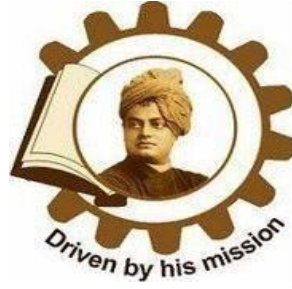


A Project on
“CHAT APPLICATION”

Submitted by
SAYAN ADHIKARY

SWAMI VIVEKANANDA INSTITUTE OF MODERN SCIENCE
Kolkata, West Bengal



SWAMI VIVEKANANDA INSTITUTE OF MODERN SCIENCE

Kolkata, West Bengal

Student Name: SAYAN ADHIKARY

Roll Number: 26401222033

Registration No: 222641010123

Year: 2022-2023

Department: Computer Application

Course: Bachelor of Computer Application

Session: 2022-2025

Paper Name: Industrial Training & Minor Project

Paper Code: BCAD581

College Code: 264

University Name: Maulana Abul Kalam Azad University of Technology

DECLARATION

I hereby declare that the project work title “**CHAT APPLICATION**” Submitted to SWAMI VIVEKANANDA INSTITUTE OF MODERN SCIENCE College Affiliated to MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, as a partial fulfillment for the award of Bachelor of Computer Application. This is an original work done by me and has not been submitted to any other Institution.

PLACE: KOLKATA

.....
WITH SINCERE REGARDS

DATE:

SAYAN ADHIKARY

ACKNOWLEDGEMENT

I think this is an opportunity to thank all the individuals without their help in this project have been incomplete.

I would like to thank "**DR. NANDAN GUPTA**", Director, Swami Vivekananda Institute of Modern Science and "**PROF. GARGI GUPTA**", Vice- Principal of Swami Vivekananda Institute of Modern Science, Karbala More, Sonarpur Station Road, Kolkata-700103.

I would like to thank "**PROF. DR. NILANJANA BHATTACHARYYA NATH**", Principal Swami Vivekananda Institute of Modern Science, Karbala More, Sonarpur Station Road, Kolkata-700103.

I would also like to thank "**PROF. DR. AMBALIKA GHOSH**", Head of the Department of Computer Application.

My profound gratitude goes to "**PROF. JYOTIRMOYEE ROY CHOWDHURY**", Assistant Professor, Department of Computer Application, as our internal guide. I would also express my sincere thanks to the persons who are helpful in explaining about domain information of the existing system.

I am thankful to all the **FACULTY MEMBERS AND LAB ASSISTANT OF DEPARTMENT OF COMPUTER APPLICATION**, also all the **FACULTY MEMBERS** of Swami Vivekananda Institute of Modern Science.

I am grateful to my **PARENTS** and **FRIENDS** for supporting me directly and indirectly during the project.

Finally, I would like to mention here that I am indebted to each and everybody who have been associated with my project at any stage but whose name doesn't find a place in this acknowledgement.

DATE:

.....
WITH SINCERE REGARD

SAYAN ADHIKARY

Table of Content

Sl. No		Topic	Page No
1.		Introduction	1
2.		Literature Review	5
3.		Objective of the Project	12
4.		Purpose of the Project	14
5.		System Analysis and Feasibility Study	15
6		Cost Estimation	20
7.		Software and Hardware Requirements	22
	7.1	Technology Used	23
	7.1.1	Backend Technology	
	7.1.2	Frontend Technology	34
8.		System Design	49
	8.1	Software Development Model	51
	8.2	Data Flow Diagram	56
	8.2.1	Context Level DFD	58
	8.2.2	1 Level DFD	59
	8.2.3	2 Level DFD	60
	8.3	Entity Relationship Diagram	61
	8.4	Normalization	62
	8.5	Database Tables	64
9.		Code and Implementation	69
10.		Case Study	143
	10.1	Testing Methodology	144
	10.2	Test Cases	145
	10.3	Validation	150
	10.4	System Security Measures	153
11.		Maintenance	156
12.		Limitation and Future Scope	159
13.		Conclusion	162

FIGURE INDEX

Sl. No	Figure	Page No.
1	Database	64
2	Feedback Table	65
3	Messages Table	66
4	Requests Table	67
5	Users Table	68
6	Home Page	69
7	Sign Up	83
8	Login In	88
9	Contract List	91
10	Chat	96
11	Message	101
12	Search Bar	107
13	Connection Request	110
14	Database Connection	114
15	Header	115
16	CSS	116

Abstract

This project involves the design and development of a comprehensive one-to-one chat application that facilitates real-time communication along with the ability to share files, images, and videos. The application is built using a combination of modern web technologies, including HTML, CSS, JavaScript, XML, PHP, and MySQL, providing a secure, efficient, and user-friendly platform for communication. The frontend of the application is developed using HTML and CSS, creating an intuitive and visually appealing interface that enhances the user experience. JavaScript is employed to handle the dynamic functionalities of the application, ensuring smooth interactions and real-time updates. XML is used as the data transport medium between the client and the server, enabling seamless communication and efficient data exchange.

On the backend, PHP manages server-side operations, including user authentication, message handling, and file uploads, ensuring the application runs smoothly and securely. MySQL is utilized for database management, where user credentials, chat logs, and shared files are stored in a structured and secure manner. This integration ensures data persistence and accessibility while maintaining privacy and security. The application offers real-time messaging capabilities, allowing users to exchange text messages instantly. It also supports file sharing, enabling users to upload and share images, videos, and documents directly within the chat interface.

The system is designed with a secure login feature to authenticate users and protect their data from unauthorized access. Additionally, the application is optimized for responsiveness, ensuring it works seamlessly across various devices, including desktops, tablets, and mobile phones. Scalability is a key consideration, allowing the application to handle a growing number of users without compromising performance.

The development of this project highlights the integration of web technologies to create a practical and versatile communication tool. By combining real-time messaging, file-sharing capabilities, and a secure infrastructure, this application serves as a valuable platform for both personal and professional use. It provides users with a reliable, efficient, and easy-to-use solution for their communication needs while ensuring data security and privacy. The project demonstrates a successful implementation of web development skills and backend integration to deliver a functional and user-centric product.

1. Introduction

Project Title: Chat Application.

In the modern digital era, communication has become an essential aspect of both personal and professional life. With the rapid advancements in technology, the demand for efficient, secure, and versatile communication platforms has grown significantly. The ability to connect with others in real-time, share multimedia files, and exchange information seamlessly has revolutionized the way we interact. To meet these growing needs, this project focuses on the development of a one-to-one chat application that combines simplicity, functionality, and reliability.

This chat application is designed to facilitate real-time messaging between users while incorporating features such as file, image, and video sharing. The platform is developed using a combination of robust web technologies, including HTML, CSS, JavaScript, XML, PHP, and MySQL, ensuring a seamless and interactive user experience. The primary goal of the project is to create a communication tool that is not only user-friendly but also scalable and secure, making it suitable for diverse use cases, from casual conversations to professional interactions.

The frontend of the application, built with HTML and CSS, provides an intuitive and visually appealing interface. Users can easily navigate through the platform, enhancing their overall experience. JavaScript adds interactivity to the application, enabling real-time updates and dynamic functionalities, such as instant message delivery and notification systems. The use of XML as a data exchange format ensures efficient communication between the client and server, minimizing latency and improving the overall responsiveness of the application.

On the backend, PHP plays a crucial role in managing server-side operations. It handles tasks such as user authentication, session management, and file handling, ensuring a smooth and secure operation of the application. MySQL is employed as the database management system to store user data, chat logs, and shared files in an organized and secure manner. By integrating these technologies, the application achieves a robust and reliable architecture that prioritizes user data security and system performance.

One of the standout features of this application is its support for file sharing. Users can upload and share various types of files, including images, videos, and documents, directly within the chat interface. This functionality adds a layer of versatility to the platform, making it suitable for exchanging information in both personal and professional contexts. For example, users can share important project documents, multimedia presentations, or even personal photos and videos effortlessly.

Security is a top priority in the development of this application. The platform incorporates a secure login system to ensure that only authenticated users can access the chat features. Data transmitted between users is protected using encryption protocols, safeguarding it from unauthorized access or interception. Additionally, the application implements error handling mechanisms to ensure reliable operation, even in the event of network disruptions or unexpected user actions.

Another important aspect of the application is its responsive design. The interface is optimized to function seamlessly across a variety of devices, including desktops, tablets, and mobile phones. This ensures that users can access the platform conveniently, regardless of the device they are using. The scalability of the application further enhances its appeal, allowing it to accommodate a growing number of users without compromising performance.

The significance of this project lies in its ability to provide a practical solution to modern communication challenges. Traditional communication tools often lack the flexibility and functionality required to meet diverse user needs. By integrating real-time messaging with multimedia sharing capabilities, this application addresses these limitations effectively. Moreover, the use of modern web technologies ensures that the platform is not only feature-rich but also future-proof, capable of adapting to emerging trends and user demands.

In conclusion, the development of this one-to-one chat application represents a valuable contribution to the field of web-based communication tools. It combines user-friendly design, robust backend support, and advanced features to deliver a platform that meets the needs of a wide range of users. The project demonstrates the potential of modern web development technologies to create innovative solutions that enhance connectivity and collaboration. As digital communication continues to evolve, applications like this will play a crucial role in shaping the way we interact and share information in the digital age.

Frontend Technology:

The frontend of the chat application serves as the interface between the users and the system, ensuring ease of use, responsiveness, and a visually appealing design. It is built using modern web technologies that focus on delivering a seamless user experience. Below are the detailed technologies used for the frontend:

1. HTML (HyperText Markup Language): HTML is the backbone of the application's structure, defining the layout of the interface. It is used to create essential elements like input fields, buttons, chat boxes, and file upload forms. The semantic structure provided by HTML ensures accessibility and compatibility across different browsers.

2. CSS (Cascading Style Sheets): CSS is responsible for the visual styling of the application, enhancing the look and feel of the interface. It is used to design and format elements such as colors, fonts, spacing, and alignment. Advanced CSS techniques, including Flexbox and Grid, are employed to create a responsive layout that adjusts to various screen sizes. Animations and transitions are added to make the application more interactive and visually engaging.

3. JavaScript: JavaScript brings interactivity and dynamic functionality to the application. It enables features like real-time message updates, notifications, and validation of user inputs. Functions such as dynamic rendering of messages, error handling, and alert prompts are implemented using JavaScript. JavaScript libraries and frameworks, such as jQuery or Vanilla JS, can further enhance the efficiency of the code.

4. XML (eXtensible Markup Language): XML is utilized for structured data exchange between the frontend and backend. It provides a lightweight and efficient way to transfer messages, user information, and metadata. Ensures smooth communication by standardizing the data format for server-client interactions.

5. Responsive Design Techniques: Media queries and responsive web design principles are implemented to ensure the application works seamlessly on desktops, tablets, and mobile devices. The design adapts dynamically to various screen sizes, providing a consistent user experience across platforms.

Backend Technology:

The backend of the chat application is the core engine that powers the functionalities, ensuring secure and efficient operations. It manages server-side logic, database interactions, and user authentication. The following technologies are employed in the backend:

1. PHP (Hypertext Preprocessor): PHP is the primary server-side scripting language used in the project. It handles critical functions like user authentication, file upload processing, and database interactions. PHP scripts ensure seamless communication between the client and server, processing requests and delivering appropriate responses. It implements secure practices such as input validation, session management, and encryption to protect user data.

2. MySQL: MySQL serves as the database management system for the application. It stores essential data, including user profiles, chat logs, file metadata, and authentication credentials. Structured Query Language (SQL) is used to execute queries for retrieving, inserting, updating, and deleting data. The database is optimized for efficient storage and retrieval of large amounts of data, ensuring fast response times. Backup and recovery mechanisms are implemented to protect data integrity.

3. File Handling: The backend manages file uploads, including images, videos, and documents, through PHP's file handling capabilities. Uploaded files are validated for size and format to prevent errors or malicious uploads. Files are securely stored on the server with proper directory structures for easy retrieval.

4. Server Environment: The application is hosted on a web server such as Apache or Nginx, which processes user requests and serves application resources. The server environment ensures the smooth execution of PHP scripts and handles multiple client connections simultaneously. The configuration is optimized for performance and security, supporting HTTPS for encrypted communication.

5. XML for Backend Operations: XML is used to format data exchanged between the frontend and backend, ensuring compatibility and readability. It allows the backend to handle structured data efficiently for message processing and user status updates.

6. Error Handling and Logging: The backend includes mechanisms for error detection and resolution, ensuring reliability. Error logs are maintained for debugging purposes, helping to identify and resolve issues in real-time.

7. Security Features: User authentication is implemented using PHP sessions and password hashing to prevent unauthorized access. Secure file storage and encrypted data transmission ensure that sensitive information remains protected. SQL injection prevention and other security measures are integrated into the backend code to safeguard the application against potential attacks.

By integrating these frontend and backend technologies, the chat application provides a robust platform for real-time communication. The combination of an interactive and responsive interface with secure and efficient backend operations ensures that the application meets the diverse needs of its users, whether for personal or professional use. The careful selection and implementation of these technologies contribute to a seamless and reliable user experience, making the application a valuable tool for modern communication.

1.Literature Review

Chat applications have become an integral part of modern communication systems, facilitating instant communication and collaboration among users across different platforms. With the increasing reliance on digital communication, the development and evaluation of real-time chat applications have garnered significant attention from researchers and developers alike. This literature review aims to explore the current state of the art in real-time chat applications, focusing on their features, technologies, usability, and potential impact on various domains.

Chat applications typically offer a wide range of features to enhance user experience and functionality. These features may include text messaging, multimedia sharing (such as images, videos, and files), group chat functionality, emoticons and stickers, voice and video calling, encryption for security, presence indicators, notifications, and integration with other services and platforms (such as social media and productivity tools). Research by Kelm and Borau (2017) emphasizes the importance of features like message synchronization, offline messaging, and message status indicators for ensuring a seamless user experience in real-time chat applications.

In conclusion, real-time chat applications play a crucial role in modern communication systems, offering a wide range of features and functionalities to facilitate instant communication and collaboration among users. The development and evaluation of real-time chat applications involve considerations such as features, technologies, usability, and potential impact on various domains. Future research in this area may focus on addressing emerging challenges such as scalability, security, and privacy concerns to further enhance the effectiveness and adoption of real-time chat applications.

User Experience and Interface Design:

In the realm of e-commerce, the user experience (UX) holds a paramount role in influencing consumer behavior and shaping overall satisfaction. The significance of a seamless and intuitive interface is particularly pronounced

in the context of e-commerce grocery websites, where users seek efficiency, ease of navigation, and a sense of familiarity akin to traditional in-store shopping.

User experience encompasses every aspect of a customer's interaction with a website, from the initial landing page to the final checkout. Studies have consistently shown that a positive user experience directly correlates with increased user engagement, higher conversion rates, and customer retention. Therefore, successful e-commerce grocery websites prioritize an intuitive and user-friendly design to create a friction less shopping experience.

Key principles of effective user experience in e-commerce include clarity, simplicity, and responsiveness. Clarity ensures that information is presented in a straightforward manner, reducing cognitive load for users. Simplicity involves streamlining the user journey, minimizing unnecessary steps, and presenting information in a digestible format. Responsiveness ensures that the website adapts seamlessly to different devices, catering to users who access the platform from desktops, tablets, or smartphones.

