Tribhuvan University

Faculty of Humanities and Social Sciences

NEP CHAT USING CRON SCHEDULING

A PROJECT REPORT

Submitted To

Department of Computer Application

Aadim National College

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted By:

Bikash Khanal(105902069)

Jestha 2081

6th Semester

Under the supervision of

Er. Lokesh Gupta

Abstract

The development of the real-time chat application was aimed at offering Android users a seamless and user-friendly communication platform. This project originated from the growing demand for secure, fast, and flexible messaging solutions that cater to both personal and corporate communication needs.

The application sought to address limitations found in traditional messaging platforms by introducing features such as scheduled message delivery and the ability to delete messages from both the sender's and receiver's ends. These enhancements were designed to give users more intuitive control over their conversations.

The primary objective of the project was to create a mobile application that allows users to easily find and connect with others, while ensuring a smooth and efficient messaging experience. Another key goal was to design an elegant yet straightforward user interface that enhances interaction without compromising usability.

To achieve these goals, React Native was used for frontend development, enabling cross-platform compatibility with native performance. The backend was built using Node.js and Express.js, while MongoDB served as the database to store user and message data. Real-time communication was implemented using Socket.IO.

The final product successfully met its objectives. It supported real-time messaging, scheduled message sending, and message deletion by either party. During testing, the application demonstrated high reliability and performance, confirming its readiness for real-world use.

Keywords: *Chat application, Android Users, React Native, Node.js, Express.js, MongoDB, Socket.IO, Messaging solutions, Schedule Message, Mobile Application, User interface,Cross-platform*

Acknowledgements

I acknowledge Aadim National College for giving me an opportunity to execute this project with other available resources. The support and environment provided by the institution have played a lead role in shaping the success of this work-full realization.

A special thank is deserved by my supervisor, Er. Lokesh Gupta, who has guided me throughout the project, given me valuable feedback, and unwavering encouragement. His insights and expertise have set the direction of this work and have fully affected its outcome.

I express my sincere thanks and gratitude to the Head of Department, Er. Prem Subedi, for his support and help in creating an environment in which learning could take place.

I must thank all of the IT staff for their cooperation-all the information and help they gave me throughout the realization of the project.

Finally, I am thankful to the friends, who from time to time gave me radiant inspiration and encouraging counsel together with support through thick and thin, which kept my eyes glued and motivated through the entire completion of the project.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Nep Chat is an innovative messaging platform designed to provide users with a secure, efficient, and feature-rich communication experience. Its simplicity and versatility make it ideal for individuals, businesses, and communities seeking seamless and secure communication. This app control over personal conversations.

By combining powerful security measures with user-centric innovations, Nep Chat redefines the messaging experience. Its simplicity and versatility make it accessible for everyday users, while its privacy-focused features offer businesses a secure platform for collaboration. Whether you need safe personal messaging, secure business discussions, or a reliable chat platform, Nep Chat provides a seamless and protected communication environment. With its focus on privacy, flexibility, and innovation, Nep Chat is setting a new standard for modern messaging platforms.

1.2 Problem Statement

Without a Chat that addresses key privacy, communication, and users face several challenges. The lack of end-to-end encryption leaves sensitive information vulnerable to breaches. Users may also find it difficult to manage their chat histories, making it harder to protect private conversations.

Some of the major problems are:

- Lack of Privacy: Without strong encryption, conversations are exposed to potential data breaches.
- Connectivity Issues: Without users in areas with poor internet coverage cannot communicate.
- Risk of Data Misuse: Uncontrolled message retention increases the risk of sensitive information being misused.
- Disconnected Conversations: Poor connectivity can cause delays in sending or receiving messages, affecting real-time communication.
- Frustration Over Delays: Delays in sending messages due to poor connectivity lead to frustration and broken conversations.
- No Security Guarantees: Users may hesitate to share confidential information, fearing a lack of protection.

1.3 Objectives

The main objectives of this project can be enumerated as follows:

- To create a user-friendly android platform where users can add others users and can message with each other.
- To integrate features like scheduled messaging enabling users to plan and manage their communication effectively.
- To allow users to delete messages from both ends.

1.4 Scope and Limitation

1.4.1 Scope

- The project will allow users to send and receive real-time messages securely
- The application will support features such as one-on-one messaging, message scheduling.
- Users can view and manager their respective chat histories.

