Texas College of Management & IT

Bachelor of Information Technology (BIT)

Submitted By:

Name:

LCID:

Academic Year: Second Year / Third Semester

Submitted To:

Instructor Name: Prakash Koirala

Course: Web Technology

Date of Submission: 04/Aug/2025

Content

Task 1: Frontend (UI UX and Content)

1.) **Explain 3 tire architecture with diagram. What are the different technologies we used in the web development process?**

Answer: The **three-tier architecture** is a widely adopted software design pattern used in web development that separates applications into three logical and physical layers:

1. **Presentation Tier (Client Layer)**
2. **Application Tier (Logic Layer or Middle Layer)**
3. **Data Tier (Database Layer)**

Each tier is developed and maintained independently, promoting scalability, manageability, and security.

1. Presentation Tier (Client Layer)

This is the topmost layer that interacts directly with the user through a graphical interface. It includes web pages, forms, and other UI elements displayed in the browser.

- **Responsibilities**: Input handling, data display, user interaction.
- **Technologies Used**:
  - HTML, CSS, JavaScript
  - Front-end frameworks: React, Angular, Vue.js
  - Bootstrap for responsive design

2. Application Tier (Business Logic Layer)

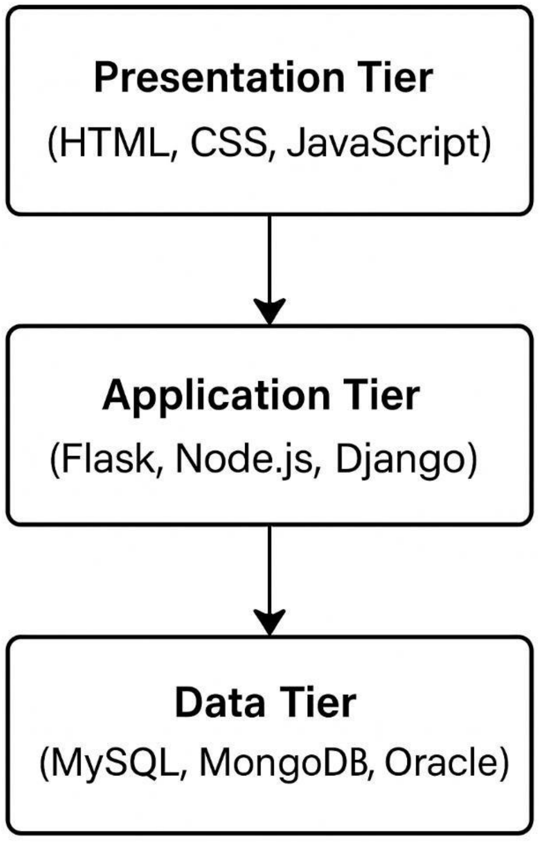
This layer processes user input, makes logical decisions, and controls the application's core functionality. It communicates between the presentation and data layers.

- **Responsibilities**: Business logic, server-side scripting, data processing.
- **Technologies Used**:
  - Programming Languages: Python (Flask, Django), Node.js, PHP, Java, .NET
  - Web Servers: Apache, Nginx
  - APIs and Middleware

3. Data Tier (Database Layer)

This is the bottom layer where all application data is stored, retrieved, and managed. It is responsible for data persistence and handling structured queries.

- **Responsibilities**: Storing, retrieving, updating, and deleting data securely.
- **Technologies Used**:
    - Relational Databases: MySQL, PostgreSQL, Oracle
    - NoSQL Databases: MongoDB, Firebase
    - ORM Tools: SQLAlchemy, Mongoose, Hibernate



2.) **Difference between UI and UX Design. Write principe of good Interface Design.**

Answer:

Aspect	**UI (User Interface) Design**	**UX (User Experience) Design**
Definition	UI design focuses on the **visual layout** of elements in a product interface.	UX design emphasizes the **overall experience** a user has with a product.
Focus Area	Concerned with color schemes, buttons, icons, typography, spacing, and visual consistency.	Deals with user flow, usability, interaction design, and task completion.

Primary Goal	To create an aesthetically pleasing and intuitive interface.	To make the product easy, efficient, and satisfying to use.
Tools Used	Adobe XD, Figma, Sketch, InVision	Wireframing tools, analytics tools, usability testing platforms
Process	Includes screen layouts, graphical elements, and responsiveness.	Involves user research, persona mapping, prototyping, and testing.
Outcome	Produces the **look** and **feel** of the interface.	Produces the **structure** and **functionality** of the experience.

❖ Principles of Good Interface Design

A good user interface should be intuitive, efficient, and user-centered. The following are key principles:

1. **Clarity**
   - Every element on the interface must have a clear meaning and purpose. Avoid ambiguity in buttons, labels, and icons.
2. **Consistency**
   - Visual elements and behaviors should remain consistent across pages. This includes fonts, colors, layouts, and interaction styles.
3. **Feedback**
   - Users should receive immediate and clear responses after interactions, such as button clicks, form submissions, or errors.
4. **Simplicity**
   - Keep the interface minimal. Avoid unnecessary elements that do not support the task the user is performing.
5. **User Control**
   - Allow users to undo actions, cancel processes, and navigate freely. Empower them with control over the system.
6. **Accessibility**
   - The design should be usable by people of all abilities. Use readable fonts, proper contrast, and keyboard navigability.
7. **Visual Hierarchy**
   - Arrange elements to reflect their importance. Use size, color, and positioning to guide user attention.
8. **Affordance and Signifiers**
   - Design elements should suggest their function. For example, buttons should look clickable, and input fields should look fillable.
9. **Mobile Responsiveness**

   - A good interface adapts to different screen sizes without losing usability or visual appeal.

10. **Error Prevention and Handling**
    - Prevent errors through constraints, validations, and confirmations. When errors occur, show meaningful messages and recovery options.

3.) **How do you add an event listener to an HTML element using JavaScript? How do you manipulate the DOM (Document Object Model) using JavaScript?**

Answer: In JavaScript, an **event listener** is used to execute code in response to user interactions such as clicks, mouse movements, key presses, or form submissions.

Example:

```
<button id="myButton">Click Me</button>

<script>
  document.getElementById("myButton").addEventListener("click", function() {
    alert("Button was clicked!");
  });
</script>
```

The DOM represents the structure of a web page. JavaScript allows developers to access, modify, and control HTML elements and their attributes using DOM manipulation.

1. Accessing Elements

- By ID:

```
let element = document.getElementById("myDiv");
```

- By Class Name:

```
let items = document.getElementsByClassName("item");
```

- By Tag Name:

```
let paragraphs = document.getElementsByTagName("p");
```

- Using Query Selectors:

```
let firstItem = document.querySelector(".item");
let allItems = document.querySelectorAll(".item");
```

2. Modifying Content

- Change Text:

```
document.getElementById("title").textContent = "New Title";
```

- Change HTML:

```
document.getElementById("info").innerHTML = "<strong>Updated content</strong>";
```

- Changing Style:

```
document.getElementById("box").style.backgroundColor = "lightblue";
document.getElementById("box").style.fontSize = "20px";
```

- Creating and Appending Elements:

```
let newPara = document.createElement("p");
newPara.textContent = "This is a new paragraph.";
document.body.appendChild(newPara);
```

- Removing Elements:

