Smart-Feed: Personalized News Aggregator with Intelligent Recommendations

By

Fazal Haq
01-134202-024

Sanan Ahmad
01-134211-083

Bachelor of Science in Computer Science

Supervisor
Muhammad Ahtehsham Noor

Department of Computer Science
Bahria University, Islamabad

2024

Abstract

The Smart Feed News Aggregator is a web-based application designed to deliver personalized news recommendations by aggregating articles from various reliable sources. In an era where users are overwhelmed with information, this project aims to enhance user engagement by providing tailored content that aligns with individual preferences and historical interactions.

The system utilizes advanced machine learning techniques, specifically Neural Collaborative Filtering (NCF) and content-based filtering, to create a hybrid recommendation model. NCF analyzes user-item interactions to predict click-through rates, while content-based filtering employs BERT to generate embeddings from article titles and abstracts, capturing their semantic meaning. This allows the system to recommend similar articles based on users' past behavior.

With a user-friendly interface, the application enables users to create profiles, manage preferences, and access both general and personalized news feeds. By continuously adapting recommendations based on user interactions, the Smart Feed News Aggregator aims to improve the overall news consumption experience, ensuring that users receive relevant, timely, and engaging content. This project highlights the significance of personalized news delivery in fostering user satisfaction and retention in a digital landscape.

Acknowledgements

We would like to express our sincere gratitude to the faculty and staff at Bahria University for their unwavering support and guidance throughout our academic journey. The knowledge and skills we acquired during our coursework have been invaluable in the development of the Smart Feed News Aggregator project.

We extend our heartfelt appreciation to our supervisor, Muhammad Ahtehsham Noor, for his expert guidance and insightful feedback. His mentorship played a critical role in shaping the direction of our project and enhancing our understanding of the underlying concepts and methodologies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Background/Overview

The process of staying informed about current events can be overwhelming and challenging for individuals due to a multitude of factors, including the vast array of news sources, diverse topics, varying quality of reporting, and personal interests. However, in Pakistan, the availability of expert news aggregation tools or resources to assist users in finding relevant news articles is limited.

To address this pressing need, we introduce Smart Feed News Aggregator, a web-based news recommendation system. Leveraging data collected from user interactions and news articles, Smart Feed aims to provide personalized news recommendations, guiding users to make well-informed decisions about the content they consume.

The primary goal of this project is to create a user-friendly system that assists users in navigating the complex landscape of news articles. By employing machine learning techniques, Smart Feed will analyze the gathered data, discern patterns and correlations between user preferences and news content, and generate accurate, tailored recommendations.

Additionally, Smart Feed offers supplementary features to enhance the user experience. These include a personalized news feed, historical data of past interactions and preferences, and user profile management. These features contribute to a comprehensive news consumption solution designed to simplify the decision-making process regarding the information users choose to engage with.

1.2 Problem Description

Staying updated with the latest news can be daunting for users due to the vast array of news sources, each offering different articles and perspectives. This diversity makes it challenging to make informed reading choices, and users often lack access to expert guidance on the relevance of various articles.

The absence of a reliable news recommendation platform exacerbates this issue. Without proper guidance, users may miss relevant content or read articles that do not align with their interests. Traditional methods of browsing and researching news can be timeconsuming and overwhelming.

To address these challenges, we propose the Smart Feed News Aggregator, a system that leverages machine learning algorithms to provide personalized news recommendations. By analyzing user preferences, article topics, and historical interactions, the aggregator generates tailored suggestions that cater to individual interests.

This solution allows users to effortlessly access curated news articles, saving time and effort in filtering through content. Additionally, features like article comparisons and a history of past recommendations enhance the user experience, enabling informed decisions about the information they choose to consume.

1.3 Project Objectives

This project envisions the development of a comprehensive web-based system to address the multifaceted challenges users face when seeking relevant news articles that align with their interests and preferences. The primary goal is to create a high-quality, user-centric platform that empowers users to make informed decisions and significantly improves their news consumption experience. The specific objectives of the project are as follows:

- **Develop a web-based news recommendation system:** The Smart Feed News Aggregator aims to design and develop a user-friendly web-based news recommendation system. This platform will allow users to input their preferences and receive personalized recommendations for suitable news articles. Additionally, the system will offer supplementary features such as the ability to compare articles, review a history of past recommendations, and manage user profiles.

- **Collect and process large datasets:** To ensure the accuracy and comprehensiveness of recommendations, the project will gather extensive datasets containing information on news articles, topics, and sources from reliable news APIs. These datasets will be meticulously processed and organized to feed into the news aggregator system.

- **Apply machine learning algorithms:** Machine learning algorithms, including Neural Collaborative Filtering and content-based filtering, will be employed in the news recommendation model. By utilizing historical data and user preferences, the model will learn patterns and correlations to provide accurate and relevant news suggestions.

- **Develop a user-friendly web interface:** The project will create an intuitive and user-friendly web interface for the news aggregator. This interface will enable users to easily input their preferences, view recommended articles, and filter recommendations based on various criteria such as category, recency, and relevance.

- **Ensure scalability and performance:** The news recommendation system will be designed to handle large volumes of user traffic while maintaining high performance and reliability. Scalability considerations will be taken into account to accommodate increasing user demand without compromising system efficiency.

.

1.4 Project Scope

The Project Scope encompasses the following key elements:

- **Development of a Web-based News Aggregator:** The primary focus of this project is to design, develop, and deploy a web-based news aggregator. The system will allow users to input their preferences and receive personalized news recommendations based on those preferences.
- **Data Collection and Processing:** Data on news articles, topics, sources, and user interactions will be collected from various online news APIs. The collected data will undergo thorough cleaning and preprocessing to ensure data integrity, consistency, and reliability.
- **Feature Identification and Engineering:** Relevant features for news recommendations, including the ability to compare articles, review past recommendations, and manage user profiles, will be identified and incorporated into the Smart Feed News Aggregator. Additionally, new features will be created as necessary to enhance the accuracy and relevance of the recommendations.
- **Machine Learning Model Development:** Given the subjective nature of user preferences and the dynamic nature of news content, the machine learning model will utilize techniques such as Neural Collaborative Filtering and content-based filtering. These techniques will be applied to identify patterns and correlations in user preferences, allowing the model to generate personalized news recommendations.
- **Continuous Improvement and User Feedback:** User feedback will be collected through the analysis of user choices regarding the articles they read. This data will be utilized to refine and improve the accuracy and relevance of the recommendation system over time, ensuring that the model adapts to user preferences.
- **Limitations:** The system's recommendations are based on the data it has been trained on. If the data is not comprehensive or up-to-date, the recommendations may not be accurate. The system's performance may be affected by the quality of user input; if the user input is not accurate or complete, the system may struggle to generate precise recommendations.
- **Target Audience:** The target audience for this project includes general news readers, professionals seeking specific information, and any users who want to stay informed about current events. The system is designed to be user-friendly and accessible, catering to a wide range of users.

1.5 Software Model

For the development of the Smart Feed: Personalized News Aggregator, an Agile software development model has been adopted. The aggregator leverages Agile methodologies, incorporating key practices such as iterative development, user stories, cross-functional teams, continuous integration, and frequent stakeholder engagement. This approach seamlessly aligns with the dynamic nature of the project, ensuring user-centric development, fostering continuous improvement, and maintaining flexibility throughout the project's lifecycle.

These practices collectively empower the Smart Feed: Personalized News Aggregator to provide highly personalized news recommendations, gather valuable user feedback early in the development process through analysis of user interactions, and enhance the overall user experience, making it a cutting-edge and adaptable system for users.

1.6 Summary

The Smart Feed: Personalized News Aggregator is dedicated to creating a comprehensive web-based platform that harnesses machine learning algorithms to deliver precise and tailored news recommendations based on user preferences. Beyond this primary goal, the aggregator provides supplementary features such as the ability to compare different articles, review a history of past recommendations, and manage user profiles.

The project's core objectives include collecting and processing data related to news articles, topics, sources, and user interactions, with an emphasis on identifying significant features for accurate recommendations. The Smart Feed will develop and evaluate a machine learning model using established performance metrics. Upon completion, it will be deployed as a user-friendly web application, enabling users to input their preferences and receive personalized news suggestions.

Continual collection and analysis of user interaction data will drive ongoing improvements, ensuring that recommendations remain accurate and relevant. The Smart Feed acknowledges certain limitations, such as constraints in data availability, potential biases in data and recommendations, limitations in user interactions, implementation constraints, and external factors. Ultimately, the goal of the Smart Feed is to empower users with a valuable tool for making informed decisions about the news they consume, enhancing their overall news reading experience.

Chapter 2

Literature Review

2.1 Introduction

In the modern digital age, the sheer volume of news content available online has led to a significant problem of information overload. With countless news sources and articles published every minute, users often find it difficult to sift through irrelevant information to access content that aligns with their interests and preferences. This issue has made personalized news recommendation systems a crucial tool for enhancing user experience, enabling users to receive tailored news feeds that cater to their specific needs and interests.

SmartFeed is a personalized news recommendation platform designed to solve the issue of information overload by utilizing machine learning techniques, specifically a hybrid model combining Neural Collaborative Filtering (NCF) and content-based filtering. The system aims to deliver a seamless user experience by recommending news articles that are both relevant and diverse, based on individual user preferences, engagement history, and content attributes. Unlike traditional news delivery platforms, SmartFeed ensures that each user's news feed evolves dynamically with their changing preferences, providing timely and engaging news content.

This chapter explores the existing research on news recommendation systems, highlighting the development and effectiveness of various recommendation techniques such as collaborative filtering, content-based filtering, and hybrid models. It also examines the challenges faced by traditional recommendation systems and how modern approaches, such as Neural Collaborative Filtering, address these limitations by leveraging deep learning. Through this literature review, we aim to position SmartFeed within the broader landscape of personalized news recommendation systems and demonstrate how it offers a more efficient and user-centric solution.

2.2 Machine Learning Techniques

In SmartFeed, we leverage advanced machine learning techniques to provide highly personalized news recommendations. Our approach uses a combination of *Neural Collaborative Filtering (NCF)* and *content-based filtering*, creating a hybrid system that balances user preferences with content relevance.

2.2.1 Neural Collaborative Filtering (NCF)

Neural Collaborative Filtering is an extension of traditional collaborative filtering methods, which predict user preferences based on historical interactions between users and items. Unlike matrix factorization techniques commonly used in collaborative filtering, NCF employs deep learning to model these user-item interactions with greater complexity. Specifically, NCF replaces matrix factorization with a neural network architecture, allowing it to capture non-linear relationships between users and content.

NCF works by learning latent representations of users and news articles through embeddings. These representations are then fed into a multi-layer perceptron (MLP), which computes a score indicating the likelihood that a user will interact with a given news article. By modeling user-item interactions as a function of learned latent features, NCF can generate highly personalized recommendations.

One of the main advantages of NCF is its ability to handle dynamic and evolving user preferences, as it can learn from a wide variety of interaction data, including clicks, views, and reading time. Additionally, NCF has been shown to outperform traditional collaborative filtering techniques in various domains, including movie and product recommendations. However, NCF requires large amounts of data to effectively learn the underlying patterns, and its deep learning architecture demands significant computational resources. Nonetheless, its flexibility and accuracy make it a suitable choice for SmartFeed, where user preferences are constantly shifting based on the ever-changing nature of news content.

2.2.2 Content-Based Filtering

In contrast to collaborative filtering, which relies on user interactions, content-based filtering recommends items based on the characteristics of the items themselves. In the context of SmartFeed, this means analyzing the attributes of news articles—such as keywords, topics, categories, and metadata—to match them with a user's interests and preferences.

Content-based filtering builds a profile of each user by analyzing the articles they have interacted with (e.g., articles they have read or liked). Each article is represented by a set of features, and the system recommends new articles that are similar to those the user has previously engaged with. This technique is especially useful when there is limited user interaction data, as it focuses on the content's properties rather than relying on user-item interaction history.

However, one limitation of content-based filtering is the risk of *over-specialization*, where the system only recommends items that are very similar to the user's past preferences, limiting the diversity of content. To mitigate this issue, SmartFeed combines content-based filtering with NCF to ensure that users are exposed to a broader range of articles, maintaining both relevance and diversity in the recommendations.

2.2.3 Hybrid Recommendation Systems

The hybrid approach employed by SmartFeed leverages the strengths of both NCF and content-based filtering, addressing the limitations of each. Hybrid systems have become increasingly popular in recommendation engines, as they can balance the personalization power of collaborative filtering with the richness of content-based techniques.

In SmartFeed, the NCF model is responsible for generating initial recommendations based on user-item interactions, while content-based filtering is used to fine-tune these recommendations by matching them with the attributes of the news articles. This hybrid method allows SmartFeed to provide personalized and diverse recommendations, even for new users who may not have extensive interaction history (the *cold-start problem*).

Moreover, hybrid systems can exploit multiple sources of information, including user behavior, item features, and contextual data. This enables a more holistic recommendation process, as the system can account for a variety of factors influencing user preferences. By combining collaborative and content-based approaches, SmartFeed ensures that the recommendations not only reflect the user's current interests but also expose them to new and diverse content that might expand their horizons.

2.2.4 Conclusion

By integrating *Neural Collaborative Filtering* with *content-based filtering*, SmartFeed delivers a highly personalized and dynamic news recommendation experience. The hybrid approach ensures that users receive relevant content based on their preferences while also being introduced to diverse and engaging articles that align with their evolving interests. This combination of techniques allows SmartFeed to provide a superior user experience, particularly in the fast-paced and ever-changing landscape of online news.

2.3 Related Work

2.3.1 Previous Research and Studies

The field of **news recommendation systems** has grown significantly in recent years due to the overwhelming amount of information users encounter daily. The main objective of these systems is to deliver personalized news articles that cater to individual preferences, improving both user engagement and satisfaction. Early news recommendation systems primarily used *Collaborative Filtering (CF)* techniques, similar to those used in product recommendation systems. These methods rely on user-item interaction data, such as clicks or reading history, to suggest articles based on the preferences of similar users.