Interface design plays a pivotal role in shaping the user experience. A visually appealing and well-organized interface not only captures users' attention but also guides them through the shopping process effortlessly. This involves strategic placement of navigation menus, clear product categorization, and visually engaging product displays. Additionally, a thoughtfully designed search functionality and an easily accessible shopping cart contribute to a positive user experience.

The application of psychological principles in interface design is another aspect that has gained prominence. For instance, employing persuasive design elements, such as scarcity notifications or personalized recommendations based on user preferences, can influence user behavior positively. The goal is to create an environment where users feel not only in control of their shopping experience but also enticed to explore and make purchases.

As the creator of a new e-commerce grocery website, the integration of these principles becomes instrumental in establishing a competitive edge. By analyzing successful e-commerce platforms and their user-centric design strategies, it becomes possible to discern patterns and best practices that can be applied to enhance the user experience of the new website. In subsequent sections, we will delve into specific aspects of user experience, such as mobile accessibility and security, to further refine our understanding of the intricate

Mobile Accessibility:

In the realm of real-time chat applications, the shift towards mobile accessibility has reshaped the way users engage in communication and collaboration. With the widespread adoption of smartphones and mobile devices, users expect seamless access to chat platforms on the go, whether they are at home, in transit, or at work. Ensuring a smooth and intuitive experience on mobile platforms is essential for meeting the evolving needs of users and maximizing engagement with the chat application.

The rise of mobile chat applications has been fueled by the convenience and flexibility they offer users. Whether it's staying connected with friends and family, collaborating with colleagues, or participating in one-to-one chats, mobile accessibility enables users to communicate anytime, anywhere. Real-time chat applications must prioritize mobile optimization to cater to this mobile-centric user base effectively.

Responsive design is a foundational principle in addressing the challenges of mobile accessibility for real-time chat applications. By designing interfaces that adapt seamlessly to

different screen sizes and resolutions, chat applications can provide a consistent and user-friendly experience across a variety of mobile devices, including smartphones and tablets. Intuitive touch navigation and gesture-based interactions further enhance usability on mobile platforms, allowing users to navigate conversations and access features with ease.

Optimizing performance is another critical aspect of mobile accessibility for chat applications. Minimizing load times, reducing data usage, and optimizing battery consumption contribute to a smoother and more efficient user experience on mobile devices. Implementing caching mechanisms, optimizing network requests, and leveraging lightweight protocols like WebSocket for real-time communication help mitigate the challenges of limited bandwidth and fluctuating network conditions.

Security remains a paramount concern in mobile chat applications, given the sensitive nature of conversations and the potential for unauthorized access. Implementing robust encryption protocols, such as end-to-end encryption, ensures that messages exchanged between users are protected from interception or eavesdropping. Additionally, implementing secure authentication mechanisms, biometric authentication, and two-factor authentication helps safeguard user accounts and prevent unauthorized access to chat conversations.

The integration of push notifications and background sync capabilities enhances the mobile experience by keeping users informed about new messages and updates, even when the chat application is not actively in use. Personalized notifications, tailored to the user's preferences and activity, help re-engage users and drive ongoing interaction with the chat platform.

In the subsequent sections, we will delve into the security and privacy considerations associated with real-time chat applications, exploring how these factors intersect with the mobile accessibility of the platform. By prioritizing mobile optimization and security, the new real-time chat application can deliver a seamless and secure communication experience for users across diverse mobile devices and operating systems.

Security and Privacy Concerns:

In the realm of real-time chat applications, ensuring the security and privacy of user data and communications is of paramount importance. These applications often involve the exchange of sensitive information, including personal messages, media files, and user identities. Therefore, addressing security and privacy concerns comprehensively is crucial to fostering trust among users and safeguarding their information from unauthorized access or misuse.

One of the primary security concerns in real-time chat applications is end-to-end encryption. This encryption method ensures that messages are encrypted on the sender's device and can only be decrypted by the intended recipient, thereby preventing interception or eavesdropping by third parties. Implementing robust end-to-end encryption protocols, such as Signal Protocol or OMEMO, is essential for protecting the confidentiality of user communications.

Privacy considerations extend beyond message encryption to encompass user data protection and anonymity. Real-time chat applications should minimize the collection and retention of

user data to the extent necessary for providing the service. Transparent privacy policies that detail data collection practices, storage methods, and third-party data sharing are essential for building user trust and compliance with privacy regulations like GDPR or CCPA.

Authentication and access control mechanisms are critical for preventing unauthorized access to user accounts and conversations. Implementing secure login processes, such as multi-factor authentication or biometric authentication, adds an extra layer of protection against account hijacking or unauthorized access. Additionally, role-based access control ensures that users have appropriate permissions to access and interact with specific chat features and functionalities.

Secure server infrastructure is fundamental to the overall security of real-time chat applications. Hosting chat servers on secure and reputable cloud platforms, implementing regular security audits and updates, and adhering to industry security standards like ISO 27001 contribute to a robust security posture. Furthermore, real-time monitoring and incident response protocols enable prompt detection and mitigation of security threats or breaches.

Educating users about security best practices is essential for promoting a culture of security awareness and vigilance. Providing resources on password hygiene, safe browsing habits, and recognizing potential security threats, such as phishing attacks or malware, empowers users to protect themselves while using the chat application. Regular security awareness campaigns and communication about security measures implemented by the platform reinforce user trust and confidence.

In the subsequent sections, we will delve into how the new real-time chat application addresses security and privacy concerns, outlining the specific measures implemented to protect user data and ensure a secure messaging environment. By prioritizing security and privacy as core principles, the chat application can differentiate itself in a competitive

Customer Engagement and Loyalty Programs:

In the realm of real-time chat applications, fostering user engagement and retention is paramount for success. These applications thrive on active user participation and ongoing interactions, making it essential to implement effective strategies to keep users engaged and loyal to the platform.

User engagement begins with providing a seamless and intuitive onboarding process for new users. Clear and straightforward interfaces, personalized onboarding experiences, and interactive tutorials can help users familiarize themselves with the application quickly and efficiently. Additionally, offering prompt and responsive customer support channels ensures that users feel supported and valued from the outset.

Loyalty programs are also instrumental in retaining users and encouraging continued usage of the real-time chat application. These programs can include rewards for active

participation, such as badges, special privileges, or access to exclusive features. By incentivizing engagement, loyalty programs foster a sense of belonging and investment in the community, encouraging users to return to the platform regularly.

Post-interaction engagement is equally important for retaining users over the long term. Encouraging users to share feedback, participate in discussions, or contribute user-generated content keeps the conversation alive and fosters a sense of ownership among users. Implementing features like user profiles, chat histories, and customizable settings enhances personalization and encourages users to invest more time and energy into the platform.

Integration with social media platforms offers additional opportunities for user engagement and community building. By leveraging social media channels to share updates, host events, or facilitate discussions, real-time chat applications can extend their reach and attract new users while keeping existing users engaged and connected.

As the developer of a new real-time chat application, understanding the importance of user engagement and retention is paramount. By studying successful chat platforms and analyzing their engagement strategies, developers can identify effective techniques for keeping users active and invested in the platform. Subsequent sections of this literature review will delve into the role of data analytics in optimizing user engagement and retention strategies, ensuring a data-driven approach that resonates with users' preferences and behaviors.

Data Analytics and Personalization:

In the realm of real-time chat applications, leveraging data analytics is pivotal for understanding user interactions, preferences, and engagement patterns. Integrating robust data analytics and personalization functionalities can profoundly enhance the overall user experience and drive the success of the application.

Data analytics within real-time chat applications encompasses various activities, including tracking user interactions, analyzing message content, and monitoring usage trends. Advanced analytics tools can provide insights into user demographics, conversation topics, and sentiment analysis, allowing for a deeper understanding of user behavior.

Personalization is a key aspect of enhancing user engagement within real-time chat applications. By leveraging data analytics, these platforms can tailor the user experience based on individual preferences and behaviors. This can include personalized message suggestions, targeted content recommendations, and customized user interfaces that adapt to user preferences.

Data analytics also facilitates performance monitoring and optimization within real-time chat applications. By analyzing metrics such as message response times, user engagement rates, and conversation flow, developers can identify areas for improvement and optimize the application's performance in real time.

However, it's essential to prioritize user privacy and data security when implementing data analytics and personalization features within real-time chat applications. Adhering to data protection regulations and implementing robust security measures are critical for maintaining user trust and confidence.

As the real-time chat application is developed, integrating data analytics and personalization functionalities becomes instrumental in providing a tailored and engaging user experience. By leveraging data insights and machine learning algorithms, the application can anticipate user needs, enhance user engagement, and drive overall satisfaction. Subsequent sections of this literature review will delve into how data analytics intersects with other aspects of real-time chat application development, such as security and scalability, to provide a comprehensive understanding of its role in application success.

Challenges and Opportunities:

In the dynamic realm of real-time chat applications, a plethora of challenges and opportunities define the course of these platforms. Comprehending and adeptly addressing these factors is fundamental for the seamless development, refinement, and continuous enhancement of any real-time chat application.

Challenges:

Intense Market Competition: The real-time chat application market is highly competitive, with numerous players ranging from established platforms to innovative startups. To stand out in this crowded space, our chat application must offer unique features and a seamless user experience that sets it apart from competitors.

Technical Challenges: Developing a real-time chat application involves addressing various technical challenges, such as optimizing server performance, ensuring message delivery reliability, and handling scalability issues to accommodate growing user bases.

User Trust and Data Security: Building trust among users is paramount in real-time chat applications, as users share personal and sensitive information during conversations. Implementing robust encryption protocols, stringent data protection measures, and

Evolution of User Preferences: User preferences and behaviors in communication technologies are continually evolving. Staying abreast of these changes and adapting the application to meet evolving user needs and expectations is crucial for maintaining relevance and competitiveness in the market.

Opportunities:

Technological Advancement: Embracing cutting-edge technologies such as artificial intelligence, machine learning, and augmented reality offers a promising pathway to revolutionize user experiences, streamline operations, and gain a competitive edge in the market.

Sustainability Measures: The increasing focus on environmental sustainability presents an opportunity for e-commerce grocery websites to adopt eco-friendly practices in packaging, delivery, and sourcing, aligning with the values of environmentally conscious consumers.

Data-Driven Decision-Making: Leveraging the vast amount of data generated through user interactions enables informed decision-making. Implementing data analytics for personalized marketing, inventory management, and strategic planning can lead to more efficient business operations.

Global Reach: With the growing acceptance of online grocery shopping worldwide, there is immense potential for global expansion. Exploring new markets and demographics can unlock untapped growth opportunities for the business.

Partnerships and Collaborations: Collaborating with local farmers, producers, and other businesses not only strengthens the supply chain but also fosters community engagement. Partnering with third-party services for expedited delivery can further enhance customer satisfaction and loyalty.

Addressing these challenges and seizing the opportunities requires a comprehensive and adaptable approach. Rather than obstacles, these challenges serve as stepping stones to success, while the opportunities, if harnessed strategically, can propel the e-commerce grocery website to new heights of innovation and market leadership.

As the literature review delves into these challenges and opportunities, it will offer a nuanced understanding of the complexities within the e-commerce grocery industry. Subsequent sections will explore specific strategies and considerations, providing a well-

3 Objective of the Project

The objective of the Chat Application project is to develop a dynamic platform that facilitates seamless communication, collaboration, and engagement among users in real-time. The system aims to address various challenges faced by traditional messaging systems, such as latency issues, limited functionalities, and lack of interactivity.

The Chat Application project aims to provide an intuitive and feature-rich platform that can handle diverse communication needs, including group chats, file sharing, and multimedia sharing. The system should also support notifications, message synchronization across devices, and encryption for enhanced security.

The overall goal of the project is to enhance the quality of user experience while also improving the efficiency of communication processes. By offering communication solution with advanced features and seamless performance, the system can foster better collaboration, productivity, and connectivity among users.

Additionally, the system should enable users to stay connected and informed across various devices and platforms, promoting flexibility and accessibility in communication.

The primary objective of our Chat Application is to redefine communication experiences by offering a dynamic, interactive, and user-friendly platform for messaging. Our goal is to create a versatile and engaging digital space that not only simplifies communication but also enhances connectivity and collaboration among users.

Seamless Communication: Enable users to engage in conversations with individuals fostering instant communication and efficient information exchange without delays or interruptions.

Versatility and Flexibility: Provide users with a versatile messaging platform that supports a wide range of communication features, including text messaging, multimedia sharing, catering to diverse communication needs.

Enhanced Connectivity: Facilitate seamless connectivity across devices and platforms, allowing users to stay connected and informed regardless of their location or device, promoting flexibility and accessibility in communication.

Privacy and Security: Prioritize user privacy and data security by implementing robust encryption protocols, secure authentication mechanisms, and privacy controls to safeguard sensitive information and ensure a secure messaging environment.

Collaboration and Productivity: Foster collaboration and productivity by integrating features such as file sharing, task management, and collaborative editing, enabling users to work together seamlessly and efficiently within the messaging platform.

Innovation and Evolution: Continuously innovate and evolve the messaging platform by incorporating new features, technologies, and user feedback to enhance functionality, improve user experience, and stay ahead of emerging communication trends.

Community Building: Foster a vibrant and inclusive community by encouraging user engagement, participation, and interaction through forums, interest groups, and community events, building strong connections and fostering a sense of belonging among users.

Purpose of our system:

- To provide a user-friendly and intuitive platform where users can engage in real-time conversations with individuals, fostering seamless communication and collaboration.
- Enhance connectivity and accessibility by enabling users to stay connected across devices and platforms, ensuring flexibility and convenience in communication.
- Facilitate efficient information exchange and decision-making by offering versatile messaging features, including text, multimedia sharing.
- Prioritize user privacy and data security by implementing robust encryption protocols and secure authentication mechanisms to safeguard sensitive information and maintain a secure messaging environment.
- Empower users to personalize their messaging experience according to their preferences and needs, offering customization options and intuitive interface designs.
- Foster a collaborative and productive environment by integrating features such as file sharing, video and document sharing, enabling users to work together seamlessly within the messaging platform.
- Continuously innovate and evolve the messaging platform by incorporating user feedback and emerging technologies to enhance functionality, improve user experience, and stay ahead of communication trends.

4. Purpose of the Project

The purpose of our chat application transcends mere communication; it is a commitment to transforming the way individuals connect and interact in the digital realm. Rooted in the belief that technology can enhance human connections, our project aspires to serve a multifaceted purpose that extends beyond messaging:

Facilitating Seamless Communication: Our purpose is to provide users with a seamless and intuitive platform for instant communication, bridging geographical distances and fostering meaningful connections in real-time.

Empowering Collaboration and Engagement: We envision our project as a catalyst for collaboration and engagement, enabling users to share ideas, collaborate on projects, and build communities around shared interests and passions.

Encouraging Inclusivity and Diversity: Beyond facilitating conversations, our purpose extends to fostering inclusivity and celebrating diversity. We are committed to providing a platform where users from diverse backgrounds feel welcomed, respected, and valued.

Enhancing Productivity and Efficiency: Our project aims to enhance productivity and efficiency by streamlining communication processes, facilitating quick decision-making, and enabling swift information exchange among team members and stakeholders.

