SNAK

MINOR PROJECT SYNOPSIS

BACHELOR OF TECHNOLOGY

INFORMATION TECHNOLOGY

SUBMITTED BY:

PAWANDEEP SINGH

URN-2004966

CRN-2021087

SIMARDEEP SINGH

URN-2004988

CRN-2021108

SUKHMANDEEP SINGH

URN-2004990

CRN-2021111

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

Contents

Topic **Page No.**

1. INTRODUCTION

Communication has always been an essential part of human interaction. Communication is now lot more effective and quick because of the rapid advancements in technology. One of the most popular ways to communicate with others in real-time is through chat applications. Chat applications are widely used by people in today's world for personal and professional purposes.

'Snak" a Danish word which means talk. This chat application, Snak will be built using MERN (MongoDB, Express, React, Node.js) stack. MERN stack is a popular technology stack used for building web applications. It contains four technologies: MongoDB, a NoSQL database, Express, a web application framework for Node.js, React, a JavaScript library for building user interfaces, and Node.js, a JavaScript runtime built on Chrome's V8 JavaScript engine. These technologies work together to offer a stable and scalable framework for creating web applications.

Snak will be a real-time chat application that allows users to communicate with each other in real-time. Snak application will have features such as user registration and authentication, chat rooms, real-time messaging, message history, and user profile. Users will be able to create their own chat rooms, join existing ones, and send messages in real-time. The messages will be displayed instantly on the screen without the need for refreshing the page.

Snak with MERN Stack will demonstrate the MERN stack's strength and scalability. This project is not just limited to building a chat application, but it will also demonstrate how to build real-time web applications using MERN stack. With the help of this project, developers can learn how to implement real-time features in web applications, how to use MERN stack for building scalable and robust web applications, and how to work with various technologies such as React, MongoDB, Node.js, and Socket.io.

Snak with MERN Stack is a simple yet powerful chat application that will provide users with a seamless communication experience. The application will be user-friendly and easy to use, providing an exceptional user experience. With its advanced features and robust technology stack, Snak with MERN Stack is a project that will impress both developers and users alike.

RATIONALE

There are many chat applications that are dependent on the operating system. In this project we aim to build a chat application that will be independent of operating system. This application 'Snak' will only require a web browser which can run javascript. You can run this web application on any type of operating systems like iOS, Android, Windows, Linux etc. all you need is a web browser.

Snak is not just limited to building a chat application, but it also provides an opportunity for developers to learn how to build real-time web applications using MERN stack. This project will help developers to gain knowledge about various technologies such as React, MongoDB, Node.js, and Socket.io.

2. OBJECTIVES

1. To provide a highly secure and private chat application.

2. To build chat application using MERN Stack.

3. To allow authorized users to send and receive messages easily in real time.

3. FEASIBILITY STUDY

1) Technical Feasibility:
   MERN stack is a well-established technology stack and has been used to build many successful applications. There are many open-source libraries and tools available to support the development of an end-to-end encrypted chat application, such as the Node.js crypto library and the open-source Signal Protocol for encryption.

2) Economic Feasibility:
   The project's economic feasibility depends on the project's budget and funding sources. Developing an end-to-end encrypted chat application can be a resource-intensive project, requiring experienced developers, designers, and security experts. On the other hand, open-source libraries and tools can help reduce the project's costs.

3) Operational Feasibility:
   An end-to-end encrypted chat application requires a robust and scalable infrastructure to handle user traffic and ensure data privacy and security. The application must be able to handle large volumes of messages and users' personal data securely. The application must also comply with relevant regulations and laws, such as data privacy laws.

4. METHODOLOGY

The development process will follow the Agile methodology, and the project will be divided into sprints. Each sprint will involve the following phases:

1) Planning: In this phase, the project team will discuss and plan the features to be implemented in the chat application.

2) Design: In this phase, the project team will design the Snak's architecture, user interface, and user experience. The design should include the security requirements, such as end-to-end encryption and authentication mechanisms.

3) Development: In this phase, the project team will develop the Snak's frontend and backend components along with it's database using the MERN stack.

4) Testing: In this phase, the project team will perform various testing activities, including unit testing, integration testing, and acceptance testing, to ensure the application is functioning as expected and meets the project requirements.

5) Deployment: In this phase, the application will be deployed to a production environment.

