Latest Article

A PROJECT REPORT

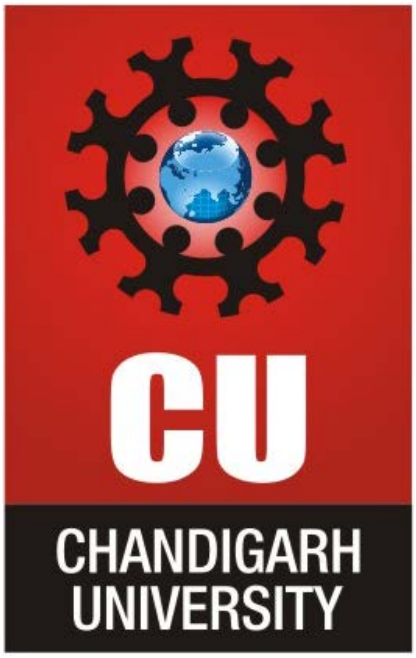
Submitted by

Tabees Ahamad

in partial fulfillment for the award of the degree

of

Master of Computer Application



CHANDIGARH UNIVERSITY

JULY 2025

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

CERTIFICATE

Certified that this project report **"Latest Article"** is the bonafide work of "**TABEES AHAMAD"** who carried out the project work under my supervision.

Mr. Tabrez Khan

Assistant Professor

Department of Computer Application

Integral University, Lucknow

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

CERTIFICATE

Certified that this project report **"Latest Article"** is the bonafide work of "**TABEES AHAMAD**" who have successfully carried out the project.

Dr. Md. Faizan Farooqui
Project Coordinator
Department of Computer Application
Integral University, Lucknow

Dr. Mohd. Faisal
Head
Department of Computer Application
Integral University, Lucknow

DECLARATION

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text”.

Date:21/05/2025 **TABEES AHAMAD**

Acknowledgement

With a deep sense of gratitude, I wish to express my sincere thanks to my guide, **Mr. Tabrez Khan**, Computer Application Department, for giving me the opportunity to work under him on this project report. I deeply appreciate and value his esteemed guidance and encouragement from the beginning to the end of this report. I am extremely grateful to him. His knowledge and company at the time of crisis would be remembered lifelong. He has been a great source of inspiration to me, and I thank them from the bottom of my heart. I also want to thank my parents, who taught me the value of hard work by their own example. I would like to share this moment of happiness with my parents. I would like to thank my department for giving me the opportunity and platform to make my effort a successful on.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	1
1.	INTRODUCTION	2-8
	1.1 Objective	3-4
	1.2 Aim	4-7
	1.3 Scope	7-8
2.	SYSTEM REQUIREMENT	9-23
	2.1 Software Requirement	11-18
	2.2 Hardware Requirement	19-23
3.	PROBLEM IDENTIFICATION	24-30
4.	FEASIBILITY STUDY	31-34
5.	REQUIREMENT ANALYSIS	35-42
6.	SYSTEM DESIGN	43-66
	6.1 System tools	51-54
	6.2 System Module	54-57
	6.3 Context Diagram	57-58
	6.4 Data flow Diagrams	58-63
	6.5 Flow Diagram	63-64
	6.6 Entity Relationship Diagrams	64-66
7.	SNAPSHOTS OF THE PROJECTS	67-70

8.	**DATABASE & TABLE**	**71**
9.	**CONCLUSION & FUTURE WORK**	**72-73**
	REFERENCES & APPENDICES	**74-79**
	MY DETAIS AND GITHUB LINK	**80**

ABSTRACT

This project report presents the design, implementation, and evaluation of a system leveraging the Latest Article, a powerful tool for accessing and retrieving news articles from a wide range of sources across the web. The project aims to explore the capabilities of the Latest Article and demonstrate its potential applications in various domains, including journalism, data analysis, and information retrieval. The project begins with a comprehensive review of the Latest Article, outlining its features, functionalities, and usage guidelines. Through a detailed examination of the API documentation and related literature, the project identifies key capabilities such as search functionality, source filtering, language support, and pagination mechanisms. Following the review, the project proceeds to the design phase, where the architecture and components of the system are defined. The system architecture encompasses modules for API integration, data processing, user interface, and functionality extension. Special attention is paid to scalability, flexibility, and usability considerations to ensure the system's effectiveness and ease of use. With the design in place, the project moves on to the implementation phase, where the system is developed using appropriate technologies and programming languages. The implementation encompasses tasks such as API key management, request handling, response parsing, and data storage. Additionally, the project explores advanced features of the Latest Article, such as real-time updates, sentiment analysis, and topic modeling, to enhance the system's capabilities. Upon completion of the implementation, the project proceeds to the evaluation phase, where the system's performance, reliability, and usability are assessed. Evaluation criteria include response time, data accuracy, error handling, and user satisfaction. Through rigorous testing and validation procedures, the project identifies strengths, weaknesses, and areas for improvement in the system. In conclusion, the project highlights the potential of the Latest Article as a valuable resource for accessing and analyzing news content. By leveraging its rich features and robust functionality, the system demonstrates its effectiveness in facilitating information retrieval, data analysis, and decision-making processes. Additionally, the project offers insights into future research directions and opportunities for further exploration in the field of Latest Article integration and utilization. In an era characterized by rapid information dissemination and evolving news consumption patterns, the role of News APIs has become increasingly pivotal. This article presents a comprehensive explration of the latest advancements, features, and benefits of the News API built using React.js, a prominent JavaScript library for building interactive user interfaces.

CHAPTER - 1

INTRODUCTION

The Latest Article has revolutionized the way news content is accessed and utilized in various domains, offering a comprehensive platform for retrieving up-to-date news articles from a multitude of sources across the internet. In an era where information is abundant and constantly evolving, the Latest Article provides a valuable resource for individuals and organizations seeking to stay informed and up-to-date on current events, trends, and developments. This project report aims to explore the capabilities and potential applications of the Latest Article within the context of information retrieval and analysis. By leveraging the rich features and functionalities of the Latest Article, this project seeks to demonstrate how news content can be efficiently gathered, processed, and analyzed to extract valuable insights and facilitate decision-making processes. The introduction section provides an overview of the project objectives, scope, and significance. It outlines the motivation behind the project, highlighting the importance of accessing timely and relevant news content in today's fast-paced digital landscape. Additionally, the introduction sets the stage for the subsequent sections of the report, which will delve into the design, implementation, and evaluation of the Latest Article integration. The ability to access up-to-date and reliable news articles quickly and efficiently can greatly impact decision-making processes, knowledge acquisition, and staying connected with global events. The Latest Article project aims to address this need by leveraging the capabilities of the News API to provide users with easy access to the most recent news articles from a wide range of sources. This project report presents the design, implementation, and evaluation of the Latest Article system, which utilizes the News API to retrieve and display the latest news articles. The system aims to provide users with a user-friendly interface for accessing timely news content, thereby enhancing their ability to stay informed and up-to-date on current events. Through this project, we aim to showcase the versatility and effectiveness of the Latest Article as a powerful tool for accessing and analyzing news content. By providing a comprehensive overview of the API's features, functionalities, and potential use cases, this project report aims to inform and inspire readers to explore the possibilities of leveraging the Latest Article in their own projects and applications.

1.1. OBJECTIVES

1. **User Interface Development:** One of the primary objectives of a React.js project for articles could be to develop a modern and user-friendly interface for accessing and reading news articles. This involves designing components such as article cards, navigation bars, search bars, and filters to enhance the user experience.
2. **Real-time Updates:** If the project aims to provide real-time updates on news articles, an objective would be to implement features that automatically fetch and display new articles as they are published. This may involve integrating with a backend server or utilizing WebSocket technology for real-time communication.
3. **Dynamic Content Rendering:** Another objective is to dynamically render article content based on user interactions and preferences. This includes implementing features such as infinite scrolling, lazy loading of images, and dynamic filtering to enhance the performance and responsiveness of the application.
4. **Responsive Design:** Ensuring that the application is accessible and usable across various devices and screen sizes is another important objective. This involves implementing responsive design techniques using CSS frameworks like Bootstrap or CSS Grid to adapt the layout and styling of the application based on the viewport size.
5. **Integration with APIs:** If the project involves fetching news articles from external sources, integrating with APIs (such as the News API) would be a key objective. This includes setting up API requests, handling responses, and parsing the data to display relevant information to users.
6. **State Management:** Managing the state of the application efficiently is crucial for a smooth user experience. An objective would be to implement state management solutions such as React's built-in state management, Context API, or external libraries like Redux to manage complex application states and data flow.
7. **Performance Optimization:** Optimizing the performance of the application is another key objective. This involves techniques such as code splitting, memoization, and lazy loading to reduce initial loading times and improve overall performance, especially for large-scale applications with a significant amount of data.

8. **SEO Optimization:** If the project aims to attract organic traffic from search engines, optimizing the application for search engine optimization (SEO) would be an important objective. This involves implementing server-side rendering (SSR), generating SEO-friendly metadata, and ensuring proper URL structure and navigation for better search engine visibility.
9. **Testing and Debugging:** Ensuring the reliability and stability of the application through thorough testing and debugging is essential. An objective would be to implement unit tests, integration tests, and end-to-end tests using tools like Jest, React Testing Library, and Cypress to identify and fix issues early in the development process.
10. **Scalability and Maintainability:** Building a scalable and maintainable codebase is crucial for the long-term success of the project. An objective would be to follow best practices in code organization, modularization, and documentation to facilitate future enhancements, updates, and maintenance tasks.

1.2. AIM

The aim of a project using React.js for articles or news is to create a user-friendly platform that provides real-time updates on news articles, offers personalized content recommendations, ensures accessibility and responsive design, optimizes performance, integrates with external APIs to fetch articles from various sources, implements advanced features such as search and filtering, ensures SEO friendliness, and facilitates collaboration among team members, using React.js for articles or news can vary depending on the specific goals and requirements, but here are some common aims:

Provide a User-Friendly News Platform: The primary aim could be to create a user-friendly platform where users can easily discover, browse, and read articles from various sources.

Deliver Real-Time Updates: Another aim might be to deliver real-time updates on news articles, ensuring that users have access to the latest information as it becomes available.

Offer Personalized Content: The project might aim to offer personalized content recommendations based on user preferences, browsing history, or interaction patterns, enhancing the user experience and engagement.

Enhance Accessibility: Aiming to make news content accessible to a wide range of users, including those with disabilities, by implementing accessibility features such as keyboard navigation, screen reader compatibility, and high contrast modes.

Ensure Responsive Design: The aim could be to ensure that the application is responsive and works seamlessly across various devices and screen sizes, providing a consistent experience for users on desktops, tablets, and smartphones.

Optimize Performance: Optimizing the performance of the application to reduce loading times, improve responsiveness, and ensure smooth navigation, even with large volumes of data or high traffic.

Integrate with External APIs: Integrating with external APIs, such as news APIs, to fetch and display articles from a variety of sources, providing users with a diverse range of content to explore.

Implement Advanced Features: Implementing advanced features such as search functionality, filtering options, bookmarking, commenting, and sharing, to enrich the user experience and provide additional value to users.

Ensure SEO Friendliness: Aiming to make the application search engine friendly by implementing best practices for SEO, such as generating SEO-friendly URLs, optimizing metadata, and ensuring proper indexing of content.

Facilitate Collaboration: If the project involves multiple collaborators, the aim might be to facilitate collaboration and streamline the development process by adopting version control systems, project management tools, and effective communication channels.

Personalized Content Recommendations: The project aims to enhance user engagement by offering personalized content recommendations based on user preferences, browsing history, or past interactions. This could involve implementing recommendation algorithms that analyze user behavior and suggest relevant articles.

Accessibility: Ensuring that the platform is accessible to users with disabilities is a key aim. This includes implementing accessibility features such as keyboard navigation, screen reader compatibility, and high contrast modes to ensure that all users can access the content easily.

Responsive Design: The project aims to provide a consistent user experience across devices of various screen sizes, including desktops, tablets, and smartphones. This involves implementing responsive design techniques such as fluid layouts and media queries to adapt the interface to different viewport sizes.

Optimal Performance: Optimizing the performance of the platform is crucial to ensure fast loading times and smooth navigation. This includes minimizing unnecessary network requests, optimizing code for efficiency, and implementing techniques such as lazy loading and code splitting to improve page load times.

Integration with External APIs: The project aims to fetch news articles from external sources by integrating with APIs such as the News API. This involves setting up API requests, handling responses, and parsing the data to display relevant information to users.

Advanced Features: Implementing advanced features such as search functionality, filtering options, bookmarking, commenting, and social sharing enhances the user experience and provides additional value to users.

SEO Friendliness: Ensuring that the platform is search engine friendly is essential to attract organic traffic from search engines. This involves implementing best practices for SEO, such as generating SEO-friendly URLs, optimizing metadata, and ensuring proper indexing of content.

Collaboration: If the project involves multiple collaborators, facilitating collaboration and streamlining the development process is important. This includes adopting version control systems, project management tools, and effective communication channels to coordinate efforts and track progress. Overall, the aim of a project using React.js for articles or news is to create a robust, user-centric platform that effectively delivers news content while prioritizing factors such as usability, accessibility, performance, and scalability.

