A REPORT OF ONE MONTH TRAINING

at

SCIENCE & TECHNOLOGY ENTREPRENEURS' PARK (STEP-GNDEC)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

Computer Science and Engineering

JUNE-JULY,2025

SUBMITTED BY:

NAME: EKAMJOT KAUR

UNIVERSITY ROLL NO.2302867

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

ABSTRACT

This report provides a detailed overview of the design and development of two web-based projects created using HTML, CSS, Bootstrap, and JavaScript. The primary goal of these projects was to gain a clear understanding of how front-end web development works and how different technologies are used together to build modern websites. These technologies were combined to structure web pages, style content, and add interactive elements, resulting in fully functional, user-friendly web interfaces.

The development process began with HTML, which was used to create the basic structure of the web pages. CSS was then applied to style the elements, giving the websites a clean and visually appealing layout. Bootstrap, a popular front-end framework, was used to make the designs responsive, meaning the websites adjust properly across different screen sizes and devices. Finally, JavaScript was used to add interactivity to the web pages, such as button actions, form validation, and dynamic content.

During the training, several important topics were covered, including the use of semantic tags in HTML, the implementation of responsive design principles using Bootstrap's grid system, and writing scripts in JavaScript to control page behavior. Each project was built step-by-step, which allowed for a gradual understanding of how code affects the look and functionality of a web page.

ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to all those who supported and guided me throughout the completion of this web development project and training.

I am especially thankful to **Guru Nanak Dev Engineering College (GNDEC), Ludhiana**, for providing me with this opportunity to undertake industrial training as part of the academic curriculum. I extend my sincere appreciation to the **Science & Technology Entrepreneurs' Park (STEP-GNDEC)** for organizing and facilitating a well-structured and practical training environment. The resources, infrastructure, and expert guidance provided at STEP were instrumental in enhancing my understanding of web technologies.

I would also like to thank my training coordinators and faculty mentors for their continuous support, timely feedback, and motivation throughout the training period. Their insights helped me stay focused and deepened my knowledge in HTML, CSS, Bootstrap, and JavaScript.

I am equally thankful to my peers and colleagues during this training, whose collaboration and discussions contributed significantly to my learning experience.

Lastly, I express my deepest gratitude to my family and friends for their constant encouragement and support during the project duration.

This training has proven to be a valuable learning journey and has equipped me with practical skills essential for modern web development.

ABOUT THE INSTITUTE

My training was completed at **STEP–GNDEC (Science & Technology Entrepreneurs' Park)**, which is a part of **Guru Nanak Dev Engineering College (GNDEC), Ludhiana)**. STEP is a great place where students like us get to experience how things work in the real world, outside of books and classrooms.

What makes STEP different is that it doesn't feel like a regular classroom. The whole setup is made to look and feel like an actual workplace, which helps us understand how work is done in companies. The focus here is more on practical learning instead of just theory.

During the training, we worked on real projects and learned how to use different tools and technologies that are used in the IT industry. The environment was very supportive, and the mentors were helpful throughout the learning process. This hands-on experience helped me improve my technical skills and get a better idea of what working in the tech field is really like.

LIST OF TABLES

Definitions, Acronyms and Abbreviations

- HTML (HyperText Markup Language) – The standard language used to create the structure of web pages.
- CSS (Cascading Style Sheets) – A styling language used to design the look and layout of web pages (e.g., colors, fonts, spacing).
- JS (JavaScript) – A scripting language that makes web pages interactive (e.g., forms, buttons, animations).
- Bootstrap – A front-end framework that helps in building responsive and mobile-friendly websites easily using pre-written CSS and JavaScript.
- Web Development – The process of building and maintaining websites, including everything from structure to design and functionality.
- Frontend – The part of a website that users see and interact with (includes HTML, CSS, JavaScript).
- Responsive Design – A design approach that makes websites look good on all devices (desktops, tablets, mobiles).
- IDE (Integrated Development Environment) – A software tool (like VS Code) that helps developers write and manage their code efficiently.
- Tag – A command in HTML that defines the type and structure of the content (e.g., <p>, , <h1>).
- Attribute – Extra information added to an HTML tag to modify its behavior or appearance (e.g., href, src, alt).

Chapter 1 INTRODUCTION

1.1 HTML

1.1.1 WHAT IS HTML

- HTML, short for **Hyper Text Markup Language**, is the basic language used to create and design web pages. It plays a key role in deciding the structure and layout of a webpage.
- Instead of being a programming language, HTML is a **markup language** that uses special tags or "elements" to organize content. These elements help web browsers understand how to display various parts of a page — like headings, paragraphs, images, and links.
- For example, with HTML you can tell the browser:
- “This is a title” using a heading tag
- “This is a paragraph” using a paragraph tag
- “This is a link” using an anchor tag
- In simple terms, HTML acts like the **skeleton** of a website, giving it shape and structure.

1.1.2 HTML EDITORS

Webpages are usually built and updated using specialized **HTML editors** that provide helpful features like syntax highlighting and code suggestions. These tools make web development easier and faster.

However, if you are just beginning to learn HTML, it's best to start with a **basic text editor** such as **Notepad** (on Windows) or **TextEdit** (on Mac). Using a simple editor helps beginners focus on understanding the structure and syntax of HTML, without relying too much on automated tools.

This hands-on method is great for building a strong foundation in HTML coding.

1.1.3 HTML ELEMENTS

TABLE 1.1

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

1.1.4 HTML ATTRIBUTES

- The href attribute in an <a> (anchor) tag is used to specify the **URL** that the link should open.
- The src attribute in an <img> tag defines the **path to the image** that will be displayed on the webpage.
- The width and height attributes in <img> tags are used to set the **size** of the image.
- The alt attribute in <img> tags gives a short **description of the image**, which is shown if the image fails to load or for screen readers.
- The style attribute allows developers to apply **CSS styles** directly to an element — such as font, color, and layout.
- The lang attribute in the <html> tag is used to specify the **language** of the content on the page.
- The title attribute gives **extra information** about an element, which is often shown as a tooltip when the user hovers over it.

1.1.5 HTML STYLES

- The style attribute in HTML allows inline styling of elements, enabling developers to control the appearance of content directly within the HTML tags. It is commonly used for quick styling without writing separate CSS code.
- Key style properties include:
- **background-color** – Sets the background color of an element.

  Example: style="background-color: lightblue;"
- **color** – Changes the color of the text inside the element.

  Example: style="color: red;"
- **font-family** – Specifies the font used for the text.

  Example: style="font-family: Arial;"
- **font-size** – Adjusts the size of the text.

  Example: style="font-size: 18px;"
- **text-align** – Aligns the text within the element (e.g., left, center, right).

  Example: style="text-align: center;"
- These properties help in customizing the layout and appearance of content directly from the HTML code, especially useful in smaller projects or for learning purposes.

1.1.6 HTML FORMATTING

HTML provides a variety of formatting tags designed to present special types of text on a webpage. These elements allow developers to enhance the readability and emphasis of the content in different ways. Below are some commonly used formatting tags:

- **<b>** – Displays the text in bold, typically used for drawing visual attention.
- **** – Indicates important text with semantic emphasis, usually displayed in bold.
- **<i>** – Renders text in italic style, often used for technical terms or titles.

- **** – Emphasizes the text, usually shown in italics, and carries semantic meaning.
- **<mark>** – Highlights text with a background color, similar to using a highlighter.
- **<small>** – Displays text in a smaller font size than the surrounding text.
- **<del>** – Represents deleted or removed text, typically shown with a strikethrough.
- **<ins>** – Denotes inserted text, often underlined to indicate new additions.
- **<sub>** – Used for subscript text, usually for chemical formulas or mathematical notations.

1.1.7 HTML COMMENTS

Comments are not displayed by the browser, but they can help document your HTML source code.

<! -- Write your comments here -->

1.1.8 HTML COLORS

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
<h1 style="background-color:Gray;">Gray</h1>
<h1 style="background-color:SlateBlue;">SlateBlue</h1>
<h1 style="background-color:Violet;">Violet</h1>
<h1 style="background-color:LightGray;">LightGray</h1>

</body>
```

</html>

1.1.9 HTML CSS HTML and CSS work hand-in-hand to control the structure and appearance of web content. Styling in HTML can be achieved in multiple ways, allowing developers to define how elements are displayed in a browser.

Inline Styling with the style Attribute

The style attribute can be applied directly to individual HTML elements to modify their appearance. This method is useful for quick styling but is not ideal for larger projects due to maintainability concerns.

Internal CSS using the <style> Tag

CSS rules can be embedded directly within an HTML document using the <style> element, typically placed inside the <head> section. This method is effective for applying styles to a single page.

External CSS with the <link> Tag

For consistent styling across multiple pages, an external CSS file is linked using the <link> element, which is also placed inside the <head> section. This separates content from design, promoting reusability and cleaner code.

Commonly Used CSS Properties

- **color** – Defines the text color.

- **font-family** – Sets the typeface of the text.
- **font-size** – Specifies the size of the font.
- **border** – Adds a border around an element.
- **padding** – Controls the spacing inside the element's border.
- **margin** – Determines the spacing outside the element's border.

These styling methods and properties are fundamental to modern web development, allowing developers to create visually appealing and user-friendly websites.

1.1.10 HTML LINKS

The HTML <a> tag defines a hyperlink. It has the following syntax:

link text

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

The *link text* is the part that will be visible to the reader.

Clicking on the link text, will send the reader to the specified URL address.

1.1.11 HTML IMAGES

The HTML <img> tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The <img> tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

```
<img src="url" alt="alternatetext">
```

1.1.12 HTML PAGE TITLE

The <title> element adds a title to your page:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Tutorial</title>
</head>
<body>

The content of the document.....

</body>
</html>
```

1.1.13 HTML LAYOUT

HTML has several semantic elements that define the different parts of a web page:

- <header> - Defines a header for a document or a section
- <nav> - Defines a set of navigation links
- <section> - Defines a section in a document
- <article> - Defines independent, self-contained content
- <aside> - Defines content aside from the content (like a sidebar)
- <footer> - Defines a footer for a document or a section
- <details> - Defines additional details that the user can open and close on demand

- <summary> - Defines a heading for the <details> element

1.2 GITHUB

1.2.1 Creating a Repository on GitHub

A repository (or repo) is a space on GitHub where your project files are stored. Think of it like a project folder in the cloud, with extra features like version control, collaboration tools, and change history.

Why create a repository?

If you're working on code, documentation, or even just want to organize files and track changes over time, a GitHub repository helps you store your work safely, share it with others, and manage versions efficiently.

Steps to create a new repository:

1. Open https://github.com in your browser.
2. Make sure you're logged into your GitHub account.
3. In the top-right corner of the screen, click the "+" icon, then click "New repository".
4. You'll see a form. Fill out the following:
    a. Repository name: Choose a clear name (e.g., my-portfolio or calculator-app).
    b. Description: Optional, but helpful. Write what the project is about.
    c. Visibility:
        i. Public: Anyone can see it.
        ii. Private: Only you and people you invite can view it.
5. You can choose to:
    a. Add a README file (recommended): It's like a homepage for your repo.
    b. Add a .gitignore file: Useful if you're working in specific languages (like Python or Node.js).

c. Add a license: Lets others know how they can use your code.

6. Click "Create repository".

Now you have an empty repository! You can start adding files, cloning it to your computer, or inviting others to collaborate.

1.2.2 Adding Files Using a Text Editor (e.g., Notepad)

If you're just starting out, you might not be using a fancy code editor like VS Code yet. That's totally fine — you can write code in a basic editor like Notepad and upload it to GitHub.

Example workflow:

1. Open Notepad (or Notepad++, Sublime Text, or any other editor).
2. Type in your code or content. Example:

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head><title>My Page</title></head>
<body><h1>Hello, GitHub!</h1></body>
</html>
```

3. Click File > Save As.
4. Save it with a proper name and extension, e.g., index.html. Make sure "Save as type" is set to "All Files".
5. Go back to your GitHub repository in the browser.
6. Click the "Add file" button, then choose "Upload files".
7. Drag and drop your file into the upload box or click to select the file from your computer.

8. Below the upload box, write a commit message (something like: “Added index.html homepage”).
9. Click "Commit changes" to upload the file.

You’ve now created a file locally using Notepad and uploaded it to GitHub! This is great for quick edits, writing README files, HTML, markdown, or small scripts.

1.2.3 Using GitHub Desktop

If you’re not comfortable using the command line, GitHub Desktop is a free app that makes it easy to work with GitHub repositories on your computer with a graphical interface.

Why use GitHub Desktop?

- No need to learn Git commands.
- Easily sync your local files with GitHub.
- Great for beginners who want a visual tool to manage code.

Steps to get started:

1. Download GitHub Desktop: https://desktop.github.com
2. Install it on your computer and sign in with your GitHub credentials.
3. After login, you'll see options to:
    a. Clone an existing repository from GitHub.
    b. Create a new repository on your local machine.
    c. Add an existing local repository (if you already have files).

Creating a new repo with GitHub Desktop:

1. Click "File" > "New Repository".
2. Fill in:
    a. Name of your repo
    b. Description (optional)
    c. Local path (where to save the project on your PC)

    d. Initialize with README? (yes, if you want a starting point)

3. Click Create Repository.
4. Now go to your project folder and add or edit files using Notepad or your editor.
5. Go back to GitHub Desktop. It will detect changes automatically.
6. You'll see a list of modified files.
7. Write a summary message for your commit (e.g., "Added about.html").
8. Click "Commit to main".
9. Now click "Push origin" to upload those changes to GitHub.

That's it! You've made changes locally, committed them, and synced them to your GitHub repository — without touching the command line.

1.2.4 Cloning a Repository

What is Cloning?

Cloning means you're copying a project from GitHub.com to your computer. This gives you your own local version that you can edit.

Why Clone?

- You want to work on a project from GitHub
- You want to make changes locally
- You want to learn from someone else's code

Steps to Clone using GitHub Desktop:

1. Open GitHub Desktop
2. Click on File > Clone Repository
3. Choose a repo:
    a. From the list of your repositories
    b. Or paste a GitHub URL in the "URL" tab
4. Choose a local folder to save it

5. Click Clone

Now the entire project is on your computer. You can edit the code using your favorite text editor.

1.2.5 Push: Upload Your Changes to GitHub

What is Push?

Push means sending your local commits (saved changes) to GitHub.com so others can see your work.

Why Push?

- To back up your work to GitHub
- To share your updates with teammates
- To track your progress

Steps to Push using GitHub Desktop:

1. Open the cloned repo folder on your computer
2. Make changes to your files (e.g., update index.html)
3. Return to GitHub Desktop
4. Under the "Changes" tab:
    a. Review your modified files
    b. Enter a summary (e.g., “Fixed footer layout”)
5. Click Commit to main
6. Click Push origin

Your changes are now live on GitHub.com!

1.2.6 Pull: Get the Latest Changes from GitHub

What is Pull?

Pulling means downloading the latest version of the project from GitHub.com to your computer.

Why Pull?

- To make sure your local files are up-to-date
- To avoid conflicts when working with others
- To see what teammates have changed

Steps to Pull using GitHub Desktop:

1. Open the repo in GitHub Desktop
2. Click Fetch origin to check if there are new updates
3. If updates exist, click Pull origin

Now your local project includes the latest changes from GitHub.

Tip: Always pull before you start working to avoid editing outdated code.

1.2.7 Fork: Make Your Own Copy of Someone Else's Project

What is a Fork?

Forking is creating your own copy of someone else's repository on GitHub.com.

Why Fork?

- You want to experiment or improve someone else's project
- You want to submit changes (pull requests)
- You don't have permission to edit the original repo

Steps to Fork a Repo:

1. Go to the repository page on GitHub.com
2. Click the Fork button (top-right corner)
3. GitHub will copy the repo to your account
4. Open GitHub Desktop
5. Click File > Clone Repository and choose your fork

Now you can edit the project freely in your account and submit pull requests if you want to contribute back.

1.2.8 Create and Use Branches

What is a Branch?

A branch is like a copy of your code where you can safely experiment with new features.

Main branch = published version

Other branches = test environments

Why Use Branches?

- You can work on new features without breaking your main code
- Great for teamwork — each person works on their own branch
- Helps organize features and bug fixes

Steps to Create a Branch in GitHub Desktop:

1. Open the repo in GitHub Desktop
2. At the top, click on the current branch name (e.g., "main")
3. Click New Branch
4. Give it a name (e.g., feature-contact-form)
5. Click Create Branch

You are now working in your new branch.

Make your changes, commit them, and push just like normal. GitHub Desktop will push your branch separately.

When you're ready to add your changes to the main project, you can go to GitHub.com and create a Pull Request from your new branch into main.

1.3 CSS

1.3.1 What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once

1.3.2 Using CSS in HTML

CSS can be added to HTML documents in 3 ways:

- Inline - by using the style attribute inside HTML elements
- Internal - by using a <style> element in the <head> section
- External - by using a <link> element to link to an external CSS file

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

Example

```
<h1 style="color:blue;">A Blue Heading</h1>
<p style="color:red;">A red paragraph.</p>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
```

```
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

"styles.css":

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

1.3.3 CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Example

The CSS element Selector

```
p {
  text-align: center;
```

```
  color: red;
}
```

The CSS id Selector

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#para1 {
  text-align: center;
  color: red;
}
```

The CSS class Selector

To select elements with a specific class, write a period (.) character, followed by the class name.

```
.center {
  text-align: center;
  color: red;
}
```

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

```
* {
  text-align: center;
  color: blue;
}
```

The CSS Grouping Selector

the h1, h2, and p elements have the same style definitions

```
h1, h2, p {
  text-align: center;
```

```
  color: red;
}
```

1.3.4 CSS Border Style

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
</style>
</head>
<body>

<h2>The border-style Property</h2>
```

```
<p>This property specifies what kind of border to display:</p>

<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>

</body>
</html>
```

1.3.5 Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- margin-left

```
p {
  margin-top: 100px;
```

```
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

1.3.6 CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

EXAMPLE

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

1.3.7 CSS Setting height and width

The height and width properties are used to set the height and width of an element.

```
div {
  height: 200px;
  width: 50%;
```

```
  background-color: powderblue;
}
```

1.3.8 The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

Example

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

1.3.9 CSS Fonts

In CSS there are five generic font families:

1. Serif fonts have a small stroke at the edges of each letter. They create *a* sense of formality and elegance.
2. Sans-serif fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.

3. Monospace fonts - here all the letters have the same fixed width. They create a mechanical look.
4. Cursive fonts imitate human handwriting.
5. Fantasy fonts are decorative/playful fonts.

1.4 BOOTSTRAP

1.4.1 BOOTSTRAP VERSIONS

Bootstrap 3 is the most stable version of Bootstrap, and it is still supported by the team for critical bugfixes and documentation changes.

Bootstrap 4 is a newer version of Bootstrap; with new components, faster stylesheet and more responsiveness. However, Internet Explorer 9 and down is not supported.

Bootstrap 5 is the newest version of Bootstrap; with a smooth overhaul. However, Internet Explorer 11 and down is not supported, and jQuery is replaced with vanilla JavaScript

We will study Bootstrap 5.

1.4.2 What is Bootstrap?

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

1.4.3 Bootstrap 5 Containers

Containers are used to pad the content inside of them, and there are two container classes available:

1. The .container class provides a responsive fixed width container

2. The .container-fluid class provides a full width container, spanning the entire width of the viewport

Fixed Container

Use the .container class to create a responsive, fixed-width container.

Example

```
<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```

Fluid Container

Use the .container-fluid class to create a full width container, that will always span the entire width of the screen (width is always 100%):

Example

```
<div class="container-fluid">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```

Container Padding

By default, containers have left and right padding, with no top or bottom padding. Therefore, we often use spacing utilities, such as extra padding and margins to make them look even better. For example, .pt-5 means "add a large top padding"

```
<div class="container pt-5"></div>
```

Container Border and Color

```
<div class="container p-5 my-5 border"></div>
```

```
<div class="container p-5 my-5 bg-dark text-white"></div>

<div class="container p-5 my-5 bg-primary text-white"></div>
```

1.4.4 Bootstrap 5 Grids

bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.

The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Make sure that the sum adds up to 12 or fewer

Example:

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	sp
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

Example:

```
<div class="container-fluid mt-3">
<div class="row">
    <div class="col-sm-3  p-3 bg-primary text-white">.col</div>
    <div class="col-sm-3  p-3 bg-dark text-white">.col</div>
    <div class="col-sm-3  p-3 bg-primary text-white">.col</div>
    <div class="col-sm-3  p-3 bg-dark text-white">.col</div>
  </div>
</div>
```

1.4.5 Bootstrap 5 Text

<h1> - <h6>

Bootstrap 5 styles HTML headings (<h1> to <h6>) with a bolder font-weight and a responsive font-size.

Example

```
<p class="h1">h1 Bootstrap heading</p>

<p class="h2">h2 Bootstrap heading</p>

<p class="h3">h3 Bootstrap heading</p>

<p class="h4">h4 Bootstrap heading</p>

<p class="h5">h5 Bootstrap heading</p>

<p class="h6">h6 Bootstrap heading</p>
```

Display Headings

```
<h1 class="display-1">Display 1</h1>

 <h1 class="display-2">Display 2</h1>

 <h1 class="display-3">Display 3</h1>

 <h1 class="display-4">Display 4</h1>

 <h1 class="display-5">Display 5</h1>

 <h1 class="display-6">Display 6</h1>
```

1.4.6 Bootstrap 5 Colors

Text Colors

The classes for text colors are: .text-muted, .text-primary, .text-success, .text-info, .text-warning, .text-danger, .text-secondary, .text-white, .text-dark, .text-body (default body color/often black) and .text-light

Background Colors

The classes for background colors are: .bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light.

1.4.7 Bootstrap 5 Tables

Basic Table

- Use the .table class for basic styling, which includes light padding and horizontal dividers.

Table Variants and Features

1. Striped Rows
    a. .table-striped adds alternating background colors to rows.
2. Bordered Table
    a. .table-bordered adds borders around all cells and the table.
3. Hover Rows
    a. .table-hover adds a grey background to rows on hover.
4. Dark Table
    a. .table-dark applies a black background with white text to the table.
5. Dark Striped Table
    a. Combine .table-dark and .table-striped to create a dark table with striped rows.
6. Hoverable Dark Table
    a. Add .table-hover to a dark table for hover effects on rows.
7. Borderless Table
    a. .table-borderless removes all borders from the table.

Contextual Classes

- Used to apply color to an entire table, a row (<tr>), or a cell (<td>).

- Common classes:
    - .table-primary, .table-success, .table-danger, .table-info, .table-warning, .table-active, .table-secondary, .table-light, .table-dark.

1.4.8Bootstrap 5 Images

Rounded Corners

Example

```
<img src="cinqueterre.jpg" class="rounded" alt="Cinque Terre">
```

Circle

Example

```
<img src="cinqueterre.jpg" class="rounded-circle" alt="Cinque Terre">
```

Thumbnail

Example

```
<img src="cinqueterre.jpg" class="img-thumbnail" alt="Cinque Terre">
```

1.4.9 Bootstrap 5 Alerts

Basic Alerts

- Created using .alert class plus a contextual class:
    - Examples: .alert-success, .alert-danger, .alert-info, .alert-warning, etc.
- Displays colored alert messages to convey different types of information.

Example:

```
<div class="alert alert-success">
  <strong>Success!</strong> Indicates a successful or positive action.
</div>
```

Alert Links

- Use .alert-link on links inside alerts to make them styled like the alert.

Example:

