Project Proposal Report on

Ek-Kaan - A Chat Web App

Submitted in Partial Fulfillment of the Requirement for

the Degree of **Bachelors in Software Engineering**

Under Pokhara University

Submitted by:

Drishya Neupane, 201648

Sandilya Silwal, 201636

Shahil Dhakal, 201631

Date:

11th July, 2024

Department of Software Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

ABSTRACT

This project proposes the development of a modern chatting web application utilizing the MERN (MongoDB, Express.js, React.js, Node.js) stack for robust backend and frontend architecture. Tailwind CSS and DaisyUI Library will be employed for responsive and aesthetically pleasing user interfaces.

The application aims to provide a seamless chatting experience with features including real-time messaging facilitated by WebSocket technology, user authentication, and message encryption for security. Users will be able to create profiles and engage in private or group conversations. The integration of MongoDB ensures scalable data storage while Express.js and Node.js facilitate efficient server-side operations. React.js, coupled with Tailwind CSS and DaisyUI, enhances frontend development by offering component-based architecture and customizable styling.

This project targets users seeking a modern, responsive, and secure platform for communication.

Keywords: *MERN stack, MongoDB, Express.js, React.js, Node.js, WebSocket, real-time messaging, user authentication, message encryption, chat rooms, private conversations, group conversations, scalable data storage, frontend development.*

Table Of Contents

List of Figures

1. Introduction

Efficient and reliable messaging platforms are essential in today's digital landscape. In response to this need, we propose the creation of a chat web application utilizing the MERN stack (MongoDB, Express.js, React.js, and Node.js).

The proposed chat application will provide users with a robust platform for real-time messaging, ensuring a responsive and dynamic user experience. By leveraging the MERN stack, we aim to develop a scalable and maintainable application that supports essential features for modern communication. These features include user authentication, message history, group chats, multimedia sharing, and notifications. Real-time communication will be facilitated using WebSocket, ensuring instantaneous message delivery and updates.

Our project will prioritize creating an intuitive and user-friendly interface, allowing users to easily navigate and utilize the chat functionalities. Tailwind CSS will be employed to design a clean and responsive interface.

The application will emphasize performance, security, and data integrity, with the MongoDB database being hosted in the cloud to ensure reliability and scalability. This chat web application will serve as a versatile tool for individuals and organizations, facilitating smooth and efficient communication across various contexts.

2. Problem Statement

In today's fast-paced educational and professional environments, communication often feels like it's lagging behind. Current messaging platforms struggle to keep up with our need for quick responses and a secure space to exchange ideas.

Educators, students, and professionals find themselves frustrated by tools that don't offer seamless group chats or reliable ways to track past conversations. This leads to missed messages, confusion over who said what, and delays in important decisions.

The scattered nature of communication tools also makes it hard to stay organized. Important messages get lost in a sea of notifications, making it challenging to find what you need when you need it. This inefficiency not only slows down work and learning but also raises concerns about privacy and data security.

To solve these challenges, we envision a chat web application that's not just fast and reliable but also intuitive and secure. It should support instant messaging with robust features for managing conversations, secure file sharing, and easy access to message history. By creating such a tool, we aim to streamline communication for educators, students, and professionals alike, fostering smoother collaboration and more productive interactions in both educational and organizational settings.

3. Project Objectives

The primary aim of this project is to develop a web application that offers intuitive real-time messaging, robust security features, and a user-friendly interface. Key objectives include implementing reliable message history management. By prioritizing scalability, reliability, and enhanced collaboration, our application aims to empower educators, students, and professionals with a tool that enhances productivity and communication efficiency.