Promoting Privacy and Security: At the core of our purpose is the commitment to safeguarding user privacy and ensuring data security. We prioritize implementing robust encryption measures and adhering to stringent privacy policies to protect user confidentiality.

Cultivating a Positive Digital Experience: We seek to cultivate a positive digital experience for our users, characterized by user-friendly interfaces, intuitive navigation, and personalized features tailored to individual preferences and needs.

Fostering Innovation and Creativity: Our purpose extends to fostering innovation and creativity by providing a platform that encourages experimentation, idea generation, and knowledge sharing among users.

In essence, our chat application is not just a messaging tool but a purpose-driven endeavor to facilitate connections, empower collaboration, and enrich the digital landscape with meaningful interactions.

5. System Analysis and Feasibility Study

System Analysis for Chat Application Website

The system analysis delves into the essential components and factors necessary for the effective implementation of the chat application, aiming to create a seamless, secure, and intuitive platform for instant communication. Continuous monitoring, iteration, and user input will play vital roles in refining and optimizing the system for optimal performance and user satisfaction.

System Overview:

Purpose: The chat application aims to provide users with a seamless and instant messaging platform for efficient communication.

Scope: The system encompasses user interfaces for creating and joining chat rooms, exchanging messages in , managing user profiles, and administrative functionalities for moderation and system management.

Functional Requirements:

User Authentication: Implement robust user authentication mechanisms to ensure secure access to the chat application and protect user privacy.

Chat: Develop functionality for sending friend request and accepting friend request, allowing users to engage in conversations.

Messaging: Design features for exchanging messages, supporting text, image and video messages.

User Profiles: Enable users to create profiles, customize settings, and manage their messaging preferences.

Moderation Tools: Implement administrative features for monitoring chat activity, managing user accounts, and enforcing community guidelines.

Technical Architecture:

Front-End Technologies: Utilize HTML, CSS and JavaScript to build dynamic and responsive user interfaces for seamless interaction.

Back-End Development: Implement the back-end using PHP to handle server-side logic, data processing, and communication with the database.

Database Management: Utilize MySQL as the database system for storing user data, chat room information, and message logs.

Database Design:

Database Schema: Develop a schema for the database system, ensuring efficient storage and retrieval of data related to user accounts, messages, and system settings.

Normalization: Apply normalization principles to organize data into logical tables, minimizing redundancy and ensuring data consistency.

Indexing: Utilize indexing techniques to optimize database performance, especially for frequently accessed data fields and queries

User Experience (UX):

Responsive Design: Implement responsive design principles to ensure that the chat application is accessible and visually appealing across various devices and screen sizes.

Intuitive Interface: Design an intuitive user interface with clear navigation, allowing users to easily create or join chat rooms, send messages, and manage their profiles.

Feedback Mechanisms: Incorporate feedback mechanisms such as message delivery indicators, typing indicators, and read receipts to enhance user engagement and communication clarity.

Security Measures:

End-to-End Encryption: Implement end-to-end encryption to secure message transmission and protect user privacy from unauthorized access.

Input Validation: Validate user inputs to prevent common security threats such as SQL injection, ensuring data integrity and system security.

Access Control: Enforce access control measures to restrict unauthorized access to sensitive features and data within the chat application.

Integration and External Services:

Real-Time Communication: Implement real-time communication using JavaScript and PHP in combination with WebSocket libraries like PHP-based WebSocket framework. This ensures instant message delivery and synchronization across multiple devices.

Authentication Services: Use PHP authentication mechanisms to handle user authentication and authorization. Implement secure practices such as password hashing, session management, and token-based authentication to ensure a reliable and seamless user experience.

Cloud Storage Integration: Integrate PHP with cloud storage services like AWS S3 or Google Cloud Storage for efficient storage and retrieval of multimedia files shared within chat rooms. Alternatively, use MySQL for local multimedia storage and management with well-structured database design.

This database design and user experience considerations ensure a secure, efficient, and user-friendly real-time chat application, meeting the needs of modern communication requirements.

Testing Strategy:

Unit Testing: Perform comprehensive unit testing for each module and component to validate their functionality in isolation and ensure they meet the specified requirements.

Integration Testing: Conduct integration testing to verify the seamless interaction and data exchange between different modules, identifying and resolving any integration issues.

User Acceptance Testing (UAT): Engage end-users in UAT sessions to evaluate the system's usability, functionality, and adherence to requirements, incorporating their feedback for refinement.

Performance Testing: Evaluate the system's performance under various load conditions, ensuring it can handle expected user traffic and maintain responsiveness.

Security Testing: Conduct security testing to identify and address potential vulnerabilities, ensuring the system's resilience against security threats and unauthorized access.

Maintenance and Scalability:

Code Maintainability: Adhere to coding best practices and documentation standards to ensure code maintainability, facilitating easier debugging, troubleshooting, and future enhancements.

Scalability: Design the system architecture with scalability in mind, employing scalable technologies and distributed systems principles to accommodate future growth in user base and data volume.

Continuous Monitoring: Implement robust monitoring tools and processes to proactively identify and address performance issues, security breaches, and other system anomalies in real-time.

Regular Updates and Patches: Stay vigilant about software updates, security patches, and bug fixes, ensuring the system remains up-to-date and protected against emerging threats and vulnerabilities.

Feedback Mechanisms: Establish channels for collecting user feedback and feature requests, leveraging user input to prioritize and implement system improvements and enhancements over time.

Feasibility Study for Chat Application Website

The feasibility study for the chat application aims to evaluate its viability, potential risks, and benefits before proceeding with development. This study involves analyzing market demand, technical requirements, financial projections, and risk assessments to inform decision-making and ensure project success.

Market Analysis:

Market Overview: Examine the current landscape of chat applications, identifying emerging trends, market leaders, and potential growth opportunities in the communication software sector.

Target Audience: Define the target audience for the chat application, considering factors such as age demographics, communication preferences, and usage patterns across different platforms.

Market Size: Estimate the size of the market for chat applications, taking into account the number of active users, market penetration, and projected growth rates based on industry reports and user surveys.

Technical Feasibility:

Application Development: Assess the technical requirements for building a real-time chat application using HTML, CSS, JavaScript, PHP, and MySQL. Focus on platform compatibility, efficient database management, and implementing robust data encryption techniques to ensure secure communication.

Cross-Platform Compatibility: Design the application to work seamlessly across various devices and operating systems, ensuring a broad user base and enhanced accessibility.

Integration: Implement integration with cloud storage platforms for multimedia file management and explore the use of APIs for enhanced multimedia sharing capabilities, all within the PHP and MySQL environment. Implement integration with cloud storage platforms for multimedia file management and explore the use of APIs for enhanced multimedia sharing capabilities, all within the PHP and MySQL environment.

Operational Feasibility:

User Engagement: Assess strategies for fostering user engagement and retention within the chat application, such as gamification elements, multimedia support, and interactive features.

Scalability: Evaluate the application's ability to scale efficiently to accommodate a growing user base and increasing message traffic without compromising performance or reliability.

Moderation and Content Management: Determine the feasibility of implementing moderation tools and content management systems to ensure a safe and enjoyable user experience while preventing abuse and inappropriate behavior.

Financial Feasibility:

Cost Estimation: Analyze the costs associated with developing, launching, and maintaining the chat application, including software development, server infrastructure, marketing campaigns, and ongoing operational expenses.

Monetization Strategies: Explore potential revenue streams, such as subscription plans, in-app purchases, targeted advertising, or premium features, to generate sustainable income and ensure long-term profitability.

Revenue Projections: Develop financial projections based on user acquisition forecasts, pricing models, and monetization strategies to assess the application's potential return on investment and financial viability over time.

Legal and Regulatory Compliance:

Compliance Requirements: Identify and assess legal and regulatory requirements relevant to chat applications, including data privacy laws, user data protection regulations, and telecommunications regulations.

Permits and Licenses: Determine the feasibility of obtaining any necessary permits or licenses for operating a chat application, ensuring compliance with local and international laws.

Risk Analysis:

Market Risks: Evaluate potential market risks such as competition from existing chat applications, shifts in user preferences, and market saturation in the communication software sector.

Operational Risks: Identify operational risks related to server downtime, network outages, software bugs, and user experience issues that may impact the application's performance and reputation.

Regulatory Risks: Assess risks associated with non-compliance with legal and regulatory requirements, including fines, penalties, and legal disputes arising from data protection violations or failure to adhere to telecommunications regulations.

Conclusion and Recommendations:

Summarize the feasibility study's findings, emphasizing the potential benefits and risks associated with developing and launching a chat application. Provide recommendations for mitigating risks, maximizing opportunities, and ensuring compliance with legal and regulatory requirements.

Appendices:

Include supplementary materials such as detailed risk assessments, regulatory compliance checklists, legal documentation, and any additional information supporting the feasibility study's conclusions and recommendations.

6. Cost Estimation

Cost estimation in a feasibility study for a grocery management system involves identifying various expenses associated with developing, implementing, and maintaining the system. Here is a breakdown of potential costs:

1. Development Costs:

Software Development: Hiring developers to create the grocery management system. Hardware Cost of servers, computers, and other necessary hardware. Software Tools and Licenses Expenses for purchasing software tools, licenses, and development frameworks. Testing and Quality Assurance Funds for testing the system to ensure it works efficiently and meets requirements.

2. Implementation Costs:

Training Expenses for training staff to use the new system effectively. Data Migration Cost associated with migrating existing data to the new system.

3. Operational Costs:

Maintenance Ongoing expenses for system maintenance, updates, and bug fixes. Support: Costs for providing technical support to users. Cloud Services or Hosting Fees If using cloud-based services, include subscription or hosting fees.

4. Infrastructure Costs:

Networking Expenses for establishing and maintaining network infrastructure. Security: Cost of implementing security measures to protect data. Backup and Recovery Funds for backup systems and recovery procedures.

5. Miscellaneous Costs:

Consultation and Professional Services Fees for consulting services or professional advice. Contingency Buffer funds for unforeseen expenses or changes in project scope.

6. Total Cost Estimation:

Summing up all the aforementioned costs will give an estimate of the total expenditure required for the grocery management system. Given the code limit of 10,000 characters, a more detailed breakdown of the costs, along with specific figures and calculations, might exceed the character limit. You'll need to create a comprehensive spreadsheet or detailed document to provide accurate figures for each cost category.

Component	Description	Estimated Effort (Person Month)	Hourly Rate (INR)	Estimated Cost (INR)
Project Planning	Define Project Scope requirements and plan development Phases	1	20	4,800
UI/UX Design	Design user interface and Experience	2	50	28,000
Front-End Developer	Implementing the client side of the application	3	100	72,000
Back-End Developer	Develop Server Side Logic and Database	3	120	86,400
Security Implementation	Implement Security Measures as Inscription	1	70	16,800
Testing and QA	Ensuring application function bug- free	1	30	7,200
Development and Optimization	Application to Production and Optimize	1	45	10,800
Total		12	-	2,00,800

7. SOFTWARE AND HARDWARE REQUIREMENTS

Hardware Requirement Specification:

Client Machine		Server Machine	
HDD	500 GB	HDD	500 GB
Processor	Pentium 4 or Newer Processor	Processor	Dual Core or Newer Process or
Memory	4 GB	Memory	4 GB

3.3.2 Software Requirement Specification:

Client Machine		Server Machine	
Browser	Any Standard Browser	Software	Compass
Client Side Markup/ Scripting Language	HTML, CSS JavaScript	Database Management System Software	Mysql

7.1 Technology Used

7.1.1 Backend Technology

PHP:

PHP is a widely-used, open-source, server-side scripting language specifically designed for web development. It is highly versatile, making it suitable for creating dynamic web pages, handling form submissions, interacting with databases, and more. PHP runs on various platforms and integrates seamlessly with databases like MySQL, offering an efficient way to develop robust backend solutions.

Key Features of PHP

1. **Server-Side Execution:**
 - PHP code is executed on the server, and the output (usually HTML) is sent to the user's browser.
 - This ensures that the user's browser only sees the result of the PHP script, enhancing security and efficiency.
2. **Embedded in HTML:**
 - PHP can be easily embedded into HTML code, making it convenient to integrate dynamic content within static web pages.
3. **Platform Independent:**
 - PHP scripts can run on various operating systems, including Windows, Linux, macOS, and Unix, when paired with a compatible web server.
4. **Database Integration:**
 - PHP supports integration with a wide variety of databases, such as MySQL, PostgreSQL, SQLite, and MongoDB, allowing developers to create database-driven applications.
5. **Open Source:**
 - PHP is free to use, and its source code is available, enabling developers to study, modify, and extend the language.
6. **Extensive Library Support:**
 - PHP has numerous built-in libraries and functions for tasks like working with strings, arrays, file handling, and session management.
7. **Object-Oriented Programming (OOP):**
 - PHP supports OOP principles, enabling developers to write reusable and modular code.
8. **Support for Web Technologies:**
 - PHP can handle various protocols such as HTTP, FTP, and SMTP and supports integration with JavaScript, CSS, and XML.

Common uses of PHP:

- Perform server-side tasks such as creating, reading, updating, and deleting files on a system.
- Handle forms to gather data, save data to a database, and return responses to the client.
- Manage user authentication, including login and session handling.
- Set and access cookies for user preferences or session tracking.
- Connect and interact with databases
- Power popular CMS platforms
- Collect and process data from users or other sources, and generate reports or visualizations.
- Securely process sensitive information using encryption techniques.
- Generate dynamic web pages and manage API endpoints for communication between the client and server.

Characteristics of PHP:

Five important characteristics make PHP practical nature possible-

- **Open-Source:** Freely available and supported by a vast community of developers.
- **Cross-Platform Compatibility:** Runs on major operating systems like Windows, Linux, and macOS.
- **Easy Integration:** Works seamlessly with databases, front-end technologies, and third-party libraries.
- **Rich Library Support:** Includes built-in functions for common tasks such as file handling, database interaction, and session management.
- **Performance Optimization:** Efficient execution for both small-scale and large-scale applications.

Limitations of PHP :

1. Performance Issues

- **Interpreted Language:** PHP scripts are executed on the server side and interpreted at runtime, which can make them slower compared to compiled languages like C++ or Java.
- **Concurrency Limitations:** PHP is less efficient when handling a high number of simultaneous requests, especially compared to asynchronous languages like Node.js.

2. Scalability

- **Not Ideal for Very Large Applications:** While PHP can scale, it may require more effort and additional tools (e.g., load balancers, caching systems) to handle large-scale systems compared to some other languages.

3. Lack of Strong Typing

- **Dynamic Typing:** PHP's dynamic typing can lead to unexpected behavior or errors that are caught only at runtime, making debugging more challenging.

4. Security Concerns

- **Frequent Vulnerabilities:** Older versions of PHP are prone to security vulnerabilities if not properly updated or configured.
- **Developer Responsibility:** Security depends heavily on the developer's understanding of PHP. For example, not sanitizing inputs can lead to SQL injection, XSS, or CSRF attacks.

5. Standard Library Limitations

- **Inconsistent Function Naming:** PHP's standard library has functions with inconsistent naming conventions and argument orders, which can be confusing for developers.
- **Limited Multithreading Support:** PHP does not natively support multithreading, which can limit performance for certain applications.

6. Dependency on Web Server

- **Server Requirement:** PHP code requires a web server (like Apache or Nginx) and cannot run standalone without specific configurations.

