Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025

CHALLENGE BASED SOCIAL MEDIA APP

Aman Kumar 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India aman2021@gift.edu.in
Priyansu Ranjan Samantaray 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India psamantaray2021@gift.edu.in
Mrs. Smrurti Samaraki Sarangi Assistant Professor, Department of CSE, Gandhi Institute for Technology, BPUT, India

Abstract—
This paper presents the development of a cross-platform social media application that promotes user engagement through challenge-based interactions. Built using React Native for the frontend and Node.js with MongoDB for the backend, the app allows users to create, participate in, and respond to multimedia challenges. It integrates real-time chat via Firebase, a reel-style content feed, and a competitive leaderboard system. The system aims to foster creativity, enhance social interaction, and gamify content sharing. This paper discusses the system architecture, implementation details, testing, and performance evaluation of the app.

Keywords:
React native, Node js, Express.js, MongoDB, Firebase , Cloudinary

1. Introduction

Social media has revolutionized how people connect, communicate, and express themselves in the digital age. With the rise of platforms like Instagram, TikTok, and Snapchat, the focus has shifted from simple text-based updates to highly engaging visual content. While these platforms allow users to share photos and videos, the interaction is often passive, lacking direct user-to-user engagement through content-driven tasks or challenges.

This paper presents the design and development of a **Challenge-Based Social Media Application**, a cross-platform mobile app that aims to enhance user interaction through gamified and media-driven social experiences. The core idea of the application is to allow users to **create challenges** in the form of photo or video prompts, which can be **accepted and completed by others**, promoting creativity, fun, and social engagement.

The app incorporates modern features like a **reel-style feed**, **real-time chat**, and a **leaderboard system** to foster competition and visibility among users. It is built using **React Native**, which ensures cross-platform compatibility across Android and iOS devices, while the backend is powered by **Node.js** and **Express.js**, with **MongoDB** as the primary database. The app also integrates **Firebase** for real-time chat functionality and media storage services to manage user-uploaded content.

The project addresses a gap in current social media platforms by encouraging **active participation** rather than passive content consumption. It combines the appeal of visual content with the thrill of competition and the satisfaction of social recognition. This paper outlines the system architecture, development methodology, implementation details, testing procedures, and experimental results of the application.

Through this work, we aim to demonstrate how challenge-based interaction can reshape user engagement on social platforms, providing a foundation for future enhancements using artificial intelligence, personalization, and scalable cloud technologies.

2. Related Works (Literature Review)

The rapid evolution of social media platforms has led to the development of diverse applications that enable users to connect, share, and engage with content in real-time. Platforms like **Instagram**, **TikTok**, and **Snapchat** have transformed the way people interact by emphasizing multimedia content

such as images and videos. While these platforms encourage creativity, user interaction is typically limited to likes, comments, and shares, with fewer opportunities for task-based or challenge-driven engagement.

In recent years, there has been a growing interest in **gamification in social networks**, where traditional social features are enhanced with game-like mechanics to increase user participation and retention. According to research by Deterding et al. (2011), gamification significantly improves user motivation and behavior, especially when applied in contexts involving peer interaction and recognition systems like leaderboards.

Apps such as **TikTok** have experimented with challenges, where users mimic or build on a particular trend. However, these challenges are not directly initiated or tracked between users, and the system lacks structured challenge-response workflows. This presents an opportunity for platforms that **directly enable users to post and respond to individual challenges**, which can be monitored, rated, and ranked.

Several research studies also explore **real-time communication** in mobile applications. Firebase, for example, has been widely adopted for its low-latency performance and ease of integration in cross-platform apps. It enables features such as live chat, notifications, and real-time database updates, which are essential for interactive social applications.

Moreover, the use of **cross-platform development frameworks** like **React Native** has become popular in recent years. React Native allows developers to write a single codebase for both Android and iOS, reducing development time and improving consistency across devices.

This paper builds upon existing research and practices by combining challenge-based interaction, gamification elements (like leaderboards), real-time chat, and cross-platform compatibility to develop a more engaging and socially interactive mobile application. The system aims to fill a unique space in the social media ecosystem by encouraging **active participation through user-generated challenges**.

3. System Architecture

The system architecture of the **Challenge-Based Social Media App** is designed using a modular and scalable approach, ensuring flexibility, maintainability, and support for real-time interactions. The application consists of four primary layers: the **Client (Frontend)**, the **Backend (Server)**, the **Database Layer**, and **Third-party Services** for real-time communication and media handling.