- **Real-Time Messaging:** Develop a chat web application that supports seamless, real-time communication to facilitate instant messaging among users.Implement features that allow instant messaging, notifications, and updates in real-time, enhancing user engagement and interaction.
- **Security and Privacy:** Ensure high-level security standards by implementing strong authentication mechanisms and encrypted data transmission using Node.js and also focus on securing WebSocket connections to protect user privacy and maintain data integrity, adhering to best practices in web application security.
- **Comprehensive Message Management:** Provide robust features for managing messages, including reliable history management using MongoDB's flexible document-based storage and query capabilities, and efficient search capabilities using Node.js.
- **Reliable Message History Management:** Provide robust features for managing message history to facilitate efficient information retrieval and review.
- **Scalability and Reliability:** Build a scalable architecture utilizing cloud-based MongoDB Atlas to handle increasing user demands reliably. Implement features that support scalability without compromising performance, ensuring the application remains responsive under varying loads.

4. Significance of the Study

This study aims to develop a web application that enables real-time messaging, providing significance as:

- **Enhanced Communication Efficiency:** The application aims to revolutionize communication by offering real-time messaging capabilities that facilitate instant communication and collaboration among users. This feature is crucial for enhancing productivity and responsiveness in educational and professional environments.
- **Heightened Security Standards:** Implementing strong security measures, including robust authentication and encrypted data transmission, ensures user privacy and data integrity. These measures are essential in safeguarding sensitive information exchanged within the application.
- **Improved Message Management:** By providing comprehensive message management features, including reliable history management and efficient search capabilities, the application enhances efficiency in information retrieval and review processes. This capability supports seamless communication and decision-making.
- **Scalability and Reliability:** Built on a scalable architecture leveraging cloud-based MongoDB Atlas, the application accommodates increasing user demands without compromising performance. This scalability ensures the application remains responsive and reliable under varying workloads, supporting its long-term usability.
- **Facilitation of Collaboration:** The application fosters effective collaboration by enabling clear communication and information exchange within educational and professional contexts. It enhances teamwork and productivity through intuitive tools that streamline collaborative efforts.
- **Accessibility and User-Friendliness:** Designed with a user-centric approach, the application provides a user-friendly interface accessible to users of varying technical expertise. This accessibility promotes widespread adoption and usability across different platforms and environments.

5. Scope and Limitations

This project focuses on developing a robust chat web application using the MERN stack (MongoDB, Express.js, React.js, Node.js) to facilitate real-time messaging and collaboration among users. The scope includes:

- **Real-Time Messaging:** Implementing features for instant messaging, notifications, and updates to enable seamless communication.
- **Security Features:** Integrating strong authentication mechanisms and encrypted data transmission to ensure user privacy and data integrity.
- **Message Management:** Providing comprehensive tools for managing message history, including efficient search capabilities and reliable history management.
- **Scalability and Reliability:** Designing a scalable architecture using cloud-based MongoDB Atlas to support increasing user demands while maintaining performance.
- **User-Friendly Interface:** Developing an intuitive interface using React.js that enhances usability and accessibility for users of varying technical backgrounds.

Limitations:

- **Multimedia Sharing:** The application does not support multimedia sharing functionalities such as file uploads or video conferencing.
- **Group Chats:** While the application supports real-time messaging between individual users, it does not include features for group chat interactions for now.
- **Offline Functionality:** Limited offline functionality may impact user experience in environments with intermittent or no internet connectivity.
- **Browser Compatibility:** The application focuses on compatibility with modern web browsers and may not fully support older browser versions.
- **Complex Data Analysis:** Advanced data analytics beyond basic message management and user interactions are outside the current scope of this project.
- **Hardware Dependency:** The application's performance may vary depending on users' hardware capabilities, especially in resource-intensive tasks.

6. Literature Study/Review

WhatsApp

Methodology: WhatsApp (Maheshar et al., 2017, #) employs a client-server architecture using a combination of native mobile applications (iOS, Android) and a web client. It utilizes end-to-end encryption (E2EE) for secure messaging, powered by the Signal Protocol, ensuring messages are encrypted before leaving the sender's device and decrypted only on the recipient's device.

Key Features and Scope:

- **Real-Time Messaging:** Supports instant text messaging, voice messages, and voice and video calls.
- **Multimedia Sharing:** Allows users to share images, videos, documents, and location information.
- **Group Chats:** Facilitates group conversations with multimedia sharing capabilities.
- **Platform Compatibility:** Available on multiple platforms, including mobile devices and web browsers.