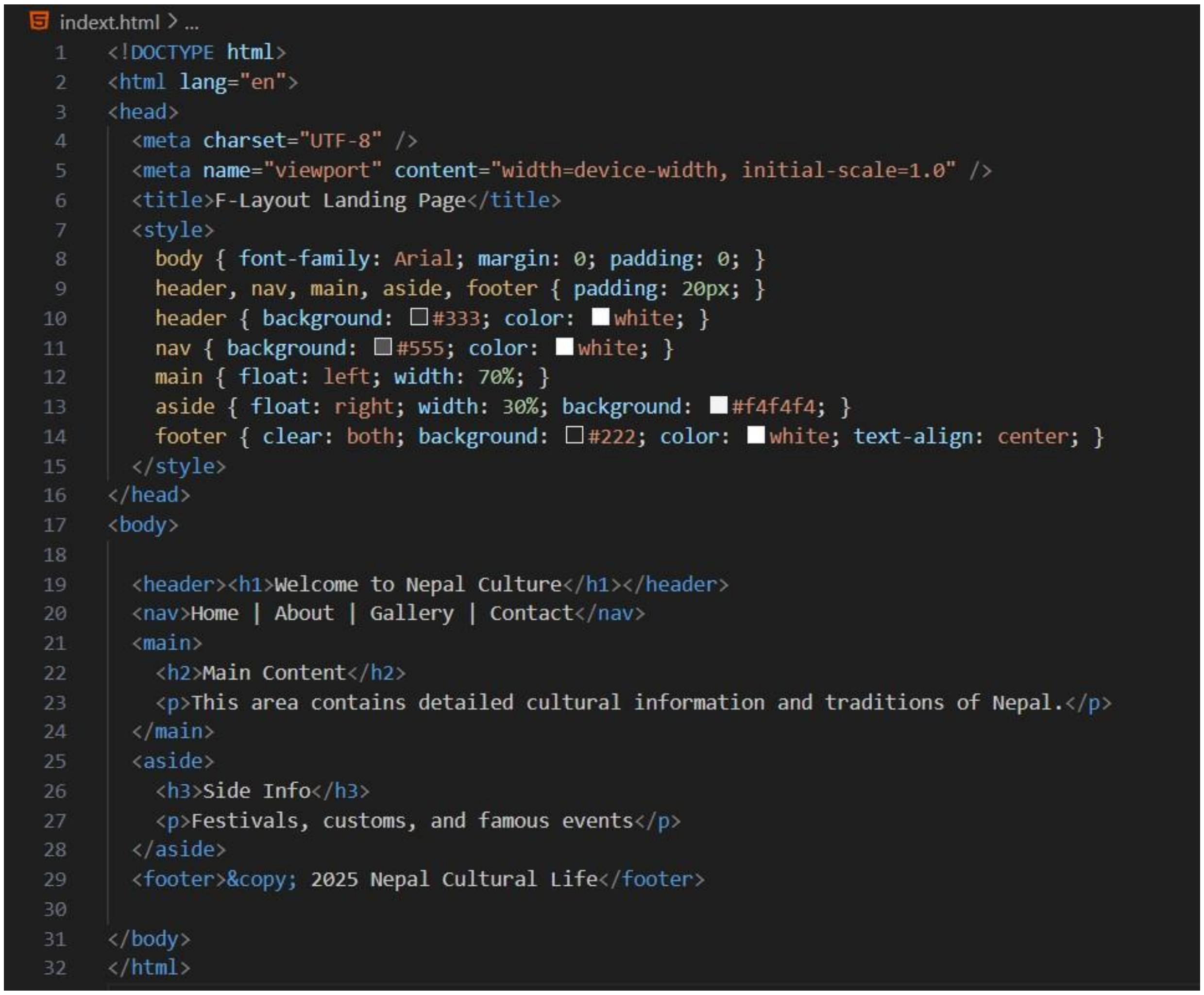
```
let elementToRemove = document.getElementById("oldDiv");
elementToRemove.remove();
```

4.) **Create Landing page using F-Layout, Z-Layout and Featured Image Layout.**

Answer:

1. F-Layout Landing Page

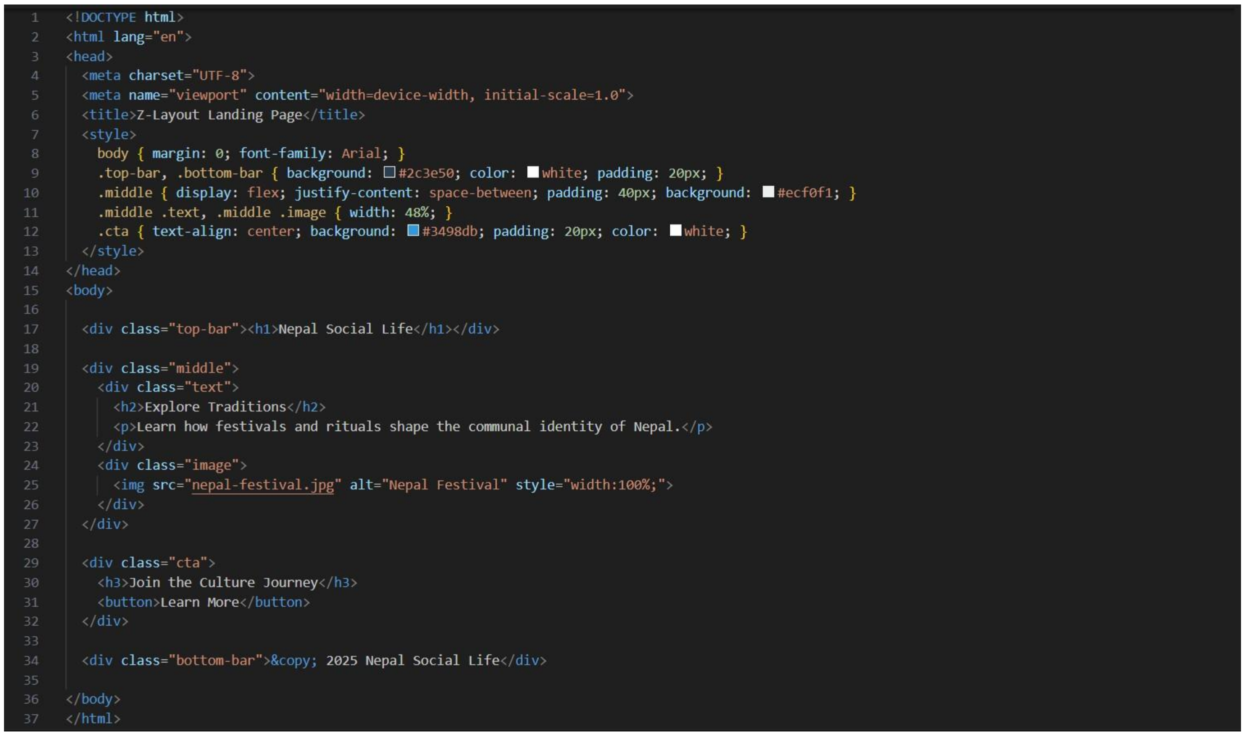
F-Layout mimics the natural reading pattern (left to right, top to bottom) and places key content along the "F" shape.