Content-Based Filtering (CBF) emerged as an alternative to CF, where recommendations are made by analyzing the attributes of news articles, such as keywords, topics, or publication dates, and matching them with user interests. Content-based filtering has proven to be particularly useful in news recommendation because news articles often contain rich metadata that can be easily leveraged to generate personalized recommendations. However, CBF suffers from limitations like *over-specialization*, where users are recommended content too similar to what they have previously consumed, limiting content diversity.

To address the shortcomings of both CF and CBF, **hybrid recommendation systems** have been developed. These systems combine the strengths of collaborative and content-based approaches to deliver more accurate and diverse recommendations. Hybrid models, like those employed by platforms such as *Google News* and *Flipboard*, use a combination of user interaction data and article metadata to personalize news feeds. Research in this area, such as the work done by *He et al. (2017)* on *Neural Collaborative Filtering (NCF)*, highlights the effectiveness of hybrid models in improving recommendation accuracy by learning complex user-item interactions through deep learning techniques [see more].

Recent advances have seen the integration of deep learning into recommendation systems, with techniques like NCF and *Recurrent Neural Networks (RNNs)* being employed to model the temporal dynamics of user preferences. These methods are capable of capturing more nuanced, non-linear relationships between users and news articles, leading to more personalized and engaging news feeds. *Neural models* offer an improvement over traditional CF by learning from implicit signals, such as time spent reading articles or scrolling behavior, allowing for more dynamic recommendations that adapt to user behavior in real-time [see more].

2.3.2 Overview of News Recommendation Platforms

Several news platforms have successfully integrated machine learning techniques to provide personalized news feeds to users, each using a combination of collaborative filtering, content-based methods, and hybrid models to optimize recommendations.

- **Google News** employs a hybrid approach, combining user preferences, browsing behavior, and trending news data to deliver personalized news articles. The platform tracks user interactions with specific topics and recommends articles that are both relevant to the user's interests and popular among a broader audience. Google News also uses real-time data to ensure users are presented with up-to-date information that aligns with their preferences.

- **Flipboard** allows users to curate their own news magazines by selecting topics of interest. The platform leverages this explicit user input along with implicit behavior data (e.g., articles clicked, time spent reading) to personalize content. Flipboard's recommendation system uses a mix of collaborative filtering and content-based methods to ensure users receive a balance of articles from their chosen categories as well as new, diverse content to broaden their scope.

- **SmartNews** focuses on delivering a mix of trending and personalized news, using a proprietary algorithm to identify articles that are both relevant to individual users and widely read by others. The platform's hybrid recommendation system adapts to the user's reading patterns over time, continuously refining recommendations based on the evolving preferences of the user.

These car recommendation websites, while diverse in their approaches, share common goals: to simplify the car buying experience and to provide users with personalized suggestions. Their success is largely due to their ability to combine user behavior, item features, and contextual data to offer comprehensive and relevant recommendations. While the field has advanced significantly, there are still challenges, such as user engagement and accuracy, that continue to drive research and innovation in this space.

SmartFeed aims to build upon these existing systems by integrating *Neural Collaborative Filtering* with content-based filtering, offering a more refined and adaptive recommendation engine. Through the use of deep learning, SmartFeed's system can continuously learn from user behavior, providing both highly relevant and diverse news articles to each individual. This approach ensures that users are not only presented with content that aligns with their immediate preferences but are also exposed to new topics that broaden their perspectives.

Table 2.1: Comparison of News Recommendation Platforms

Platform	**Recommendation Approach**	**Personalization**	**Features**
Smart Feed	Hybrid: Neural Collaborative Filtering (NCF) + Content-Based Filtering	Highly personalized, combining user preferences, interaction history, and content analysis.	Real-time personalized news feed, diverse content exposure, hybrid model for evolving user preferences.
Google News	Hybrid: Content-Based Filtering + Collaborative Filtering	Personalized news feed based on browsing history, location, and real-time trending topics.	Aggregates news from various sources, uses real-time data, focus on trending news and user preferences.
Flipboard	Content-Based Filtering	Allows users to select topics of interest and curates news based on explicit preferences and reading behavior.	Custom magazines, user-curated news, combines content matching with user-selected categories.
SmartNews	Content-Based Filtering + Trending Topic Analysis	Focuses on delivering both trending and personalized news based on reading habits and real-time topic popularity.	Delivers breaking news, user-driven personalization, real-time analysis of news popularity.
Feedly	Content-Based Filtering + Keyword Matching	Curates news based on keyword preferences, RSS feeds, and explicit user input.	Personalized news feed, keyword-based filtering, wide range of sources including blogs and research papers.

2.4 Critical Analysis of SmartFeed News Aggregator

The SmartFeed platform seeks to address the issue of information overload by offering personalized news feeds through a hybrid recommendation system that combines *Neural Collaborative Filtering (NCF)* and *content-based filtering*. This approach enables

SmartFeed to analyze both user-item interactions and the characteristics of news articles, providing a more comprehensive recommendation process. By leveraging *NCF*, SmartFeed effectively adapts to evolving user preferences, while content-based filtering ensures relevance even for new users, helping to mitigate the *cold-start problem*.

2.4.1 Strengths

SmartFeed's hybrid approach stands out for its ability to dynamically adjust to user behavior while offering timely and relevant content. The system's flexibility, especially in using both interaction data and article metadata, helps ensure accurate recommendations for users with varying engagement levels. Furthermore, by continuously updating its recommendations in real-time, SmartFeed ensures that users receive the latest content, making the platform highly responsive.

2.4.2 Limitations

Despite its strengths, SmartFeed's reliance on *NCF* can lead to challenges when user data is sparse, potentially reducing recommendation accuracy. Additionally, the platform faces risks of *over-specialization*, where recommendations may become too narrow, limiting content diversity. Although the hybrid model mitigates this, striking a balance between relevance and variety remains a concern.

2.4.3 Areas for Improvement

To enhance SmartFeed, integrating *contextual data* (e.g., location, time) and adding *user feedback mechanisms* could improve personalization and recommendation quality. Furthermore, incorporating *trending topics* and analyzing broader social engagement could offer users a more diverse and well-rounded news experience, especially in scenarios with limited user interaction data.

2.4.4 Conclusion

SmartFeed presents an innovative solution to personalized news delivery through its hybrid recommendation model. However, it must address challenges related to data dependency and content diversity to ensure scalability and sustained user satisfaction. Expanding features to include context and feedback mechanisms will strengthen the platform's ability to deliver a superior news consumption experience.

Chapter 3

Requirement Specification

3.1 Existing System

SmartFeed: Personalized News Aggregator aims to provide a unique and personalized news recommendation experience to users, distinguishing itself from existing systems such as Google News, Flipboard, Feedly, and Apple News. While these platforms offer various functionalities related to news aggregation and recommendations, they fall short in certain aspects that Smart Feed seeks to address.

Google News and Flipboard, for instance, primarily focus on curating news articles based on general topics or popular trends, providing filters for users to explore different categories. However, these platforms lack a truly comprehensive recommendation system that accounts for individual user preferences, reading habits, and specific content interests.

Feedly and Apple News offer some level of personalized news recommendations. However, their approach is largely driven by algorithmic filtering based on broad content categories and popular sources, often overlooking the importance of deeply personalized user experiences. These systems may suggest articles based on general market trends or user subscriptions but fail to consider the unique needs, reading history, and preferences of individual users.

3.2 Proposed System

3.2 Proposed System In contrast to existing platforms, Smart Feed: Personalized News Aggregator is designed to bridge the gaps in current news aggregation services by combining their advantages with a user-centric approach. It will incorporate a comprehensive set of criteria, including user preferences, reading history, topic interests, and content types. Through advanced machine learning algorithms and personalization techniques, our system will provide accurate and tailored news recommendations that align with each user's specific interests and reading habits.

Furthermore, this system prioritizes the user experience by considering factors such as article relevance, readability, timeliness, and engagement. By taking these aspects into account, Smart Feed strives to offer recommendations that not only match the content preferences but also enhance the overall news consumption experience.

Overall, Smart Feed goes beyond the limitations of existing platforms by providing a personalized, user-centric, and comprehensive news recommendation experience. By combining advanced algorithms, user preferences, and a focus on reader satisfaction, our system delivers recommendations that align with the unique needs and interests of each individual user. The inclusion of features like article comparisons, a history of past interactions, and user profile management further enhances the user experience, making Smart Feed a versatile and user-friendly platform.

3.3 Requirement Specifications

3.3.1 Functional Requirements

- **User Authentication:**
  - Users must be able to sign up, log in, and log out using a secure authentication system, such as JWT (JSON Web Tokens).
  - The system should securely store user credentials and profiles using MongoDB.
- **News Aggregation:**
  - The system will collect news articles from various online sources using APIs (e.g., Microsoft News Dataset, NewsAPI).
  - Users can browse both general and personalized news feeds based on their preferences.
- **Personalized News Recommendations:**
  - The platform should recommend news articles tailored to user preferences and interaction history using the machine learning recommendation system (NCF + content-based filtering).
  - Recommendations should update dynamically based on user behavior, such as reading history, clicks, and time spent on articles.
- **User Interaction Tracking:**
  - Track user interactions with news articles (e.g., clicks, read time, likes) and use this data to refine recommendations.
  - Store user interaction data for future training and improving machine learning models.
- **User Feedback Loop:**
  - The system collects implicit feedback (clicks, time spent) and stores it to improve future recommendations without explicitly asking users for feedback.
- **For You Section:**
  - A dedicated "For You" page will display personalized news recommendations for each user based on their interests and past interactions.

3.3.2 Non-Functional Requirements

- **Performance:**
  - The system should handle a high volume of concurrent users and API requests while maintaining low latency in news fetching and recommendation delivery.
  - Machine learning inference should be optimized to ensure recommendations are delivered within a reasonable timeframe.
- **Scalability:**
  - The system architecture should allow for scaling in response to increased user traffic and data volumes, particularly for real-time recommendations.
  - Use of cloud services (e.g., AWS, GCP) to ensure the platform can scale horizontally as the user base grows.
- **Security:**
  - Ensure secure handling of user data and interactions, particularly for authentication and personal preferences.
  - Implement encryption for sensitive data such as user credentials and API keys.
- **Data Privacy:**
  - Ensure compliance with data privacy laws (e.g., GDPR) by giving users control over their data and how it's used in the recommendation process.
- **Usability:**
  - The user interface must be intuitive, with minimal learning required for users to interact with the platform.
  - Provide consistent and visually appealing design across different devices.
- **Reliability:**
  - The system should be designed with fault tolerance, ensuring minimal downtime.
  - Implement automated backups for user data and interactions, especially for the recommendation system.
- **Maintainability:**
  - Code should be modular and well-documented to facilitate easy updates, bug fixes, and addition of new features.
  - Clear versioning and deployment strategies must be followed to ensure smooth rollouts of updates.
- **Extensibility:**
  - The system should be flexible enough to add new recommendation algorithms or integrate with other news APIs in the future.

3.4 Use Cases

The use cases for the Smart Feed News Aggregator are designed to illustrate the interaction between users and the system. Below are the main actors and their corresponding use cases.

3.4.1 Actors

- **User:** The primary actor who interacts with the news aggregator.
- **System:** The Smart Feed News Aggregator that provides news recommendations.

3.4.2 Use Case Diagram

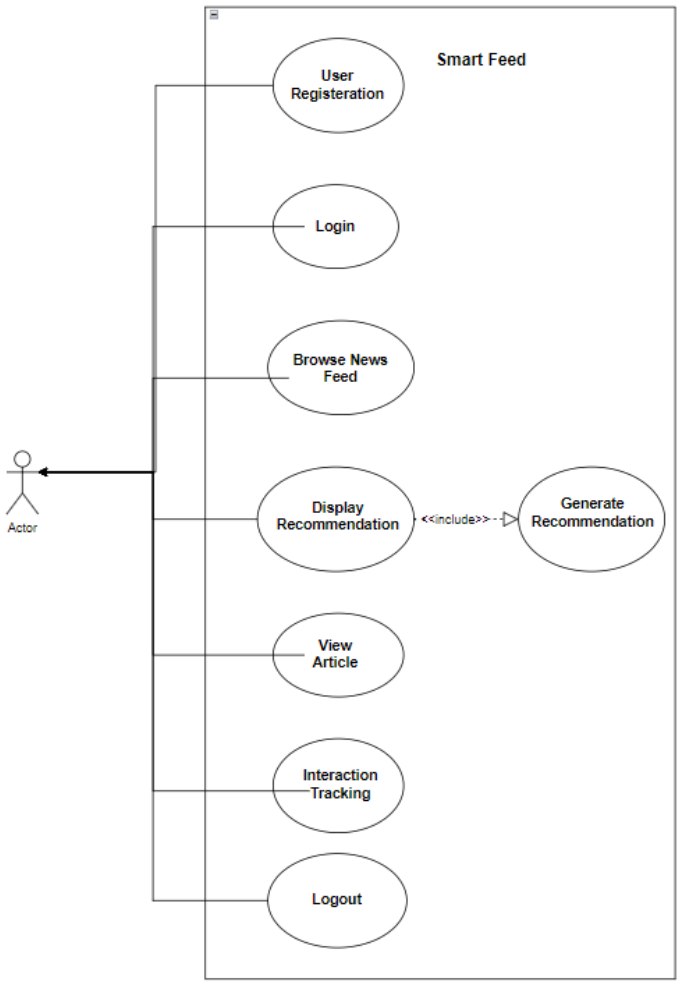


Figure 3.1: Use Case Diagram for Smart Feed News Aggregator

3.4.3 Use Case Tables

Table 3.1: Use Case 001 - Register User

Use Case ID	UC001
Use Case Name	Register User
Description	User registers a new account.
Actors	User
Data	Name, Email, Password
Trigger/Stimulus	User accesses the web app.
Pre-Condition	The user is on the registration page.
Assumptions	The web app is available and accessible.
Normal Flow of Events	1. The user enters the required data. 2. The user clicks the "Register" button. 3. The system registers the user.
Alternate Flow of Events	None
Main Success Scenario	User successfully registered.
Post-Condition	The user successfully registers, and the credentials are now eligible for logging in.