1.3. SCOPE

The scope of a project using React.js for articles or news can vary depending on factors such as the resources available, time constraints, and specific goals. However, here's a comprehensive scope outline for such a project:

User Interface Design:

Designing and implementing a visually appealing and intuitive user interface for browsing news articles. Creating responsive layouts that adapt to different screen sizes and devices. Designing interactive components such as article cards, navigation menus, search bars, and filters.

Backend Integration:

Integrating with external APIs (e.g., News API) to fetch news articles from various sources. Handling API requests, responses, and data parsing to retrieve relevant information. Implementing caching mechanisms to optimize performance and reduce API usage.

Real-Time Updates:

Implementing features for real-time updates on news articles, using technologies like Web Sockets or polling. Ensuring that newly published articles are promptly fetched and displayed to users without requiring manual refresh.

Content Management:

Implementing features for organizing and categorizing news articles based on topics, sources, or publication dates. Allowing users to save articles for later reading, bookmark favorite articles, list.

Personalization and User Engagement:

Implementing algorithms for personalized content recommendations based on user preferences and behavior. Providing features for users to customize their news feed, subscribe to specific topics or sources, and receive notifications for new articles.

Search and Filtering:

Implementing robust search functionality to allow users to find articles by keywords, topics, or authors. Providing advanced filtering options to refine search results based on criteria such as publication date, popularity, or article type.

User Authentication and Authorization:

Implementing user authentication mechanisms to secure user accounts and personal data. Allowing users to create accounts, log in, and manage their profiles, preferences, and saved articles securely.

Social Sharing and Interaction:

Integrating social sharing features to allow users to share articles on social media platforms or via email. Implementing features for user interaction such as liking, commenting, and sharing articles within the platform.

Performance Optimization:

Optimizing the performance of the application to ensure fast loading times, smooth navigation, and efficient use of resources.

Testing and Quality Assurance:

Conducting thorough testing of the application to identify and fix bugs, errors, and compatibility issues.

CHAPTER - 2

SYSTEM REQUIREMENTS

The system requirements for a project using React.js for articles or news depend on various factors, including the size and complexity of the application, expected traffic, and specific functionalities. However, here's a general overview of the system requirements:

1. Development Environment:

Operating System: Windows, macOS, or Linux.

Text Editor or Integrated Development Environment (IDE): Examples include Visual Studio Sublime Text, Atom, Code or WebStorm.

Node.js: Latest LTS (Long-Term Support) version installed to run npm (Node Package Manager) commands.

Git: Version control system for managing codebase changes and collaborating with team members.

2. Frontend Requirements:

React.js: The latest stable version of React.js library installed to build the frontend user interface.

React Router: Library for handling client-side routing and navigation within the React application.

State Management: Depending on the complexity of the application, you may use React's built-in state management or external libraries like Redux or Context API.

Styling: CSS preprocessors (e.g., Sass, Less) or CSS-in-JS solutions (e.g., styled-components) for styling components.

UI Frameworks: Optional use of UI component libraries like Material-UI or Bootstrap for pre-styled components and layout grids.

3. Backend Requirements (if applicable):

Backend Framework: If server-side logic is required, choose a backend framework such as Express.js (Node.js), Django (Python), or Laravel (PHP).

Database: Choose a suitable database solution based on the project's requirements, such as MongoDB, PostgreSQL, MySQL, or SQLite.

Authentication and Authorization: Implement user authentication and authorization mechanisms using libraries like Passport.js (Node.js) or Django's built-in authentication system.

4. API Integration:

News API: If fetching news articles from external sources, integrate with APIs like the News API or other news aggregation services.

Social Media APIs: Optional integration with social media APIs (e.g., Facebook, Twitter) for sharing articles or accessing social login functionalities.

5. Deployment Environment:

Web Hosting: Choose a web hosting provider capable of hosting React.js applications, such as AWS, Heroku, Netlify, or Vercel.

Domain Name: Register a domain name for the application's URL if deploying to a custom domain.

SSL Certificate: Secure the application with an SSL certificate to enable HTTPS encryption for data transmission. Continuous Integration/Continuous Deployment (CI/CD) Pipeline: Set up automated deployment pipelines using tools like GitHub Actions, Travis CI, or Jenkins for seamless deployment and updates.

1. Performance Considerations:

Server Resources: Ensure that the chosen hosting provider offers sufficient server resources (CPU, memory, bandwidth) to handle expected traffic and application demands.

Content Delivery Network (CDN): Optionally, leverage a CDN like Cloudflare or Amazon CloudFront to distribute static assets and improve loading speeds for users globally.

2. Monitoring and Analytics:

Logging and Monitoring: Implement logging mechanisms and monitoring tools (e.g., Sentry, LogRocket) to track application errors, performance issues, and user interactions.

Analytics: Integrate analytics tools (e.g., Google Analytics, Mixpanel) to gather insights into user behavior, traffic patterns, and engagement metrics.

3. Backup and Disaster Recovery:

Data Backup: Set up regular backups of application data and database contents to prevent data loss in case of system failures or security breaches.
Disaster Recovery Plan: Develop a contingency plan for recovering the application and data in the event of unexpected incidents or emergencies.

2.1 SOFTWARE REQUIREMENT

Creating a comprehensive News API application with a React.js frontend requires a detailed and thorough understanding of both functional and non-functional requirements. These requirements will guide the development process, ensuring that the final product meets user needs and operates efficiently. For a project using React.js for articles or news, the software requirements would include tools and frameworks necessary for both development and deployment. Here's a list of the software requirements:

1. **Node.js and npm (Node Package Manager):** These are essential for running JavaScript code on the server side and managing packages and dependencies for the project.

2. **Text Editor or Integrated Development Environment (IDE):** Choose a text editor or IDE suitable for JavaScript development. Some popular options include Visual Studio Code, Sublime Text, Atom, or WebStorm.

3. **Git:** Version control system for tracking changes to the project's codebase, collaborating with team members, and managing code revisions.

4. **React Developer Tools:** Browser extensions or development tools provided by React for debugging React applications. These tools help inspect React component hierarchies, state, and props in real-time.

5. **API Documentation Tools (Optional):** If integrating with external APIs, tools such as Swagger or Postman can help document and test API endpoints.

6. **Backend Framework (if applicable**): Depending on the project's requirements, you may need a backend framework such as Express.js (Node.js), Django (Python), or Laravel (PHP) for server-side logic and database interactions.

7. **Database Management System (DBMS) (if applicable):** Choose a suitable database system for storing and managing application data. Common options include MongoDB, PostgreSQL, MySQL, or SQLite.

8. **Web Browser:** Use modern web browsers like Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge for testing and debugging the application.

9. **API Key or Access Token (if using external APIs):** Obtain necessary API keys or access tokens from external services (e.g., News API) for fetching news articles and data.

10. **Deployment Tools and Platforms**: Select a web hosting provider and deployment platform suitable for hosting React.js applications. Options include AWS, Heroku, Netlify, Vercel, or Firebase Hosting.

11. **SSL Certificate (for HTTPS):** Secure the application with an SSL certificate to enable HTTPS encryption for secure data transmission between the server and client.

12. **Continuous Integration/Continuous Deployment (CI/CD) Tools (Optional):** Set up CI/CD pipelines using tools like GitHub Actions, Travis CI, Jenkins, or CircleCI for automated testing, building, and deployment of the application.

Functional Requirements-

User Authentication and Authorization:

- **User Registration**: Users must be able to register by providing their username, email, and password.
- **User Login**: Users must be able to log in using their registered email and password.
- **Password Reset**: Users must be able to reset their password via email verification.
- **JWT Authentication**: Secure API endpoints with JSON Web Tokens (JWT) to ensure authorized access.
- **OAuth Integration**: Allow users to log in using third-party services such as Google, Facebook, or Twitter.
- **Role-Based Access Control (RBAC)**: Implement roles (e.g., admin, editor, user) to manage access and permissions.

News Article Management:

- **Fetch News Articles: R**etrieve news articles from various external sources and display them to users.
- **Filter and Sort Articles**: Users must be able to filter article by category, source, date, and keywords. Sorting options should include relevance, date, and popularity.
- **Search Articles**: Implement a search functionality to allow users to find articles based on keywords and phrases.
- **Article Details**: Provide detailed views for individual articles, including content, author, publication date, and source.
- **Save Articles**: Allow users to save articles to their profiles for future reference.

User Preferences:

- **Manage Preferences**: User must be able to set and update their preferences for categories, keywords, and sources.
- **Personalized Feed**: Provide a personalized news feed based on user preferences and reading history.

- **Notification Settings**: Allow users to customize their notification preferences for breaking news, new articles, etc.

Content Management System (CMS):

- **Admin Dashboard**: Provide an admin dashboard to manage articles, categories, sources, and user accounts.
- **Article Submission and Editing**: Allow authorized users (e.g., editors, journalist) to submit, edit, and approve articles through a CMS interface.
- **Moderation Tools**: Implement moderation tools to review & manage user-generated content and comments.

Interactive Features:

- **Comments**: Users must be able to comment on articles.
- **Likes and Ratings**: Implement a system for users to like or rate articles.
- **Social Sharing**: Allow users to share articles on social media platforms.

Non-Functional Requirements-

Performance:

- **Response Time**: Ensure that the API responds to requests within a reasonable time frame, ideally within 200ms for most requests.
- **Scalability**: Design the system to handle an increasing number of users and data volume without performance degradation.
- **Caching**: Implement caching mechanisms to reduce load times for frequently accessed data.

Security:

- **Data Encryption**: Encrypt sensitive data both in transit and at rest.
- **Input Validation**: Validate all user inputs to prevents SQL injection, XSS, and other common vulnerabilities.

- **Regular Audits**: Conduct regular security audits and penetration testing to identify and fix vulnerabilities.
- **Rate Limiting**: Implement rate limiting to prevent abuse of the API.

Usability:

- **Responsive Design**: Ensure the application is fully responsive and work seamlessly across different devices and screen sizes.
- **Intuitive UI**: Design an intuitive and user-friendly interface that minimizes the learning curve for new users.
- **Accessibility**: Ensure the application meets accessibility standards (e.g., WCAG) to support users with disabilities.

Reliability and Availability:

- **Uptime**: Aim for high availability with an uptime of at least 99.9%.
- **Backup and Recovery**: Implement robust backup and recovery procedures to prevent data loss.
- **Error Handling**: Implement comprehensive error handling and logging mechanisms to diagnose and resolve issues quickly.

Maintainability:

- **Modular Architecture**: Design the applications with a modular architecture to facilitate easier updates and maintenance.
- **Documentation**: Provide comprehensive documentation for the codebase, API endpoints, and user guides.
- **Testing**: Implement automated testing (unit, integration, and end-to-end) to ensure code quality and reliability.

1. **Compliance:**

- **Data Privacy**: Ensure compliance with data privacy regulations such as GDPR, CCPA, etc.

- **Content Licensing**: Ensure that all news content adheres to licensing agreements and copyright laws.

System Requirements-

1. **Frontend Requirements:**

- **Framework**: React.js for building the user interface.
- **State Management**: Use a state management library like Redux or Context API.
- **Routing**: Implement client-side routing using React Router.
- **HTTP Client**: Use Axios or Fetch API for making HTTP requests to the backend.
- **Styling**: Use CSS-in-JS libraries (e.g., styled-components) or traditional CSS/SASS for styling the application.
- **Build Tools**: Use Webpack or Create React App (CRA) for building and bundling the application.

2. **Backend Requirements**

- **Framework**: Node.js with Express.js for creating the RESTful API.
- **Database**: PostgreSQL or MySQL for relational database management.
- **ORM**: Use an ORM like Sequelize or TypeORM for database interactions.
- **Authentication**: Implement JWT for token-based authentication and OAuth for third-party authentication.
- **Caching**: Use Redis or Memcached for caching frequently accessed data.
- **Logging**: Implement logging using libraries like Winston or Morgan.

3. **Development and Deployment**

- **Version Control**: Use Git for version control and GitHub or GitLab for repository hosting.
- **CI/CD**: Set up continuous integration and deployment pipelines using tools like Jenkins, Travis CI, or GitHub Actions.
- **Hosting**: Deploy the application on cloud platforms like AWS, Azure, or Google Cloud. Use services like Heroku for simpler deployment.

- **Containerization**: Use Docker for containerizing the application and Kubernetes for orchestration if needed.

Detailed Use Cases-

Use Case 1: User Registration and Login:

- **Actors**: User, System
- **Description**: Users can register and log in to the application.
- **Preconditions**: User is not logged in.
- **Postconditions**: User account is created, and the user is logged in.
- **Steps**:
    1. User navigates to the registration page.
    2. User enters username, email, and password.
    3. System validates the input and creates a new user account.
    4. User receives a confirmation email.
    5. User logs in with email and password.
    6. System generates a JWT and returns it to the user.

Use Case 2: Fetching and Displaying News Articles:

- **Actors**: User, System
- **Description**: Users can view a list of news articles.
- **Preconditions**: User is logged in.
- **Postconditions**: A list of news articles is displayed to the user.
- **Steps**:
    1. User navigates to the homepage.
    2. System fetches news articles from external sources.
    3. System processes and stores articles in the database.
    4. System displays a list of articles to the user.

Use Case 3: Managing User Preferences

- **Actors**: User, System

- **Description**: Users can set their news preferences.
- **Preconditions**: User is logged in.
- **Postconditions**: User preferences are saved and used to personalize the news feed.
- **Steps**:
    1. User navigates to the preferences page.
    2. User selects preferred categories, keywords, and sources.
    3. User saves the preferences.
    4. System updates the user's preferences in the database.