```
<div class="alert alert-success">
  <strong>Success!</strong> You should <a href="#" class="alert-link">read this message</a>.
</div>
```

Closing Alerts

- Add .alert-dismissible to make alerts closable.
- Include a button with class="btn-close" and data-bs-dismiss="alert".

Example:

```
<div class="alert alert-success alert-dismissible">
  <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
  <strong>Success!</strong> This alert box could indicate a successful or positive action.
</div>
```

Animated Alerts

- Use .fade and .show classes for a smooth fade-out effect when the alert is dismissed.

Example:

```
<div class="alert alert-danger alert-dismissible fade show">
  <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
  <strong>Danger!</strong> This is an animated alert.
```

1.4.10 Bootstrap buttons

Bootstrap 5 provides different styles of buttons using the .btn class with contextual classes like .btn-primary, .btn-secondary, .btn-success, .btn-info, .btn-warning, .btn-danger, .btn-dark, .btn-light, and .btn-link.

Buttons can be used with <a>, <button>, or <input> elements.

The href="#" is used when there is no real page to link to, to avoid a "404" error. In real use, it should be replaced with an actual URL.

Bootstrap 5 also provides eight outline buttons using .btn-outline-* classes. These buttons have a hover effect.

Button sizes can be changed using .btn-lg for large buttons and .btn-sm for small buttons.

To make block-level buttons that span the full width of the parent, use the .d-grid class. Use .gap-* to control spacing between multiple block buttons.

Buttons can be made active using the .active class or disabled using the disabled attribute. <a> elements use the .disabled class instead of the disabled attribute.

Spinners can be added to buttons using .spinner-border or .spinner-grow along with the .spinner-border-sm or .spinner-grow-sm class.

1.4.11 Bootstrap 5 Progress Bars

Basic Progress Bar

- Structure:

```
<div class="progress">
  <div class="progress-bar"  style="width:70%"></div>
```

- Use the width style to represent progress.

Progress Bar Height

- Default height: 1rem (usually 16px).
- Change height with inline style:

```
<div class="progress"  style="height:20px">...</div>
```

Progress Bar Labels

- Add text inside the .progress-bar:

```
<div class="progress-bar"  style="width:70%">70%</div>
```

Colored Progress Bars

Use contextual background classes:

- .bg-success (green)
- .bg-info (turquoise)
- .bg-warning (orange)
- .bg-danger (red)
- .bg-secondary, .bg-light, .bg-dark, .bg-white

Example:

```
<div class="progress">
  <div class="progress-bar bg-warning" style="width:40%"></div>
</div>
```

Striped Progress Bars

- Add .progress-bar-striped for stripes:

```
<div class="progress-bar progress-bar-striped"></div>
```

Animated Progress Bars

- Add .progress-bar-animated for moving stripes:

```
<div class="progress-bar progress-bar-striped progress-bar-animated"></div>
```

Multiple (Stacked) Progress Bars

- Place multiple .progress-bar divs inside one .progress:

```
<div class="progress">
  <div class="progress-bar bg-success" style="width:40%">Free Space</div>
  <div class="progress-bar bg-warning" style="width:10%">Warning</div>
  <div class="progress-bar bg-danger" style="width:20%">Danger</div>
</div>
```

1.4.12 Bootstrap 5 Spinners

Basic Spinner

- Use .spinner-border for a spinning loader:

```
<div class="spinner-border"></div>
```

Colored Spinners

- Use text color utility classes to change spinner color:

```
<div class="spinner-border text-primary"></div>
<div class="spinner-border text-success"></div>
```

- Available colors:
    - text-muted
    - text-primary
    - text-success
    - text-info
    - text-warning
    - text-danger
    - text-secondary
    - text-dark
    - text-light

Growing Spinners

- Use .spinner-grow for a pulsing/growing effect:

```
<div class="spinner-grow  text-info"></div>
```

Spinner Sizes

- Add .spinner-border-sm or .spinner-grow-sm for smaller spinners:

```
<div class="spinner-border  spinner-border-sm"></div>
<div class="spinner-grow  spinner-grow-sm"></div>
```

Spinner in Buttons

- Add spinners inside buttons to show loading states:

```
<button class="btn btn-primary">
  <span class="spinner-border  spinner-border-sm"></span>  Loading..
</button>
```

- You can disable the button with the disabled attribute:

```
<button class="btn btn-primary"  disabled>
  <span class="spinner-grow  spinner-grow-sm"></span>  Loading..
</button>
```

1.5 Javascript

1.5.1 Introduction to JavaScript

What is JavaScript?

JavaScript is a programming language primarily used for web development. It enables dynamic behavior on web pages by allowing changes to HTML and CSS, as well as performing calculations, data manipulation, and validation.

Why Learn JavaScript?

JavaScript is one of the core technologies of web development, alongside:

1. HTML – defines content
2. CSS – defines layout
3. JavaScript – defines behavior

Key JavaScript Capabilities:

- Change HTML content using methods like getElementById(), e.g.:

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

- Change HTML attributes, such as modifying an image's src value.
- Change HTML styles (CSS), such as font size:

```
document.getElementById("demo").style.fontSize = "35px";
```

- Hide elements by setting display to none
- Show elements by setting display to block

1.5.2 where to

JavaScript code is inserted into an HTML document using the <script> tag. This tag can be placed either within the <head> section or the <body> section of the HTML page. You can include multiple scripts anywhere in the document, but placing scripts at the end of the <body> is recommended because it allows the page content to load faster before the scripts run.

JavaScript functions, which are blocks of code, can be executed when triggered by events like button clicks. Functions can be defined either in the <head> or <body>, depending on where the <script> tag is placed.

External JavaScript files with a .js extension can also be linked using the <script src="filename.js"></script> tag, placed either in the <head> or <body>. Using external files helps keep HTML and JavaScript separate, improves code maintenance, and allows browsers to cache scripts for faster loading. External script references can use full URLs, file paths, or just filenames, depending on the location of the JavaScript file.

1.5.3 Javascript Keywords

TABLE 1.2

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

1.5.4 Javascript Variables

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

1.5.5 Javascript operators

Arithmetic Operators

TABLE 1.3

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Assignment Operators

TABLE 1.4

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Comparison Operators

TABLE 1.5

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Logical Operators

TABLE 1.6

Operator	Description
&&	logical and
||	logical or
!	logical not

Chapter 2 TRAINING WORK UNDERTAKEN

2.1 HTML WORK AND PROJECTS

2.1.1 A Simple HTML Document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
```

```
</head>
<body>
 <h1>My First Heading</h1>
<p>My first paragraph. </p>
 </body>
</html>
```

2.1.2 HTML HEADINGS

```
<!DOCTYPE html>
<html>
<body>
 <h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
 </body>
</html>
```

2.1.3 HTML TABLES

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  font-family: arial, sans-serif;
```

```
  border-collapse: collapse;
  width: 100%;
}
 td, th {
  border: 1px solid #dddddd;
  text-align: left;
  padding: 8px;
}
tr:nth-child(even) {
  background-color: #dddddd;
}
</style>
</head>
<body>
 <h2>HTML Table</h2>
 <table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
```

```
</tr>
<tr>
  <td>Centro comercial Moctezuma</td>
  <td>Francisco Chang</td>
  <td>Mexico</td>
</tr>
<tr>
  <td>Ernst Handel</td>
  <td>Roland Mendel</td>
  <td>Austria</td>
</tr>
<tr>
  <td>Island Trading</td>
  <td>Helen Bennett</td>
  <td>UK</td>
</tr>
<tr>
  <td>Laughing Bacchus Winecellars</td>
  <td>Yoshi Tannamuri</td>
  <td>Canada</td>
</tr>
<tr>
  <td>Magazzini Alimentari Riuniti</td>
  <td>Giovanni Rovelli</td>
  <td>Italy</td>
```

```
  </tr>
</table>
 </body>
</html>
```

2.1.4 HTML AND CSS PROJECT -1

Topic- Create a personal hobby website using HTML

```
<html>
<head>
  <title>Ekam's Hobbies</title>
  <style>
       section {
     display: none;
         }
   section:target {
     display: block;
   }
  </style>
</head>
<body style=" background: linear-gradient(to right, #f0f4f7, #d9e2ec);">

  <nav>
    <ul style="list-style:none; margin:0; padding:0; background:#333; display:flex;">
      <li style="flex:1;"><a href="#home" style="display:block; padding:15px; color:white; text-
align:center; text-decoration:none;">Home</a></li>
```

```
    <li style="flex:1;"><a href="#about" style="display:block; padding:15px; color:white; text-
align:center; text-decoration:none;">About Me</a></li>
    <li style="flex:1;"><a href="#hobbies" style="display:block; padding:15px; color:white; text-
align:center; text-decoration:none;">My Hobbies</a></li>
    <li style="flex:1;"><a href="#shows" style="display:block; padding:15px; color:white; text-
align:center; text-decoration:none;">Mystery Shows</a></li>
    <li style="flex:1;"><a href="#recipes" style="display:block; padding:15px; color:white; text-
align:center; text-decoration:none;">Recipes</a></li>
    <li style="flex:1;"><a href="#contact" style="display:block; padding:15px; color:white; text-
align:center; text-decoration:none;">Contact</a></li>
  </ul>
 </nav>
  <div id="slogan" style="font-size:28px; font-weight:bold; text-align:center; margin:40px 20px;
color:#333;">
  Discover Your Passion — Dive Into Mystery and Flavor!
 </div>
  <section id="home">
  <h1 style="color:#333;">Ekam's Hobbies</h1>
  <h3 style="color:red;">Welcome to my hobby website!</h3>
  <img src="https://i.pinimg.com/736x/a0/cf/66/a0cf666d0a6e881ebdb3c4f3df7be945.jpg"
width="300" style="display:block; margin-bottom:15px;">
  <p>Hi there! I’m Ekam, and I’m so glad you stopped by. This is where I share two of my
passions — mystery shows and cooking.</p>
  <h4 style="color:red;">Not sure what your hobby is? Click below!</h4>
```

Find Your Hobbies
</section>
<section id="about">
<h1 style="color:#333;">About Me</h1>
<h3 style="color:red;">Who is Ekam?</h3>
I’m a tech student and a hobbyist who enjoys solving mysteries and making delicious food.
 I’m a curious soul who finds joy in two very different but equally fascinating worlds: mystery shows and cooking. I love solving puzzles and uncovering secrets in the shows I watch, and when I’m not glued to the screen, you’ll find me trying out new recipes or inventing my own. This site is my personal space to share the things I love — welcome aboard!
Currently, I’m pursuing a Bachelor of Technology (B.Tech) degree in Computer Science and Engineering (CSE). My academic journey has given me a deep appreciation for logic, problem-solving, and creativity — which are the very same skills I enjoy applying in both mystery shows and cooking! Whether it’s writing code to solve a real-world problem or decoding a suspenseful storyline, I find joy in thinking critically and exploring new ideas.
Balancing tech with my hobbies has helped me grow in unique ways — from debugging code to mastering a new recipe, I believe that learning never stops. Through this website, I hope to share not just what I love, but also how I learn, create, and have fun along the way.
</section>
<section id="hobbies">
<h1 style="color:#333;">My Hobbies</h1>
<h3 style="color:red;">My Two Great Passions</h3>
<h2 style="color:#222;">Watching Mystery Shows</h2>

```
   <img src="https://i.pinimg.com/736x/f7/cd/6f/f7cd6fb0be705fd87c66ef186b5ce00c.jpg"
width="300" style="display:block; margin-bottom:15px;">
   <p>I love piecing together clues, analyzing characters, and solving puzzles before the show
reveals them.</p>
   <a href="https://youtube.com/shorts/JxL249JOtxA?si=UHJn_Wkm6MK1X-MN"
style="color:#0066cc;">Best  Mystery Movies</a>
 <h2 style="color:#222; margin-top:30px;">Cooking</h2>
   <img src="https://i.pinimg.com/736x/04/cf/54/04cf54e29ea427ea94b2e84b9d965208.jpg"
width="300" style="display:block; margin-bottom:15px;">
   <p>Cooking helps me relax and be creative. I enjoy baking cookies, preparing Indian curries,
and trying new dishes.</p>
   <a href="https://youtu.be/pKAGEC12bog?si=RLSqFUTh-RsZjwxs"
style="color:#0066cc;">Learn  Cooking Basics</a>
  </section>
 <section id="shows">
   <h1 style="color:#333;">Mystery  Shows</h1>
   <table border="1" style="border-collapse: collapse; width: 100%; background:white;">
     <tr style="background:#eee;">
       <th>Show Name</th>
       <th>Genre</th>
       <th>Main Theme</th>
       <th>Year</th>
     </tr>
     <tr><td>Sherlock (BBC)</td><td>Crime</td><td>Logical detective
work</td><td>2010</td></tr>
```

<tr><td>Stranger Things</td><td>Fantasy</td><td>Supernatural mystery</td><td>2016</td></tr>

<tr><td>Criminal Minds</td><td>Crime</td><td>Criminal profiling</td><td>2005</td></tr>

<tr><td>Dark</td><td>Sci-Fi</td><td>Time travel secrets</td><td>2017</td></tr>

<tr><td>Only Murders</td><td>Comedy</td><td>Podcast mystery</td><td>2021</td></tr>

</table>

<h3 style="color:red;">Top 5 Mystery Shows</h3>

<h2> What Are Mystery Shows?</h2>

 Mystery shows are gripping narratives that revolve around solving a crime or unraveling a mystery. They keep you guessing, often filled with twists and turns, while following detectives, investigators, or regular people trying to crack a case. These shows are a thrilling ride that challenges the mind and pulls you deeper into the story as it unfolds.

<h2> 1.Sherlock(BBC)</h2>

A modern, fast-paced version of the classic Sherlock Holmes stories. This series follows the brilliant but eccentric detective Sherlock Holmes (played by Benedict Cumberbatch) and his partner Dr. John Watson (played by Martin Freeman) as they solve complex cases using sharp logic and cutting-edge technology. The dynamic between the characters, along with the clever storytelling and mind-bending mysteries, makes it a must-watch for any mystery lover.

<h2> 2.Stranger Things </h2>

While Stranger Things is known for its supernatural elements, it’s also a mystery at heart. A group of kids in the 1980s discovers strange occurrences in their small town, leading them to uncover a government experiment and a parallel universe. As the group navigates the mysteries of the Upside Down, they uncover deeper secrets and face unknown threats. It’s a unique blend of sci-fi, horror, and mystery, with a lot of heart and suspense.

<h2>3.Criminal Minds</h2>

Criminal Minds focuses on the FBI's Behavioral Analysis Unit (BAU), which solves crimes by profiling and understanding the psychology of criminals. Each episode presents a different case, typically involving violent crimes like serial murders or kidnappings, and explores how the team identifies and tracks down the perpetrator. The show is intense, fascinating, and sheds light on the darker side of the human mind.

<h2>4.Dark (German)</h2>

Dark is a complex time-travel mystery that blends science fiction and suspense. Set in a small German town, the series follows multiple families and their interwoven destinies over several generations. With a mysterious disappearance leading to the discovery of time travel, Dark explores themes of fate, free will, and paradoxes, all while building a dense, multi-layered plot that keeps you hooked. It’s a truly mind-bending show that requires your full attention, but it rewards with brilliant storytelling.

<h2>5.Only Murders in the Building.</h2>

A quirky and light-hearted mystery show that takes a more comedic approach. The series follows three strangers, played by Steve Martin, Martin Short, and Selena Gomez, who live in a luxury apartment building in New York. After a murder occurs in their building, they start their own podcast to investigate the crime, all while becoming amateur detectives. It’s a fun, charming, and full of surprises, blending mystery with humor and touching moments.

</section>

<section id="recipes">

<h1 style="color:#333;">Recipes</h1>

Cooking is one of my greatest joys, and I love experimenting with flavors, trying out new recipes, or perfecting old favorites. Here are a few of my go-to dishes that always bring a smile to my face, whether I’m cooking alone or sharing them with family and friends!

<table border="1">

```
<thead>
  <tr>
    <th>Dish Name</th>
    <th>Description</th>
  </tr>
</thead>
<tbody>
  <tr>
    <td>🍝 Creamy Garlic Pasta</td>
    <td>This dish is my ultimate comfort food. It’s simple to make yet packs so much flavor. The creamy, garlicky sauce combines with the richness of Parmesan to coat each strand of pasta. Perfect for busy weeknights or lazy weekends, and you can customize it with veggies or grilled chicken.</td>
  </tr>
  <tr>
    <td>🍪 Chocolate Chip Cookies</td>
    <td>Who can resist a warm, freshly baked chocolate chip cookie? With a crispy edge and gooey center, this classic treat is always a crowd-pleaser. I balance semi-sweet and milk chocolate chips with a pinch of sea salt for that perfect bite. A true homemade classic!</td>
  </tr>
  <tr>
    <td>🥙 Mixed Veggie Salad</td>
```

<td>This fresh, healthy salad is packed with crunchy cucumbers, juicy tomatoes, colorful bell peppers, and creamy avocado. A simple vinaigrette made from olive oil, balsamic vinegar, and Dijon mustard ties everything together. Great as a side dish or a light meal!</td>

</tr>

<tr>

<td>🍛 Butter Paneer</td>

<td>A rich, creamy dish made with soft paneer (Indian cottage cheese) simmered in a tomato gravy with butter, cream, and aromatic Indian spices. It’s indulgent and flavorful, perfect with naan or rice for a satisfying meal. A must-try for any fan of Indian cuisine!</td>

</tr>

<tr>

<td>🍰 No-Bake Cheesecake</td>

<td>This easy, no-bake cheesecake is a dream for those who crave something sweet without the oven time. With a graham cracker base and a creamy filling of cream cheese, sugar, and vanilla, it's topped with fresh fruit or berry compote. Perfect for any occasion and so simple to make!</td>

</tr>

</tbody>

</table>

<h3 style="color:red;">My Go-To Dishes in the Kitchen</h3>

<h2> 1. Creamy Garlic Pasta</h2>

<p>This dish is my ultimate comfort food. It’s simple to make yet packs so much flavor. I love the creamy, garlicky sauce that coats each strand of pasta. The richness of the cream combined with the savory depth of garlic and Parmesan is a match made in heaven. It’s the kind of dish that feels

indulgent but doesn’t require a lot of time in the kitchen. I often throw in a little bit of sautéed spinach or grilled chicken to make it a complete meal. Whether it's a busy weeknight or a lazy weekend, this pasta never disappoints

<h2> 2.Chocolate Chip Cookies </h2>

Who can resist a warm, freshly baked chocolate chip cookie? I’ve spent years perfecting my recipe, and I think I've finally nailed it. The key is in the balance between the crispy edge and the soft, gooey center. When baking, I like to use a mix of semi-sweet and milk chocolate chips for that perfect chocolatey bite. I often add a pinch of sea salt on top right before baking — it enhances the sweetness and adds a little something extra. These cookies are always a crowd-pleaser, and they disappear faster than you can say "cookie monster"!

<h2>3.Mixed Veggie Salad</h2>

For those days when I want something fresh and healthy, this salad is my go-to. It's filled with a variety of colorful vegetables, from crunchy cucumbers and bell peppers to juicy tomatoes and fresh lettuce. I love adding some avocado for creaminess and roasted nuts or seeds for a bit of crunch. The dressing is a simple vinaigrette made from olive oil, balsamic vinegar, and Dijon mustard — tangy and zesty, just the way I like it. Not only is this salad refreshing, but it also makes a great side dish to balance out any heavy meal. Plus, it's packed with vitamins and nutrients, so you can't go wrong!

<h2>4.Butter Paneer </h2>

Rich, creamy, and full of bold spices — this dish is a true indulgence. Butter Paneer is one of my absolute favorites when I’m craving something flavorful and hearty. The soft cubes of paneer (Indian cottage cheese) are simmered in a luscious, spiced tomato gravy made with butter, cream, and a mix of aromatic Indian spices. The result is a dish that’s velvety smooth and full of flavor. I love serving this with warm naan or basmati rice, and I can’t resist a little extra drizzle of cream on

top. Whether you’re a fan of Indian food or new to it, butter paneer is a dish that never fails to impress.

<h2>5.No-Bake Cheesecake</h2>

If I’m craving something sweet but don’t want to spend hours baking, this no-bake cheesecake is my solution. It's incredibly easy to make, and you don’t even need to turn on the oven! The base is made from crushed graham crackers mixed with melted butter, and the creamy filling is a blend of cream cheese, sugar, and vanilla. After chilling in the fridge for a few hours, the cheesecake sets into a smooth, velvety texture that’s both rich and refreshing. To top it off, I often add a drizzle of berry compote or a handful of fresh fruits like strawberries and blueberries. It’s the perfect dessert for any occasion — simple yet decadent.

</section>

<section id="contact">

<h1 style="color:#333;">Contact</h1>

Email me at ekamjotkaur3636@gmail.com

<ul style="padding-left: 20px;">

Twitter

Facebook

Instagram

</section>

<footer style="background:#333; color:white; text-align:center; padding:20px; margin-top:40px;">

© 2025 Ekam's Hobbies. All rights reserved. | Designed with ♥ by Ekam

</footer>

</body>

</html>

2.1.5 Html project –2

How to link different web pages using HTML?

First .html

```
<html>

<head>

</head>

<body>

<a href="Web dev/middle.htm">Go to Middle</a><br>

<a href="Web dev/web2/last.htm">Go to last</a><br>

</body>

</html>
```

Middle.html

```
<html>

<head>

</head>

<body>

<a href="../first.htm">Back to First</a><br><a href="web2/last.htm">Go to last</a><br>

</body>

</html>
```

Last.html

```
<html>
<head>
</head>
<body>
<a href="../../first.htm">Back to First</a><br>
<a href="../middle.htm">Back to Middle</a><br>
 </body>
</html>
```

2.2 BOOTSTRAP WORK

2.2.1 A Simple bootstrap program

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Bootstrap Page</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
  <header class="bg-primary  text-white text-center p-4">
    <h1>Welcome to My Page</h1>
  </header>
  <div class="container mt-4">
    <div class="row">
      <div class="col-md-3 bg-warning p-3">
       This is Column 1
```

```
    </div>
    <div class="col-md-6 bg-success text-white p-3">
      This is Column 2 (larger)
    </div>
    <div class="col-md-3 bg-info p-3">
      This is Column 3
    </div>
   </div>
  </div>
  <footer class="bg-dark text-white text-center p-3 mt-4">
   &copy; 2025 My Footer
  </footer>
</body>
</html>
```

2.3 JAVASCRIPT WORK

2.3.1 SIMPLE JAVASCRIPT PROGRAM

```
<!DOCTYPE html>
<html>
<body>
<button onclick="myFunction()">Driving Test </button>
<p id="demo"></p>
<p id="text"></p>
<script>
function myFunction() {
  let text;
```

```
  let age = prompt("Please enter your age:", "15");
  if (age == null || age == "") {
    text = "User cancelled the prompt.";
  } else {
    text = "Your age is " + age ;
  }
  document.getElementById("demo").innerHTML = text;
if (age >= 18) {
    document.getElementById("text").innerHTML = "The user is eligible for driving.";
  } else {
    document.getElementById("text").innerHTML = "The user is not eligible for driving.";
  }
}</script>
</body>
</html>
```

2.4 FINAL PROJECT –1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>SkillSphere - SkillSwap Hub</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" />
```