1.4.2 Limitations

- There is no push notifications to notify users about new messges and requests
- There is no feature for group chats

1.5 Development Methodology

For the methodology of the proposed system, I am using the Agile methodology. Agile focuses on breaking the development process into small, manageable tasks called sprints (usually 2-4 weeks long). Each sprint results in a working part of the application that I can improve based on feedback. Agile is flexible and adaptable, meaning I can easily adjust to changing requirements. This approach allows me to deliver parts of the project faster and continuously refine the system as I go. By using Agile, I can quickly adapt, get regular feedback, and provide working software more frequently.

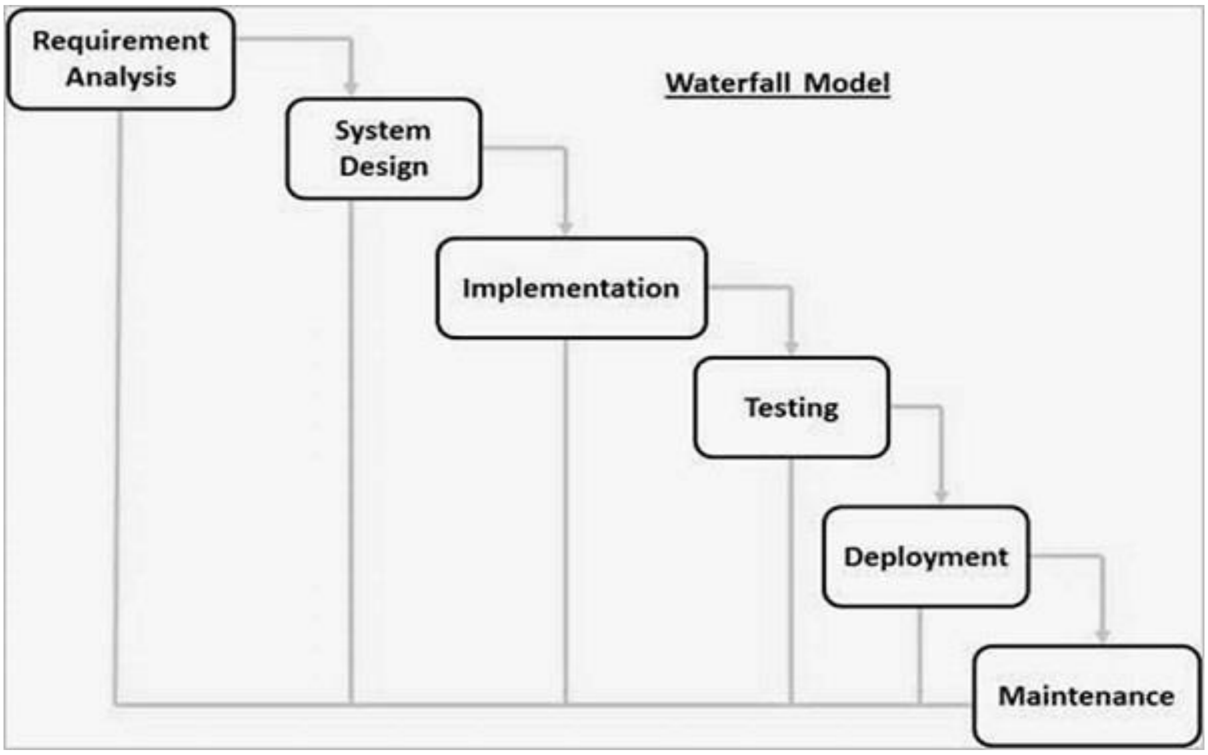

Figure 1.1 Waterfall Methodology

1.6 Report Organization

The first part of the report contains the summarized introduction of the whole report. It includes the overview, scope and limitation, problem statement and objectives of this project.

The second chapter includes background study i.e., description of fundamental theories, general concepts and terminology related to the project. It also includes the literature review i.e., review of the similar projects, research and theories done by other researchers.

The third chapter includes the system analysis and design phase in which the report of functional and non-functional requirements of the project is stated using use case and system diagrams. It also includes the feasibility study about the system which explains whether the system development process is affordable and within the knowledge range of the developers. It shows the technical, operational and economic feasibility of the project development phase. The explanation of the designing of the system is also done in this chapter. It includes data modeling and process modeling which is explained by using ER diagram and Data Flow Diagram. The architectural design, database design and the user interface design are also listed in this chapter.

Chapter 2: Background Study and Literature Review

2.1 Background Study

In today's digital era, communication is central to both personal and professional interactions. Traditional methods of communication such as SMS, emails, and phone calls have been supplemented—and in many cases replaced—by instant messaging platforms due to their speed, convenience, and multimedia capabilities.