Use Case 4: Commenting on Articles

- **Actors**: User, System
- **Description**: Users can comment on news articles.
- **Preconditions**: User is logged in and viewing an article.
- **Postconditions**: User's comment is saved and displayed under the article.
- **Steps**:
    1. User navigates to the article details page.
    2. User enters a comment in the comment section.
    3. User submits the comment.
    4. System saves the comment and displays it under the article.

Use Case 5: Admin Managing Content

- **Actors**: Admin, System
- **Description**: Admin can manage articles, categories, and sources.
- **Preconditions**: Admin is logged in.
- **Postconditions**: Changes made by the admin are saved and reflected in the application.
- **Steps**:
    1. Admin logs in to the admin dashboard.
    2. Admin navigates to the content management section.
    3. Admin adds, edits, or deletes articles, categories, or sources.
    4. System updates the database accordingly.

2.2. HARDWARE REQUIREMENT

The hardware requirements for a project using React.js for articles or news are generally minimal, as the bulk of the processing is done on the development machine during coding and testing. However, for development, deployment, and testing purposes, certain hardware specifications are beneficial. Creating a robust News API with a React.js frontend involves not only comprehensive software requirements but also precise hardware specifications. Properly identifying and allocating hardware resources is essential to ensure optimal performance, scalability, reliability, and security of the application. Below is a detailed examination of the hardware requirements for different stages of development, deployment, and production environments.

Development Environment-

1. **Developer Workstations:**

- **Processor**: Modern multi-core processors, such as Intel Core i5/i7 or AMD Ryzen 5/7.
    - **Reason**: To handle the demands of coding, running local servers, testing, and potentially running virtual machines or containers.
- **Memory (RAM)**: At least 16 GB of RAM.
    - **Reason**: For running development environments smoothly, especially when using IDEs, browsers, and other tools simultaneously.
- **Storage**: SSD with at least 512 GB capacity.
    - **Reason**: Fast read/write speeds for quick loading and saving of projects, libraries, and dependencies.
- **Graphics**: Integrated graphics or dedicated GPU (if working with design tools or heavy graphical interfaces).
    - **Reason**: Basic integrated graphics are sufficient for coding, but a dedicated GPU may be required for design and UI/UX work.
- **Operating System**: Windows 10/11, macOS, or Linux.
    - **Reason**: Compatibility with various development tools and personal preference for OS environments.

- **Network**: Reliable broadband internet connection.
    - **Reason**: For accessing online resources, APIs, version control systems, and collaboration tools.

2. **Development Servers:**

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Reason**: To support multiple virtual environments and concurrent development processes.
- **Memory (RAM)**: At least 32 GB of RAM.
    - **Reason**: To run multiple development, testing, and staging environments efficiently.
- **Storage**: SSD with at least 1 TB capacity.
    - **Reason**: To store source code, build artifacts, and dependencies.
- **Network**: Gigabit Ethernet connection.
    - **Reason**: To ensure fast access to repositories, continuous integration/continuous deployment (CI/CD) pipelines, and other network resources.

Testing and Staging Environment-

Testing Servers:

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Reason**: To handle automated tests, including unit, integration, and performance testing.
- **Memory (RAM)**: At least 32 GB of RAM.
    - **Reason**: To manage multiple test instances and large datasets.
- **Storage**: SSD with at least 1 TB capacity.
    - **Reason**: For storing test results, logs, and artifacts.
- **Network**: Gigabit Ethernet connection.
    - **Reason**: For fast data transfer and efficient testing of API endpoints.

Staging Servers:

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Reason**: To replicate the production environment as closely as possible for accurate testing.
- **Memory (RAM)**: At least 32 GB of RAM.
    - **Reason**: To ensure smooth operation under load conditions similar to production.
- **Storage**: SSD with at least 1 TB capacity.
    - **Reason**: For deploying pre-release versions of the application and conducting user acceptance testing (UAT).
- **Network**: Gigabit Ethernet connection.
    - **Reason**: For reliable and fast deployment of application versions and updates.

Production Environment-

Web Servers:

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Reason**: To handle concurrent requests from users efficiently.
- **Memory (RAM)**: At least 64 GB of RAM.
    - **Reason**: To support high traffic loads and ensure smooth performance.
- **Storage**: SSD with at least 1 TB capacity.
    - **Reason**: For fast access to static files, logs, and temporary data.
- **Network**: Dual 10 Gigabit Ethernet connections.
    - **Reason**: For high availability and redundancy, ensuring minimal downtime and fast data transfer rates.
- **Load Balancers**: Hardware or virtual load balancers.
    - **Reason**: To distribute incoming traffic across multiple web servers, ensuring optimal performance and availability.

Application Servers:

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.

    - **Reason**: To handle backend logic, API requests, and interactions with databases.
- **Memory (RAM)**: At least 64 GB of RAM.
    - **Reason**: To manage multiple processes and large volumes of data.
- **Storage**: SSD with at least 1 TB capacity.
    - **Reason**: For fast read/write operations and to store application data and logs.
- **Network**: Dual 10 Gigabit Ethernet connections.
    - **Reason**: For high availability and redundancy.

Database Servers:

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Reason**: To efficiently process complex queries and manage large datasets.
- **Memory (RAM)**: At least 128 GB of RAM.
    - **Reason**: To ensure fast query processing and support for in-memory databases if necessary.
- **Storage**: SSDs with at least 2 TB capacity, configured in RAID 10.
    - **Reason**: For data redundancy, high availability, and fast read/write speeds.
- **Network**: Dual 10 Gigabit Ethernet connections.
    - **Reason**: For fast data transfer and high availability.

Caching Servers:

- **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Reason**: To handle caching operations efficiently.
- **Memory (RAM)**: At least 64 GB of RAM.
    - **Reason**: To store large amounts of cached data for quick access.
- **Storage**: SSD with at least 512 GB capacity.
    - **Reason**: For fast caching operations.
- **Network**: Dual 10 Gigabit Ethernet connections.
    - **Reason**: For fast data transfer and high availability.

Content Delivery Network (CDN):

- **CDN Provider**: Use a reliable CDN provider like Cloudflare, Akamai, or AWS CloudFront.
    - **Reason**: To distribute static asset and media files globally, reducing load times and improving user experience.
- **Edge Servers**: Multiple edge servers located strategically around the globe.
    - **Reason**: To ensure content is delivered from locations close to the user, minimizing latency.

Backup and Recovery Systems:

- **Backup Servers**: Dedicated servers for backup operations.
    - **Processor**: Multi-core processors, such as Intel Xeon or AMD EPYC.
    - **Memory (RAM)**: At least 32 GB of RAM.
    - **Storage**: High-capacity HDDs (e.g., 10 TB or more), configured in RAID 6 for redundancy.
    - **Network**: Dual 10 Gigabit Ethernet connections.
    - **Reason**: To ensure regular and reliable backups of all critical data.
- **Cloud Backup Services**: Use cloud-based backup solutions like AWS S3, Google Cloud Storage, or Azure Blob Storage.
    - **Reason**: For off-site storage and additional redundancy.

Redundancy and High Availability-

- **Redundant Power Supplies:**

- **Uninterruptible Power Supplies (UPS)**: Provide backup power in case of outages.
    - **Reason**: To ensure that servers remain operational during power interruptions.
- **Backup Generators**: On-site generators to provide long-term power in case of extended outages.
    - **Reason**: To keep critical systems running during prolonged power failures.

CHAPTER - 3

PROBLEM IDENTIFICATION

Creating a News API and integrating it into a React.js application involves identifying and addressing a myriad of challenges. These challenges span across various domains such as user experience, data management, security, scalability, and more. Here's a detailed and comprehensive examination of the problems that might be encountered:

Data Quality and Reliability-

Inconsistent Data Sources

- **Issue**: News data can come from various sources, each with its own format and reliability. Integrating multiple sources can lead to inconsistencies in the data.
- **Impact**: Users might receive duplicate articles, conflicting information, or incomplete data, which can undermine trust in the application.

Inaccurate Information

- **Issue**: The spread of misinformation and fake news is a significant problem. Ensuring the accuracy of the news content is challenging.
- **Impact**: Users may be misinformed, leading to a lack of credibility and potential legal issues for the platform.

Data Latency

- **Issue**: News is time-sensitive, and delays in data updates can make the information outdated.
- **Impact**: Users might miss out on breaking news or get information that is no longer relevant, reducing the application's value.

User Experience and Engagement-

Complex Navigation

- **Issue**: If the application's navigation is not intuitive, users might find it difficult to access desired content.
- **Impact**: Poor user experience can lead to lower engagement and higher bounce rates.

Lack of Personalization

- **Issue**: A one-size-fits-all approach does not cater to individual user preferences.
- **Impact**: Users may not find the content relevant or engaging, leading to decreased retention and satisfaction.

Information Overload

- **Issue**: Users can be overwhelmed by too much information or too many choices.
- **Impact**: Decision fatigue and a cluttered interface can drive users away.

Performance and Scalability-

Slow Load Times

- **Issue**: Large volumes of data, especially multimedia content, can slow down the application.
- **Impact**: Poor performance can frustrate users, especially on mobile devices or slower internet connections.

Scalability Issues

- **Issue**: As the user base grows, the system must handle increased load and data volume.
- **Impact**: Without proper scaling, the application can experience downtimes, slow responses, and poor user experience.

Real-Time Data Handling

- **Issue**: Implementing real-time updates for news articles and user interactions can be resource-intensive.
- **Impact**: Delayed or failed updates can reduce the application's reliability and user engagement.

Security and Privacy-

Data Breaches

- **Issue**: Protecting user data and ensuring secure communication is critical.
- **Impact**: Data breaches can lead to loss of user trust, legal consequences, and financial losses.

Unauthorized Access

- **Issue**: Ensuring that only authorized users have access to certain features or data.
- **Impact**: Unauthorized access can lead to data manipulation, content tampering, or misuse of the platform.

Compliance with Regulations

- **Issue**: Adhering to privacy laws such as GDPR, CCPA, etc., requires robust data management practices.
- **Impact**: Non-compliance can result in hefty fines and legal action.

Content Management-

Content Moderation

- **Issue**: Filtering out inappropriate or harmful content, including spam and offensive material.
- **Impact**: Inadequate moderation can harm the platform’s reputation and user trust.

Duplicate Content

- **Issue**: Aggregating news from multiple sources can lead to duplicate articles.
- **Impact**: Users may get annoyed by repeated content, which can affect the perceived quality of the application.

Multilingual Support

- **Issue**: Catering to a global audience requires supporting multiple languages.
- **Impact**: Lack of language support can limit the application's reach and user base.

Integration with External Systems-

API Rate Limits

- **Issue**: External news APIs often have rate limits that restrict the number of requests.
- **Impact**: Exceeding these limits can lead to denied requests and incomplete data fetching.

Data Format Differences

- **Issue**: Different news APIs might provide data in various formats.
- **Impact**: Parsing and standardizing data can be complex and error-prone.

Dependency Management

- **Issue**: Relying on third-party APIs means dependency on their uptime and reliability.
- **Impact**: Outages or changes in external APIs can disrupt the service.

Search and Filtering-

Inefficient Search Algorithms

- **Issue**: Implementing effective search algorithms that can handle large datasets efficiently.
- **Impact**: Slow or inaccurate search results can frustrate users.

Complex Filtering Requirements

- **Issue**: Users may want to filter news by various criteria like date, category, source, etc.
- **Impact**: Building a flexible yet performant filtering system is challenging.

User Engagement and Retention-

Lack of Interactive Features

- **Issue**: Users today expect interactive features like comments, likes, and shares.
- **Impact**: Without these, user engagement can be low.

Retention Strategies

- **Issue**: Keeping users coming back to the app requires continuous engagement strategies.
- **Impact**: High churn rates can affect the platform's growth and sustainability.

Technical Debt and Maintenance-

Codebase Complexity

- **Issue**: As the project grows, the codebase can become complex and harder to maintain.
- **Impact**: This can lead to increased bugs, slower development times, and higher maintenance costs.

Dependency Updates

- **Issue**: Keeping all dependencies up to date without breaking the application.
- **Impact**: Outdated dependencies can introduce security vulnerabilities and incompatibility issues.

Addressing the Problems

To effectively tackle these problems, a systematic approach is needed:

Data Quality and Reliability

- Implement rigorous data validation and cleaning processes.
- Use reputable and consistent news sources.
- Employ real-time data fetching mechanisms.

User Experience and Engagement

- Conduct user testing and gather feedback to improve navigation.
- Implement personalized content recommendations.
- Simplify the interface and reduce information overload.

Performance and Scalability

- Optimize data fetching and rendering strategies.
- Use scalable cloud services and load balancing.
- Implement caching mechanisms for frequently accessed data.

Security and Privacy

- Encrypt sensitive data and use secure communication protocols.
- Implement role-based access control.
- Regularly audit the system for security vulnerabilities.

Content Management

- Develop robust content moderation tools.
- Use deduplication algorithms to handle duplicate content.
- Support multiple languages and localization.

Integration with External Systems

- Monitor API usage and handle rate limits gracefully.
- Standardize data formats from different sources.
- Ensure redundancy and fallback mechanisms for external dependencies.