7. Fragmented Ecosystem

- **Legacy Code:** Many PHP applications rely on older versions of the language, creating compatibility challenges.
- **Framework Diversity:** While frameworks like Laravel, Symfony, and CodeIgniter are powerful, the diversity can make it hard to standardize development practices.

8. Limited Built-In Tools for Modern Development Practices

- **No Native Package Management (Pre-Composer):** Before Composer, PHP lacked a strong package management system, which hindered dependency management.
- **Testing:** PHP does not have robust, built-in testing tools. Developers often rely on external tools like PHPUnit.

9. Community Perception

- **Historical Stigma:** PHP is sometimes seen as less sophisticated or outdated compared to newer languages, which can influence its adoption in modern projects.

10. Complex Error Handling

- **Error Reporting:** PHP has historically had less intuitive error handling mechanisms, although recent versions (like PHP 7 and later) have improved this significantly.

Advantages of PHP

1. Easy to Learn and Use

- PHP has a simple syntax, making it beginner-friendly.
- A vast amount of documentation and tutorials are available online.

2. Open-Source

- PHP is free to use, making it cost-effective for developers and organizations.
- The source code is open, allowing for community contributions and custom modifications.

3. Platform Independence

- PHP runs on various operating systems, including Windows, Linux, and macOS, ensuring compatibility across platforms.

4. Wide Framework Support

- PHP has a variety of powerful frameworks, such as Laravel, Symfony, CodeIgniter, and Zend, that streamline development and enforce best practices.

5. Large Community Support

- PHP has a massive and active community that provides extensive libraries, tools, and forums for troubleshooting.

6. Integration with Databases

- PHP supports various databases like MySQL, PostgreSQL, SQLite, and Oracle, making it versatile for different projects.

7. Scalability

- PHP applications can scale effectively when combined with tools like load balancers, caching mechanisms (e.g., Memcached), and cloud services.

8. Rapid Development

- Its built-in functions and frameworks enable developers to build applications quickly.
- PHP's flexibility allows for easy updates and changes during the development process.

9. Robust Content Management Systems

- Popular CMS platforms like WordPress, Joomla, and Drupal are built with PHP, making it a go-to choice for web development.

10. Versatility

- PHP can handle a wide range of applications, from small websites to large-scale enterprise applications.

Disadvantages of PHP

1. Performance Issues

- PHP is slower compared to compiled languages like C++ and Java.
- Interpreted nature and lack of native multithreading can hinder performance for large-scale applications.

2. Weak Typing

- PHP's dynamic typing can lead to subtle bugs and unexpected behavior.

3. Security Vulnerabilities

- Older PHP versions are prone to security issues.
- Security depends heavily on the developer's skills in writing secure code, such as proper input validation and sanitization.

4. Poor Error Handling in Older Versions

- Prior to PHP 7, error handling mechanisms were less robust, which could lead to complex debugging processes.

5. Inconsistent Function Naming

- The PHP standard library has inconsistently named functions and parameter orders, leading to potential confusion.

6. Scalability Challenges

- While scalable, PHP requires significant effort and additional tools to manage high traffic effectively.

7. Not Ideal for Modern Development Needs

- PHP is not inherently suited for asynchronous programming, which is increasingly used in modern applications.

8. Stigma in the Developer Community

- PHP has a reputation for being an "outdated" or "inelegant" language, which can make it less appealing to some developers.

9. Dependency on Web Servers

- PHP scripts require a server environment (e.g., Apache or Nginx) to execute, adding complexity to the setup.

10. Legacy Code Challenges

- Many PHP projects rely on outdated codebases, which can be difficult to maintain or upgrade.

MySQL :

MySQL is a popular open-source relational database management system (RDBMS) used to store and manage application data. It is designed for high performance, scalability, and reliability, making it an ideal choice for managing structured data in web applications.

Common uses of MySQL:

- Storing user data, messages, and multimedia file metadata in relational tables.
- Retrieving and updating data dynamically for real-time user interactions.
- Managing complex queries and relationships between tables using SQL.

- Providing secure storage for sensitive data with access control and encryption features.
- Enabling efficient indexing for quick search and retrieval.

Characteristics of MySQL:

- **Relational Database:** Data is organized into structured tables with rows and columns.
- **Database Management:** Store, organize, and retrieve data efficiently for websites and applications.
- **Web Applications:** Power dynamic websites and applications like blogs, forums, and content management systems by storing user data, content, and settings.
- **User Authentication:** Store and manage user credentials, roles, and permissions for secure login systems.
- **Scalability:** Handles large volumes of data effectively.
- **Cross-Platform:** Runs on major operating systems, ensuring flexibility.
- **Performance-Oriented:** Optimized for read and write operations in web applications.
- **Secure:** Supports authentication, access control, and encryption.

Limitations of MySQL

Despite being one of the most popular relational database management systems (RDBMS), MySQL has some limitations. These limitations vary based on its use case, version, and configuration.

1. Lack of Full SQL Compliance

- **Standard Limitations:** MySQL does not fully support all features of the SQL standard, such as certain types of subqueries, triggers, and window functions (though newer versions have added some of these).
- **Feature Gaps:** Advanced features like Common Table Expressions (CTEs) and full outer joins were absent in earlier versions.

2. Performance Issues

- **High-Concurrency Problems:** MySQL might struggle with performance under heavy read/write loads when compared to other systems like PostgreSQL.
- **Locking:** In some storage engines, table-level locking (e.g., in MyISAM) instead of row-level locking can reduce performance in high-concurrency environments.

3. Limited Scalability

- **Sharding Complexity:** Scaling MySQL horizontally requires sharding or clustering, which can be complex to implement and maintain.
- **Limited Native Partitioning Support:** While MySQL offers partitioning, its capabilities are more restrictive compared to some competitors like PostgreSQL or Oracle.

4. No Built-In Full-Text Search for Large Data

- MySQL's full-text search capabilities are limited compared to specialized tools like Elasticsearch, especially for large datasets and advanced search requirements.

5. Storage Engine Limitations

- **Engine-Specific Constraints:** The choice of storage engine (e.g., MyISAM, InnoDB) impacts functionality. For example:
 - MyISAM does not support transactions or foreign keys.
 - InnoDB supports transactions but may have performance overhead for specific use cases.
- **Incompatibility:** Switching between storage engines may result in feature incompatibility or data migration challenges.

6. Data Size Limitations

- **Large Datasets:** While MySQL can handle large datasets, performance may degrade with very large databases or when working with massive tables and joins.

7. Weak Native Security

- **Role Management:** Earlier versions of MySQL lacked robust role-based access control (RBAC), which made it harder to enforce complex security policies.
- **Encryption:** Native encryption support is limited compared to some other database systems.

8. Limited Analytics and OLAP Capabilities

- MySQL is designed for Online Transaction Processing (OLTP) and lacks the advanced analytics and OLAP (Online Analytical Processing) features present in databases like PostgreSQL or specialized systems like Snowflake.

9. Backup and Recovery Challenges

- **Manual Backups:** While tools like mysqldump exist, they can be slow for large datasets, and point-in-time recovery requires additional setup.
- **Replication Challenges:** Setting up and maintaining replication for backup purposes can be complex and error-prone.

10. Community Edition Limitations

- **Paid Features Missing:** Some advanced features, like high-availability clustering and enhanced security options, are available only in the paid versions (MySQL Enterprise Edition).

11. No Native JSON Indexing Before Version 5.7

- While MySQL supports JSON, earlier versions lacked proper indexing for JSON fields, making it inefficient for handling JSON data.

12. Limited Support for Advanced Features

- **Advanced Queries:** Features like recursive queries and materialized views are either unavailable or limited in functionality compared to competitors like PostgreSQL.
- **Extensibility:** While MySQL supports plugins, its ecosystem for extending database functionality is less developed compared to other databases.

13. Logging and Debugging

- MySQL's error logs and diagnostic tools are less comprehensive compared to other systems, making debugging and monitoring more challenging in some cases.

Advantages of MySQL

1. Open Source

- MySQL is open-source, which means it is free to use, modify, and distribute.
- Regular updates and community support enhance its features and usability.

2. Cross-Platform Support

- MySQL runs on various platforms, including Windows, Linux, macOS, and UNIX, making it versatile and flexible.

3. Ease of Use

- MySQL has a user-friendly interface, making it accessible even to beginners.
- Tools like phpMyAdmin simplify database management.

4. High Performance

- MySQL offers fast query execution and reliable performance for small-to-medium-sized applications.
- Optimizations like indexing and caching enhance speed.

5. Scalability

- MySQL can scale vertically (upgrading hardware) and horizontally (replicating databases or sharding) to accommodate growing workloads.

6. Robust Security

- MySQL provides robust security features, including user authentication, SSL support, and data encryption.
- Permissions and user roles can be configured to secure sensitive data.

7. Wide Community and Support

- An extensive community of developers and users offers support, plugins, and solutions.
- Many tutorials, forums, and documentation are available online.

8. Compatibility with Web Applications

- MySQL integrates well with popular web technologies, such as PHP, Python, and Java.
- It powers many content management systems (CMS) like WordPress, Joomla, and Drupal.

9. Replication and Backup

- MySQL supports master-slave replication and clustering for high availability.
- Tools like mysqldump facilitate data backup and recovery.

10. Cost-Effective

- The free Community Edition is sufficient for many projects, while paid versions offer advanced features for enterprise needs.

Disadvantages of MySQL

1. Limited SQL Compliance

- MySQL does not fully comply with the SQL standard, leading to inconsistencies or limitations in complex queries.
- Advanced features like window functions, Common Table Expressions (CTEs), and full outer joins were only introduced in later versions.

2. Performance Challenges

- **Concurrency Issues:** MySQL may struggle with high-concurrency environments compared to databases like PostgreSQL.
- **Table Locking:** Storage engines like MyISAM use table-level locking, which can affect performance in write-heavy operations.

3. Lack of Advanced Features

- Missing or limited support for features like materialized views, advanced analytics, and OLAP functionality.
- JSON support is available but less efficient compared to specialized NoSQL databases.

4. Scalability Limitations

- Horizontal scaling (e.g., sharding) requires significant manual effort and expertise.
- Not ideal for extremely large-scale systems without external tools.

5. Storage Engine Constraints

- The choice of storage engine affects functionality:
 - MyISAM lacks support for transactions and foreign keys.
 - InnoDB supports these features but may introduce performance overhead in some cases.

6. Security in Older Versions

- Earlier versions of MySQL had weaker security mechanisms, though recent updates have addressed many of these issues.

7. Limited Community Edition Features

- Some advanced functionalities, like data masking, audit logging, and enterprise backup, are available only in the paid MySQL Enterprise Edition.

8. Backup and Recovery Limitations

- Tools like mysqldump can be slow for large databases, and point-in-time recovery requires extra setup.
- Replication setup for disaster recovery can be complex and error-prone.

9. Logging and Debugging

- MySQL's diagnostic tools and error logging are less comprehensive compared to other RDBMS options, making it harder to debug complex issues.

10. No Native NoSQL Functionality

- Although MySQL supports JSON and other non-relational data types, it is not a true hybrid database like some modern systems that blend SQL and NoSQL.

7.1.2 Frontend Technology

HTML (Hypertext Markup Language):

Purpose: HTML is the standard markup language used to create the structure of web pages. It defines the elements and their relationships, such as headings, paragraphs, lists, links, images, forms, and more.

Role: HTML provides the basic building blocks for web content, allowing browsers to interpret and display information.

Key Features of HTML

1. Platform-Independent

- HTML is universally supported by all modern web browsers (e.g., Chrome, Firefox, Safari), regardless of the operating system (Windows, macOS, Linux). It allows web pages to be accessible across different devices and platforms.

2. Extensibility

- HTML is flexible and can be easily extended by using technologies like **CSS** (for styling) and **JavaScript** (for interactivity), which work together to create complex and dynamic websites.

3. Semantic Structure

- HTML uses **semantic tags** (e.g., <header>, <footer>, <article>, <section>) to improve the readability and accessibility of web content. This helps search engines and screen readers understand the content's meaning and structure.

4. Hyperlinks and Navigation

- HTML enables linking between different web pages using **hyperlinks** (). This is fundamental to the web, as it allows easy navigation between various resources on the internet.

5. Multimedia Support

- HTML supports embedding various types of media such as images, videos, and audio. Using tags like , <audio>, and <video>, HTML allows seamless integration of multimedia content into web pages.

6. Form Elements for User Input

- HTML offers powerful **form elements** (e.g., <input>, <textarea>, <button>) that enable user interaction, such as filling out forms, submitting data, and selecting options. This is vital for creating interactive websites and web applications.

7. Responsive Design with CSS

- While HTML defines the structure, combined with CSS, HTML supports **responsive design**—adapting content to different screen sizes and devices (e.g., mobile, tablet, desktop). CSS media queries allow websites to adjust dynamically to different display resolutions.

8. Document Object Model (DOM)

- HTML content is structured as a **DOM** (Document Object Model), a tree-like structure where every element is represented as a node. JavaScript can interact with the DOM to modify the structure, content, or style of a webpage dynamically.

9. Data Representation

- HTML supports specialized elements for **data representation**, such as:
 - <table> for tabular data.
 - , , and for listing information.
 - <form> for collecting data from users.

10. Built-in Validation

- HTML5 introduced built-in **form validation** features, such as required, pattern, min, and max attributes. These help ensure that user inputs meet certain criteria before submission.

11. Meta Tags for SEO and Performance

- HTML includes **meta tags** (e.g., <meta name="description" content="...">) to improve search engine optimization (SEO) and provide metadata like author, charset, and viewport settings for mobile optimization.

Advantages of HTML

1. Simplicity and Easy to Learn

- HTML is simple and intuitive, making it easy for beginners to learn and implement. Its tag-based structure allows even non-technical users to create basic web pages without extensive knowledge of programming.

2. Free and Open-Source

- HTML is an open standard and is freely available to anyone. There are no licensing fees or restrictions, and it can be used by anyone to create websites.

3. Rapid Development

- HTML allows for quick prototyping and development. With minimal code, developers can quickly create and test web pages. This is particularly beneficial for startups and developers looking to launch projects quickly.

4. SEO-Friendly

- HTML is designed with search engine optimization (SEO) in mind. Properly structured HTML documents with semantic tags can improve a website's visibility in search engines, driving more traffic.

5. Cross-Platform Compatibility

- HTML pages are designed to work across all browsers and devices (e.g., mobile, tablet, desktop). This cross-platform compatibility ensures that web pages can be accessed from any device with a web browser.

6. Supports Rich Content

- HTML supports a wide range of content types, such as text, images, audio, video, links, forms, and more. This allows for the creation of rich and engaging web pages that cater to different types of content.

7. Widely Supported

- HTML is the standard markup language supported by all major web browsers (Chrome, Firefox, Safari, Edge, etc.), ensuring that web pages will render consistently across different platforms and devices.

8. Search Engine Indexing

- Since HTML is easy for search engines to read and index, it plays a crucial role in helping search engines understand the content of a web page, which can improve a site's ranking.

9. Enhanced User Experience

- HTML enables the creation of structured, accessible, and navigable web pages. Features like forms, navigation menus, and multimedia content provide a more interactive and engaging user experience.

10. Facilitates Easy Content Management

- With HTML, developers and content managers can quickly update and manage web content, such as text, images, and links, without requiring a complicated back-end system.

