**PROJECT NAME :INSIGHTSTREAM:**

**NAVIGATE THE NEWS LANDSCAPE**

**(NEWS APP)**

**TEAM MEMBERS AND THEIR ROLES:**

**C.KAVIYA (TEAM LEADER AND TESTER)**

**Y.HARI PRIYA (FRONTEND DEVELOPER)**

**M.JAYASRI(BACKEND DEVELOPER)**

**V.JAYASHREE(MANAGES THE EXTERNAL API AND EDITING)**

**PROJECT OVERVIEW:**

**PURPOSE:**

The News Application is developed to provide users with real-time updates on the latest news across multiple categories such as business, sports, technology, and politics. The goal of the project is to deliver a simple, efficient platform for users to access news content conveniently, categorized by interest.

**FEATURES:**

Real-Time News Updates: Fetches the latest news articles dynamically using JavaScript.

**Category Filtering:** Users can filter news based on specific categories like business, sports, technology, and politics.

**Responsive Design:** Ensures the application works seamlessly across different devices, including desktops, tablets, and mobile phones.

**User-Friendly Interface:** Simple navigation and clean design for an intuitive user experience.

**Fast and Lightweight:** The application is built using HTML, CSS, and JavaScript, making it efficient and fast to load.

**ARCHITECTURE:**

The architecture of the News Application describes how the main components interact and how the overall system is structured. It is divided into the following key areas:

**1. COMPONENT STRUCTURE:**

The News Application is structured with separate HTML files for different pages and sections (e.g., homepage, category-specific pages). The content and layout are controlled using CSS, while the functionality—like fetching news data and updating the UI—is handled using JavaScript. The main components include:

**Header:** Contains navigation links to different news categories.

**Main Content:** Displays news articles dynamically based on the selected category.

**Footer:** Provides additional navigation or informational links.

**2.STATE MANAGEMENT:**

Since the project uses JavaScript, it manages data by fetching news articles from an external API and stores them in variables. The state (current category, articles) is updated every time the user interacts with the app (e.g., switching categories), and the UI is updated accordingly. No external state management libraries (like Redux) are required, as the state management is simple and localized.

**3.ROUTING:**

The application does not use advanced routing frameworks. Instead, category-based navigation is managed through JavaScript, dynamically updating the displayed content without reloading the page. The user can switch between categories (e.g., Business, Sports, Technology) by clicking on the links in the header, which triggers the appropriate data fetch and content rendering.

**SETUP INSTRUCTIONS:**

**1. PREREQUISTIES:**

A modern web browser (e.g., Chrome, Firefox) to view the application.

**SOFTWARE DEPENDENCIES:**

**Visual Studio Code:** Code editor to write and manage your project files.

**Node.js:** For handling any backend or build processes (if applicable).

**React:** If you plan to enhance the project using React.js.

**Git:** To clone repositories and manage version control.

**Live Server (VS Code Extension):** For running your project locally with live reloading.

**2.INSTALLATION:**

Install Visual Studio Code, Node.js, React, and Git if not already installed.

* Clone the project repository using Git: git clone <repository-url>.
* Open the project folder in Visual Studio Code.
* Run npm install to install necessary packages (if any).
* Launch the project using Live Server in VS Code or open index.html directly in a browser.

This version includes all the software dependencies for setting up and running the project properly.

**FOLDER STRUCTURE:**

**1. Root Directory:**

**index.html:** The main entry point of the application, where the basic structure of the webpage is defined.

**logo.png / logo.svg:** These image files represent the application’s logo and are used in the header or favicon.

**manifest.json:** Contains metadata for the web app (like app name, icons, and theme) for Progressive Web App (PWA) support.

**2. Source Files:**

**newsletter.jsx:** A React component that manages the newsletter section of the application.

**pages.jsx:** A React component that handles the rendering of different news categories or pages.

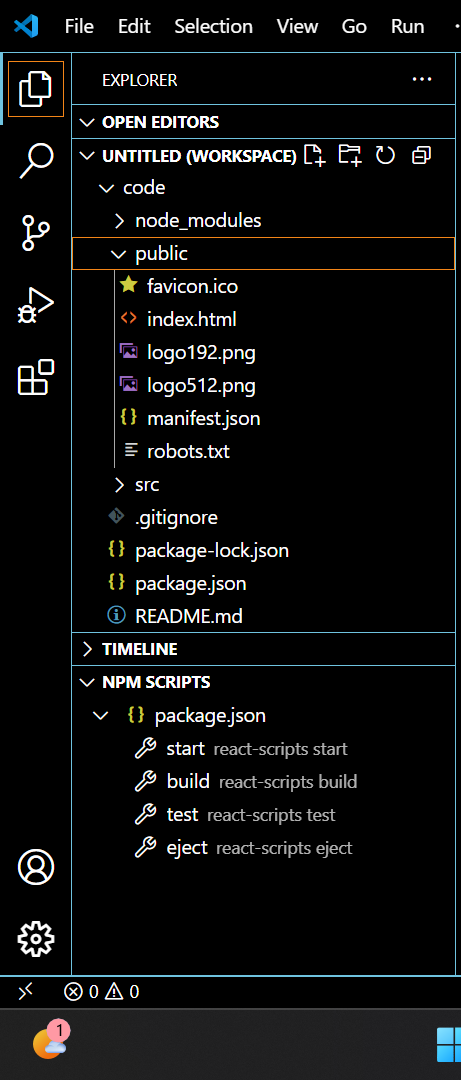
**app.css:** The main stylesheet that controls the visual layout and styling of the entire application.