```
  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600&display=swap"
rel="stylesheet" />
<style>
    * { font-family: 'Inter', sans-serif; }

    body, html {
      height: 100%;
      margin: 0;
      color: #fff;
      scroll-behavior: smooth;
    }

    .bg-img {
      background-image: url('https://images.unsplash.com/photo-1522071820081-009f0129c71c');
      background-size: cover;
      background-position: center;
      height: 100vh;
      position: relative;
      overflow-y: auto;
    }

    .overlay {
      background-color: rgba(0, 0, 0, 0.6);
      backdrop-filter: blur(2px);
      top: 0;
```

```
  left: 0;
  width: 100%;
  height: 100%;
  padding: 2rem;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.tab-content {
  margin-top: 2rem;
  width: 100%;
  max-width: 900px;
  background: rgba(255, 255, 255, 0.05);
  padding: 1.5rem;
  border-radius: 0.5rem;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.7);
  overflow-y: auto;
}

.nav-tabs .nav-link.active {
  background-color: #0d6efd;
  color: #fff;
}
```

```
.nav-tabs .nav-link {
  color: #ccc;
  transition: color 0.3s;
}

input, textarea, select {
  background-color: #222;
  border: 1px solid #444;
  color: #eee;
  transition: 0.2s ease;
}

input::placeholder {
  color: #888;
}

.form-control:focus {
  background-color: #333;
  border-color: #0d6efd;
  color: #fff;
  box-shadow: 0 0 0 0.2rem rgba(13, 110, 253, 0.25);
}

.table thead th {
  color: #ddd;
```

```
  border-bottom: 2px solid #0d6efd;
}

.table tbody tr:hover {
  background-color: rgba(13, 110, 253, 0.2);
}

.moving-banner {
  position: fixed;
  top: 0;
  width: 100%;
  background: #0d6efd;
  color: #fff;
  text-align: center;
  font-weight: 600;
  padding: 0.4rem 1rem;
  z-index: 1000;
  overflow: hidden;
}

.moving-text {
  display: inline-block;
  white-space: nowrap;
  animation: scrollText 12s linear infinite;
}
```

```
@keyframes scrollText {
  0% { transform: translateX(100%); }
  100% { transform: translateX(-100%); }
}

footer {
  text-align: center;
  color: #ccc;
  font-size: 0.8rem;
  margin-top: 1rem;
}

.btn {
  transition: 0.3s ease-in-out;
}

.btn:hover {
  transform: scale(1.03);
}

#formMsg {
  font-weight: bold;
  margin-top: 0.5rem;
}
```

```
    .toast-container {
      position: fixed;
      bottom: 1rem;
      right: 1rem;
      z-index: 1050;
    }

    .toast {
      background-color: #0d6efd;
      color: white;
    }
  </style>
</head>
<body>
<div class="moving-banner">
    <div class="moving-text">
      Created by Ekamjot Kaur | URN: 2302867 | CRN: 2315264
    </div>
  </div>
  <div class="bg-img">
    <div class="overlay text-center">

      <h1>Welcome to skillsphere</h1>
      <p>This is a peer to peer skill learning platform </p>
```

```
    <!-- Navigation Tabs -->
    <ul class="nav nav-tabs justify-content-center mt-4" id="mainTabs" role="tablist">
      <li class="nav-item"><button class="nav-link active" data-bs-toggle="tab" data-bs-target="#home">Home</button></li>
      <li class="nav-item"><button class="nav-link" data-bs-toggle="tab" data-bs-target="#add">Add Skill</button></li>
      <li class="nav-item"><button class="nav-link" data-bs-toggle="tab" data-bs-target="#browse">Browse Skills</button></li>
      <li class="nav-item"><button class="nav-link" data-bs-toggle="tab" data-bs-target="#dashboard">Dashboard</button></li>
      <li class="nav-item"><button class="nav-link" data-bs-toggle="tab" data-bs-target="#courses">MiniCourse Builder</button></li>
      <li class="nav-item"><button class="nav-link" data-bs-toggle="tab" data-bs-target="#matches">Skill Matches</button></li>
<li class="nav-item"><button class="nav-link" data-bs-toggle="tab" data-bs-target="#liveSessions">Build My Exchanges</button></li>

    </ul>

    <!-- Tab Contents -->
    <div class="tab-content text-start text-light mt-3" style="overflow-y:auto; max-height:60vh;">

      <!-- Home Tab -->
      <div class="tab-pane fade show active" id="home">
```

```
        <p>Welcome to SkillSphere homepage.Here you can connect with others through our time based barter system.</p>
      </div>

      <!-- Add Skill Tab -->
      <div class="tab-pane fade" id="add">
        <h3>Add Your Skill Exchange</h3>
        <form id="skillForm" class="mb-3">
          <input type="text" id="username" class="form-control mb-2" placeholder="Your Name" required />
          <input type="text" id="teachSkill" class="form-control mb-2" placeholder="Skill You Can Teach" required />
          <input type="text" id="learnSkill" class="form-control mb-2" placeholder="Skill You Want to Learn" required />
          <input type="number" id="hours" class="form-control mb-2" placeholder="Hours You Can Teach" min="1" max="20" required />
          <button type="submit" class="btn btn-primary">Add Skill Exchange</button>
        </form>
        <div id="formMsg" class="text-success"></div>
      </div>

      <!-- Browse Skills Tab -->
      <div class="tab-pane fade" id="browse">
        <h3>The available skill exchanges </h3>
        <table class="table table-dark table-striped">
```

```
          <thead>
            <tr><th>Name</th><th>Teaches</th><th>Wants to
Learn</th><th>Hours</th><th>Action</th></tr>
          </thead>
          <tbody id="skillsTableBody"></tbody>
        </table>
      </div>

      <!-- Dashboard Tab -->
      <div class="tab-pane fade" id="dashboard">
        <h3>Your Dashboard</h3>
        <p><strong>No.of Hours to Teach:</strong> <span id="totalTeach">0</span> hrs</p>
        <p><strong>No.of Hours Requested:</strong> <span id="totalLearn">0</span> hrs</p>
        <p><strong>Balance:</strong> <span id="balance">0</span> hrs</p>
      </div>
<!-- Build My Exchanges Tab -->
<div class="tab-pane fade" id="liveSessions">
  <h3>Schedule a Live Skill Exchange Session</h3>
  <form id="sessionForm" class="mb-4">
    <input type="text" id="sessionName" class="form-control mb-2" placeholder="Your Name"
required />
    <input type="text" id="sessionSkill" class="form-control mb-2" placeholder="Skill You Want
to Exchange" required />
    <input type="datetime-local" id="sessionDateTime" class="form-control mb-2" required />
```

```
    <input type="url" id="sessionLink" class="form-control mb-2" placeholder="Live Session Link (Zoom/Meet)" required />
    <button type="submit" class="btn btn-primary">Schedule Session</button>
  </form>

  <h4>Upcoming Sessions</h4>
  <div id="upcomingSessions" class="mb-3"></div>

  <hr />
  <h4>Past Sessions</h4>
  <div id="pastSessions"></div>
</div>

        <!-- MiniCourse Builder Tab -->
        <div class="tab-pane fade" id="courses">
          <h3>Create a MiniCourse</h3>
          <form id="courseForm" class="mb-4">
            <input type="text" id="courseTitle" class="form-control mb-2" placeholder="Course Title" required />
            <input type="text" id="courseAuthor" class="form-control mb-2" placeholder="Your Name" required />
            <textarea id="courseContent" class="form-control mb-2" placeholder="Course Content" rows="4" required></textarea>
            <input type="url" id="pdfLink" class="form-control mb-2" placeholder="Optional PDF Link" />
```

```
          <input type="text" id="quizQuestion" class="form-control mb-2" placeholder="Quiz Question (optional)" />
          <input type="text" id="quizAnswer" class="form-control mb-2" placeholder="Quiz Answer (optional)" />
          <button type="submit" class="btn btn-success">Publish Course</button>
        </form>

        <h4>Published Courses</h4>
        <div id="courseList"></div>
      </div>

      <!-- Skill Matches tab -->
      <div class="tab-pane fade" id="matches">
        <h3>Here is list of smart skill matches</h3>
        <div id="matchesContainer"></div>
      </div>

    </div>
  </div>

</div>

<!-- Scripts -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
```

```
<script>
  // ---------- this is Skill Exchange Logic ----------
  let skills = JSON.parse(localStorage.getItem('skills')) || [];
  let learningRequests = JSON.parse(localStorage.getItem('learningRequests')) || [];

  function renderSkillsTable() {
    const tbody = document.getElementById('skillsTableBody');
    tbody.innerHTML = '';
    if (skills.length === 0) {
      tbody.innerHTML = '<tr><td colspan="5" class="text-center">No exchanges yet.</td></tr>';
      return;
    }

    skills.forEach((entry, i) => {
      const tr = document.createElement('tr');
      tr.innerHTML = `
        <td>${entry.username}</td>
        <td>${entry.teachSkill}</td>
        <td>${entry.learnSkill}</td>
        <td>${entry.hours}</td>
        <td><button class="btn btn-sm btn-outline-light request-btn" data-
index="${i}">Request</button></td>
      `;
      tbody.appendChild(tr);
    });
```

```
  }
function renderSessions() {
  const now = new Date();
  const upcomingContainer = document.getElementById('upcomingSessions');
  const pastContainer = document.getElementById('pastSessions');

  upcomingContainer.innerHTML = '';
  pastContainer.innerHTML = '';

  if (sessions.length === 0) {
    upcomingContainer.innerHTML = '<p class="text-muted">No upcoming sessions yet.</p>';
    pastContainer.innerHTML = '<p class="text-muted">No past sessions yet.</p>';
    return;
  }

  sessions.forEach((session) => {
    const sessionTime = new Date(session.dateTime);
    const isUpcoming = sessionTime > now;

    const sessionHTML = `
      <div class="border p-2 mb-2 rounded bg-dark">
        <p><strong>${session.name}</strong> is exchanging <strong>${session.skill}</strong></p>
        <p><strong>Time:</strong> ${sessionTime.toLocaleString()}</p>
        ${isUpcoming ? `<p><a href="${session.link}" target="_blank" class="btn btn-sm btn-
outline-info">Join Session</a></p>` : ''}
```

```
      </div>
    `;

    if (isUpcoming) {
      upcomingContainer.innerHTML += sessionHTML;
    } else {
      pastContainer.innerHTML += sessionHTML;
    }
  });

  if (!upcomingContainer.innerHTML) {
    upcomingContainer.innerHTML = '<p class="text-muted">No upcoming sessions.</p>';
  }
  if (!pastContainer.innerHTML) {
    pastContainer.innerHTML = '<p class="text-muted">No past sessions.</p>';
  }
}
    function updateDashboard() {
      const totalTeach = skills.reduce((sum, e) => sum + Number(e.hours), 0);
      const totalLearn = learningRequests.reduce((sum, e) => sum + Number(e.hours), 0);
      document.getElementById('totalTeach').textContent = totalTeach;
      document.getElementById('totalLearn').textContent = totalLearn;
      document.getElementById('balance').textContent = totalTeach - totalLearn;
    }
```

```
document.getElementById('skillForm').addEventListener('submit', function (e) {
  e.preventDefault();
  const entry = {
    username: username.value.trim(),
    teachSkill: teachSkill.value.trim(),
    learnSkill: learnSkill.value.trim(),
    hours: parseInt(hours.value)
  };
  if (!entry.username || !entry.teachSkill || !entry.learnSkill || entry.hours < 1) {
    alert('Fill all the given fields carefully.');
    return;
  }
  skills.push(entry);
  localStorage.setItem('skills', JSON.stringify(skills));
  this.reset();
  formMsg.textContent = 'the given skill is added';
  renderSkillsTable();
  updateDashboard();
  renderMatches();
  setTimeout(() => formMsg.textContent = '', 3000);
});

document.getElementById('skillsTableBody').addEventListener('click', function (e) {
  if (e.target.classList.contains('request-btn')) {
    const index = e.target.getAttribute('data-index');
```

```
        const skill = skills[index];
        const hours = prompt(`How many hours do you want to learn the skill ${skill.teachSkill}
(max ${skill.hours})`, "1");
        const hoursNum = Number(hours);
        if (hours === null || !hoursNum || hoursNum < 1 || hoursNum > skill.hours) {
          alert("Invalid input.");
          return;
        }
        learningRequests.push({ teacher: skill.username, skill: skill.teachSkill, hours: hoursNum });
        localStorage.setItem('learningRequests', JSON.stringify(learningRequests));
        updateDashboard();
        alert(`Requested ${hoursNum} hour(s) from ${skill.username}`);
      }
    });
// ---------- Live Session Exchange Logic ----------
let sessions = JSON.parse(localStorage.getItem('sessions')) || [];
    // ---------- MiniCourse Logic ----------
    let courses = JSON.parse(localStorage.getItem('courses')) || [];

    document.getElementById('courseForm').addEventListener('submit', function (e) {
      e.preventDefault();
      const newCourse = {
        title: courseTitle.value.trim(),
        author: courseAuthor.value.trim(),
        content: courseContent.value.trim(),
```

```
    pdf: pdfLink.value.trim(),
    quizQ: quizQuestion.value.trim(),
    quizA: quizAnswer.value.trim(),
    rating: []
  };
  courses.push(newCourse);
  localStorage.setItem('courses', JSON.stringify(courses));
  this.reset();
  renderCourses();
});

function renderCourses() {
  const container = document.getElementById('courseList');
  container.innerHTML = '';
  if (!courses.length) {
    container.innerHTML = '<p class="text-muted">No courses yet.</p>';
    return;
  }
  courses.forEach((course, index) => {
    const div = document.createElement('div');
    div.className = 'border p-3 mb-3 bg-dark rounded';
    div.innerHTML = `
      <h5>${course.title}</h5>
      <p><strong>Author:</strong> ${course.author}</p>
      <p>${course.content}</p>
```

```
        ${course.pdf ? `<a href="${course.pdf}" class="btn btn-sm btn-outline-light"
target="_blank">Download the given PDF</a>` : ''}
        ${course.quizQ ? `<p><strong>Quiz:</strong> ${course.quizQ}<br><em>Answer:
${course.quizA}</em></p>` : ''}
        <label>Rate this course:</label>
        <select class="form-select form-select-sm d-inline w-auto ms-2" data-index="${index}">
          <option value="">Select from given option </option>
          <option value="1">★</option>
          <option value="2">★ ★</option>
          <option value="3">★ ★ ★</option>
          <option value="4">★ ★ ★ ★</option>
          <option value="5">★ ★ ★ ★ ★</option>
        </select>
        <small class="ms-2 text-info">Avg Rating: ${averageRating(course.rating)}</small>
      `;
      container.appendChild(div);
    });
  }

  function averageRating(ratings) {
    if (!ratings.length) return 'N/A';
    const avg = ratings.reduce((a, b) => a + b, 0) / ratings.length;
    return avg.toFixed(1);
  }
```

```
  document.addEventListener('change', function (e) {
    if (e.target.tagName === 'SELECT' && e.target.hasAttribute('data-index')) {
      const i = e.target.getAttribute('data-index');
      const val = Number(e.target.value);
      if (val >= 1 && val <= 5) {
        courses[i].rating.push(val);
        localStorage.setItem('courses', JSON.stringify(courses));
        renderCourses();
      }
    }
  });
function renderSkillsTable() {
  const tbody = document.getElementById('skillsTableBody');
  tbody.innerHTML = '';
  if (skills.length === 0) {
    tbody.innerHTML = '<tr><td colspan="6" class="text-center">No exchanges yet.</td></tr>';
    return;
  }

  skills.forEach((entry, i) => {
    const tr = document.createElement('tr');
    tr.innerHTML = `
      <td>${entry.username}</td>
      <td>${entry.teachSkill}</td>
```

```
      <td>${entry.learnSkill}</td>
      <td>${entry.hours}</td>
      <td>
        <button class="btn btn-sm btn-outline-light request-btn" data-
index="${i}">Request</button>
        <button class="btn btn-sm btn-outline-warning edit-skill-btn" data-
index="${i}">Edit</button>
        <button class="btn btn-sm btn-outline-danger delete-skill-btn" data-
index="${i}">Delete</button>
      </td>
    `;
    tbody.appendChild(tr);
  });
}

document.getElementById('skillsTableBody').addEventListener('click', function (e) {
  const index = e.target.getAttribute('data-index');
  if (e.target.classList.contains('request-btn')) {
    const skill = skills[index];
    const hours = prompt(`How many hours do you want to learn the skill ${skill.teachSkill} (max
${skill.hours})`, "1");
    const hoursNum = Number(hours);
    if (hours === null || !hoursNum || hoursNum < 1 || hoursNum > skill.hours) {
      alert("Invalid input.");
      return;
```

```
    }
    learningRequests.push({ teacher: skill.username, skill: skill.teachSkill, hours: hoursNum });
    localStorage.setItem('learningRequests', JSON.stringify(learningRequests));
    updateDashboard();
    alert(`Requested ${hoursNum} hour(s) from ${skill.username}`);
  }
  else if (e.target.classList.contains('delete-skill-btn')) {
    if (confirm('Are you sure you want to delete this skill exchange?')) {
      skills.splice(index, 1);
      localStorage.setItem('skills', JSON.stringify(skills));
      renderSkillsTable();
      updateDashboard();
      renderMatches();
    }
  }
  else if (e.target.classList.contains('edit-skill-btn')) {
    const skill = skills[index];
    // Populate form with current values for editing
    username.value = skill.username;
    teachSkill.value = skill.teachSkill;
    learnSkill.value = skill.learnSkill;
    hours.value = skill.hours;

    // Change form button text
    const skillForm = document.getElementById('skillForm');
```

```
skillForm.querySelector('button[type="submit"]').textContent = 'Update Skill Exchange';

// Temporarily remove submit listener and add update listener
skillForm.removeEventListener('submit', skillFormSubmitHandler);
skillForm.addEventListener('submit', function updateHandler(e) {
  e.preventDefault();
  skill.username = username.value.trim();
  skill.teachSkill = teachSkill.value.trim();
  skill.learnSkill = learnSkill.value.trim();
  skill.hours = parseInt(hours.value);
  if (!skill.username || !skill.teachSkill || !skill.learnSkill || skill.hours < 1) {
    alert('Fill all the given fields carefully.');
    return;
  }

  localStorage.setItem('skills', JSON.stringify(skills));
  renderSkillsTable();
  updateDashboard();
  renderMatches();

  skillForm.reset();
  skillForm.querySelector('button[type="submit"]').textContent = 'Add Skill Exchange';

  // Remove this update handler and reattach the original submit handler
  skillForm.removeEventListener('submit', updateHandler);
```

```
      skillForm.addEventListener('submit', skillFormSubmitHandler);
    }, { once: true });
  }
});

   // ---------- Skill Match Logic ----------
  function renderMatches() {
  const container = document.getElementById('matchesContainer');
  container.innerHTML = '';

  if (skills.length < 2) {
    container.innerHTML = '<p class="text-muted">There is not enough data to show matches.</p>';
    return;
  }

  let matchesFound = false;

  for (let i = 0; i < skills.length; i++) {
    for (let j = i + 1; j < skills.length; j++) {
      const a = skills[i];
      const b = skills[j];

      if (
        a.teachSkill.toLowerCase() === b.learnSkill.toLowerCase() &&
```

```
        a.learnSkill.toLowerCase() === b.teachSkill.toLowerCase()
      ) {
        matchesFound = true;
        const div = document.createElement('div');
        div.className = 'border p-3 mb-3 bg-dark rounded';
        div.innerHTML = `
          <p><strong>${a.username}</strong> can teach <strong>${a.teachSkill}</strong> and
wants to learn <strong>${a.learnSkill}</strong>.</p>
          <p><strong>${b.username}</strong> can teach <strong>${b.teachSkill}</strong> and
wants to learn <strong>${b.learnSkill}</strong>.</p>
          <p class="text-success">This  is a perfect match found!</p>
        `;
        container.appendChild(div);
      }
    }
  }

  if (!matchesFound) {
    container.innerHTML = '<p class="text-warning">There are no skill matches found yet.</p>';
  }
}

    // ---------- Initial Load ----------
renderSkillsTable();
updateDashboard();
```