3.1 Client Layer (Frontend)

The client-side of the application is developed using **React Native**, which supports both Android and iOS platforms from a single codebase. The user interface includes screens for:

- User authentication (login/sign-up)
- Challenge creation and submission
- Reel-style challenge feed
- Chat interface
- Leaderboard and user profiles

The app communicates with the backend server via RESTful APIs and WebSocket/Firebase channels for real-time features. It also integrates third-party media services to handle media uploads efficiently.

3.2 Backend Layer (Server)

The backend is built using **Node.js** with the **Express.js** framework. It serves as the central point for business logic, handling:

- User management and authentication using JWT
- CRUD operations for challenges and responses
- Leaderboard computation
- API endpoints for chat, media management, and notifications

The server also ensures data validation, session handling, and role-based access control.

3.3 Database Layer

The database layer uses **MongoDB**, a NoSQL document-based database well-suited for flexible, JSON-like data structures. Key collections include:

- Users: user credentials, profiles, preferences
- Challenges: challenge content, media URLs, metadata
- Responses: user submissions with references to the original challenge
- Chats: chat messages and timestamps
- Leaderboard: scores and rankings of users

MongoDB's schema-less design supports fast development and efficient querying, especially for dynamic content like media and user interactions.

3.4 Third-party Integrations

- **Firebase**: Used for real-time chat and notifications due to its reliable real-time database and messaging capabilities.
- **Cloud Storage Services** (e.g., Cloudinary or Firebase Storage): Used to store and serve media files like challenge videos and images.
- **Authentication Tools**: bcrypt for password hashing and jsonwebtoken for secure user sessions.

This modular architecture ensures that the application is responsive, secure, and scalable for increasing user demands. The integration of real-time features and media handling within a well-structured backend allows for smooth and engaging user experiences.

4. Implementation Details

4.1 Frontend (React Native)

The frontend was developed using **React Native**, which enables cross-platform development for Android and iOS using a shared JavaScript codebase. Major UI elements include:

- **Authentication Screens**: Includes login and signup forms, JWT token-based session handling, and secure validation.
- **Challenge Creation**: Users can create a new challenge by uploading an image/video, adding a title and description.
- **Challenge Feed (Reels UI)**: Infinite scrollable feed displaying trending and new challenges using flatlist components with optimized media rendering.
- **Response Submission**: Allows users to respond to a challenge by recording or uploading media and submitting it with a caption.
- **Leaderboard Page**: Displays user rankings based on number of challenges completed and likes received.
- **Chat Interface**: A real-time chat interface built using **Firebase Firestore** to support instant messaging between users.
- **Navigation**: Implemented with **React Navigation**, supporting tab and stack navigators for smooth transitions.

4.2 Backend (Node.js + Express.js)

The backend uses a RESTful architecture and is built with **Node.js** and **Express.js**, responsible for all business logic and API endpoints.

- **Authentication & Security**: Uses bcrypt for password encryption and jsonwebtoken (JWT) for secure session management.
- **API Endpoints**:
  - POST /register and POST /login
  - POST /challenges – create challenge
  - GET /feed – fetch feed of challenges
  - POST /response – submit challenge response
  - GET /leaderboard – fetch ranked user list
- **Middleware**: Auth token validation middleware protects private routes.

- **Media Upload Handling**: Integrates with services like **Cloudinary** or **Firebase Storage** via signed URLs for secure media handling.

4.3 Database (MongoDB)

MongoDB is used to store all application data in a NoSQL document format. Key collections include:

- **Users**: { _id, name, email, passwordHash, profileImage, stats }
- **Challenges**: { _id, userId, title, mediaUrl, description, timestamp }
- **Responses**: { _id, challengeId, userId, responseMediaUrl, timestamp }
- **Leaderboard**: Dynamically computed based on responses and likes.
- **Messages**: Stored using Firebase Firestore for low-latency real-time access.

The flexible schema of MongoDB supports rapid iteration, allowing easy updates as features expand.

4.4 Real-time Communication (Firebase)

- **Firebase Firestore** is integrated to implement real-time chat.
- Chat messages are stored in a NoSQL format with timestamps, sender/receiver IDs, and message content.
- Real-time listeners ensure instant updates in chat interfaces.

4.5 Media Handling and Optimization

- Media files are compressed before upload to reduce load time.
- A CDN (via Cloudinary or Firebase Storage) delivers optimized content to users quickly.
- Thumbnails and lazy-loading techniques are used in the feed for improved performance.