**3. Node Modules:**

A directory that contains all the dependencies and packages installed via npm (Node Package Manager). These are necessary for running and building the React-based project.

**package.json:** Lists the project's dependencies, scripts, and metadata. It is used by npm to manage the project’s packages

This folder structure makes the project organized and easy to navigate, especially when working with React and Node.js.



**RUNNING THE APPLICATION:**

To run the News Application locally, follow these steps:

**1. Install Dependencies:**

Open a terminal or command prompt in the project directory and run the following command to install all the necessary packages and dependencies:

**npm install**

**2. Start the Application:**

After the installation is complete, run the following command to start the development server:

**npm start**

This will start a local development server, and the application will be accessible at http://localhost:3000/ (or the URL specified in the terminal).

**3. Frontend:**

The main application is located in the src directory. Ensure that the index.html or main entry point is set up properly to load the React components.

**COMPONENT DOCUMENTATION:**

**the component documentation should include information about the key components used in your React (or JavaScript) setup. Based on the folder structure shown in the image, here’s how you can document the **components:****

****1. Key Components:****

****App.js (or App.jsx):** The main component that serves as the root of the application. It is responsible for rendering other components like the header, news sections, and footer.**

****Header.js:** This component contains the navigation links (categories like Business, Sports, Technology) and the logo. It allows users to switch between different sections of the news application.**

****NewsList.js:** This component fetches and displays the list of news articles. It dynamically updates based on the selected category (e.g., business news, sports news).**

****Footer.js:** Contains additional information or links (e.g., contact, about).**

****2. Reusable Components:****

****NewsCard.js:** A reusable component to display individual news articles in a card format. It shows the title, description, and a link to the full article. This component is used multiple times to render each news item.**

****CategoryFilter.js:** A component that allows users to filter news based on the category (e.g., Business, Sports). It re-renders the NewsList component based on the selected category.**

**Each of these components has a specific role in making the application modular and easy to manage.**

**STATE MANAGEMENT:**

State management refers to how state is handled within a React application. It helps to store, manage, and pass data between components. In React, state can be managed at two levels:

**1. Local State:**

Managed within a specific component using the useState hook.

The state affects only that component and is not shared.

**2. Global State:**

Used to manage data that needs to be accessed by multiple components across the application.

**Common approaches include:**

**Context API:** A built-in React feature for sharing state across components.

**Redux:** A more structured state management library, useful for larger application.

**USER INTERFACE:**

**1. UI Design Overview:** Describe the overall look and feel of the interface, including the layout and theme (color scheme, fonts, etc.). Explain if the design follows any specific design principles or is inspired by any popular UI frameworks.

**2. Main Components:**Header and Navigation: Explain how the header contains your branding, like the logo, and navigation links such as Home, General, Technology, Politics, etc., which users can click to browse different sections.

**Content Display:** Discuss the main content area where news articles, images, and headlines are displayed. Explain how the articles are laid out and how the content is dynamic (i.e., if it updates based on the selected category).

**Newsletter Section:** You have a subscribe to the newsletter section where users can enter their email and click on a button to subscribe. This part is important for user interaction.

**Footer:** Detail what links or information are provided in the footer (e.g., categories like Lifestyle, Business, Geo-Politics).

**3.Interaction Elements:** Explain the interactive components such as buttons, clickable images, forms, and how users can navigate or interact with the site. For example:

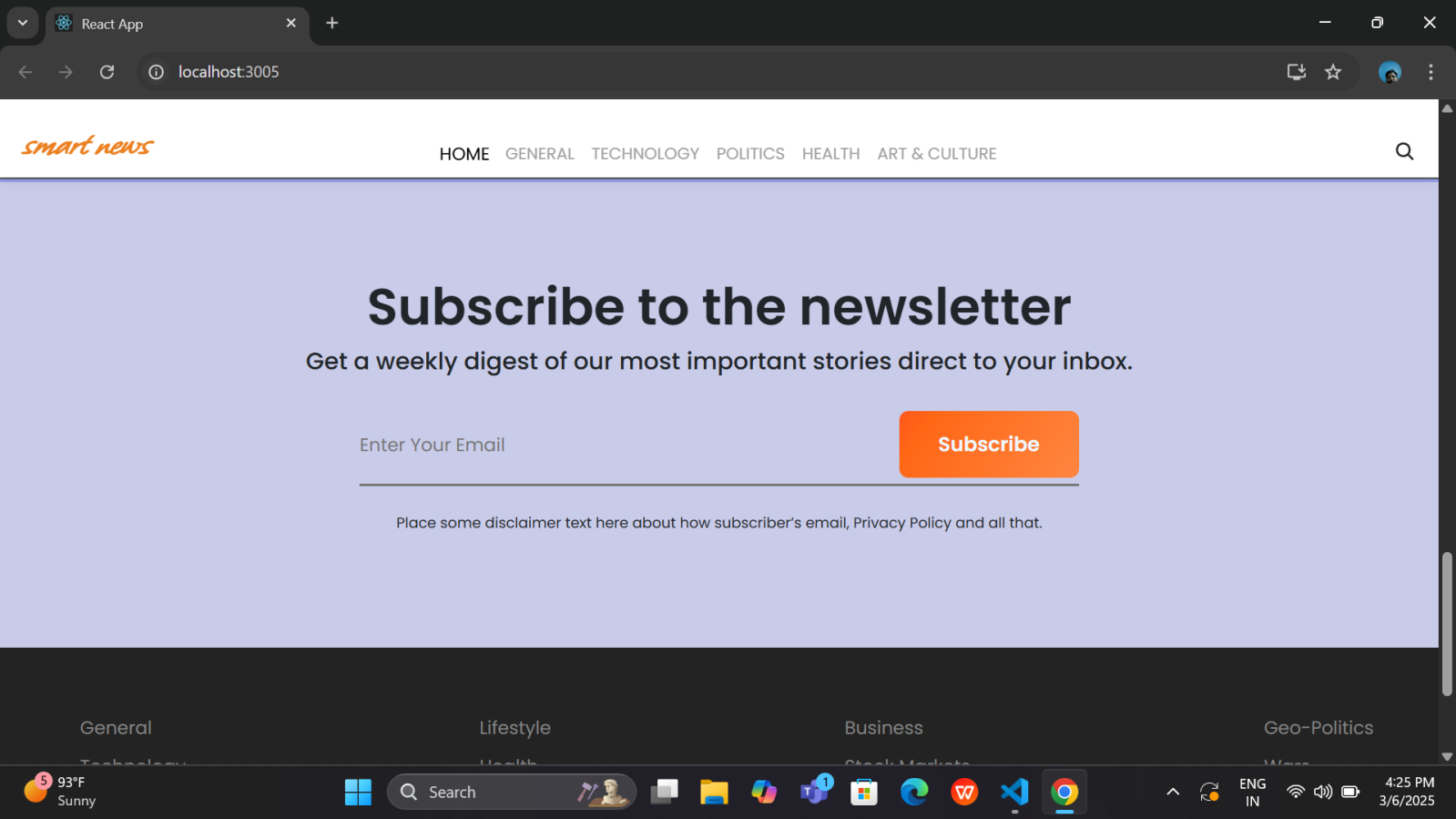
**Subscribe Button:** Clicking this triggers an email submission process, followed by confirmation.

**Links and Buttons:** Describe how users interact with buttons like the “subscribe” button or news categories.

**Search Functionality:** If there is a search feature, describe its functionality and how it enhances user experience.

**4. Responsiveness:** Mention if your application is responsive, i.e., whether it adjusts to different screen sizes (mobile, tablet, desktop).

This section is essential because it helps other developers and users understand how to navigate and interact with your application, ensuring a smooth user experience.



**STYLING:**

**1. CSS Frameworks/Libraries:**

the styling is implemented using a combination of CSS frameworks and custom styles. The following tools and libraries are used:

**CSS Frameworks:** If you are using a CSS framework like Bootstrap, Material UI, or Tailwind CSS, describe its purpose and how it's integrated into the project.

**Bootstrap:** For creating responsive layouts, components like buttons, forms, and navigation bars.

**Material UI:** For predefined UI components that follow Material Design guidelines.

**Tailwind CSS:** For utility-first CSS classes, allowing rapid UI development.

**CSS Libraries:** If using libraries like Styled-Components or Emotion, explain how they help with component-level styling.

**Styled-Components:** A CSS-in-JS library that allows you to write actual CSS code within your JavaScript files. This library is used to create reusable, isolated styled components that can be customized using props.

**Sass:** A preprocessor that adds features to CSS like variables, nested rules, and mixins, which helps in writing cleaner and more organized stylesheets.

**2. Theming:**

**Custom Design System:** If the project follows a custom design system, describe the guidelines set for consistent design.

**Color Palette:** Define a color scheme for the project (e.g., primary, secondary, accent colors). For example, Orange for the branding (logo), light purple for the background, and dark text for headings.

**Typography:** Explain the font-family choices and sizes, such as using Roboto or Open Sans for a clean and modern look.

