ISSN 2277-2685
IJESR/April-June. 2026/ Vol-16/Issue-2/1045-1051
B Soumya *et. al.*, /International Journal of Engineering & Science Research
Full Length Article

Personalized Real Time Update

Ms G Rajeshwari[1], B Soumya[2], Ch Tejaswini[3], P Vaishnavi[4]
[1]Assistant Professor; Department of Computer Science and Engineering Bhoj Reddy Engineering College for Women Hyderabad India.
[2,3,4]B.Tech Student's; Department of Computer Science and Engineering Bhoj Reddy Engineering College for Women Hyderabad India.
Mail Id; bodasoumya22@gmail.com@gmail.com[2]
Accepted 02-06-2026

Abstract
*The **Personalized Real-Time Update System** is designed to deliver relevant and timely information to users by analyzing their preferences, interests, and behavioral patterns. Traditional notification systems often generate excessive and irrelevant updates, leading to information overload and reduced user engagement. To address these challenges, the proposed system utilizes real-time data processing and personalization techniques to provide customized content updates that match individual user needs.*
The system continuously collects and processes live data from multiple sources, enabling instant notification delivery based on user-defined preferences and interaction history. Modern technologies such as Node.js, cloud-based databases, and real-time communication frameworks are integrated to ensure high scalability, low latency, reliability, and efficient performance. Features including user profile management, content filtering, preference analysis, and push notification services contribute to an enhanced user experience by delivering meaningful and context-aware updates.
By combining real-time computing with intelligent personalization mechanisms, the proposed system improves information accessibility, increases user engagement, and minimizes unnecessary notifications. The solution demonstrates how advanced real-time technologies can be leveraged to build responsive, efficient, and user-centric information delivery platforms.

Keywords: *Personalized Real-Time Updates, Real-Time Data Processing, Push Notifications, User Preference Analysis, Content Filtering, Node.js, Cloud Database, User Engagement, Information Delivery System, Personalized Notifications.*

Introduction

The rapid growth of digital platforms and online information sources has significantly increased the volume of data available to users. As a result, individuals often experience information overload due to the continuous flow of updates, notifications, and content from multiple channels. Traditional information delivery systems typically distribute the same updates to all users regardless of their interests, resulting in reduced relevance and lower user engagement. To address these challenges, a Personalized Real-Time Update System is proposed to deliver customized information based on individual user preferences, interests, and behavioral patterns.

The proposed system continuously collects and processes live data from various sources and intelligently filters information before delivering it to users. By maintaining personalized user profiles that store preferences such as content categories, notification settings, and priority levels, the system ensures that only relevant updates are transmitted. Real-time communication technologies enable instant delivery of notifications without requiring users to refresh web pages or applications manually. Through the integration of personalization and real-time processing, the system enhances user experience, improves engagement, and provides efficient access to meaningful information.

Existing System

Existing information delivery systems primarily focus on broadcasting updates to a broad audience with limited personalization capabilities. Although

some modern applications incorporate basic preference settings and behavioral tracking, many systems still provide generic notifications that may not align with individual user interests. In such environments, users frequently receive excessive or irrelevant information, reducing the overall effectiveness of the notification process.

Many conventional systems support profile management and content subscriptions; however, they often lack advanced mechanisms for real-time content filtering and personalized update generation. Furthermore, information delivery may depend on periodic synchronization or manual refresh operations, leading to delays in communication and reduced responsiveness. While these systems can support multiple users simultaneously, their ability to provide highly targeted and context-aware updates remains limited.

Proposed System

The proposed Personalized Real-Time Update System introduces an intelligent framework for delivering customized information through real-time communication technologies. The system continuously monitors incoming data streams and instantly distributes relevant updates to users based on their stored preferences and interests. Unlike traditional approaches, the proposed solution eliminates the need for manual page refreshes by utilizing push-based notification mechanisms.