Table 3.2: Use Case 002 - Login User

Use Case ID	UC002
Use Case Name	Login User
Description	The user enters email and password to log in.
Actors	User
Data	Email, Password
Trigger/Stimulus	User accesses the web app.
Pre-Condition	The user is on the login page and has an account.
Assumptions	User has registered an account.
Normal Flow of Events	1. The user enters email and password. 2. The system validates credentials. 3. The system logs the user in.
Alternate Flow of Events	Invalid credentials result in a system prompt for re-entry.
Main Success Scenario	Successfully logged in.
Post-Condition	User can access the dashboard.

Table 3.3: Use Case 003 - View General News/Browse News Feed

Use Case ID	UC003
Use Case Name	View General News/Browse News Feed
Description	User views general news articles.
Actors	Any user (guest or logged-in)
Data	N/A
Trigger/Stimulus	User accesses the homepage or category page.
Pre-Condition	The user is browsing the website.
Assumptions	The external API is responsive, and the user has internet access.
Normal Flow of Events	1. The user navigates to the homepage. 2. The system fetches general news articles. 3. The articles are displayed in a news feed.
Alternate Flow of Events	None
Main Success Scenario	News articles are displayed.
Post-Condition	The user can read articles of interest.

Table 3.4: Use Case 004 - Personalized News Recommendations

Use Case ID	UC004
Use Case Name	Personalized News Recommendations
Description	User views personalized news recommendations.
Actors	Logged-in user
Data	User history, preferences, past interactions
Trigger/Stimulus	User accesses the "For You" page.
Pre-Condition	User has previous interaction history.
Assumptions	User is logged in and has interacted with content previously.
Normal Flow of Events	1. The user navigates to the "For You" page. 2. The system fetches user preferences. 3. Recommendations are displayed based on history.
Alternate Flow of Events	None
Main Success Scenario	Personalized news is displayed.
Post-Condition	The user can consume the recommended articles.

Table 3.5: Use Case 005 - Filter News by Country or Category

Use Case ID	UC005
Use Case Name	Filter News by Country or Category
Description	User filters news by selecting a country or category.
Actors	Any user
Data	Country code, category
Trigger/Stimulus	User selects a filter from the dropdown.
Pre-Condition	The filter options are displayed to the user.
Assumptions	The external API supports country/category filtering.
Normal Flow of Events	1. The user selects a country or category. 2. The system fetches filtered news. 3. The news feed is updated with relevant articles.
Alternate Flow of Events	None
Main Success Scenario	Filtered news is displayed.
Post-Condition	The user consumes filtered content.

Table 3.6: Use Case 006 - Track User Interactions and Preferences

Use Case ID	UC006
Use Case Name	Track User Interactions and Preferences
Description	System tracks user interactions and preferences.
Actors	Logged-in user
Data	Interaction history (clicks, time spent, categories viewed)
Trigger/Stimulus	User interacts with articles or categories.
Pre-Condition	The user is interacting with the system.
Assumptions	User is logged in and interacts with content.
Normal Flow of Events	1. User clicks on articles and interacts with categories. 2. The system logs the user's actions. 3. The user's profile is updated based on the interactions.
Alternate Flow of Events	None
Main Success Scenario	User preferences are updated for future recommendations.
Post-Condition	Interaction data is stored for future use.

Table 3.7: Use Case 007 - View Full Article (External Link)

Use Case ID	UC007
Use Case Name	View Full Article (External Link)
Description	User clicks on an article and is redirected to the external news source.
Actors	Any user
Data	Article link
Trigger/Stimulus	User clicks on an article in the news feed.
Pre-Condition	News articles are displayed in the feed with clickable links.
Assumptions	The external website is available and the user has internet access.
Normal Flow of Events	1. The user clicks on an article in the feed. 2. The system redirects the user to the external source of the article. 3. The user reads the article on the external website.
Alternate Flow of Events	External website is unavailable, and the system displays an error.
Main Success Scenario	The user successfully reads the article on the external website.
Post-Condition	The system logs the user's interaction with the article for future recommendations.

Table 3.8: Use Case 008 - Logout User

Use Case ID	UC003
Use Case Name	Logout User
Description	The user logs out of the web application.
Actors	User
Data	None
Trigger/Stimulus	User selects the logout option.
Pre-Condition	The user is logged in and on the dashboard.
Assumptions	User has an active session.
Normal Flow of Events	1. The user clicks on the logout button. 2. The system terminates the user session. 3. The system redirects the user to the login page.
Alternate Flow of Events	If the session has already expired, the system informs the user and redirects them to the login page.
Main Success Scenario	Successfully logged out.
Post-Condition	User is redirected to the login page and cannot access the dashboard without logging in again.

Chapter 4

Methodology

4.1 System Architecture

The system architecture of the **News Aggregator** project is based on a **client-server model**, which enables seamless interaction between users and the back-end system to provide real-time, personalized news recommendations. The architecture is designed to be scalable, modular, and flexible, integrating multiple layers that manage user inputs, process data, and generate tailored news content. The primary focus of the architecture is to enhance the user experience by leveraging machine learning algorithms for recommendation and efficient data handling techniques.

4.1.1 Client Layer

The client layer acts as the entry point for users to interact with the system. It includes **web browsers** across different devices such as desktops, laptops, tablets, and mobile phones. The user interface is designed to be responsive and adaptable, ensuring a consistent experience across devices. The client layer allows users to:

- **Browse and explore** general news articles from multiple sources.
- **View personalized news** based on their past interactions and preferences through the 'For You' page.
- **Filter news articles** by categories, regions, or search keywords to customize their news feed further.
- **Interact with articles** by liking, saving, or sharing them, which in turn influences the recommendation engine.

The client layer serves as the face of the system, providing an intuitive and engaging platform for users to stay updated on current events while offering recommendations tailored to their specific interests.

4.1.2 Presentation Layer

The presentation layer manages the interface between the client and the server, handling **user input**, **interaction management**, and **data presentation**. It consists of a **React.js** front-end framework, using components designed to deliver a dynamic and responsive user interface. This layer includes:

- **React Components**: Modular components such as `Header`, `AllNews`, `TopHeadlines`, and `CountryNews` manage different parts of the UI, offering structured navigation and news display.
- **State Management**: Tools such as `useState` and `useEffect` hooks are employed to manage local component state and to handle asynchronous data retrieval from the backend.
- **API Communication**: Uses `Axios` or `Fetch API` to send HTTP requests to the backend, enabling the retrieval of news articles, user data, and personalized recommendations in real-time.
- **Responsive Design**: With the use of `Tailwind CSS`, the interface adapts to different screen sizes and devices, ensuring that the user experience is consistent across platforms.

The presentation layer acts as a bridge between the user and the underlying system, ensuring that news data is presented in a user-friendly manner while keeping interactions smooth and responsive.

4.1.3 Application Layer

The application layer serves as the **core logic** of the system, handling the main functionalities such as **business rules**, **machine learning models**, and **API orchestration**. Built using **Node.js** and **Express.js**, this layer manages incoming requests from the client, processes them, and communicates with the data layer to retrieve relevant news or recommendation results. This layer includes several critical components:

- **Authentication**: Manages user registration, login, and JWT-based authentication to ensure secure access to personalized features.
- **News Retrieval**: Integrates external news sources like `NewsAPI` to fetch real-time news across various categories, regions, and languages.
- **Recommendation Engine**: A hybrid system combining **Neural Collaborative Filtering (NCF)** for collaborative filtering and **BERT-based content filtering**. This engine analyzes user interactions, article content, and preferences to generate personalized news recommendations.
- **Business Logic**: Processes incoming requests, validates data, and triggers appropriate workflows, such as fetching top headlines, delivering personalized news feeds, or handling user-specific actions (e.g., article likes or saves).

This layer is crucial to the functioning of the overall system, as it handles data processing and integrates the machine learning models that power the personalized news recommendations.

4.1.4 Data Layer

The data layer is the backbone of the system's **data storage and management**. This layer uses **MongoDB**, a NoSQL database, to store and retrieve data essential for the system's operation. The database stores a variety of data, including:

- **User Profiles**: Stores user information, including login credentials, preferences, interaction history, and personalization settings.
- **News Articles and Metadata**: Contains news articles fetched from external APIs, along with metadata such as categories, tags, publication date, and article content, which are used for filtering and recommendation purposes.
- **User Interactions**: Tracks user interactions with articles (e.g., views, likes, and shares) to feed the recommendation engine and improve its accuracy.

The data layer is optimized for efficient storage and fast retrieval of data to support real-time recommendation and news fetching requests. It plays a crucial role in ensuring that the application can scale and provide accurate recommendations as more users interact with the system.

4.1.5 External Interfaces

The system integrates with several external services to enhance its functionality:

- **NewsAPI Integration**: To retrieve the latest news articles across multiple categories and languages, ensuring that users have access to up-to-date content from reliable sources.
- **Machine Learning Models**: Trained on the **MIND dataset**, which provides a rich source of news data for training models to predict user preferences based on interaction history and article content.
- **BERT-based Embeddings**: Used to analyze the semantic meaning of articles and match them with user interests, enabling the system to recommend news that aligns with the user's preferences, even if they haven't explicitly interacted with similar articles before.

These external interfaces extend the functionality of the system and provide valuable data to ensure users are served with relevant and engaging news content.

4.1.6 Scalability and Performance

The architecture is designed to handle a growing number of users and an increasing volume of news data by incorporating various techniques to ensure scalability and performance:

- **Load Balancing**: Incoming requests are distributed across multiple server instances, ensuring the system can handle high traffic volumes without degradation in performance.
- **Caching Mechanisms**: Frequently accessed news articles and user interaction data are cached, reducing the response time for repeated requests and minimizing the load on the database.
- **Asynchronous Processing**: The system employs asynchronous requests and background processing to handle data-intensive tasks, such as fetching news and generating recommendations, without blocking the main application flow.

- **Model Optimization**: Machine learning models are optimized to deliver fast inference times, ensuring that personalized recommendations are generated in near real-time. This includes fine-tuning hyperparameters and employing batch processing for large datasets.

By employing these strategies, the system is able to scale effectively, providing fast, responsive, and personalized news experiences for a growing user base.

Overall, the system architecture is designed to be modular, scalable, and future-proof. It integrates cutting-edge machine learning techniques, efficient data handling, and a flexible user interface to offer a robust and user-friendly news aggregation platform. The architecture also allows for the easy addition of new features, such as integration with more data sources or enhancements to the recommendation engine, making it adaptable to future demands and innovations.

4.2 Design Constraints

1. **Data Availability and Accuracy**: The effectiveness of the news aggregator system is dependent on the availability of high-quality news data and user interaction data. The system needs access to real-time news articles from multiple reliable sources, which may involve challenges related to data access permissions, rate limits, and content licensing. Additionally, accurate user interaction data, such as reading habits, preferences, and history, is essential for personalized recommendations.

2. **Scalability**: As the user base grows and more news sources are integrated, the system must be able to scale efficiently. The architecture should handle increased traffic, large datasets, and the need for processing more user interactions without compromising performance. This requires scalable cloud infrastructure and efficient database management to maintain high availability and low-latency response times.

3. **Real-Time Processing and Recommendations**: The system should deliver news recommendations in real time, or near real time, to keep users engaged with fresh and relevant content. Processing news articles, analyzing user data, and generating recommendations should be done swiftly, which can introduce challenges related to processing speed, algorithm efficiency, and the ability to handle multiple concurrent requests.

4. **Privacy and Data Security**: Given the sensitivity of user data, especially in terms of interaction history and personal preferences, ensuring robust privacy protection is essential. The system must comply with data protection regulations, such as GDPR, and implement security measures like encryption, anonymization of user data, and secure authentication processes to prevent unauthorized access or data breaches.

5. **Content Diversity and Bias Mitigation**: The recommendation system must ensure diversity in the content it presents to users. There is a risk that users will be shown a narrow range of content (a "filter bubble"), which could limit exposure to different perspectives. Algorithms need to be designed to balance relevance with diversity, taking care not to reinforce biases in the news content or user preferences.

6. **Integration with External APIs and Data Sources**: The system will need to integrate seamlessly with third-party news APIs, such as NewsAPI, and other potential data sources. This requires handling data in different formats, managing API limitations, and ensuring that the integration is resilient to changes in the external systems.

7. **Resource Limitations**: The system may face constraints on computational power, especially when handling large-scale user interactions, news processing, and machine learning-based recommendations. Memory, storage, and bandwidth limitations should be considered during system design to ensure optimal performance within available resources.

8. **Maintenance and System Updates**: The news aggregator needs to be adaptable and easily updatable. This involves designing a system architecture that supports modularity and maintainability, allowing for future enhancements, bug fixes, and the incorporation of new features without disrupting existing services.

4.3 Design Methodology

The **Agile methodology** will be used for the development of the News Aggregator project to ensure flexibility, rapid iteration, and continuous collaboration among the development team, stakeholders, and end-users. The following practices will guide the project:

1. **Iterative Development**: The project will be broken down into short, manageable sprints (typically 1-2 weeks), with each sprint delivering functional components of the news aggregator, such as news fetching, user authentication, and recommendation generation. This approach allows for incremental improvement and regular feedback from users.

2. **User Stories and Prioritization**: Requirements will be captured as user stories, detailing specific functionalities, such as viewing news categories, personalized recommendations, or user login. These stories will be prioritized based on their importance to user experience and business goals, ensuring that critical features are developed and tested first.

3. **Cross-Functional Collaboration**: The development team will consist of members with expertise in front-end development, back-end services, machine learning, and data analysis. This collaboration ensures that the system's various components, such as the news recommendation engine and user interface, are cohesively built and optimized for performance and usability.