**Spacing and Layout:** Mention if you follow any specific grid or spacing system to ensure uniform margins, padding, and overall alignment.

**Dark Mode/Light Mode:** If the project supports theming, mention how users can switch between themes (e.g., light and dark modes). Describe if there is a global theming context or hook used in React to toggle themes.

**TESTING:**

**1. Testing Strategy:**

For ensuring the correctness and reliability of the application, various levels of testing are implemented. Here’s how testing is approached:

**Unit Testing:**

**Purpose:** Focus on testing individual components and functions to ensure they work as expected in isolation.

**Tool Used:** Jest and React Testing Library.

**Jest:** Used as a test runner to execute unit tests efficiently.

**React Testing Library:** Helps in testing React components by simulating user interactions (e.g., clicking buttons, filling out forms) and verifying the DOM output.

**Example:** For a button component, tests check if it renders correctly and if clicking it triggers the correct function.

**Integration Testing:**

**Purpose:** Ensures that different components work together as expected. This could involve testing how a form interacts with a backend service or how multiple UI elements update each other.

**Tool Used:** React Testing Library or custom integration tests within Jest.

**Example:** Test a search bar component in combination with a list display component to ensure the search results are shown correctly.

**End-to-End (E2E) Testing:**

**Purpose:** Simulates a real user experience by testing the entire application flow from start to finish.

**Tool Used:** Cypress or Selenium (if applicable).

**Cypress:** Often used for E2E testing to automate browser interactions and verify the functionality of the app from a user perspective.

Example: Testing a user signing up, navigating the app, and subscribing to the newsletter by interacting with different pages.

**2. Code Coverage:**

**Purpose:** Code coverage ensures that the tests written cover a significant portion of the codebase, minimizing untested areas and reducing the likelihood of bugs.

**Tools Used**: Jest has built-in support for measuring code coverage, generating a report to show the percentage of the code being tested.

**Coverage Metrics:**

**Statements:** The percentage of executable code statements that have been run by tests.

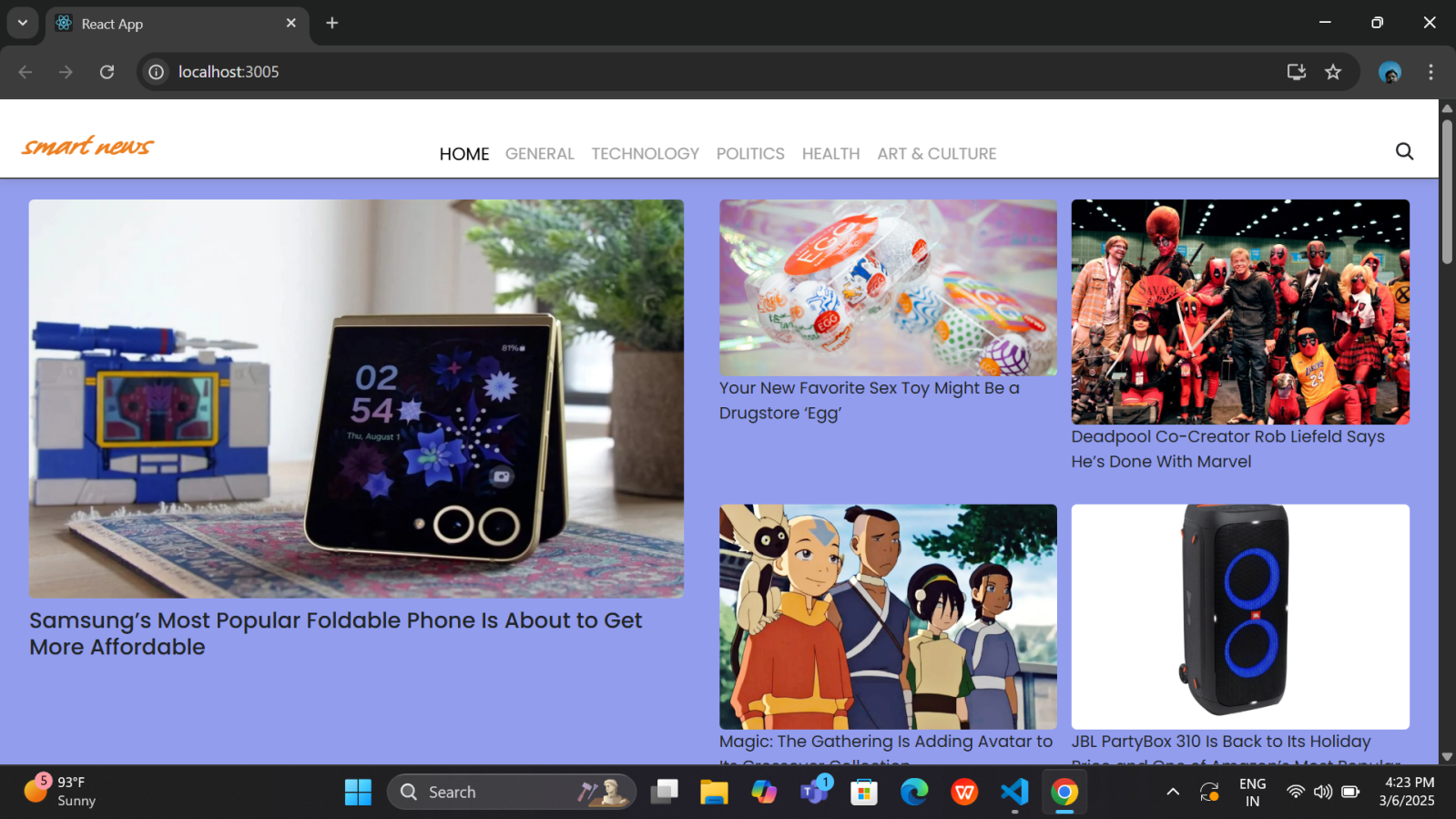
**Branches:** Ensures that all conditional logic (e.g., if-else statements) is covered.