5. Methodology

The development of the Challenge-Based Social Media App followed a structured, iterative approach using the **Agile Software Development Methodology**. This methodology was chosen due to its adaptability, support for rapid prototyping, and continuous feedback integration, which were essential for building a feature-rich, user-centric mobile application.

5.1 Requirement Gathering

The process began with defining the core functionalities required for the app. These included:

- User registration and login
- Challenge creation and participation
- Real-time chat system
- Feed-style browsing experience
- Leaderboard system
- Media upload and handling

Requirements were prioritized based on user engagement potential and implementation complexity.

5.2 System Analysis and Design

Once the features were finalized, a system-level design was created. This involved:

- Designing the architecture (client-server model with third-party integrations)
- Choosing appropriate technology stacks (React Native, Node.js, MongoDB, Firebase)
- Defining database schemas
- Creating UI wireframes and user flows using Figma

APIs were planned with RESTful principles to ensure modularity and scalability.

5.3 Agile Development Process

The development cycle was divided into **sprints**, each lasting one to two weeks. Each sprint involved:

- Planning tasks and features
- Implementing components (frontend, backend, database)
- Continuous testing
- Regular feedback and iteration

Tools used included:

- **GitHub** for version control
- **Trello** or **Jira** for sprint planning and task tracking

- **Postman** for API testing
- **Expo CLI** for live preview and debugging of React Native apps

5.4 API Development and Integration

Backend APIs were developed using Express.js, with middleware for:

- Authentication (JWT)
- Media validation
- Rate limiting and error handling

Frontend components fetched data via Axios calls, and Firebase SDK was integrated for chat and notifications.

5.5 Real-Time Features Implementation

Firebase Firestore was selected for its simplicity and reliability in handling real-time chat. Listeners were used to update the UI instantly upon receiving messages.

5.6 Testing and Deployment

- **Unit Testing**: Performed on individual components and backend functions
- **Integration Testing**: Verified communication between frontend and backend
- **User Testing**: Collected feedback from initial testers to improve usability
- **Deployment**: Backend was deployed on platforms like **Render** or **Heroku**, while the app was built and tested using **Expo Go**

This methodology ensured that the app was built efficiently, with attention to performance, usability, and extensibility. Agile allowed continuous refinement based on real-time feedback, making the final product robust and user-focused.

6. System Testing and Evaluation

System testing plays a critical role in ensuring that all components of the application function correctly, both individually and when integrated. The goal was to identify defects, ensure performance standards, and verify that the system met all user requirements. A combination of **manual testing**, **automated testing**, and **user testing** was employed to evaluate the application's functionality, usability, and reliability.

6.1 Types of Testing Performed

6.1.1 Unit Testing

- Focused on testing individual functions, components, and modules.
- Examples:
  - Validating user input fields (e.g., login form)
  - API response handling for challenge submissions
  - Media upload logic and validations

6.1.2 Integration Testing

- Ensured that different components of the system worked well together.
- Tested interactions such as:
  - Submitting a challenge and retrieving it in the feed
  - Chat messages sent from one user appearing instantly to another
  - Leaderboard updates after challenge completion

6.1.3 System Testing

- Complete end-to-end testing of the application.
- Verified workflows such as:
  - User registration to challenge posting
  - Browsing and responding to challenges
  - Viewing leaderboard and profile stats

6.1.4 Usability Testing

- Conducted with a group of real users to gather feedback on:
  - User interface layout and design

- Navigation flow and ease of use
- App responsiveness and intuitiveness

6.1.5 Performance Testing

- Evaluated the speed, scalability, and responsiveness of the app.
- Key metrics tested included:
  - Load time of challenge feed
  - Media upload/download speeds
  - Real-time performance of chat module

6.2 Bug Tracking and Fixes

Bugs were documented using a bug-tracking tool (e.g., GitHub Issues). Each reported issue was categorized based on severity and addressed in the next sprint. Common issues included:

- Media file upload failures due to size limitations
- API rate limit problems
- Firebase real-time sync delays under slow network

6.3 Evaluation Results

The application was tested across multiple Android and iOS devices to ensure compatibility and performance consistency. Key outcomes:

- Over **95% of critical functionalities** worked as expected
- User feedback reported a **positive experience** with the challenge feed and chat responsiveness
- The app showed **stable performance** under moderate load (up to 100 concurrent users in simulation)

6.4 Tools and Frameworks Used

- **Postman**: API testing
- **Jest & React Testing Library**: Component/unit testing for frontend
- **Firebase Emulator Suite**: Local testing for real-time services
- **MongoDB Compass**: Database query validation
- **Expo CLI & Android/iOS Simulators**: UI testing across devices

The comprehensive testing approach ensured that the Challenge-Based Social Media App is stable, scalable, and provides a smooth user experience. Continuous testing during development allowed for early bug detection and faster deployment.