Chat applications have grown rapidly, driven by the widespread availability of the internet, smartphones, and cloud-based technologies. They enable real-time exchange of messages, voice notes, images, and videos, supporting both individual and group conversations. Popular applications like WhatsApp, Messenger, Slack, and Telegram have demonstrated how messaging tools can enhance collaboration, productivity, and social engagement.

The need for a secure, efficient, and reliable communication platform has led to the development of custom chat applications tailored for specific user bases, such as internal workplace communication, educational environments, or customer support systems. These systems often offer features such as user authentication, contact management, media sharing, and administrative control, and can be extended to include end-to-end encryption and voice/video calling.

This project aims to design and implement a chat application that addresses the need for real-time communication within a defined user community. It focuses on delivering core messaging functionalities along with essential user management and security features.

2.2 Literature Review

The development of chat applications has evolved significantly, with various platforms offering real-time communication, multimedia sharing, and security features. This literature review examines key existing messaging applications, including WeChat, Viber, WhatsApp, Telegram, and Signal highlighting their features, strengths, and limitations. The insights gained from these platforms help in identifying gaps that Nep Chat aims to address.

1. WeChat

Overview:

WeChat, developed by Tencent, is one of the most widely used messaging applications in China. It offers text messaging, voice and video calls, social media integration, and mobile payments.

Strengths:

- Multi-purpose functionality integrating chat, social media, and financial transactions.
- Mini-programs that allow businesses to build services within the app.
- Strong integration with Chinese services, making it a super app.

Limitations:

- Concerns over privacy and government surveillance.
- Limited availability outside China due to strict regulations and restrictions.
- Heavy data consumption and resource usage.

2. Viber

Overview:

Viber is a messaging and VoIP application known for its high-quality voice and video calls. It provides **end-to-end encryption** for private chats and group conversations.

Strengths:

- Secure communication with encryption for individual and group chats.
- High-quality voice and video calls.
- Public communities and chat extensions for businesses.

Limitations:

- Limited user base compared to competitors like WhatsApp and Telegram.
- Higher battery and data consumption in prolonged usage.
- Relies on a centralized server model, which poses privacy concerns.

3. WhatsApp

Overview:

WhatsApp, owned by Meta, is one of the most popular messaging apps globally. It provides instant messaging, voice calls, video calls, and media sharing with end-to-end encryption.

Strengths:

- Simple and user-friendly interface.
- End-to-end encryption for private conversations.
- Widespread adoption, making it easier to connect with users worldwide.

Limitations:

- Owned by Meta, leading to concerns about data privacy and metadata collection.
- Limited customization options for users.

- Fully dependent on centralized servers, making it vulnerable to downtimes and cyberattacks.

4. Telegram

Overview:

Telegram is a cloud-based messaging platform known for its speed, large group capacities, and optional encryption (via Secret Chats).

Strengths:

- Supports large groups and channels with up to 200,000 members.
- Cloud-based architecture, allowing message access from multiple devices.
- Bot integration for automation and enhanced user experience.

Limitations:

- Default chats are not end-to-end encrypted, making them less secure.
- Requires a phone number for registration, raising privacy concerns.
- Can be blocked in certain countries due to its encryption policies.

Chapter 3: System Analysis and Design

3.1 System Analysis

This chapter discusses about the proposed model of the system, the methodology used in building the system, tools and techniques, requirement analysis, requirement specification. It also talks about the functional requirements of the system, the system design. The application architecture of the system will also be shown in this chapter, the use case diagram, data design, activity diagrams, dataflow diagram, control flow diagram, entity-relationship diagram (ERD) and user interface design will all be shown in this chapter.

3.1.1 Requirement Analysis

The following section presents the complete set of functional requirements

i. Functional Requirements

Following figure shows the requirements

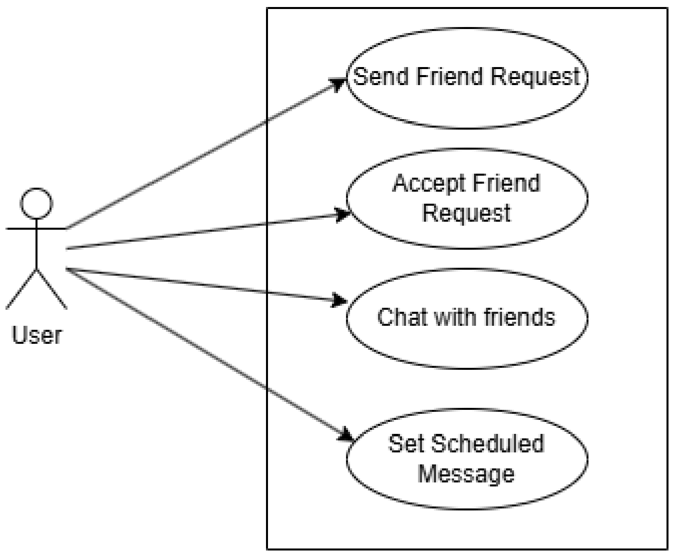

Figure 3.1 Usecase Diagram

Table 3.1 Functional Requirements

Req. No.	**Description**	**Type**
R-101	The android application will contain user interface.	Functional
R-102	The android application will require internet to operate.	Configuration