The system supports multiple concurrent users while maintaining low response times and high processing efficiency. User profiles, preference settings, notification history, and interaction data are securely maintained within a cloud-based database environment. Advanced filtering techniques analyze user preferences and determine the relevance of incoming information before delivering updates. The architecture is designed to be scalable, enabling future integration of artificial intelligence, recommendation engines, analytics modules, and cross-platform communication services.

Requirements Analysis

Functional Requirements

The Personalized Real-Time Update System is designed to provide secure user management, real-time data processing, and personalized content delivery. The system enables users to register and authenticate themselves through secure login mechanisms. Once authenticated, users can create, modify, and manage their profiles, including preference settings related to content categories, notification priorities, and delivery options.

The platform continuously collects data from various sources and processes incoming information streams in real time. Based on user preferences and behavioral data, the system filters content and generates customized notifications. The architecture ensures seamless communication between data sources, processing modules, and notification services, enabling efficient delivery of relevant updates.

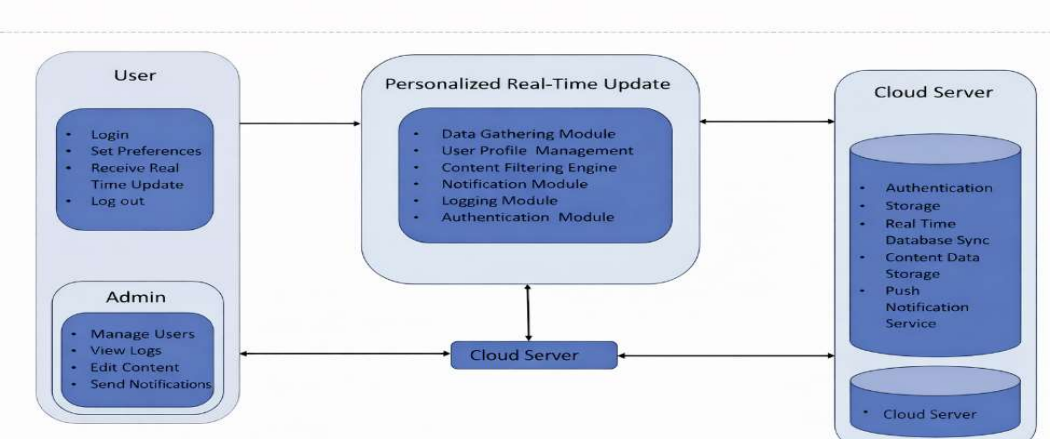
Resource Requirements

The proposed system requires modest computational resources for deployment and operation. A computing environment equipped with an Intel Core i3 processor or higher, a minimum of 4 GB RAM, and approximately 20 GB of available storage is sufficient for development and testing purposes. A stable internet connection is necessary for real-time communication and cloud-based services.

The software environment includes modern web development technologies, cloud database services, and real-time communication frameworks. These technologies collectively support scalable deployment, efficient processing, and secure management of user data and notifications.

System Design

Software Architecture



1.Software Architecture

The software architecture follows a layered design approach consisting of client, application, and cloud service layers. The client layer includes end users and administrators who interact with the system through web or mobile interfaces. Users configure preferences and receive personalized updates, while administrators manage content, users, and system operations.

The application layer serves as the central processing component and includes modules responsible for data collection, user profile management, content filtering, authentication, notification management, and activity logging. These modules collaborate to analyze incoming information and generate customized updates based on user-specific requirements. The modular design improves maintainability, flexibility, and scalability.