```
renderCourses();
renderMatches();
renderSessions();
document.getElementById('sessionForm').addEventListener('submit', function (e) {
  e.preventDefault();
  const session = {
    name: sessionName.value.trim(),
    skill: sessionSkill.value.trim(),
    dateTime: sessionDateTime.value,
    link: sessionLink.value.trim()
  };
  if (!session.name || !session.skill || !session.dateTime || !session.link) {
    alert('Please fill all fields correctly.');
    return;
  }
  sessions.push(session);
  localStorage.setItem('sessions', JSON.stringify(sessions));
  this.reset();
  renderSessions();
});
  </script>
</body>
</html>
```

2.5 FINAL PROJECT –2

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Ekamjot's Student Dashboard</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet"/>
  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap"
rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap"
rel="stylesheet">

<style>

body { margin: 0; font-family: 'Inter', sans-serif; color: #fff; background:
url('https://images.unsplash.com/photo-1483058712412-
4245e9b90334?fm=jpg&ixid=M3wxMjA3fDB8MHxzZWFyY2h8Nnx8d29ya3NwYWNlfGVufD
B8fDB8fHww&ixlib=rb-4.1.0&q=60&w=3000') no-repeat center center fixed; background-size:
cover; min-height: 100vh; position: relative; }

/* Optional: semi-transparent dark overlay for contrast */ body::before { content: ''; position: fixed;
top: 0; left: 0; width: 100vw; height: 100vh; background: rgba(0, 0, 0, 0.5); /* dark overlay */ z-
index: -1; }

.session-row.current { font-weight: bold; font-size: 1.2em; /* Remove yellow background */
background-color: transparent; /* or use a subtle color like #f0f0f0 */ padding: 10px; border-radius:
5px; /* Optional subtle border to highlight */ border: 1px solid #ccc; }
```

```
.session-row.current div { margin-bottom: 6px; }

.session-row.current span.label { color: #ffd740; margin-right: 8px; font-weight: 600; }

.container { padding-top: 30px; }

.marquee { width: 100%; overflow: hidden; white-space: nowrap; padding: 15px 0; font-size: 1.5rem; font-weight: 700; color: #ffffffcc; text-shadow: 0 0 6px rgba(0, 0, 0, 0.5); border-bottom: 1px solid rgba(255,255,255,0.2); }

.marquee span { display: inline-block; padding-left: 100%; animation: marquee-left 18s linear infinite; }

@keyframes marquee-left { 0% { transform: translateX(0); } 100% { transform: translateX(-100%); } }

footer.marquee { border-top: 1px solid rgba(255,255,255,0.2); font-size: 1rem; padding: 12px 0; color: #ffffffcc; text-shadow: 0 0 6px rgba(0, 0, 0, 0.4); text-align: center; }

.glass-card {
    background: rgba(0, 0, 0, 0.55);
    border-radius: 16px;
    padding: 25px 30px;
    box-shadow: 0 4px 20px rgba(0,0,0,0.4);
    backdrop-filter: blur(12px);
    margin-bottom: 30px;
```

```
}

h3 {
  font-size: 1.8rem;
  font-weight: 700;
  margin-bottom: 20px;
  color: #ffffff;
  text-shadow: 0 0 6px rgba(0,0,0,0.4);
}

.nav-tabs {
  border-bottom: 2px solid rgba(255, 255, 255, 0.3);
}

.nav-tabs .nav-link {
  color: #ffffffcc;
  font-weight: 600;
```

```
    font-size: 1.1rem;
  }

  .nav-tabs .nav-link:hover {
    color: #fff;
    background-color: rgba(255,255,255,0.1);
    border-radius: 8px 8px 0 0;
  }

  .nav-tabs .nav-link.active {
    background-color: rgba(255,255,255,0.2);
    color: #fff;

 border-radius: 10px 10px 0 0;
    font-weight: 700;
  }
</style>
```

```
</head>

<body>

  <!--  Top Banner -->

  <div class="marquee"><span  id="dateBanner"></span></div>

  <div class="container mt-4">

    <!--  Main Tabs -->

    <ul class="nav  nav-tabs mb-3" id="mainTabs">

      <li class="nav-item"><a  class="nav-link  active" data-bs-toggle="tab"
href="#welcome">Welcome</a></li>

      <li class="nav-item"><a  class="nav-link"  data-bs-toggle="tab"
href="#subjects">Subjects</a></li>

      <li class="nav-item"><a  class="nav-link"  data-bs-toggle="tab"
href="#timetable">Timetable</a></li>

      <li class="nav-item"><a  class="nav-link"  data-bs-toggle="tab"
href="#attendance">Attendance</a></li>

    </ul> <div class="tab-content">

      <!--  Welcome Tab -->

      <div class="tab-pane fade show active" id="welcome">
```

```
    <div class="glass-card">
      <h3>Welcome!</h3>
      <p>Hello , this dashboard helps you manage your subjects, keep track of your timetable, and monitor your attendance — all in one elegant place!</p>
    </div>
  </div>

  <!-- Subjects Tab -->
  <div class="tab-pane fade" id="subjects">
    <div class="glass-card">
      <h3>Your Subjects</h3>
      <ul class="list-group">
        <li class="list-group-item">Artificial Intelligence</li>
        <li class="list-group-item">Database Management Systems</li>
        <li class="list-group-item">FLAT</li>
        <li class="list-group-item">DAA</li>
        <li class="list-group-item">Statistics for Data Science</li>
```

```
      <li class="list-group-item">Constitution of India</li>
      <li class="list-group-item">Artificial Intelligence Lab</li>
      <li class="list-group-item">Database Management Lab</li>
      <li class="list-group-item">DAA Lab</li>
      <li class="list-group-item">Mentoring Class</li>
    </ul>
  </div>
</div>

<!-- Timetable Tab -->
<div class="tab-pane fade" id="timetable">
  <div class="glass-card">
    <h3>Timetable</h3>
    <div id="timetableContainer"></div>

    <!-- Extra Timetable Views -->
    <ul class="nav nav-tabs mt-4" id="extraTimetableTabs" role="tablist">
```

```
            <li class="nav-item">
              <button class="nav-link active" id="day-tab" data-bs-toggle="tab" data-bs-target="#day" type="button" onclick="showFullDay()">Full  Day</button>
            </li>
            <li class="nav-item">
              <button class="nav-link"  id="week-tab" data-bs-toggle="tab" data-bs-target="#week" type="button" onclick="showFullWeek()">Full  Week</button>
            </li>
            <li class="nav-item">
              <button class="nav-link"  id="next-tab" data-bs-toggle="tab" data-bs-target="#next" type="button" onclick="showNextPeriod()">Next Period</button>
            </li>
          </ul>

          <div class="tab-content p-3 border border-top-0" id="extraTimetableContent">
            <div class="tab-pane fade show active" id="day"></div>
            <div class="tab-pane fade" id="week"></div>
            <div class="tab-pane fade" id="next"></div>
```

```
        </div>
      </div>
    </div>

    <!-- 🔸 Attendance Tab -->
    <div class="tab-pane fade" id="attendance">
      <div class="glass-card">
        <h3>Attendance</h3>

        <!-- Subject Tab Headers -->
        <ul class="nav nav-tabs" id="subjectTabs" role="tablist">
  <li class="nav-item">
    <button class="nav-link active" data-bs-toggle="tab" data-bs-target="#artificial-intelligence"
type="button">
      Artificial Intelligence
    </button>
  </li>
```

```
  <li class="nav-item">
    <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#database-management-systems" type="button">
      Database Management Systems
    </button>
  </li> <li class="nav-item">
    <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#flat" type="button">
      FLAT
    </button>
  </li>
  <li class="nav-item">
    <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#daa" type="button">
      DAA
    </button>
  </li>
  <li class="nav-item">
    <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#statistics-for-data-science" type="button">
```

```
    Statistics for Data Science
   </button>
  </li>
  <li class="nav-item">
   <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#constitution-of-india"
type="button">
    Constitution of India
   </button>
  </li>
  <li class="nav-item">
   <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#artificial-intelligence-lab"
type="button">
    Artificial Intelligence Lab
   </button>
  </li>
  <li class="nav-item">
   <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#database-management-lab"
type="button">
```

```
    Database Management Lab
   </button>
  </li>
  <li class="nav-item">
   <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#daa-lab" type="button">
    DAA Lab
   </button>
  </li>
  <li class="nav-item">
   <button class="nav-link"  data-bs-toggle="tab" data-bs-target="#mentoring-class"
type="button">
    Mentoring Class
   </button>
  </li>
</ul>
<!-- Subject Tab Content Panes -->
      <div class="tab-content mt-3">
```

```
  <div class="tab-pane fade show active" id="artificial-intelligence"></div>

  <div class="tab-pane fade" id="database-management-systems"></div>

  <div class="tab-pane fade" id="flat"></div>

  <div class="tab-pane fade" id="daa"></div>

  <div class="tab-pane fade" id="statistics-for-data-science"></div>

  <div class="tab-pane fade" id="constitution-of-india"></div>

  <div class="tab-pane fade" id="artificial-intelligence-lab"></div>

  <div class="tab-pane fade" id="database-management-lab"></div>

  <div class="tab-pane fade" id="daa-lab"></div>

  <div class="tab-pane fade" id="mentoring-class"></div>

</div>



      </div>

    </div>

  </div>

 </div>
```

```
  <!--  🔹 Footer -->

  <div class="marquee"><span>Made  by Ekamjot Kaur </span></div>



</body>

<!-- JavaScript -->

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

<script>

const timetable = {

  "Monday": [

    ["08:30", "Statistics for Data Science", "S218"],

    ["09:30", "DAA", "G17"],

    ["10:30", "Database Management Lab", "SL-II Lab"],

    ["11:30", "Database Management Lab", "SL-II Lab"],

    ["12:30", "No class", "No venue"],

    ["13:30", "Statistics for Data Science", "S215"],

    ["14:30", "No class", "No venue"]
```

```
  ],
  "Tuesday": [
    ["08:30", "No class", "No venue"],
    ["09:30", "Database Management Systems", "S218"],
    ["10:30", "DAA", "S217"],
    ["11:30", "No class", "No venue"],
    ["12:30", "Constitution of India", "F107"],
    ["13:30", "Artificial Intelligence", "G18"],
    ["14:30", "FLAT", "S217"]
  ],
"Wednesday": [
    ["08:30", "Artificial Intelligence", "S218"],
    ["09:30", "Statistics for Data Science", "S218"],
    ["10:30", "Artificial Intelligence Lab", "OS Lab"],
    ["11:30", "Artificial Intelligence Lab", "OS Lab"],
    ["12:30", "No class", "No venue"],
    ["13:30", "FLAT", "S218"],
```

```
  ["14:30", "Mentoring Class", "G18"]

],

"Thursday": [

  ["08:30", "Database Management Systems", "G17"],

  ["09:30", "FLAT", "G17"],

  ["10:30", "Artificial Intelligence", "S219"],

  ["11:30", "No class", "No venue"],

  ["12:30", "Constitution of India", "F107"],

  ["13:30", "DAA", "G18"],

  ["14:30", "No class", "No venue"]

],

"Friday": [

  ["08:30", "DAA", "S216"],

  ["09:30", "Statistics for Data Science", "S216"],

  ["10:30", "DAA Lab", "SL-II Lab"],

  ["11:30", "DAA Lab", "SL-II Lab"],

  ["12:30", "No class", "No venue"],
```

```
    ["13:30", "Database Management Systems", "G17"],
    ["14:30", "No class", "No venue"]
  ]
};

 function toMinutes(hm){
   const [h, m] = hm.split(":").map(Number);
   return h * 60 + m;
 }

 // Mark attendance or cancellation and update localStorage
 function mark(id, subject, status){
   localStorage.setItem(id, status);

   // Update attendance only for present/absent, not for cancelled or empty
   if(status === "present" || status === "absent"){
     const totalKey = T-${subject};
```

```
    const attKey = A-${subject};
    let total = Number(localStorage.getItem(totalKey)) || 0;
    let att = Number(localStorage.getItem(attKey)) || 0;

    // Only count once per session
    if(!localStorage.getItem(id + "-counted")){
      total++;
      if(status === "present") att++;
      localStorage.setItem(totalKey, total);
      localStorage.setItem(attKey, att);
      localStorage.setItem(id + "-counted", "yes");
    }
  }

  renderAttendance();
 }
function formatTime(h, m){
```

```
  let ampm = h >= 12 ? "p.m." : "a.m.";
  let hour12 = h % 12;
  if(hour12 === 0) hour12 = 12;
  return ${hour12}:${m.toString().padStart(2,'0')} ${ampm};
}

function showFullDay(){
  const now = new Date();
  const day = now.toLocaleDateString('en-US', { weekday: 'long' });
  const cont = document.getElementById('day');
  if(!timetable[day]){
    cont.innerHTML = <p>No classes scheduled for today (${day}).</p>;
    return;
  }
  let html = <ul class="list-group">;
  timetable[day].forEach(([time, subject, room])=>{
    let [h,m] = time.split(':').map(Number);
```

```
    let endH = h + 1;

    let timeStr = ${formatTime(h,m)}  - ${formatTime(endH,m)};

    html += <li class="list-group-
item"><strong>${timeStr}</strong><br>${subject}<br><em>${room}</em></li>;

  });

  html += </ul>;

  cont.innerHTML = html;

}



function showFullWeek(){

  const cont = document.getElementById('week');

  let html = ``;

  const days = ["Monday","Tuesday","Wednesday","Thursday","Friday"];

  days.forEach(day=>{

    if(!timetable[day]){

      html += <h6>${day}</h6><p>No classes scheduled.</p>;

    } else {
```

```
html += <h6>${day}</h6><ul class="list-group  mb-3">;

      timetable[day].forEach(([time, subject, room])=>{

        let [h,m] = time.split(':').map(Number);

        let endH = h + 1;

        let timeStr = ${formatTime(h,m)}  - ${formatTime(endH,m)};

        html += <li class="list-group-
item"><strong>${timeStr}</strong><br>${subject}<br><em>${room}</em></li>;

      });

      html += </ul>;

    }

  });

  cont.innerHTML = html;

}



function showNextPeriod(){

  const now = new Date();

  const day = now.toLocaleDateString('en-US', { weekday: 'long' });
```

```
const minutesNow = now.getHours() * 60 + now.getMinutes();

const cont = document.getElementById('next');



if(!timetable[day]){

  cont.innerHTML = <p>No classes today (${day}).</p>;

  return;

}



const nextSession = timetable[day].find(([time]) => toMinutes(time) > minutesNow);



if(!nextSession){

  cont.innerHTML = <p>No upcoming periods today.</p>;

  return;

}



const [time, subject, room] = nextSession;

let [h,m] = time.split(":").map(Number);
```

```
  let endH = h + 1;

  let timeStr = ${formatTime(h,m)}  - ${formatTime(endH,m)};



  cont.innerHTML = `

    <div class="session-row">

      <div><span class="label">Time:</span>  ${timeStr}</div>

      <div><span class="label">Subject:</span>  ${subject}</div>

      <div><span class="label">Venue:</span>  ${room}</div>

    </div>

  `;

}




  function renderTimetable(){

    const cont = document.getElementById("timetableContainer");

    cont.innerHTML = ""; // Clear old content
```

```
const now = new Date();

const day = now.toLocaleDateString('en-US', { weekday: 'long' });

const minutesNow = now.getHours() * 60 + now.getMinutes();



if (!timetable[day]) {

  cont.innerHTML = <p>No classes today.</p>;

  return;

}



const current = timetable[day].find(([time]) => {

  const start = toMinutes(time);

  return minutesNow >= start && minutesNow < start + 60;

});



if (current) {

  const [time, subject, room] = current;

  const id = ${day}-${time};
```

```
const prev = localStorage.getItem(id) || "";



// Calculate end time string (add 60 mins)

let [h,m] = time.split(":").map(Number);

let endH = h + 1;



// AM/PM formatting helper

function formatHour(hr){

  if(hr === 0) return 12;

  if(hr > 12) return hr - 12;

  return hr;

}

function getAMPM(hr){

  return hr >= 12 ? "p.m." : "a.m.";

}



const startHourFormatted = formatHour(h);
```

```
    const endHourFormatted = formatHour(endH);

    const ampmStart = getAMPM(h);

    const ampmEnd = getAMPM(endH);



    // If AM/PM different, show both, else just one at end

    let timeStr = ${startHourFormatted}:${m.toString().padStart(2,'0')} -
${endHourFormatted}:${m.toString().padStart(2,'0')} ${ampmEnd};

    if(ampmStart !== ampmEnd){

      timeStr = ${startHourFormatted}:${m.toString().padStart(2,'0')} ${ampmStart} -
${endHourFormatted}:${m.toString().padStart(2,'0')} ${ampmEnd};

    }

cont.innerHTML = `

 <div class="session-row  current" id="${id}">

  <div><span class="label">Time:</span>  ${timeStr}</div>

  <div><span class="label">Subject:</span>  ${subject}</div>

  <div><span class="label">Venue:</span>  ${room}</div>

   </div>`;
```

```
  } else {
    cont.innerHTML = <p>No class currently happening.</p>;
  }
}function autoUpdateTotalLectures() {
const now = new Date();
const day = now.toLocaleDateString('en-US', { weekday: 'long' });
if (!timetable[day]) return;

const currentMinutes = now.getHours() * 60 + now.getMinutes();

timetable[day].forEach(([time, subject]) => {
  const startMinutes = toMinutes(time);
  const id = ${day}-${time};
  const totalKey = T-${subject};

  // If time has passed (class is over) and wasn't already counted
  if (currentMinutes >= startMinutes + 60 && !localStorage.getItem(id + "-counted-auto")) {
```

```
      // Only count if subject is not "No class"
      if (subject !== "No class") {
        let total = Number(localStorage.getItem(totalKey)) || 0;
        total++;
        localStorage.setItem(totalKey, total);
        localStorage.setItem(id + "-counted-auto", "yes");
        renderAttendance(); // Optional: update UI if user is on attendance tab
      }
    }
  });
}


  function renderAttendance(){
    const cont = document.getElementById("attendanceStats");
    cont.innerHTML = "";
    const subjects = new Set(Object.values(timetable).flat().map(s=>s[1]));
```

```
    subjects.forEach(subject => {
      const total = Number(localStorage.getItem(T-${subject})) || 0;
      const att = Number(localStorage.getItem(A-${subject})) || 0;
      const pct = total ? ((att / total) * 100).toFixed(1) : 0;
      cont.innerHTML += <div class="attendance-stat"><strong>${subject}:</strong>
${att}/${total} (${pct}%)</div>;
    });
  } // Constants
const startDate = new Date(2025, 6, 21); // July is month 6 (zero-based)
const subjects = [
  "Artificial Intelligence",
  "Database Management Systems",
  "FLAT",
  "DAA",
  "Statistics for Data Science",
  "Constitution of India",
  "Artificial Intelligence Lab",
```

```
  "Database Management Lab",
  "DAA Lab",
  "Mentoring Class"
];

// Initialize attendance storage - no default present/absent, just create keys as "pending"
function initAttendance() {
  const today = new Date();
  for (let d = new Date(startDate); d <= today; d.setDate(d.getDate() + 1)) {
    const dayName = d.toLocaleDateString('en-US', { weekday: 'long' });
    const dateKey = d.toISOString().slice(0, 10);
    if (!timetable[dayName]) continue;
    timetable[dayName].forEach(([time, subject]) => {
      if (!subjects.includes(subject)) return;
      const id = ${dateKey}-${time}-${subject};

      // If attendance status not set, mark as pending (empty string)
```

```
      if (localStorage.getItem(id) === null) {
        localStorage.setItem(id, "pending");
      }

      // Default cancelled: false if not set
      if (localStorage.getItem(id + "-cancelled") === null) {
        localStorage.setItem(id + "-cancelled", "false");
      }
    });
  }
}

// Save attendance status when user marks present/absent
function setAttendance(id, status) {
  localStorage.setItem(id, status);
  refreshAll();
```

```
} // Toggle cancel checkbox handler function toggleCancel(id, checkbox) { localStorage.setItem(id + "-cancelled", checkbox.checked ? "true" : "false"); refreshAll(); } // Render attendance UI for a subject with mark options + cancel checkbox function renderSubject(subject) { const pane = document.getElementById(normalize(subject)); if (!pane) return; pane.innerHTML = '';

const today = new Date(); let attended = 0; let total = 0;

let tableHTML = <table class="table table-bordered table-striped"> <thead> <tr> <th>Date & Time</th> <th>Attendance</th> <th>Cancelled</th> </tr> </thead> <tbody> ;

for (let d = new Date(startDate); d <= today; d.setDate(d.getDate() + 1)) { const dayName = d.toLocaleDateString('en-US', { weekday: 'long' }); const dateKey = d.toISOString().slice(0, 10); if (!timetable[dayName]) continue;

timetable[dayName].forEach(([time, subj]) => {
  if (subj !== subject) return;

  const id = ${dateKey}-${time}-${subject};
  let status = localStorage.getItem(id) || "pending";
  const cancelled = localStorage.getItem(id + "-cancelled") === "true";

  // If attendance pending and not cancelled, mark as absent by default
  if (status === "pending" && !cancelled) {
  status = "absent";
    localStorage.setItem(id, status);
  }
```

```
  if (!cancelled) {
    total++;
    if (status === "present") attended++;
  }

  // Attendance buttons with current status highlighted
  const presentChecked = status === "present" ? "btn-primary" : "btn-outline-primary";
  const absentChecked = status === "absent" ? "btn-danger" : "btn-outline-danger";

  tableHTML += `
    <tr>
      <td>${dateKey} @ ${time}</td>
      <td>
        <div class="btn-group btn-group-sm" role="group">
          <button type="button" class="btn ${presentChecked}" onclick="setAttendance('${id}',
'present')">Present</button>
          <button type="button" class="btn ${absentChecked}" onclick="setAttendance('${id}',
'absent')">Absent</button>
        </div>
      </td>
      <td class="text-center">
        <input type="checkbox" ${cancelled ? "checked" : ""}
          onchange="toggleCancel('${id}', this)" title="Mark class as cancelled">
      </td>
```

```
    </tr>
  `;
});



}

tableHTML += ;

// Summary table const summaryHTML = <table class="table table-sm table-bordered w-auto mt-3"> <thead><tr><th>Lectures Attended</th><th>Total Lectures</th></tr></thead> <tbody><tr><td>${attended}</td><td>${total}</td></tr></tbody> </table> ;

pane.innerHTML = tableHTML + summaryHTML; }

// Helpers and other code remain unchanged...

// Call on page load initAttendance(); refreshAll(); setInterval(refreshAll, 60601000); // hourly refresh

// Helper: normalize subject into valid ID (dashes/lowercase) function normalize(str) { return str.toLowerCase() .replace(/ & /g, '-') .replace(/ and /g, '-') .replace(/\s+/g, '-') .replace(/[^a-z0-9-]/g, ''); }

// 3️⃣ Refresh all subjects function refreshAll() { subjects.forEach(s => renderSubject(s)); }

// ⏱️ Initialization initAttendance(); refreshAll(); setInterval(refreshAll, 60601000); // every hour
```

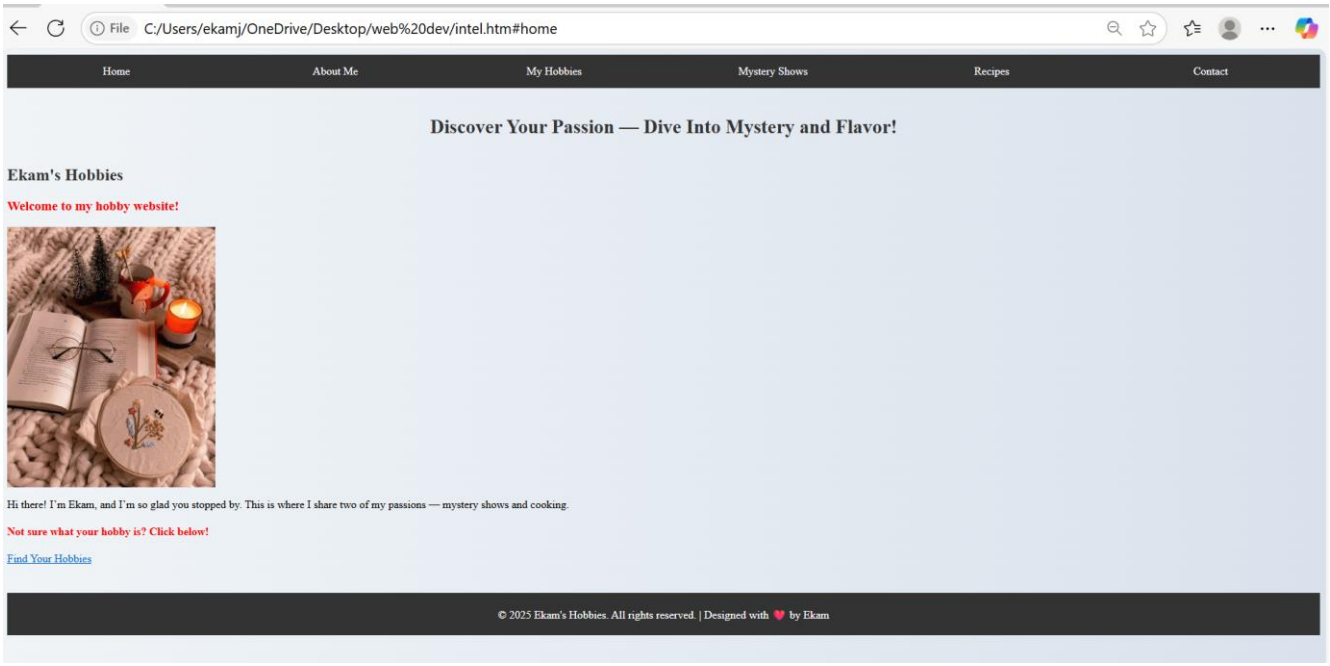
why does this font type is chnging in notepad

```
function bannerDate(){ document.getElementById("dateBanner").textContent = new Date().toLocaleDateString(undefined, { weekday:'long', month:'short', day:'numeric', year:'numeric' }); }

// Initialize on page load bannerDate(); renderTimetable(); renderAttendance(); setInterval(renderTimetable, 60000); // Refresh every minute

</script>
```

