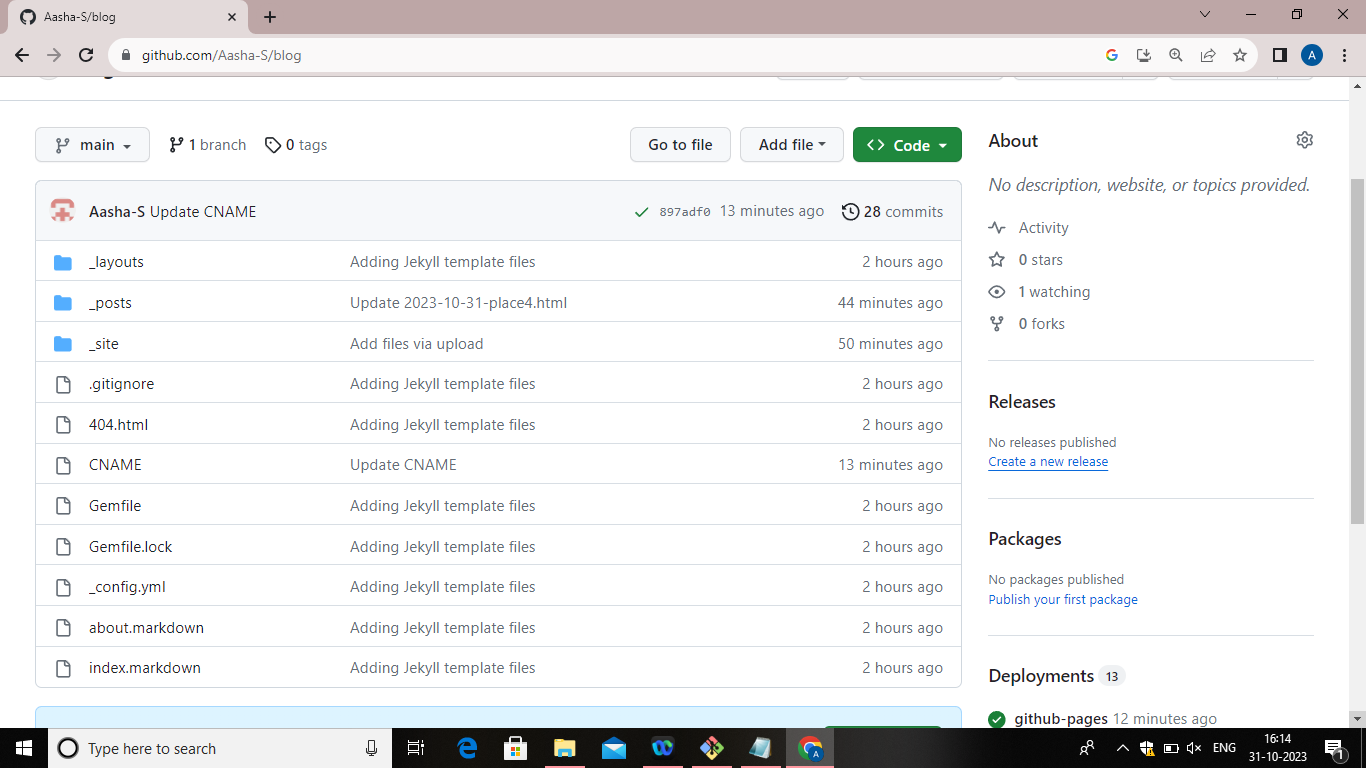
****

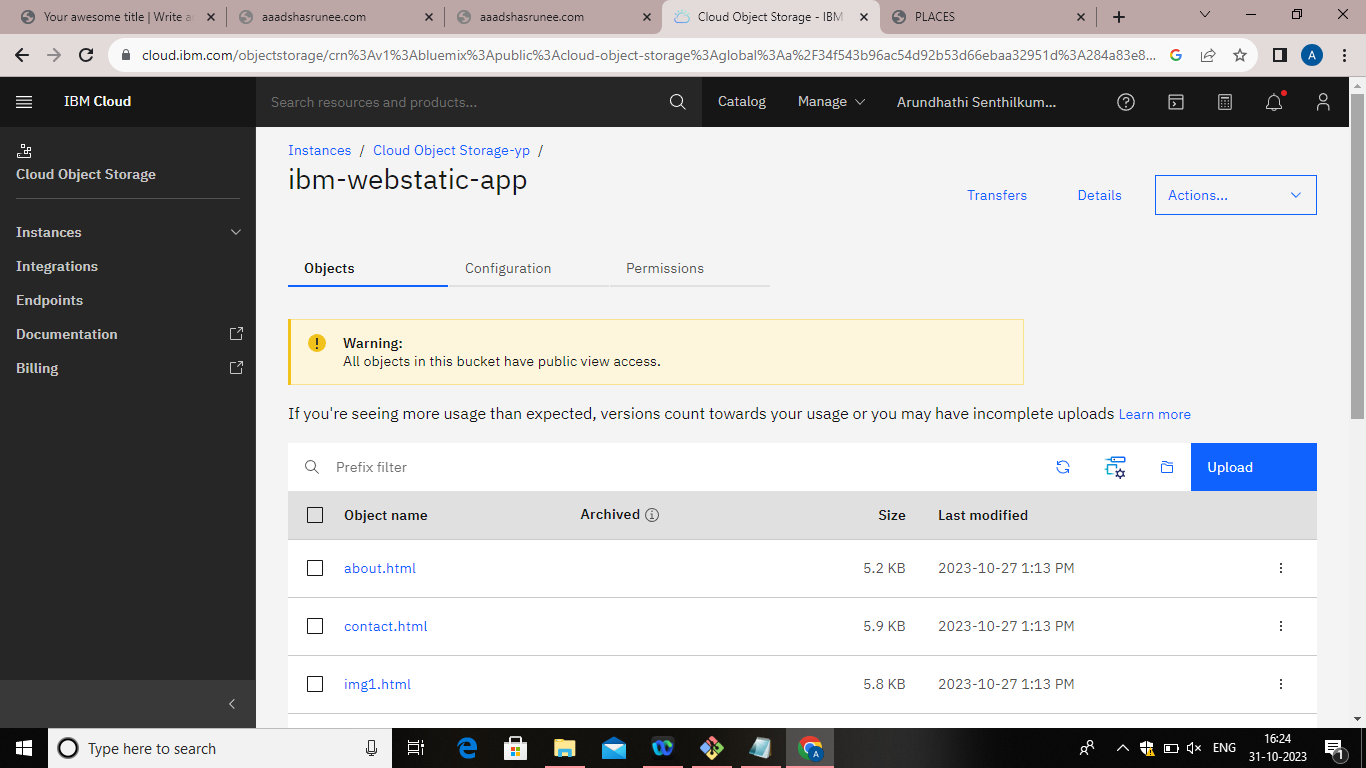