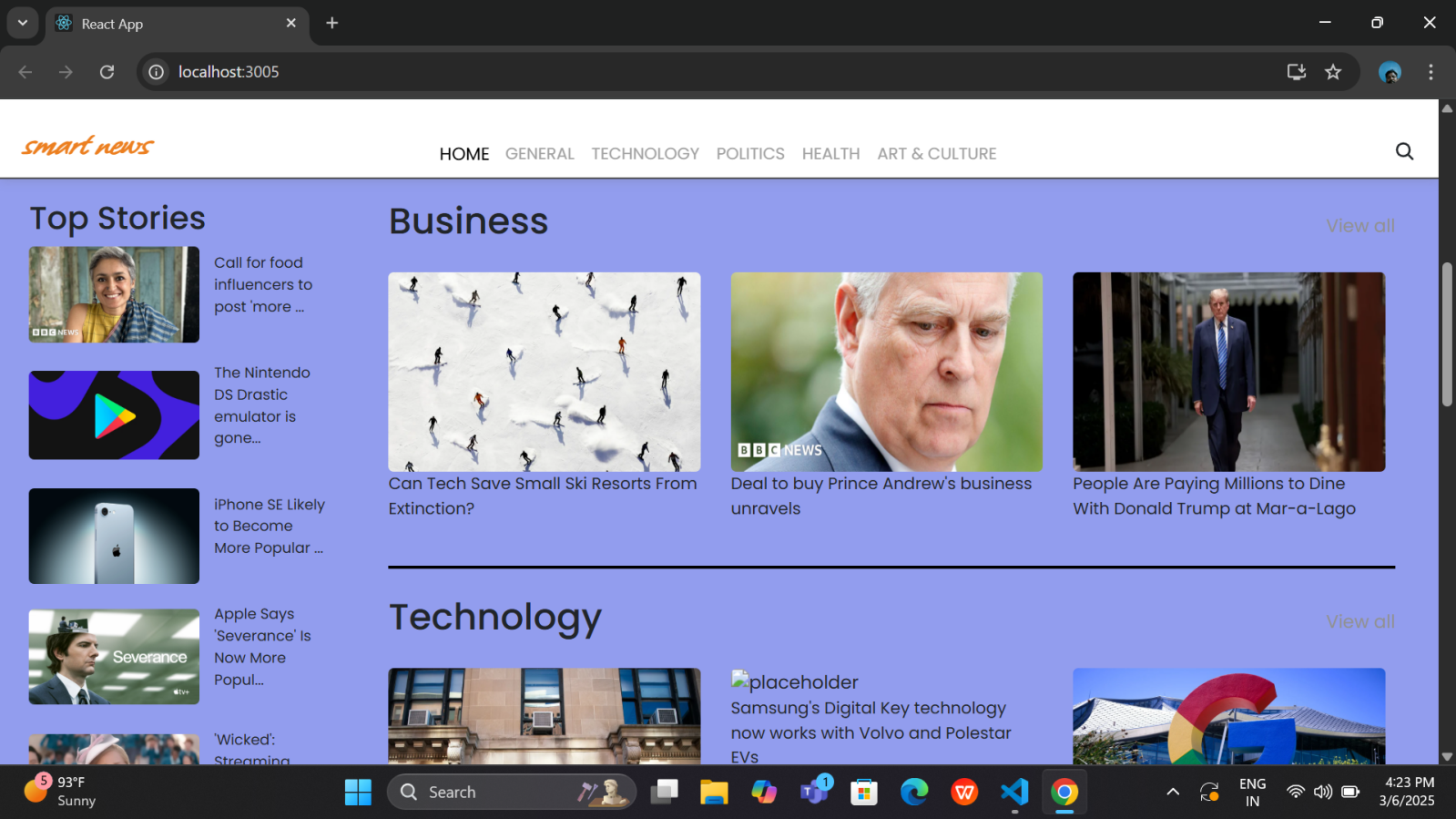
**Functions:** Verifies that all functions are called and tested.