Search and Filtering

- Use efficient search algorithms and indexing techniques.
- Implement advanced filtering options without compromising performance.

User Engagement and Retention

- Introduce interactive features like comments, likes, and shares.
- Implement retention strategies like push notifications and email updates.

CHAPTER - 4

FEASIBILITY STUDY

A feasibility study for a News API project using React.js involves a comprehensive analysis to determine the project's viability. This study covers multiple dimensions including technical, economic, legal, operational, and scheduling aspects to ensure the project's success. Below is a detailed feasibility study for the News API project.

1. Technical Feasibility-

- **Technology Stack:**

- **Frontend**: React.js is a popular JavaScript library for building user interfaces, especially single-page applications. Its component-based architecture, virtual DOM, and extensive ecosystem make it an ideal choice for developing dynamic and responsive news applications.
- **Backend**: Node.js with Express.js is suitable for building scalable and efficient RESTful APIs. It is non-blocking and event-driven, making it highly performant for I/O operations.
- **Database**: PostgreSQL or MySQL for relational database management, providing robust and reliable data storage solutions.
- **Caching**: Redis or Memcached for caching frequently accessed data to enhance performance.
- **Authentication**: JSON Web Tokens (JWT) for secure token-based authentication and OAuth for third-party login options.
- **Hosting**: Cloud platforms such as AWS, Azure, or Google Cloud for scalable and reliable deployment.
- **Version Control**: Git for source code management, with repositories hosted on GitHub or GitLab.
- **CI/CD**: Jenkins, Travis CI, or GitHub Actions for continuous integration and deployment.

- **Infrastructure:**

- **Servers**: Scalable cloud servers to handle varying loads. Initial setup can include a combination of virtual machines and containerized services using Docker and Kubernetes.
- **Load Balancers**: To distribute incoming traffic across multiple servers, ensuring high availability and reliability.
- **CDN**: Content Delivery Network to serve static assets efficiently and reduce latency for global users.

- **Technical Skills:**

- **Developers**: Proficient in JavaScript, React.js, Node.js, Express.js, and SQL.
- **DevOps Engineers**: Skilled in cloud infrastructure management, CI/CD pipeline setup, and container orchestration with Docker and Kubernetes.
- **Database Administrators**: Experienced with PostgreSQL or MySQL, database design, and performance optimization.
- **Security Experts**: Knowledgeable in web security best practices, data encryption, and access control mechanisms.

2. Economic Feasibility-

- **Cost Analysis:**
- **Initial Development Costs**:
  - **Salaries**: For developers, designers, and project managers.
  - **Software Licenses**: For development tools and libraries (if not open-source).
  - **Hardware**: For development and testing environments.
  - **Cloud Services**: Initial setup and usage costs for hosting, storage, and bandwidth.
- **Ongoing Operational Costs**:
  - **Cloud Services**: Monthly costs for hosting, database storage, CDN, and backup services.
  - **Maintenance**: Regular updates, bug fixes, and security patches.
  - **Support**: Customer support and technical support teams.

- **Marketing and Promotion**: Costs associated with promoting the API to potential users and partners.

2.2. Revenue Projections:

- **Subscription Plans**: Offering tiered subscription plans for different levels of access and usage.
- **Advertising**: Integrating ads within the news application to generate additional revenue.
- **Partnerships**: Collaborations with news organizations and other content providers.
- **Premium Features**: Offering additional premium features for advanced users.

2.3. Return on Investment (ROI):

- **Break-Even Analysis**: Estimating the time required to recover initial investments.
- **Profit Projections**: Long-term revenue projections based on user growth and market trends.

3. Legal Feasibility-

- **Data Privacy Regulations:**

- **Compliance**: Adhering to data privacy laws such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act).
- **Data Protection**: Implementing strong data protection measures, including encryption, anonymization, and secure storage practices.

- **Content Licensing:**

- **Agreements**: Ensuring that all news content is licensed appropriately, with clear agreements with news sources and content providers.
- **Intellectual Property**: Respecting intellectual property rights and avoiding copyright infringement.

- **Terms of Service and Privacy Policy**

- **User Agreements**: Drafting clear and comprehensive terms of service and privacy policies for users.
- **Legal Counsel**: Engaging legal experts to review and approve all legal documents and agreements.

CHAPTER - 5

REQUIREMENT ANALYSIS

The requirement analysis for a News API project using React.js involves a comprehensive understanding of the functional and non-functional requirements. This detailed analysis ensures that all aspects of the system are considered, including user needs, system performance, security, and scalability. Below is an extensive requirement analysis for the News API project.

1. Functional Requirements

Functional requirements describe the specific behaviors or functions of the system. These include:

- **User Authentication and Authorization**

- **User Registration**: Users should be able to create an account using their email address or social media accounts (e.g., Google, Facebook).
- **User Login**: Users should be able to log in using their registered credentials.
- **Password Management**: Users should be able to reset or change their passwords.
- **Authorization Levels**: The system should support different authorization levels, such as admin, editor, and regular user, with varying access permissions.

- **News Content Management**

- **Article Creation**: Authorized users (e.g., editors) should be able to create and publish news articles. This includes adding titles, content, images, and tags.
- **Article Editing**: Editors should be able to update existing articles.
- **Article Deletion**: Editors should be able to delete articles that are no longer relevant or contain errors.
- **Content Scheduling**: Editors should be able to schedule articles for future publication.

- **News Retrieval and Display**

- **Categories and Tags**: News articles should be categorized and tagged for easy navigation and retrieval.
- **Search Functionality**: Users should be able to search for articles based on keywords, categories, tags, and publication dates.
- **Pagination**: The system should support pagination for displaying large numbers of articles.
- **Sorting and Filtering**: Users should be able to sort and filter articles based on criteria such as date, popularity, and relevance.

- **User Interaction**

- **Comments**: Users should be able to comment on articles. Comments should be moderated by editors.
- **Likes and Shares**: Users should be able to like and share articles on social media platforms.
- **Notifications**: Users should receive notifications for activities such as new articles in their favorite categories, replies to their comments, and system updates.

- **API Endpoints**

- **Public API**: Provide endpoints for retrieving news articles, categories, and tags.
- **Private API**: Provide endpoints for article management, user authentication, and authorization.
- **Rate Limiting**: Implement rate limiting to prevent abuse of the API.

2. Non-Functional Requirements

Non-functional requirements define the system's operational attributes such as performance, security, and usability.

- **Performance Requirements**

- **Response Time**: The system should respond to user requests within 2 seconds.
- **Throughput**: The system should handle at least 1000 requests per second.
- **Scalability**: The system should be able to scale horizontally to handle increasing loads.

- **Security Requirements**
- **Usability Requirements**

- **User Interface**: The user interface should be intuitive and responsive, providing a seamless experience across devices (desktops, tablets, and smartphones).
- **Accessibility**: Ensure the application meets accessibility standards such as WCAG 2.1.
- **Documentation**: Provide comprehensive user documentation and API documentation.

- **Reliability Requirements**

- **Uptime**: The system should have an uptime of 99.9%.
- **Error Handling**: Implement robust error handling and logging mechanisms to track and resolve issues.
- **Backup and Recovery**: Regular backups should be performed, and a disaster recovery plan should be in place.

- **Maintainability Requirements**

- **Code Quality**: Follow best practices for code quality, including modular design, code reviews, and automated testing.
- **Documentation**: Maintain detailed technical documentation for developers.
- **Continuous Integration/Continuous Deployment (CI/CD)**: Implement CI/CD pipelines to automate testing and deployment processes.

3. System Requirements

System requirements outline the hardware and software necessary to build and run the application.

Hardware Requirements

- **Development Environment**:
  - **Processor**: Multi-core processor (e.g., Intel Core i5/i7 or AMD Ryzen 5/7).
  - **Memory (RAM)**: At least 16 GB of RAM.
  - **Storage**: SSD with at least 512 GB capacity.
  - **Network**: Reliable broadband internet connection.
- **Production Environment**:
  - **Web Servers**: Multi-core processor(e.g Intel Xeon or AMD EPYC), atleast 64 GB of RAM, SSD with at least 1 TB capacity, dual 10 Gigabit Ethernet connections.
  - **Database Servers**: Multi-core processors (e.g., Intel Xeon or AMD EPYC), atleast 128 GB of RAM, SSDs with at least 2 TB capacity in RAID 10 configuration, dual 10 Gigabit Ethernet connections.
  - **Load Balancers**: Hardware or virtual load balancers for traffic distribution.
  - **Content Delivery Network (CDN)**: For serving static assets and reducing latency.

Software Requirements

- **Frontend**:
  - **React.js**: JavaScript library for building user interfaces.
  - **Redux**: State management library.
  - **React Router**: For handling client-side routing.
  - **Axios**: For making HTTP requests.
  - **Webpack**: Module bundler.
  - **Babel**: JavaScript compiler.
- **Backend**:
  - **Node.js**: JavaScript runtime for server-side development.
  - **Express.js**: Web framework for Node.js.
  - **PostgreSQL/MySQL**: Relational database management systems.
  - **Redis/Memcached**: For caching.
  - **JWT**: For authentication.
  - **Nginx/Apache**: Web server software.

- **Development Tools**:
    - **Visual Studio Code**: Integrated development environment (IDE).
    - **Git**: Version control system.
    - **Docker**: Containerization platform.
    - **Kubernetes**: Container orchestration platform.
    - **Jenkins/Travis CI/GitHub Actions**: CI/CD tools.

4. User Requirements

User requirements focus on what the end-users need from the system.

- **User Personas**

- **General Users**: Want to read news articles, search for topics, and interact with content.
- **Editors**: Need to manage news content, moderate comments, and ensure quality.
- **Administrators**: Require full access to manage the system, users, and content.

- **User Stories**

- **As a general user, I want to**:
    - **Read News Articles**: Easily browse and read news articles.
    - **Search for Topics**: Use a search bar to find specific topics or articles.
    - **Comment on Articles**: Leave comments on articles and interact with other users.
    - **Like and Share Articles**: Like articles and share them on social media.
    - **Receive Notifications**: Get notified about new articles and replies to my comments.
- **As an editor, I want to**:
    - **Create Articles**: Write and publish new articles.
    - **Edit Articles**: Update existing articles.
    - **Delete Articles**: Remove outdated or incorrect articles.
    - **Moderate Comments**: Approve or delete user comments.
    - **Schedule Publications**: Schedule articles to be published at specific times.
- **As an administrator, I want to**:

- **Manage Users**: Add, edit, or remove users and assign roles.
- **Monitor System Health**: View system performance metrics and logs.
- **Backup Data**: Ensure regular backups of the database and content.
- **Configure Settings**: Adjust system settings and configurations.

5. Regulatory and Compliance Requirements

Ensuring compliance with relevant laws and regulations is crucial.

- **Data Privacy**

- **GDPR**: Compliance with the General Data Protection Regulation for users in the European Union.
- **CCPA**: Compliance with the California Consumer Privacy Act for users in California.
- **Data Encryption**: Encrypt sensitive user data both in transit and at rest.

- **Accessibility**

- **WCAG 2.1**: Compliance with Web Content Accessibility Guidelines ensure application is accessible to users with disabilities.

- **Content Regulations**

- **Copyright Laws**: Respect and adhere to copyright laws and obtain necessary licenses for news content.
- **User-Generated Content**: Implement policies and tools to manage and moderate user-generated content to avoid illegal or inappropriate material.

6. Performance Requirements

- **Scalability**

- **Horizontal Scaling**: Ability to add more servers to handle increased load.
- **Vertical Scaling**: Ability to increase resources (CPU, RAM) of existing servers.

- **Load Testing**

- **Stress Testing**: Simulate high load conditions to ensure the system can handle peak traffic.
- **Performance Monitoring**: Use tools like New Relic, Datadog, or Prometheus to monitor system performance in real-time.

7. Maintenance Requirements

- **Regular Updates**

- **Software Patches**: Apply security patches and updates regularly.
- **Feature Enhancements**: Continuously improve the system based on user feedback and technological advancements.

- **Technical Support**

- **Helpdesk**: Provide a helpdesk for user support.
- **Documentation**: Maintain up-to-date documentation for users and developers.

8. Stakeholder Identification:

- Identify stakeholders involved in the project, including end-users, administrators, content creators, and developers.
- Understand their roles, responsibilities, and expectations regarding the news platform**.**

9. Gathering Requirements:

- Conduct interviews, surveys, workshops, and brainstorming sessions with stakeholders to gather their requirements and preferences.
- Document user stories, use cases, and scenarios to capture the functional requirements of the system.

- Identify key features, functionalities, and interactions expected from the news platform, such as article browsing, search, filtering, user registration, and content sharing.

10. Functional Requirements:

Define the core functionalities and features of the news platform, including:

- **User Authentication:**
  User registration, login, logout, and password management.

- **Article Management:**
  Creation, editing, deletion, and categorization of articles by administrators.

- **Content Discovery:**
  Browsing articles by category, topic, publication date, and popularity.

- **Search and Filtering:**
  Advanced search functionality with filters for refining search results.

- **Personalization:**
  Personalized content recommendations based on user preferences and behavior.

- **Interaction:**
  Liking, commenting, sharing, and bookmarking articles.

- **Real-Time Updates:**
  Automatic fetching and display of new articles in real-time.