11. Support for External Resources

- HTML allows easy integration with external resources such as **JavaScript** for dynamic behavior, **CSS** for styling, and **AJAX** for asynchronous data loading, making it easier to build modern web applications.

12. Data-Driven Web Pages

- HTML supports embedding and displaying data, making it easy to create data-driven web pages. Using forms and elements like `<input>` and `<select>`, developers can create interactive forms to collect and display data from users.

13. No Dependency on Specific Software

- HTML doesn't require proprietary software or a specific operating system to be used. As long as you have a text editor and a web browser, you can create and view HTML documents anywhere.

14. Increased Flexibility with HTML5

- The introduction of **HTML5** brought numerous new features like local storage, audio/video embedding, geolocation, and advanced form controls, which significantly expand its capabilities and flexibility.

15. Easy Debugging and Testing

- Since HTML is essentially a text file, it is easy to debug and test. Errors can be quickly identified using developer tools in browsers, and any issues with the page's structure can be corrected in real-time.

16. Supports Modular Design

- HTML allows for a modular approach to design. Using **external CSS** and **JavaScript files**, developers can keep the structure (HTML), design (CSS), and behavior (JavaScript) separate, improving code maintainability and reusability.

17. Interoperability with Other Web Technologies

- HTML works seamlessly with other web technologies such as **CSS** for design, **JavaScript** for interactivity, **PHP** or **Node.js** for server-side logic, and **Databases** for data storage. This makes it the backbone of most web applications.

18. Accessibility Features

- HTML provides several elements like `<label>`, `<fieldset>`, and `<legend>` to enhance accessibility. Developers can also use ARIA (Accessible Rich Internet Applications) attributes to create web content that is more accessible to users with disabilities.

19. Local Storage and Offline Capabilities (HTML5)

- HTML5 introduced the ability to store data locally on the user's device using the **localStorage** and **sessionStorage** APIs, enabling offline functionality for web apps, thus improving the user experience even when there is no internet connection.

20. Community Support and Documentation

- HTML benefits from a large, active community of developers, and there is extensive documentation available online. This support makes it easy for developers to find solutions to problems, share knowledge, and keep up with best practices.

Limitations of HTML

While HTML is essential for creating web pages, it does have certain limitations. These limitations often require the use of additional technologies (like CSS, JavaScript, or server-side languages) to overcome them. Below are the key limitations of HTML:

1. Lack of Interactivity

- **Limitation:** HTML is a static markup language and does not provide built-in mechanisms for interactivity. It is primarily used to structure content, and without JavaScript, HTML can't handle dynamic user interactions such as form validation, real-time content updates, or complex animations.
- **Solution:** JavaScript is typically used to add interactivity, such as event handling, animations, and asynchronous data requests.

2. Limited Styling Capabilities

- **Limitation:** HTML provides basic styling elements (like fonts and colors), but it has limited control over layout and design. Complex styles (e.g., grids, responsive layouts) can't be fully achieved with HTML alone.
- **Solution:** CSS (Cascading Style Sheets) is used alongside HTML to control the layout, design, and responsive behavior of web pages.

3. No Built-in Support for Complex Logic

- **Limitation:** HTML cannot perform calculations, conditional logic, or manipulate data. For example, it can't process forms or handle server-side logic on its own.
- **Solution:** To add logic or backend operations, HTML needs to be paired with server-side scripting languages like **PHP**, **Node.js**, **Python**, or **Ruby**, as well as client-side scripting like **JavaScript**.

4. Limited Multimedia Support

- **Limitation:** While HTML can embed images, audio, and video, it lacks advanced features for handling multimedia content such as editing, compression, or creating interactive multimedia experiences.
- **Solution:** For richer multimedia experiences, HTML should be used in conjunction with JavaScript or specialized libraries (e.g., for video playback, games, or image manipulation).

5. SEO Limitations

- **Limitation:** Although HTML is SEO-friendly, it requires careful structuring and tagging for optimal results. Without proper use of semantic tags like `<header>`, `<footer>`,
- `<article>`, or `<nav>`, SEO rankings can suffer.
- **Solution:** Developers need to follow best practices, like using the correct headings and alt text for images, and adding meta tags for better SEO.

6. Accessibility Challenges

- **Limitation:** Without proper markup and consideration for accessibility (like using alt text for images or ARIA attributes), HTML pages can be inaccessible to people with disabilities (e.g., blind users relying on screen readers).

- **Solution:** Web developers need to be mindful of accessibility standards, ensuring that HTML documents are structured to be navigable and readable by assistive technologies.

7. Browser Inconsistencies

- **Limitation:** HTML renders differently across various browsers, leading to inconsistencies in appearance and functionality. This can cause issues with older browsers or versions that do not support modern HTML standards (e.g., HTML5).
- **Solution:** Developers often use browser testing tools and workarounds (like CSS prefixes or polyfills) to ensure compatibility across different browsers.

8. Limited Control over Layouts

- **Limitation:** HTML alone offers limited control over layouts. For example, elements in HTML flow in a linear structure by default, and complex layouts require CSS techniques like Flexbox, Grid, or frameworks like Bootstrap.
- **Solution:** CSS is used to create flexible layouts, responsive designs, and manage the positioning of elements.

9. No Data Storage or Persistence

- **Limitation:** HTML does not support data storage or database integration. It is purely a markup language and cannot store data on the server or client-side.
- **Solution:** To store and manage data, HTML must be combined with databases (e.g.,
- **MySQL, MongoDB**) and server-side languages (e.g., **PHP, Node.js**).

10. Limited Security Features

- **Limitation:** HTML itself doesn't provide security features to protect data or users. It's vulnerable to issues like cross-site scripting (XSS) and cross-site request forgery (CSRF).
- **Solution:** Security measures must be implemented through server-side validation, input sanitization, encryption, and other security best practices.

11. Dependence on Other Technologies

- **Limitation:** HTML alone cannot create fully functional websites. For dynamic content, interactivity, or server communication, HTML must be paired with **CSS, JavaScript**, and server-side technologies like **PHP, Node.js**, etc.
- **Solution:** To build feature-rich web applications, HTML works together with other technologies to enhance functionality.

12. No Native Support for Server-Side Logic

- **Limitation:** HTML cannot process or execute server-side logic (e.g., interacting with databases, authentication, or user session management). It only defines the structure of the content to be displayed.
- **Solution:** HTML must be used in combination with server-side languages (like **PHP, ASP.NET, Node.js**) to handle server-side logic.

13. Limited Data Validation

- **Limitation:** Although HTML5 offers some basic form validation (e.g., required, pattern, email), it cannot fully validate complex data inputs or enforce business logic before submission.

- **Solution:** JavaScript or server-side validation is typically used to ensure that the submitted data is valid and meets required conditions.

CSS (Cascading Style Sheets):

Purpose: CSS is used for styling and layout of web pages. It allows developers to control the visual presentation of HTML elements, defining aspects like colors, fonts, spacing, and positioning.

Role: CSS enhances the visual appeal and user experience of web pages by providing a way to customize the presentation of HTML content.

Features of CSS

1. Separation of Content and Presentation:

CSS allows you to separate the content (HTML) from the presentation (style), making it easier to manage and maintain web pages. This separation also allows for a cleaner, more organized code structure.

- **Styling Control:** CSS provides precise control over how HTML elements are displayed on a webpage. You can control fonts, colors, layouts, borders, spacing, and more to enhance the visual appeal of your content.
- **External Style Sheets:** One of the main features of CSS is the ability to link an external stylesheet to your HTML document. This means that you can maintain one CSS file to control the style of multiple pages, reducing redundancy and improving maintainability.
- **Responsiveness and Flexibility:** CSS supports responsive design, allowing you to design websites that adapt to different screen sizes, devices, and orientations. This is achieved using media queries, flexbox, and CSS grid systems.
- **Cross-Browser Compatibility:** CSS ensures that web pages can look consistent across different browsers and platforms. It provides tools like vendor prefixes and polyfills to ensure that styles are rendered correctly across all major browsers (e.g., Chrome, Firefox, Safari, Edge).
- **Cascading Effect:** The cascading nature of CSS allows styles to be inherited from parent elements, and rules to be overridden by more specific or later declarations. This makes it easy to manage and adjust styles without affecting the entire page.

2. Multiple Styling Options

CSS provides various ways to style HTML elements, including:

- **Inline styles:** Directly within HTML tags.
- **Internal styles:** Within a <style> tag in the HTML document.
- **External styles:** In a separate .css file.

3. Text Styling and Fonts

CSS allows you to define font types, sizes, weights, and styles, and control the spacing between lines and characters. You can use **web fonts** like Google Fonts to ensure consistent typography across different devices.

4. Box Model

The **CSS box model** allows you to control the dimensions of elements using padding, borders, margins, and content area. It gives you full control over the spacing and layout of elements on the page.

5. Positioning and Layout

CSS provides multiple positioning techniques such as **static**, **relative**, **absolute**, **fixed**, and **sticky** to control the position of elements on the page.

CSS also enables the creation of complex layouts using techniques like **flexbox** and **grid layout**.

6. Transitions and Animations

CSS allows you to create **smooth transitions** between different states (e.g., hover effects) and define **animations** with keyframes to add dynamic, visual effects to your website elements.

7. Z-Index and Layering

CSS provides the **z-index** property, which allows you to control the stacking order of elements. This feature is useful for managing overlapping elements like images, pop-ups, and menus.

8. Border, Margin, Padding Control

CSS gives you full control over the **borders**, **margins**, and **padding** of elements, allowing for precise spacing and alignment. You can customize border thickness, color, style, and apply rounded corners using **border-radius**.

9. Pseudo-classes and Pseudo-elements

CSS includes **pseudo-classes** (e.g., `:hover`, `:focus`, `:active`) that allow you to apply styles based on the state of an element, such as when a user hovers over it or clicks on it.

Pseudo-elements (e.g., `::before`, `::after`) enable you to style specific parts of an element without modifying the HTML structure.

10. Custom Properties (CSS Variables)

CSS Variables enable you to store reusable values that can be applied throughout the stylesheet. This makes it easy to manage and update global values like colors and fonts across a website.

11. User Interface Controls

CSS allows you to style form elements and create custom buttons, checkboxes, radio buttons, and input fields, ensuring a consistent and attractive user interface (UI).

12. Custom Grid Systems

CSS provides **grid-based layouts** that make it easier to create complex, flexible, and responsive designs. Using **CSS Grid**, you can define a two-dimensional grid to place elements in specific rows and columns.

13. Media Queries

CSS enables the use of **media queries**, which help create responsive designs by applying different styles based on the screen size, orientation, resolution, and other device characteristics.

14. CSS Frameworks and Preprocessors

CSS frameworks like **Bootstrap**, **Foundation**, and **Tailwind CSS** offer pre-defined styles and components to speed up development.

CSS preprocessors like **Sass** and **LESS** allow for advanced features like variables, mixins, and nested rules, making CSS more modular and maintainable.

Advantages of CSS

1. Separation of Content and Presentation

CSS allows you to separate the content (HTML) from the presentation (style). This makes the HTML code cleaner, easier to read, and maintain. You can make changes to the design without modifying the actual content.

2. Improved Website Performance

When CSS is stored in external files, it reduces the size of the HTML files. Browsers can cache these CSS files, making subsequent page loads faster and improving the overall performance of the website.

3. Consistency Across Web Pages

By linking a single CSS file to multiple web pages, you ensure a consistent design throughout the entire website. Any changes made to the CSS will be reflected across all linked pages, ensuring uniformity.

4. Easier to Maintain and Update

CSS makes website maintenance easier since styles are centralized in one location (or a few files). If you want to change the design, you only need to update the CSS, rather than going through each HTML file individually.

5. Improved Accessibility

CSS enhances accessibility by allowing easy adjustments to styles, such as font sizes or color schemes, to accommodate users with different needs, including those with visual impairments.

6. Responsive Design

CSS allows developers to design websites that are responsive to various screen sizes and devices. Using techniques like **media queries**, **flexbox**, and **grid layout**, CSS ensures that websites look great on desktops, tablets, and smartphones.

7. Flexibility in Layouts

With CSS, you can create complex and flexible layouts using properties like **flexbox** and **grid**. These layout models give you greater control over how elements are positioned, aligned, and distributed on a page.

8. Improved User Experience

CSS enables the creation of visually attractive web pages with smooth transitions, animations, and interactive hover effects. These features enhance the user experience by

making websites more engaging.

9. Customization of UI Elements

CSS allows you to customize the appearance of various UI elements like buttons, form fields, dropdown menus, and more. This ensures that all elements of the website have a consistent and attractive look.

10. Better Control Over Typography

CSS provides comprehensive control over text styling, including font type, size, weight, line spacing, letter spacing, and text alignment. This allows developers to create visually appealing and readable text.

11. Cross-Browser Compatibility

CSS makes it easier to ensure that your website appears consistently across different browsers, such as Chrome, Firefox, Safari, and Edge. Vendor prefixes and other techniques help deal with browser-specific differences.

12. Reusability and Scalability

With CSS, you can reuse style rules across multiple elements, pages, or even projects, saving time and effort. It allows you to scale your website by applying styles uniformly across different sections.

13. Simpler Code Structure

CSS eliminates the need for inline styles in HTML, reducing clutter and improving the structure of your code. This leads to better readability, easier collaboration, and more manageable projects.

14. Enhanced Visual Effects

CSS provides support for visual effects like **hover effects**, **transitions**, and **animations**. These effects add a dynamic feel to your website, making it more interactive and engaging for users.

15. Easy Integration with Other Technologies

CSS integrates seamlessly with HTML and JavaScript, making it a versatile tool for web development. It allows developers to use JavaScript for dynamic content manipulation while CSS handles the visual styling.

16. Reduced HTML File Size

CSS reduces redundancy in HTML by keeping style information in a separate file. This results in smaller HTML files and better performance, especially on websites with large amounts of content.

17. User-Controlled Preferences

CSS allows users to override certain styles (like text size or color scheme) based on their personal preferences or accessibility needs, enhancing the website's usability.

18. Support for Print Styles

CSS allows you to define different styles for printing documents. You can specify how elements should appear when printed (e.g., hiding unnecessary images or adjusting font sizes), providing a better user experience for print users.

19. Improved SEO (Search Engine Optimization)

CSS makes web pages more SEO-friendly by reducing the use of inline styles and optimizing the structure of the content. A cleaner, more organized HTML structure can help search engines better crawl and index your pages.

20. Cost-Effective and Time-Saving

CSS reduces the amount of work needed for styling by enabling the reuse of style rules across multiple pages. It makes development faster and more cost-effective, especially for large websites.

Limitation

1. Limited Control Over Layouts

While CSS provides powerful layout tools like **flexbox** and **grid**, it still has limitations when dealing with very complex or non-standard layouts. For example, certain intricate layouts may require workarounds or additional techniques such as JavaScript for precise positioning.

2. Browser Compatibility Issues

Different browsers may interpret and render CSS in slightly different ways. Despite CSS3 and modern techniques, developers still face issues with browser inconsistencies, especially with older browsers that may not fully support newer CSS features (e.g., **CSS Grid** or **CSS Variables**).

3. Difficulty with Complex Animations

Although CSS supports animations and transitions, for complex, highly interactive animations or visual effects, it may not be as efficient or flexible as JavaScript. Certain animation effects (like physics-based motion or detailed interactions) are often difficult to achieve solely with CSS.