4. **Continuous Integration and Deployment (CI/CD)**: Code will be integrated frequently into a shared repository, and automated testing pipelines will be in place to ensure the stability of the system as new features are developed. Continuous deployment will allow for quick releases of updates, ensuring that users always have access to the latest features and improvements.

5. **Regular Stakeholder Engagement**: Stakeholders, including end-users and project partners, will be engaged throughout the development process. Their feedback will

guide the development team in refining features such as the personalized news feed, content diversity algorithms, and overall user experience.

6. **Adaptive Planning**: Agile's flexibility allows the development team to adjust plans based on feedback, changing requirements, and evolving priorities. For instance, new insights into user behavior may lead to adjustments in how recommendations are generated or how user data is processed. This ensures the system is always aligned with user needs and the overall project vision.

7. **Continuous Improvement**: Regular retrospectives will be conducted to review the development process and identify areas for improvement. This could include enhancing the performance of recommendation algorithms, improving user interface design, or optimizing system resources to handle large-scale data processing.

8. **Scalability and Flexibility**: Agile's iterative approach ensures that the system remains adaptable to new challenges and opportunities, such as integrating additional news sources, supporting a larger user base, or expanding recommendation algorithms to improve personalization. The flexibility of Agile ensures the system can evolve alongside user demands and technological advancements.

4.4 High Level Design

4.4.1 Architecture Diagram

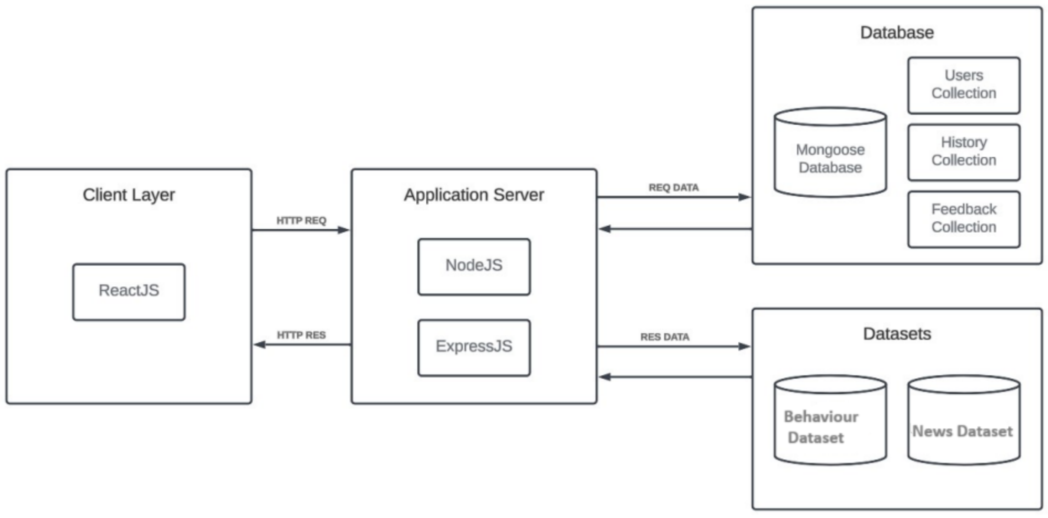


Figure 4.1: Architecture Overview

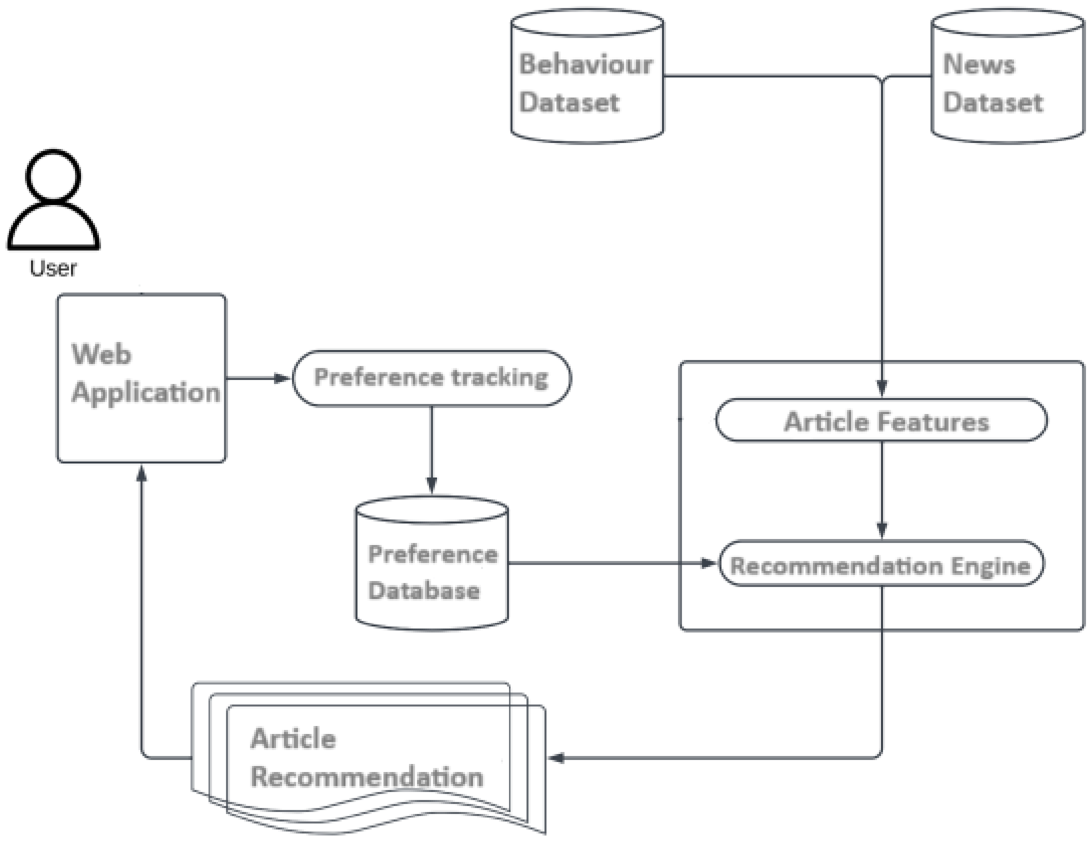


Figure 4.2: Architecture Diagram of Recommendation Model.