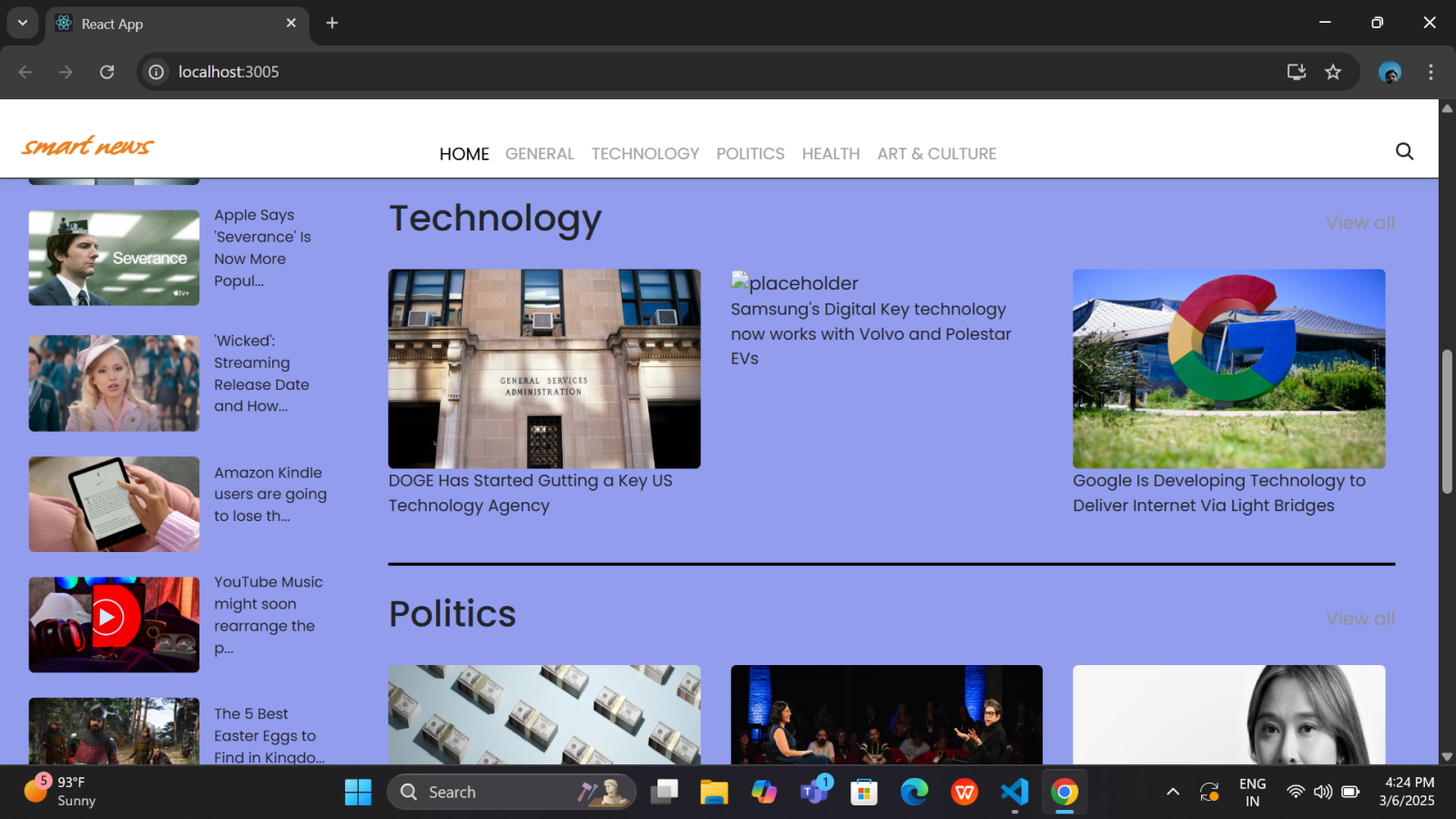
**Lines:** Measures the total number of lines executed during tests.