Limitations:

- **Third-Party Integrations:** Limited support for third-party integrations and plugins.
- **Business Solutions:** WhatsApp for Business provides additional features tailored for small and medium enterprises.

Slack

Methodology: Slack operates on a client-server architecture with a focus on real-time messaging and collaboration. It utilizes WebSocket technology for instant communication and API integrations for seamless connectivity with third-party applications and services.

Key Features and Scope:

- **Channels and Messaging:** Organizes communication into channels for teams, projects, and topics.
- **File Sharing:** Allows users to share files, images, and documents within channels and

direct messages.

- **Integration Platform:** Offers an extensive range of integrations with tools such as Google Drive, GitHub, and Jira.
- **Customization:** Customizable notifications, themes, and workflows enhance user experience and productivity.

Limitations:

- **Free Tier Restrictions:** Limits on message history and storage for free-tier users.
- **Security Features:** While Slack provides security controls, data encryption is not end-to-end by default.

7. Proposed Methodology

This part explains how we plan to develop our project, covering both the steps we'll follow to create the software and the technical setup we'll use.

7.1 Software development Life Cycle

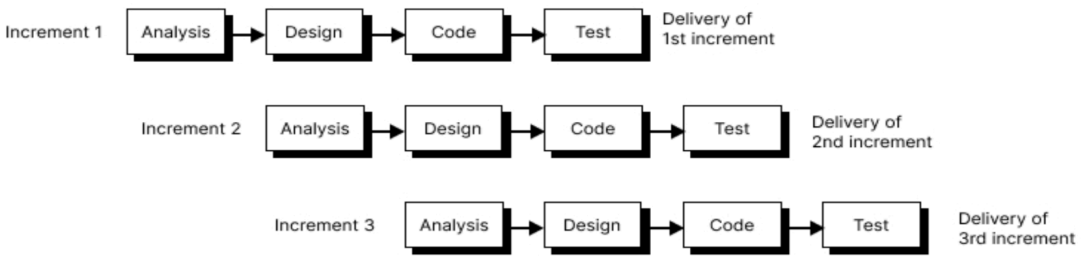

Fig 1: Incremental Model

We will follow an increment model for this project and work on the core features first then create the rest around it.The whole project can be divided into different stages. The major stages in the process includes:

Stage 1: Planning and Setup

- **Objective:** Lay the groundwork and plan for the development of the chat application.
- **Activities:**
    - Gather and document user requirements, functional specifications, and technical requirements.
    - Define the project scope, including features and timelines.
    - Set up a development environment and tools, including IDE, version control, and project management tools.
    - Plan database schema and backend architecture requirements.
    - Establish documentation standards and templates for ongoing project documentation.

Stage 2: Core Development

- **Objective:** Implement core features and functionalities of the chat application.

- **Activities:**
    - Develop user registration and authentication systems.
    - Implement real-time messaging using WebSocket technology.
    - Design and develop the user interface (UI) for sending and receiving messages.
    - Set up a cloud database structure for storing user data and message history.
    - Document codebase, APIs, and database schema for future reference and maintenance.

Stage 3: Testing, Deployment, and Improvement

- **Objective:** Ensure quality, deploy the application, and iterate for continuous improvement.
- **Activities:**
    - Conduct functional testing to verify the correctness of messaging features.
    - Perform compatibility testing across different browsers and devices.
    - Implement performance testing to optimize speed and responsiveness.
    - Gather feedback from users to identify areas for improvement.
    - Iterate based on feedback to enhance UI/UX, security, and performance.
    - Update documentation with changes made during testing and improvement phases.

This model also emphasizes the importance of documentation throughout the development lifecycle, ensuring clarity, maintainability, and supportability of the web-based chat application from planning through deployment and beyond.

