**Frontend Development with React.js**

**Project Documentation**

**Introduction**

Project Title: React News App - Insight Stream

**Team Members**:

- Vivek.S - Frontend Development & Project Lead [Mail: [svvk2004@gmail.com](mailto:svvk2004@gmail.com) ]

- Adithyan.D – Design & Testing [Mail: [adhidoss32@gmail.com](mailto:adhidoss32@gmail.com) ]

-Dhanasekar.M - Design & API Integration [mail: [Sekardhanasekar596@gmail.com](mailto:Sekardhanasekar596@gmail.com) ]

-Vinayagamoorthi.A – Design & Testing [Mail: [avinayagamoorthi96@gmail.com](mailto:avinayagamoorthi96@gmail.com) ]

**Project Overview**

**-Purpose:** The React News App is designed to provide users with a clean, responsive interface to browse, search, and read the latest news articles from various sources. The goal is to create an intuitive and fast frontend experience powered by a third-party news API (e.g., NewsAPI).

**- Features:**

- Display a list of trending news articles.

- Search functionality to filter news by keywords.

- Category-based news browsing (e.g., Technology, Sports).

- Responsive design for desktop and mobile users.

- Article detail view with a link to the original source.

**Architecture**

**Component Structure:**

- `App.js`: Root component that sets up routing and global state.

- `Navbar`: Navigation bar for accessing search and categories.

- `NewsList`: Displays a list of news articles fetched from the API.

- `NewsCard`: Reusable component for individual news article summaries.

- `NewsDetail`: Displays the full details of a selected article.

- `SearchBar`: Handles user input for news searches.

**State Management:**

- \*\*Global State:\*\* Managed using the Context API to store fetched news data, search queries, and selected categories.

- \*\*Local State:\*\* Individual components (e.g., `SearchBar`) use React’s `useState` hook for local form inputs or UI toggles.

**Routing:**

- Uses `react-router-dom` for client-side routing:

- `/` - Home page with trending news.

- `/category/:categoryName` - News filtered by category.

- `/article/:articleId` - Detailed view of a specific article.

**Setup Instructions**

**Prerequisites:**

- Node.js (v16 or higher)

- npm or Yarn

- A News API key (e.g., from [NewsAPI.org](https://newsapi.org/))

**Installation:**

1. Clone the repository:

```bash

git clone https://github.com/[your-username]/react-news-app.git

```

2. Navigate to the project directory:

```bash

cd react-news-app

```

3. Install dependencies:

```bash

npm install

```

4. Create a `.env` file in the root directory and add your API key:

```

REACT\_APP\_NEWS\_API\_KEY=your\_api\_key\_here

```

5. Start the development server:

```bash

npm start

**Folder Structure**

**Client:**

src/

├── components/ # Reusable UI components

│ ├── Navbar.js

│ ├── NewsCard.js

│ ├── NewsList.js

│ └── SearchBar.js

├── pages/ # Page-level components

│ ├── Home.js

│ ├── Category.js

│ └── NewsDetail.js

├── assets/ # Images, fonts, etc.

│ └── logo.png

├── context/ # Context API for global state

│ └── NewsContext.js

├── utils/ # Utility functions

│ └── api.js # API fetch logic

├── App.js # Root component

└── index.js # Entry point

**Utilities:**

- `api.js`: Contains helper functions to fetch data from the News API, including error handling and caching logic using `localStorage`.

- Custom Hook: `useNewsFetch` - A hook to manage API calls and loading states.

**Running the Application**

**Frontend:**

- Run the app locally:

```bash

npm start

```

- The app will be available at `http://localhost:3000`.

**Component Documentation**

**Key Components:**

- \*\*`Navbar`:\*\*

- \*\*Purpose:\*\* Provides navigation links (Home, Categories) and houses the `SearchBar`.

- \*\*Props:\*\* None (uses Context for search state).

- \*\*`NewsList`:\*\*

- \*\*Purpose:\*\* Renders a grid or list of `NewsCard` components based on fetched data.

- \*\*Props:\*\* `articles` (array of news objects).

- \*\*`NewsCard`:\*\*

- \*\*Purpose:\*\* Displays a single news article summary (title, image, source, date).

- \*\*Props:\*\* `title`, `imageUrl`, `source`, `publishedAt`, `articleId`.

**Reusable Components:**

- \*\*`SearchBar`:\*\*

- \*\*Purpose:\*\* Allows users to input search terms and trigger API requests.

- \*\*Configuration:\*\* Uses `onSearch` callback to pass search terms to parent components.

**State Management**

**Global State:**

- Managed via `NewsContext`:

- `articles`: Array of fetched news articles.

- `category`: Current selected category (e.g., "technology").

- `searchQuery`: User-entered search term.

- `loading`: Boolean to show/hide loading spinner.

**Local State**:

- `SearchBar` uses `useState` to manage the input field value.

- `NewsDetail` uses `useState` to toggle a "Read More" button if applicable.

---

**User Interface**

- \*\*Home Page:\*\* Displays trending news in a grid layout.

- \*\*Category Page:\*\* Filters news by category with a dropdown or sidebar.

- \*\*Search:\*\* Updates `NewsList` dynamically as users type.

- \*\*Screenshots:\*\*

\*(In a real project, include screenshots of the Home page, Search in action, and Article detail view here.)\*

**Styling**

\*\*CSS Frameworks/Libraries:\*\*

- Uses `Tailwind CSS` for utility-first styling.