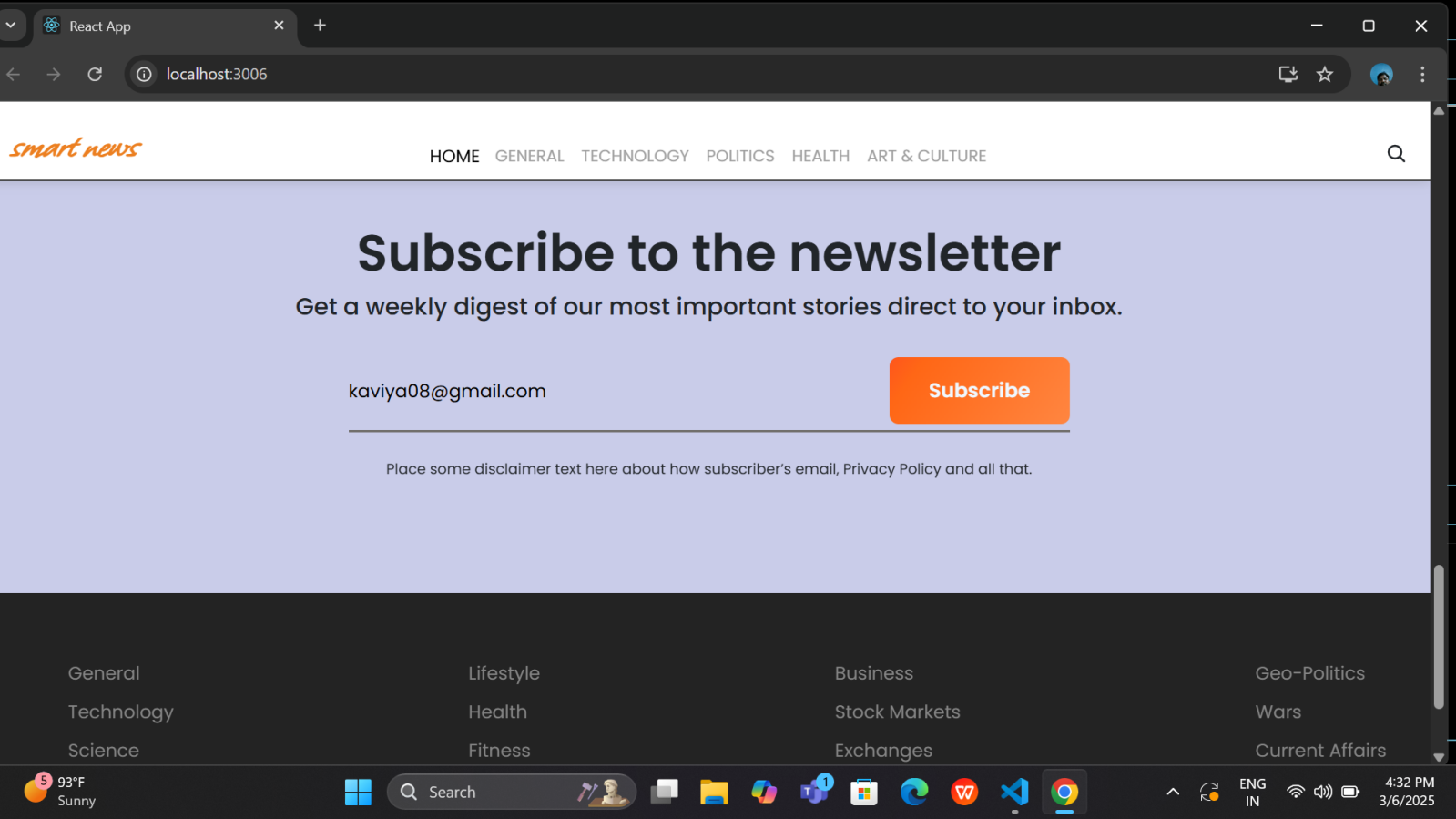
**Example:** If 90% of the code is covered, this means 90% of the codebase is executed during tests. Tools like Coveralls or Codecov can also be integrated for more advanced coverage reporting and tracking over time.