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>F-Layout Landing Page</title>
  <style>
    body { font-family: Arial; margin: 0; padding: 0; }
    header, nav, main, aside, footer { padding: 20px; }
    header { background: #333; color: white; }
    nav { background: #555; color: white; }
    main { float: left; width: 70%; }
    aside { float: right; width: 30%; background: #f4f4f4; }
    footer { clear: both; background: #222; color: white; text-align: center; }
  </style>
</head>
<body>

  <header><h1>Welcome to Nepal Culture</h1></header>
  <nav>Home | About | Gallery | Contact</nav>
  <main>
    <h2>Main Content</h2>
    <p>This area contains detailed cultural information and traditions of Nepal.</p>
  </main>
  <aside>
    <h3>Side Info</h3>
    <p>Festivals, customs, and famous events</p>
  </aside>
  <footer>&copy; 2025 Nepal Cultural Life</footer>

</body>
</html>
```

2. Z-Layout Landing Page

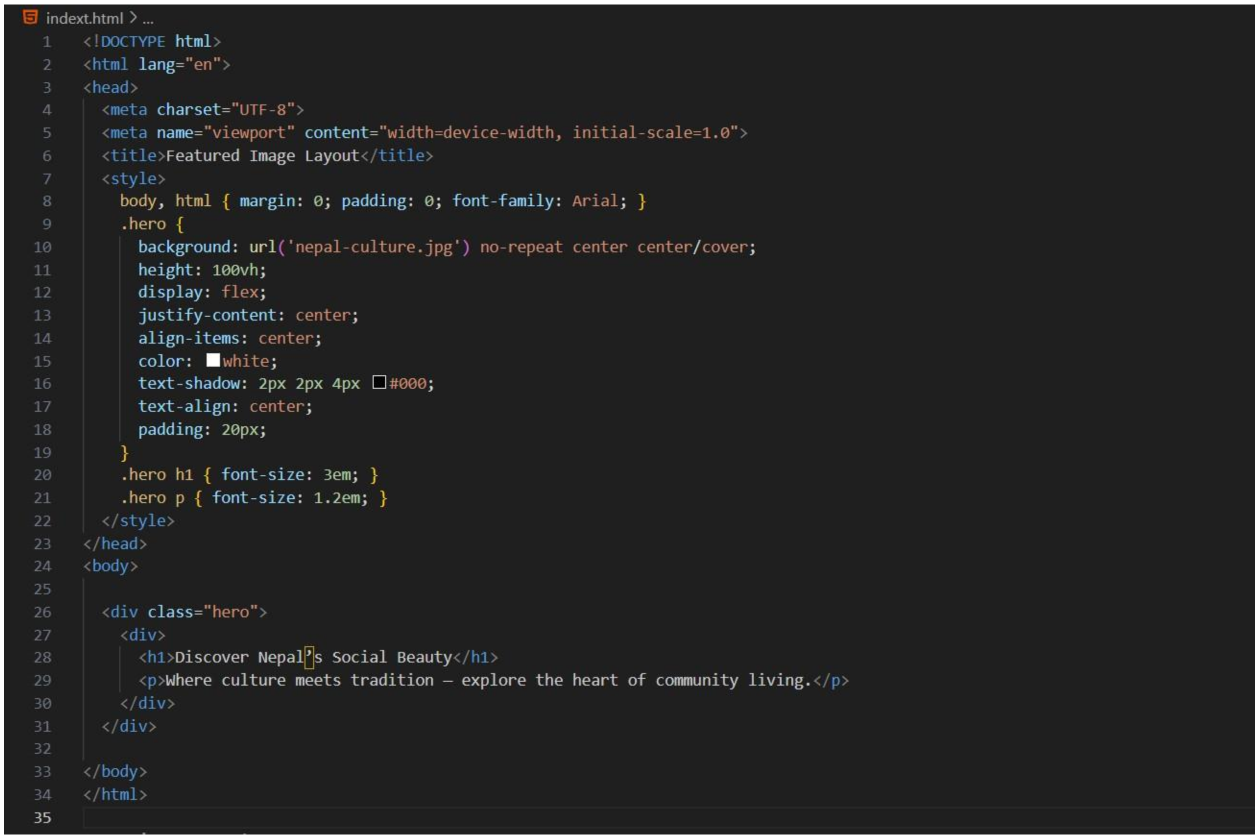
Z-Layout follows a diagonal path, ideal for engaging visual flow and CTAs (Calls to Action).



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Z-Layout Landing Page</title>
  <style>
    body { margin: 0; font-family: Arial; }
    .top-bar, .bottom-bar { background: #2c3e50; color: white; padding: 20px; }
    .middle { display: flex; justify-content: space-between; padding: 40px; background: #ecf0f1; }
    .middle .text, .middle .image { width: 48%; }
    .cta { text-align: center; background: #3498db; padding: 20px; color: white; }
  </style>
</head>
<body>

  <div class="top-bar"><h1>Nepal Social Life</h1></div>

  <div class="middle">
    <div class="text">
      <h2>Explore Traditions</h2>
      <p>Learn how festivals and rituals shape the communal identity of Nepal.</p>
    </div>
    <div class="image">
      <img src="nepal-festival.jpg" alt="Nepal Festival" style="width:100%;">
    </div>
  </div>

  <div class="cta">
    <h3>Join the Culture Journey</h3>
    <button>Learn More</button>
  </div>

  <div class="bottom-bar">&copy; 2025 Nepal Social Life</div>

</body>
</html>
```

3. Featured Image Layout Landing Page

This layout uses a **hero image** with prominent text and call-to-action over the image.



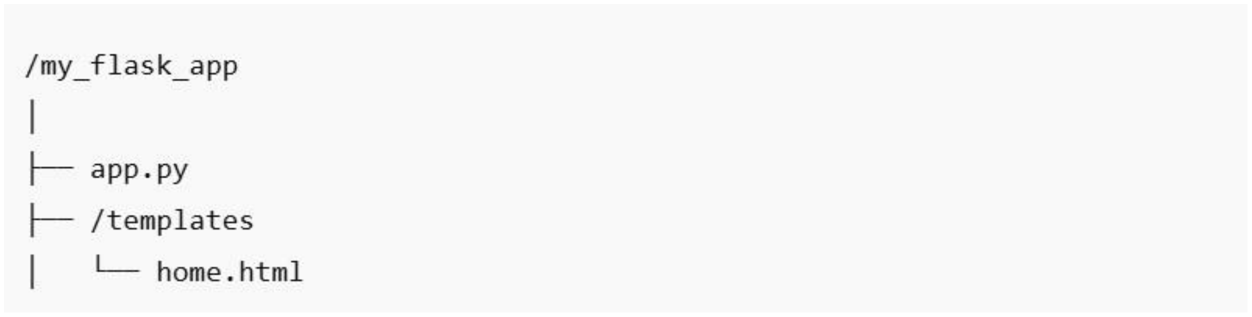
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Featured Image Layout</title>
  <style>
    body, html { margin: 0; padding: 0; font-family: Arial; }
    .hero {
      background: url('nepal-culture.jpg') no-repeat center center/cover;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      color: white;
      text-shadow: 2px 2px 4px #000;
      text-align: center;
      padding: 20px;
    }
    .hero h1 { font-size: 3em; }
    .hero p { font-size: 1.2em; }
  </style>
</head>
<body>

  <div class="hero">
    <div>
      <h1>Discover Nepal's Social Beauty</h1>
      <p>Where culture meets tradition – explore the heart of community living.</p>
    </div>
  </div>