****

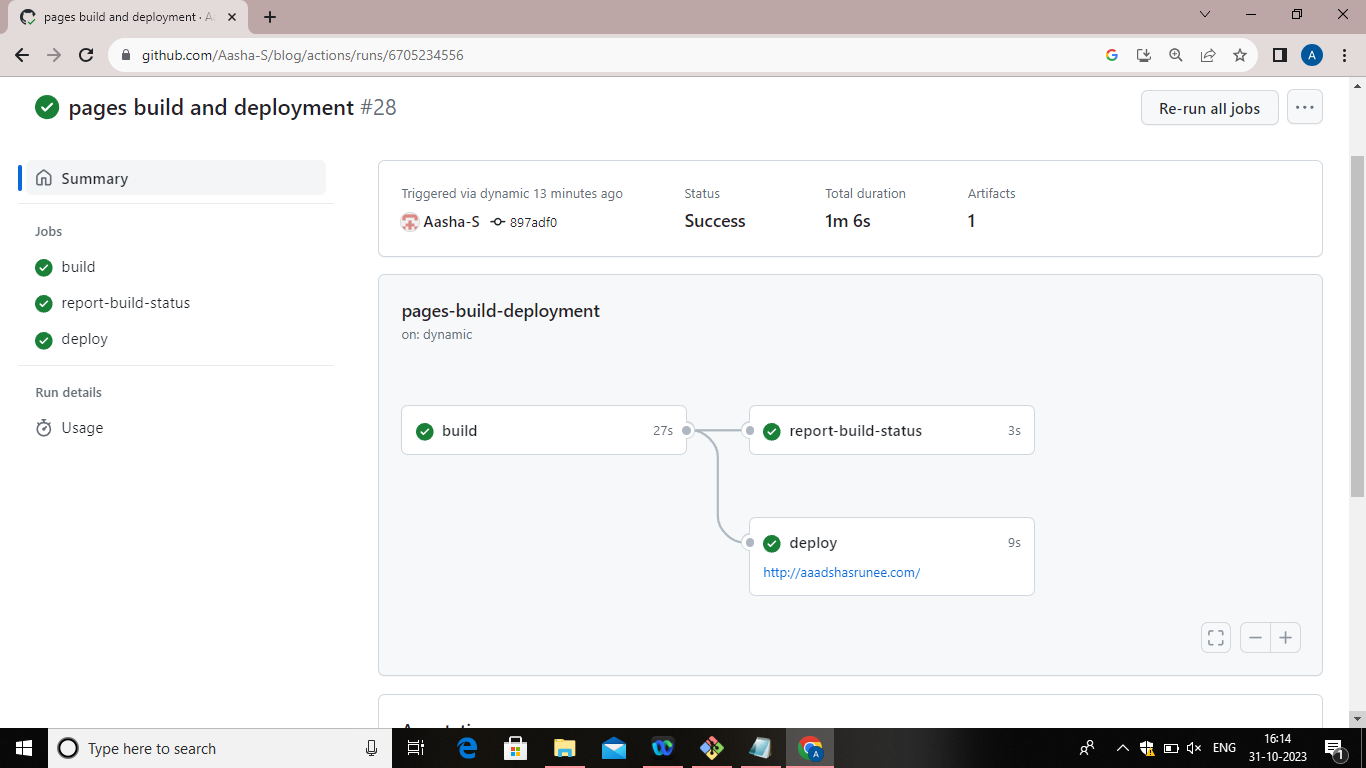
****

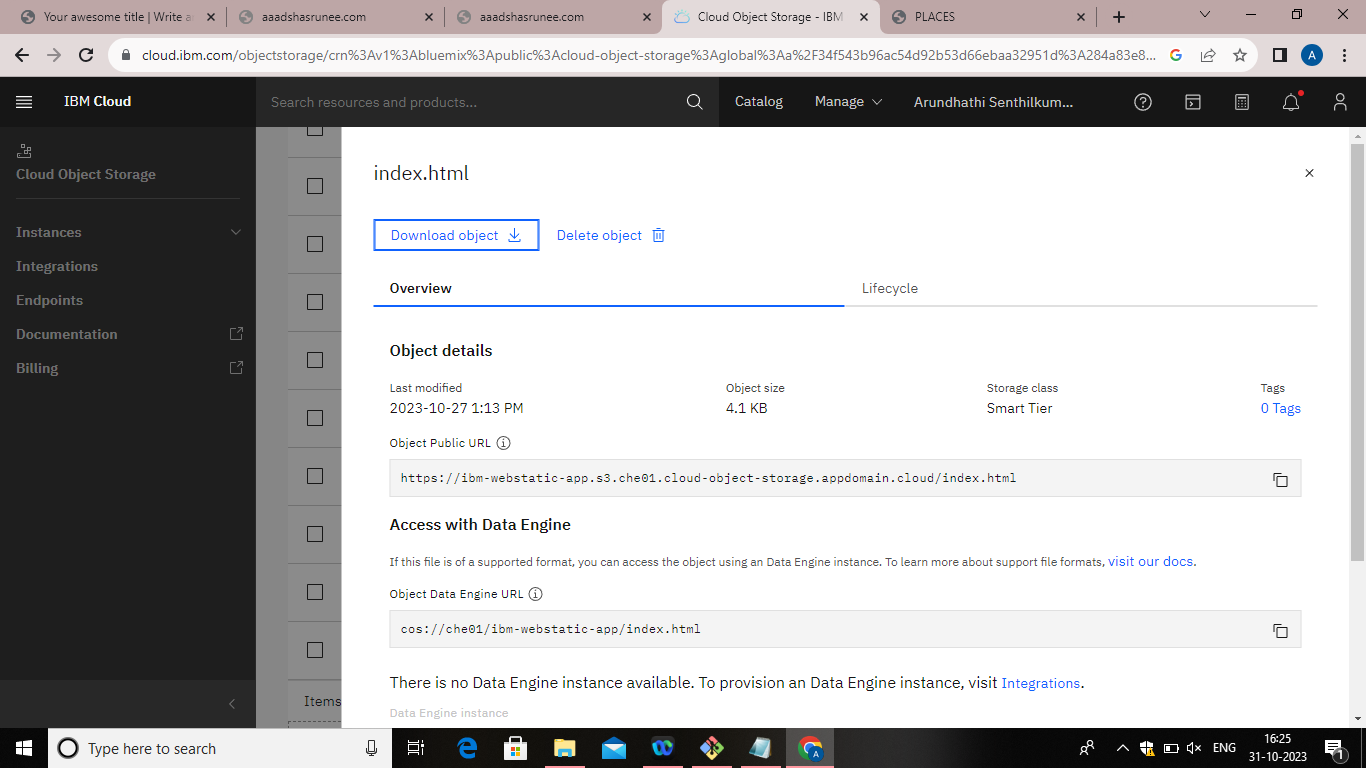
****

****