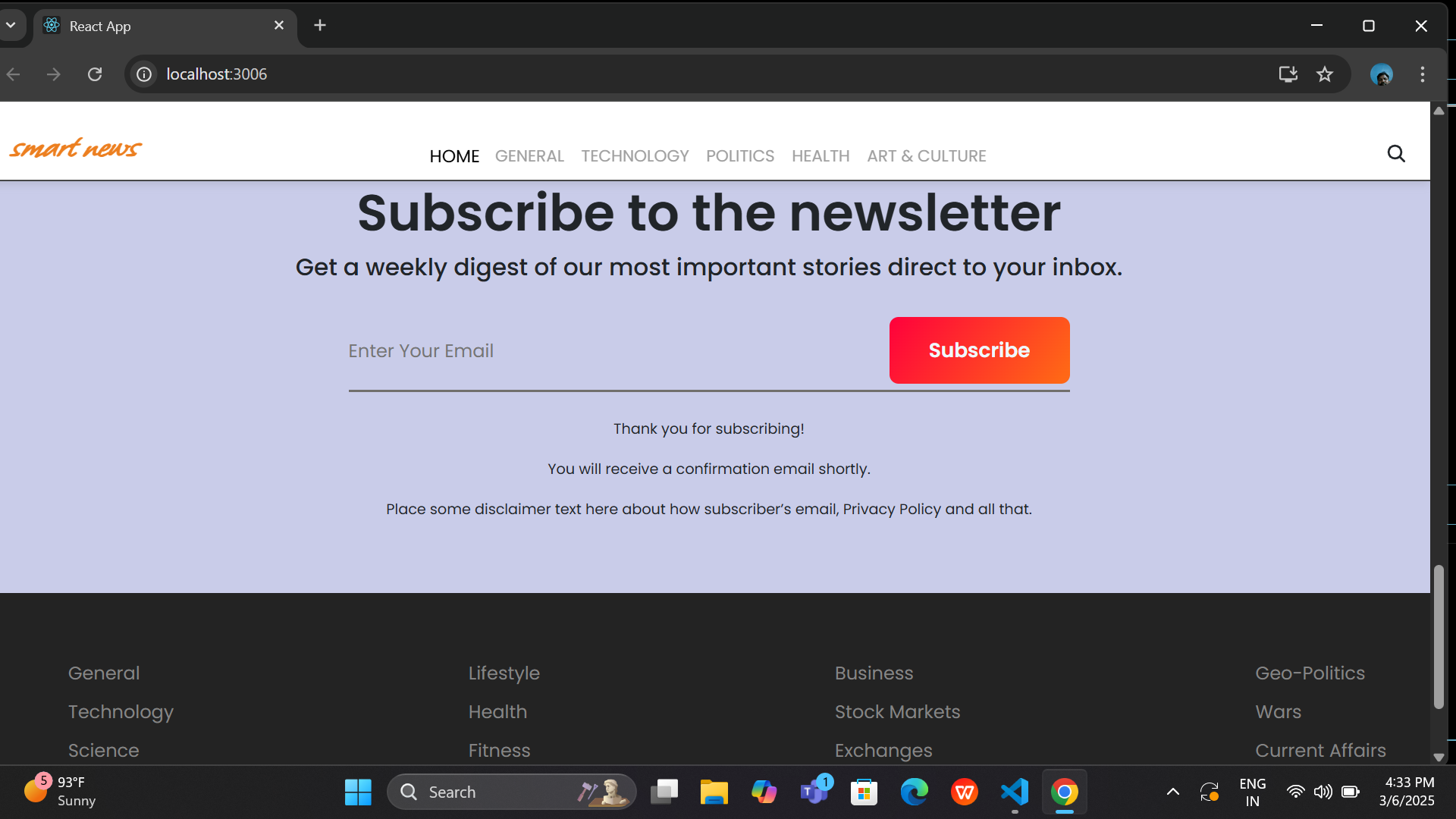
**SCREENSHOTS:**











**KNOWN ISSUES:**

**1. UI Layout Issues on Different Screen Sizes:**

**Issue:** Some elements in the news cards and layout appear misaligned or stretched when viewed on different screen resolutions, especially mobile devices.

**Fix Needed:** The responsive design needs adjustment using CSS media queries and grid/flexbox layouts to ensure proper scaling.

**2. Broken Links or Image URLs:**

**Issue:** Some article links or image URLs on the homepage do not load correctly, causing blank spaces or broken image icons.

**Fix Needed:** Check for broken links and validate the image paths or URLs to ensure they are correct.

**3. Newsletter Subscription Validation:**

Issue: The newsletter subscription form may not properly validate email addresses in some cases, allowing empty or incorrect email formats.

**Fix Needed:** Improve email validation logic and add proper error messages for invalid inputs.

**FUTURE ENHANCESMENT:**

**1. Dark Mode/Theming:** Introduce a dark mode that users can switch to for a better viewing experience, especially in low-light environments. Theming could allow users to customize the look of the application based on preferences.

**2. User Authentication:** Implement a secure user login and registration system to enable users to create accounts, sign in, and have personalized experiences within the application.

**3. Comment and Like System:** Allow users to engage with the news articles by adding comments or liking the content. This increases interaction and makes the app more social.

**4. Infinite Scroll or Pagination:** To handle large amounts of content, either infinite scrolling (continuously loading more news as the user scrolls) or pagination (breaking content into pages) can be added for a better user experience.

**5. Push Notifications:** Implement push notifications to alert users about breaking news, updates, or personalized content, keeping them engaged with the app even when not actively using it.

**6. Multilingual Support:** Add support for multiple languages so users from different regions can view the content in their preferred language, broadening the application's reach.

**7.Offline Mode:** Implement an offline mode where users can view previously loaded content even without an internet connection, improving usability in low-connectivity areas.

**8. Search Functionality:** Add a search bar to allow users to easily find specific news articles or topics within the app, making it more user-friendly and efficient.

**9. Article Bookmarking:** Enable users to bookmark or save articles for later reading. This can help users keep track of important stories they want to revisit.

**10. News Personalization:** Introduce personalized news feeds based on user preferences, history, or reading habits, ensuring the content is relevant to each individual user.

**11. Voice Search Integration:** Implement voice search functionality, allowing users to search for news or perform actions using voice commands, adding convenience and accessibility.

**12.Content Filtering:** Allow users to filter news articles based on categories, sources, or dates, enabling a more tailored and focused reading experience.

THANK YOU