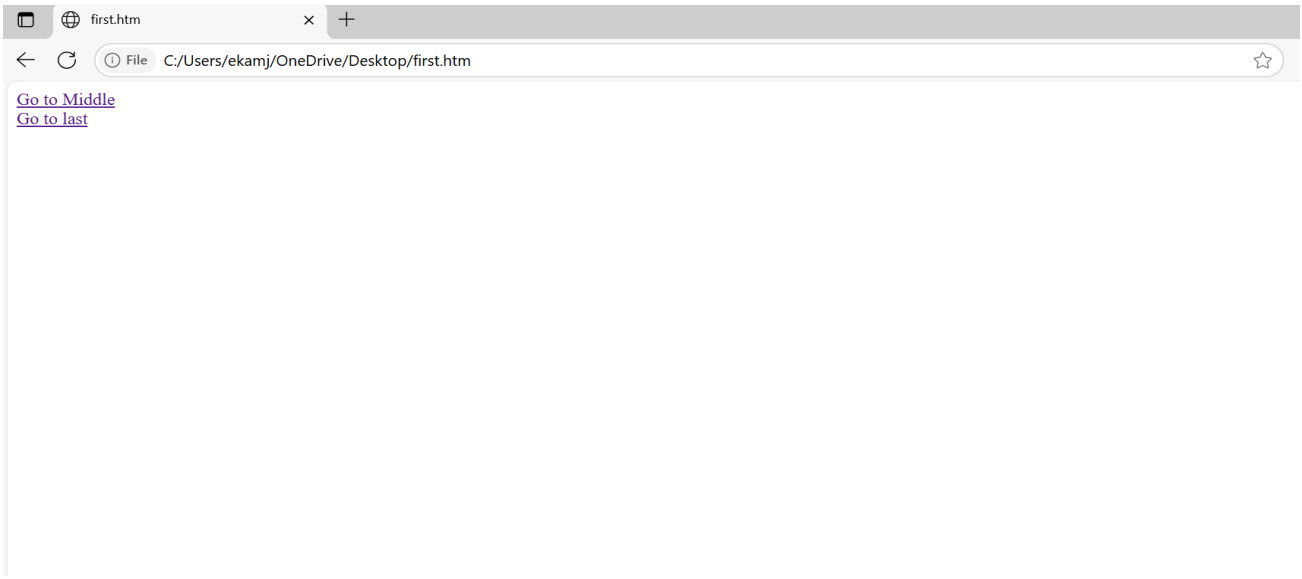
Chapter 3 RESULTS AND DISCUSSIONS

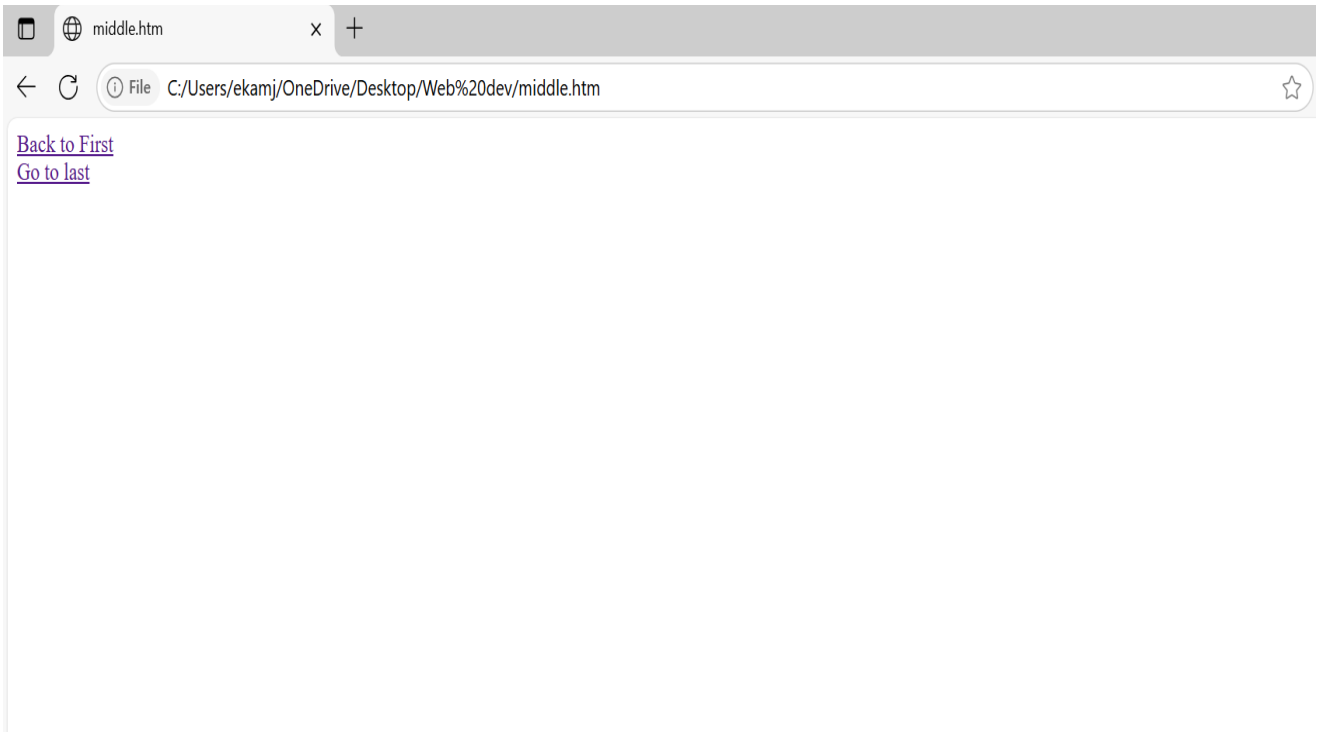
3.1 HTML PROJECTS

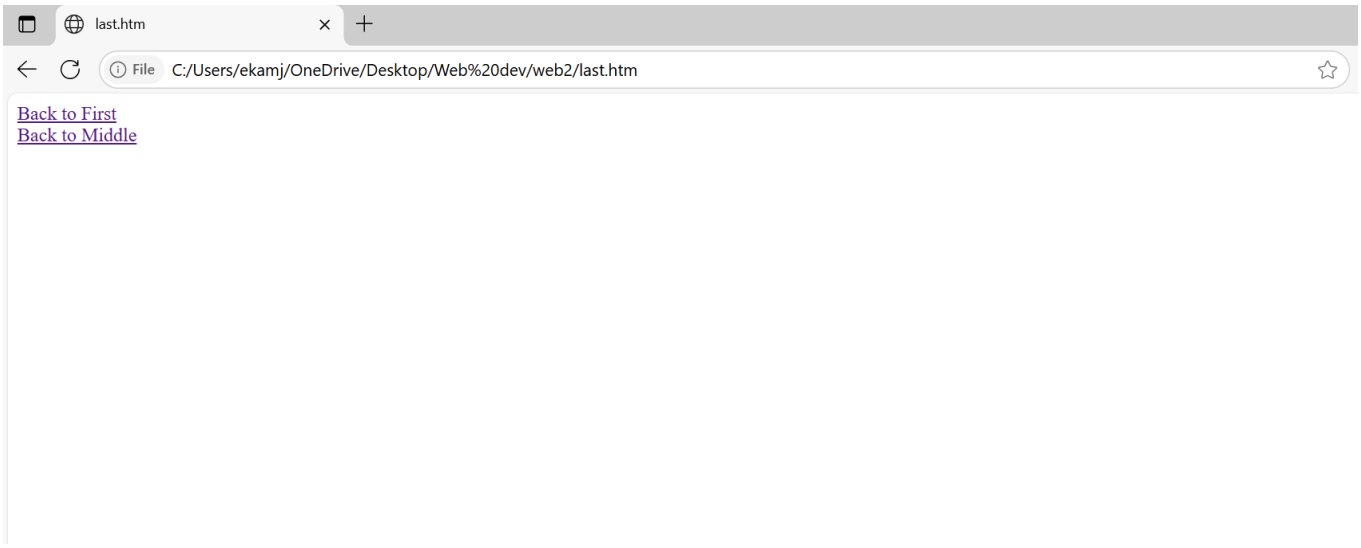
3.1.1 Html project –1



3.1.2 HTML PROJECT –2

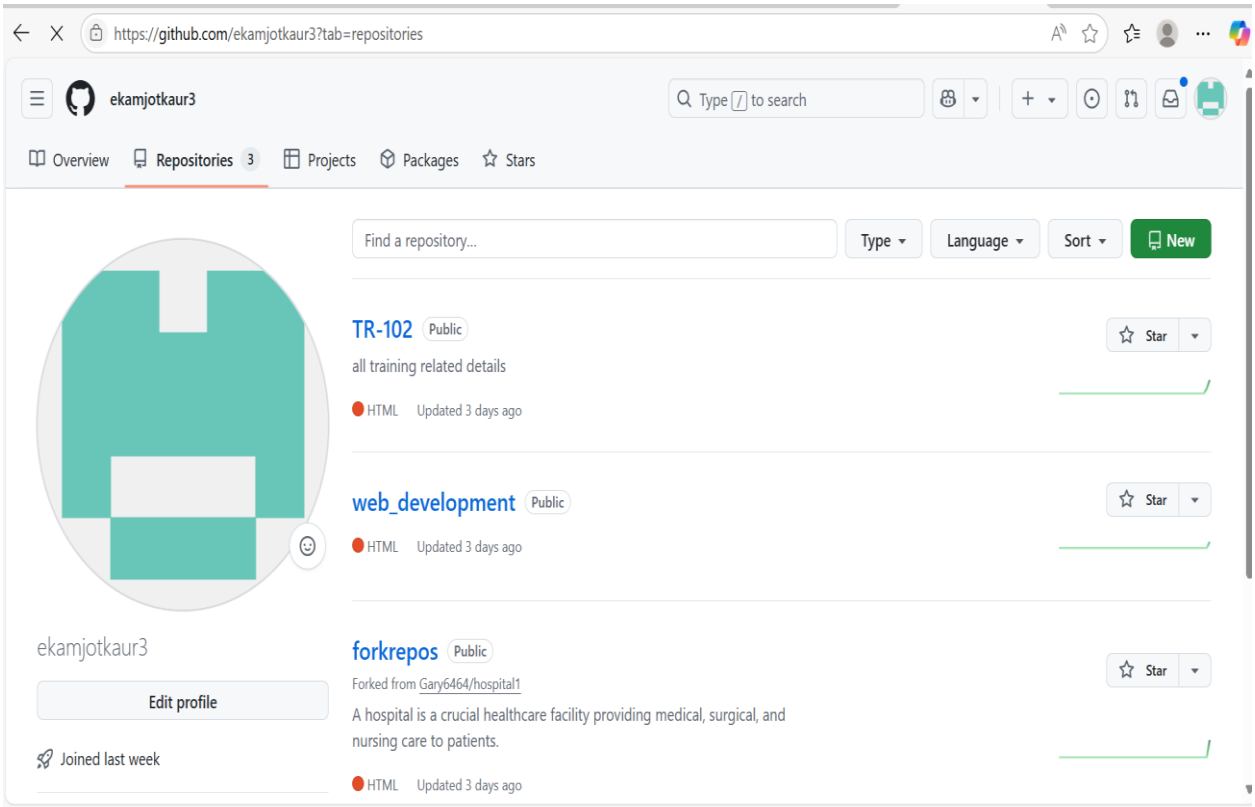




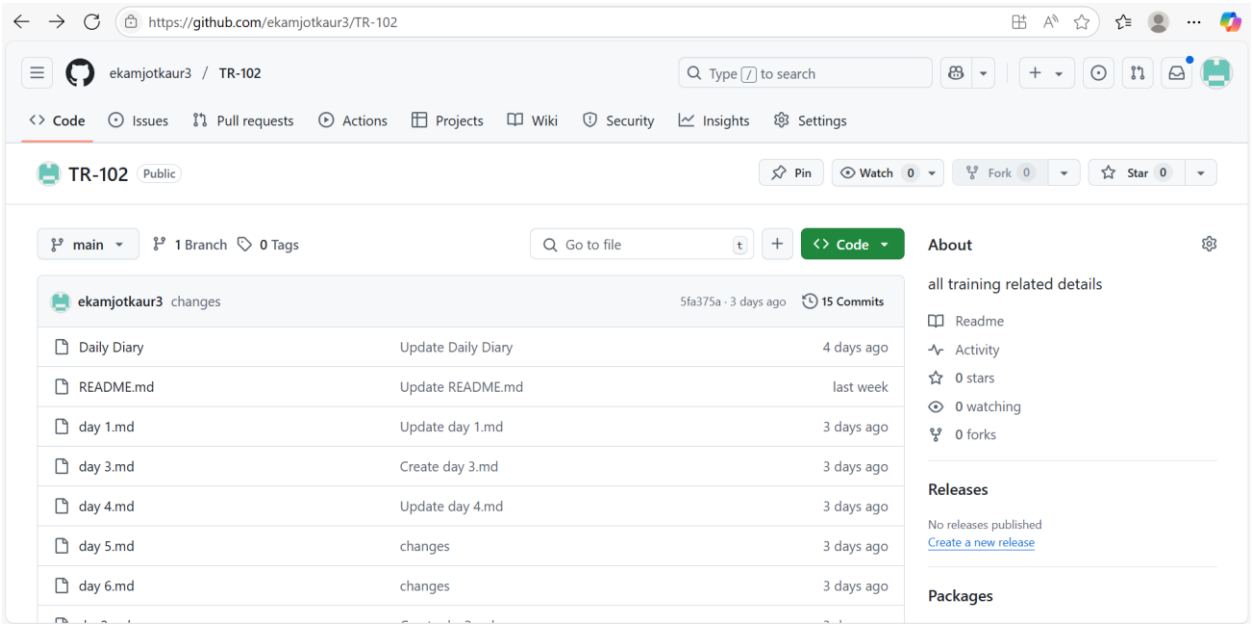


3.2 Github results

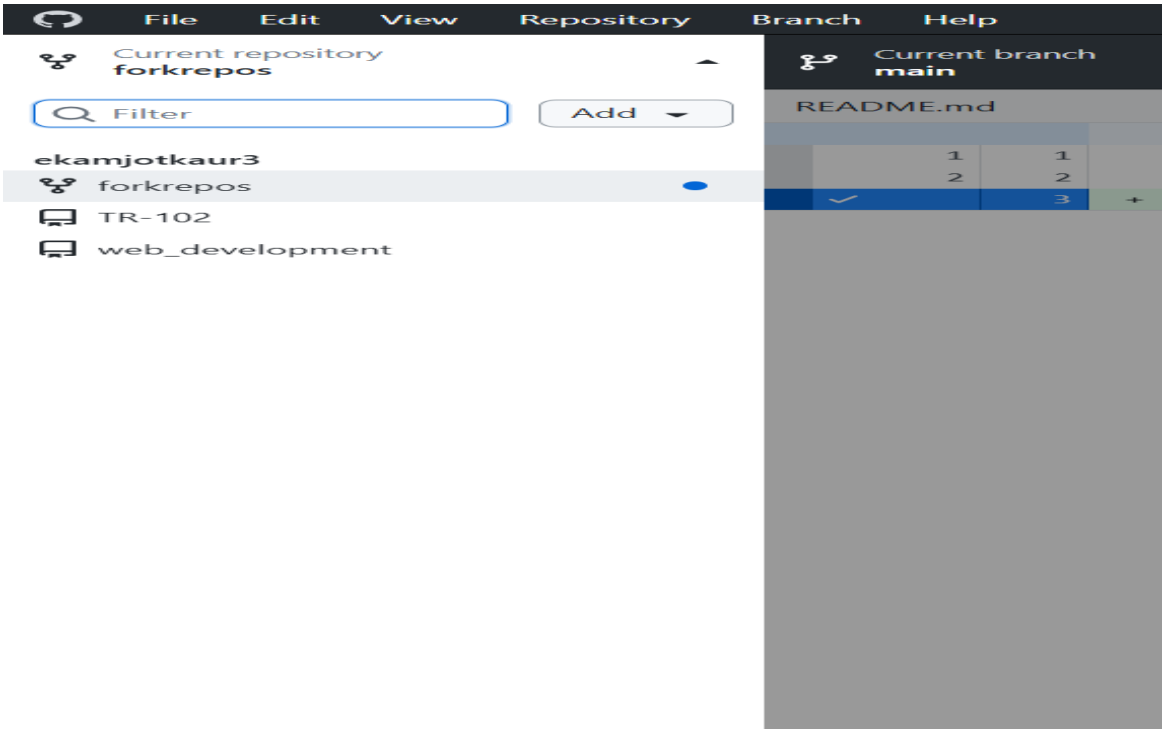
3.2.1 Creating repositories



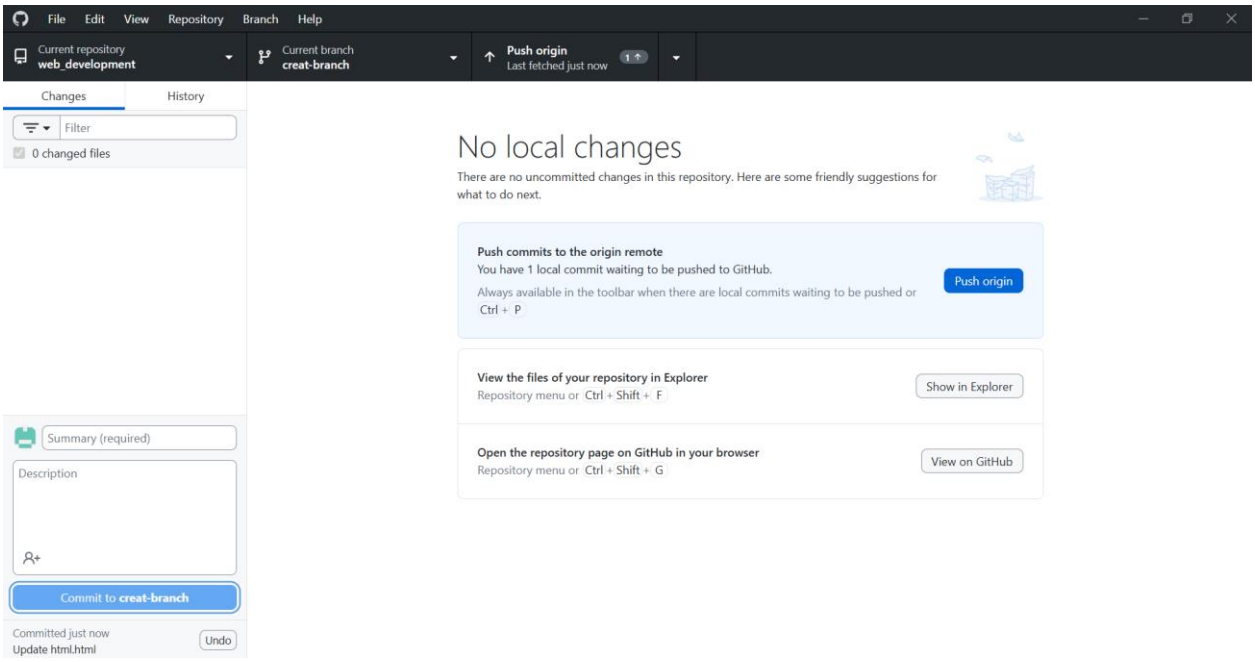
3.2.2 Adding Files



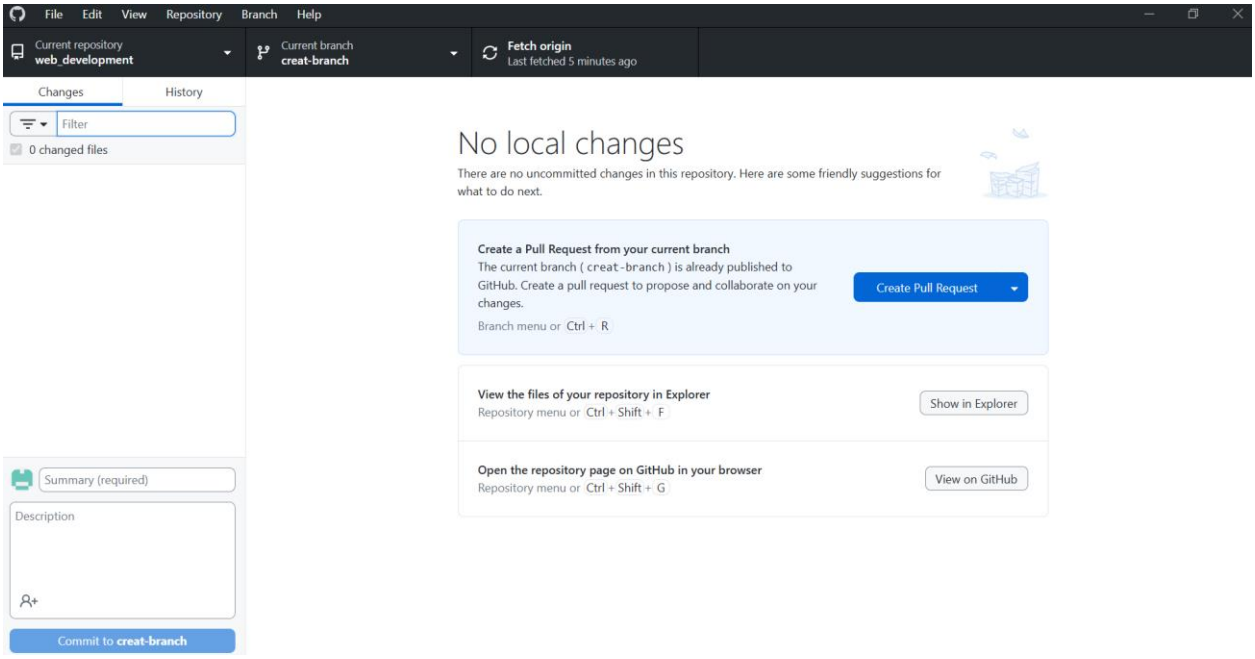
3.2.3 Cloning repository in Github Dekstop