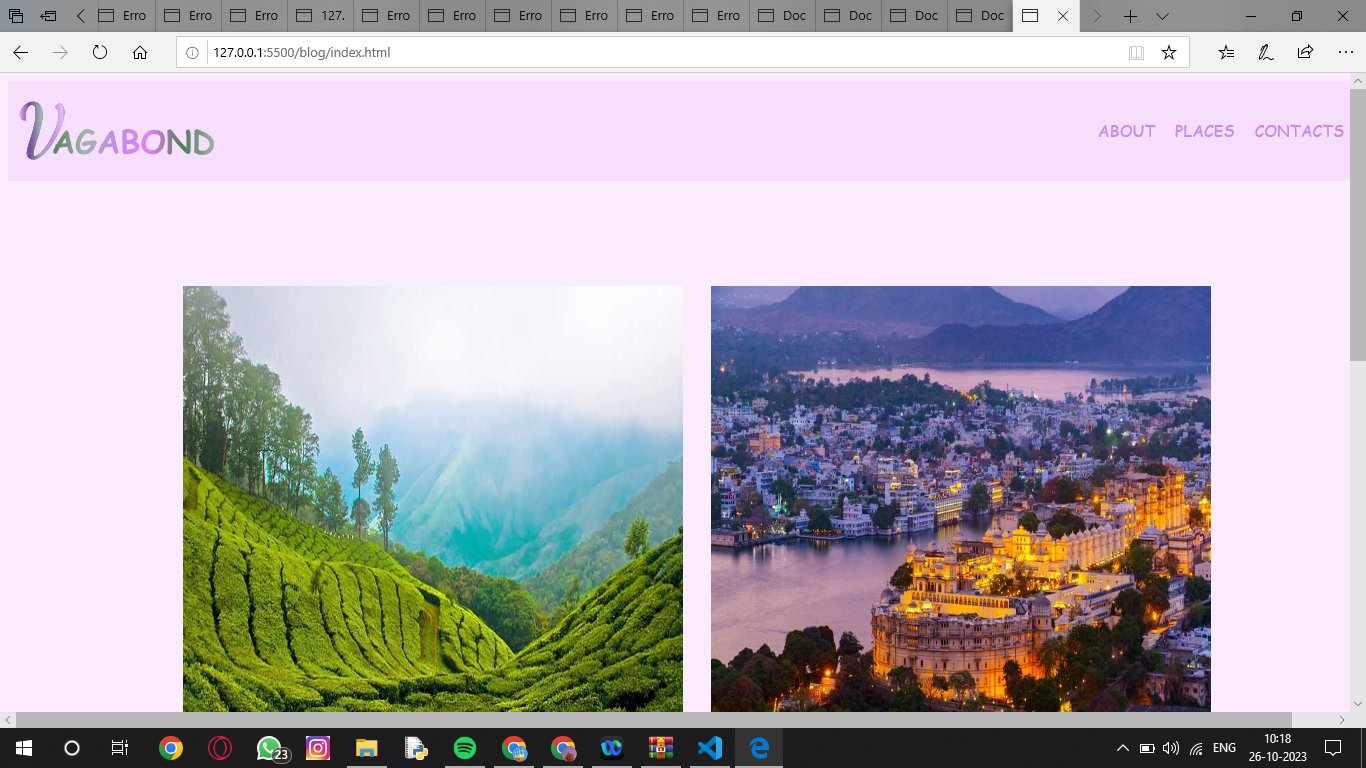
****

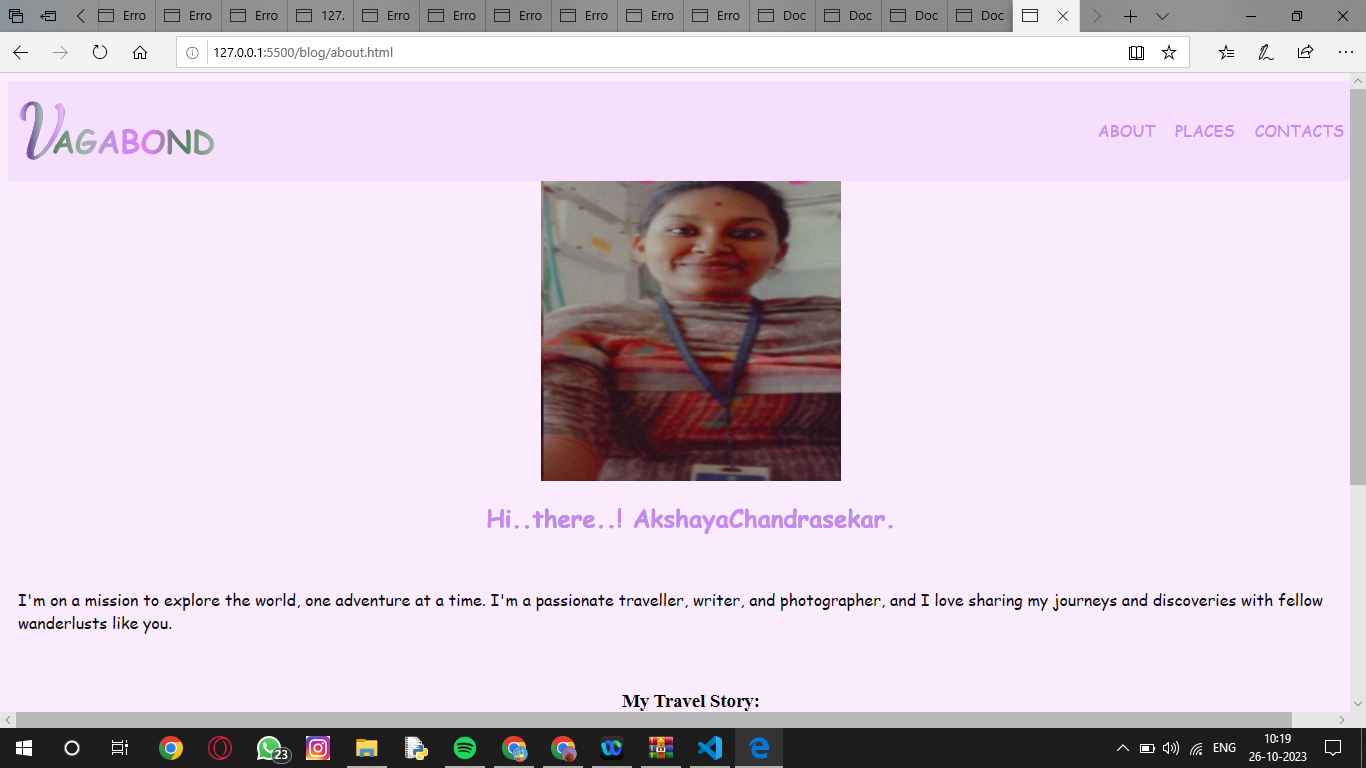
****

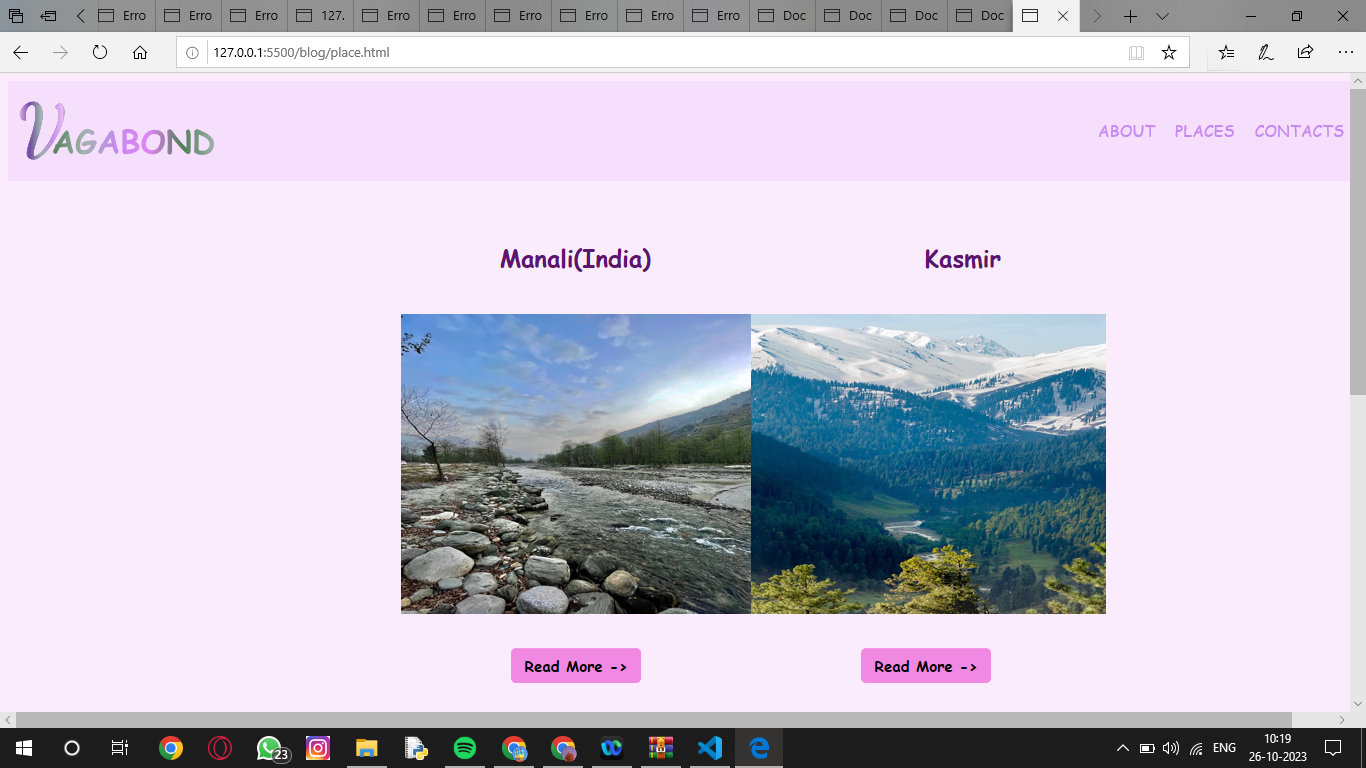