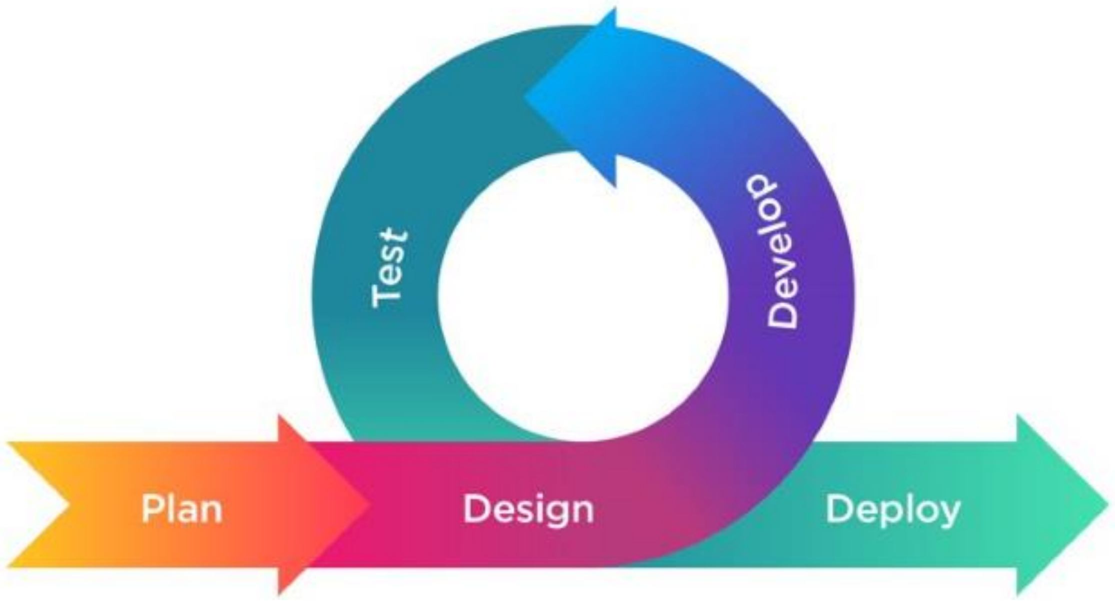

Fig.: Agile methodology

5. FACILITIES REQUIRED FOR PROPOSED WORK

- **Hardware Requirements:**

1) CPU: A multi-core processor with a clock speed of at least 2.5 GHz is recommended to handle the server-side processing load.
2) RAM: At least 4GB of RAM is recommended for a basic chat application. For larger applications, 8GB or more of RAM may be required.
3) Storage: As a starting point, a server with at least 50GB of storage is recommended.
4) Network: A high-speed internet connection with sufficient bandwidth is necessary to handle real-time communication between the server and the clients. A minimum of 1 MBps of upload and download speed is recommended.

- **Software Requirements:**

1) Operating System: You can use any operating system that supports Node.js and MongoDB. Common choices include Linux, macOS, and Windows.
2) Node.js: This is a JavaScript runtime built on the Chrome V8 engine that allows you to run JavaScript on the server-side. You will need to install Node.js on your system to run the server-side code.
3) Express.js: This is a popular Node.js framework that provides a simple and flexible way to build web applications. You will use Express.js to handle HTTP requests and responses.
4) MongoDB: This is a NoSQL database that provides a flexible and scalable way to store data. You will use MongoDB to store user data and chat messages.
5) Socket.IO: This is a real-time communication library that enables bidirectional communication between the server and the client. You will use Socket.IO to handle real-time messaging in your chat application.
6) React: This is a popular JavaScript library for building user interfaces. You will use React to build the front-end of your chat application.
7) Encryption Libraries: To implement end-to-end encryption in your chat application, you will need to use encryption libraries such as CryptoJS, sjcl, or Node.js' built-in crypto module.
8) Version Control System: A version control system like Git can be used to keep track of changes to your code and collaborate with other developers.
9) Integrated Development Environment (IDE): You can use any IDE or code editor that supports JavaScript development. Some popular choices include Visual Studio Code, WebStorm, and Atom.

6. REFERENCES

[1] MEAN Stack Official Website: https://mean.io/

[2] Signal Protocol: https://signal.org/docs/

[3] MongoDB Official Documentation: https://docs.mongodb.com/

[4] Angular Official Documentation: https://angular.io/docs

[5] Node.js Official Documentation: https://nodejs.org/en/docs/

[6] Socket.io Official Documentation: https://socket.io/docs/