Technical Architecture

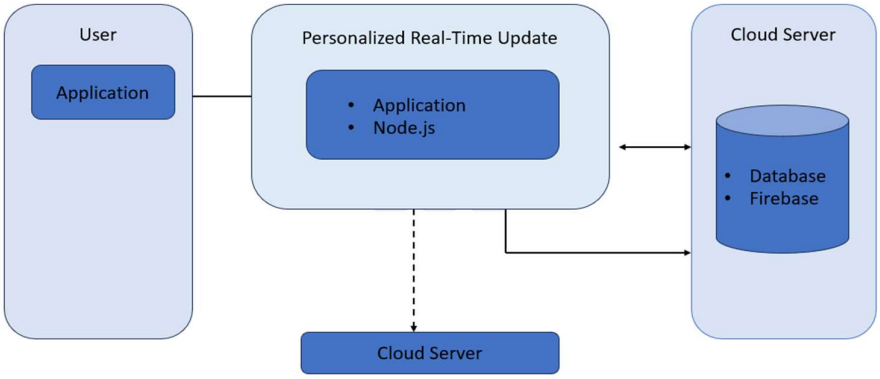


Fig.2 Technical Architecture

The technical architecture integrates cloud computing services, real-time communication technologies, and intelligent content processing mechanisms. User requests and preference settings are transmitted to the application server, where data processing and filtering operations are performed. The notification engine evaluates incoming information streams and determines content relevance for individual users.

Cloud infrastructure provides secure authentication services, scalable storage capabilities, and real-time database synchronization. Push notification services enable immediate communication between the system and end users. This architecture ensures efficient data flow, low latency, high availability, and reliable operation while supporting personalized information delivery on a large scale.

IMPLEMENTATION

System Implementation

The Personalized Real-Time Update System was developed using modern web technologies and real-time communication mechanisms to provide customized information delivery based on individual user preferences. The implementation focuses on collecting live data, processing

information streams, filtering relevant content, and delivering personalized updates with minimal delay. The architecture follows a client-server model that enables efficient communication between users, application services, and cloud-based data storage systems.

JavaScript and Node.js serve as the primary development technologies for the application. JavaScript is utilized to implement dynamic user interactions and real-time content rendering on the client side, while Node.js provides a scalable server-side environment for handling requests, processing data, and managing communication between system components. The event-driven architecture of Node.js allows the application to efficiently handle multiple concurrent users and real-time update requests.

The user interface is developed using HTML and CSS. HTML provides the structural foundation for the application, including profile management, update feeds, notification panels, and dashboard components. CSS enhances the visual appearance through responsive layouts, styling elements, and user-friendly navigation mechanisms. Together, these technologies create an intuitive interface that improves user experience and accessibility.

Express.js is employed as the backend framework to simplify server-side development and API management. The framework facilitates request routing, middleware integration, authentication handling, and communication between frontend and backend modules. Its lightweight architecture enables rapid processing of user requests and efficient delivery of personalized content.

Real-time functionality is achieved through JavaScript-based dynamic update mechanisms that allow users to receive information instantly without requiring manual page refreshes. Incoming updates are continuously monitored and processed by the server. Relevant information is then delivered to users according to their stored preferences and interests. This approach significantly improves responsiveness and ensures timely access to information.

The project also utilizes Node Package Manager (npm) for dependency management and integration of third-party libraries. npm simplifies software maintenance and accelerates development by providing access to a large ecosystem of modules and development tools. The combined use of these technologies results in a scalable, responsive, and efficient platform capable of supporting personalized real-time information delivery.

System Workflow

The operational workflow of the system begins with user registration and authentication. During registration, users provide credentials such as username, email address, and password. Input validation procedures ensure that mandatory information is provided and security requirements are satisfied before account creation. Passwords are encrypted prior to storage to enhance system security and protect user privacy.

Following successful authentication, users define their interests and preferences, which are stored within the database. These preferences may include categories such as technology, education, employment opportunities, entertainment, sports, or current events. The stored preferences serve as the basis for content personalization and recommendation processes.

The system continuously monitors external information sources and application programming interfaces for newly available content. When new updates are detected, the information is processed and temporarily stored. A personalization engine then evaluates the relevance of each update by comparing content categories with user preferences. Only information that matches a user's interests is selected for delivery.

Personalized updates are subsequently displayed within the user dashboard and may also be delivered through notification services. This workflow ensures that users receive relevant information in real time while minimizing unnecessary notifications and reducing information overload.