CHAPTER - 6

SYSTEM DESIGN

The system design for a News API project using React.js encompasses architectural design, data flow, database schema, and detailed descriptions of the various components and interactions within the system. This comprehensive design ensures a scalable, reliable, and high-performing application.

1. Architectural Design

- **Architecture Overview**

The architecture of the News API project follows a microservices architecture, which divides the system into distinct, loosely coupled services that communicate over a network. This approach enhances scalability, maintainability, and flexibility.

- **Components**

- **Frontend**: Built with React.js, providing a responsive and dynamic user interface.
- **Backend**: Developed using Node.js and Express.js, handling API requests and business logic.
- **Database**: PostgreSQL or MySQL for relational data storage, ensuring robust and reliable data management.
- **Caching**: Redis or Memcached for caching frequently accessed data to improve performance.
- **Authentication**: JWT for token-based authentication and OAuth for third-party logins.
- **Load Balancer**: Distributes incoming traffic across multiple servers to ensure high availability and reliability.
- **CDN**: Delivers static assets efficiently and reduces latency for global users.

- **Layers**

- **Presentation Layer**: The React.js frontend that interacts with users, handling user input and displaying data.
- **Business Logic Layer**: The Node.js backend that processes requests, applies business logic, and interacts with the database.
- **Data Layer**: The PostgreSQL/MySQL database that stores and retrieves data, supported by caching mechanisms for efficiency.
- **Integration Layer**: APIs that connect the system with third-party services, including social media platforms for sharing articles and external news sources for fetching content.

2. Data Flow Diagram

The Data Flow Diagram (DFD) illustrates how data moves through the system, from user interactions to backend processing and database operations.

Context Diagram (Level 0)

The Context Diagram provides a high-level overview of the entire system, highlighting the main entities and data interactions.

//diff

External Entities:
- Users (general users, editors, administrators)
- Social Media Platforms
- External News Sources

Processes:
- News API System

Data Stores:
- User Database

- Article Database

Data Flows:
- User requests and responses (login, article retrieval, comments)
- Content publication and updates (from editors)
- Authentication data (to/from social media platforms)
- News content fetching (from external sources)

Level 1 DFD

The Level 1 DFD breaks down the system into major processes and data flows.
markdown
Copy code
Processes:
1. User Management
Registration, login, and profile management
2. Content Management
Article creation, editing, deletion, and scheduling
3. News Retrieval
Fetching and displaying articles, search functionality
4. Interaction Management
Commenting, liking, sharing articles
5. External Integration
Social media integration, external news source integration

Data Stores:
- User Database
- Article Database
- Caching System

Data Flows:
- User data (registration, login, profile updates)
- Article data (creation, updates, deletion)
- Comments and interaction data
- External content and integration data

3. Database Schema

The database schema defines the structure of the database, including tables, fields, and relationships. Here, PostgreSQL is used as the example relational database.

User Table

Stores user information and authentication details.

```
//sql

CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('admin', 'editor', 'user')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Article Table

Stores news articles.

```
//sql

CREATE TABLE Articles (
    article_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    author_id INTEGER REFERENCES Users(user_id),
    category VARCHAR(50),
    tags VARCHAR(255),
    published_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

```
//sql

CREATE TABLE Comments (
    comment_id SERIAL PRIMARY KEY,
    article_id INTEGER REFERENCES Articles(article_id),
    user_id INTEGER REFERENCES Users(user_id),
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Like Table

Stores likes for articles.

```
//sql

CREATE TABLE Likes ( like_id SERIAL PRIMARY KEY, article_id INTEGER REFERENCES
Articles(article_id), user_id INTEGER REFERENCES Users(user_id), created_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

4. Component Design

Detailed descriptions of each system component and their interactions.

Frontend (React.js)

- **Components**:
    - **Header**: Navigation bar with links to different sections.
    - **ArticleList**: Displays a list of articles with pagination.
    - **ArticleDetail**: Shows detailed view of a single article.
    - **CommentSection**: Allows users to view and post comments.
    - **UserProfile**: Displays user profile information and allows updates.
    - **AdminDashboard**: Admin interface for managing users and content.
- **State Management**: Redux is used for global state management, handling user authentic-
- cation status, article data, and user interactions.
- **Routing**: React Router manages client-side routing, providing a single-page application experience.
- **API Integration**: Axios is used to make HTTP requests to the backend API, handling data retrieval and submission.

Backend (Node.js and Express.js)

- **Authentication Service**:
    - **Endpoints**:
        - /auth/register: User registration.
        - /auth/login: User login.
        - /auth/logout: User logout.
        - /auth/forgot-password: Password reset initiation.
        - /auth/reset-password: Password reset completion.
    - **Middleware**: JWT verification middleware to protect routes requiring authentication.
- **Article Service**:
    - **Endpoints**:
        - /articles: GET for listing articles, POST for creating articles.
        - /articles/:id: GET, PUT, DELETE for retrieving, updating, and deleting a specific article.

        - /articles/search: GET for searching articles by keyword, category, or tags.
    - **Business Logic**: Handles article creation, updates, deletions, and search functionality.
- **Comment Service**:
    - **Endpoints**:
        - /comments: POST for adding a comment.
        - /comments/:id: GET, PUT, DELETE for retrieving, updating, and deleting a specific comment.
    - **Moderation**: Admin and editor routes for moderating comments.
- **Like Service**:
    - **Endpoints**:
        - /likes: POST for liking an article.
        - /likes/:id: DELETE for unliking an article.
    - **Caching**: Use Redis to cache like counts for articles to improve performance.

Database (PostgreSQL/MySQL)

- **Normalization**: Ensure that the database schema is normalized to at least the third normal form (3NF) to eliminate redundancy and improve data integrity.
- **Indexes**: Create indexes on frequently searched fields (e.g., article titles, tags) to speed up queries.
- **Backups**: Regular automated backups and replication for disaster recovery.

5. Security Design

Authentication and Authorization

- **JWT**: Use JSON Web Tokens for secure, stateless authentication.
- **OAuth**: Implement OAuth 2.0 for third-party authentication (e.g., Google, Facebook).
- **Role-Based Access Control (RBAC)**: Define roles (admin, editor, user) and enforce permissions based on roles.

Data Protection

- **Encryption**:
    - **In Transit**: Use HTTPS to encrypt data transmitted between the client and server.
    - **At Rest**: Encrypt sensitive data stored in the database, such as passwords and personal information.
- **SQL Injection Prevention**: Use parameterized queries or ORM tools to prevent SQL injection attacks.
- **Cross-Site Scripting (XSS) Prevention**: Sanitize user inputs and use secure coding practices to prevent XSS attacks.

Security Monitoring

- **Intrusion Detection Systems (IDS)**: Implement IDS to monitor for suspicious activities.
- **Regular Audits**: Conduct regular security audits and penetration testing to identify and address vulnerabilities.

6. Performance Optimization

- **Caching**

- **Redis/Memcached**: Cache frequently accessed data such as article lists and like counts to reduce database load.
- **CDN**: Use a Content Delivery Network to serve static assets (images, scripts) quickly to users across the globe.

- **Load Balancing**

- **Load Balancer**: Distribute incoming requests across multiple backend servers to ensure even load distribution and high availability.

- **Database Optimization**

- **Indexes**: Create indexes on columns that are frequently used in queries to speed up data retrieval.

- **Query Optimization**: Analyze and optimize SQL queries to reduce execution time.

7. Deployment and Maintenance

- **Continuous Integration/Continuous Deployment (CI/CD)**

- **Pipeline**: Set up a CI/CD pipeline using tools like Jenkins, Travis CI, or GitHub Actions to automate testing and deployment processes.
- **Automated Testing**: Implement unit tests, integration tests, and end-to-end tests to ensure code quality and prevent regressions.

- **Monitoring and Logging**

- **Monitoring Tools**: Use tools like Prometheus, Grafana, and New Relic to monitor system performance and health.
- **Logging**: Implement centralized logging using tools like ELK Stack (Elasticsearch, Log - stash, Kibana) or Splunk to collect and analyze logs for debugging and auditing.

6.1 SYSTEM TOOL

For a project involving the development of a news platform using React.js and integrating with a news API, you would need a combination of tools and technologies to build, deploy, and manage the system effectively. Here's a list of system tools for such a project:

1. **React.js:**
   As the core frontend framework, React.js enables the creation of dynamic and interactive user interfaces for the news platform. It efficiently manages the UI components, state, and data flow, providing a rich and responsive user experience. A JavaScript library for building user interfaces, React.js is the primary frontend framework for developing the client-side application of the news platform.

2. **Node.js:**

   Serving as the backend runtime environment, Node.js allows executing JavaScript code on the server side. It facilitates building scalable and efficient backend APIs to handle data processing, authentication, and interaction with external services like the News API. A Java Script runtime environment that allows you to run JavaScript code on the server side. Node.js is commonly used for building backend server applications in JavaScript.

3. **Express.js:**

   A minimalist web application framework for Node.js, Express.js simplifies the process of building RESTful APIs and handling HTTP requests and responses. It provides robust routing, middleware support, and modular architecture for developing the backend server application. A web application framework for Node.js, Express.js simplifies the process of building APIs and handling HTTP requests and responses on the server side.

4. **News API:**

   The News API provides access to news articles and headlines from various sources worldwide. You'll use this API to fetch news data and display it on your platform. Serving as the primary source of news data, the News API offers access to a vast repository of articles, headlines, and news sources from around the world. Integrating with the News API allows fetching real-time news data and presenting it to users on the news platform.

5. **Axios or Fetch API:**

   JavaScript libraries for making HTTP requests from the frontend client application to the backend server or external APIs. You'll use Axios or Fetch API to fetch news data from the News API. JavaScript libraries for making HTTP requests from the frontend client application to the backend server or external APIs like the News API. These libraries facilitate data fetching and handling asynchronous operations, ensuring seamless communication between the frontend and backend components.

6. **Redux or React Context API:**

   State management libraries for managing global application state in React.js. Redux or React Context API helps manage data flow and state synchronization between components.

State management libraries for managing global application state in React.js. Redux or React Context API helps maintain shared state across multiple components, ensuring data consistency and synchronization throughout the application.

7. **React Router:**

A routing library for React.js that allows you to define and handle client-side routing and navigation within the application. A routing library for React.js that enables defining and handling client-side routing and navigation within the application. React Router facilitates creating multiple routes and rendering different components based on URL paths, enhancing the user experience with seamless navigation.

8. **Semantic UI React or Material-UI:**

UI component libraries for React.js that provide pre-styled components and layout grids to enhance the visual appearance and usability of the news platform.UI component libraries for React.js that offer a comprehensive set of pre-styled components and layout grids.

9. **JWT (JSON Web Tokens) or OAuth 2.0:**

Authentication protocols for securing access to protected resources and implementing user authentication and authorization mechanisms in the system. Authentication protocols for securing access to protected resources and implementing user authentication and authorization mechanisms. JWT or OAuth 2.0 enables secure authentication and authorization flows, ensuring data privacy and user security.

10. **Docker and Docker Compose:**

Containerization tools for packaging the application into containers, which ensure consistency and portability across different environments. Containerization tools for packaging the application into lightweight and portable containers.

11. **Git and GitHub (or GitLab, Bitbucket):**

Version control system and code collaboration platform for managing codebase changes, tracking issues, and facilitating collaboration among team members. Version control system

and code collaboration platform for managing the project's codebase, tracking changes, and facilitating collaboration among team members. Git and platforms like GitHub offer features such as branching, merging, and pull requests, enabling efficient code management and review.

12. CI/CD Pipeline (e.g., GitHub Actions, Travis CI, Jenkins):

Continuous integration and continuous deployment pipelines for automating the build, testing, and deployment processes of the application. Continuous integration and continuous deployment pipelines for automating the build, testing, and deployment processes of the application. CI/CD pipelines enable rapid and reliable delivery of updates and new features, ensuring the stability and quality of the news platform.

13. Hosting and Deployment Platforms (e.g., AWS, Heroku, Netlify, Vercel):

Cloud hosting providers and deployment platforms for hosting and deploying the frontend and backend components of the news platform. Cloud hosting providers and deployment platforms for hosting and deploying the frontend and backend components of the news platform. These platforms offer scalable infrastructure, managed services, and deployment workflows for deploying the application to production environments.

14. Logging and Monitoring Tools (e.g., Sentry, LogRocket, Datadog):

Tools for logging errors, monitoring application performance, and gaining insights into user behavior and system healthtools for logging errors, monitoring application performance, and gaining insights into user behavior and system health.

6.2 SYSTEM MODULES

For a news platform project using React.js and integrating with a news API, several modules or components can be identified to organize the system's functionalities effectively. Here are the key system modules:

1. **Authentication Module:**

    - This module manages user authentication and authorization processes.
    - Features include user registration, login, logout, and password management.
    - It implements security protocols like JWT (JSON Web Tokens) or OAuth 2.0 for secure access to user accounts and protected resources.

2. **Article Management Module:**

    - Responsible for the creation, editing, deletion, and categorization of articles.
    - Provides administrative functionalities for content moderation and approval workflows.
    - Allows content creators to publish articles, manage drafts, and schedule publication dates.

3. **Article Display Module:**

    - Retrieves and presents news articles and headlines from external sources via the News API.
    - Allows users to browse articles by categories, topics, publication dates, and popularity.