</body>
</html>
```

5.) **How do you create and render dynamic content in a Flask template using Python Flask and HTML?**

Answer: Flask is a lightweight web framework in Python that allows developers to build web applications quickly. One of its core strengths is the ability to pass **dynamic data from the Python backend to HTML templates**, which are rendered using **Jinja2** templating syntax.

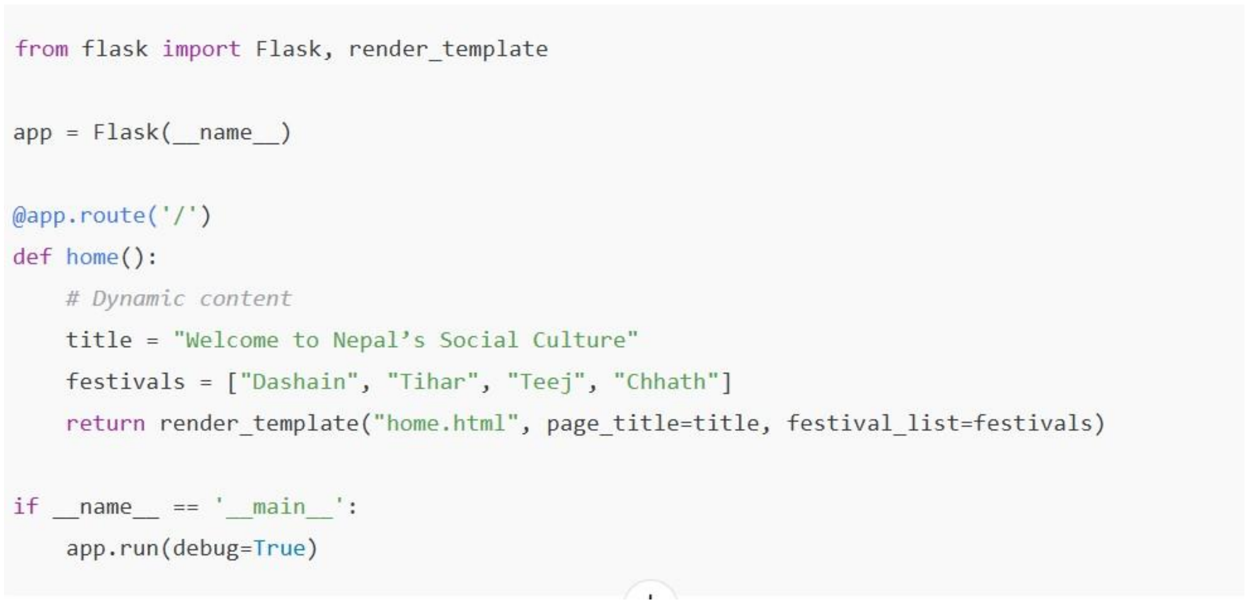
❖ **Step-by-Step Process**

1. Project Structure Example



```
/my_flask_app
|
├── app.py
├── /templates
|   └── home.html
```

2. Creating the Flask Application (app.py)



```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    # Dynamic content
    title = "Welcome to Nepal's Social Culture"
    festivals = ["Dashain", "Tihar", "Teej", "Chhath"]
    return render_template("home.html", page_title=title, festival_list=festivals)

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

3. Creating the HTML Template (templates/home.html)

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>{{ page_title }}</title>
</head>
<body>
  <h1>{{ page_title }}</h1>

  <h2>Major Festivals of Nepal</h2>
  <ul>
    {% for fest in festival_list %}
      <li>{{ fest }}</li>
    {% endfor %}
  </ul>
</body>
</html>
```

6.) **How do you pass data between HTML, CSS, JavaScript, and Python Flask in a web application?**

Answer:

- **HTML → Flask**: Use forms or fetch requests.
- **Flask → HTML**: Use render_template() with variables.
- **JavaScript → Flask**: Use fetch() or AJAX to send data.
- **CSS → HTML**: Via <link rel="stylesheet">, unaffected by Flask directly.

HTML:

```html
<form action="/submit" method="POST">
  <input name="username">
  <button type="submit">Submit</button>
</form>
```

Python:

```python
from flask import request

@app.route('/submit', methods=['POST'])
def submit():
    username = request.form['username']
    return f"Thanks, {username}!"
```

7.) **How do you pass variables or parameters to routes in Flask?**

Answer:

1. Passing Variables via URL Path Parameters

Flask uses angle brackets `< >` in route definitions to capture dynamic segments from the URL.

python

```python
@app.route('/user/<username>')
def greet_user(username):
    return f"Hello, {username}!"
```

2. Passing Parameters via Query Strings

Query strings use the `?key=value` format and are accessed with `request.args`.

python

```python
from flask import request

@app.route('/search')
def search():
    keyword = request.args.get('q')
    return f"Search results for: {keyword}"
```

3. Passing Parameters via Forms (POST Method)

HTML forms send data to routes using the `POST` method.

```python
from flask import request

@app.route('/submit', methods=['POST'])
def submit():
    username = request.form.get('username')
    return f"Submitted by: {username}"
```

4. Passing Data Using JSON (JavaScript → Flask)

Used in API routes or JavaScript fetch requests.

```python
from flask import request, jsonify

@app.route('/api/data', methods=['POST'])
def api_data():
    data = request.get_json()
    return jsonify({"received": data["name"]})
```

8.) How can you interact with a database using Flask? What are some popular database libraries used with Flask?

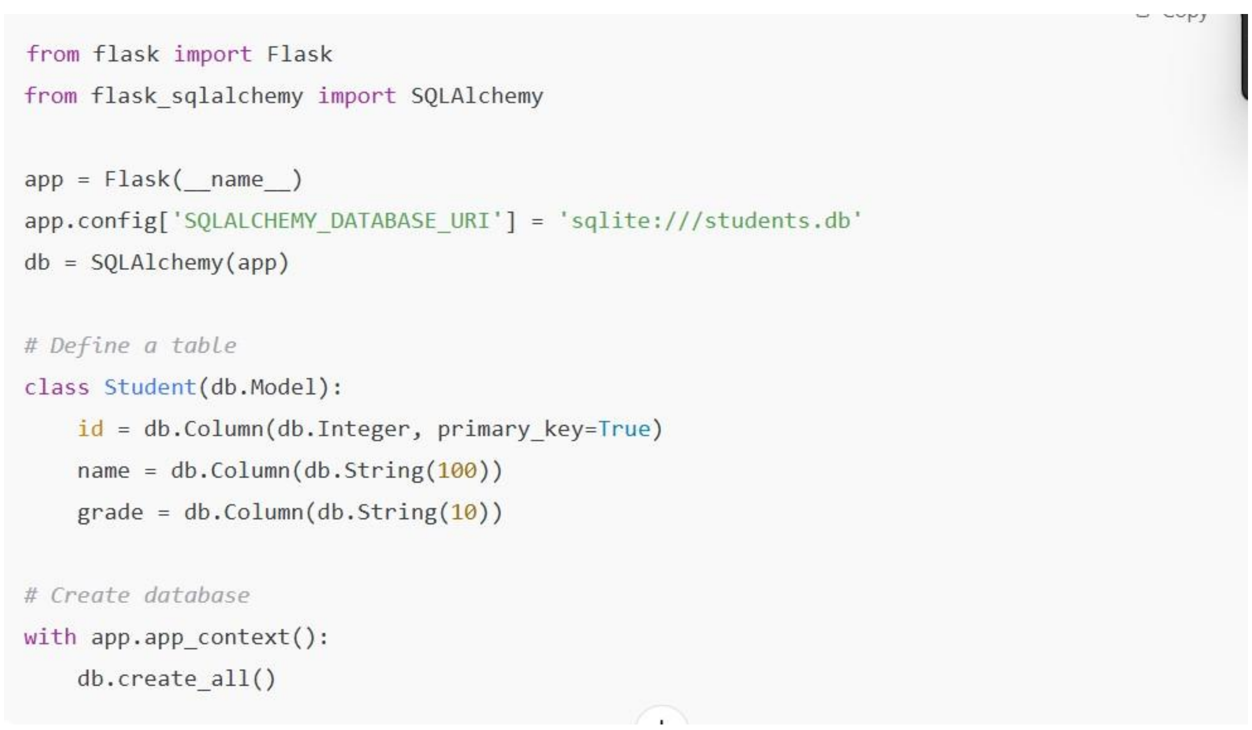
Answer: Flask does **not have built-in database support**, but it integrates easily with external database libraries and ORM tools. These allow seamless communication between the Flask backend and relational or NoSQL databases.

1. Using Flask with SQL Databases (SQLite / MySQL / PostgreSQL)

o Example: Connecting Flask to SQLite using SQLAlchemy