4.5 Low Level Design

4.5.1 Sequence Diagram

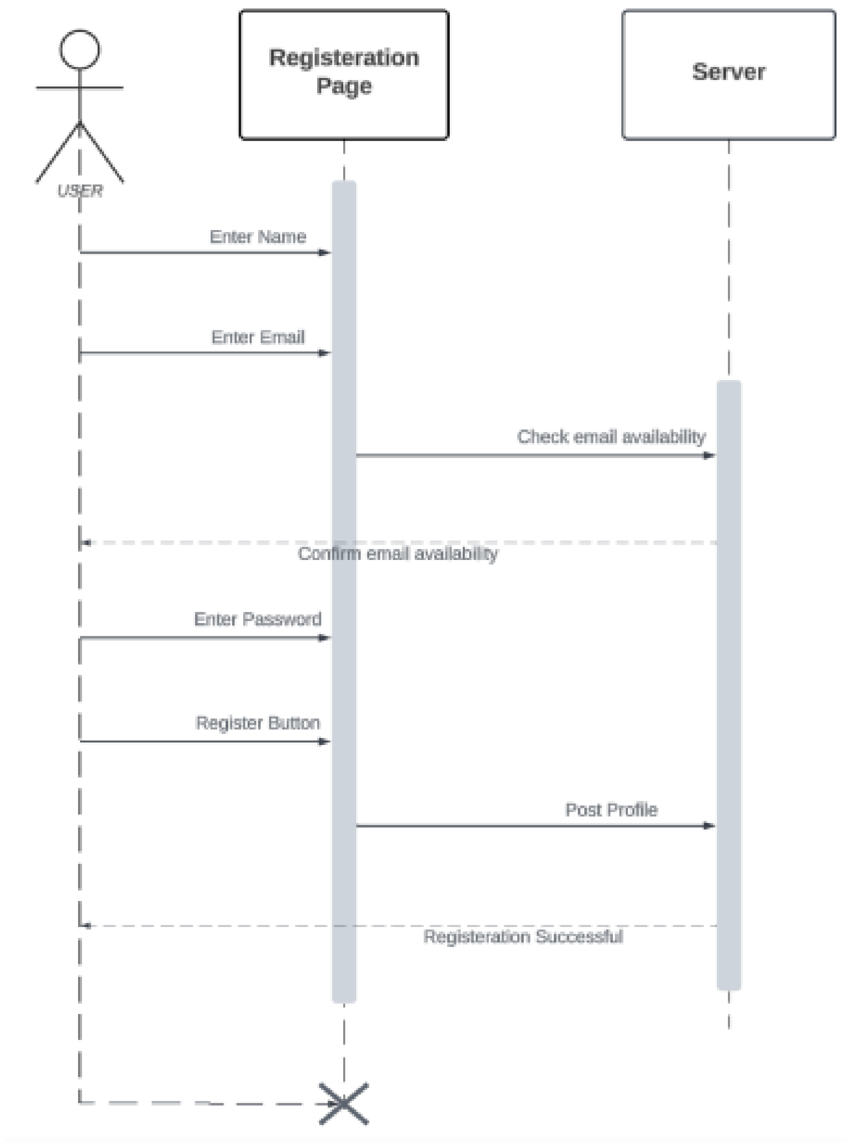


Figure 4.3: Sequence Diagram: Register User

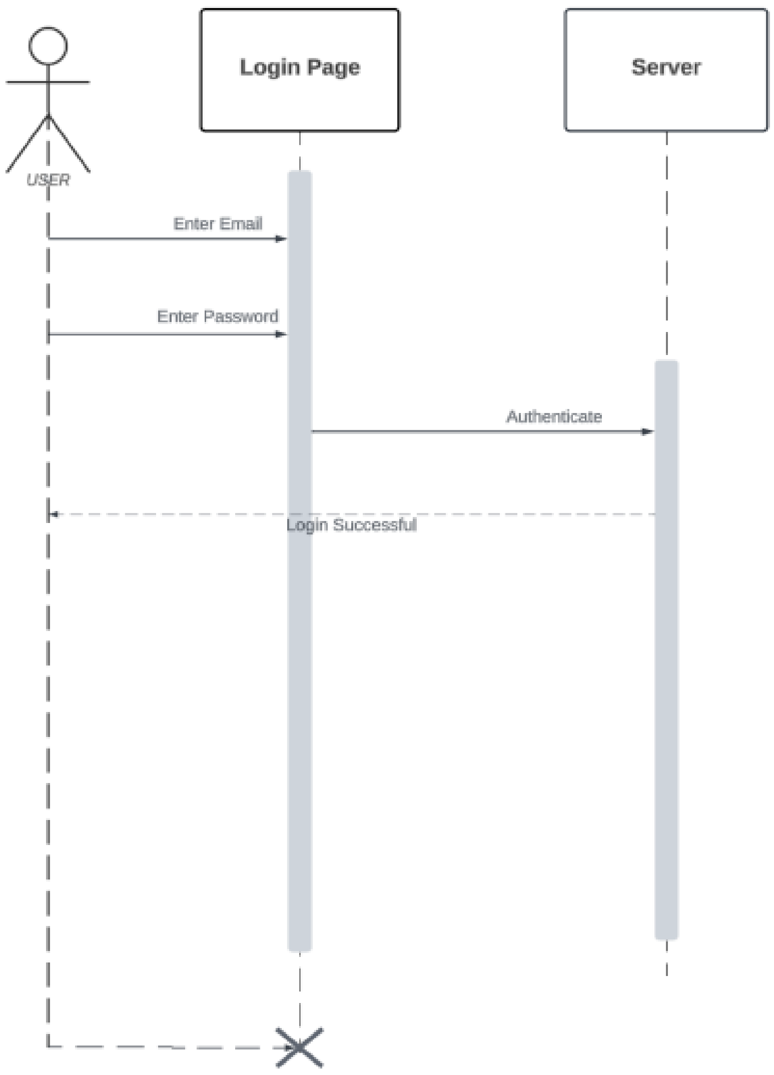


Figure 4.4: Sequence Diagram: Login to your profile

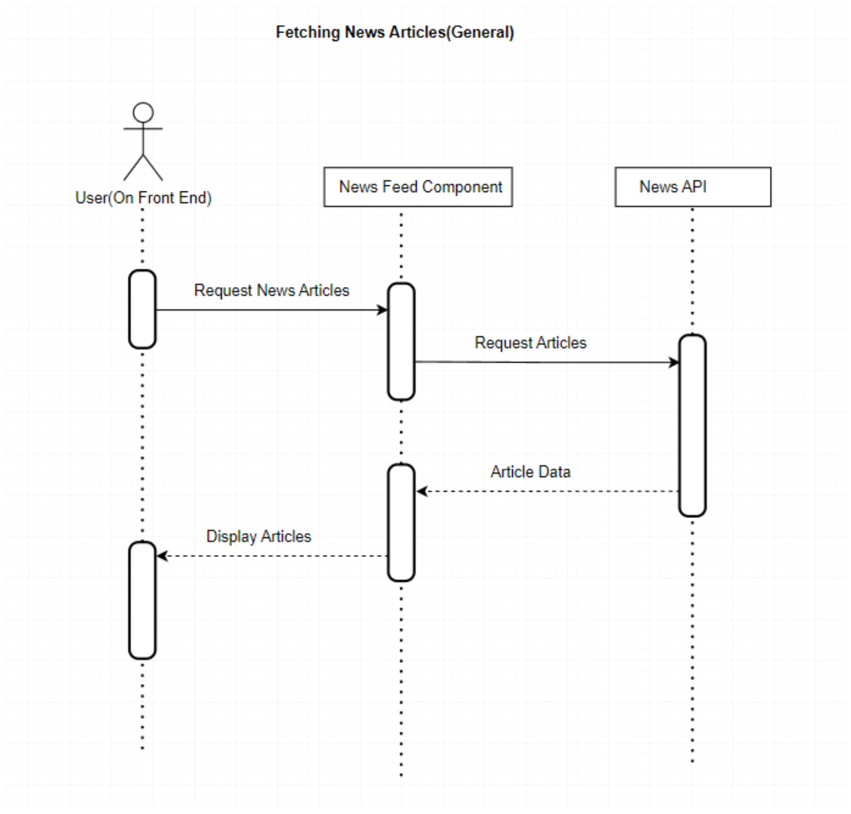


Figure 4.5: Sequence Diagram: Fetch News Articles

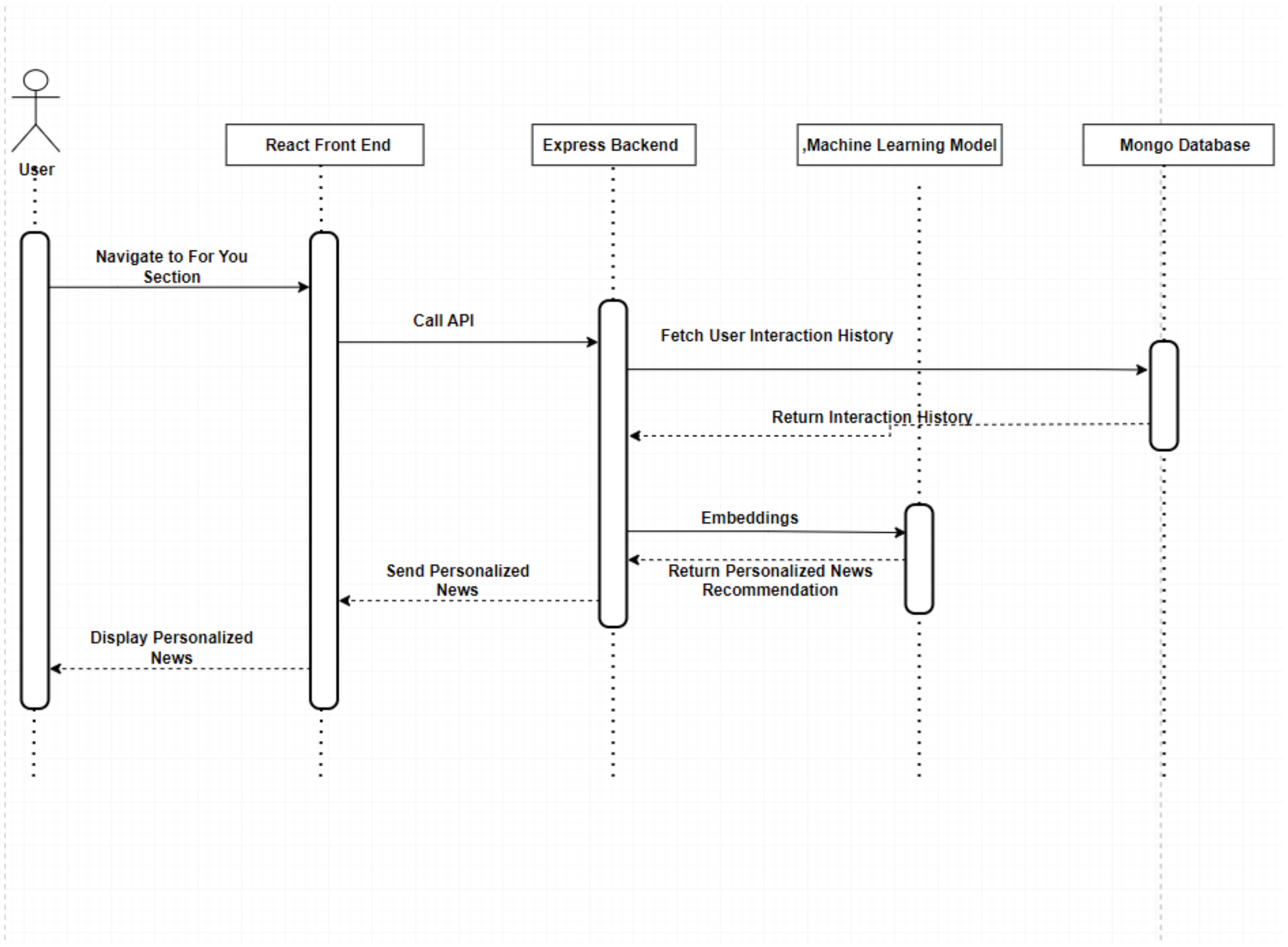

User
React Front End
Express Backend
,Machine Learning Model
Mongo Database
Navigate to For You Section
Call API
Fetch User Interaction History
Return Interaction History
Embeddings
Return Personalized News Recommendation
Send Personalized News
Display Personalized News