7. Experimental Results

The **Challenge-Based Social Media App** was evaluated through experimental testing in a controlled environment and among a group of initial users to measure its effectiveness, performance, and user engagement. The primary aim of this evaluation was to verify that the app meets its functional goals, performs reliably under expected load, and delivers a seamless user experience.

7.1 Test Setup

- **Devices Used**:
  - Android (Samsung Galaxy M31, Redmi Note 10 Pro)
  - iOS (iPhone XR, iPhone 12)
  - Web simulator via Expo
- **Network Conditions**:
  - 4G and Wi-Fi networks
  - Simulated poor network using throttling tools
- **Test Users**:
  - 20 early adopters including students, developers, and social media users
  - Usage simulated over 1 week with active challenge creation and responses

7.2 Performance Metrics

Metric	Observed Result
App Launch Time	2.3 seconds (avg)
Feed Load Time	1.8 seconds
Media Upload Time (5MB file)	~4.2 seconds
Chat Message Sync Time	< 500ms (real-time)
API Response Time (avg)	~200ms
Backend Uptime (Simulated)	99.5%

7.3 User Engagement Results

- **Challenges Created**: 78
- **Responses Submitted**: 150+
- **Messages Exchanged in Chat**: ~1,000
- **Average Session Duration**: 6.5 minutes
- **Feedback Score (UI/UX)**: 8.7/10

Participants responded positively to the:

- Intuitive UI and smooth navigation
- Engaging challenge/reel format
- Real-time chat and notifications
- Easy media upload and response system

7.4 Observations

- **Strengths**:
  - Seamless cross-platform support
  - Real-time features work well even in poor networks
  - Efficient media handling with CDN integration
- **Areas for Improvement**:
  - Occasional lag during video playback in feed
  - Challenge search and filtering can be enhanced
  - Push notifications can be expanded for better user retention

7.5 Summary

The experimental results confirm that the application is robust, responsive, and offers a valuable platform for social interaction through challenges. The app successfully meets its core objectives and performs well under typical usage conditions, making it ready for broader deployment and future enhancements.

8. Benefits and Limitations

This section highlights the key strengths of the Challenge-Based Social Media App, along with the limitations identified during development and testing. Understanding both aspects is crucial for future improvements and scaling the application for wider adoption.

8.1 Benefits

1. Cross-Platform Compatibility

The app is developed using **React Native**, which allows seamless deployment on both Android and iOS platforms using a single codebase. This reduces development time and maintenance effort.

2. Engaging and Interactive User Experience

The challenge-based format promotes active user participation and content generation, making the platform more engaging than traditional social media apps.

3. Real-Time Communication

The integration of **Firebase Firestore** enables users to chat in real-time, fostering better interaction and community building among users and challenge participants.

4. Scalable Backend Architecture

Built using **Node.js** and **Express.js**, the backend supports a scalable and modular architecture suitable for future feature expansion and increased user load.

5. Media Optimization

The use of **CDN-based media storage (e.g., Cloudinary or Firebase Storage)** ensures fast content delivery and smooth performance even under network constraints.

6. Gamification and Social Motivation

Features like **leaderboards**, **likes**, and **challenge completion stats** encourage user competition and motivation, which can lead to higher user retention and app usage.

8.2 Limitations

1. Limited Offline Functionality

Since the app heavily relies on real-time data (feed, chat, leaderboard), it offers limited usability in offline or poor network conditions.

2. Basic Search and Filtering

The current version has limited filtering and search options for challenges, which could impact content discoverability as the app scales.

3. Media Upload Constraints

Although optimized, media upload is subject to size and format limitations, which can restrict user creativity or slow performance on lower-end devices.

4. Push Notification Support

While chat is real-time, push notification support for events like new challenges or responses is basic and needs further enhancement for better user engagement.

5. Security Measures

Basic security measures like JWT and password hashing are implemented, but advanced security features (e.g., multi-factor authentication, end-to-end encryption for chat) are not yet available.