```bash
pip install flask flask_sqlalchemy
```

app.py



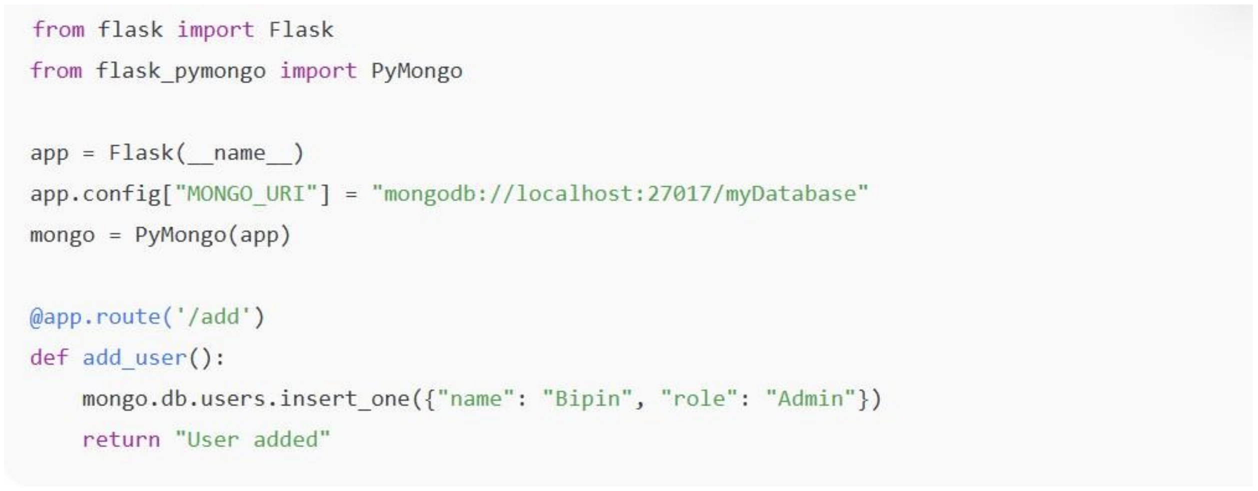
```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.db'
db = SQLAlchemy(app)

# Define a table
class Student(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    grade = db.Column(db.String(10))

# Create database
with app.app_context():
    db.create_all()
```

2. Using Flask with NoSQL Databases (MongoDB)

MongoDB is a popular NoSQL database. You can use **Flask-PyMongo** to connect Flask with MongoDB.

```
pip install flask-pymongo
```

app.py



```python
from flask import Flask
from flask_pymongo import PyMongo

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://localhost:27017/myDatabase"
mongo = PyMongo(app)