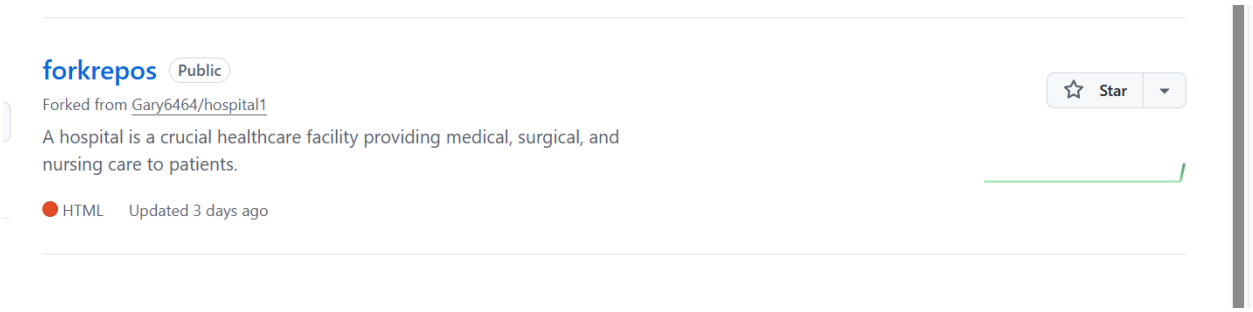
3.2.4 Pushing Changes In Github Dekstop



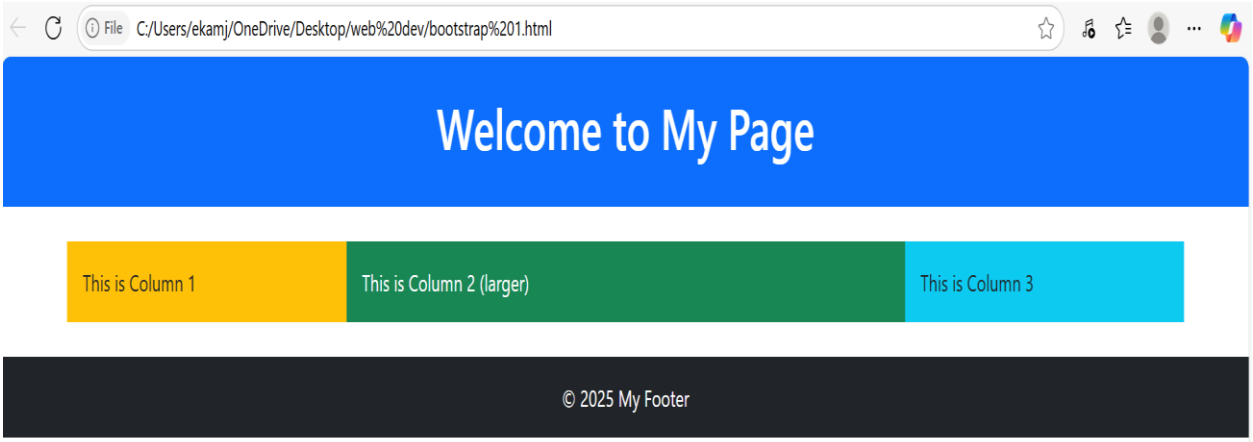
3.2.5 Pulling changes in GitHub dekstop



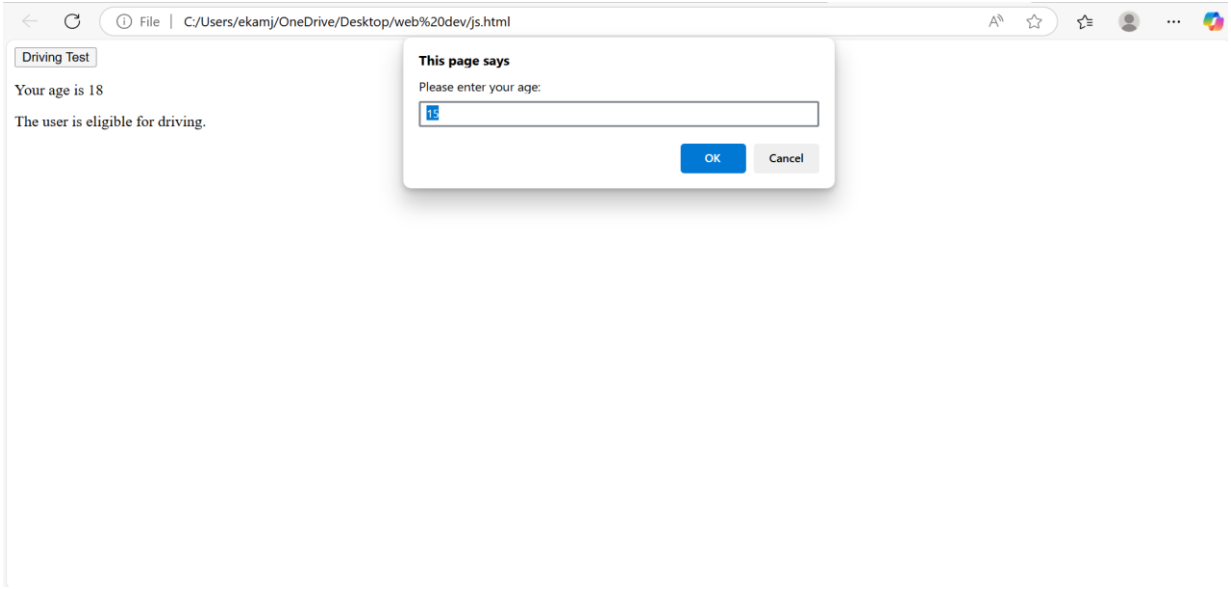
3.2.6 Forking repository



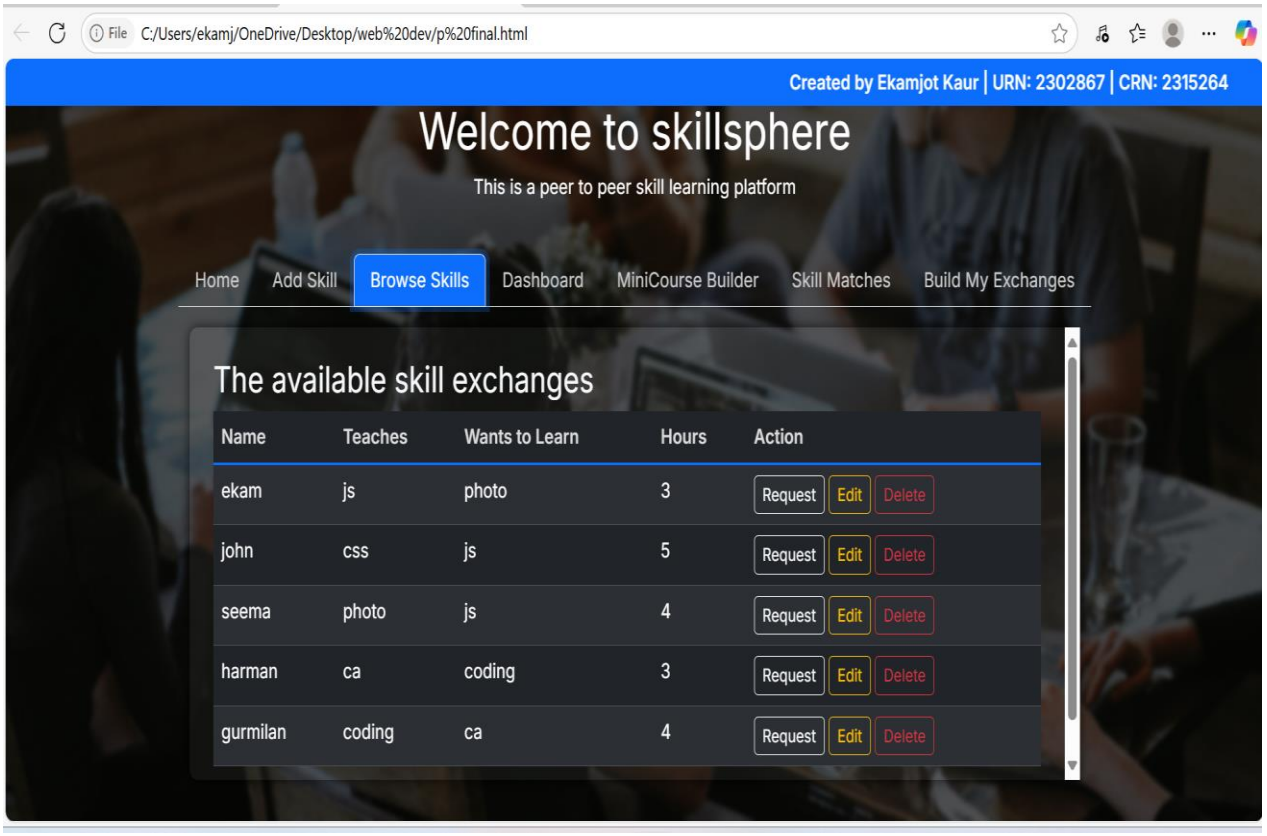
3.3 Bootstrap result

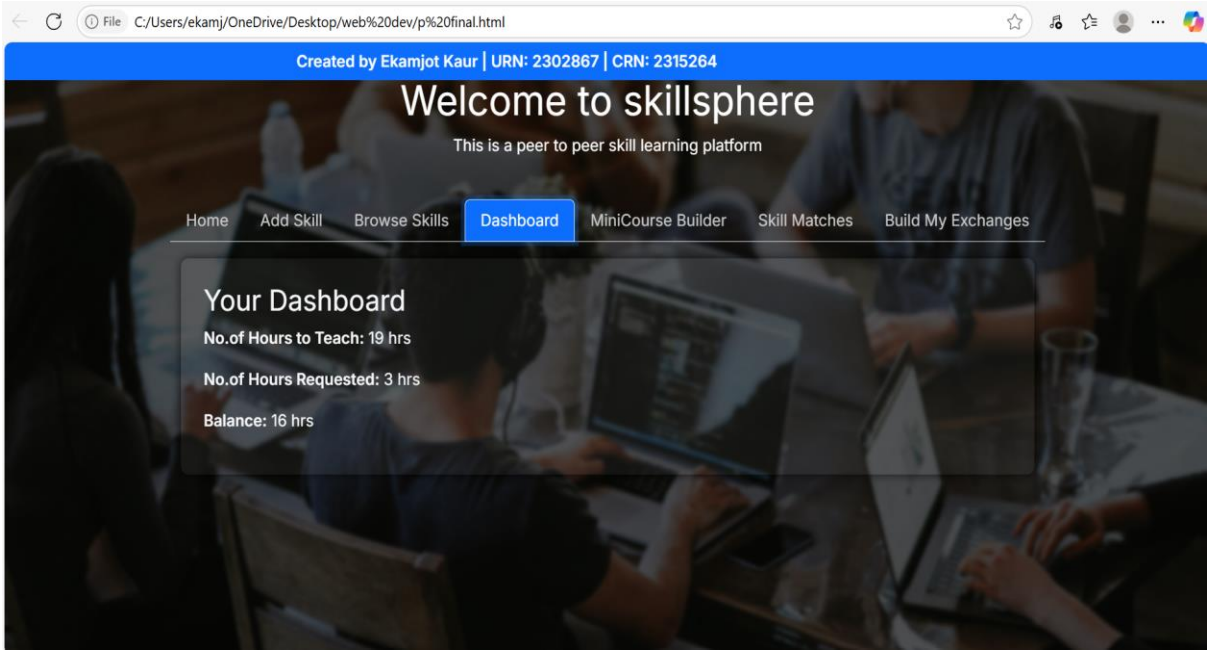


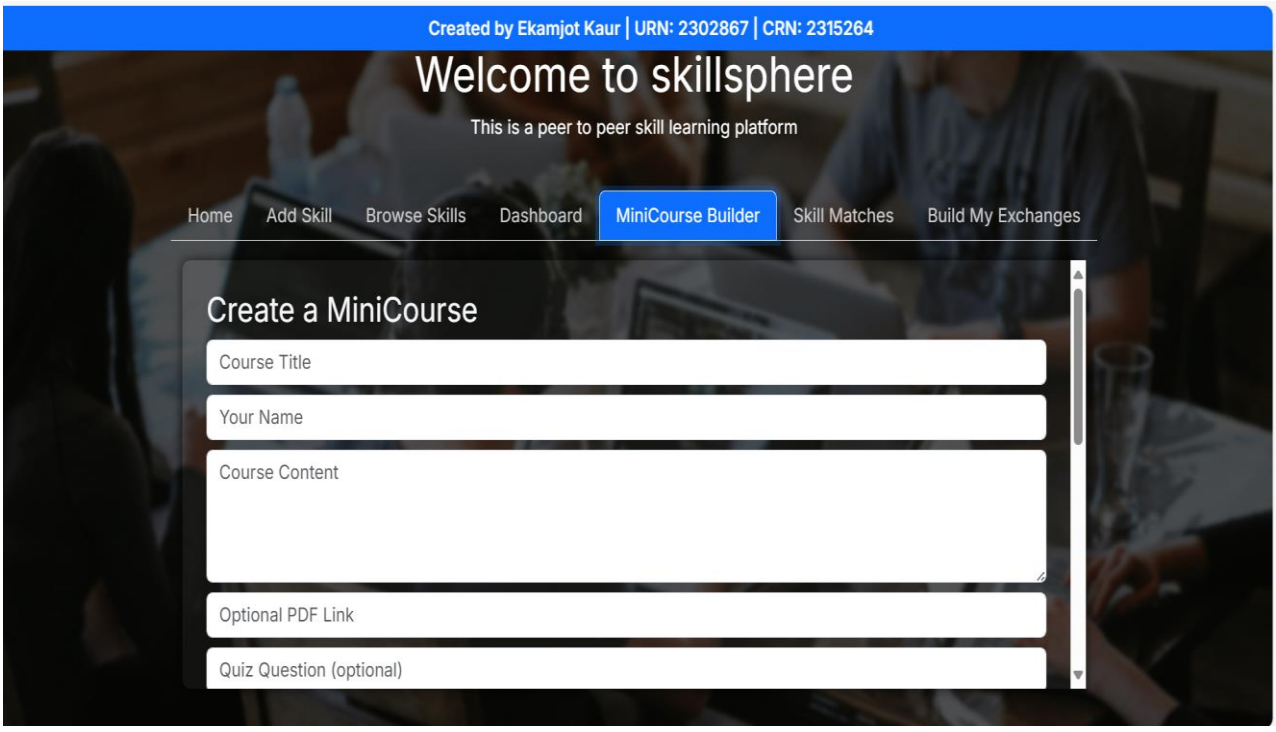
3.4 JAVASCRIPT RESULT

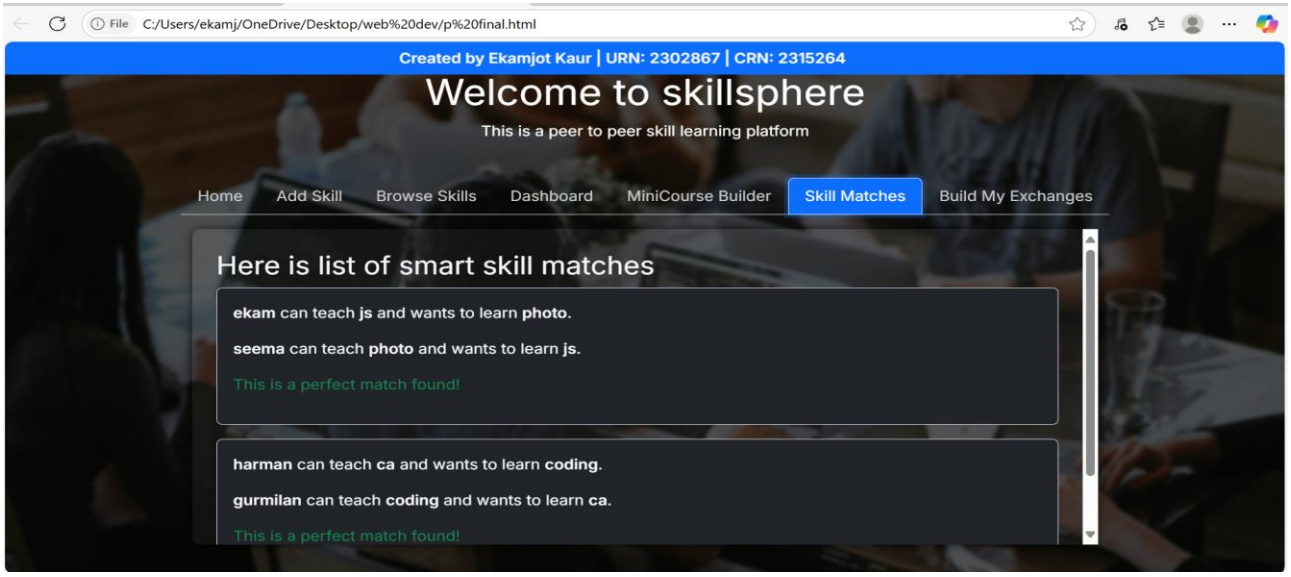


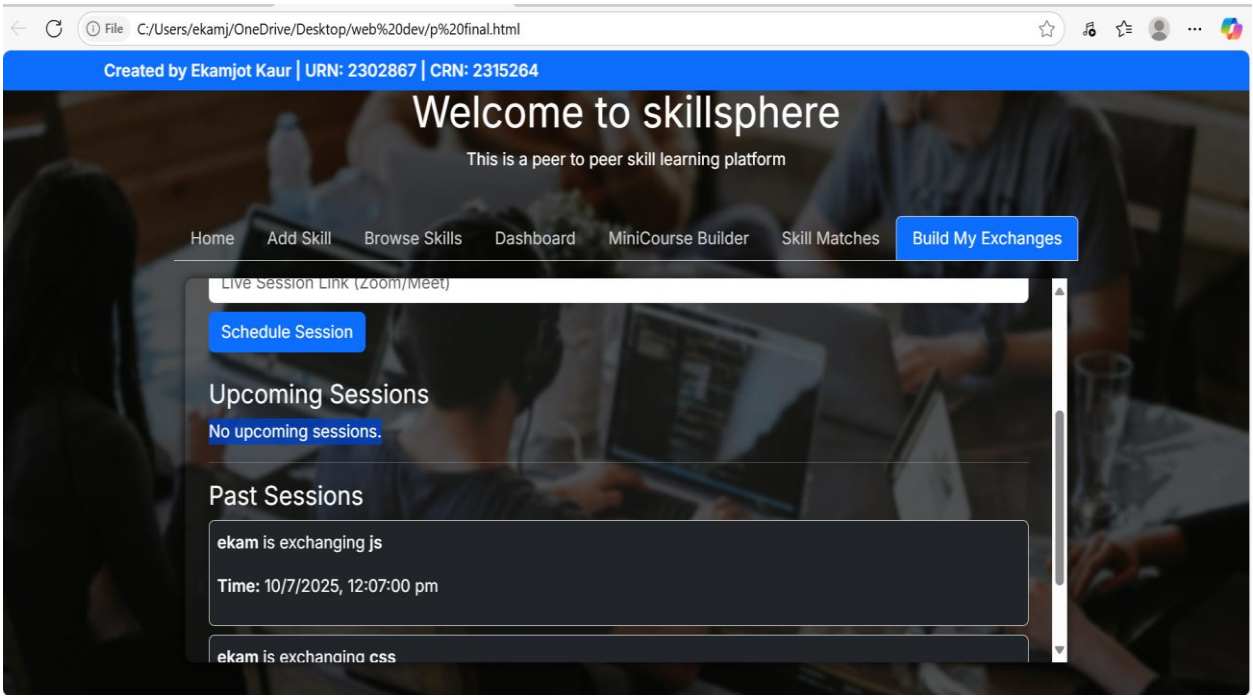
3.5 FINAL PROJECT –1 RESULT



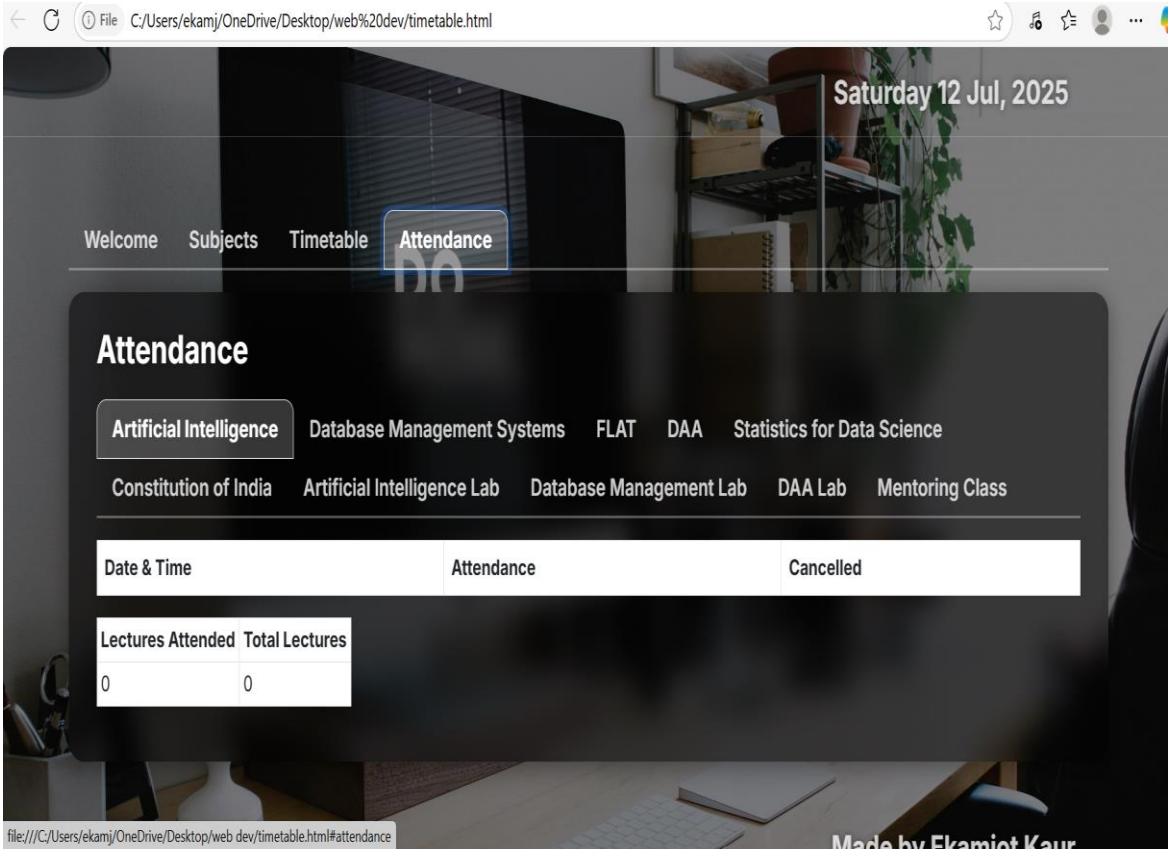


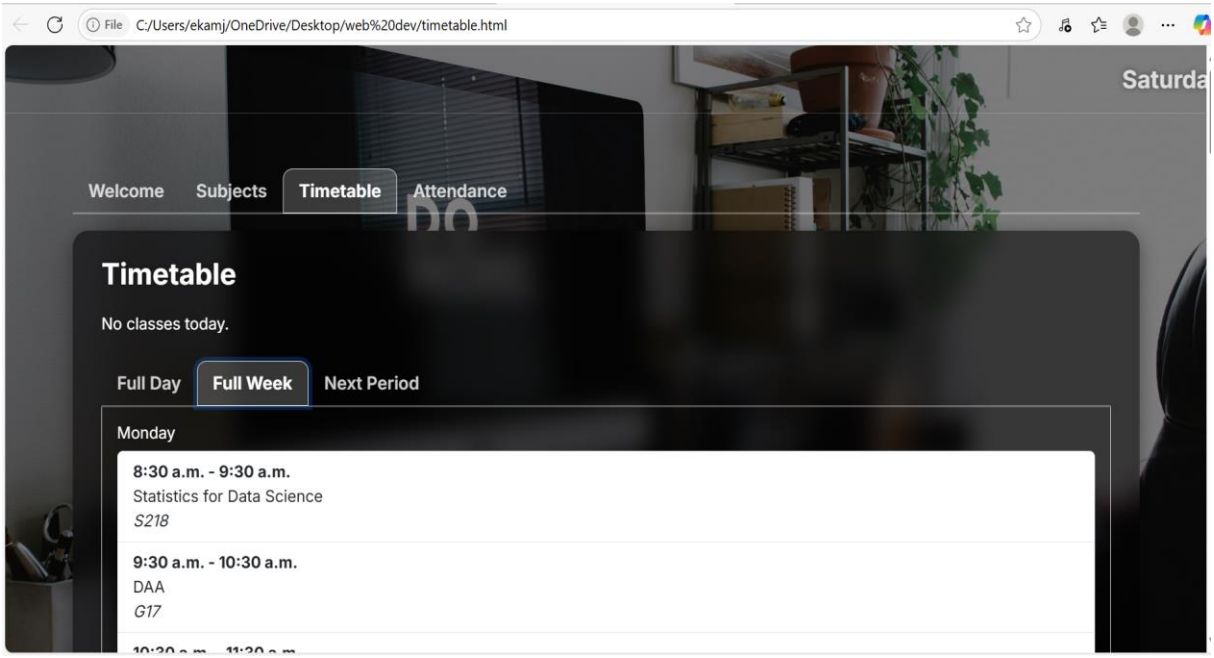

Created by Ekamjot Kaur | URN: 2302867 | CRN: 2315264
Welcome to skillsphere
This is a peer to peer skill learning platform
Home
Add Skill
Browse Skills
Dashboard
MiniCourse Builder
Skill Matches
Build My Exchanges
Create a MiniCourse
Course Title
Your Name
Course Content
Optional PDF Link
Quiz Question (optional)


File
C:/Users/ekamj/OneDrive/Desktop/web%20dev/p%20final.html
Created by Ekamjot Kaur | URN: 2302867 | CRN: 2315264
Welcome to skillsphere
This is a peer to peer skill learning platform
Home
Add Skill
Browse Skills
Dashboard
MiniCourse Builder
Skill Matches
Build My Exchanges
Here is list of smart skill matches
ekam can teach js and wants to learn photo.
seema can teach photo and wants to learn js.
This is a perfect match found!
harman can teach ca and wants to learn coding.
gurmilan can teach coding and wants to learn ca.
This is a perfect match found!