4. Limited Styling for Content Manipulation

CSS is designed for styling, not content manipulation. While it allows for visual changes, it cannot modify the actual content structure or dynamic elements as JavaScript can. For example, altering HTML structure based on conditions requires JavaScript, not CSS.

5. No Built-In CSS Variables in Older Browsers

Older browsers do not support **CSS variables** (custom properties), which can make it difficult to implement certain modern CSS features. As a result, developers must find alternative solutions or use JavaScript to handle dynamic styling.

6. Styling Restrictions for Form Elements

While CSS allows you to style form elements like buttons, input fields, and checkboxes, some form elements are difficult to fully customize, especially on mobile devices. Browsers often impose limitations on how these elements can be styled.

7. Performance Issues with Complex Selectors

Using overly specific or complex CSS selectors can impact performance, particularly in large web applications. **Deeply nested selectors** or overly broad selectors (like `*`) can slow down the rendering process, especially on pages with many elements.

8. Lack of Logic and Conditional Styles

CSS does not support logic or conditional statements (e.g., `if`, `else`). This means that you cannot apply styles based on dynamic conditions like user behavior or application state without using JavaScript. For example, changing styles based on user interaction beyond hover effects requires JavaScript.

9. Global Styles Can Overlap

The **cascade** effect in CSS can sometimes cause issues with global styles conflicting with local or specific styles. Overriding styles across large projects can become cumbersome if specificity is not handled carefully.

10. No Built-in Component-Based Design

Unlike modern JavaScript frameworks (e.g., React, Vue), CSS does not have built-in support for component-based designs. While frameworks like **Sass** or **CSS-in-JS** try to address this, pure CSS doesn't natively support breaking up styles into smaller, reusable components.

11. Steep Learning Curve for Advanced Features

Advanced CSS concepts, such as **CSS Grid**, **Flexbox**, and **Custom Properties (CSS Variables)**, can have a steep learning curve for beginners. Understanding the intricacies of these systems and their interdependencies requires experience and practice.

12. Lack of Precise Control Over Typography

Although CSS offers control over basic typography (font size, weight, family), achieving perfect typographic control across devices and platforms can be challenging. For example, fonts may render differently across browsers and operating systems, affecting consistency.

13. Dependency on External Libraries

To overcome some of the limitations of CSS, developers often rely on external CSS libraries (e.g., **Bootstrap**, **TailwindCSS**) or preprocessors like **Sass** or **LESS**. However, this can increase the size of the project and make it more dependent on external resources.

14. Limited Interaction With Non-Visual Elements

CSS is primarily focused on visual styling and lacks the ability to interact with non-visual elements like data, business logic, or application flow. Complex logic (e.g., checking whether a user is logged in or manipulating data) requires JavaScript or server-side code.

15. No Direct Support for Variables (Until Recently)

While **CSS Variables** were introduced in CSS3, they are not universally supported by all browsers, particularly older versions. Some styles or layouts may need to be hard-coded without the flexibility of variables.

JavaScript:

Purpose: JavaScript is a versatile programming language that adds interactivity and dynamic behavior to web pages. It allows developers to manipulate the DOM (Document Object Model), handle events, and create responsive and interactive user interfaces.

Role: JavaScript enables client-side scripting, making it possible to create interactive features, validate forms, fetch data from servers and more.

Key Features of JavaScript

1. Client-Side Scripting

JavaScript is executed on the **client side**, meaning that code runs directly in the user's browser. This enables faster interactions without needing to communicate with a server for every user action.

2. Dynamic Typing

JavaScript is **dynamically typed**, meaning that variables do not have a fixed data type. This allows variables to hold values of any data type, such as strings, numbers, objects, or arrays.

3. Event-Driven

JavaScript is often used for event-driven programming, where functions (called **event handlers**) are triggered in response to user actions like clicks, keypresses, or mouse movements. This enables dynamic and interactive websites.

4. Cross-Platform Compatibility

JavaScript runs on all major web browsers, including Chrome, Firefox, Safari, and Edge. This ensures that JavaScript code works across various platforms, from desktops to mobile devices.

5. Asynchronous Programming

JavaScript supports **asynchronous programming** through mechanisms like **callbacks**, **promises**, and **async/await**. This allows developers to perform tasks (like API requests or data loading) without blocking the rest of the program.

6. Object-Oriented Programming (OOP)

JavaScript supports **object-oriented programming** features like classes, objects, inheritance, and methods. This allows for more structured and reusable code.

7. Functional Programming

JavaScript also supports **functional programming** paradigms, where functions are treated as first-class citizens. Functions can be passed around as arguments, returned from other functions, and assigned to variables.

8. Manipulation of the DOM

JavaScript can access and modify the **DOM** (Document Object Model), allowing you to change the content, structure, and style of a webpage dynamically. This is one of the core features for creating interactive web applications.

9. Compatibility with HTML and CSS

JavaScript works seamlessly with HTML and CSS to create dynamic web pages. JavaScript can interact with HTML elements and update their styles, content, or structure based on user input or other triggers.

10. Frameworks

There are many popular JavaScript libraries and frameworks, such as that simplify and streamline the development of complex chat application.

11. Rich Ecosystem of APIs

JavaScript has access to a vast ecosystem of APIs (Application Programming Interfaces) for tasks like geolocation, working with storage (localStorage, sessionStorage), interacting with external services via AJAX or Fetch API, and much more.

12. One to One Side JavaScript

JavaScript is also used on the **One side** with environments like. This allows developers to use JavaScript for both client-side and server-side development, creating full-stack applications with a single programming language.

Advantages

1. Client-Side Execution

JavaScript is executed on the **client side**, meaning the code runs directly in the user's browser, reducing the load on the server. This results in faster response times and a better user experience by providing immediate feedback without needing to reload the entire page.

2. Interactivity

JavaScript allows for **dynamic, interactive content** such as image sliders, form validation, pop-ups, and animations. It enables user interactions without needing to refresh the page, making websites more engaging and responsive.

3. Asynchronous Programming

JavaScript supports **asynchronous programming** through callbacks, promises, and async/await. This allows developers to execute tasks such as data fetching, form submissions, and animations without blocking the rest of the program, leading to smoother experiences.

4. Cross-Platform Compatibility

JavaScript is supported by all major web browsers, such as Chrome, Firefox, Safari, and Edge, making it a cross-platform language that works seamlessly on various devices (desktop, mobile, tablet) without requiring modifications.

5. Rich Ecosystem of Libraries and Frameworks

JavaScript has a vast ecosystem of libraries (e.g., **jQuery**, **Lodash**) and frameworks (e.g., **React**, **Angular**, **Vue.js**) that simplify development, provide ready-made solutions, and reduce the amount of code developers need to write.

6. Real-Time Applications

JavaScript is often used in **real-time web applications**, such as chat applications or live notifications. Through technologies like **WebSockets**, JavaScript enables real-time updates without the need for page reloads.

7. Development

JavaScript can be used for **Development** as well, enabling full-stack development using a single language. This means developers can handle both front-end and back-end logic using JavaScript, improving consistency and reducing context switching.

8. Easy to Learn and Use

JavaScript is relatively **easy to learn** for beginners, with simple syntax and abundant resources available for learning. This makes it a great starting point for new developers and accelerates the development process for experienced developers.

9. Dynamic Typing

JavaScript is **dynamically typed**, meaning variables do not require a predefined type. This makes it easier and faster to write code without having to define specific types for each variable, though it requires developers to be cautious with type coercion.

Limitation

1. Browser Compatibility

JavaScript relies heavily on the browser's JavaScript engine for execution, and different browsers (especially older ones) may interpret JavaScript code differently. Although modern browsers support most features, certain features may not be compatible with all browsers or their older versions. This can lead to inconsistent behavior across browsers unless specific workarounds are implemented.

2. Single-Threaded Nature

JavaScript is **single-threaded**, meaning it can only execute one operation at a time. This can cause performance issues when handling CPU-intensive tasks or large data processing, as the browser may become unresponsive. However, JavaScript mitigates this limitation through **asynchronous programming** (e.g., **callbacks**, **Promises**, **async/await**) and **Web Workers**, which run in the background.

3. Security Concerns

JavaScript is often targeted by malicious attacks such as **Cross-Site Scripting (XSS)**, **Cross-Site Request Forgery (CSRF)**, and **injection attacks**. Since JavaScript is executed on the client-side, it can be vulnerable to attacks if proper security measures are not taken. Developers need to sanitize input data, use secure coding practices, and implement security protocols to mitigate risks.

4. Limited Access to System Resources

JavaScript running in the browser has **limited access to system resources** for security reasons. For example, JavaScript cannot interact directly with the file system or perform operations like running system-level processes. This restricts its use for tasks that require such access (e.g., operating system management or heavy computation).

5. Difficult to Debug

While modern JavaScript development tools have significantly improved debugging, **tracking down bugs** in JavaScript can still be challenging, especially with large and complex codebases. Errors can sometimes be difficult to isolate, and issues may not become apparent until runtime, which can complicate the debugging process.

8. System Design

System design plays a pivotal role in shaping the architecture and functionality of a chat application. Here's how we adapt the key aspects of system design to the context of a chat application:

Architectural Design:

Define the overall architecture of the chat application, including client-server architecture, messaging protocols, and communication channels. Consider scalability, fault tolerance, and real-time synchronization requirements.

Data Design:

Design the database schema to store user profiles, chat histories, and metadata. Normalize the database structure to optimize storage and retrieval. Implement caching mechanisms for frequently accessed data to enhance performance.

User Interface Design:

Create wireframes and mockups to design the chat interface for desktop and mobile devices. Focus on usability, responsiveness, and intuitive navigation. Incorporate features like message threading and media attachments.

Input and Output Design:

Design input forms for sending messages, including text input fields and file upload options. Define the output format for displaying messages in real-time, including message bubbles, timestamps, and user avatars.

Process Design:

Specify the message delivery process, including message routing, queuing, and delivery confirmation. Implement algorithms for handling concurrent connections, message prioritization, and error recovery.

Security Design:

Identify security risks such as unauthorized access, message interception, and data breaches. Implement end-to-end encryption for message transmission.

Software and Hardware Design:

Select appropriate technologies for frontend development for dynamic user interfaces. Choose backend frameworks like PHP and MySQL for server-side logic. Design server infrastructure for scalability and high availability.

Error Handling and Recovery Design:

Develop error handling mechanisms to handle network errors, server failures, and message delivery issues. Implement retry strategies for failed message deliveries and provide users with informative error messages.

Testing Strategy:

Define testing approaches for unit testing, integration testing, and end-to-end testing of chat features. Use testing frameworks like Jest or Mocha for automated testing. Perform stress testing to evaluate system performance under peak loads.

Documentation:

Create comprehensive design documentation detailing system architecture, database schema, API endpoints, and integration points. Document chat protocols, message formats, and error codes for reference.

Prototyping:

Build prototypes or mock servers to validate the chat application's functionality and user experience. Gather feedback from stakeholders and users to iterate on the design.

Scalability and Performance Design:

Design the chat application to handle increasing user concurrency and message volume. Implement features like message pagination, message caching, and load balancing for scalability. Optimize database queries and API endpoints for performance.

Maintainability and Extensibility:

Design the chat application with modular components and clear separation of concerns to facilitate maintenance and updates. Use design patterns like MVC or MVVM to organize codebase. Plan for future feature enhancements and integrations with third-party services.

Implementation Planning:

Develop a detailed implementation plan outlining development tasks, timelines, resource allocation, and deployment strategies. Break down development sprints and milestones to track progress effectively.

By incorporating these system design principles, a real-time chat application can deliver a seamless, secure, and scalable communication experience for users across various platforms.

8.1 Software Development Model

Agile Model

The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

- 1.** Requirements gathering
- 2.** Design the requirements
- 3.** Construction/ iteration
- 4.** Testing/ Quality assurance
- 5.** Deployment
- 6.** Feedback

Features of Agile Method:

- **Iterative and Incremental Development:** Agile focuses on delivering small, functional portions of the project in iterations, allowing for continuous delivery and frequent feedback.
- **Customer Collaboration:** Close collaboration with customers throughout the project ensures that the product aligns with their needs and expectations.
- **Flexibility and Adaptability:** Agile teams can adapt to changing requirements and make adjustments during the project, ensuring flexibility in response to customer needs or market changes.
- **Continuous Delivery:** Agile encourages the delivery of working software at the end of each iteration, providing stakeholders with frequent, tangible progress.

- **Focus on Individuals and Interactions:** Agile emphasizes the importance of communication and collaboration among team members, valuing people over processes and tools.
- **Simplicity and Efficiency:** Agile promotes delivering only essential features and avoids unnecessary complexity, leading to simpler and more efficient development.
- **Frequent Communication and Feedback:** Agile promotes regular communication within the team and with stakeholders to ensure everyone is aligned and feedback is incorporated early.
- **Self-Organizing Teams:** Agile teams are empowered to make decisions and take ownership of tasks, allowing for more autonomy and effective problem-solving.
- **Prioritization of Features (Product Backlog):** Agile prioritizes features based on business value, ensuring that the most important tasks are addressed first.
- **Frequent Testing:** Agile encourages regular testing to identify issues early, ensuring high-quality software and reducing the time spent fixing bugs later.
- **Time-Boxed Sprints:** Agile development is structured around fixed-length iterations (sprints), which provide focus and maintain steady progress toward project goals.
- **Prioritize Working Software Over Comprehensive Documentation:** Agile focuses on delivering functional software instead of excessive documentation, speeding up the development process.
- **Continuous Improvement:** Agile teams engage in retrospectives after each iteration to reflect on the process and identify ways to improve for the next sprint.
- **Transparency:** Agile ensures visibility into the project's progress and challenges through regular updates and reviews, building trust among stakeholders.
- **Minimal Documentation:** Agile emphasizes creating just enough documentation to support development, avoiding the overhead of excessive paperwork.
- **Risk Management:** Agile minimizes risks by delivering working software frequently and incorporating feedback, allowing teams to address issues early.
- **Customer-Centric Focus:** Agile ensures that the customer's needs are central to the development process, with continuous feedback ensuring the product aligns with expectations.
- **High Customer Satisfaction:** Agile prioritizes delivering value to the customer through working software and frequent communication, leading to higher satisfaction.
- **Increased Visibility:** Agile provides transparency through tools like burn-down charts and progress reports, ensuring that everyone stays informed on project progress.
- **Collaboration with Business Stakeholders:** Agile involves business stakeholders throughout the development process, ensuring that the product meets business requirements and objectives.

Advantage(Pros) of Agile Method:

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.