****

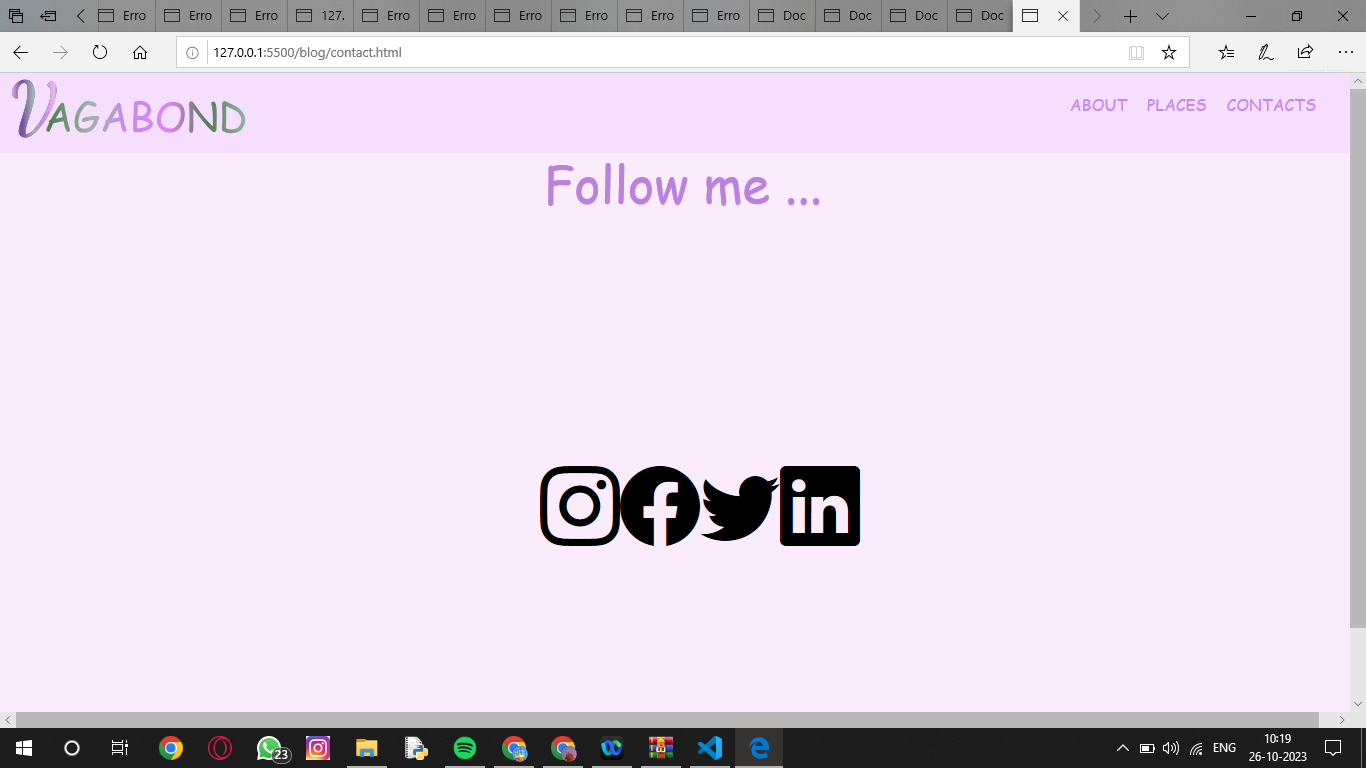
****

**6. OUTPUT OF THE CODE:**

****

****

****

****

**7. CONCLUSION:**

In this document we have shown all the features that have used to build our personal static web blog. We have implemented all the necessities including the platforms that we have used to create our app. Our web blog will enhance all the features that we have described.