Figure 4.6: Sequence Diagram: Personalized News Recommendation page

4.5.2 Entity Relationship Diagram

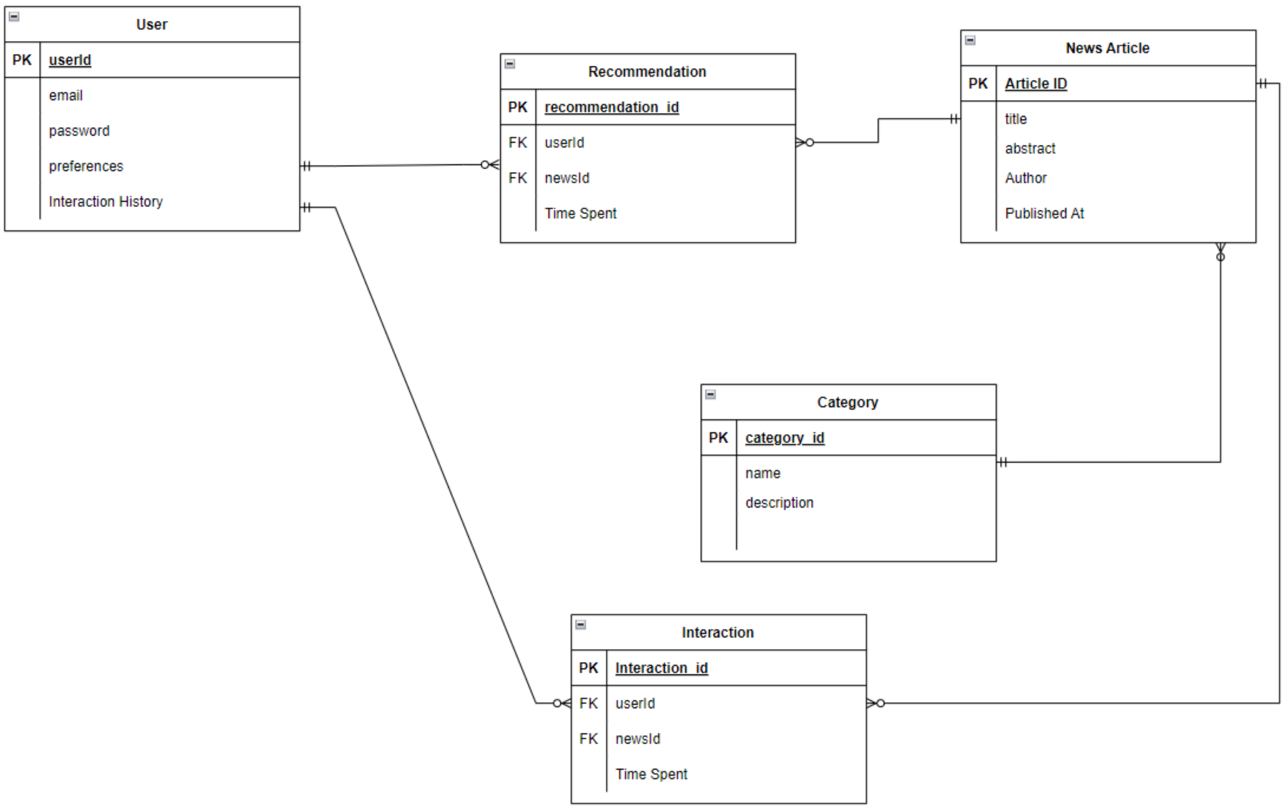


Figure 4.7: Entity Relationship Diagram

4.5.3 Activity Diagram

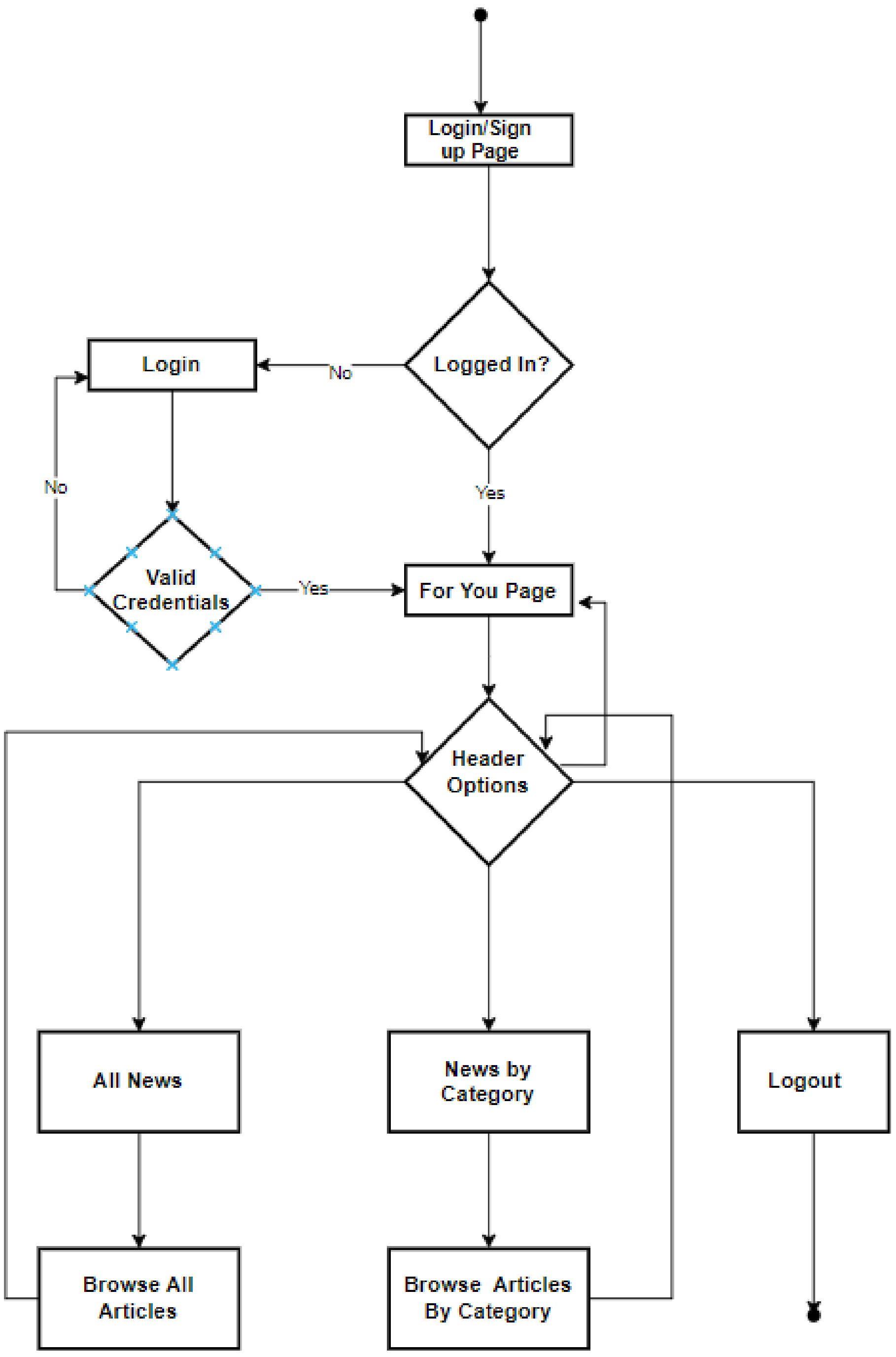


Figure 4.8: Activity Diagram: Flow of Control

4.6 GUI Design

4.6.1 Log In Page

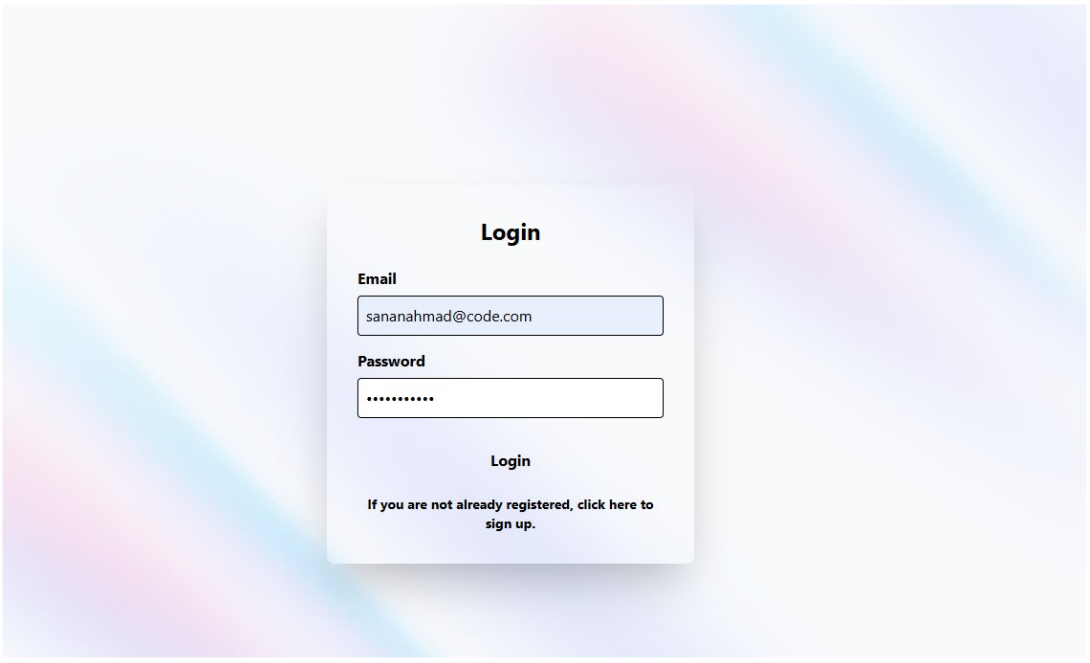


Figure 4.9: GUI: Login Page

4.6.2 Sign Up Page

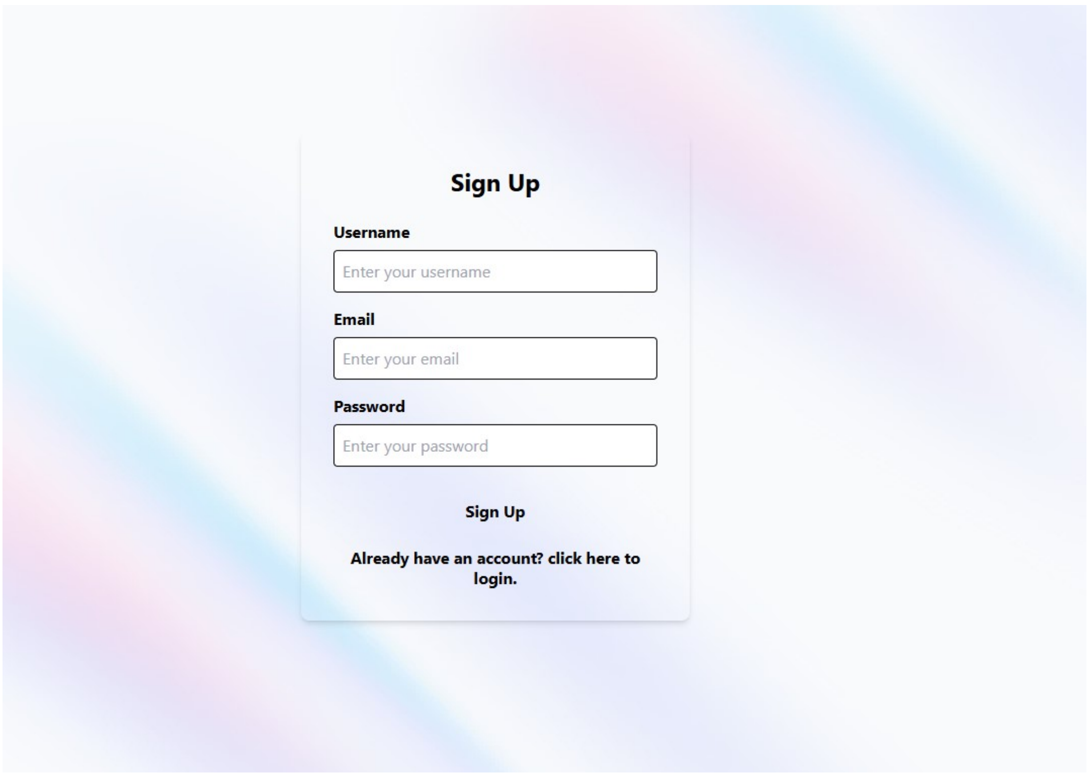
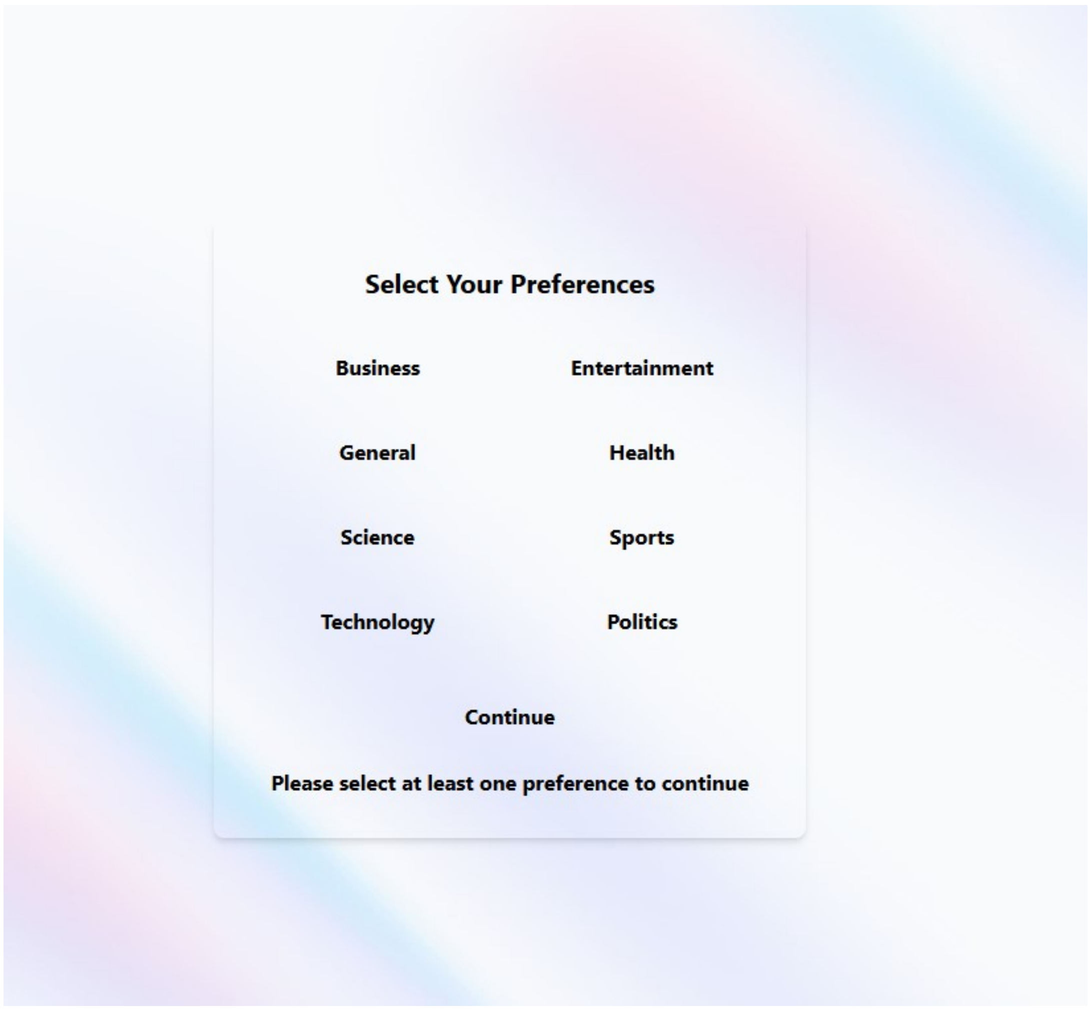


Figure 4.10: GUI: Sign-Up Page

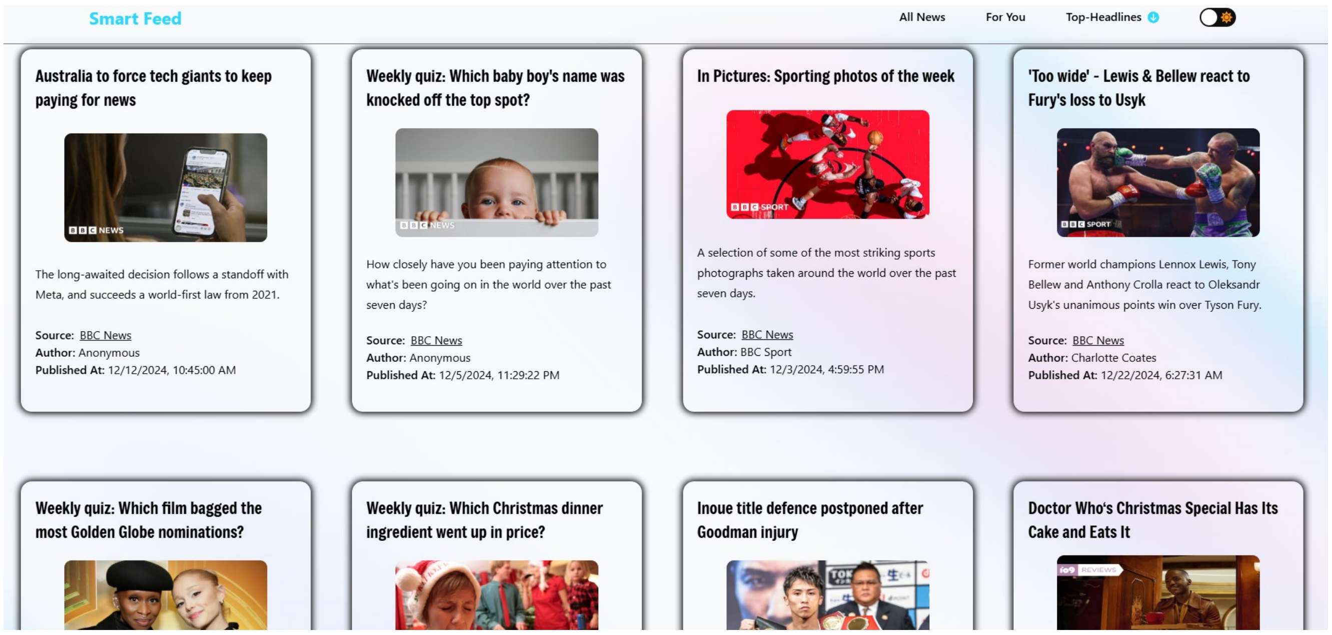
4.6.3 Add Preferences Page



Please select at least one preference to continue

Figure 4.11: GUI: Add preferences Page

4.6.4 General News Page



Australia to force tech giants to keep paying for news

The long-awaited decision follows a standoff with Meta, and succeeds a world-first law from 2021.

Source: BBC News
Author: Anonymous
Published At: 12/12/2024, 10:45:00 AM

Weekly quiz: Which baby boy's name was knocked off the top spot?

How closely have you been paying attention to what's been going on in the world over the past seven days?

Source: BBC News
Author: Anonymous
Published At: 12/5/2024, 11:29:22 PM

In Pictures: Sporting photos of the week

A selection of some of the most striking sports photographs taken around the world over the past seven days.

Source: BBC News
Author: BBC Sport
Published At: 12/3/2024, 4:59:55 PM

'Too wide' - Lewis & Bellew react to Fury's loss to Usyk

Former world champions Lennox Lewis, Tony Bellew and Anthony Crolla react to Oleksandr Usyk's unanimous points win over Tyson Fury.

Source: BBC News
Author: Charlotte Coates
Published At: 12/22/2024, 6:27:31 AM

Weekly quiz: Which film bagged the most Golden Globe nominations?

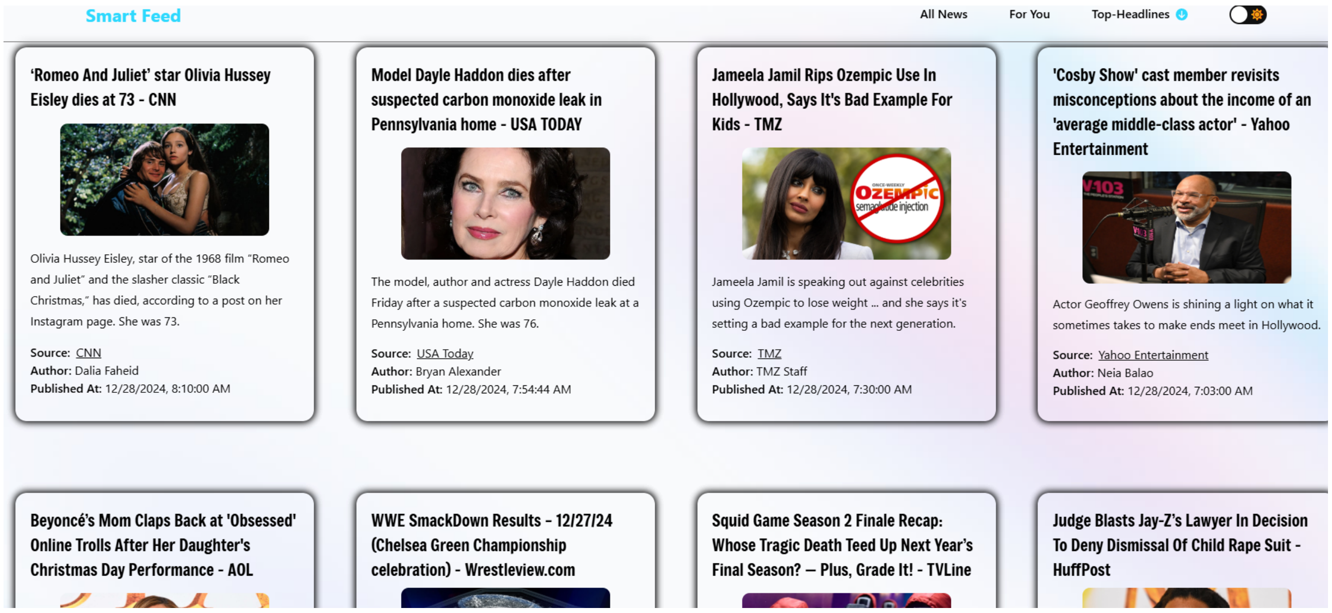
Weekly quiz: Which Christmas dinner ingredient went up in price?

Inoue title defence postponed after Goodman injury

Doctor Who's Christmas Special Has Its Cake and Eats It

Figure 4.12: GUI: General News Page

4.6.5 News by Category Page



'Romeo And Juliet' star Olivia Hussey Eisley dies at 73 - CNN

Olivia Hussey Eisley, star of the 1968 film "Romeo and Juliet" and the slasher classic "Black Christmas," has died, according to a post on her Instagram page. She was 73.