- Custom styles in `index.css` for global resets and minor overrides.

**Testing**

**Testing Strategy:**

- \*\*Unit Tests:\*\* Test individual components (e.g., `NewsCard`) using Jest and React Testing Library.

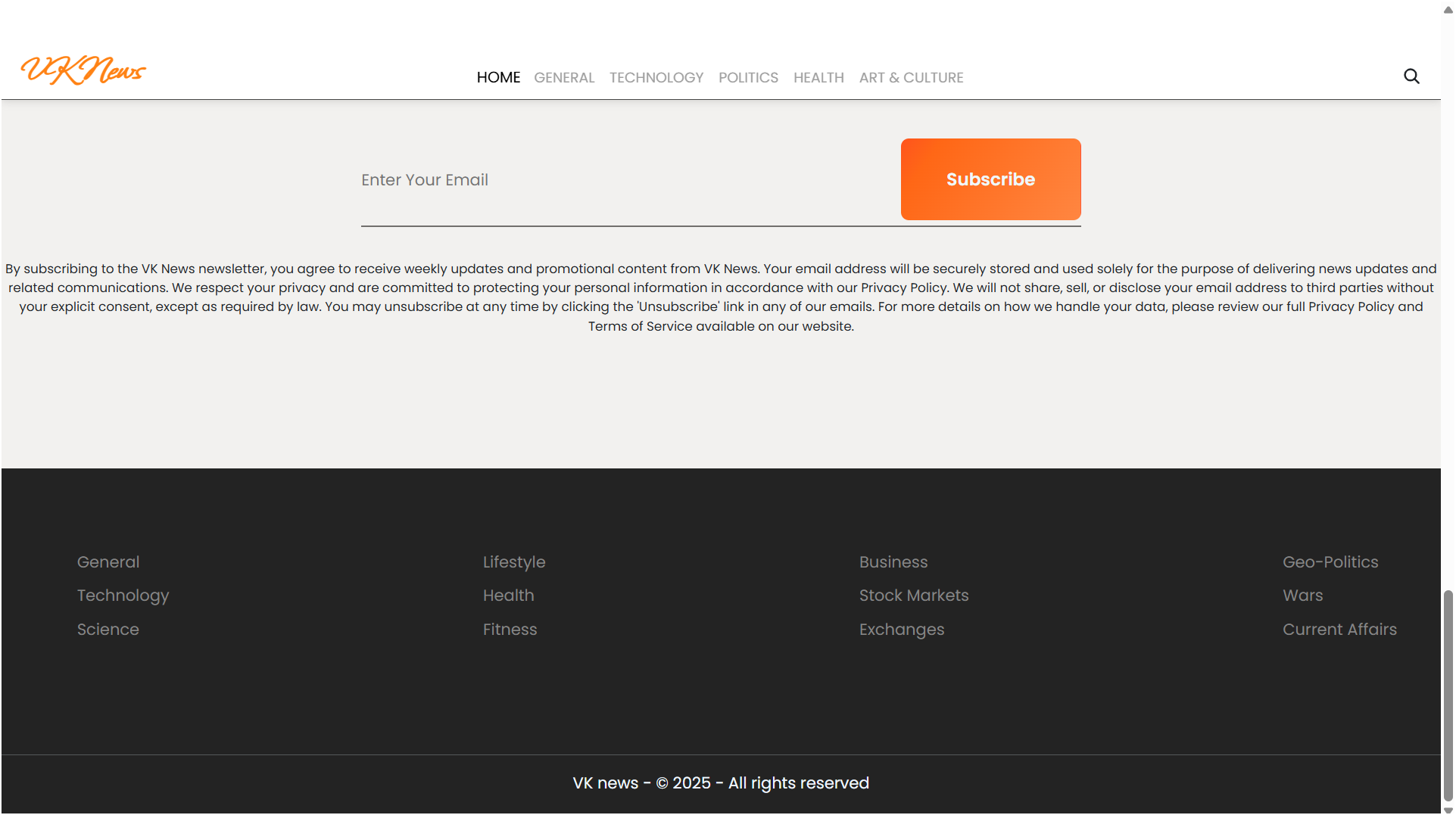
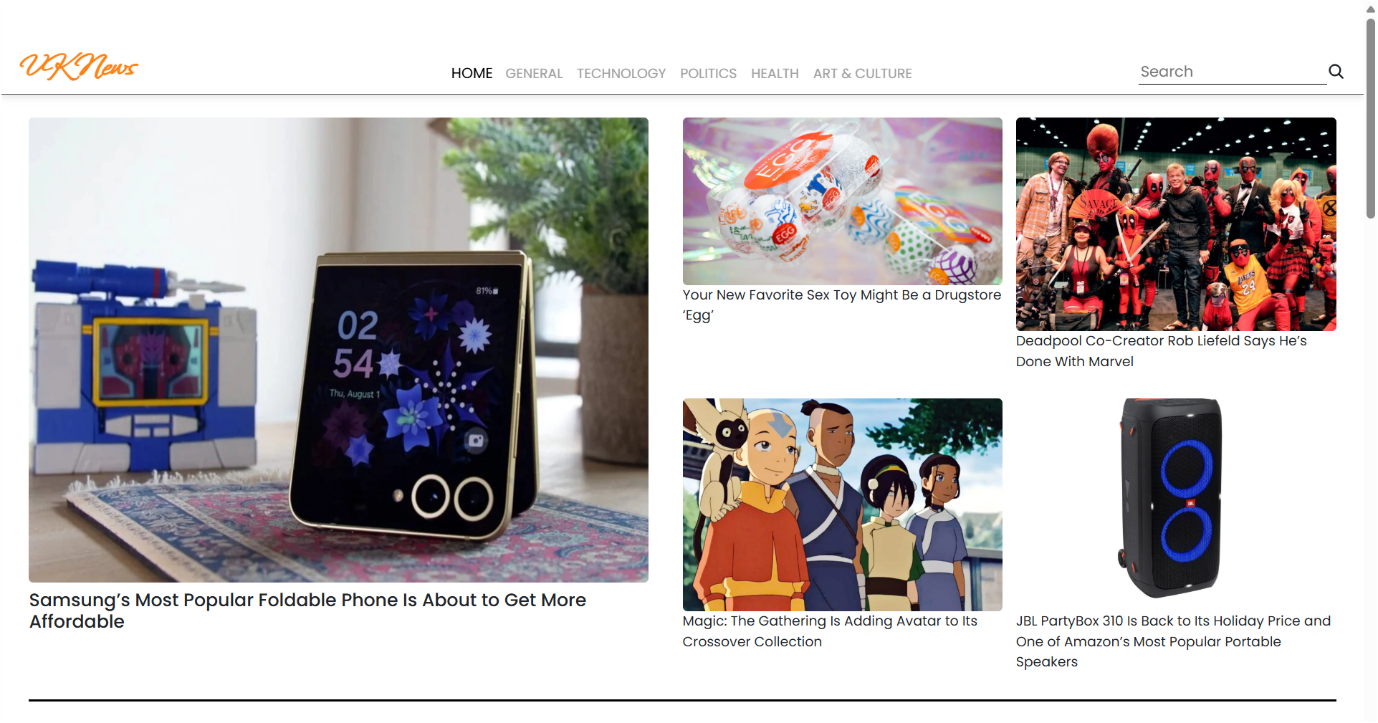
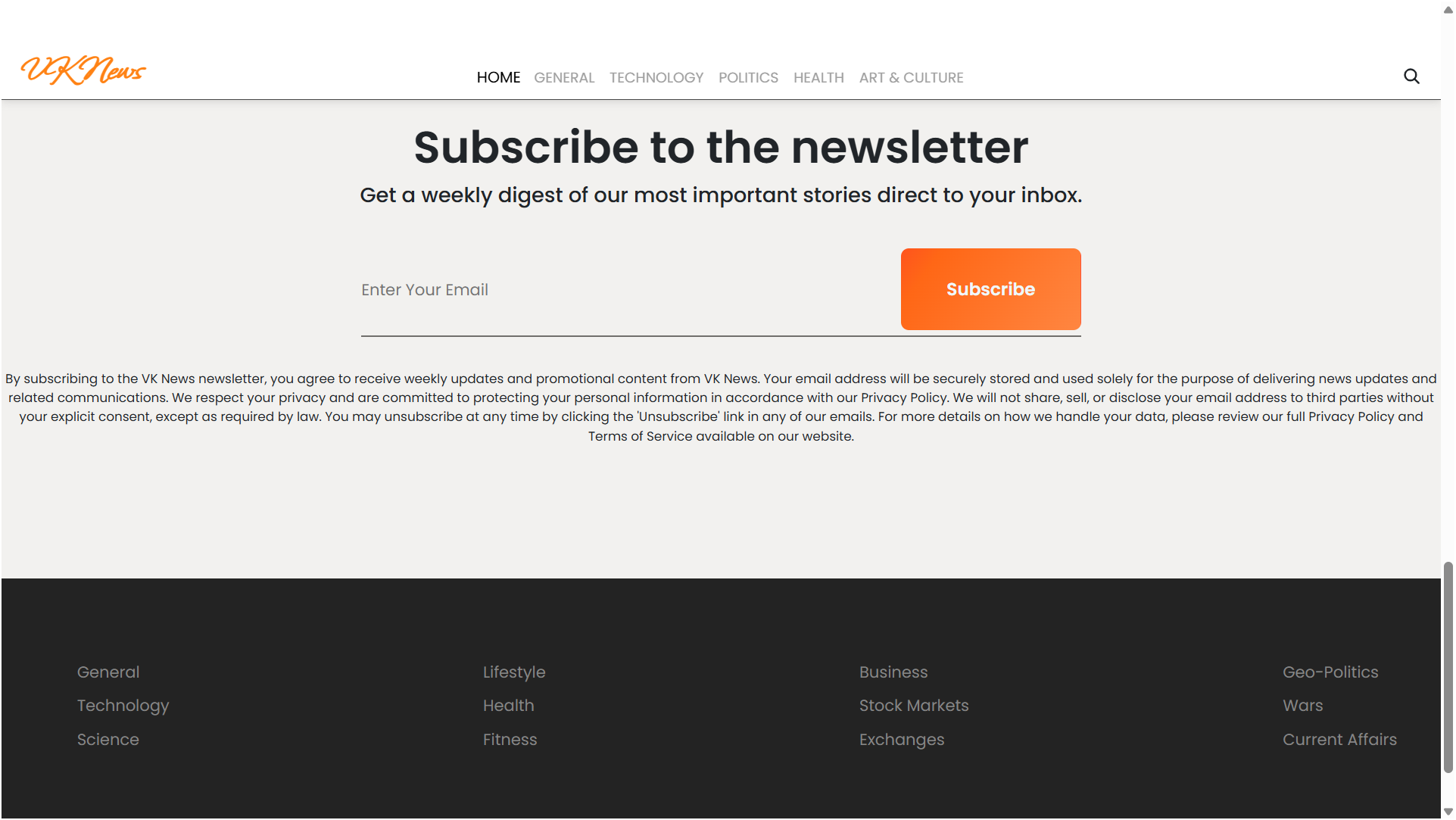
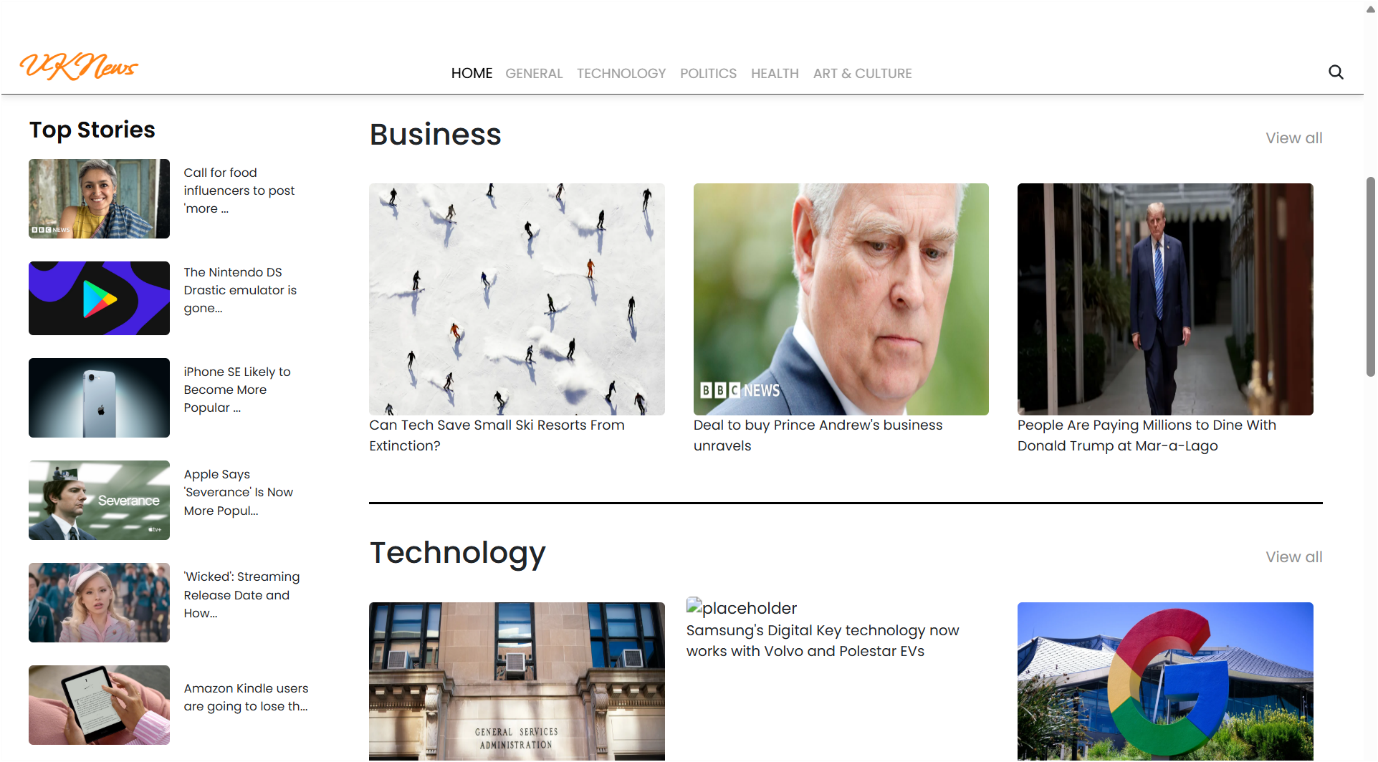
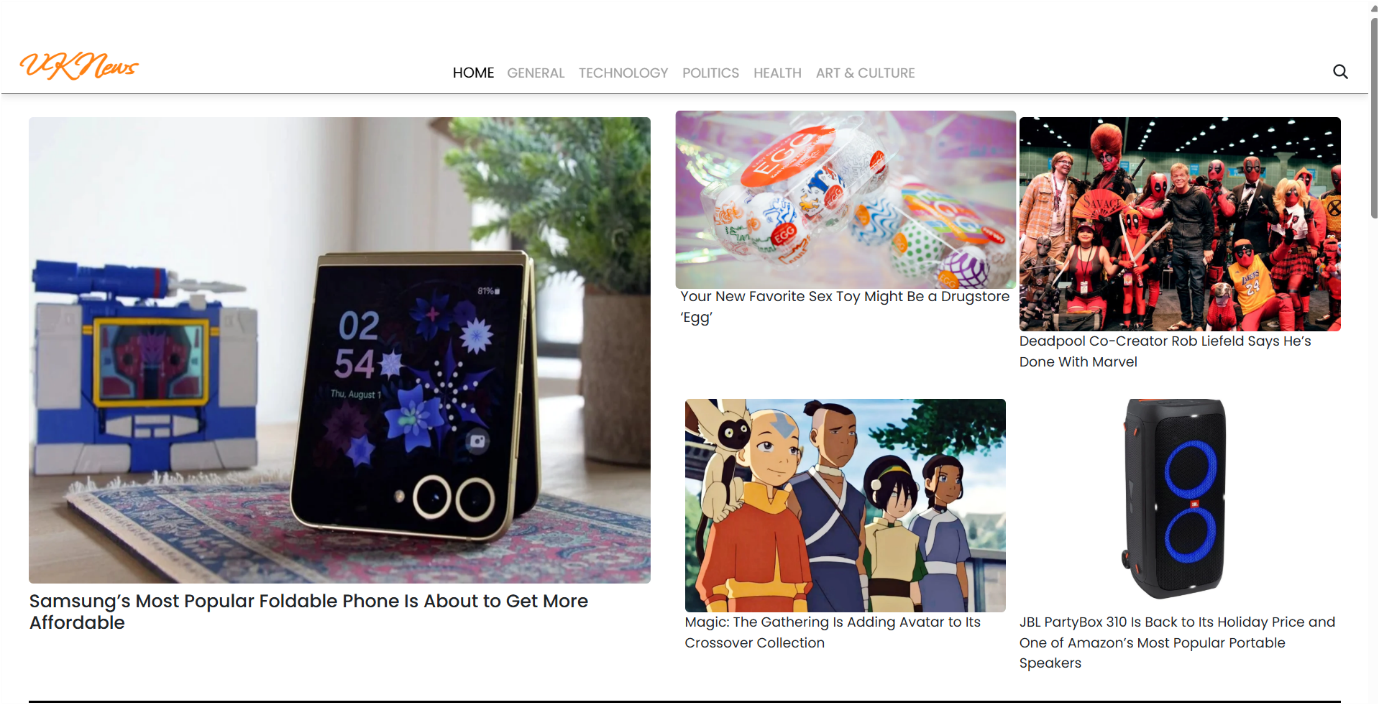
- \*\*Integration Tests:\*\* Ensure `NewsList` renders multiple `NewsCard` components correctly with mock API data.

- \*\*End-to-End Tests:\*\* Use Cypress to simulate user flows (e.g., search → view article).

**Code Coverage:**

- Aim for 80%+ coverage using Jest’s built-in coverage reports.

**Screenshots**



**Known Issues**

- API rate limits may cause fetch errors during heavy testing (mitigated with caching).

- Mobile responsiveness breaks slightly on very small screens (<320px).

- Search debounce not fully optimized for rapid typing.

**Future Enhancements**

- Add offline support using Service Workers.

- Implement infinite scroll for news articles.

- Enhance styling with animations (e.g., fade-in for `NewsCard`).

- Add user authentication to save favorite articles.