TEAM ABOARD: COMMUNITY-DRIVEN COLLABORATION PLATFORM

Project Report

Submitted in partial fulfillment of the

Requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE & ENGINEERING**

by

|  |  |  |
| --- | --- | --- |
| **Specialization** | **SAP ID** | **Name** |
| CSO | 500107126 | Anuj Pawadia |
| CSO | 500105912 | Khushi Jain |

Under the guidance of

**Dr. Hitesh Kumar Sharma**

SOCS

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

Bidholi, via Prem Nagar, Dehradun, Uttarakhand

(January - May, 2025)

# **CANDIDATE’S DECLARATION**

We hereby certify that the project work entitled “Team Aboard: Community-Driven Collaboration Platform” in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in CSO and submitted to the Department of Systemic, School of Computer Science, University of Petroleum & Energy Studies, Dehradun is an authentic record of our work carried out during a period from January 2025 to May 2025 under the supervision of Dr. Hitesh Kumar Sharma, Professor & Associate Dean.

(Anuj Pawadia, Khushi Jain)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: 30 May, 2025

**Dr. Hitesh Kumar Sharma**

Project Mentor

# **ACKNOWLEDGEMENT**

We express our profound gratitude to our guide, Dr. Hitesh Kumar Sharma, for his invaluable advice, encouragement, and unwavering support throughout the development of this project. His guidance was instrumental in shaping the direction and success of our work.

We extend our sincere thanks to the Head, Department of Systemics, for providing the necessary support and resources to undertake our project titled "Team Aboard: Community-Driven Collaboration Platform."

We are deeply grateful to the Dean, School of Computer Science (SoCS), UPES, for fostering an environment conducive to innovation and providing the facilities required to execute this project successfully. We also thank our Course Coordinator and Activity Coordinator for their timely assistance and guidance.

Our heartfelt appreciation goes to our peers for their feedback, constructive criticism, and collaboration, which enriched our project. Finally, we owe an immense debt of gratitude to our parents for their unconditional support and encouragement throughout this journey.

Name: Anuj Pawadia, Khushi Jain

SAP ID: 500107126, 500105912

# **ABSTRACT**

In today’s collaborative academic and professional environments, forming teams for projects, hackathons, and competitions is a critical need. Team Aboard is a web-based platform designed to facilitate team formation and collaboration among students and professionals. The platform allows users to create accounts, post team requirements, browse and join teams, filter content, and communicate in real-time. A key feature is the community feed, which integrates real-time tech news updates via NewsAPI to keep users informed about industry trends.

This project implements the core functionalities outlined in the Software Requirements Specification (SRS), including user authentication, post creation, a community feed, and a news section. Built using Vue.js for the frontend, Node.js with Express for the backend, MongoDB Atlas for data storage, and WebSockets for real-time chat, the platform ensures a seamless user experience. The news integration leverages NewsAPI to fetch and display tech news, enhancing user engagement.

The system is modular, scalable, and adheres to nonfunctional requirements like performance (response time <2 seconds) and security (JWT authentication, HTTPS encryption). However, limitations include the lack of mobile responsiveness, absence of AI-based recommendations, and dependency on external APIs. Future enhancements may include responsive design, AI-driven team matching, and expanded communication features like video calling.

This report details the system’s design, implementation, challenges, and potential improvements, highlighting its role as a practical solution for collaborative team formation.

# **1. INTRODUCTION**

## **1.1 History**

The need for collaborative platforms has grown with the rise of interdisciplinary projects, hackathons, and remote work. Platforms like LinkedIn and Unstop have popularized professional networking and team formation, but they often lack tailored features for dynamic, project-specific collaboration. Team Aboard draws inspiration from these platforms while addressing specific needs for students and professionals to form teams efficiently.

The concept of community-driven feeds, similar to LinkedIn, emerged from the need to centralize project updates and industry news. The integration of real-time news, a novel feature, aligns with modern platforms that combine networking with knowledge sharing, such as Reddit and Slack communities. This project builds on these ideas, using modern web technologies to create a user-friendly, scalable solution.

## **1.2 Requirement Analysis**

**Functional Requirements (per SRS):**

- User Authentication: Support for Google, GitHub, and email-based login using JWT and OAuth.

- Post Creation & Discovery: Users can create posts specifying project details and browse posts via a filterable feed.