- It reduces total development time.
- Iterative and Incremental Development: Agile focuses on delivering small, functional portions of the project in iterations, allowing for continuous delivery and frequent feedback.
- Customer Collaboration: Close collaboration with customers throughout the project ensures that the product aligns with their needs and expectations.
- Flexibility and Adaptability: Agile teams can adapt to changing requirements and make adjustments during the project, ensuring flexibility in response to customer needs or market changes.
- Continuous Delivery: Agile encourages the delivery of working software at the end of each iteration, providing stakeholders with frequent, tangible progress.
- Focus on Individuals and Interactions: Agile emphasizes the importance of communication and collaboration among team members, valuing people over processes and tools.
- Simplicity and Efficiency: Agile promotes delivering only essential features and avoids unnecessary complexity, leading to simpler and more efficient development.
- Frequent Communication and Feedback: Agile promotes regular communication within the team and with stakeholders to ensure everyone is aligned and feedback is incorporated early.
- Self-Organizing Teams: Agile teams are empowered to make decisions and take ownership of tasks, allowing for more autonomy and effective problem-solving.
- Prioritization of Features (Product Backlog): Agile prioritizes features based on business value, ensuring that the most important tasks are addressed first.
- Frequent Testing: Agile encourages regular testing to identify issues early, ensuring high-quality software and reducing the time spent fixing bugs later.
- Time-Boxed Sprints: Agile development is structured around fixed-length iterations (sprints), which provide focus and maintain steady progress toward project goals.
- Prioritize Working Software Over Comprehensive Documentation: Agile focuses on delivering functional software instead of excessive documentation, speeding up the development process.
- Continuous Improvement: Agile teams engage in retrospectives after each iteration to reflect on the process and identify ways to improve for the next sprint.
- Transparency: Agile ensures visibility into the project's progress and challenges through regular updates and reviews, building trust among stakeholders.
- Minimal Documentation: Agile emphasizes creating just enough documentation to support development, avoiding the overhead of excessive paperwork.
- Risk Management: Agile minimizes risks by delivering working software frequently and incorporating feedback, allowing teams to address issues early.
- Customer-Centric Focus: Agile ensures that the customer's needs are central to the development process, with continuous feedback ensuring the product aligns with expectations.

Disadvantages(Cons) of Agile Model:

- Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.
- Iterative and Incremental Development: Agile focuses on delivering small, functional portions of the project in iterations, allowing for continuous delivery and frequent feedback.
- Customer Collaboration: Close collaboration with customers throughout the
 - project ensures that the product aligns with their needs and expectations.
- Flexibility and Adaptability: Agile teams can adapt to changing requirements and make adjustments during the project, ensuring flexibility in response to customer needs or market changes.
- Continuous Delivery: Agile encourages the delivery of working software at the end of each iteration, providing stakeholders with frequent, tangible progress.
- Focus on Individuals and Interactions: Agile emphasizes the importance of communication and collaboration among team members, valuing people over processes and tools.
- Simplicity and Efficiency: Agile promotes delivering only essential features and avoids unnecessary complexity, leading to simpler and more efficient development.
- Frequent Communication and Feedback: Agile promotes regular communication within the team and with stakeholders to ensure everyone is aligned and feedback is incorporated early.
- Self-Organizing Teams: Agile teams are empowered to make decisions and take ownership of tasks, allowing for more autonomy and effective problem- solving.
- Prioritization of Features (Product Backlog): Agile prioritizes features based on business value, ensuring that the most important tasks are addressed first.
- Frequent Testing: Agile encourages regular testing to identify issues early, ensuring high-quality software and reducing the time spent fixing bugs later.
- Time-Boxed Sprints: Agile development is structured around fixed-length iterations (sprints), which provide focus and maintain steady progress toward project goals.
- Prioritize Working Software Over Comprehensive Documentation: Agile focuses on delivering functional software instead of excessive documentation, speeding up the development process.

- **Continuous Improvement:** Agile teams engage in retrospectives after each iteration to reflect on the process and identify ways to improve for the next sprint.
- **Transparency:** Agile ensures visibility into the project's progress and challenges through regular updates and reviews, building trust among stakeholders.
- **Minimal Documentation:** Agile emphasizes creating just enough documentation to support development, avoiding the overhead of excessive paperwork.
- **Risk Management:** Agile minimizes risks by delivering working software frequently and incorporating feedback, allowing teams to address issues early.
- **Customer-Centric Focus:** Agile ensures that the customer's needs are central to the development process, with continuous feedback ensuring the product aligns with expectations.
- **High Customer Satisfaction:** Agile prioritizes delivering value to the customer through working software and frequent communication, leading to higher satisfaction.
- **Increased Visibility:** Agile provides transparency through tools like burn-down charts and progress reports, ensuring that everyone stays informed on project progress.
- **Collaboration with Business Stakeholders:** Agile involves business stakeholders throughout the development process, ensuring that the product meets business requirements and objectives.

Limitation

- **Requires Experienced Team Members:** Agile is most effective when the team is highly skilled and experienced. Inexperienced teams may struggle to manage the iterative process and produce high-quality results.
- **Lack of Predictability:** The flexible nature of Agile makes it challenging to predict the final product and its timeline, which can be a disadvantage for projects requiring fixed deadlines and budgets.
- **Heavy Customer Involvement Needed:** Agile requires constant feedback and involvement from the customer. If the customer is unavailable or unable to provide timely feedback, it can slow down the development process.
- **Resource Intensive:** Agile demands significant time and resources for regular meetings, iterations, and reviews, which can lead to inefficiency and higher costs, especially for smaller teams.
- **Scope Creep:** The continuous adaptation of the project based on customer feedback can lead to scope creep, where the project's scope expands without clear boundaries, potentially leading to delays and resource strain.
- **Requires Cultural Shift:** Organizations accustomed to traditional methodologies may face resistance when transitioning to Agile, requiring significant effort to change their culture and practices.

8.2 Data Flow Diagram

A **data flow diagram (DFD)** for a real-time chat application provides a visual representation of how data, such as messages, user interactions, and system events, moves within the application. It employs symbols and labels to depict the flow of information, helping to understand the architecture and functionality of the chat system.

Components of DFD for a Chat Application:

- **Process:** In the context of a real-time chat application, processes represent the various functionalities or actions performed within the system. These may include sending messages, receiving messages, storing messages, managing user authentication, and handling system events such as notifications or updates.
- **Data Flow:** Data flows in a chat application represent the movement of messages and information between different components of the system. Arrows symbolize the flow of messages between users, chat rooms, and servers. Each data flow represents the transmission of a message or data packet from one entity to another.
- **Data Store:** Data stores in a chat application represent repositories where messages and user data are stored temporarily or permanently. This could include databases for storing user profiles, chat histories, and session information. Data stores ensure that messages are persisted and can be accessed across sessions.
- **Terminator:** Terminators in a real-time chat application represent external entities interacting with the system, such as users, devices, or external services. They initiate data exchanges with the chat application by sending or receiving messages. Terminators may also include third-party integrations or APIs used for authentication, message delivery, or other functionalities.

By creating a data flow diagram for a real-time chat application, we can visualize the flow of messages, user interactions, and system processes, helping to identify potential bottlenecks, optimize performance, and ensure smooth communication between users.

Levels of DFD

DFDs are created in different levels to represent systems at varying levels of detail:

1. Context Diagram (Level 0 DFD):

- Shows the system as a single process with its interactions with external entities.
- Provides a high-level overview of the system.
- Example: A library management system with interactions between users and the library.

2. **Level 1 DFD:**

- Breaks down the single process from the context diagram into sub-processes.
- Shows major functions of the system and their data flows.

3. **Level 2 DFD and Beyond:**

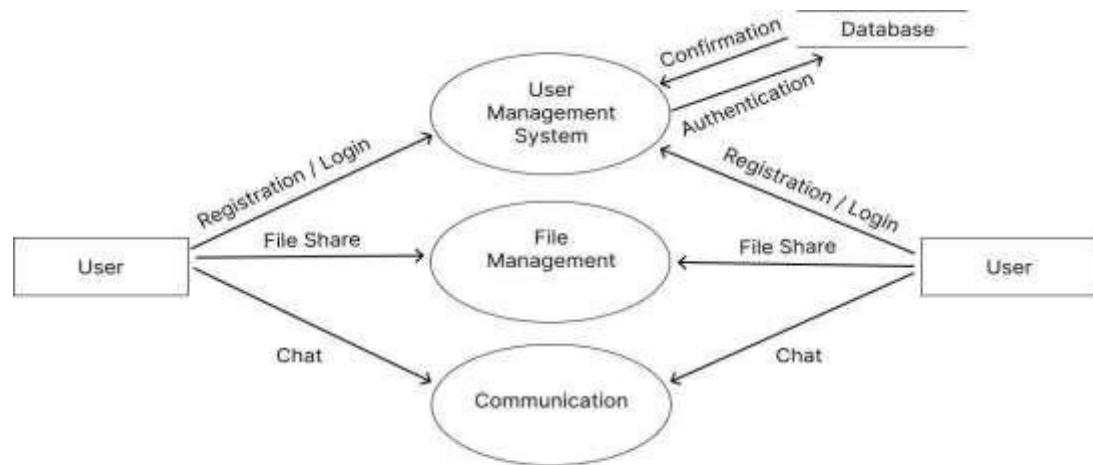
- Provides further decomposition of the processes into smaller, more detailed processes.
- Continues until the desired level of granularity is achieved.

8.2.1 Context Level DFD



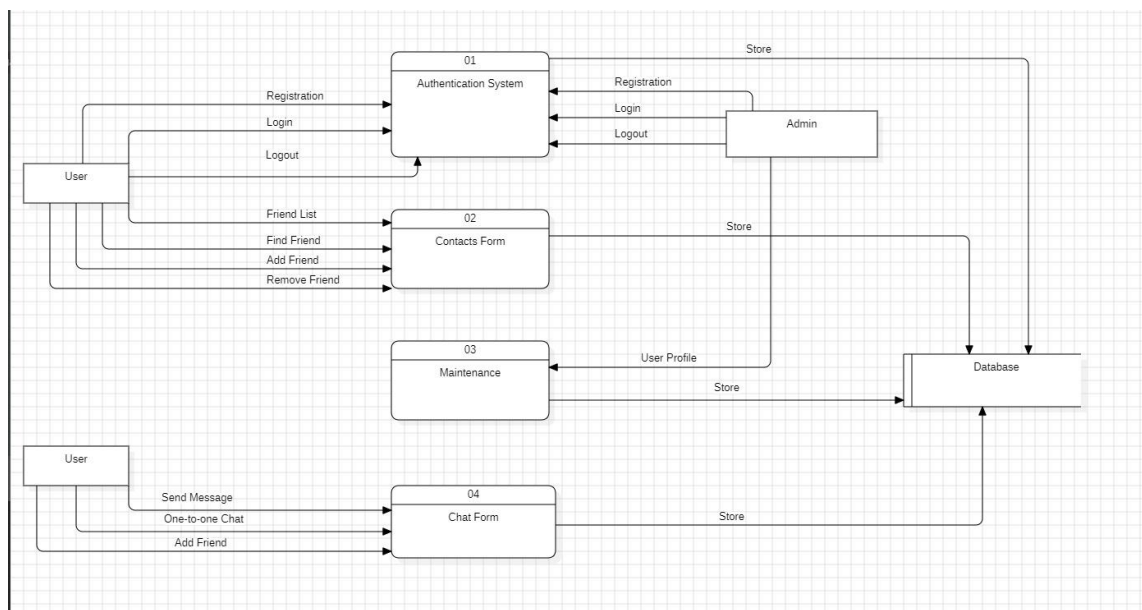
DFD Level 0

8.2.2 1 Level DFD

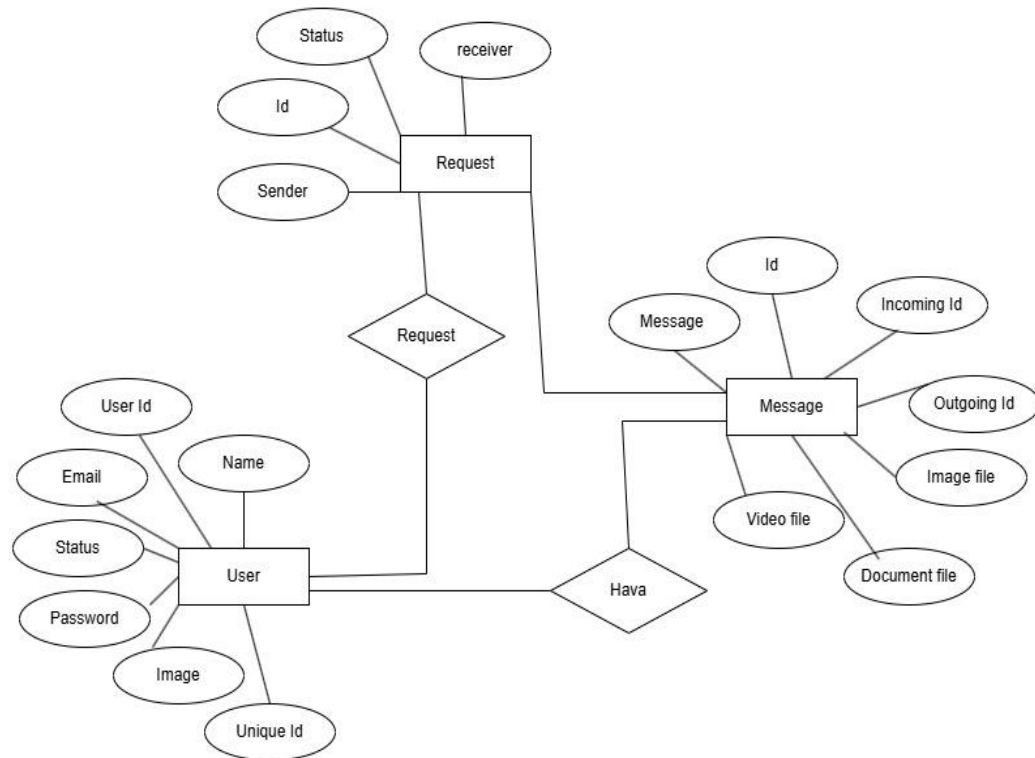


DFD Level 1

8.2.2 2 Level DFD



8.3 Entity Relationship Diagram



8.4 Normalization

Normalization, is the process of organizing the columns and tables of a relational database to reduce data redundancy and improve data integrity. Normalization is also the process simplifying the design of a database so that it achieves the optimal structure.

Normalization involves arranging attributes in relations based on dependencies between attributes, ensuring that the dependencies are properly enforced by database integrity constraints. Normalization is accomplished by applying some formal rules either by a process of synthesis or decomposition. Synthesis creates a normalized database design based on a known set of dependencies. Decomposition takes an existing database design and improved it based on the known set of dependencies.

First Normal Form(1NF)

As per First Normal Form, no two rows of data must contain repeating group of information i.e. each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows and each row should have a primary key that distinguishes it as unique. The primary key usually a single column, but sometimes more than one column can be combined to create a single primary key.

Second Normal Form(2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails Second Normal Form.

Third Normal Form(3NF)

Third Normal Form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non- prime attribute is determined by another no-prime attribute. So this transitive function dependency should be remove from the table and also the table must be in second normal form. For example, consider a table with following fields.

Boyce And Codd Normal Form(BCNF)

Boyce and Codd Normal Form (BCNF) is a higher version of the third normal form. This form deals with certain type of anomaly that is not handled by 3nf. A 3nf table which does not have multiple overlapping super keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form.
- For each functional dependency (X→Y), X should be a super key.

Users Table:

Columns: id (PK), name, username, bio, avatar, password, created_at, updated_at.

Chats Table:

Columns: id (PK), name, GroupChat, members (FK references Users.id), created_at, updated_at.

Messages Table:

Columns: id (PK), sender, chatId (FK references Chats.id), content, attachments, created_at, updated_at.

