SpacEd

Live Demo

REPORT

Web Application

Rathnayake R.M.U.V.
IT21271182

Table of contents

Introduction

The SpacEd web application was conceived with the vision of creating an immersive platform for space education and exploration. Built on the foundation of accessibility, interactivity, and educational value, SpacEd aims to provide users with a captivating journey through the wonders of the cosmos.

Integrating data from NASA's vast repositories of space-related imagery and information, SpacEd leverages the power of NASA APIs to offer users access to a wealth of curated content, including daily astronomical images, Earth views from space, and stunning imagery captured by Mars rovers.

SpacEd

Live Demo

A live demo of the SpacEd web application is available at :

- https://spaced-83610.web.app/
- https://spaced-83610.firebaseapp.com /

Chosen APIs

1. NASA's Astronomy Picture of the Day (APOD) API

The APOD API provides a curated collection of astronomical images and information, showcasing a different astronomical image or photograph each day, along with a brief explanation written by a professional astronomer.

This API enriches the SpacEd application by offering users a daily dose of stunning imagery and educational content related to space exploration. It allows users to explore and learn about various celestial phenomena, fostering a deeper appreciation for the wonders of the universe.

2. NASA's Earth Polychromatic Imaging Camera (EPIC) API

The EPIC API offers access to imagery captured by the Earth Polychromatic Imaging Camera, a camera on board the NOAA's Deep Space Climate Observatory (DSCOVR) satellite. EPIC captures images of the Earth from a unique vantage point in space, providing views of the sunlit side of our planet from afar.

By integrating the EPIC API, SpacEd enables users to view and marvel at breathtaking views of Earth from space, offering a perspective that is both awe-inspiring and educational. These images serve as a reminder of the fragility and beauty of our home planet.

3. NASA's Mars Rover Photos API

The Mars Rover Photos API provides access to a vast collection of images captured by NASA's various Mars rovers, including Curiosity, Opportunity, and Perseverance. These images offer insights into the Martian landscape, geological features, and scientific discoveries made by the rovers.

Integrating the Mars Rover Photos API allows users of SpacEd to explore the surface of Mars through the lens of NASA's rovers, offering a glimpse into ongoing missions and discoveries on the Red Planet. Users can immerse themselves in the exploration of Mars and gain a better understanding of our neighboring planet.

Importance of Chosen APIs in SpacEd

The chosen APIs collectively enhance the educational value of the SpacEd application by providing users with access to a wealth of space-related imagery, information, and educational content.

By offering a diverse range of content, including daily astronomical images, Earth views from space, and Mars rover imagery, SpacEd aims to engage users of all ages and backgrounds in the wonders of space exploration.

The imagery and information provided by the APIs serve to inspire curiosity, wonder, and a sense of awe about the universe, encouraging users to explore further and learn more about space science and exploration.

Project Overview

The development of SpacEd was divided into several key phases, each focusing on different aspects of the application's functionality and user experience. From conceptualization and planning to implementation and testing, every stage of the project was meticulously executed to ensure the delivery of a high-quality and engaging web application.

Phase 1: Planning and Research

The project kicked off with an extensive planning phase, during which the scope, requirements, and technical specifications of SpacEd were defined. Research was conducted to identify suitable APIs and technologies for implementing the desired features, with a focus on accessibility, data availability, and ease of integration.

Phase 2: Design and Prototyping

With a clear understanding of the project requirements in place, the design and prototyping phase commenced. User interface wireframes and mockups were created to visualize the layout, navigation, and overall look and feel of the application. Feedback from stakeholders and usability testing guided iterative improvements to the design.

Phase 3: Frontend Development

Frontend development formed the core of the project, with a focus on implementing the user interface, integrating APIs, and ensuring a seamless browsing experience across devices. React.js, Vite, and Tailwind CSS with Flowbite were chosen for their flexibility, performance, and extensive ecosystem of libraries and tools.

Technologies Used:

- **React.js:** Chosen for its component-based architecture, state management capabilities, and virtual DOM rendering, React.js provided a solid foundation for building dynamic and responsive user interfaces.
- **Vite:** Leveraged as the frontend build tool for its blazing-fast development server, rapid hot module replacement (HMR), and efficient bundling capabilities, Vite enabled speedy iteration and deployment of frontend code.
- **Tailwind CSS with Flowbite:** Utilized for its utility-first approach to styling, Tailwind CSS combined with Flowbite offered a comprehensive set of pre-designed components and utilities for rapid UI development, ensuring consistency and maintainability across the application.

Phase 4: Backend Integration and Data Management

Integration with external APIs, including NASA's APOD, EPIC, and Mars Rover Photos APIs, presented a significant challenge during the backend integration phase. Strategies were devised to efficiently fetch, parse, and display data from these APIs while adhering to best practices for data management and API usage.

Phase 5: Testing and Quality Assurance

Thorough testing and quality assurance were conducted throughout the development process to identify and address any bugs, issues, or inconsistencies. Using **Jest** framework Unit tests and integration tests testing were performed to validate the functionality, performance, and usability of the application across different browsers and devices.

Phase 6: Deployment and Launch

Upon successful completion of development and testing, the SpacEd web application was deployed to a production environment using **Firebase** Hosting. A comprehensive deployment checklist was followed to ensure a smooth and error-free launch, with monitoring and analytics tools put in place to track user engagement and performance metrics.