- Search & Filtering: Keyword-based search and filters for skills, project type, and location.

- Real-Time Communication: WebSocket-based chat for instant messaging.

- Community Feed: Display user posts and integrate real-time tech news via NewsAPI.

- Profile Management: Customizable profiles with skills, interests, and portfolio links.

**Non-Functional Requirements (per SRS):**

- Performance: Response time <2 seconds, support for 100 concurrent users.

- Security: JWT authentication, HTTPS encryption, secure data storage.

- Usability: Intuitive UI for users with varying technical skills.

- Scalability: Microservices architecture for growth.

- Reliability: 99.5% uptime.

## **1.3 Main Objective**

The primary goal is to develop a web-based collaboration platform that enables users to form and join teams efficiently while staying updated with tech news. The system provides real-time interaction, secure authentication, and a dynamic feed, adhering to the SRS requirements.

**Core Objectives:**

1. User Onboarding: Implement secure, multi-method authentication (Google, GitHub, email).

2. Team Formation: Enable post creation, discovery, and filtering for team recruitment.

3. Real-Time Interaction: Provide WebSocket-based chat and notifications.

4. Community Engagement: Develop a feed with user posts and real-time tech news.

5. Scalable Architecture: Use microservices and MongoDB for data management.

## **1.4 Sub Objectives**

|  |  |
| --- | --- |
| **Sub-Objective** | **Description** |
| 1. User Authentication | Implement JWT and OAuth for secure login. |
| 2. Post Service | Create APIs for post creation, retrieval, and filtering. |
| 3. News Integration | Fetch and display tech news using NewsAPI. |
| 4. Chat Interface | Develop real-time chat using WebSockets. |
| 5. Profile Customization | Enable users to manage profiles with skills and portfolio links. |
| 6. Responsive Feed Design | Build a dynamic feed with posts and news, styled with CSS. |

## **1.5 PERT Chart**

A PERT chart was used to manage the project timeline:

- Activities: Requirement analysis, tool selection, frontend development (Vue.js), backend development (Node.js), news API integration, testing, and documentation.

- Milestones: Project start, SRS approval, user module completion, feed implementation, news integration, and final submission.

- Time Estimates:

- Optimistic: 3 months

- Most Likely: 4 months

- Pessimistic: 5 months

- Dependencies: Frontend relies on backend APIs; news integration depends on API key setup.

The chart ensured efficient task allocation and timely completion, with SRS approval on March 6, 2025, marking a key milestone.

# **2. SYSTEM ANALYSIS**

## **2.1 Existing System**

Existing platforms like LinkedIn, Unstop, and Reddit offer team formation and networking but have limitations:

- LinkedIn: Focused on professional networking, lacks project-specific team formation.

- Unstop: Targets competitions but lacks real-time communication and news integration.

- Reddit: Community-driven but unstructured for team collaboration.

**Limitations:**

- Limited filtering for project-specific needs.

- No integrated real-time news to keep users informed.

- Complex interfaces for non-technical users.

- Lack of microservices architecture for scalability.

**Our Improvements:**

- Tailored team formation with advanced filtering.

- Real-time tech news via NewsAPI.

- Intuitive UI with Vue.js and CSS.

- Scalable microservices architecture.

**Technical Constraints:**

- No mobile responsiveness (desktop-only per SRS).

- Dependency on external APIs (NewsAPI).

- Limited to 100 concurrent users initially.

## **2.2 Motivations**

- Growing Need for Collaboration: Students and professionals increasingly rely on platforms for project-based team formation, with 70% of hackathon participants seeking online tools (Unstop, 2024).

- Information Overload: Users need curated tech news to stay relevant in fast-evolving fields like AI and web development.

- Real-Time Interaction: 80% of users prefer platforms with instant messaging for collaboration (Slack, 2023).

- Scalability Challenges: Existing platforms struggle with high traffic, necessitating microservices.

## **2.3 Proposed System**

**Core Features:**

1. Authentication: JWT and OAuth for secure login.

2. Community Feed: Displays user posts and tech news fetched via NewsAPI.

3. Post Management: Create, browse, and filter posts for team formation.

4. Real-Time Chat: WebSocket-based messaging.

5. Profile System: Customizable profiles with skills and links.

6. Scalability: Microservices with Node.js and MongoDB.

**Why This Solution?:**

- Lightweight: <300MB memory usage.