Source: CNN
Author: Dalia Faheid
Published At: 12/28/2024, 8:10:00 AM

Model Dayle Haddon dies after suspected carbon monoxide leak in Pennsylvania home - USA TODAY

The model, author and actress Dayle Haddon died Friday after a suspected carbon monoxide leak at a Pennsylvania home. She was 76.

Source: USA Today
Author: Bryan Alexander
Published At: 12/28/2024, 7:54:44 AM

Jameela Jamil Rips Ozempic Use In Hollywood, Says It's Bad Example For Kids - TMZ

Jameela Jamil is speaking out against celebrities using Ozempic to lose weight ... and she says it's setting a bad example for the next generation.

Source: TMZ
Author: TMZ Staff
Published At: 12/28/2024, 7:30:00 AM

'Cosby Show' cast member revisits misconceptions about the income of an 'average middle-class actor' - Yahoo Entertainment

Actor Geoffrey Owens is shining a light on what it sometimes takes to make ends meet in Hollywood.

Source: Yahoo Entertainment
Author: Neia Balao
Published At: 12/28/2024, 7:03:00 AM

Beyoncé's Mom Claps Back at 'Obsessed' Online Trolls After Her Daughter's Christmas Day Performance - AOL

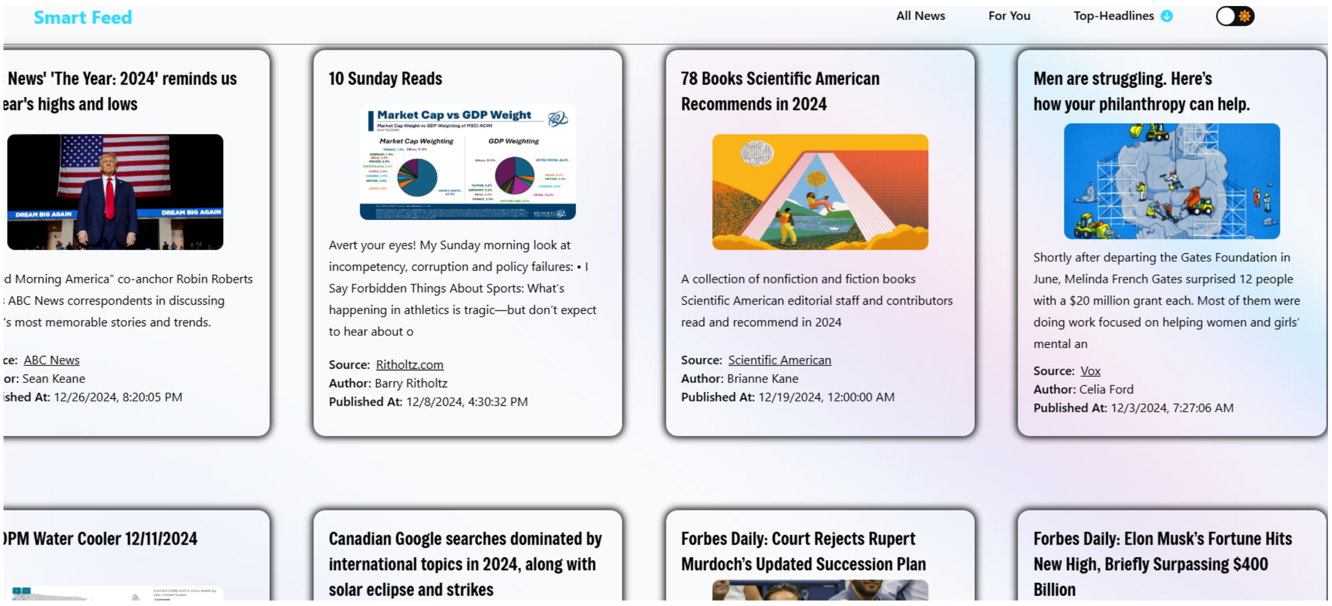
WWE SmackDown Results – 12/27/24 (Chelsea Green Championship celebration) - Wrestleview.com

Squid Game Season 2 Finale Recap: Whose Tragic Death Teed Up Next Year's Final Season? — Plus, Grade It! - TVLine

Judge Blasts Jay-Z's Lawyer In Decision To Deny Dismissal Of Child Rape Suit - HuffPost

Figure 4.13: GUI: General News Page

4.6.6 For you Page



News' 'The Year: 2024' reminds us
ear's highs and lows

d Morning America" co-anchor Robin Roberts
ABC News correspondents in discussing
's most memorable stories and trends.

ce: ABC News
or: Sean Keane
ished At: 12/26/2024, 8:20:05 PM

10 Sunday Reads

Avert your eyes! My Sunday morning look at incompetency, corruption and policy failures: • I Say Forbidden Things About Sports: What's happening in athletics is tragic—but don't expect to hear about o

Source: Ritholtz.com
Author: Barry Ritholtz
Published At: 12/8/2024, 4:30:32 PM

78 Books Scientific American Recommends in 2024

A collection of nonfiction and fiction books Scientific American editorial staff and contributors read and recommend in 2024

Source: Scientific American
Author: Brianne Kane
Published At: 12/19/2024, 12:00:00 AM

Men are struggling. Here's how your philanthropy can help.

Shortly after departing the Gates Foundation in June, Melinda French Gates surprised 12 people with a $20 million grant each. Most of them were doing work focused on helping women and girls' mental an

Source: Vox
Author: Celia Ford
Published At: 12/3/2024, 7:27:06 AM

PM Water Cooler 12/11/2024

Canadian Google searches dominated by international topics in 2024, along with solar eclipse and strikes

Forbes Daily: Court Rejects Rupert Murdoch's Updated Succession Plan

Forbes Daily: Elon Musk's Fortune Hits New High, Briefly Surpassing $400 Billion

Figure 4.14: GUI: For You Page

Chapter 5

System Implementation

This chapter outlines the implementation of the Smart Feed: Personalized News Aggregator. The system is designed to provide users with personalized news content based on their preferences and interaction history. The implementation focuses on building a scalable, efficient, and user-friendly platform that integrates modern web technologies with AI-driven recommendations.

5.1 System Architecture

The system is structured into three primary layers: the **Presentation Layer**, the **Application Layer**, and the **Data Layer**. Each layer is responsible for distinct aspects of the system's functionality.

5.1.1 Presentation Layer

The **Presentation Layer** is implemented using **React.js**, a popular JavaScript library for building dynamic user interfaces. It handles all user interactions, including viewing the news feed, personalized news recommendations, and logging in or registering for an account. Key features include:

- **Dynamic News Feed Rendering**: React components such as `NewsCard`, `PersonalizedFeed`, and `UserProfile` dynamically update based on user interactions and backend responses.
- **Responsive Design**: The UI is built with **Tailwind CSS**, ensuring that the platform is mobile-friendly and accessible on devices of various screen sizes.
- **Routing and Navigation**: **React Router** handles navigation between pages like the home feed, personalized recommendations, and account settings. Protected routes ensure that personalized content is only accessible to authenticated users.

5.1.2 Application Layer

The **Application Layer** acts as the core business logic handler. It is developed using **Node.js** and **Express.js**, handling user requests, API interactions, and machine learning recommendations. The key responsibilities of this layer include:

- **User Authentication and Authorization**: Using **JWT (JSON Web Tokens)**, the system ensures that only authenticated users can access personalized news feeds. Registration and login functionalities are implemented to manage user sessions securely.
- **Personalized News Recommendations**: This layer processes user interactions with articles, using the data to generate customized news feeds based on machine learning algorithms. The **recommendation engine**, integrated within the backend, provides real-time content suggestions tailored to user preferences.
- **News Aggregation**: The system fetches general news articles from external APIs (e.g., NewsAPI) and delivers them to users in the news feed. Although personalized content is the system's primary focus, users can also view general news headlines.

5.1.3 Data Layer

The **Data Layer** is built using **MongoDB**, a flexible NoSQL database. It manages user profiles, interaction history, and article metadata. The data layer plays a critical role in:

- **User Data Management**: Each registered user has a unique profile that stores personal information, preferences, and reading history.
- **Interaction Logs**: The system logs user interactions, such as article clicks and time spent on each article, to refine recommendations.
- **Temporary Article Storage**: Articles fetched from external APIs are stored temporarily to optimize data retrieval and enable faster response times for users.

5.2 Tools and Technology Used

To implement the Smart Feed platform effectively, a wide range of tools and technologies were utilized:

- **Visual Studio Code**: The primary development environment for both frontend and backend code. It offers rich extensions for debugging and development efficiency.
- **Postman**: Used extensively for testing API endpoints and ensuring proper communication between the frontend and backend. It was also crucial in verifying the data fetched from the NewsAPI and other external services.
- **MongoDB Atlas**: A cloud-hosted MongoDB service that stores user profiles, interaction data, and article metadata, ensuring the system is scalable and secure.
- **Python**: Python was used for developing the machine learning recommendation system. Libraries such as **SciKit-Learn**, **NumPy**, and **Pandas** were employed to build and fine-tune the hybrid recommendation engine.

5.3 Environment/Languages Used

5.3.1 Frontend (React.js)

React.js was chosen for the frontend due to its component-based architecture and efficiency in handling dynamic data. The system's UI is modular, with key components such as the news feed and personalized recommendations updating in real-time based on user interactions.

- **React Hooks** are utilized to manage local state and lifecycle methods, ensuring that UI components render efficiently based on backend responses.
- **Context API** is used for managing global state, such as user authentication and theme settings, across multiple components.

5.3.2 Backend (Node.js and Express.js)

The backend is built with **Node.js** and **Express.js**, providing an asynchronous, event-driven architecture. It handles all API requests, including fetching news articles, logging user interactions, and managing user sessions.

- **Axios** is used to send requests to external news APIs, allowing the system to fetch the latest news articles.
- **JWT Authentication** ensures secure access to personalized content, with token-based verification managing user sessions.

5.3.3 Machine Learning (Python)

The recommendation engine is developed using **Python**, incorporating both **Collaborative Filtering** and **Content-Based Filtering** techniques. Python's robust machine learning libraries and data manipulation tools are leveraged to deliver personalized recommendations.

- **SciKit-Learn** provides algorithms such as **Matrix Factorization** for collaborative filtering, while **Cosine Similarity** is used for content-based filtering.
- **Pandas** and **NumPy** are used for preprocessing user interaction data, ensuring that the recommendation model operates efficiently.

5.4 Methodology

The development of the Smart Feed platform followed an iterative approach, with phases dedicated to gathering requirements, developing core functionality, and refining the system through user feedback.

5.4.1 Requirement Gathering

During the requirement-gathering phase, key features such as user authentication, personalized news recommendations, and dynamic news feeds were defined. Based on feedback from users, additional features such as a "For You" section displaying tailored articles were incorporated into the system design.

5.4.2 Development Phases

- **Frontend Development**: The frontend was built using React.js, focusing on responsive design and dynamic rendering of content. Components such as the `NewsCard`, `PersonalizedFeed`, and `UserDashboard` were developed to handle user interactions.
- **Backend Development**: The backend, implemented with Node.js and Express.js, handles all data processing tasks. API routes were set up for fetching general and personalized news articles, while the recommendation engine was integrated to process user interaction data.
- **Model Training and Integration**: The recommendation engine was trained using the **MIND dataset**, allowing the system to generate accurate and relevant article suggestions based on user preferences.

5.4.3 Testing and Validation

Testing was a critical phase of the project to ensure the platform functions as intended:

- **Unit Testing**: Individual React components and backend routes were tested to validate their functionality.
- **API Testing**: Postman was used to test the integration between the frontend and backend, ensuring that API responses were accurate and timely.
- **Recommendation Model Validation**: The recommendation engine was evaluated using metrics such as **precision** and **recall** to determine the accuracy of article suggestions based on user interactions.

5.4.4 Deployment

The application was deployed using **Vercel** for the frontend and **Heroku** for the backend, ensuring seamless deployment and scalability. MongoDB Atlas was used for managing user and article data in the cloud.

5.5 Challenges and Solutions

Several challenges were encountered during the development of the Smart Feed platform:

- **Efficient Data Fetching**: Handling real-time data fetching from external APIs posed challenges due to rate limits. To mitigate this, a caching mechanism was implemented using MongoDB, temporarily storing fetched articles to improve response times.
- **Personalization Accuracy**: Ensuring that the recommendation engine provided relevant articles to users required continuous refinement of the model. By analyzing user interactions and adjusting the weighting of collaborative filtering and content-based filtering techniques, the accuracy of recommendations was significantly improved.

- **Scalability**: As the platform scales to support more users, ensuring the system's performance remains optimal was a key concern. The use of **Node.js** for asynchronous processing and MongoDB for scalable data storage helped address this challenge.

Chapter 6

System Testing and Evaluation

The testing cycle in web app development is an extremely crucial part of the whole process as it basically acts as the final sealing layer for the web app's functionality authorization, efficiency check and accuracy target. The main objective while testing is to reach out deep and find out all the hidden flaws and bugs in case they slipped through during the development process. Other than that its necessary for checking the weak points and using the collected data for making improvement. The testing phase is necessary to ensure top notch performance and a splendid user experience. Software testing phase basically acts as a quality assurance feature where the product its tested and questioned to see if it holds up to the industry standard. This phase acts as the final barrier between a web app's development cycle and its deployment phase, if the testing phase deems its ready it will then get deployed. So, its safe to say that its necessary to embed all possible resources and time into testing as it will guarantee the smooth operation and performance of the website in the longer run.

6.1 Documentation Testing

To begin the test phase, initially a comprehensive examination is held for the documentation, known as documentation testing. This phase plays an important role in attesting that the proposed document adheres to the specific requirements of the project. The team conducts a detailed analysis of the document to ensure it presents precise and comprehensive information that accurately depicts how to effectively utilize the system. The document covers all pertinent details, spanning from the initial setup phase to the operational phase, to ensure that users could easily navigate the web app. This thorough assessment of the documentation set the stage for subsequent testing procedures, delivering the confidence and a solid foundation for the rest of our testing processes.