8.3 Summary

Despite some limitations, the Challenge-Based Social Media App successfully delivers an interactive and scalable solution for social engagement through challenges. Its current architecture and modular design provide a strong foundation for future enhancements, such as AI-based content recommendations, advanced security, and a more refined user experience.

9. Future Scope

The Challenge-Based Social Media App has demonstrated strong potential as a unique and engaging platform for interactive content sharing. However, to enhance its reach, performance, and user experience, several future enhancements and expansions can be pursued. These improvements will ensure the app remains competitive, scalable, and relevant to evolving user needs and technological advancements.

9.1 Advanced Recommendation System

To improve content discovery, an AI-powered recommendation engine can be integrated. By analyzing user activity, preferences, and engagement history, the app could recommend:

- Relevant challenges
- Friends or users to follow
- Trending or location-based content

This will increase user engagement and personalize the user experience.

9.2 Video Editing and AR Filters

Incorporating a lightweight in-app video/photo editor with features like trimming, filters, stickers, and augmented reality (AR) effects will boost the quality and creativity of user-submitted challenges. This also encourages longer session times and more engaging media.

9.3 Enhanced Push Notifications

A more comprehensive push notification system can improve retention and engagement by informing users about:

- New challenges from friends or trending creators
- Replies or completions to their posted challenges
- Leaderboard updates or milestone achievements

Integration with services like Firebase Cloud Messaging (FCM) can provide real-time, device-specific alerts.

9.4 Monetization Features

To support creators and generate revenue, the app can introduce:

- In-app purchases or tokens for premium features
- Brand-sponsored challenges
- Creator monetization through views or participation metrics

This would make the platform more appealing for influencers and content creators.

9.5 Community and Moderation Tools

As the user base grows, there is a need for:

- User reporting and content moderation tools
- Admin dashboards for content oversight
- AI-based content filtering for offensive or inappropriate media

These tools will ensure a safe and respectful environment for all users.

9.6 Scalability and Deployment

Moving to cloud-native infrastructure (e.g., AWS, Azure) and using microservices can help scale the application for a large user base. This would ensure high availability, load balancing, and improved performance under high traffic.

9.7 Accessibility and Localization

To reach a broader audience, the app can support:

- Multilingual interfaces
- Accessibility features (e.g., screen readers, high-contrast themes)
- Regional content filters based on user location

9.8 Web Version

Developing a responsive web application would allow users to interact with the platform through browsers, increasing platform accessibility and adoption.

9.9 Integration with Wearables and IoT

Future updates can explore the use of wearable devices (e.g., smartwatches) for fitness or outdoor challenges, expanding the range of challenge categories and making the app more immersive.

Summary

With a solid foundational architecture in place, the Challenge-Based Social Media App has immense room for growth. Integrating advanced technologies, expanding platform reach, and improving user experience will allow it to evolve into a comprehensive, mainstream social engagement platform.

10. Conclusion

The development of the **Challenge-Based Social Media App** marks a significant step forward in the evolution of interactive social platforms. By combining elements of gamification, media sharing, and real-time communication, the app creates a unique environment for users to engage with one another through challenges, enhancing both creativity and social connection.

Through careful planning and implementation using **React Native**, **Node.js**, **Express.js**, **MongoDB**, and **Firebase**, the app provides a seamless experience across multiple platforms (Android and iOS). It offers an intuitive interface that promotes participation, whether through creating challenges, responding to them, or interacting with others via chat.

Key Findings:

- The app successfully meets its core objectives: facilitating challenge-based interactions, providing real-time communication, and supporting multimedia content sharing.
- The experimental results demonstrated strong performance, with fast media uploads, real-time chat synchronization, and smooth navigation across devices.
- Feedback from initial users confirmed high engagement levels, with users spending significant time interacting with challenges, browsing the feed, and participating in chats.

Challenges Encountered:

Despite its successes, the project faced challenges such as limited offline functionality, basic search features, and media upload constraints. Additionally, the push notification system and security features require further enhancement to improve overall user retention and safety.

Future Directions:

As highlighted in the previous section, there is substantial room for improvement and future expansion. Incorporating advanced features such as AI-based recommendations, enhanced push notifications, and more robust monetization strategies will significantly enhance the app's value proposition. Scalability improvements and the introduction of a web version will further increase the platform's accessibility and user base.

In conclusion, the **Challenge-Based Social Media App** lays the foundation for a dynamic and engaging platform that combines social interaction with creative content generation. With ongoing improvements and the integration of new technologies, it has the potential to evolve into a leading platform for user-generated challenges, social engagement, and multimedia content sharing.