8. Software Specifications

The web application aims to facilitate real-time messaging and collaboration among users. The other software components are:

Backend Technology Stack:

- **Programming Language:** JavaScript (Node.js) - Utilized for server-side scripting and backend development.
- **Framework:** Express.js - A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- **Database:** MongoDB (via MongoDB Atlas) - A scalable NoSQL database service designed for modern applications, offering flexibility and performance.

Frontend Technology Stack:

- **Framework/Library:** React.js - A JavaScript library for building user interfaces, enabling efficient rendering and seamless updates.
- **Styling:** Tailwind CSS - A utility-first CSS framework for rapid UI development, providing a customizable and responsive design approach.
- **UI/UX Design:** Figma - A collaborative interface design tool used for creating and prototyping user interfaces with intuitive design capabilities.
- **Build Tool:** Vite - A build tool that provides a faster and leaner development experience by serving source files over native ES modules and bundling code with Rollup for optimized deployment.

Development Tools:

- **IDE:** Visual Studio Code - A lightweight and powerful code editor with built-in support for debugging, syntax highlighting, and Git integration.
- **Version Control:** Git - A distributed version control system for tracking changes in source code during software development.
- **Diagramming:** PlantUML - A tool that allows users to create UML diagrams from plain text descriptions, enhancing documentation and system design clarity.

9. System Model And UML Diagram

This section introduces the project's system model and UML diagrams. The system model shows how the components interact with each other, and the UML diagrams use standardized symbols to give a visual overview. These tools help explain the project's structure, function, and how its parts are connected, making it easier to understand and communicate about the project.

