COLLEGE CODE	:	9111
COLLEGE NAME	:	SRM Madurai College For Engineering And Technology
DEPARTMENT	:	B.E Computer Science and Engineering
STUDENT NM-ID	:	A0D2DF9E42A2518BFCC4400C4B39CCB0
REGISTER NO	:	911123104034
DATE	:	22/09/2025

Completed the project named as : IBM-FE-Dynamic Image Slider

SUBMITTED BY,

NAME : Prasanth Narayanan V S

MOBILE NO : 8015928461

Project Setup

The Project Setup phase is the foundational step where you create the basic structure for your dynamic image slider. This involves more than just creating a folder; you'll set up the development environment and version control.

- Project Initialization: Create a root folder for your project. Inside it, create sub-folders for your code (src), images (images), and any other assets you might use. * HTML, CSS, and JavaScript Files: Create your main index.html, a styles.css file for styling, and a script.js file for the dynamic functionality. These three files will be the core of your slider.
- Dependencies: While a simple slider might not need external libraries, you could consider including a framework like Bootstrap for pre-built components and responsive design, or a lightweight library like Swiper.js if you need advanced features. The provided code snippet already uses Bootstrap, which is an excellent choice for this.
- Version Control: This is a critical step. Initialize a new Git repository in your project folder (git init) and connect it to a remote repository on GitHub (git remote add origin [your-repo-url]). This allows you to track changes, collaborate with others, and easily revert to previous versions if needed.

Core Features Implementation

This is where you build the main functionality of the image slider. For a Minimum Viable Product (MVP), the core features are the ones that make it a functional slider.

- Image Display: The primary function is to display images. Your HTML will contain a container ( ) where the images will be dynamically inserted using JavaScript. Each image needs to be wrapped in a carousel item container.
- Navigation Controls: You need buttons to move back and forth between images. The provided HTML uses Bootstrap's built-in carousel-control-prev and carousel-control-next buttons. These buttons interact with the carousel's JavaScript to change the active slide.
- Indicators (Pagination): These small dots at the bottom show the user which image they are on and allow for direct navigation to a specific slide. The provided code dynamically generates these based on the number of images, which is a key part of the MVP.

- Dynamic Loading: Instead of hard-coding each image, your JavaScript code will use an array of image objects. The code then loops through this array, creating the necessary HTML elements (div for the slide, img for the image, and div for the caption) for each image. This makes the slider dynamic, so you can easily add or remove images by just editing the array.
- Captions: Displaying a title and description for each image is a core feature that adds context. The carousel-caption div in your code handles this.

Data Storage (Local State / Database)

For a simple MVP, local state is a perfect and straightforward choice.

- Local State: This is what the provided code uses. The image data (source, title, and description) is stored directly in a JavaScript array (const images = [...]). This data is part of the client-side code and is loaded every time the page is accessed. It's the simplest method and is ideal for an MVP.
- Database: For a more advanced version, you'd use a database (like MongoDB or PostgreSQL) to store the image data. You would have a backend server (e.g., Node.js with Express) that fetches the data from the database and sends it to your frontend as a JSON response. This approach is more scalable, allowing you to manage and update images without changing the code itself.

Testing Core Features

Testing is crucial to ensure your slider works as expected. For an MVP, you'll focus on manual testing.

- Functionality Test: Check if the "next" and "previous" buttons work correctly. Verify that the slider loops from the last image back to the first.
- Indicator Test: Click on each indicator dot to ensure it navigates to the correct image. Check that the active indicator dot changes as you slide through the images.
- Responsiveness Test: Resize your browser window to simulate different screen sizes (e.g., mobile, tablet, desktop). Ensure the images and controls scale properly without breaking the layout.
- Edge Case Test: Test what happens when you only have one image. Does the navigation disappear or break? What if the image array is empty? A well-built MVP should handle these cases gracefully.

Version Control (GitHub)

Using GitHub is essential for managing your project and collaborating.

- Committing Changes: Regularly commit your changes with clear, descriptive messages (e.g., "Add dynamic image array" or "Fix caption display"). This creates a history of your project, allowing you to see what changes were made and when.
- Branching: For larger projects or when adding new features, it's a good practice to create a new branch (e.g., git checkout -b new-feature). This allows you to work on the new feature in isolation without affecting the main codebase (main or master branch). Once the feature is complete and tested, you can merge it back into the main branch.
- Pushing to Remote: After committing your changes, push them to your GitHub repository (git push origin main). This backs up your code and makes it accessible to other collaborators.
- README File: Create a README.md file in your repository. This file should contain a project title, a brief description, instructions on how to run the project, and any other relevant information. This makes your project easy for others (and future you) to understand.

Implementation

```
<div id="carouselExampleAutoplaying" class="carousel slide" data-bs-ride="carousel">
 <div class="carousel-inner">
  <div class="carousel-item active">
   <img src="..." class="d-block w-100" alt="...">
  </div>
  <div class="carousel-item">
   <img src="..." class="d-block w-100" alt="...">
  </div>
  <div class="carousel-item">
   <img src="..." class="d-block w-100" alt="...">
  </div>
 </div>
 <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleAutoplaying" data-bs-slide="prev">
```

```
<span class="carousel-control-prev-icon" aria-hidden="true"></span>
<span class="visually-hidden">Previous</span>
</button>
<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleAutoplaying" data-bs-slide="next">
<span class="carousel-control-next-icon" aria-hidden="true"></span>
<span class="visually-hidden">Next</span>
</button>
</div>
```

Demo Output Link:
https://drive.google.com/file/d/1vtEdeo2Dwm0Gqx_h9Tk2rgs1Iniw3Ppd/view?usp=sharing

Output