R-103	The android application will allow navigating menus.	Functional
R-104	The android applications will allow users to add and accept friends.	Functional
R-105	The android application will allow users to chat with each other and set scheduled messages	Functional

ii. Non-functional Requirements

Some of the contents of non-functional requirements are shown table below.

Table 3.2 Non-functional requirements

Req. No.	Description	Type
NR-101	The android application shall ensure sensitive information is secure	Security
NR-102	The android application will be user friendly	Usability
NR-103	Unauthorized users will not be able to access the system.	Security
NR-104	The android application shall run well on desktop and mobile devices	Configuration

3.1.2 Feasibility Analysis

Some important feasibility studies are mentioned below:

i. Technical Feasibility

It is technically feasible as I already have hardware and software required for development of a software. Also, I have technical knowledge on how the project is made through programming language.

ii. Operational Feasibility

It is operationally feasible as I am making this system by removing the threats and weakness of existing non manageable system which is reliable for the users.

iii. Economic Feasibility

The system does not require extra software and hardware. So, there is no recurring cost than just the internet connection.

3.1.3 Data Modeling (ER Diagram)

Since this system uses NoSql Database, there will be no ER Diagram

3.2 System Design

3.2.1 Architecture Design

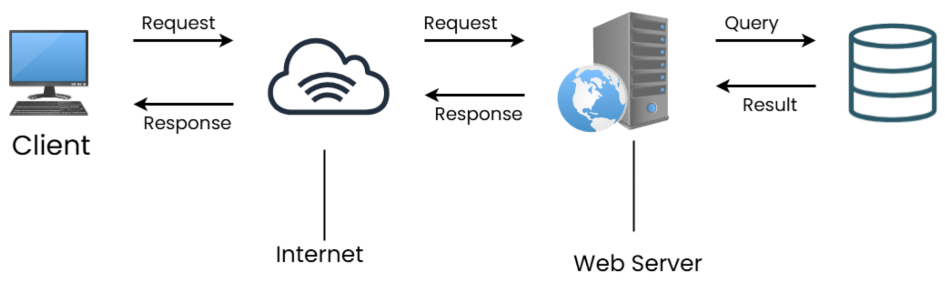

Fig 3.2 Architecture Design

3.2.2 Database Schema Design

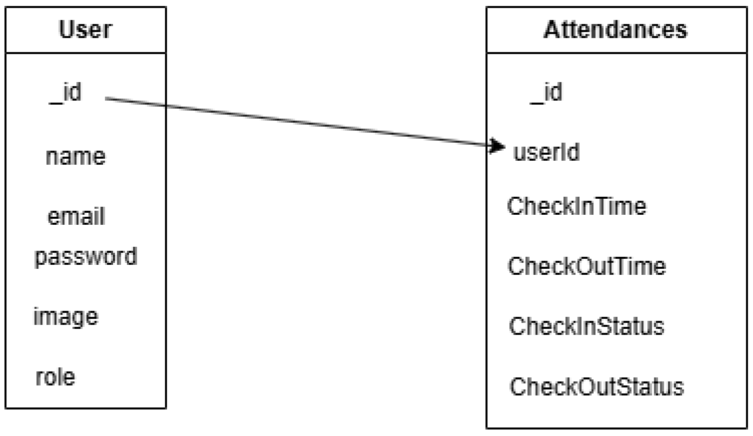

Fig 3.3 Database Schema Diagram

3.3 Algorithm Details

A cron scheduling algorithm is used to determine the next time(s) a job should run based on a cron expression. Cron expressions are widely used in Unix-based systems (like Linux) and scheduling tools (like Kubernetes, AWS CloudWatch, etc.) to define time-based job execution.

What is a Cron Expression?