9.1 Use Case Diagram

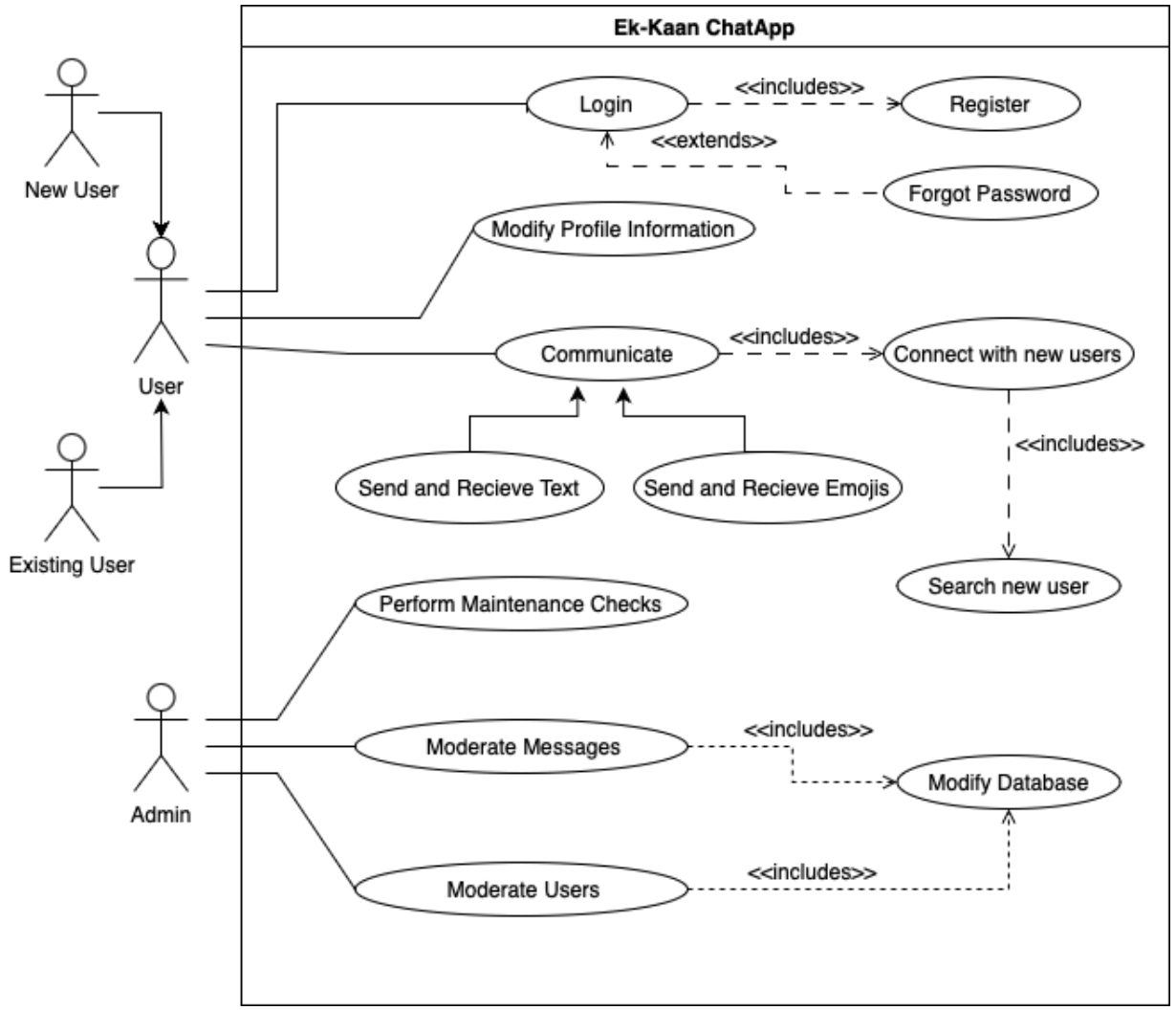

Fig 2: Use Case Diagram

9.2 Activity Diagram

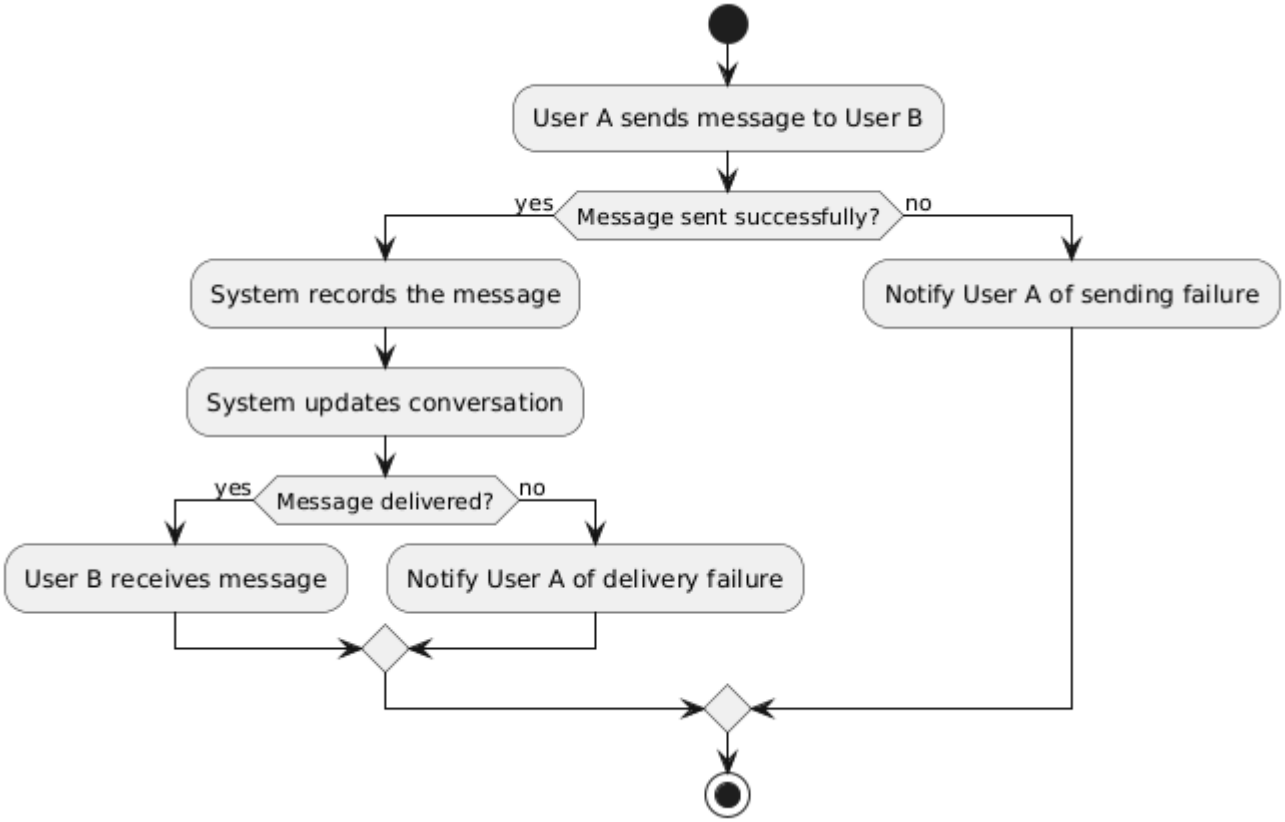

Fig 3: Activity Diagram