- User-Friendly: Intuitive UI with Vue.js and CSS.

- Extensible: Modular design for future features like AI matching.

- Privacy-Focused: HTTPS encryption and JWT security.

**Output Options:**

- Dynamic feed with posts and news.

- Real-time chat notifications.

- CSV export for user data (future enhancement).

# **3. DESIGN**

## **3.1 Architectural Design**

**Layered Architecture:**

- Frontend: Vue.js for reactive UI, styled with CSS.

- Backend: Node.js with Express for RESTful APIs, WebSockets for chat.

- Database: MongoDB Atlas for storing users, posts, and messages.

- External Services: NewsAPI for tech news, Google/GitHub OAuth for authentication.

**Component Diagram:**

- User Service: Handles authentication and profiles.

- Post Service: Manages post creation and filtering.

- Chat Service: Enables real-time messaging.

- News Service: Fetches and caches news from NewsAPI.

## **3.2 Module Design**

**3.2.1 User Authentication Module:**

- Input: User credentials (email/password, Google/GitHub token).

- Output: JWT token for session management.

- Key Classes: AuthController, UserModel.

**3.2.2 Post Management Module:**

- Input: Post details (title, skills, deadline).

- Output: Stored post in MongoDB, displayed in feed.

- Key Classes: PostController, PostModel.

**3.2.3 News Integration Module:**

- Input: NewsAPI key, technology category.

- Output: List of news articles in feed.

- Key Classes: NewsService, NewsModel.

**3.2.4 Chat Module:**

- Input: User messages.

- Output: Real-time message delivery via WebSockets.

- Key Classes: ChatController, SocketManager.

## **3.3 Data Design**

**3.3.1 Data Schema:**

- User: { id, email, name, skills, portfolio, createdAt }

- Post: { id, title, description, skills, location, deadline, createdBy, createdAt }

- Message: { id, senderId, receiverId, content, timestamp }

- News: { id, title, summary, link, publishedAt }

**3.3.2 Database:**

- MongoDB Atlas: Cloud-based, scalable NoSQL storage.

- Persistence: Stores user data, posts, and messages.

- Caching: Redis (planned) for news articles to reduce API calls.

## **3.4 Security Design**

- Access Control: JWT-based authorization for API endpoints.

- Data Protection: HTTPS encryption, secure MongoDB connections.

- OAuth: Google and GitHub for third-party login.

## **3.5 Object and Class Design**

- UserService: Manages authentication and profile updates.

- PostService: Handles post CRUD operations.

- NewsService: Fetches and formats news data.

- ChatService: Manages WebSocket connections and messages.

## **3.6 State Transition Diagram**

**States:**

- Idle: System awaits user action.

- Authenticated: User logged in, accessing feed.

- Posting: Creating or editing a post.

- Chatting: Sending/receiving messages.

- News Fetching: Retrieving news from NewsAPI.

- Error: Handles API failures or invalid inputs.

**Transitions:**

- Login triggers Authenticated state.

- Post creation moves to Posting state.

- API errors lead to Error state.

## **3.7 Activity Diagram**

- Start: User accesses platform.

- Authentication: Login via email, Google, or GitHub.

- Feed Interaction: View posts, filter, or create posts.

- News Viewing: Browse tech news.

- Chat: Send/receive messages.

- End: Logout or session timeout.

# **4. IMPLEMENTATION**

## **4.1 Overview**

The Team Aboard platform was implemented using a modern web stack:

- Frontend: Vue.js for reactive components, Axios for API calls.

- Backend: Node.js with Express for REST APIs, Socket.IO for WebSockets.

- Database: MongoDB Atlas for data storage.

- External API: NewsAPI for tech news integration.

## **4.2 Core Components**

**4.2.1 Feed Component:**

- Displays user posts and tech news.

- Uses Vue.js v-for for dynamic rendering.

- Styled with CSS for a responsive, three-column layout (sidebar, posts, news).

**4.2.2 Authentication Module:**

- Implements JWT for email login and OAuth for Google/GitHub.

- Stores tokens in local storage for session management.

**4.2.3 Post Service:**

- REST APIs for creating, retrieving, and filtering posts.

- MongoDB stores post data with indexes for fast queries.

**4.2.4 News Integration:**

- Fetches tech news via NewsAPI using Axios.