TESTING AND VALIDATION

Testing Overview

Testing is a critical phase in software development that ensures the developed application performs according to specified requirements. The Personalized Real-Time Update System was subjected to comprehensive testing procedures to verify functionality, performance, reliability, and user experience. The primary objective of testing was to confirm that each component of the system operated correctly and that all modules interacted seamlessly within the overall architecture.

The evaluation process focused on validating user authentication, profile management, preference handling, data retrieval, personalization logic, notification delivery, dashboard functionality, and

session management. Testing activities helped identify potential defects and ensured that the application met expected quality standards before deployment.

Functional Testing

Functional testing was performed to verify the correctness of all major system features. User registration functionality was tested to ensure that new accounts could be created successfully and stored within the database. Login functionality was evaluated using both valid and invalid credentials to confirm proper authentication behavior and error handling mechanisms.

Preference management functionality was tested to verify that user interests and notification settings were correctly stored and retrieved. Real-time update retrieval mechanisms were evaluated by requesting data from external sources and confirming successful processing of incoming information. The personalization module was tested to ensure that users received only content relevant to their selected interests.

Dashboard functionality was examined to confirm that personalized updates were displayed correctly and updated dynamically without requiring page refreshes. Notification services were tested to verify the successful delivery of alerts and email notifications. Logout functionality was evaluated to ensure proper session termination and secure redirection to the login interface.

Test Results and Validation

The testing results demonstrated that all major system components performed according to their intended functionality. User registration and authentication processes operated successfully, while preference management features accurately stored and retrieved user data. Real-time data acquisition and processing mechanisms consistently delivered updated information from external sources.

The personalization engine successfully filtered content according to user interests, ensuring that notifications remained relevant and meaningful. Dashboard updates were displayed correctly and reflected newly available information without significant delays. Notification services functioned effectively across multiple testing scenarios, confirming the reliability of real-time communication mechanisms.

The validation process further confirmed that the system achieved its primary objective of delivering personalized updates while minimizing information overload. The combination of real-time processing and user-centric content filtering contributed to improved usability, responsiveness, and overall system performance.