3.6 Project 2 result





3.7 PROJECT 1 DISCUSSION Skill Exchange Platform

3.7.1 What Is This Website?

This is a smart, easy-to-use platform where people can teach, learn, and exchange skills with others. It's perfect for learners, hobbyists, and professionals who want to grow by sharing knowledge in a community-driven way — all without needing to pay or code.

3.7.2 What Can You Do on This Website?

1. Skill Exchange

You can list the skills you want to teach and the skills you want to learn.

Other users can request to learn from you — and you can do the same with them.

You can even edit or delete your entries anytime.

2. Live Sessions

Plan and schedule live learning sessions with others.

You can add a session with date, time, and a video link (like Zoom or Google Meet).

Upcoming and past sessions are neatly shown in separate lists.

3. Mini Courses

Share your knowledge by creating a short course.

Each course can include:

- A title and description
- A PDF download (optional)
- A quiz question and answer
- A rating system so others can rate your course

All courses are saved and displayed for others to read and learn from.

4. Smart Skill Matching

The website can automatically find and show matches — people who are looking to learn what you teach, and vice versa.

You don’t need to search manually — the system finds perfect learning partners for you!

5. Dashboard

Track your learning progress at a glance:

- How many hours you’ve taught
- How many hours you’ve learned
- Your current balance (to keep things fair and rewarding)

No Sign-Up Required

Everything you do is saved in your browser, so you won't lose your data unless you clear your history.

No login, password, or email is needed — it's fully private and local to your device.

3.7.3 Who Is This For?

- Students looking to learn from peers
- Teachers sharing skills casually
- Professionals upskilling or reskilling
- Communities promoting free knowledge exchange
- Anyone who loves learning and teaching

3.7.4 SECTION 1: Moving Banner

What Is It?

The moving banner is a scrolling text strip placed at the very top of the webpage. It displays the creator's name and registration details, acting like a digital signature or ID bar. It is usually the first thing a user notices when the page loads.

What Does It Do?

- It continuously scrolls a line of text from right to left across the screen.
- It creates a dynamic and professional-looking header.
- It draws attention and adds visual interest to the site.

What Languages Are Used?

- HTML is used to create the structure of the banner, including where the text goes.
- CSS is used to animate the text and define how it should behave and appear on the screen.

How the Styling Works:

- The text is placed inside a container and styled to stay on one line.
- The animation tells the text to move across the screen in a loop.
- The background is often styled in a dark color, while the text is light to increase visibility.
- The banner is placed at the top using layout techniques like padding and positioning.

Functions and Classes Used

1. class="moving-banner"

- This is a CSS class name used to target the entire banner section.
- It is applied to the outer div that holds the scrolling text.
- The CSS for this class usually includes:
    - A fixed height and width to define the banner area.
    - A background color to make it visually distinct.
    - overflow: hidden to hide the part of the text that goes outside the container.

2. class="moving-text"

- This is the CSS class applied to the actual text inside the banner.
- It defines how the text behaves inside the moving-banner container.
- This class includes animation properties such as:
    - white-space: nowrap to ensure the text stays on one line.
    - A CSS animation (defined using @keyframes) that moves the text from right to left.

- animation: slideText linear infinite which starts the movement and ensures it loops forever.

3. @keyframes slideText

- This is a CSS animation function.
- It defines the path or direction the text should move.
- In this case, it moves the text from 100% (off-screen right) to -100% (off-screen left).
- The name of the animation (slideText) is then called inside the moving-text class to apply the effect.

3.7.5 SECTION 2: Welcome Section with Background Image

What Is It?

This section appears right below the moving banner and acts as the main welcome area of the website. It includes a large heading ("Welcome to SkillSphere") and a short description about the platform.

It sits on top of a full-screen background image, giving the website a modern and professional look.

What Does It Do?

- Welcomes users to the site with a bold heading and a short paragraph.
- Shows the main purpose of the platform: a peer-to-peer skill learning platform.
- Sets the visual tone of the site using a large background image with styled text over it.
- Uses layout and positioning techniques to center the text both vertically and horizontally.

What Languages Are Used?

- HTML is used to create the structure — headings, paragraph, and section containers.

- CSS is used to:
    - Add and style the background image.
    - Overlay a transparent layer over the image for readability.
    - Center the text inside the section.
    - Adjust spacing, alignment, and font sizes.

Functions and Classes Used

1. class="bg-img"

- This class is applied to a that covers the whole background of the welcome section.
- It sets a background image using the background-image property in CSS.
- Additional properties often used:
    - background-size: cover — makes sure the image fills the screen without distortion.
    - background-position: center — centers the image for visual balance.
    - height: 100vh — sets the height of the section to 100% of the visible screen (vh = viewport height).
    - This class holds the image but does not directly contain the visible text.

2. class="overlay"

- This sits on top of the background image like a thin layer.
- Purpose: To dim the background slightly so that white or light-colored text becomes easier to read.
- In CSS, this is done by:
    - Giving it a semi-transparent black background using background-color: rgba(0, 0, 0, 0.5).
    - Using position: absolute or position: relative to layer it over the image.

3. class="text-center"

- This is a utility class used to center the text horizontally.
- In CSS, it usually applies: text-align: center.
- Ensures that both the heading (<h1>) and the paragraph (<p>) are centered on the screen.

4. <h1> and <p>

- <h1> is a top-level heading element. It is used to display “Welcome to SkillSphere”.
- <p> stands for "paragraph" and is used to show “This is a peer-to-peer skill learning platform”.
- These elements are styled using CSS (possibly with custom font sizes, colors, and margins) to make them bold, clear, and visually attractive.

3.7.6 SECTION 3: Navigation Tabs

What Is It?

This section contains a row of navigation tabs — clickable buttons arranged horizontally across the screen. Each tab opens a different section or content area on the page without needing to reload the site.

It works like an interactive menu, giving users quick access to multiple tools or sections like "Home", "Dashboard", or "MiniCourse Builder".

What Does It Do?

- Lets users switch between different sections of the site (like a tabbed notebook).
- Only one tab section is shown at a time — the rest stay hidden until selected.
- The selected tab is visually highlighted to show it is active.

What Languages Are Used?

- HTML: For the structure of the tabs and buttons.
- CSS: To style the tabs — including layout, spacing, colors, and highlighting.
- JavaScript / Bootstrap JS: For dynamic behavior — allowing content to switch when a tab is clicked (without refreshing the page).

Functions and Classes Used

1. class="nav nav-tabs"

- These are Bootstrap classes:
    - nav turns the list into a navigation bar.
    - nav-tabs styles the nav bar into tab-style buttons (rather than plain text links).
- These classes come from Bootstrap, a popular CSS framework that simplifies responsive and clean web design.

2. and <li class="nav-item">

- (unordered list) holds all the tab buttons.
- Each <li class="nav-item"> is a single tab item in that list.
- Using list items makes it easy to arrange tabs horizontally and manage their layout.

3. <button class="nav-link">

- This is the clickable tab button that the user interacts with.
- nav-link is a Bootstrap class that gives the button its tab appearance.
- When clicked, it activates the content related to that tab.

4. class="nav-link active"

- The active class means this tab is currently selected (shown when the page loads).
- Only one tab should have active at a time — it tells Bootstrap to show that tab's content.

5. data-bs-toggle="tab"

- This is a Bootstrap function (using data attributes).
- It tells the browser: “When this button is clicked, switch to tab mode.”
- This works with Bootstrap’s JavaScript to activate the correct content panel below the tabs.

6. data-bs-target="#home" (and other IDs like #add, #browse...)

- This function links each button to a specific content area.
- For example:
    - The "Home" button points to #home.
    - The "Add Skill" button points to #add.
    - When you click a tab, it activates the content inside the matching that has the same ID.

7. class="justify-content-center mt-4"

- These are utility classes:
    - justify-content-center: Centers all the tabs horizontally.
    - mt-4: Adds top margin (spacing above the tabs) to separate them from the section above.

3.7.7 SECTION 4: Tab Contents

What Is It?

This section contains the actual content for each tab. When you click a tab (like "Home", "Add Skill", or "Dashboard"), the browser will show the content for that tab here.

You can think of it like a book with many pages — the tabs are the index, and the tab content section is the page being read.

What Does It Do?

- Each inside this section holds the content for one tab.
- When a tab is clicked, the corresponding becomes visible, and the others are hidden.
- This switching is done dynamically (without page refresh), using Bootstrap's built-in tab switching feature.

What Languages Are Used?

- HTML: To define and organize the content inside each tab.
- CSS: To style the content area (e.g., text color, scroll behavior, layout).
- Bootstrap JS: Handles the actual switching of visible tabs — using classes like active, fade, and show.

Functions and Classes Used

1.

- This is the main container that holds all the tab panels.
- All tab content areas must go inside this tab-content container for Bootstrap to manage switching correctly.

2. text-start text-light mt-3

- text-start: Aligns text to the left.
- text-light: Makes text color light (usually white), useful on dark backgrounds.
- mt-3: Adds top margin for spacing.

3. style="overflow-y:auto; max-height:60vh;"

This inline CSS styling controls how the content behaves:

- overflow-y: auto; means if the content is too tall, a vertical scrollbar will appear so you can scroll through it.
- max-height: 60vh; means the height of this section will not go beyond 60% of the screen height (vh = viewport height). This keeps it neat and prevents it from taking over the whole page.

4.

This is the actual content for the "Home" tab.

Let's break down the classes and attributes:

- tab-pane: This tells Bootstrap that this div is a tab content panel.
- fade: Adds a smooth fade-in effect when the tab becomes active.
- show active: These two classes mean this tab is currently visible and selected.
- id="home": This must match the data-bs-target="#home" from the nav button — it links the tab with this content.

So when the "Home" button is clicked, Bootstrap knows to show this particular .

5. Welcome to SkillSphere homepage...

This is just regular paragraph text, used to describe what the "Home" tab is about. You can insert any content here — text, images, forms, etc.

3.7.8 SECTION 5: Add Skill Tab

What Is It?

The "Add Skill" tab is an input form where users can submit their skill exchange details — like what skill they can teach, what they want to learn, and how many hours they're available for. This is a key feature of your peer-to-peer skill exchange system.

What Does It Do?

- Allows users to enter personal details and submit a skill exchange.
- Accepts 4 types of input:
    - Their name
    - Skill they can teach
    - Skill they want to learn
    - Hours they can offer
- A submit button lets them send the information (usually to be saved or processed).
- A confirmation message appears after submission.

What Languages Are Used?

- HTML: To build the structure of the form and the input fields.
- CSS (mostly Bootstrap): For layout, spacing, and styling the form nicely.
- JavaScript (used behind the scenes, not shown here): To handle what happens when the user submits the form (like validating or saving data).

Functions and Classes Used

1.

- This is the content block for the “Add Skill” tab.
- tab-pane lets Bootstrap know it's a tab section.
- fade gives a smooth fade-in transition.
- id="add" connects this block to the tab button that says “Add Skill”.

2. <form id="skillForm">

- This creates the form element, which groups all input fields together.

- id="skillForm" is used for JavaScript — to identify and process this form when the user hits Submit.
- class="mb-3" adds a bottom margin to the form for spacing.

3. <input> Fields

Each input is a form control styled by Bootstrap:

TABLE 3.1

Field ID	Purpose	Explanation
username	User's name	Text input
teachSkill	Skill user can teach	Text input
learnSkill	Skill user wants to learn	Text input
hours	Hours user can offer	Number input (min 1, max 20)

Each field uses:

- class="form-control" → Makes the input look professional and full-width.
- mb-2 → Adds bottom spacing between fields.
- placeholder → Shows hint text inside the box.
- required → Makes sure the field must be filled out before the form can be submitted.

4. <button type="submit" class="btn btn-primary">

- This is the form submission button.
- type="submit" makes it submit the form.
- btn btn-primary is a Bootstrap class that gives the button a blue color and styling.

5.

- This is an empty message area that shows a success message when the form is submitted correctly.
- text-success makes the message appear in green, indicating success.
- It's often used by JavaScript to show feedback like: "Skill added successfully!"

3.7.9 SECTION 6: Browse Skills Tab

What Is It?

The "Browse Skills" tab shows a list of all users who have submitted skill exchange forms. It displays their name, the skill they can teach, what they want to learn, their available hours, and gives an option to take action (e.g., match, message, connect).

What Does It Do?

- It dynamically displays skill data from other users.
- It uses a table layout for clarity.
- Data is usually added here using JavaScript, so it updates as new people fill the form in the "Add Skill" tab.

What Languages Are Used?

- HTML: For the basic table structure.
- CSS (Bootstrap): For styling the table so it's clean, dark-themed, and striped.
- JavaScript: Fills the table with real user data using innerHTML or DOM functions.

Functions and Classes Used

1.

- This is the tab panel for "Browse Skills".
- fade adds a soft transition.
- id="browse" matches with the navigation tab's data-bs-target="#browse".

2. <table class="table table-dark table-striped">

This is a Bootstrap-powered table. Here's what each class does:

- table: Base class to create a table.
- table-dark: Applies a dark background to the table.
- table-striped: Adds alternating row colors for better readability.

3. <thead> and <tr><th>...</th></tr>

- <thead> defines the header row of the table.
- Each <th> (table header) defines one column: Name, Teaches, Wants to Learn, Hours, and Action.

4. <tbody id="skillsTableBody">

This is where all the actual skill entries go.

- The id="skillsTableBody" allows JavaScript to insert rows here.
- JavaScript might use a loop to add multiple <tr> (table rows) based on form data.
- The “Action” column might include a button like “Connect” or “Match” — added dynamically.

3.7.10 SECTION 7: Dashboard Tab

What Is It?

The Dashboard tab gives the user a summary of their time-based balance in the skill exchange system.

It calculates:

- How many hours they can teach.
- How many hours they’ve requested to learn.

- Their remaining balance of time.

This is the “personal account” or “wallet” section for the user.

What Does It Do?

- Shows three numbers:
    - Total teaching hours (#totalTeach)
    - Total learning hours (#totalLearn)
    - Balance = Teach – Learn (#balance)
- These numbers are updated using JavaScript based on the user’s submissions.

What Languages Are Used?

- HTML: To structure and label the content.
- CSS (Bootstrap): For text styling and layout.
- JavaScript: To calculate totals and display them live.

Functions and IDs Used

1.

- tab-pane: Identifies this as a tab content block.
- id="dashboard": Linked to the Dashboard button/tab.
- fade: Adds transition.

2. , ,

- These are placeholder spans where JavaScript inserts numbers.
- When the form is submitted in the “Add Skill” tab, JavaScript updates these spans:
    - totalTeach: Tracks all the hours the user can offer.

- totalLearn: Tracks all the hours the user requested to learn.
- balance: The result of totalTeach - totalLearn.

3.7.11 SECTION 8: Build My Exchanges Tab

What Is It?

This tab allows users to schedule a live skill exchange session — like a one-on-one Zoom or Google Meet between two learners.

What Does It Do?

- Lets users enter details like their name, skill, time, and a meeting link.
- Shows Upcoming Sessions and Past Sessions dynamically.
- Acts like a personal calendar for peer learning.

What Languages Are Used?

- HTML: To create the input fields and layout.
- CSS / Bootstrap: For styling the form and layout.
- JavaScript: To save the session details, display them under "Upcoming" and later move them to "Past".

Functions, Classes, and IDs Used

1. <form id="sessionForm">

- This is the session scheduling form.
- JavaScript uses the form's id to listen for form submissions.

- Prevents page reload using event.preventDefault() in JavaScript.

2. Form Inputs:

Each <input> collects a different detail:

- id="sessionName" — user's name.
- id="sessionSkill" — the skill being exchanged.
- id="sessionDateTime" — date & time for session using datetime-local input type.
- id="sessionLink" — URL to join the session (Zoom/Meet).

Each of these IDs is accessed in JavaScript to read the values when the form is submitted.

3. id="upcomingSessions" and id="pastSessions"

- Two sections to display session info using JavaScript.
- Sessions are shown in readable cards or text blocks under these sections.

3.7.12 SECTION 9: MiniCourse Builder

What Is It?

This tab lets users create and publish mini-courses — including text-based lessons, optional PDFs, and quizzes.

What Does It Do?

- Collects a title, content, and author info.
- Users can optionally include a quiz and a PDF link.
- The course appears below in the "Published Courses" area.

What Languages Are Used?

- HTML: Used to structure the input fields and form.
- CSS / Bootstrap: For visual styling and layout.
- JavaScript: Handles publishing logic, displays new courses instantly in the page.

Functions, Classes, and IDs Used

1. <form id="courseForm">

- The main form to submit a course.
- JavaScript listens for the submit event on this form.

2. Key Input Fields:

- id="courseTitle" — course title.
- id="courseAuthor" — creator's name.
- id="courseContent" — main content area (textarea for longer text).
- id="pdfLink" — optional link to a downloadable PDF.
- id="quizQuestion" and id="quizAnswer" — optional quiz.

3. id="courseList"

- Where published courses are shown.
- JavaScript creates course cards or blocks and places them inside this div.

3.7.13 SECTION 10: Skill Matches Tab

What Is It?

This tab shows intelligent or smart matches — connecting users who can benefit from each other's skills.

What Does It Do?

- Automatically shows matched pairs (e.g., A teaches what B wants to learn).
- Helps users find ideal partners for learning.

What Languages Are Used?

- HTML: Basic structure of the tab.
- JavaScript: Handles the logic to generate matches based on form data.
- CSS / Bootstrap: Used to keep the section visually clean and modern.

Functions, Classes, and IDs Used

1. id="matches"

- Tab container, connected to the navigation bar.

2. id="matchesContainer"

- Where the list of smart matches appears.
- Filled dynamically using JavaScript.
- Matches may include names, skills, and a button to connect.

3.7.14 SECTION 11: Skill Exchange Logic and Dashboard Updates

What Is It?

This section of the script manages the core logic behind skill exchange functionality. It allows users to submit skill listings, make learning requests, update the dashboard, and manage session data.

What Does It Do?

- Stores and retrieves skill exchange data using localStorage.
- Renders the skill exchange table dynamically.
- Allows users to request hours from listed skills.
- Updates the dashboard with total hours taught, learned, and balance.
- Handles scheduling logic for upcoming and past live skill sessions.

What Languages Are Used?

- JavaScript: Controls the logic for form submissions, event handling, table rendering, session tracking, and localStorage interaction.
- HTML: Referenced by getElementById() for UI updates and rendering data.
- Bootstrap (CSS): Used to style buttons, tables, and alert messages consistently.

Functions, Classes, and IDs Used

1. Function: renderSkillsTable()
    a. Populates the "Browse Skills" table using data from skills[].
    b. If no data exists, displays a placeholder row.
    c. Appends a "Request" button for each skill entry.
2. Function: renderSessions()
    a. Sorts skill exchange sessions into upcoming and past based on current date/time.
    b. Populates #upcomingSessions and #pastSessions containers with session cards.
    c. Displays session time and a join button (if upcoming).
3. Function: updateDashboard()
    a. Calculates total teach and learn hours from skill data.
    b. Updates the HTML elements: #totalTeach, #totalLearn, and #balance.
4. Event Listener: #skillForm Submit

a. Captures new skill exchange entries (username, skills, hours).

b. Validates the form and pushes data into the skills[] array.

c. Saves updated data in localStorage.

d. Calls renderSkillsTable(), updateDashboard(), and renderMatches().

e. Displays temporary confirmation text via formMsg.

5. Event Listener: #skillsTableBody Click

a. Detects clicks on .request-btn buttons.

b. Prompts user for hours to request.

c. Validates and stores request in learningRequests[].

d. Updates localStorage and the dashboard.

6. Variable: skills

a. Holds all teaching skill entries.

b. Initialized from localStorage or set as an empty array.

7. Variable: learningRequests

a. Stores all requested learning hours for each user.

b. Also saved to localStorage.

8. IDs Referenced in DOM

a. #skillsTableBody: Table body where skill rows are rendered.

b. #totalTeach, #totalLearn, #balance: Dashboard statistics.

c. #formMsg: Temporary confirmation message on form submission.

d. #upcomingSessions, #pastSessions: Containers for session scheduling.

3.7.15 SECTION 12: Live Sessions, Mini-Courses, and Skill Editing Logic

What Is It?

This section handles logic for live session tracking, mini-course management, and skill table editing. It includes form submissions, table rendering, editing, deleting, and rating systems for added content.

What Does It Do?

- Stores live sessions (sessions[]) and mini-courses (courses[]) using localStorage.
- Allows users to add courses with optional PDFs and quizzes.
- Renders course content and includes a rating feature.
- Enhances the skill table with edit and delete options for each entry.

What Languages Are Used?

- JavaScript: Controls form actions, rating system, content rendering, and button logic.
- HTML: Accessed for displaying dynamic course and skill data.
- Bootstrap (CSS): Styles buttons, rating dropdowns, and containers.

Functions, Classes, and IDs Used

1. Variable: sessions
    a. Stores live skill exchange session data.
    b. Pulled from localStorage.
2. Variable: courses
    a. Contains all user-created mini-courses.
    b. Saved and retrieved using localStorage.
3. Event Listener: #courseForm Submit
    a. Gathers course data (title, author, content, optional PDF, quiz, and answer).

b. Pushes to courses[] and updates localStorage.

c. Calls renderCourses() to refresh display.

4. Function: renderCourses()

    a. Displays all mini-courses inside #courseList.

    b. Shows title, author, content, optional PDF download button, and quiz.

    c. Includes a rating dropdown per course and shows average rating.

5. Function: averageRating(ratings)

    a. Calculates and returns average of an array of numeric ratings.

    b. Used for displaying average course ratings.

6. Event Listener: document → Rating Dropdown Change

    a. Monitors changes to <select> elements with data-index.

    b. On valid input (1-5), appends rating to course, updates storage, and rerenders.

7. Function: renderSkillsTable() (updated)

    a. Displays skill table rows with three buttons:

        i. Request: Send a learning request.

        ii. Edit: Load entry into form for modification.

        iii. Delete: Remove entry from list.

8. Event Listener: #skillsTableBody Click

    a. Handles three possible interactions:

        i. Request Button (.request-btn)

Prompts for hours, validates input, stores request in learningRequests[].

        ii. Delete Button (.delete-skill-btn)

Confirms action, removes entry from skills[], updates storage and UI.

        iii. Edit Button (.edit-skill-btn)

Pre-fills the form with current values.