@app.route('/add')
def add_user():
    mongo.db.users.insert_one({"name": "Bipin", "role": "Admin"})
    return "User added"
```

❖ **Popular Database Libraries Used with Flask**

Library

- SQLAlchemy
- Flask-SQLAlchemy
- Flask-PyMongo
- Peewee
- Pony ORM
- Firebase Admin SDK

Abstract

Objective:

The mission of this project, developed by **Chitra Bahadur Thapa Chhetri**, is to develop website using HTML, CSS, and JavaScript. The website will include key sections such as Home, Dashain, Tihar, Chhath and Contact for the proper presentation of skills, work experience, and projects. The mission is to create an engaging and user-friendly site that illustrates my skills and accomplishments as a fashion model and content creator.

Scope:

-The project is all about front-end development, with the most critical areas of concern being:

-Responsive Design: Rendering the site adaptable across various devices (desktop, tablet, mobile) employing CSS media queries, Flexbox, and Grid.

- Interactivity: Including interactive elements through JavaScript functionality such as form validation, animations, and dynamic content updating.

- Deployment: Publishing the code on GitHub for version control and collaboration.

1.2 Expected Outcomes:

-A fully functional website with a clean, modern look and easy navigation.

- Responsive Layouts: The site is built to provide a seamless experience across all screen sizes.

-Interactive Features: Form validation, hover effects, and smooth scrolling features driven by JavaScript, making the site interactive and user-friendly.

1.3 Connection to the Project:

- Goal: The website organization (Home, Dashain, Tihar, Chhath, Contact) is to best highlight my projects and abilities.

- Scope: Responsiveness and interactivity are ensured through CSS media queries and JavaScript usage, coupled with a smooth and sleek user experience.

- Expected Outcomes: The completed website will be a professional, responsive, and interactive portfolio, hosted on GitHub, that demonstrates both my technical expertise and adherence to best practices in web development.

Introduction

This project, undertaken by Chitra Bahadur Thapa Chhetri**,** demonstrates the practical application of front- end web technologies such as HTML, CSS, and JavaScript in creating a professional and user- friendly website. It emphasizes the significance of responsiveness and interactivity in modern web development, demonstrating how these technologies can be combined to provide an effortless and interactive user experience. The website is fully responsive, with CSS Flexbox, Grid, and media queries for optimal viewing on desktop, tablet, and mobile devices. JavaScript was utilized to include interactive features like form validation and animation, which raised user interaction. Moreover, the project follows best practices in development, with code hosted on GitHub for version control and collaboration. By incorporating these important concepts, this project meets its technical objectives and is a solid example of well-constructed, modern web development.

Project Objectives

The project objectives are closely associated with its implementation and anticipated outcomes:

1. Create a Portfolio Website with Home, Dashain, Tihar, Chhath and Contact Pages:

The website is structured into four main sections:

- **Home:** Includes an intro header, a projects display image slider, and a brief summary of festivals.

- **Dashain:** Has a personal biography, an area showing skills in the form of icons or progress bars, and a mission statement.

- **Tihar:** Presents education, work history, and technical skills in timeline or grid style.

- **Chhath:**

- **Contact:** Has a contact form for user inquiries and links to social network profiles.

2. Make the Site Responsive Across Devices with HTML, CSS, and Media Queries:

The site is fully responsive to make it easy to switch between desktops, tablets, and mobile devices.

- CSS Flexbox and Grid were employed for layout design, while media queries enable scaling and aligning properly on smaller screens.

- **Example:** On smaller screens, the navigation bar collapses into a hamburger menu, and the Resume page switches from timeline to single-column format.

3. Add Interactivity (e.g., Form Validation, Animations) using JavaScript:

- JavaScript was utilized to improve the user experience:

- **Form Validation:** Ensures the Contact form is filled in correctly (e.g., proper email format, non-empty fields).

- **Animations:** Adds hover animations on buttons, navigation smooth scrolling, and an image slider on the Home page.

4. Publish Code on GitHub:

- The project code is hosted on GitHub, demonstrating expertise in version control.

Connection to the Project:

- Design: The website layout and structure are consistent with the goal of creating a professional portfolio.

- Responsiveness: Responsive layouts and media queries ensure the site offers a seamless experience on any device.

- Interactivity: JavaScript features like form validation and animation enhance usability and user engagement.

Literature Review

Review of Relevant Sources:

1. Responsive Web Design:

- These include the application of CSS media queries, Flexbox, and Grid to test methods for building responsive layouts. These are tools that allow websites to respond graciously to varying screen sizes, and this renders their experience standardized and accessible on all devices.

- Source: The MDN Web Docs for Responsive Design were consulted to obtain step-by-step guidance on how responsive design practices, including fluid grids, flexible images, and media queries, can be followed.

2. JavaScript for Interactivity:

- How JavaScript had been applied to give dynamic qualities to web pages was examined. This included the utilization of features such as form validation, animation, and interactive elements to enhance user interaction and render the overall experience more usable.