9.3 Entity Relationship (ER) Diagram

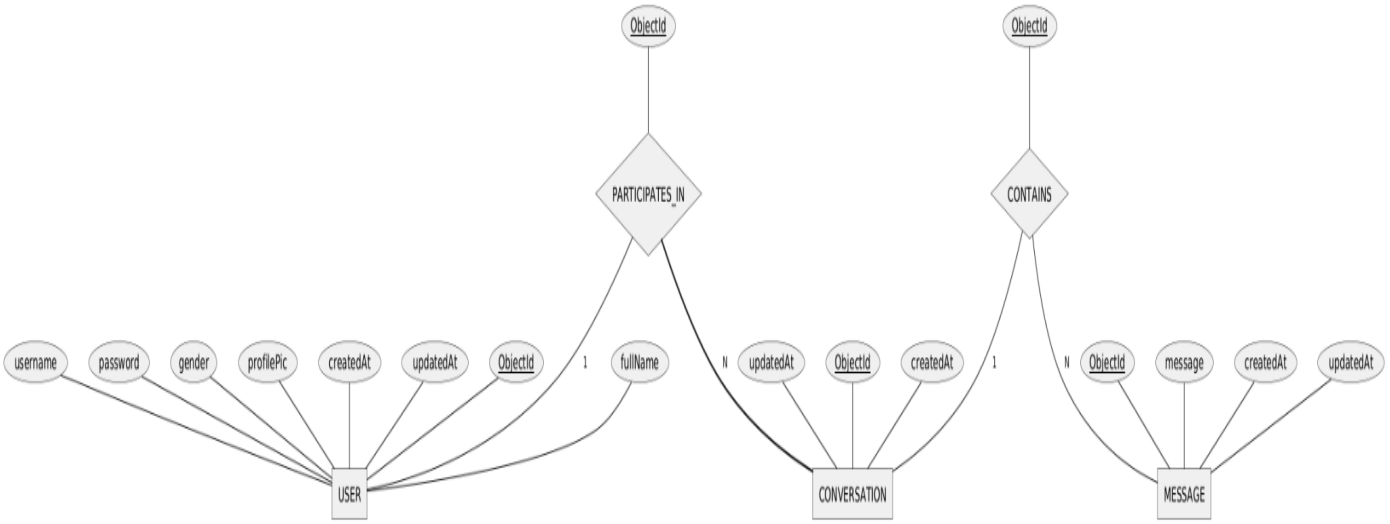

Fig 4: ER Diagram

9. Projected Output

At the end of the project, the following outcomes will be achieved:

- **Secure User Authentication and Authorization**: Develop a secure authentication system to ensure safe user logins and profile management, prioritizing data privacy and user account security.
- **Intuitive User Interface**: Design an intuitive and responsive user interface using React.js and Tailwind CSS, ensuring seamless navigation and an engaging user experience across various devices.
- **Comprehensive User Profile Management**: Enable users to register, create profiles, and manage account settings effortlessly, enhancing personalization and interaction within the platform.
- **Interactive Features**: Introduce interactive features such as emoji reactions, file attachments, and message editing to enrich communication dynamics and user interaction.
- **Search and Filtering Capabilities**: Implement advanced search and filtering options to allow users to efficiently find conversations, messages, or users based on specific criteria, streamlining user navigation and content discovery.
- **Notification and Messaging System**: Develop a notification system that alerts users about new messages, mentions, or updates, ensuring timely responses and improved user engagement.
- **Scalable Architecture**: Utilize Node.js backend and MongoDB Atlas for scalable database management, ensuring the application can accommodate increasing user activity and data storage demands effectively.
- **Comprehensive Documentation**: Provide detailed documentation covering system architecture, API endpoints, and user guides to facilitate smooth deployment, maintenance, and future enhancements.

.

10. Project Task and Time Schedule

The following table and Gantt chart show how project tasks are distributed, scheduled, and assigned to team members:

10.1 Tasks

Shahil Dhakal	API, Database and Documentation
Drishya Neupane	UI/UX Design, Backend and Documentation
Sandilya Silwal	Frontend, Graphics Designing and Documentation

10.2 Gantt Chart

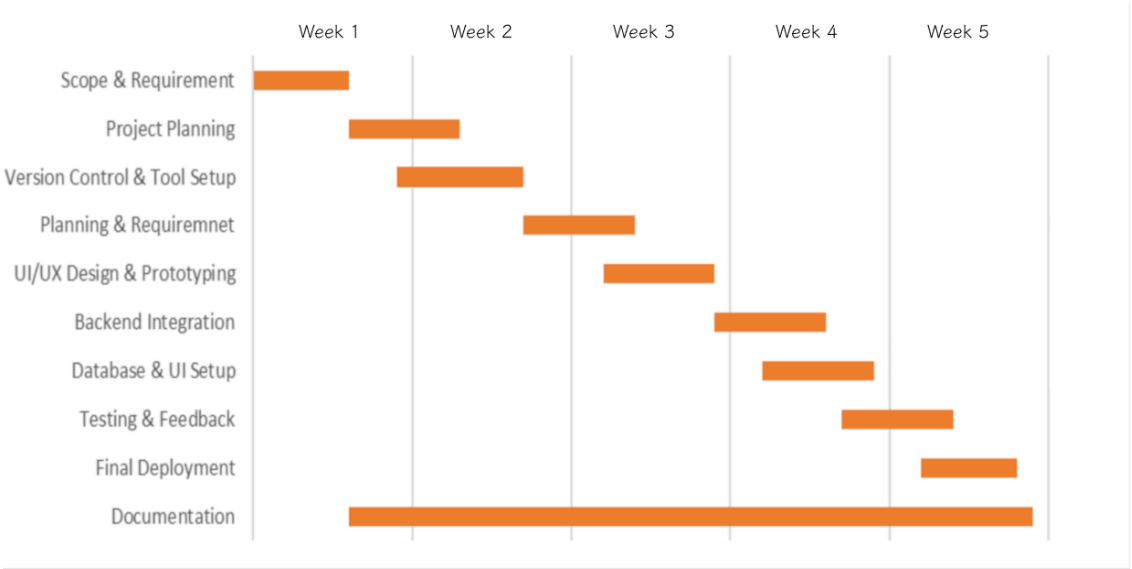

Fig 5: Gantt Chart

References

- Johnson, H. A. (2018). *Slack* [Slack is a cloud-based digital workspace and information management system used to manage productivity and improve team efficiency].
- Maheshar, R., Hussain, D. Z., & Shah, R. H. (2017). *WhatsApp usage frequency by university students: A case study of Sindh University*.
- Ontario Tech University & Ontario Tech University. (2013). *Exploring the Use of Text and Instant Messaging in Higher Education Classrooms*.