Temporarily replaces the submit handler with an update handler.

After saving, resets form and restores original handler.

9. DOM IDs and Elements Used
    a. #courseForm: Mini-course submission form.
    b. #courseList: Container to show course cards.
    c. #skillsTableBody: Skill list table body.
    d. #skillForm: Main form for adding/editing skills.

3.7.16 SECTION 13: Skill Matching Logic and Session Submission

What Is It?

This final part of the script enables automatic skill pairing between users (matching teach/learn needs) and processes live session creation. It also contains the initial load logic to ensure all dynamic elements are rendered when the page loads.

What Does It Do?

- Finds perfect skill exchange matches between users and displays them.
- Handles form submission to create a live learning session.
- Runs key rendering functions when the app starts to ensure the UI is populated with existing data.

What Languages Are Used?

- JavaScript: Matching logic, DOM manipulation, form validation, and data rendering.
- HTML: Renders skill match results and session entries.
- Bootstrap: Used to style the match and session displays.

Functions, Classes, and IDs Used

1. Function: renderMatches()
    a. Loops through all skill entries to find mutual exchanges:
        i. Person A teaches what Person B wants to learn and vice versa.
    b. If match found:
        i. Displays both users' skills and a confirmation message.
    c. If no match or not enough data:
        i. Shows a message in #matchesContainer.
2. HTML Element: #matchesContainer
    a. Main area where the matched skill exchange pairs are shown.
    b. Filled dynamically with Bootstrap-styled blocks.
3. Function: renderSessions()
    a. Called again in this section to ensure live sessions appear after a new one is added.
4. Initial Load Section
    a. Calls rendering functions immediately:
        i. renderSkillsTable()
        ii. updateDashboard()
        iii. renderCourses()
        iv. renderMatches()
        v. renderSessions() ← This was missing earlier and now included.
5. Event Listener: #sessionForm Submit
    a. Adds new live session based on form input:
        i. sessionName, sessionSkill, sessionDateTime, sessionLink.
    b. Validates inputs.

c. Saves session to sessions[] and updates localStorage.

d. Refreshes display with renderSessions().

]

3.8 PROJECT 2 DISCUSSION Student Dashboard

3.8.1 What Is This Website?

This is a personal student dashboard. It helps students keep track of their subjects, class timetable, and attendance — all in one place. It is designed to be simple to use, even for people who have no background in coding or technology.

3.8.2 What Can You Do With It?

1. Track Your Attendance

- You can mark yourself present or absent for each lecture.
- If a class did not take place, you can mark it as cancelled.
- The site will automatically calculate how many classes you attended, how many were held in total, and what your attendance percentage is for each subject.

2. View Your Timetable

- You can see your daily and weekly class schedule.
- There are three viewing options:
    - Full Day – shows today's classes.
    - Full Week – shows your weekly schedule.
    - Next Period – shows the next upcoming class.

3. See Your Subjects

- A simple list shows all your current subjects.

- This includes both theory and lab sessions, as well as any mentoring classes.

4. Welcome Tab

- The first tab contains a short welcome message explaining the purpose of the dashboard.

5. Footer Banner

- At the top and bottom of the page, there's a banner that:
    - Displays the current date.
    - Shows a message: "Made by Ekamjot Kaur".

3.8.3 How to Use It

1. Open the dashboard in a web browser (like Google Chrome).
2. Use the row of tabs at the top to switch between different sections: Welcome, Subjects, Timetable, and Attendance.
3. In the Attendance section, click the buttons to mark each class as Present, Absent, or Cancelled.
4. You can revisit any section at any time without reloading the page.

3.8.4 Who Is It For?

This tool is designed for students who want a simple way to manage their academic life, especially tracking attendance and understanding their weekly schedule — all without needing to install anything or know how to code.

3.8.5 SECTION: Dashboard Navigation Tabs and Content

What Is It?

This section defines a dashboard interface consisting of a top banner and a set of navigation tabs. Each tab corresponds to a specific panel like "Welcome", "Subjects", "Timetable", and "Attendance". When a user clicks on a tab, the corresponding content section is shown dynamically, while others remain hidden.

This setup provides an interactive way for users to explore different tools and views in the application without reloading the entire page.

What Does It Do?

- Displays a date banner at the top (presumably populated via JavaScript).
- Shows a tabbed interface using Bootstrap-styled navigation tabs.
- Allows switching between different content panels (e.g., Welcome, Subjects, Timetable, Attendance).
- Ensures only one content panel is visible at a time.
- Automatically highlights the currently active tab.
- Uses Bootstrap's JavaScript to manage the behavior of tab switching.

What Languages Are Used?

- HTML: For structuring the interface — including tabs, tab panels, and layout containers.
- CSS (Bootstrap and Custom): For styling — including tab appearance, spacing, and visual aesthetics (e.g., glass effect).
- JavaScript (Bootstrap JS): For dynamic tab behavior — enabling smooth transitions without reloading the page.
- JavaScript (Custom): Expected for updating the date banner dynamically (not shown but implied by id="dateBanner").

Functions and Classes Used

1. class="marquee" and id="dateBanner"
    a. marquee is likely a custom or Bootstrap utility class that provides scrolling or styled text.
    b. id="dateBanner" acts as a target for JavaScript, which likely inserts the current date or time into the banner.
2. class="container mt-4"
    a. container: A Bootstrap layout class for fixed-width responsive spacing.
    b. mt-4: Adds top margin to separate the tab section from the banner above.
3. class="nav nav-tabs mb-3"
    a. nav: Turns the list into a Bootstrap navigation menu.
    b. nav-tabs: Styles the navigation menu into tab-like buttons.
    c. mb-3: Adds bottom margin to separate the tabs from the content below.
4. <li class="nav-item">
    a. Each list item defines a single tab in the tab group.
    b. Organizes tabs horizontally using Bootstrap layout rules.
5. <a class="nav-link active" data-bs-toggle="tab" href="#welcome">
    a. nav-link: Applies Bootstrap styling to the link to make it appear as a tab.
    b. active: Highlights this tab as the currently selected one when the page loads.
    c. data-bs-toggle="tab": Tells Bootstrap to treat this link as a tab trigger.
    d. href="#welcome": Links to the ID of the corresponding content panel.
6. id="mainTabs"
    a. Assigns an ID to the <ul> tab list. Useful for targeting the tab group with JavaScript if needed.
7.

a. Wrapper for all tab content panels.

b. Only one child tab-pane is shown at a time based on the selected tab.

8.

a. tab-pane: Defines a single tab content panel.

b. fade: Adds a smooth transition effect when tabs are switched.

c. show active: Makes this tab panel visible by default.

d. id="welcome": Matches the href of the corresponding tab link to associate them.

9. class="glass-card"

a. Likely a custom class for a visually styled card with a frosted glass effect.

b. Enhances aesthetics of the welcome message area.

10. Content Elements (<h3>, <p>)

- Provide a welcoming message and a short summary of the dashboard's purpose.

3.8.6 SECTION: Subjects and Timetable Tabs

Subjects Tab

What Is It?

This section displays a list of subjects that the user is currently enrolled in. It is part of the tabbed interface described earlier and becomes visible when the "Subjects" tab is selected.

What Does It Do?

- Presents a clean, scrollable list of all subjects, including theoretical and lab-based courses.
- Organizes the subjects using Bootstrap's list group component for a consistent and styled display.

- Ensures a visually cohesive layout using a custom-styled container (glass-card).

What Languages Are Used?

- HTML: For the structural layout of the subjects list.
- CSS (Bootstrap & Custom): For visual styling via list-group components and the glass-card effect.

Functions and Classes Used

1. class="tab-pane fade" and id="subjects"
    a. Defines the hidden tab panel that becomes visible when the "Subjects" tab is activated.
    b. fade: Enables a fade-in effect.
    c. id="subjects": Connects to the href="#subjects" from the navigation tab to display this panel.
2. class="glass-card"
    a. A custom class likely applying a translucent or blurred background for aesthetic presentation.
3. class="list-group"
    a. Bootstrap class to group multiple items together into a vertically stacked list.
    b. Each list item is styled consistently with padding, background color, and borders.
4. class="list-group-item"
    a. Applies Bootstrap styling to individual subject names.
    b. Each item is treated as a block-level element within the list.

3.8.7 Timetable Tab

What Is It?

This section provides a structured and dynamic timetable system. It includes a main container where the schedule is displayed, plus secondary navigation tabs allowing users to toggle between different timetable views: Full Day, Full Week, and Next Period.

What Does It Do?

- Acts as a placeholder for the timetable content via the #timetableContainer div.
- Allows switching between three additional views of the timetable using nested tabs.
- Integrates JavaScript functions (showFullDay(), showFullWeek(), showNextPeriod()) to dynamically load content into corresponding panels.

What Languages Are Used?

- HTML: For structure of the timetable and nested tabs.
- CSS (Bootstrap & Custom): For layout, spacing, tab styles, and the glass effect.
- JavaScript: Used for loading and displaying content dynamically when the user switches views (functions expected but not shown in current code).

Functions and Classes Used

1. class="tab-pane fade" and id="timetable"
    a. Defines the Timetable tab content area, hidden by default.
    b. Will be shown when the "Timetable" tab is clicked.
2. id="timetableContainer"
    a. Acts as a placeholder where JavaScript can insert the main timetable content.
3. class="nav nav-tabs mt-4" and id="extraTimetableTabs"
    a. Creates a nested tab system inside the Timetable section for different timetable views.
    b. mt-4 adds spacing above this section.

4. <button class="nav-link active"...>
    a. Each button switches between "Full Day", "Full Week", and "Next Period" timetable views.
    b. data-bs-toggle="tab" and data-bs-target="#...": These Bootstrap attributes enable dynamic switching between content sections without reloading.
    c. onclick="showFullDay()", etc.: Connects to external JavaScript functions (not shown) that likely fetch and display data in real time.
5. class="tab-content p-3 border border-top-0"
    a. Container for the three different timetable views.
    b. p-3: Adds internal padding.
    c. border and border-top-0: Adds borders around the content area but omits the top border to integrate smoothly with the tabs.
6.
    a. Represents the content area for the “Full Day” view.
    b. show active: Makes this the default visible pane.
7. and
    a. Additional content areas for "Full Week" and "Next Period" views.
    b. Hidden until activated by user interaction.

3.8.8 SECTION: Attendance Tab

What Is It?

The Attendance tab is an interactive section of the dashboard that organizes attendance tracking by subject. It uses a nested tab structure to display individual attendance panels for each subject. Each tab is labeled with a subject name, and selecting one will display the associated attendance details.

This structure allows users to manage and review their attendance records in an organized, subject-specific manner without needing to reload the page or scroll through a long list.

What Does It Do?

- Provides a tabbed interface within the Attendance tab to select different subjects.
- Displays a separate content panel for each subject, allowing for detailed, modular attendance tracking.
- Enhances user experience by keeping the UI clean and subject-specific.
- Uses Bootstrap's dynamic tab system to switch views instantly.

What Languages Are Used?

- HTML: For defining tab layout and subject content containers.
- CSS (Bootstrap & Custom): For styling the tabs, panels, and card container.
- JavaScript (Bootstrap): Enables tab switching and behavior through Bootstrap's data-bs-* attributes.
- JavaScript (Custom - Implied): Likely used to dynamically fill in attendance data within each subject panel (though not shown in current code).

Functions and Classes Used

1. class="tab-pane fade" and id="attendance"
    a. Defines the Attendance tab content container, hidden until activated.
    b. The fade class adds a transition effect.
2. class="glass-card"

    a. A custom styling class that applies visual effects such as a blurred glass background and padding.

3. <ul class="nav nav-tabs" id="subjectTabs" role="tablist">
    a. Creates a nested Bootstrap tabbed navigation bar for individual subjects.
    b. Each subject is listed as a separate button tab.
4. <button class="nav-link active" data-bs-toggle="tab" data-bs-target="#artificial-intelligence" type="button"> (and similar for other subjects)
    a. nav-link: Styles the button to appear as a tab.
    b. active: Marks the default selected tab (only applied to the first subject).
    c. data-bs-toggle="tab": Enables tab behavior using Bootstrap JavaScript.
    d. data-bs-target="#...": Specifies which panel should be shown when this tab is clicked.
    e. type="button": Declares the element as a button (not a link or submit).
5.
    a. Container that holds all individual subject panels for attendance.
    b. mt-3 adds vertical spacing above the panel content area.
6. (and similar for other subjects)
    a. Each div is a separate tab content panel for an individual subject.
    b. tab-pane: Required Bootstrap class to make the content behave as part of a tab system.
    c. fade: Enables smooth visibility transitions.
    d. show active: Only used on the default active subject's content (Artificial Intelligence).

e. The actual attendance content is expected to be injected into these panels dynamically or filled in later.

3.8.9 SECTION: Footer

What Is It?

A simple banner/footer is placed at the bottom of the page, styled similarly to the top banner.

What Does It Do?

- Displays a message: “Made by Ekamjot Kaur”.
- Acts as a signature or credit section for the developer.

What Languages Are Used?

- HTML: For structure.
- CSS (Likely Custom): Styled using the marquee class to match the top banner.

Functions and Classes Used

- Made by Ekamjot Kaur
    - Uses the same marquee class as the top banner, likely involving animated scrolling or highlighting.
    - Wraps the credit message inside a span for inline styling or JavaScript manipulation if needed.

3.8.10 SECTION: JavaScript – Timetable & Attendance Logic

What Is It?

This JavaScript block is the core logic for managing:

1. Timetable display (today's classes, week schedule, next period, current class).
2. Attendance tracking, including:
    a. Manual marking (present/absent/cancelled),
    b. Automatic lecture count updates,
    c. Storage and retrieval using localStorage.

It interacts with the HTML structure to dynamically display content and maintain persistent attendance records across sessions.

What Does It Do?

- Stores the timetable in an object (timetable) for each weekday.
- Dynamically displays:
    - Current class,
    - Next period,
    - Full day's timetable,
    - Full week's timetable.
- Allows attendance marking for each session (with logic to prevent double-counting).
- Uses localStorage to persist attendance data.
- Auto-increments total lecture count when a class has passed.

What Languages/Technologies Are Used?

- JavaScript (Vanilla) – Core logic.
- Bootstrap JS – Included via CDN for tab behavior.

- localStorage API – For persistent attendance tracking.

Functions and Their Roles

toMinutes(hm)

- Converts "HH:MM" to total minutes.
- Used for time comparisons.

mark(id, subject, status)

- Called when marking attendance.
- Stores the mark (present/absent/cancelled) in localStorage.
- Increments total and attended counts once per session.

formatTime(h, m)

- Formats time to 12-hour format with a.m./p.m.

showFullDay()

- Displays all periods for today, formatted with subject and room.
- Injects list into the element with ID day.

showFullWeek()

- Renders the entire week's timetable.
- Injects formatted subject details into the element with ID week.

showNextPeriod()

- Displays next upcoming class based on current time.
- Injects result into the element with ID next.

renderTimetable()

- Displays the current ongoing class, if any.
- Checks against current time to find active slot.

- Injects class info into timetableContainer.

autoUpdateTotalLectures()

- Automatically increments total lecture count after class ends.
- Avoids user needing to mark anything manually.
- Ensures accurate tracking by checking if already counted (-counted-auto flag).
- Skips “No class” entries.

Persistent Storage (localStorage) Keys Used

TABLE 3.2

Purpose	Format	Example
Mark for session	${day}-${time}	"Monday-08:30"
Total lectures	T-${subject}	"T-DAA"
Attended lectures	A-${subject}	"A-DAA"
Counted flag	${id}-counted	"Monday-08:30-counted"
Auto-counted flag	${id}-counted-auto	"Monday-08:30-counted-auto"

3.8.11 SECTION: Attendance & Timetable Controller Script

What Is It?

A comprehensive JavaScript module responsible for initializing, tracking, rendering, and updating attendance and timetable-related user interface components.

It manages:

- Daily attendance status
- Per-subject attendance records
- Class cancellations
- Attendance summaries and statistics

- Timetable rendering and refresh cycles

What Does It Do?

- Initializes default attendance records for every subject and scheduled time between the start date and the current date.
- Automatically marks unrecorded attendance as “absent” unless a class is cancelled.
- Allows users to manually mark attendance status as present or absent, or flag a lecture as cancelled.
- Renders attendance data in tabular format per subject.
- Generates overall attendance statistics displayed at the top of the Attendance tab.
- Refreshes the UI at regular intervals (every hour for attendance, every minute for the timetable).
- Updates a banner with the current date.

What Languages Are Used?

- JavaScript: Logic for initialization, rendering, interaction, and local data storage.
- HTML (Generated Dynamically): For constructing tables and UI elements per subject.
- localStorage API: For persistent data storage of attendance and cancellations.
- Date API: For iterating through date ranges and formatting dates.

Functions and Classes Used

Function: renderAttendance()

- Selects the attendanceStats container.
- Gathers all subjects found in the timetable.

- For each subject:
    - Retrieves attendance and total values from localStorage.
    - Calculates percentage.
    - Appends a summary line in the format: Subject: X/Y (Z%).

Function: initAttendance()

- Iterates from the defined start date to the current date.
- For each day:
    - Checks if the subject is scheduled on that day.
    - Creates default localStorage entries:
        - ♣ pending for attendance status.
        - ♣ false for cancellation status (if not already set).

Function: setAttendance(id, status)

- Updates the localStorage item for a specific class ID to either "present" or "absent".
- Triggers a UI refresh via refreshAll().

Function: toggleCancel(id, checkbox)

- Updates the localStorage item <id>-cancelled based on checkbox state ("true" or "false").
- Refreshes the UI to reflect the change.

Function: renderSubject(subject)

- Fetches the container for a given subject using normalize().
- Iterates through all past dates, checking for matching classes in the timetable.

- Builds a table containing:
    - Class date and time
    - Present/Absent buttons
    - Cancelled checkbox
- Automatically marks unmarked entries as "absent" if they are not cancelled.
- Counts total and attended classes for summary display.

Function: normalize(str)

- Converts subject names to lowercase ID-safe strings:
    - Replaces spaces and symbols with dashes.
    - Removes non-alphanumeric characters.
    - Ensures ID consistency with HTML tab pane targets.

Function: refreshAll()

- Loops through each subject in the subject list.
- Calls renderSubject() to refresh the UI for each.

Function: bannerDate()

- Selects the dateBanner element.
- Formats and sets the current date in a readable form (e.g., “Friday, Jul 12, 2025”).

Other Logic and Calls

```
initAttendance();          // Initialize data
refreshAll();              // Render all subject tabs
```

```
setInterval(refreshAll, 60*60*1000); // Refresh attendance hourly


bannerDate();              // Set date banner
renderTimetable();         // Show current timetable
renderAttendance();        // Show attendance summary
setInterval(renderTimetable, 60000); // Refresh timetable every minute
```

localStorage Key Structure

TABLE 3.3

Key Format	Description
2025-07-21-09:00-DAA	Attendance record ID
2025-07-21-09:00-DAA-cancelled	Cancellation flag for that class
A-DAA	Total attended count (for summary display)
T-DAA	Total lectures held (for summary display)

Chapter 4 CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

Through the completion of my two projects using **HTML, CSS, Bootstrap, and JavaScript**, I gained valuable hands-on experience in the field of **web development**. This journey provided me with both theoretical understanding and practical application of how modern websites are designed, structured, and made interactive.

HTML formed the foundation of both projects, allowing me to define the structure and layout of each web page. Using CSS, I was able to style the pages to enhance their appearance, making them visually appealing and user-friendly. Bootstrap played a significant role in streamlining the design process with its pre-built responsive components, ensuring that my websites looked good across different screen sizes and devices. JavaScript added interactivity and functionality, enabling features like dynamic content updates, tab switching, attendance tracking, and time-based displays.

The first project helped reinforce the basics — such as creating layouts, navigation bars, and styled components — while the second project allowed me to integrate more advanced features like dynamic rendering, form interaction, and local storage management.

Overall, these projects strengthened my understanding of front-end web development. I learned how to build responsive, interactive, and well-structured web applications. Most importantly, I gained confidence in using core technologies that power the front end of the modern web. This experience has laid a strong foundation for deeper learning in web development and has motivated me to explore backend technologies and full-stack development in the future.

4.2 FUTURE SCOPE

Web development is a fast-growing field with many opportunities for learning and innovation. The projects I completed using HTML, CSS, Bootstrap, and JavaScript have helped me build a strong foundation in front-end development. However, there is still a lot more that can be explored in this domain.

In the future, one of the main areas to work on is moving from static websites to dynamic ones. This can be achieved by learning backend technologies like Node.js, Django, or PHP. Adding a backend will allow features like storing user data, login systems, and interaction with databases.

Another important area is using JavaScript frameworks such as React or Vue.js. These frameworks make websites faster and more responsive. They are widely used in modern web development and can help in building more complex and interactive applications.

There is also scope to improve the design and user experience. Tools like Tailwind CSS, SASS, or UI libraries can be used to make websites more attractive and easier to use. Making sure that the websites work well on all devices (like phones and tablets) is also important.

In addition, learning how to connect external APIs can make the projects more useful by adding real-world features like live weather updates, maps, or search results. Deploying websites online using platforms like GitHub Pages or Netlify can help in sharing the projects with others and building a portfolio.

Overall, these projects are just the beginning. There is a lot of potential to grow by learning full-stack development, improving design skills, and building real-world applications that solve actual problems.

REFERENCES

[1] J. Duckett, *HTML and CSS: Design and Build Websites*, 1st ed. Indianapolis, USA: Wiley, 2011.

[2] Mozilla Contributors. (2024, Mar. 10). HTML Basics [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

[3] Bootstrap Team. (2023, Aug. 15). Introduction to Bootstrap [Online]. Available: https://getbootstrap.com/docs/5.3/getting-started/introduction/

[4] W3Schools. (2024, Feb. 20). JavaScript Tutorial [Online]. Available: https://www.w3schools.com/js/

[5] M. Galloway, "Introduction to Web Development," *International Journal of Computer Science Education*, vol. 18, no. 2, pp. 45–52, Apr. 2022.

[6] A. Brown and R. K. Singh, "Responsive design in modern websites," *Proceedings of the National Conference on Computing Trends*, Ludhiana, India, 2023, pp. 101–106.

APPENDIX

A. Overview of Developed Projects

During the training period, I developed two web-based projects focused on front-end development using HTML, CSS, Bootstrap, and JavaScript:

1. Timetable and Attendance Tracker
    a. A functional interface to display and manage class schedules.
    b. Includes an attendance management system with local storage, interactive buttons, and cancel options.
2. Dynamic Web Dashboard
    a. A web layout with sections like live date display, subject-wise attendance statistics, and responsive design using Bootstrap.
    b. Integrated JavaScript functions to dynamically update data, render tables, and handle user interaction.

B. Tools and Technologies Used

- Languages: HTML5, CSS3, JavaScript
- Framework: Bootstrap 5
- Editor: Visual Studio Code
- Browser: Google Chrome
- Version Control: Git (used locally)

C. Key Learning Outcomes

- Gained practical experience in DOM manipulation and event handling in JavaScript.

- Learned to create responsive and user-friendly web interfaces using Bootstrap.
- Understood how to store and retrieve data using localStorage for persistent web apps.
- Improved skills in debugging, browser inspection tools, and layout design.