A standard cron expression is made up of 5 fields (or 6 with seconds), typically in this order:

```
# ┌───────────── minute (0 - 59)
# │ ┌───────────── hour (0 - 23)
# │ │ ┌──────────────────── day of the month (1 - 31)
# │ │ │ ┌──────────────────── month (1 - 12)
# │ │ │ │ ┌──────────────────── day of the week (0 - 6) (Sunday=0 or 7)
# │ │ │ │ │
# │ │ │ │ │
# * * * * *
```

Example:

0 14 * * 1-5 → Every weekday at 2:00 PM

How the Cron Scheduling Algorithm Works

The cron scheduling algorithm follows these steps:

1. **Parse the Expression**

Break the cron expression into its individual fields (minute, hour, day, etc.), and convert wildcards (*), ranges (1-5), and lists (1,3,5) into concrete sets of values.

2. **Current Time Initialization**

Start with the current time (now) and incrementally check the next possible time that satisfies all fields in the cron expression.

3. **Field Matching**

From left to right:

- Match minute, if not matched, go to next minute.
- Match hour, if not matched, go to next hour (reset minute).
- Match day of month/week, if not matched, go to next day (reset hour and minute).
- Match month, if not matched, go to next month (reset day, hour, minute).
- Repeat until a valid date-time match is found.

4. **Return Next Matching Time**

Once all fields match, that's your next scheduled run time.

Chapter 4: Implementation and Testing

4.1 Implementation

The aim of this chapter is to document the process of development of the main features. It gives a detailed breakdown of the problems encountered and how they were resolved. It also goes through the test plan and test report of the project to ensure all the functionalities are functioning properly. This chapter is where the project is going to be implemented. However, the software and hardware components used in the implementation of this project will be analyzed below.

4.1.1 Tools Used

React Native

React Native is an open-source mobile application framework developed by Facebook. It allows developers to build natively rendered mobile applications for iOS and Android using JavaScript and React, a popular JavaScript library for building user interfaces. Unlike traditional hybrid app frameworks, React Native renders components using native APIs, resulting in better performance and user experience. One of the major advantages of React Native is code reusability, allowing developers to write a single codebase that works across multiple platforms. It also supports features like hot reloading, modular architecture, and strong community support, making it an ideal choice for mobile app development.

Node.js and Express.js

Node.js is an open-source, cross-platform, JavaScript runtime environment built on Chrome's V8 JavaScript engine. It is designed for building scalable and efficient server-side applications. Node.js uses an event-driven, non-blocking I/O model, making it lightweight and perfect for data-intensive real-time applications.

Express.js is a minimalist and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It simplifies the development process by handling common backend functionalities such as routing, middleware integration, request handling, and session management. Express is widely used for creating RESTful APIs, which serve as a communication layer between the client (e.g., a React Native app) and the database.

MongoDB

MongoDB is a popular NoSQL database that stores data in a flexible, JSON-like format called BSON (Binary JSON). Unlike traditional relational databases, MongoDB does not require predefined schemas, making it highly adaptable to changes in data structure. This flexibility is particularly useful for applications that deal with unstructured or semi-structured data.

MongoDB is designed for scalability and performance, supporting features like horizontal scaling, replication, and high availability. Its ability to handle large volumes of data and its compatibility with JavaScript-based environments like Node.js make it a preferred choice for modern full-stack applications.

Chapter 5: Conclusion and Future Recommendations

5.1 Conclusion

The chat application developed through this project successfully fulfills the outlined objectives. A user-friendly Android platform was created using React Native, allowing users to add each other and communicate seamlessly through real-time messaging. The integration of scheduled messaging provided users with the flexibility to manage their conversations more effectively, while the delete-for-both feature enhanced control over message privacy and content management. The system demonstrates strong performance and responsiveness, proving the feasibility of combining React Native with Node.js (Express) and MongoDB for building efficient, real-time communication tools.

5.2 Lesson Learnt/Outcome

While developing the project, I learned a lot about creating a modular system, which made the code easier to manage and test. I also realized how important it is to organize data efficiently, especially when using MongoDB to handle databases. Connecting the front-end (React Native) and back-end (NodeJs) using REST APIs helped me understand how to sync data smoothly between them. I also gained experience in handling errors and validating user input, which improved the system's reliability.

5.3 Future Recommendations

To enhance the functionality and user experience of the chat application, several improvements can be considered for future development. One significant enhancement is the integration of push notifications, which would allow users to receive real-time alerts for new messages even when the application is running in the background or is closed. This would ensure continuous engagement and timely communication. Another valuable addition would be the implementation of group chat functionality, enabling multiple users to participate in a single conversation, making the app more versatile for team discussions and social interactions. Furthermore, introducing an admin dashboard would provide administrative control over user activity, allowing for monitoring, moderation, and management of the platform to ensure security and prevent misuse. Lastly, ensuring cross-platform availability by developing versions of the application for iOS and web browsers would significantly broaden the user base and accessibility, allowing users to interact seamlessly across different devices and operating systems.