- Source: JavaScript.info offered detailed lessons in JavaScript fundamentals, such as DOM manipulation, event handling, and form validation, that were needed to add interactivity to the project.

3. Deployment with GitHub:

- How to deploy static sites and link them to GitHub repositories was learned.

Project Objectives

Specific Objectives:

1. Create a Website for festivals with Pages: Home, Dashain, Tihar, Chhath and Contact:

The main objective is to create a well-structured portfolio website effectively showcasing skills, experience, and projects. The website will consist of four primary pages:

- **Home:** A welcoming home page with some highlights of key projects.

- **Dashain:** A page that provides personal background, abilities, and interests.

- **Tihar:** A professional overview of education, work experience, and technical skills.

- **Contact:** A page with a form for user inquiries and links to social media profiles.

2. Implement Responsive Design Using HTML and CSS:

-The website will be fully responsive, i.e., it will adapt smoothly between different screen sizes, like desktops, tablets, and smartphones.

- CSS Flexbox, Grid, and media queries will be used to achieve flexible, responsive layouts.

3. Add Interactivity through JavaScript:

- JavaScript will be used to add user interaction and user engagement. Key features will include:

- Form Validation: Checking that the Contact form is correctly filled in (e.g., valid email address, fields not empty).

- Animations: Adding hover effects, smooth scrolling, and dynamic content updates.

4. Publish the Code on GitHub:

- The code for the project will be posted to GitHub to demonstrate version control practices and facilitate collaboration.

Methodology

The methodology is the part which explains step-by-step methodology adopted while developing and designing the personal portfolio website. The project was executed in a systematic manner, beginning with the analysis of requirements and going through the implementation and design phases.

1. Requirement Analysis:

- The project was initiated by the identification of the tools and technologies needed.

- Front-End Technologies: HTML, CSS, and JavaScript were utilized to structure, style, and add interactivity to the website.

2. Design Phase:

- A navigation bar was created with navigation links to the main pages of the website: Home, Dashain, Tihar, Chhath, and Contact.

- A responsive design was achieved using CSS media queries, Flexbox, and Grid to enable the website to transition smoothly across different devices (desktop, tablet, mobile).

3. Implementation Phase:

- **Home Page:** Designed with a header, an image slider to display projects, and a footer.

- **Dashain Page:** Included a personal introduction, hobbies, and a mission statement.

- **Tihar Page:** Presented with a professional summary, education, work experience, and qualifications in a clean layout.

- **Chhath Page:**

- **Contact Page:** Accomplished with a user query form and social media profile links.

- **Interactivity:** JavaScript was utilized to add dynamic features like:

- **Form Validation:** Checks that the Contact form is filled in correctly (e.g., correct email address format, fields are not blank).

- **Dynamic Content Updates:** Improves user experience using animations and fluid transitions.

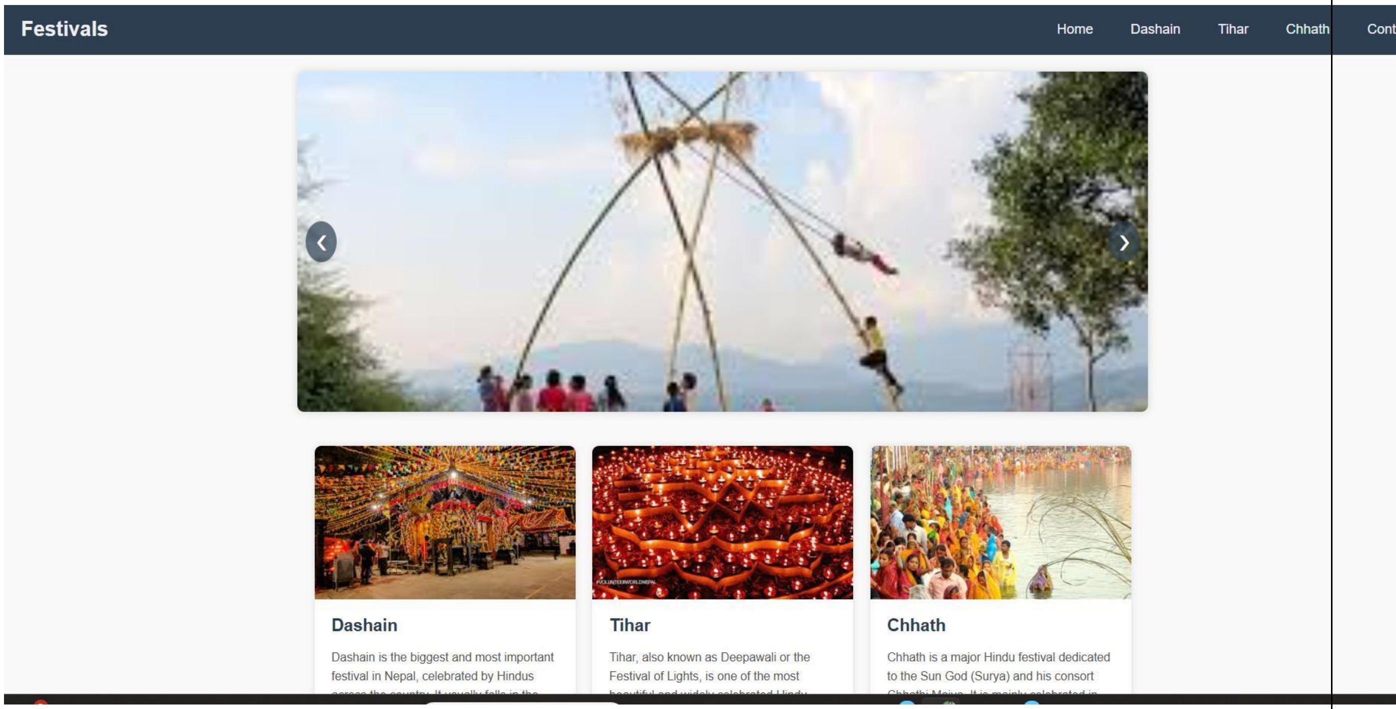
4. Deployment:

- The code was hosted on GitHub for collaboration and version control.

Results and Analysis:

The analysis and results section examines the project outputs. The website is fully operational, with a responsive design that responds favorably to varying devices and browsers. Interactivity is enhanced using JavaScript through functionalities such as form validation and refreshing of dynamic content. The code is uploaded to GitHub, hence making the website universally accessible to visitors. The site was cross-tested on different browsers (Chrome, Firefox, Edge) and devices (desktop, tablet, mobile) to test for responsiveness as well as cross-browser compatibility. JavaScript performance was tested and was found to be effective in adding interactivity and hence improving overall user experience

Home Page:



Dashain

Dashain is the biggest and most important festival in Nepal, celebrated by Hindus

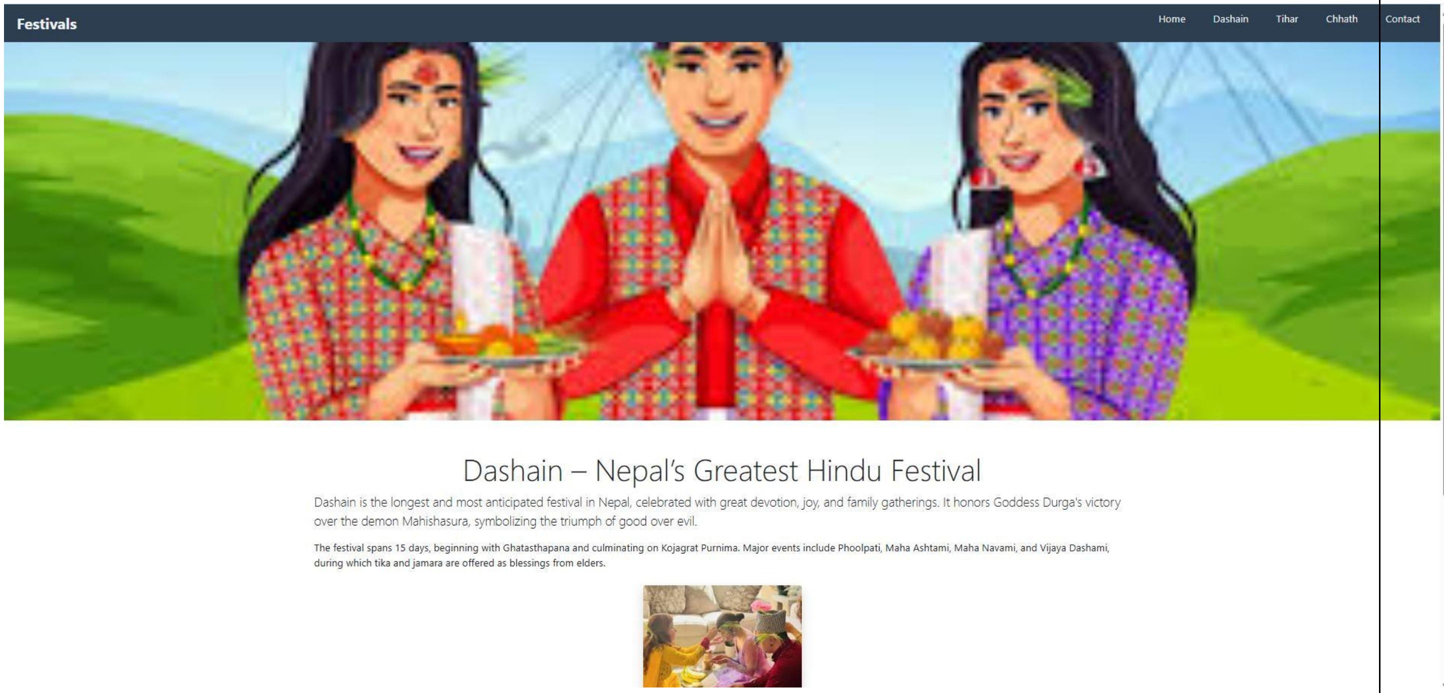
Tihar

Tihar, also known as Deepawali or the Festival of Lights, is one of the most

Chhath

Chhath is a major Hindu festival dedicated to the Sun God (Surya) and his consort

Dashain Page:

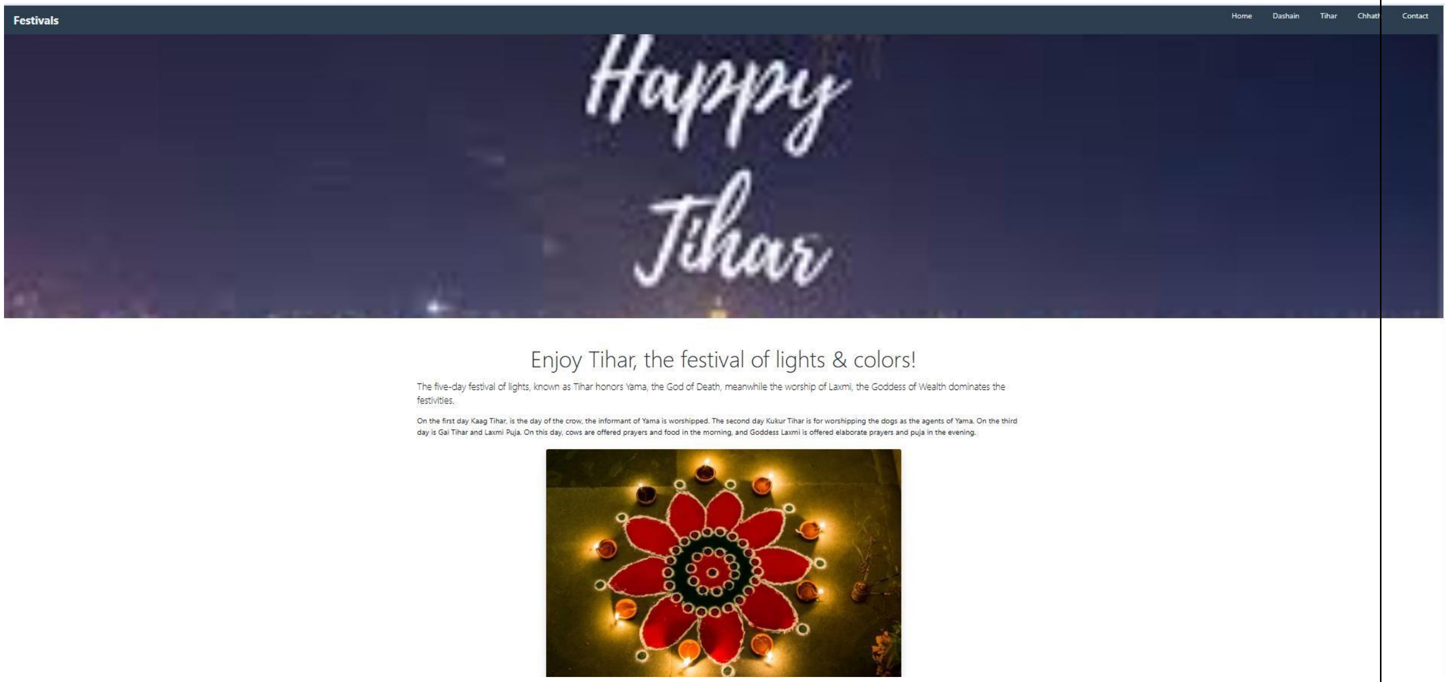


Dashain – Nepal's Greatest Hindu Festival

Dashain is the longest and most anticipated festival in Nepal, celebrated with great devotion, joy, and family gatherings. It honors Goddess Durga's victory over the demon Mahishasura, symbolizing the triumph of good over evil.

The festival spans 15 days, beginning with Ghatasthapana and culminating on Kojagrat Purnima. Major events include Phoolpati, Maha Ashtami, Maha Navami, and Vijaya Dashami, during which tika and jamara are offered as blessings from elders.

Tihar Page:

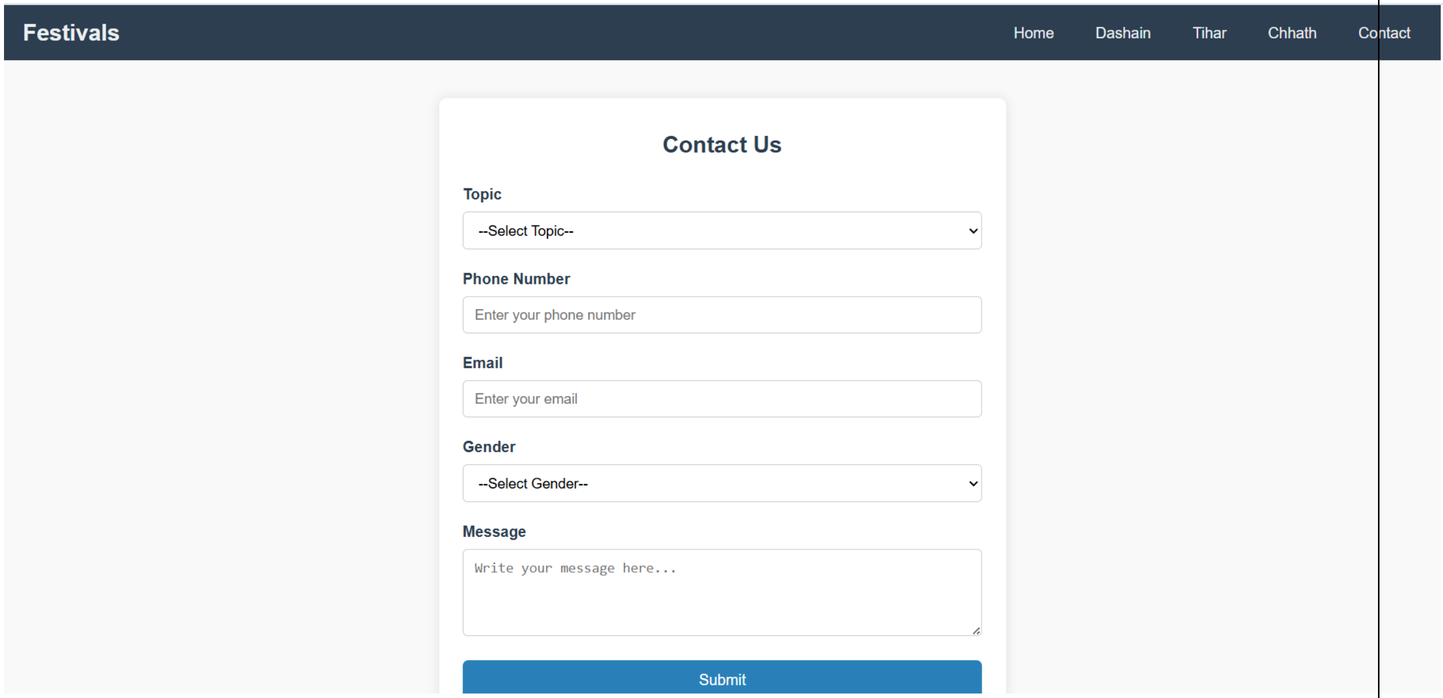


Enjoy Tihar, the festival of lights & colors!

The five-day festival of lights, known as Tihar honors Yama, the God of Death, meanwhile the worship of Laxmi, the Goddess of Wealth dominates the festivities.

On the first day Kaag Tihar, is the day of the crow, the informant of Yama is worshipped. The second day Kukur Tihar is for worshipping the dogs as the agents of Yama. On the third day is Gai Tihar and Laxmi Puja. On this day, cows are offered prayers and food in the morning, and Goddess Laxmi is offered elaborate prayers and puja in the evening.

Contact Page:



Contact Us

Topic

--Select Topic--

Phone Number

Enter your phone number

Email

Enter your email

Gender

--Select Gender--

Message

Write your message here...

GithHub Repository:

Vercel Deployed website code

Discussion

Challenges Faced:

1. Cross-Browser Compatibility:

- The biggest challenge was ensuring consistent functionality across different browsers (e.g., Chrome, Firefox, Safari).

- Certain CSS properties and JavaScript functions were handled differently in some browsers, necessitating additional testing and changes.

2. Responsive Design for Different Screen Sizes:

- It was tricky to create a layout that changed seamlessly across desktops, tablets, and phones.

- Media queries and elastic layouts (Flexbox/Grid) were utilized, but accurate adjusting of the layout for every display size required huge effort.

Drawbacks:

1. Lack of Backend for Dynamic Content Management:

- Because the site is front-end only, everything is static content, and therefore updates are in the form of direct code modification. Dynamic content management is not yet implemented.

Improvements:

1. Add a Blog Section for Articles and Updates:

- Adding a blog section would enhance the site's interest and informativeness by allowing the posting of articles, tutorials, or project updates.

2. Install a Backend for Dynamic Content Management:

- A backend (e.g., Flask, Node.js) may be installed in the future to facilitate dynamic content management, meaning that skills, projects, or blog posts could be updated without manual code edits.

Conclusion

Personally, I felt that this project was fruitful as well as a learning process. I was successfully able to design a personal portfolio website using HTML, CSS, and JavaScript, duly emphasizing my skills, experience, and projects. The website is totally responsive, with proper functionality on various devices such as desktops, tablets, and mobile phones. I used CSS media queries and Flexbox/Grid to make the layout responsive to different screen sizes, with a smooth user experience. JavaScript made the website interactive, with features such as Contact page form validation and dynamic effects like image sliders. Hosting the code on GitHub was a significant achievement, as it allowed me to share my work and demonstrate my expertise with modern development tools. Overall, this project has increased my confidence in front-end web development and provided me with a professional platform to showcase my work.

Future Work

Although I am satisfied with what I have done, I see some scope to improve and develop the website in the future. Below are some of the areas that I will explore:

1. Implement a Backend for Dynamic Content Management:

- **Why:** At present, everything is hardcoded in the HTML files, so changing my skills, projects, or blog posts takes too much time. Having a backend would make content management easier.

- **How:** I would like to learn and use a backend framework like Flask (Python) or Node.js, and add a database like SQLite or MongoDB to store data. I would also create an admin panel, where I could log in and add, edit, or delete content without manually altering the code. This would make the site more dynamic and easier to manage.

2. Add User Authentication:

- **Why:** If I include a backend, I must lock the admin panel and restrict access only to privileged users. User authentication will lock my site and prevent unauthorized changes.

- **How**: I will use libraries like Flask-Login (for Flask) or Passport.js (for Node.js) to manage user authentication. There will be a login page that will allow me to enter my credentials, and I will safely store my password using password hashing so the site is safe while I am able to easily manage content.

3. Improve the Design and User Experience:

- **Why:** Although the site functions, I intend to improve its appearance and usability. An improved-looking site will leave a longer-lasting impression on visitors.

- **How:** I will experiment with new CSS frameworks like Bootstrap or Tailwind CSS to make the styling better. I will include animations and transitions through CSS animations or from libraries like Animate.css. I will be mindful of accessibility, ensuring that the website is keyboard accessible and screen reader accessible.

4. Integrate Analytics and SEO Optimization:

- **Why:** I would like to track how users interact with my website and promote its search engine visibility. This will allow me to understand my audience and drive traffic to the website.

- **How:** I will use Google Analytics to track page views, bounce rate, and demographics. For SEO, I will use meta tags, image alt text, and structured data. In addition, I will use tools like Google Search Console to monitor and enhance the site's search engine performance.

5. Add a Dark Mode Feature:

- **Why:** Dark mode is preferred by all users, especially in low-light environments, as it reduces eye strain. Its implementation will be a better experience for the users.

- **How:** I will use CSS variables to define light and dark mode color schemes. There will be a toggle button to switch between the two, and the preference will be saved using localStorage so that it is remembered between sessions.

Bibliography:

- MDN Web Docs. (n.d.). Introduction to responsive design. Retrieved from [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design](https://developer.mozilla.org/en- US/docs/Learn/CSS/CSS_layout/Responsive_Design)

- W3Schools. (n.d.). Using CSS media queries. Retrieved from [https://www.w3schools.com/css/css_rwd_mediaqueries.asp](https://www.w3schools.com/css/css_rwd_m ediaqueries.asp)

- CSS-Tricks. (n.d.). Comprehensive guide to Flexbox. Retrieved from https://css-tricks.com/snippets/css/a-guide-to-flexbox/

- JavaScript.info. (n.d.). The complete JavaScript tutorial. Retrieved from https://javascript.info/

- Netlify. (n.d.). Official Netlify documentation. Retrieved from https://docs.netlify.com/

- GitHub. (n.d.). Official GitHub documentation. Retrieved from https://docs.github.com/en

Declaration of Originality

I,   **,**declare that this mini-project is my original work and that all references have been appropriately cited.

Signature:
Date: 04 Aug,2025