Challenges Faced and Resolutions

1. Fetching Details from APIs

One of the main challenges encountered during the development of SpacEd was efficiently fetching data from the NASA APIs, especially due to the diverse nature of data formats returned by different endpoints. To overcome this challenge, a robust data parsing and handling mechanism was implemented, ensuring compatibility with various API responses.

2. Filtering Options in Mars Rover Photos API

Implementing filtering options, such as selecting photos by sol (Martian solar day) or camera type, posed a significant challenge while integrating the Mars Rover Photos API. To resolve this, extensive research was conducted on the API documentation, followed by the implementation of custom filtering logic within the application, providing users with a seamless experience in exploring Mars rover images.

3. Fetching Images from the EPIC API

Fetching images from the EPIC (Earth Polychromatic Imaging Camera) API presented a unique challenge due to the complex data structure and retrieval process involved. To address this, a comprehensive data parsing and image extraction mechanism was developed, ensuring accurate retrieval and display of EPIC images within the SpacEd application.

4. Authentication with Firebase

Integrating Firebase authentication posed a challenge due to the need to handle user authentication securely while ensuring a smooth user experience. To resolve this, Firebase authentication methods were thoroughly researched and implemented, including email/password authentication and Google sign-in. Additionally, error handling mechanisms were put in place to provide informative feedback to users during the authentication process, ensuring a seamless onboarding experience.

5. Optimizing Image Loading Performance

Fetching and displaying high-resolution images from NASA APIs, especially for large datasets like Mars Rover Photos, presented a performance challenge. To address this, various image optimization techniques were explored and implemented, such as lazy loading, caching mechanisms, and dynamic resizing based on device resolution. These optimizations significantly improved the loading speed and responsiveness of the application, enhancing the overall user experience.

6. Handling Asynchronous Data Fetching

Managing asynchronous data fetching and ensuring data consistency across different components of the application proved challenging, especially when dealing with multiple API endpoints and complex data dependencies. To tackle this, modern asynchronous JavaScript techniques such as async/await and Promise chaining were utilized to orchestrate data fetching operations efficiently.

7. Managing API Rate Limits

Managing API rate limits imposed by NASA's APIs posed a challenge, especially during periods of heavy traffic or when making bulk data requests. To manage this, rate-limiting mechanisms were implemented within the application to throttle API requests and prevent excessive usage, thus ensuring compliance with API usage policies and maintaining service availability for all users. Additionally, caching strategies were employed to minimize redundant API calls and optimize resource utilization, further mitigating the impact of rate limits on application performance.

Setup Guide

To set up the project locally, follow these steps:

Prerequisites

Ensure you have the following prerequisites installed:

- Node.js
- npm

Installation

To set up the project locally, follow these steps:

1. **Clone the repository:**

   ```
   git clone https://github.com/your-username/spaced.git
   ```

2. **Navigate to the project directory:**

   ```
   cd spaced
   ```

3. **Install dependencies:**

   ```
   npm install
   ```

Configuration

Before running the application, you need to obtain an API key from NASA and configure Firebase for authentication and hosting. Follow these steps:

Obtain NASA API Key:

- Go to the NASA API website: https://api.nasa.gov/
- Sign up for an account and obtain your API key.

Configure Firebase:

- Go to the Firebase Console: https://console.firebase.google.com/.
- Create a new project and follow the setup instructions.
- Enable Authentication and choose the authentication methods (email/password, Google sign-in).
- Obtain your Firebase configuration settings from the Firebase project settings.
- Add API Key to Environment Variables:
- Create a .env file in the project root.

- Add the following environment variables:

  VITE_NASA_API_KEY=your-nasa-api-key

  VITE_FIREBASE_API_KEY=your-firebase-api-key

  VITE_FIREBASE_AUTH_DOMAIN=your-firebase-auth-domain

  VITE_FIREBASE_PROJECT_ID=your-firebase-project-id

  VITE_FIREBASE_APP_ID=your-firebase-app-id

  VITE_FIREBASE_STORAGE_BUCKET=your-firebase-storage-bucket

  VITE_FIREBASE_MESSAGING_SENDER_ID=your-firebase-messaging-sender-id

  VITE_FIREBASE_MEASUREMENT_ID=your-firebase-measurement-id

- Replace your-nasa-api-key with the API key obtained from NASA.
- Replace your-firebase-api-key, firebase-auth-domain, your-firebase-project-id, your-firebase-app-id, firebase-storage-bucket, your-firebase-messaging-sender-id, your-firebase-measurement-id with your Firebase configuration settings.

Usage

4. **To run the project locally:**

   ```
   npm run dev
   ```

   This will start the development server.

Build

To build the project for production:

```
npm run build
```

This will generate the production build in the dist directory.

Testing

This project utilizes Jest for testing. To run the tests, use the following command:

```
npx jest
```

This will execute the test suites and provide feedback on the code's functionality and correctness.

Deployment

To deploy the project to Firebase Hosting:

1. **Install Firebase CLI globally:**

   ```
   npm install -g firebase-tools
   ```

2. **Log in to Firebase:**

   ```
   firebase login
   ```

This will redirect to firebase login. login using your account.

3. **Initialize Firebase in your project:**

   ```
   firebase init
   ```

Follow the prompts to select Firebase Hosting and choose your project.

4. **Deploy the project:**

   ```
   firebase deploy
   ```

   After deployment, you will get the hosting URL where your app is hosted.