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	:	15/09/2025

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

SUBMITTED BY,

NAME : Prasanth Narayanan V S

MOBILE NO : 8015928461

Problem Statement :

In the current era of digital plaŋorms, engaging users visually has become essenƟal. StaƟc images on websites are no longer sufficient to capture aΣenƟon. The IBM-FE Dynamic Image Slider project aims to solve this by building an interacƟve, responsive, and dynamic slider that integrates seamlessly with a Node.js REST API.

The problem arises from the need to handle large image sets dynamically, manage performance, and provide smooth transiƟons without compromising load speed. By addressing these challenges, the slider ensures an engaging user experience while allowing easy backend integraƟon.
This also helps businesses showcase products, porŋolios, or adverƟsements effecƟvely. The project emphasizes scalability, modularity, and usability across mulƟple devices and plaŋorms.

Users & Stakeholders

Users:

1. End Users – Individuals browsing a website who will interact with the slider for viewing content.
2. Business Clients – OrganizaƟons that will use the slider to display promoƟonal content, banners, or product images.
3. Developers – Engineers responsible for integraƟng and customizing the slider into applicaƟons.

Stakeholders:

- UI/UXDesigners – Ensure the slider is visually appealing and user-friendly.
- ProductManagers – Define requirements and priориze features.
- MarkeƟng Teams – Use the slider for promoƟons and campaigns.
- Backend Developers – Maintain the Node.js REST API and ensure smooth data flow.
- QA Engineers – Test funcƟonality, performance, and responsiveness.

User Stories

1. As a user, I want to navigate through images easily using arrows, swipe gestures, or auto-play, so that I can view content seamlessly without manual effort.
2. As a user, I want the slider to be responsive so that I can view images properly across desktop, tablet, and mobile.
3. As a developer, I want the slider to fetch images dynamically from a Node.js API so that updates can be managed without redeployment.
4. As a business owner, I want to highlight promoΘonal content using the slider so that users are aΣracted to the latest offerings.
5. As a designer, I want to customize the slider's transiΘons, themes, and animaΘons so that it matches the website's branding.
6. As an accessibility advocate, I want keyboard navigaΘon and alt text enabled so that differently-abled users can also use the slider.

MVP Features

FuncΘonal Features:

- Responsive slider design that adapts to different devices.
- Dynamic loading of images via Node.js REST API.
- Auto-play feature with configurable Θme intervals.
- Manual navigaΘon using arrows, dots, and swipe gestures.
- Smooth CSS/JS-based transiΘons.
- Accessibility compliance (keyboard navigaΘon, alt aΣributes).

Non-FuncΘonal Features:

- High performancewith minimal load Θmes.
- Scalability to support large sets of images.
- Cross-browser compaΘbility.
- Secure API integraΘon to handle image data safely.

Wireframes / API Endpoint List

Wireframe Concept:

- A rectangularimage display area with leŌ and right arrows for navigaƟon.
- Small dots atthe boΣom indicaƟng the posiƟon of the current slide.
- OpƟonfor autoplay with pause on hover.

API Endpoints:

1. GET /api/images→ Returns a JSON list of image URLs and metadata.
   Example Response: { 'id':1, 'url':'/images/img1.jpg', 'Ɵtle':'Image 1' }
2. POST /api/images → Allows uploading a new image with metadata.
   Example Input: { 'url':'/images/img2.jpg', 'Ɵtle':'Image 2' }
3. PUT /api/images/:id → Updates image details such as Ɵtle or descripƟon.
4. DELETE /api/images/:id → Deletes an image by ID.

The endpoints ensure CRUD funcƟonality for managing images dynamically.

Acceptance Criteria

1. The slider must fetch and display images dynamically from the Node.js API without manual refresh.
2. It should work on all major devices (desktop, tablet, mobile) with proper responsiveness.
3. Auto-play should transiƟon slides smoothly within the configured interval.
4. Users must be able to pause, navigate forward, and backward using buΣons or gestures.
5. API must handle errors gracefully (e.g., missing or broken links should not crash the slider).
6. Accessibility standards must be met (alt text, ARIA roles, keyboard navigaƟon).
7. Performance benchmarks: The slider must load within 2 seconds and handle at least 50 images efficiently.
8. Cross-browser tesƟng: FuncƟonality must be consistent across Chrome, Firefox, Safari, and Edge.