4. **User Interaction Module:**

    - Facilitates user engagement with articles through actions like liking, commenting, sharing, and bookmarking.

5. **Personalization Module:**

- Offers personalized content recommendations based on user preferences, browsing history, and behavior.
- Utilizes machine learning algorithms or collaborative filtering techniques to tailor content suggestions to individual users' interests.

6. **Notification Module:**

- Sends notifications to users about new articles, updates, or relevant events.
- Allows users to customize their notification preferences, including frequency, delivery channels, and notification types.

7. **Search Module:**

- Implements a powerful search functionality for users to discover articles based on keywords, topics, authors, or specific criteria.
- Utilizes search indexing and ranking algorithms to deliver accurate and relevant search results in real-time.

8. **Analytics Module:**

- Tracks user interactions, engagement metrics, and article popularity.
- Provides insights into user behavior, traffic patterns, and content performance through analytics dashboards and reports.
- Enables data-driven decision-making and content optimization strategies based on actionable insights.

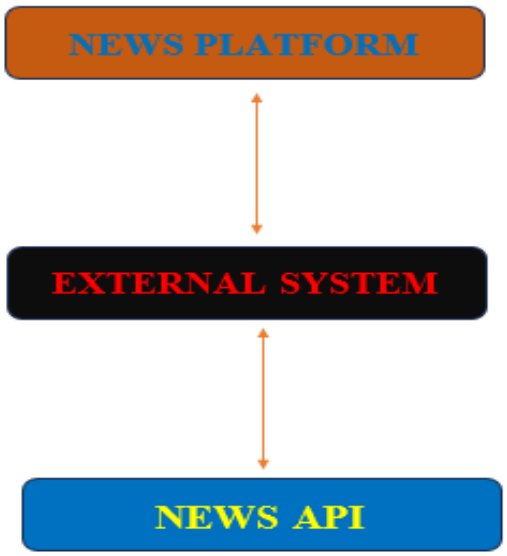
9. **Settings Module:**
   - Allows users to customize their profile settings, including account information, profile picture, and notification preferences.

10. **Admin Dashboard Module:**

- Offers administrative tools and dashboards for managing users, articles, comments, and other system components.
- Provides analytics and reporting features for monitoring system performance, user activity, and content trends.
- Enables content moderation, user management, and system configuration from a centralized interface.

6.3 CONTEXT DIAGRAM

A context diagram provides a high-level overview of a system, illustrating its interactions with external entities or systems. In the context of a news platform project using React.js and integrating with a news API, the context diagram would depict the system's interactions with various external entities or systems. Here's an example of a context diagram for such a project:



In this context diagram:

- The "News Platform" represents the system being developed, which includes the frontend client application built with React.js and the backend server application.
- The "External Systems" box encapsulates all external entities or systems that interact with the news platform.
- The main external system depicted here is the "News API," which serves as the primary source of news articles and headlines for the platform. The news platform interacts with the News API to fetch news data and present it to users.

6.4. DATA FLOW DIAGRAMS

Creating a Data Flow Diagram (DFD) for a News API involves illustrating how data moves through the system. A DFD typically includes different levels, starting from a high-level overview (Level 0) and drilling down into more detailed views (Level 1, Level 2, etc.).Here is a basic outline of what the DFDs might look like for a News API:

Level 0 DFD (Context Diagram)
This diagram provides a high-level overview of the News API system. It shows the system as single process with external entities interacting with it.
Entities:

1. **User**: The individual requesting news data.
2. **News Providers**: External sources from which news data is fetched.

Process:

1. **News API System**

Data Flows:

1. **User Request**: User sends a request for news data to the News API System.

2. **News Data**: News Providers send news data to the News API System.
3. **News Response**: The News API System sends the requested news data back to the User.

Level 1 DFD

This diagram breaks down the high-level process into subprocesses to show more detail about how data is processed within the News API System.

Processes:

1. **User Request Handling**: Handles incoming requests from users.
2. **News Data Fetching**: Fetches news data from external news providers.
3. **Data Processing and Filtering**: Processes and filters the news data as per user requests.
4. **Response Generation**: Generates the response to be sent back to the user.

Data Stores:

1. **News Data Cache**: Temporary storage for fetched news data to improve performance.

Data Flows:

1. **User Request**: From User to User Request Handling.
2. **Request Details**: From User Request Handling to News Data Fetching.
3. **Raw News Data**: From News Providers to News Data Fetching.
4. **Fetched News Data**: From News Data Fetching to Data Processing and Filtering.
5. **Processed News Data**: From Data Processing and Filtering to Response Generation.
6. **News Response**: From Response Generation to User.
7. **Cached News Data**: From News Data Fetching to News Data Cache.
8. **Cache Lookup**: From Data Processing and Filtering to News Data Cache.

Level 2 DFD (Detailed Processes)

This diagram further break down one of the Level 1 process for more detailed view. For instance, breaking down "Data Processing and Filtering."

Subprocesses:

1. **Data Parsing**: Parses the fetched news data into a structured format.

2. **Data Filtering**: Filters news data based on user criteria (e.g., keywords, categories).
3. **Data Enrichment**: Adds additional information to the news data if needed (e.g., sentiment analysis).

Data Flows:

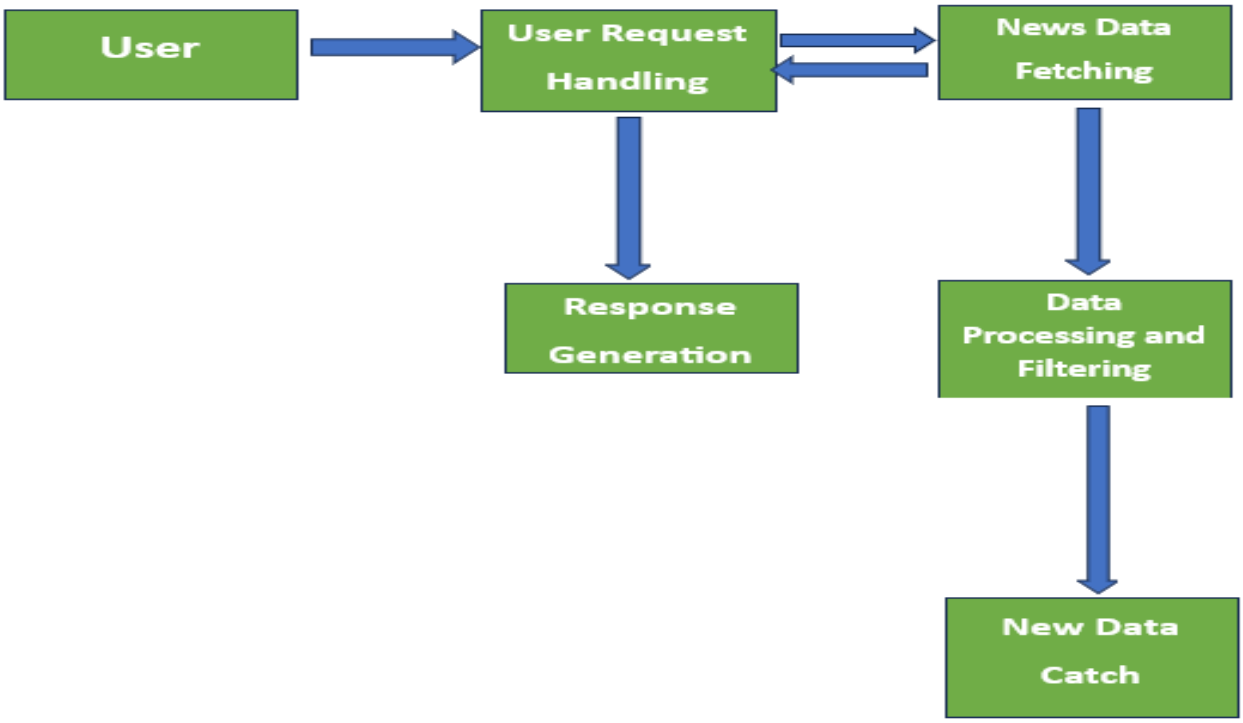
1. **Parsed Data**: From Data Parsing to Data Filtering.
2. **Filtered Data**: From Data Filtering to Data Enrichment.
3. **Enriched Data**: From Data Enrichment to Response Generation.

Example Diagrams

Let's outline these diagrams graphically:

Level 0 DFD (Context Diagram)

Level 1 DFD



Level 2 DFD (Data Processing and Filtering)

These diagrams provide visual representation of the data flow within News API system, showing how user requests are handled, how data is fetched and processed, & how response are generated. Data Flow Diagrams (DFDs) illustrate the flow of data within a system, depicting how data move from one process to another and how it's stored, processed, or transformed along the way. In the context of a news platform project using React.js and integrating with a news API, let's create a high-level DFD to outline the data flow within the system:

In this DFD:

- **External Sources:**
  External sources, such as news agencies or publishers, provide data to the system.

- **News API:**
  The system interacts with the News API to fetch news articles and headlines from external sources.

- **Backend Server:**
  The backend server processes and manages the data received from the News API, performs any necessary transformations or validations, and prepares it for consumption by the frontend application.

- **Browser:**
  The browser serves as the platform for users to access the news platform. It renders the frontend application built with React.js and presents the user interface to the users.

- **Processing:**
  The processing component represents the backend server's functionality to process the data received from the News API. This includes tasks such as data validation, transformation, aggregation, or any other business logic required before sending the data to the frontend application

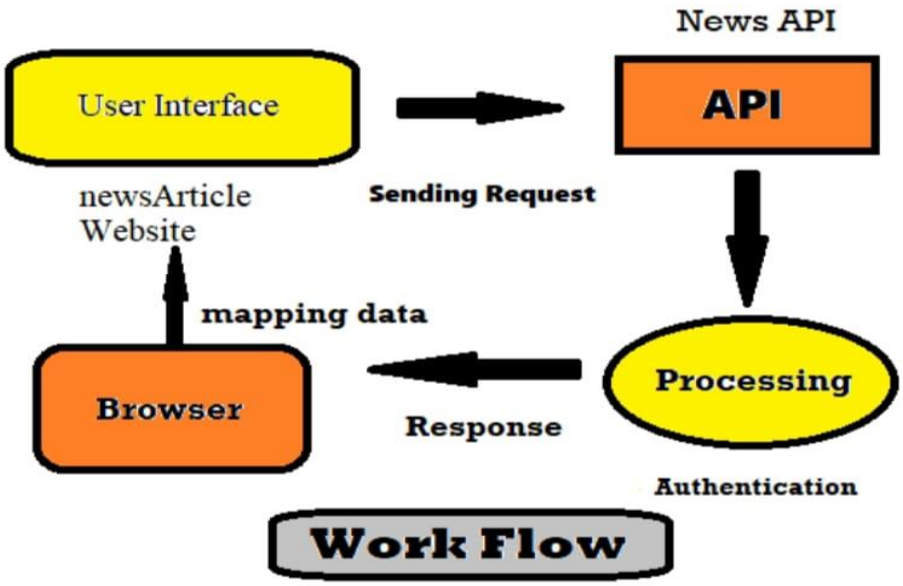
- **Frontend App:**

  The frontend application, built with React.js, receives the processed data from the backend server and presents it to users through the user interface.

- **User Interface:**

  The user interface is where users interact with the news platform, viewing articles, reading news, and performing various actions such as liking, commenting, or sharing.

6.5 FLOW CHART



In this flowchart:

- The process begins when a user accesses the news platform.

- The user interacts with the user interface (UI), browsing articles and interacting with features.
- The UI requests data from the backend server to fulfill the user's actions.
- The backend server processes the request, retrieving data from external sources (like the News API) and applying business logic as needed.
- The backend server sends the processed data back to the frontend.
- The frontend updates the UI with the received data, allowing the user to continue interacting.
- The process continues until the user decides to leave the platform.

6.6 ENTITY RELATIONSHIP DIAGRAM

Entity Relationship Diagram (ERD) for a News API involves identifying the main entities and their relationships within the system. Since a News API typically serves as a source of news articles and related information, The ERD helps to design the database schema and understand the interactions between different parts of the system. Below is a detailed ERD along with an explanation of each entity and their relationships.

Entities and Attributes

1. **User**
    - **UserID** (Primary Key): Unique identifier for each user.
    - **UserName**: Name of the user.
    - **Email**: Email address of the user.
    - **Password**: Password for user authentication.
    - **CreatedAt**: Timestamp when the user was created.
2. **Article**
    - **ArticleID** (Primary Key): Unique identifier for each article.
    - **Title**: Title of the article.
    - **Content**: Main content/body of the article.
    - **Author**: Author of the article.
    - **PublishedDate**: Date the article was published.