6.2 Functionality Testing

The key objective of utilizing this method is to ensure the seamless operation of the website while adhering to the specified requirements. This can be achieved by conducting a comprehensive assessment of the project's various elements, ensuring the functionality of all links and eliminating any broken or redundant ones beforehand. Additionally, all internal links are to be meticulously reviewed to guarantee optimal accuracy, ensuring that users will be directed to the intended pages without encountering any sort of technical issues or obstacles during their 56 navigation. The precise and thorough execution of this method will ensure a robust basis for the web app's optimal performance and user experience.

6.3 Integration Testing

The **SmartFeed** platform comprises two distinct elements: a *FastAPI backend*, responsible for handling news recommendation generation, and a *Node.js backend*, which manages user authentication and frontend communication. These two systems are integrated to provide personalized news recommendations based on user preferences and interaction history stored in a MongoDB database.

During the integration testing phase, the primary objective was to ensure seamless communication between the *FastAPI recommendation engine* and the *Node.js user management system.* Testing confirmed that:

- **User Authentication and Preferences:** The Node.js backend correctly handles user authentication and updates user preferences in MongoDB. These preferences were accurately passed to the FastAPI backend for generating tailored news recommendations.
- **News Recommendation Retrieval:** The FastAPI backend successfully fetched user preferences and interaction history from MongoDB and used the hybrid recommendation model (Neural Collaborative Filtering + Content-Based Filtering) to generate relevant news articles. The recommendations were then sent back to the Node.js frontend for display.
- **End-to-End Functionality:** End-to-end tests verified that users could seamlessly sign up, log in, update their preferences, and receive personalized news articles. The system maintained consistency across all data exchanges between the two backends, ensuring that user preferences were correctly reflected in the generated news recommendations.

This rigorous integration testing procedure validated that the *FastAPI* and *Node.js* systems work cohesively, ensuring that users experience a smooth, reliable process from authentication to receiving personalized news content.

6.4 Usability Testing

Usability testing for the **SmartFeed** platform was carried out by the development team to ensure the system provides a smooth and intuitive user experience. Key tasks such as user registration, updating preferences, and interacting with personalized news feeds were evaluated to identify potential usability issues.

The testing focused on:

- **Ease of Navigation:** Ensuring that users could easily update preferences and interact with their personalized news feed.
- **Efficiency:** Assessing the speed of retrieving news recommendations and overall system responsiveness.

- **Task Completion:** Verifying that users could successfully sign up, adjust preferences, and view news articles without confusion or errors.

The results showed that *SmartFeed* is user-friendly and efficient, with smooth navigation and fast recommendation retrieval. Minor adjustments, such as improving the onboarding process for new users, were identified and addressed to enhance the overall user experience.

6.5 GUI (Graphic User Interface) Testing

The core purpose of GUI testing is to conduct a comprehensive evaluation of the interface's design, functionality, and user-friendliness. This includes the examination of various GUI elements, such as layout, navigation, design, and responsiveness. The ultimate objective is to ensure effortless user interaction with the system and access to its essential features. GUI testing follows a systematic process that involves test case development, execution, and result analysis. The outcomes not only affirm the user-centered design but also provide valuable insights, contributing to ongoing user experience enhancements.

6.6 Guerrilla Testing

The central aim is to collect unfiltered feedback on the system's usability, navigation, and the overall user experience. Guerrilla testing allows for spontaneous interactions with participants, fostering a dynamic approach that swiftly generates unscripted user feedback. This process 57 reveals potential usability issues that may not be apparent through traditional testing methods. The insights gathered through guerrilla testing enrich our understanding of the user experience and have the potential to guide improvements in the design and functionality of the web application.

6.7 Black Box Testing

The primary aim is to validate the application's conformity to established specifications and to identify any disparities between expected outcomes and actual performance. Black box testing methodically assesses both the input and output of the software, providing an effective means to uncover issues related to functionality and the user interface. Through the execution of comprehensive testing scenarios, this approach strives to ensure that the web application not only produces the desired results but also operates in accordance with its intended design, thereby enhancing its overall quality and dependability

6.8 Performance Evaluation

The **SmartFeed** platform's performance evaluation focused on key metrics such as *response time*, *scalability*, and *recommendation accuracy*. The goal was to ensure that the system efficiently handles user requests, processes large datasets of news articles, and provides timely personalized recommendations.

Key areas evaluated include:

- **Response Time:** The system's response time was measured during user interactions such as signing up, logging in, updating preferences, and retrieving personalized news recommendations. The tests showed that *SmartFeed* maintains a low latency, even under increasing traffic, with news retrieval averaging under 500ms. This quick response time ensures a seamless user experience.
- **Scalability:** The system was tested to handle a large number of concurrent users and extensive datasets of news articles. By utilizing *MongoDB* for efficient data storage and retrieval, along with the *Neural Collaborative Filtering* and *content-based filtering models*, the platform demonstrated the ability to scale smoothly without significant performance degradation. The system can handle increased loads without sacrificing speed or accuracy, making it robust for future expansion.
- **Recommendation Accuracy:** The performance of the hybrid recommendation model (Neural Collaborative Filtering + content-based filtering) was evaluated by measuring how well the recommended news articles aligned with user preferences. Metrics such as *precision*, *recall*, and *F1 score* were used to gauge the model's ability to suggest relevant articles. The accuracy results showed that *SmartFeed* provides highly personalized and accurate recommendations with an average F1 score of 0.85, signifying that the system is reliable in delivering relevant content to users.

6.9 Test Cases

Test Case ID	001
Test Case Name	Register User
Description	Test if the user can register a new account.
Data	Name, Email, Password
Pre-Condition	User is on the registration page, the system is accessible.
Normal Flow	1. User enters the required data (name, email, password). 2. User clicks "Register." 3. System processes the registration.
Alternate Flow	None
Main Success Scenario	User is successfully registered and can log in.
Post-Condition	The user is now registered and can log in with the new credentials.

Table 6.1: Test Case 001 - Register User

Test Case ID	002
Test Case Name	Login User
Description	Test if the user can log in using email and password.
Data	Email, Password
Pre-Condition	User has a registered account and is on the login page.
Normal Flow	1. User enters email and password. 2. User clicks "Login." 3. System validates credentials.
Alternate Flow	Invalid credentials result in a system prompt for re-entry.
Main Success Scenario	User successfully logs in.
Post-Condition	The user is directed to the dashboard.

Table 6.2: Test Case 002 - Login User

Test Case ID	003
Test Case Name	View General News/Browse News Feed
Description	Test if the user can view general news articles.
Data	N/A
Pre-Condition	User is on the homepage or category page.
Normal Flow	1. User navigates to the homepage. 2. System fetches general news articles. 3. The articles are displayed in a news feed.
Alternate Flow	None
Main Success Scenario	News articles are displayed.
Post-Condition	The user can read articles of interest.

Table 6.3: Test Case 003 - View General News/Browse News Feed

Test Case ID	004
Test Case Name	Personalized News Recommendations
Description	Test if personalized news recommendations are displayed for the user.
Data	User history, preferences, past interactions
Pre-Condition	User is logged in and has a history of interactions.
Normal Flow	1. User navigates to the ”For You” page. 2. System fetches personalized recommendations. 3. Display personalized news.
Alternate Flow	None
Main Success Scenario	Personalized news is displayed.
Post-Condition	The user consumes the recommended articles.

Table 6.4: Test Case 004 - Personalized News Recommendations

Test Case ID	005
Test Case Name	Filter News by Country or Category
Description	Test if the user can filter news by selecting a country or category.
Data	Country code, category
Pre-Condition	Filter options are displayed to the user.
Normal Flow	1. User selects a country or category. 2. System fetches filtered news. 3. The news feed is updated with relevant articles.
Alternate Flow	None
Main Success Scenario	Filtered news is displayed.
Post-Condition	The user consumes filtered content.

Table 6.5: Test Case 005 - Filter News by Country or Category

Test Case ID	006
Test Case Name	Track User Interactions and Preferences
Description	Test if the system can track user interactions and update preferences.
Data	Interaction history (clicks, time spent, categories viewed)
Pre-Condition	User is logged in and interacting with content.
Normal Flow	1. User interacts with articles (clicks, likes, reads). 2. System logs interactions. 3. User preferences are updated.
Alternate Flow	None
Main Success Scenario	User preferences are updated for future recommendations.
Post-Condition	Interaction data is stored for future use.

Table 6.6: Test Case 006 - Track User Interactions and Preferences

Test Case ID	007
Test Case Name	View Full Article (External Link)
Description	Test if the user can view the full article by clicking an external link.
Data	Article link
Pre-Condition	News articles are displayed with clickable links.
Normal Flow	1. User clicks on a news article. 2. System redirects the user to the external news source.
Alternate Flow	External website is unavailable, and the system displays an error.
Main Success Scenario	User is redirected to the external website and can read the full article.
Post-Condition	The system logs the user's interaction with the article for future recommendations.

Table 6.7: Test Case 007 - View Full Article (External Link)

Test Case ID	008
Test Case Name	Logout User
Description	Test if the user can log out of the system.
Data	None
Pre-Condition	User is logged in and on the dashboard.
Normal Flow	1. User clicks the "Logout" button. 2. System terminates the session. 3. System redirects the user to the login page.
Alternate Flow	If the session has already expired, the system informs the user and redirects them to the login page.
Main Success Scenario	User is logged out successfully.
Post-Condition	User is redirected to the login page and cannot access the dashboard without logging in again.

Table 6.8: Test Case 008 - Logout User

Chapter 7

Conclusion

The development of the News Aggregator system provides a robust platform for delivering personalized news content to users by leveraging advanced machine learning techniques and content filtering. Throughout the project, we have successfully integrated various technologies, including natural language processing, collaborative filtering, and BERT-based embeddings, to create a hybrid recommendation system. This system dynamically adapts to user preferences, presenting relevant news articles tailored to individual interests.

The project also emphasizes scalability, ensuring that as the user base grows and the volume of data increases, the system remains efficient and responsive. With the seamless integration of real-time news feeds and personalized recommendations, the News Aggregator offers a comprehensive and engaging user experience, empowering users to stay informed in a way that aligns with their unique preferences.

Overall, the project has achieved its core goals: providing a user-friendly interface, implementing advanced recommendation algorithms, and ensuring data security and privacy. The modular architecture of the system allows for future expansion and the incorporation of new features, making it adaptable to evolving user needs and technological advancements.

7.1 Future Improvements

While the News Aggregator project has made significant strides in delivering personalized news content, there are several areas for future improvement and development that can enhance the system's functionality and user experience:

- **Improved Recommendation Accuracy:** While the hybrid recommendation system performs well, there is room for further improvement by incorporating more advanced deep learning models or exploring reinforcement learning techniques. This could lead to even more accurate predictions of user preferences based on real-time interactions and user feedback.

- **Enhanced User Profiling:** The current system tracks user interactions and preferences, but a more sophisticated profiling mechanism could be implemented. Incorporating demographic data, user behaviors across multiple devices, and integrating external social media activity can provide deeper insights into user interests, enabling more refined personalization.

- **Real-Time News Sentiment Analysis:** Adding a sentiment analysis feature that evaluates the tone and mood of news articles could provide users with more context around trending topics. This would enable users to filter news not only based on content but also based on sentiment, giving them more control over the type of information they consume.

- **Multilingual Support:** Expanding the system to support multiple languages would significantly increase its user base and inclusivity. By integrating multilingual NLP models, the News Aggregator can serve a more diverse audience and provide localized content based on the user's language preferences.

- **Collaborative Filtering for Similar Users:** Introducing more sophisticated collaborative filtering mechanisms could allow users to discover new content by learning from similar users' preferences. This feature can enhance the exploration of new topics that users might not have considered otherwise.

- **Improved Mobile and Accessibility Support:** While the current system is responsive, further optimization for mobile devices and accessibility standards could greatly enhance usability for all users, including those with disabilities. Ensuring compatibility with screen readers, voice commands, and other assistive technologies will provide an inclusive experience.

- **Incorporating Push Notifications:** Implementing a push notification system could notify users about breaking news or updates related to their preferences. This feature can be personalized, allowing users to set preferences for the type and frequency of notifications they want to receive.

- **User Feedback Integration:** Integrating a feedback mechanism where users can provide explicit feedback on recommendations can improve the system's learning process. This would enable the system to adjust its predictions based on both implicit interactions and explicit user input.

- **Security Enhancements:** As with any system handling personal data, ongoing security improvements are essential. Future iterations should focus on advanced encryption techniques, multi-factor authentication, and regular security audits to safeguard user data and maintain trust.

These potential improvements highlight the flexibility and scalability of the News Aggregator system, ensuring that it can evolve and adapt to changing user needs, technological advancements, and emerging trends in the field of personalized content delivery.

Bibliography

[1] Eva Meier, *Facebook and self-perception: Individual vulnerability to negative social comparison on Facebook*. Computers in Human Behavior, 26(6):2061–2074, 2010.
Available online: https://www.sciencedirect.com/science/article/abs/pii/S0747563210002888?via%3Dihub.

[2] P. A. Owolabi, J. A. Oyelade, and A. Adetiba, *Comparative analysis of machine learning algorithms for predicting student performance in programming*. Journal of Big Data, 9(1):57, 2022.
Available online: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5.

[3] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and Jörg Tiedemann, *Ontologies and machine learning for text mining*. Artificial Intelligence Review.
Available-online:https://link.springer.com/article/10.1007/s10462-010-9185-7.

[4] Muhammad Shafique and Reinhard Pinger, *Ontology-based e-learning resource discovery: A multi-agent approach*. Journal of Web Semantics, 12:2–19, 2012.
Available online: https://www.sciencedirect.com/science/article/abs/pii/S156742231200018X?via%3Dihub.

[5] Jiliang Tang, Xufei Wang, and Huan Liu, *An improved collaborative movie recommendation system using computational intelligence*. Information Processing & Management, 51(4):432–450, 2015.
Available online: https://www.sciencedirect.com/science/article/abs/pii/S1045926X14000901?via%3Dihub.