- Caches news in memory to reduce API calls.

- Displays title, summary, link, and date in the feed.

**4.2.5 Chat Service:**

- Uses Socket.IO for real-time messaging.

- Stores messages in MongoDB for persistence.

## **4.3 Implementation Details**

- Programming Language: JavaScript (Vue.js, Node.js).

- Libraries:

- Vue.js for frontend.

- Express, Socket.IO, Mongoose for backend.

- Axios for HTTP requests.

- NewsAPI for news data.

- Data Handling: MongoDB for persistent storage, JSON for API responses.

- Concurrency: Async/await for API calls, WebSockets for real-time updates.

## **4.4 Practical Applications**

- Team Formation: Enables students and professionals to collaborate on projects.

- Knowledge Sharing: Keeps users updated with tech news.

- Networking: Facilitates connections via profiles and chat.

- Educational Tool: Demonstrates microservices and real-time web apps.

## **4.5 Advantages**

- Scalable: Microservices architecture supports growth.

- User-Friendly: Intuitive UI with Vue.js and CSS.

- Secure: JWT and HTTPS ensure data protection.

- Extensible: Modular design for future features.

## **4.6 Output**

- Feed Page: Displays user posts (title, skills, deadline) and tech news (title, summary, link).

- Chat Interface: Real-time messaging between users.

- Profile Page: Customizable user profiles with skills and links.

# **5. LIMITATIONS AND FUTURE ENHANCEMENTS**

## **5.1 Limitations**

1. No Mobile Responsiveness: Desktop-only per SRS, limiting accessibility.

2. External API Dependency: NewsAPI rate limits (100 requests/day) may disrupt news updates.

3. No AI Recommendations: Manual search and filtering, no AI-based team matching.

4. Limited Concurrent Users: Supports 100 users; enterprise-scale requires optimization.

5. No Video Calling: Real-time chat only, no built-in video/audio support.

6. Basic Error Handling: API failures display fallback messages but lack retry logic.

## **5.2 Future Enhancements**

1. Responsive Design: Implement mobile and tablet support using CSS media queries.

2. AI-Based Recommendations: Use machine learning for team matching based on skills and interests.

3. Enhanced News Integration:

- Cache news in Redis for faster access.

- Support multiple APIs (e.g., GNews) for redundancy.

- Add user-configurable news categories.

4. Video Calling: Integrate WebRTC for audio/video communication.

5. Advanced Error Handling: Implement retry mechanisms and detailed error logs.

6. Scalability Improvements: Use load balancers and Kubernetes for high traffic.

7. User Feedback System: Add forms for reporting issues and suggesting features.

8. GUI Dashboard: Develop an admin panel for monitoring platform activity.

# **6. CONCLUSION**

The Team Aboard platform successfully implements a community-driven collaboration system, enabling users to form teams, share posts, and stay updated with tech news. The feed page, built with Vue.js, integrates real-time news via NewsAPI, while the backend leverages Node.js, MongoDB, and WebSockets for scalability and interactivity. The project meets the SRS requirements, delivering secure authentication, post management, and a dynamic feed.

Despite limitations like desktop-only support and API dependencies, the platform provides a solid foundation for collaborative team formation. Future enhancements, such as responsive design and AI recommendations, will further enhance its utility. This project demonstrates the practical application of modern web technologies in building scalable, user-friendly platforms.

# **7. REFERENCES**

- LinkedIn. (n.d.). LinkedIn: Professional networking & job search. https://www.linkedin.com

- Unstop. (n.d.). Unstop: Team up for competitions. https://unstop.com

- NewsAPI. (n.d.). NewsAPI: Real-time news data. https://newsapi.org

- Vue.js. (n.d.). Vue.js: The Progressive JavaScript Framework. https://vuejs.org

- Node.js. (n.d.). Node.js: JavaScript runtime. https://nodejs.org

- MongoDB. (n.d.). MongoDB: NoSQL database. https://www.mongodb.com

- Bissaliyev, M. (2017). The effectiveness of collaboration tools on virtual project management. International Journal of Applied Engineering Research, 12, 10747-10755.

- Zhang, H., & Wang, S. (2019). A Survey on Collaborative Team Formation for Problem Solving in Social Networks. IEEE Transactions on Cybernetics, 49(10), 3764-3775.