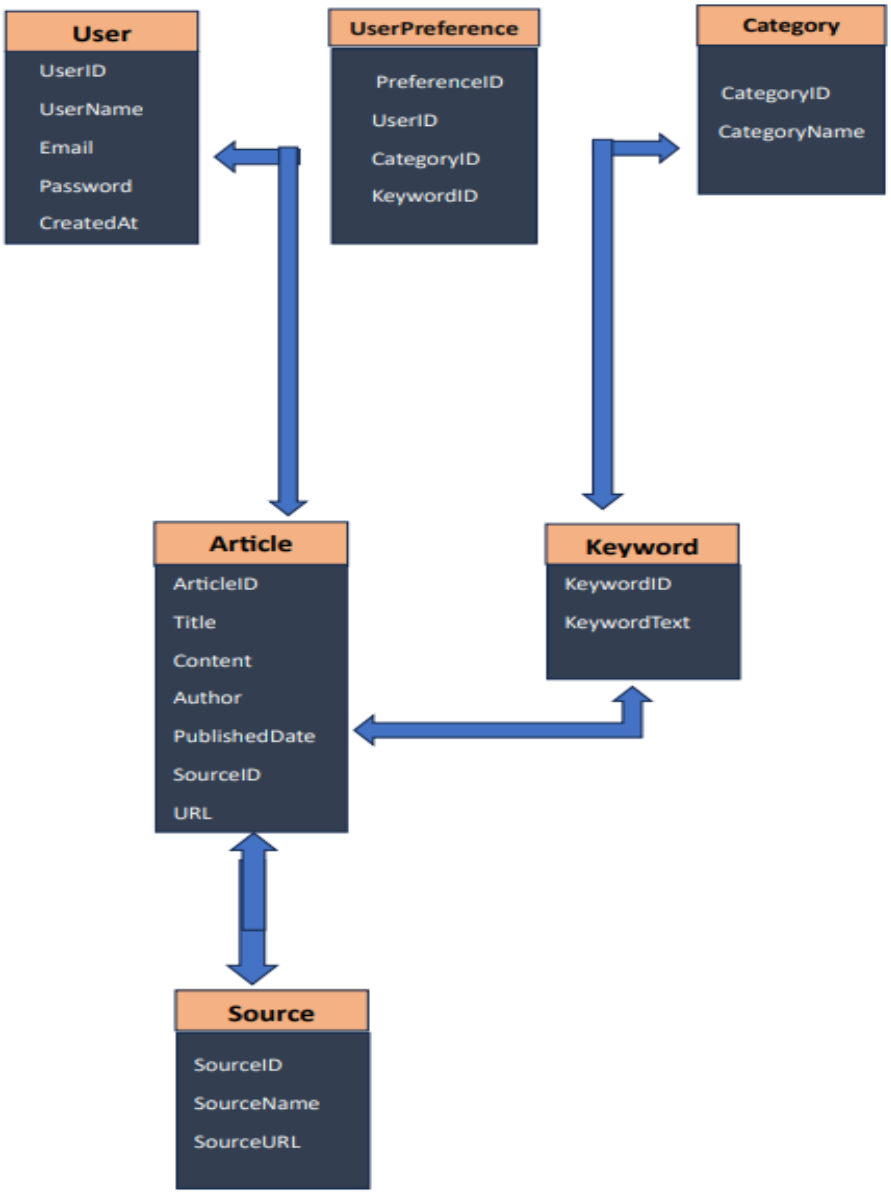
- **SourceID** (Foreign Key): Identifier for the source from which the article originated.
- **URL**: URL of the article.

3. **Source**
    - **SourceID** (Primary Key): Unique identifier for each news source.
    - **SourceName**: Name of the news source.
    - **SourceURL**: URL of the news source.
4. **Category**
    - **CategoryID** (Primary Key): Unique identifier for each news category.
    - **CategoryName**: Name of the category (e.g., Technology, Health, Sports).
5. **Keyword**
    - **KeywordID** (Primary Key): Unique identifier for each keyword.
    - **KeywordText**: Text of the keyword (e.g., "AI", "COVID-19").
6. **UserPreference**
    - **PreferenceID** (Primary Key): Unique identifier for each preference entry.
    - **UserID** (Foreign Key): Identifier for the user.
    - **CategoryID** (Foreign Key): Identifier for the category.
    - **KeywordID** (Foreign Key): Identifier for the keyword.

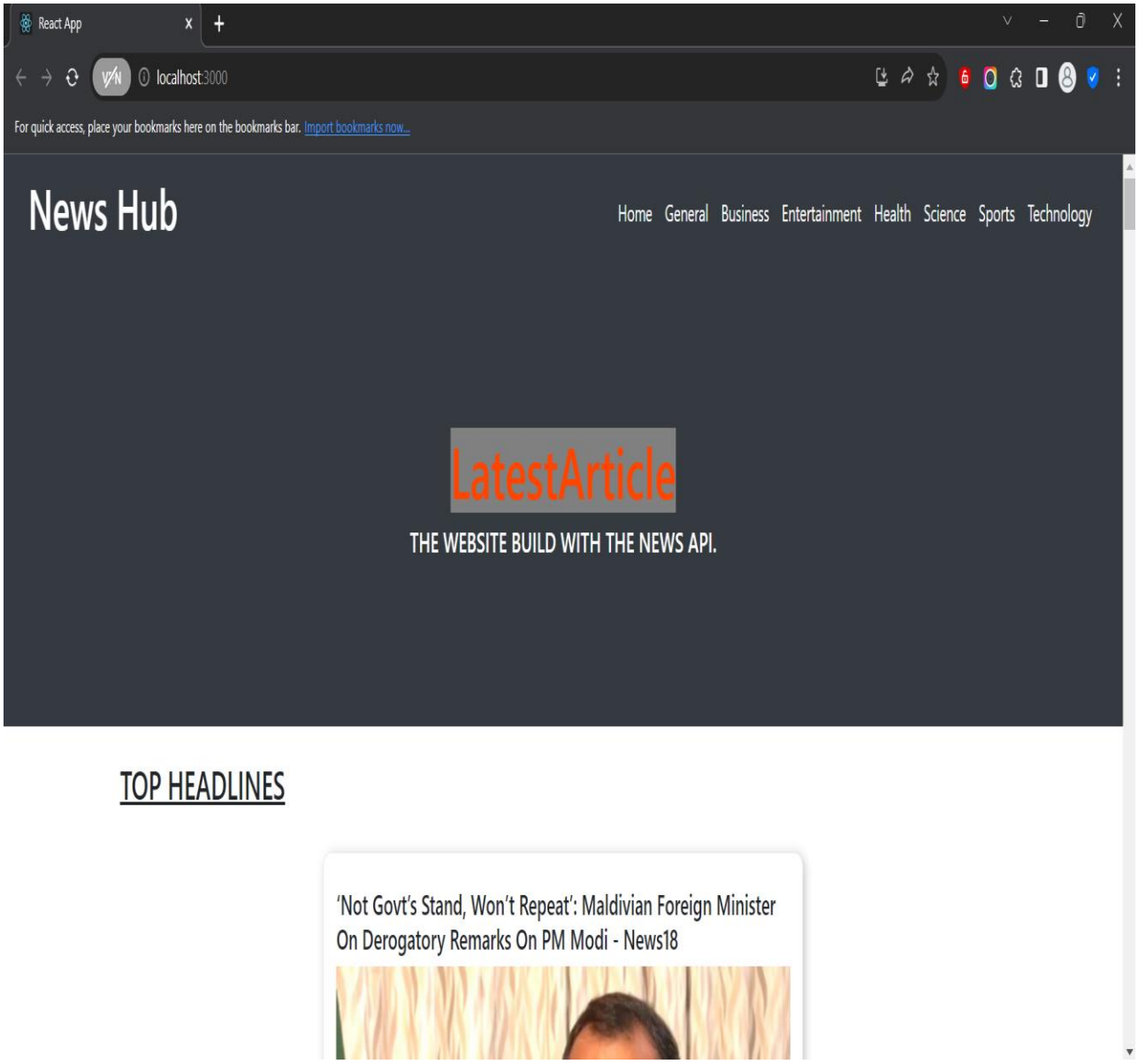
Relationships

1. **User to UserPreference**
    - One-to-Many: A user can have multiple preferences, including multiple categories and keywords.
2. **Article to Source**
    - Many-to-One: Multiple articles can originate from one source.
3. **Article to Category**
    - Many-to-Many: An article can belong to multiple categories, and a category can include multiple articles.
4. **Article to Keyword**
    - Many-to-Many: An article can have multiple keywords, and a keyword can be associated with multiple articles.

5. **Category to UserPreference**
    - Many-to-Many: A category can be a preference for multiple users, and a user can have multiple category preferences.
6. **Keyword to UserPreference**
    - Many-to-Many: A keyword can be a preference for multiple users, and a user can have multiple keyword preferences.



7. SNAPSHOTS OF THE PROJECT



1(A)

- In this experiment we are getting news articles data from NewsAPI.
- The 1(A) photo was taken through a ScreenShot. In which there are plenty of news articles loaded with images titles and description etc.

Live Latest Articles

TOP HEADLINES

'Not Govt's Stand, Won't Repeat': Maldivian Foreign Minister On Derogatory Remarks On PM Modi - News18

Referring to some Maldivian ministers derogatory remarks against Prime Minister Narendra Modi earlier this year, the island nations Foreign Minister Moosa Zameer who is on his first official visit to... [+3589 chars]

View More..

After West Bengal governor's video show, woman threatens suicide over identity leak - The Times of India

View More..

IPL 2024, PBKS vs RCB Highlights: Punjab lose by 60 runs, knocked out - Hindustan Times

first-ever look at the light of ancient stars shining around some of the biggest, brightest and oldest black holes in the un... [+4048 chars]

View More..

NASA ALERT! Asteroid 2024 JT3 To Come Scarily Close To Earth Today: Check Size, Speed And Time - Times Now

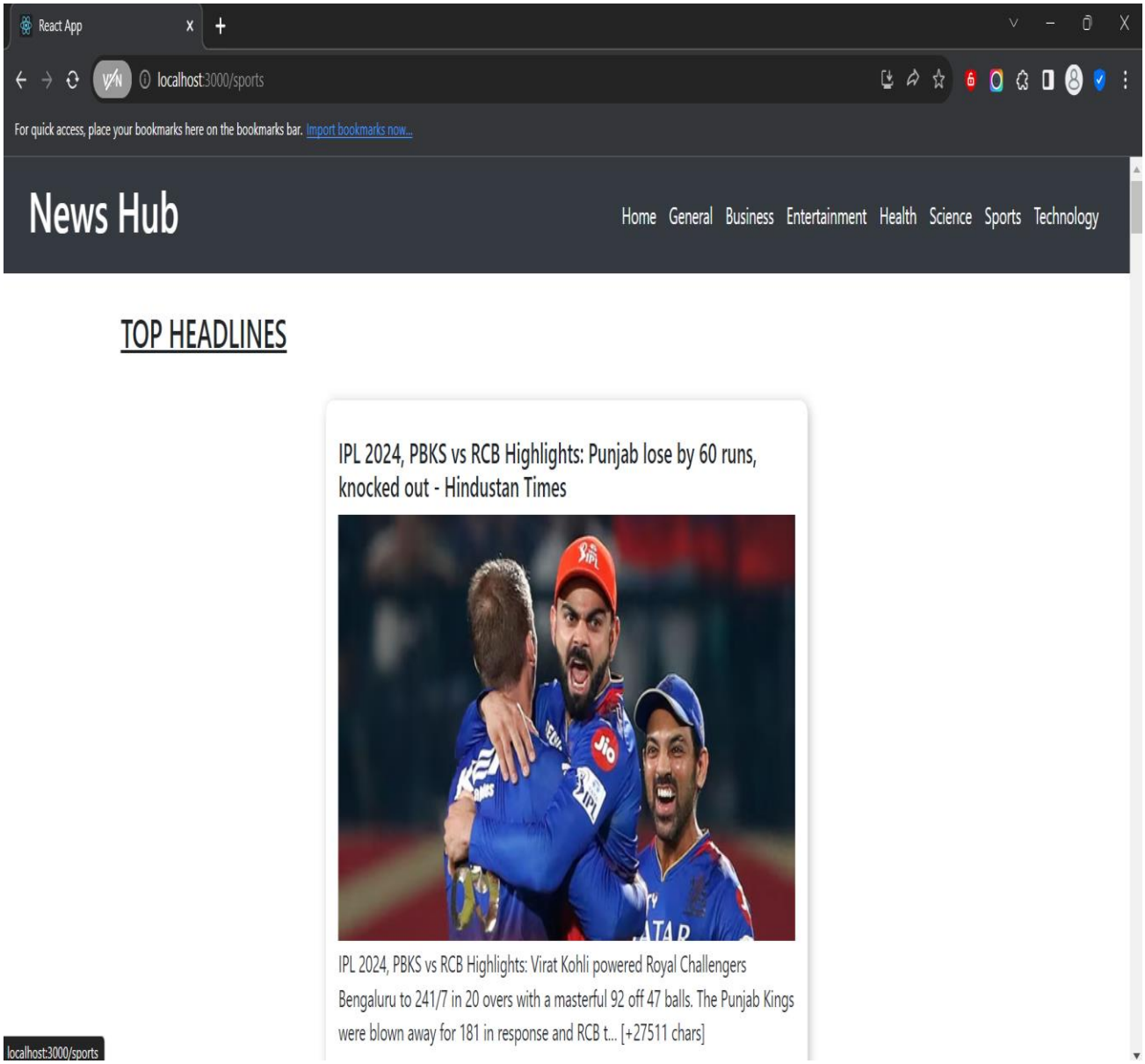
Pan India Stars Allu Arjun, Jr NTR and Ram Charan Stay Away from AP Election Campaigns

View More..