Screenshots

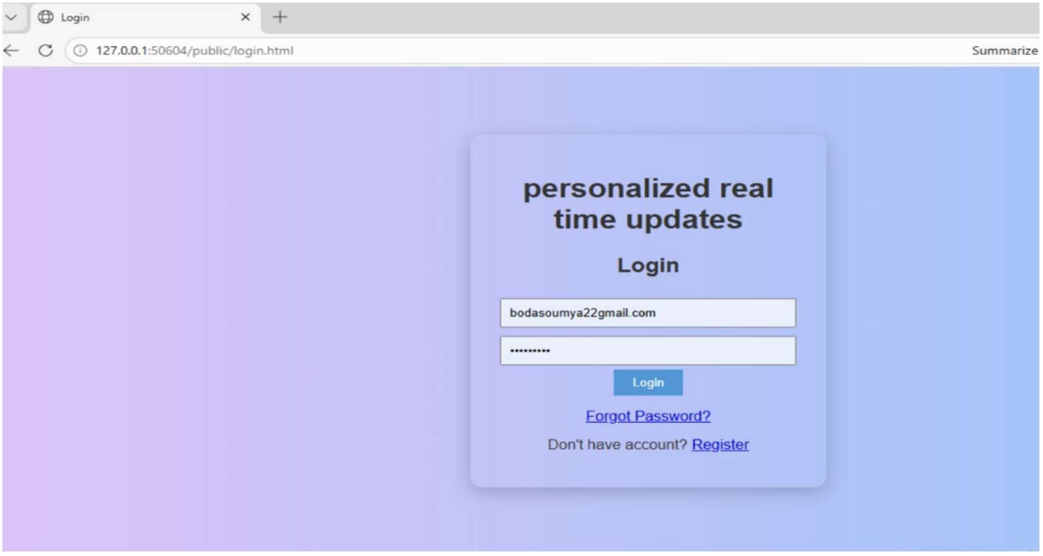


Fig 1:Login

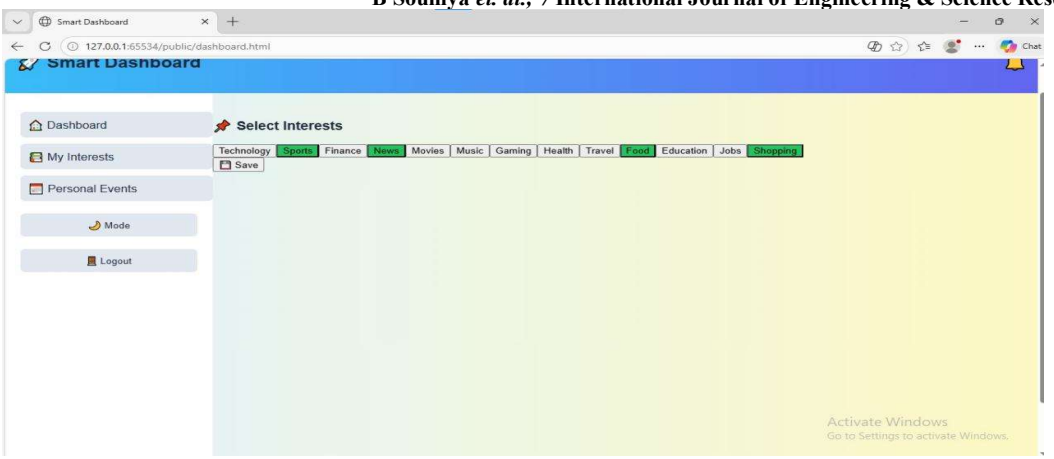


Fig 2: Select Interests

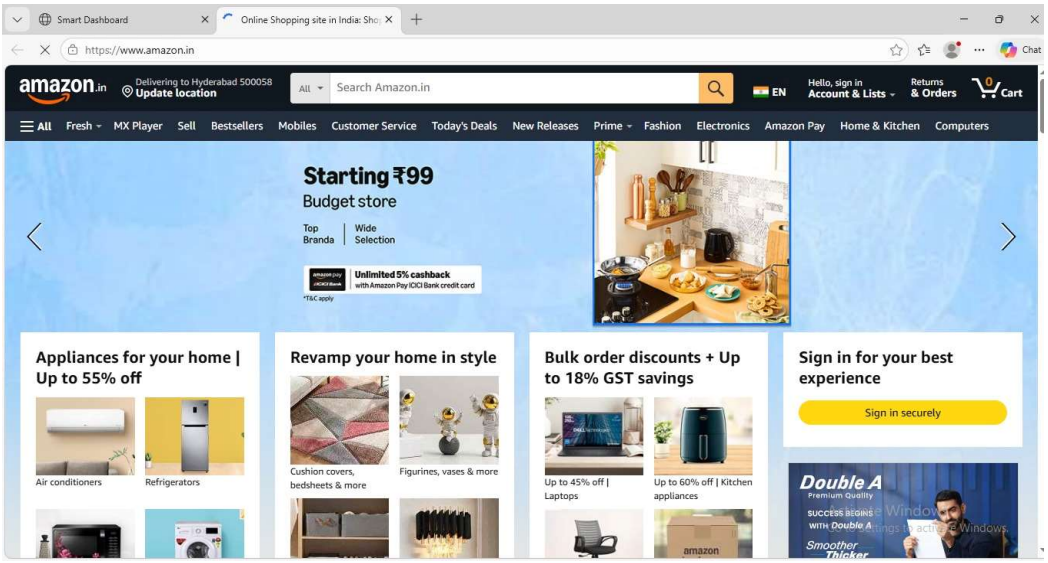


Fig 3: Shopping

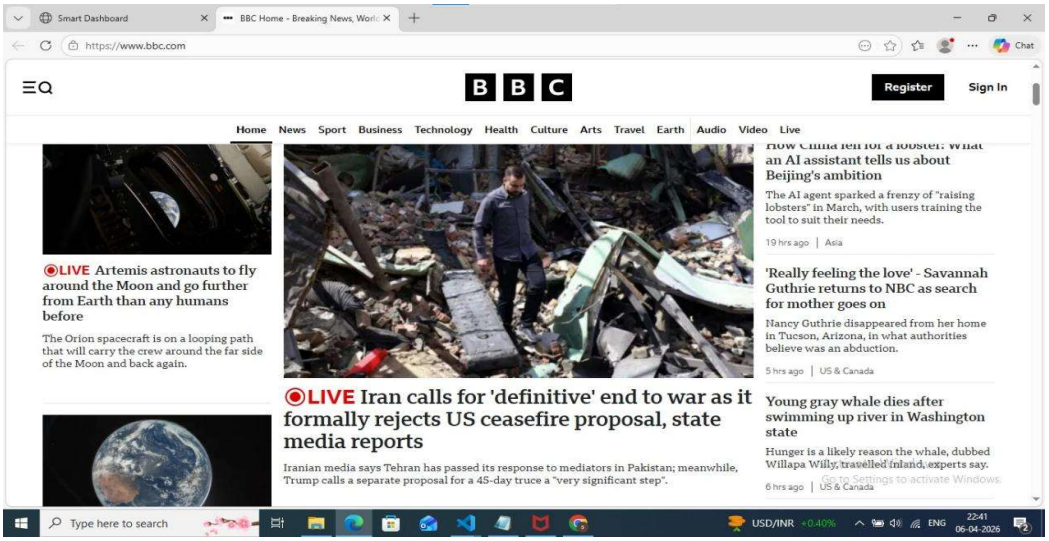


Fig 4: News

Fig 5: Food

CONCLUSION

This research presented a Personalized Real-Time Update System designed to improve information delivery through the integration of real-time communication technologies and user-centric personalization mechanisms. The proposed framework continuously collects and processes live data, analyzes user preferences, and delivers customized updates that align with individual interests. By filtering irrelevant content and prioritizing meaningful information, the system enhances user engagement and reduces information overload.

The implementation demonstrated that modern web technologies such as Node.js, Express.js, JavaScript, HTML, and CSS can be effectively utilized to build scalable and responsive real-time applications. Comprehensive testing confirmed the reliability of user authentication, content filtering, notification delivery, and dashboard management functionalities. The results indicate that personalized real-time information systems can significantly improve user experience and information accessibility across various application domains.

FUTURE SCOPE

Several opportunities exist for extending the capabilities of the proposed system. Future development may incorporate Artificial Intelligence and Machine Learning techniques to improve personalization accuracy through predictive analysis of user behavior and content consumption patterns. Intelligent recommendation algorithms could further enhance content relevance and user satisfaction.

A dedicated mobile application may be developed to provide seamless access to personalized updates across smartphones and tablets. Mobile integration would improve accessibility and enable continuous engagement regardless of user location. Additionally, cloud-native architectures and distributed processing frameworks could be adopted to support larger user populations and higher volumes of real-time data.

Future versions of the system may also integrate external APIs, social media feeds, business intelligence services, and analytics platforms to expand the range of available information sources. These enhancements would transform the platform into a comprehensive intelligent information management system capable of supporting diverse real-world applications.

REFERENCES

[1] Google Developers – Firebase Documentation, used for real-time database management and cloud messaging services.

[2] Firebase Documentation, used for handling real-time data storage and synchronization of updates.

[3] WebSockets Documentation, used for enabling real-time bidirectional communication between client and server.

[4] News API Documentation, used for fetching real-time news data and external updates.

[5] Node.js Official Documentation, used for backend development and server-side application handling.

[6] MySQL Documentation, used for structured storage and management of user data and preferences.

[7] Mozilla Developer Network (MDN), used for learning and implementing HTML, CSS, and JavaScript concepts.