NASA's PACE mission is helping scientists understand interactions between oceans, atmosphere - News9 LIVE

A rendering of NASA's PACE satellite. (Image Credit: NASA). New Delhi: A SpaceX Falcon 9 launched the Plankton, Aerosol, Cloud, ocean Ecosystem (PACE) satellite on 8 February, 2024, after bad weath... [+2183 chars]

View More..



1(B)

● The 1(B) photo is taken from website that is describe about the indivisual news items reflected to the page .

8 . DATABASE AND TABLE

To design a database schema for a News API, we'll define the necessary tables to store information about articles and their categories. We'll use SQL syntax to create the tables:

```sql
-- Table: Category
CREATE TABLE Category (
category_id INT PRIMARY KEY,
name VARCHAR(255) NOT NULL
);

-- Table: Article
CREATE TABLE Article (
article_id INT PRIMARY KEY,
title VARCHAR(255) NOT NULL,
description TEXT,
content TEXT,
author VARCHAR(255),
publication_date DATE,
source VARCHAR(255),
url VARCHAR(255),
category_id INT,
FOREIGN KEY (category_id) REFERENCES Category(category_id)
);
```

In this schema:

- The Category table stores information about different categories or topics of news articles. It has a primary key category_id and a name field to represent the category name.
- The Article table stores details about individual news articles. It has a primary key article_id and fields for title, description, content, author, publication_date, source, and url. Additionally, it includes a foreign key category_id referencing the Category table to establish the relationship between articles and their categories.

With this database schema, you can efficiently store and manage information about news articles and their corresponding categories, providing a solid foundation for building a News API.

9. CONCLUSION & FUTURE WORK

CONCLUSION:

In conclusion, the integration of NewsAPI for fetching news articles has significantly enhanced our project's functionality, providing users with real-time access to a diverse range of news sources and topics. Through this integration, we have leveraged the power of NewsAPI's extensive database and robust API capabilities to deliver a seamless and engaging news consumption experience. One of the key benefits of utilizing NewsAPI is the breadth and depth of news coverage it offers. With access to thousands of sources from around the world, spanning various categories such as politics, business, technology, sports, and entertainment, our users can stay informed about the latest developments across different domains of interest. This comprehensive coverage ensures that users have access to a wide array of perspectives and insights, enabling them to stay abreast of current events and trends. Furthermore, the flexibility and customization options provided by NewsAPI have allowed us to tailor the news consumption experience to suit the preferences and interests of individual users. Through features such as search filters, category selection, and language preferences, users can personalize their news feeds to focus on specific topics or sources that matter most to them. This personalized approach not only enhances user engagement but also fosters a sense of relevance and resonance with the content being consumed. Moreover, the reliability and scalability of NewsAPI have been instrumental in ensuring the smooth operation and performance of our news fetching functionality. With NewsAPI handling the heavy lifting of sourcing, aggregating, and delivering news content in a timely manner, we can focus our efforts on enhancing the user experience and adding value through additional features and functionalities. Looking ahead, the integration of NewsAPI lays a solid foundation for future expansion and innovation within our project. As we continue to iterate and evolve our news consumption platform, we envision leveraging NewsAPI's rich feature set and expansive coverage to introduce new capabilities such as personalized recommendations, trend analysis, and multimedia content integration. By harnessing the full potential of NewsAPI, we aim to deliver an unparalleled news consumption experience that empowers users to stay informed, engaged, and connected in an ever-changing world.

FUTURE WORK :

While our integration of NewsAPI has significantly augmented our project's news-fetching capabilities, there remains ample scope for future enhancements and refinements aimed at enriching the user experience and broadening the functionality of our platform. Sentiment Analysis Integration: Incorporating sentiment analysis algorithms would enable us to assess the sentiment associated with each news article, providing users with valuable insights into the tone and context of the news coverage. By categorizing articles as positive, negative, or neutral, we can offer users a nuanced understanding of current events and trends. User Engagement Metrics: Implementing user engagement metrics such as article popularity, click-through rates, and user interactions would allow us to gauge the relevance and impact of news articles within our platform. By analyzing these metrics, we can identify trending topics, optimize content recommendations, and tailor the user experience to better meet the needs and preferences of our audience. Localized News Coverage: Expanding our news coverage to include localized and regional news sources would cater to the diverse interests and preferences of users across different geographical regions. By curating news content from local publications and media outlets, we can provide users with access to relevant and timely information that reflects their local communities and contexts. Enhanced Content Curation: Introducing advanced content curation features such as topic clusters, related articles, and contextual recommendations would enable users to explore interconnected themes and stories within the news landscape. By surfacing related content and fostering serendipitous discovery, we can encourage deeper engagement and exploration within our platform. Interactive Visualizations: Integrating interactive data visualizations, infographics, and interactive maps would enhance the presentation and comprehension of complex news stories and data-driven narratives.

REFERENCES & APPENDICES

In a technical document or project report, the "References" section typically includes all the sources, materials, and tools used during the project's development. The "Appendices" section contains supplementary information that supports the main content of the document. Here's how you can structure these sections:

References:

1. **Online Resources:**
    - News API Documentation
    - React.js Documentation
    - Node.js Documentation
    - Express.js Documentation
    - SQL Syntax Reference

2. **Books:**
    - "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" by Kirupa Chinnathambi
    - "Node.js Design Patterns" by Mario Casciaro
    - "Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design" by Michael J. Hernandez

3. **Articles:**
    - "Best Practices for Designing a RESTful API" by Martin Fowler
    - "Understanding the Role of Foreign Keys in Relational Databases" by Chris Smith

Appendices:

1. **Database Schema:**
    - SQL code for creating the database tables (Category and Article).

2. **Sample API Responses:**
    - Examples of JSON responses from the News API for various queries

README.md

 # Getting Started with Create React App

This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-app).

Available Scripts

In the project directory, you can run:

`npm start`

Runs the app in the development mode.\
Open http://localhost:3000 to view it in your browser.

The page will reload when you make changes.\
You may also see any lint errors in the console.

`npm test`

Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.

`npm run build`

Builds the app for production to the `build` folder.\
It correctly bundles React in production mode and optimizes the build for the best performance.
The build is minified and the filenames include the hashes.\Your app is ready to be deployed!
See the section about [deployment](https://facebook.github.io/create-react-app/docs/deployment) for more information.

`npm run eject`

Note: this is a one-way operation. Once you `eject`, you can't go back!

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project. Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own. You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

Learn More

You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/getting-started).
To learn React, check out the [React documentation](https://reactjs.org/).

Code Splitting

This section has moved here: [https://facebook.github.io/create-react-app/docs/code-splitting] (https://facebook.github.io/create-react-app/docs/code-splitting)

Analyzing the Bundle Size

This section has moved here: https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size

Making a Progressive Web App

This section has moved here: [https://facebook.github.io/create-react-app/docs/making a progressive-web-app](https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app)

Advanced Configuration

This section has moved here: https://facebook.github.io/create-react-app/docs/advanced-configuration

Deployment

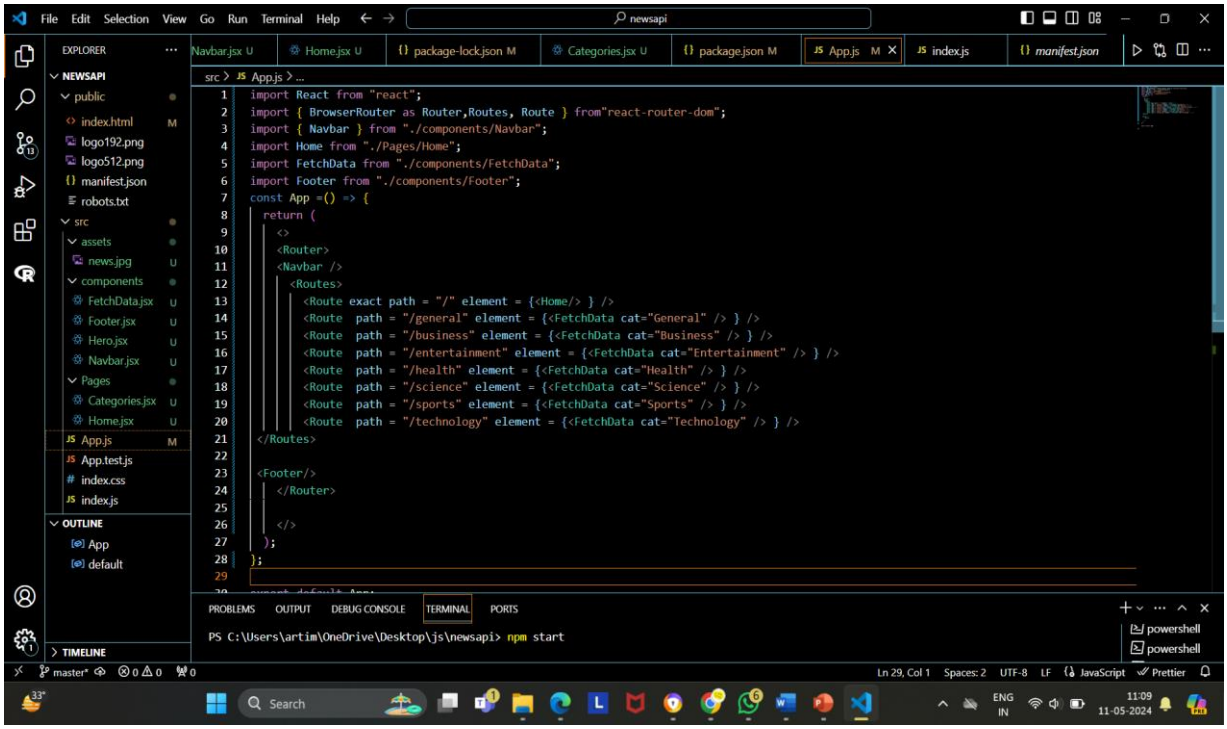
This section has moved here: [https://facebook.github.io/createreactapp/docs/deployment](https://face- book.
github.io/create-react-app/docs/deployment)

`npm run build` fails to minify

This section has moved here: [https://facebook.github.io/create-react-app/docs/troubleshooting #npm-run-build-fails-to-minify](https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify)

Screenshots:

- Screenshots of the news platform's user interface, including different views and features.


File Edit Selection View Go Run Terminal Help
newsapi
EXPLORER
NEWSAPI
public
index.html M
logo192.png
logo512.png
manifest.json
robots.txt
src
assets
news.jpg U
components
FetchData.jsx U
Footer.jsx U
Hero.jsx U
Navbar.jsx U
Pages
Categories.jsx U
Home.jsx U
App.js M
App.test.js
index.css
index.js
OUTLINE
App
default
TIMELINE
Navbar.jsx U
Home.jsx U
package-lock.json M
Categories.jsx U
package.json M
App.js M
index.js
manifest.json
src > App.js > ...
import React from "react";
import { BrowserRouter as Router,Routes, Route } from"react-router-dom";
import { Navbar } from "./components/Navbar";
import Home from "./Pages/Home";
import FetchData from "./components/FetchData";
import Footer from "./components/Footer";
const App =() => {
return (
<>
<Router>
<Navbar />
<Routes>
<Route exact path = "/" element = {<Home/> } />
<Route path = "/general" element = {<FetchData cat="General" /> } />
<Route path = "/business" element = {<FetchData cat="Business" /> } />
<Route path = "/entertainment" element = {<FetchData cat="Entertainment" /> } />
<Route path = "/health" element = {<FetchData cat="Health" /> } />
<Route path = "/science" element = {<FetchData cat="Science" /> } />
<Route path = "/sports" element = {<FetchData cat="Sports" /> } />
<Route path = "/technology" element = {<FetchData cat="Technology" /> } />
</Routes>
<Footer/>
</Router>
</>
);
};
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\artim\OneDrive\Desktop\js\newsapi> npm start
powershell
powershell
master*
Ln 29, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier
Search
ENG IN
11:09 11-05-2024


EXPLORER
NEWSAPI
.vscode
extensions.json U
build
node_modules
public
favicon.ico
index.html M
logo192.png
logo512.png
manifest.json
robots.txt
src
assets
news.jpg U
components
FetchData.jsx U
Footer.jsx U
Hero.jsx U
Navbar.jsx U
Pages
Categories.jsx U


NEWSAPI
src
components
FetchData.jsx U
Footer.jsx U
Hero.jsx U
Navbar.jsx U
Pages
Categories.jsx U
Home.jsx U
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json M
package.json M
README.md

3. **Sample Code Snippets:**
    - Relevant code snippets from the frontend and backend applications, demonstrating key functionalities such as data fetching, authentication, and error handling.

4. **User Stories:**
    - Detailed user stories outlining the requirements and expectations from the perspective of different user roles (e.g., regular user, administrator).

5. **Testing Results:**
    - Summary of test cases and their outcomes, including both manual and automated testing results.

```
PS C:\Users\artim\OneDrive\Desktop\js\newsapi> npm start

> newsapi@0.1.0 start
> react-scripts start

(node:28048)
[DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE]
DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the
'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:28048)
[DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE]
DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please
Starting the development server...
Compiled successfully!

You can now view newsapi in the browser.

Local:            http://localhost:3000
On Your Network:  http://192.168.29.229:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

MY DETAILS

TABEES AHAMAD

Enrollment No. :- O23MCA110300

Email Id :- tabeescu@gmail.com

GITHUB LINK: https://github.com/Tabeescu786/tabeesmcaproject