Requests Table:

Columns: id (PK), status, sender (FK reference Users.id), receiver (FK reference users.id), created_at, updated_at

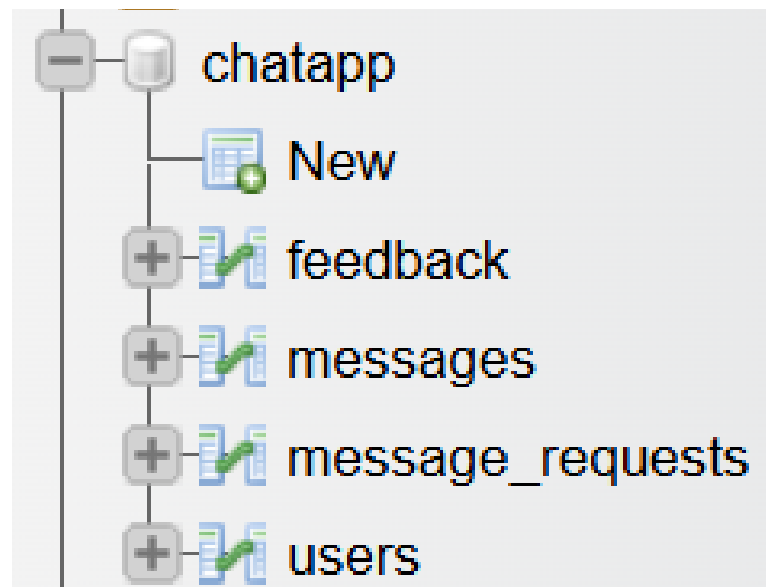
Each table has a primary key (**id for most tables**) to uniquely identify each record.

Foreign keys (**members, chatId, creator, sender, receiver**) are used to establish relationships between tables. These keys reference the primary keys of other tables. Redundancy is reduced, and the data is organized to minimize anomalies. Remember, normalization is about striking a balance between reducing redundancy and simplifying queries. Depending on the specific requirements of your application, you may need to adjust the structure accordingly.

8.5 Database Table

Database:

The application uses four primary tables: feedback, message, message_requests, and users. Each table serves a specific purpose and collectively supports the functionalities of the chat platform.

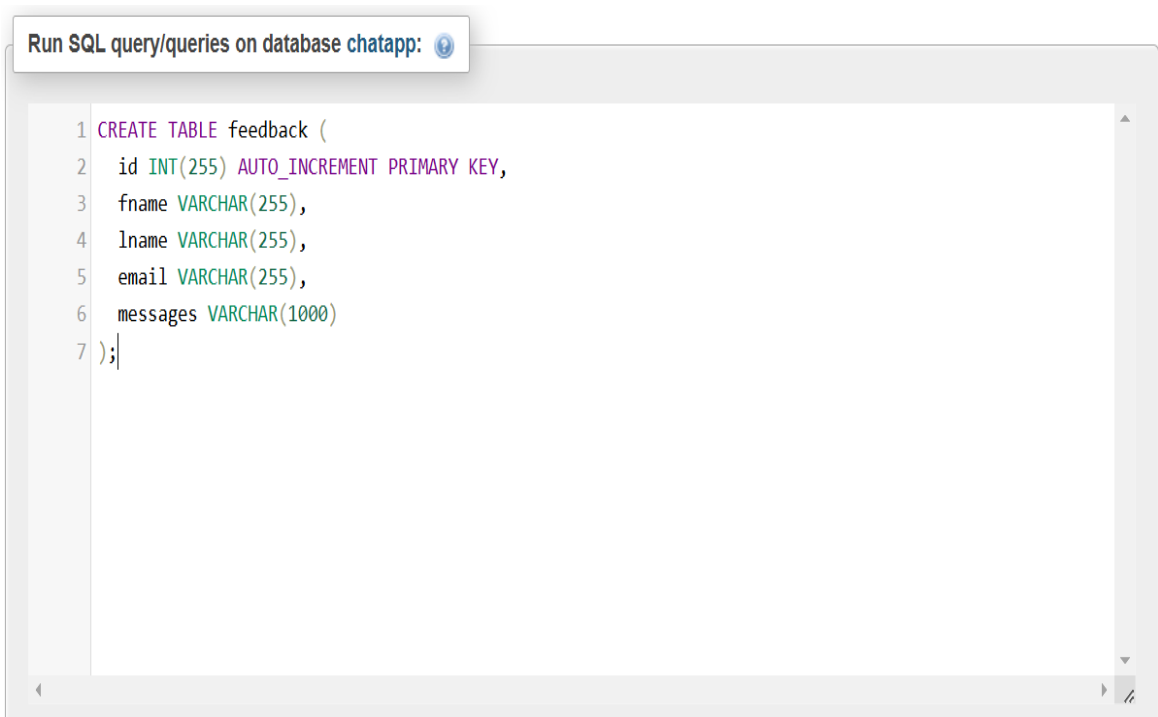


Feedback Table (feedback):

Purpose: Stores feedback provided by users about their experience with the application.

Key Columns:

- **id (Primary Key):** A unique identifier for each feedback entry.
- **fname & lname:** Store the name of the users.
- **email:** Store the email of the users.
- **meassage:** The text of the feedback



The screenshot shows a web-based SQL query editor. At the top, there is a button labeled "Run SQL query/queries on database chatapp:" with a blue circular icon. Below the button is a text area containing the following SQL code:

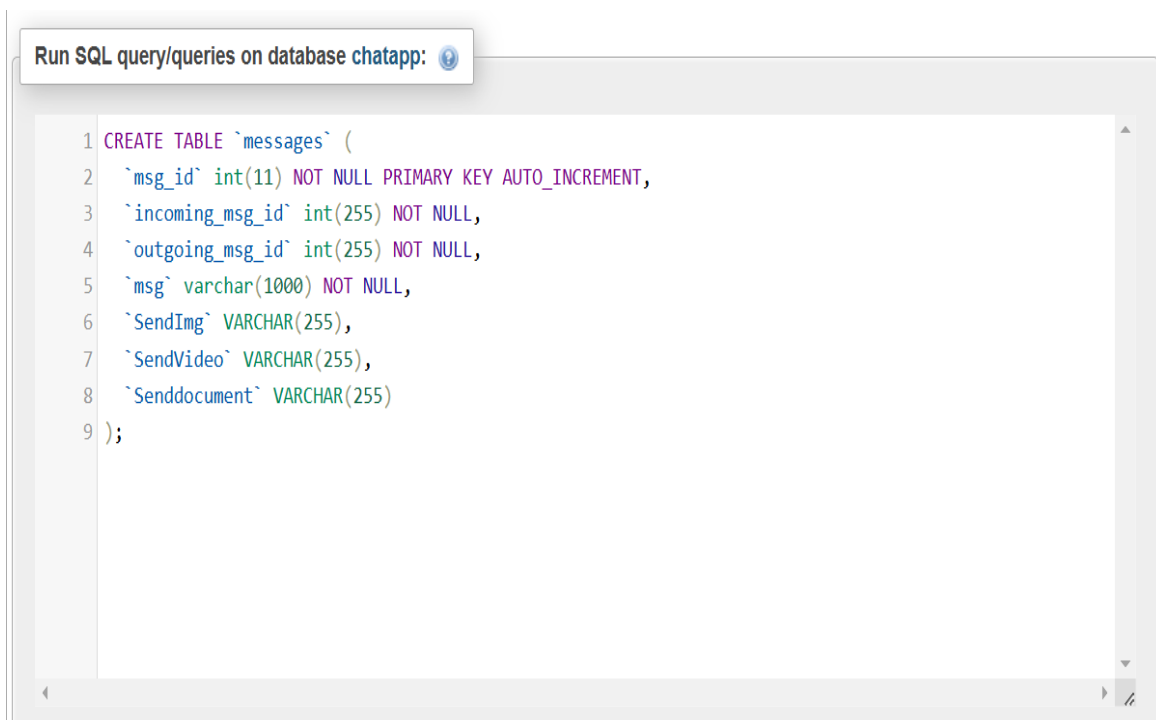
```
1 CREATE TABLE feedback (  
2   id INT(255) AUTO_INCREMENT PRIMARY KEY,  
3   fname VARCHAR(255),  
4   lname VARCHAR(255),  
5   email VARCHAR(255),  
6   messages VARCHAR(1000)  
7 );
```


Message Table (message):

Purpose: Stores the messages exchanged between users.

Key Columns:

- **msg_id (Primary Key):** A unique identifier for each message.
- **outgoing_msg_id:** Foreign key referencing the user_id of the sender.
- **incoming_msg_id:** Foreign key referencing the user_id of the receiver.
- **msg:** Store the text of the message.
- **SendImg:** Store file path for images files.
- **SendVideo:** Store file path for videos files.
- **Senddocument:** Store file path for shared files.



The screenshot shows a SQL query editor window titled "Run SQL query/queries on database chatapp:". The query is as follows:

```
1 CREATE TABLE `messages` (  
2   `msg_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,  
3   `incoming_msg_id` int(255) NOT NULL,  
4   `outgoing_msg_id` int(255) NOT NULL,  
5   `msg` varchar(1000) NOT NULL,  
6   `SendImg` VARCHAR(255),  
7   `SendVideo` VARCHAR(255),  
8   `Senddocument` VARCHAR(255)  
9 );
```

Requests Table (message_requests):

Purpose: Manages message requests sent by users who are not yet connected.

Key Columns:

- **id (Primary Key):** A unique identifier for each request.
- **sender_id:** Foreign key referencing the user_id of the sender.
- **receiver_id:** Foreign key referencing the user_id of the receiver.
- **status:** Indicates whether the request is pending, accepted, or rejected.
- **request_date:** The date when the message request was st.
- **created_at:** The date and time when the message request was sent.
- **update_at:** The date and time when users update the request status.
- **UNIQUE:** The UNIQUE constraint ensures that sender_id and receiver_id are different

Run SQL query/queries on database chatapp: ⓘ

```
1 CREATE TABLE message_requests (  
2   id INT AUTO_INCREMENT PRIMARY KEY,  
3   sender_id INT NOT NULL,  
4   receiver_id INT NOT NULL,  
5   status ENUM('pending', 'accepted', 'rejected') DEFAULT 'pending',  
6   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
7   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
8   UNIQUE (sender_id, receiver_id)  
9 );
```

Users Table (users):

Purpose: Stores user information and manages authentication.

Key Columns:

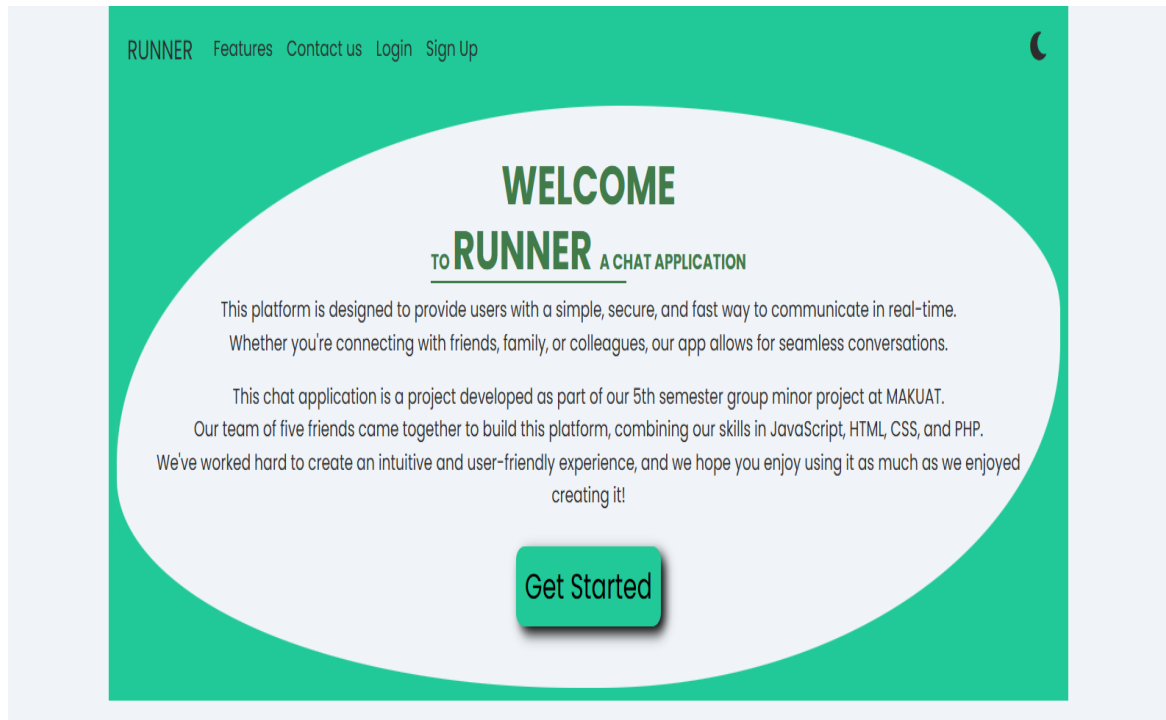
- **user_id (Primary Key):** A unique identifier for each user.
- **unique_id:** Assign a unique number for each user for identifier each users.
- **fname:** Name of the users.
- **lname:** Last name of the users.
- **email:** The email address used for registration and communication.
- **password:** The hashed password for secure authentication.
- **img:** Store users image.
- **status:** Indicates if the user is online or offline.

Run SQL query/queries on database chatapp: ⓘ

```
1 CREATE TABLE `users` (  
2   `user_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,  
3   `unique_id` int(255) NOT NULL,  
4   `fname` varchar(255) NOT NULL,  
5   `lname` varchar(255) NOT NULL,  
6   `email` varchar(255) NOT NULL,  
7   `password` varchar(255) NOT NULL,  
8   `img` varchar(255) NOT NULL,  
9   `status` varchar(255) NOT NULL  
10 );
```

9 Code and Implementation

Home Page:



Source Code:

Index.php:

```
<?php include_once "header.php"; ?>
```

```
<body>
```

```
<div class="container">
```

```
<nav class="navbar navbar-expand-lg navbarDesign " id="navbarSection">
```

```
<div class="container-fluid containerdiv">
```

```
<a class="navbar-brand brandName" href="#">RUNNER</a>
```

```
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarText"
```

```
aria-controls="navbarText" aria-expanded="false" aria-label="Toggle navigation">
```

```

        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarText">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
                <a class="nav-link hover-underline-animation"
href="#featuresSection">Features</a>
            </li>
            <li class="nav-item">
                <a class="nav-link hover-underline-animation"
href="#contactUs">Contact us</a>
            </li>
            <li class="nav-item">
                <a class="nav-link hover-underline-animation"
href="login.php">Login</a>
            </li>
            <li class="nav-item">
                <a class="nav-link hover-underline-animation" href="signup.php">Sign
Up</a>
            </li>
        </ul>
        <span class="navbar-text">
            <label for="theme-toggle" id="theme-toggle-icon">
                <i class="fa-solid fa-moon"></i>
            </label>
            <input type="checkbox" class="theme-toggle" id="theme-toggle">
        </span>
    </div>
</div>
</nav>
<div class="hero">
    <div class="heroClass">
        <h1 id="heroHiding">WELCOME <span> <br>TO </span> RUNNER <span> A
CHAT APPLICATION<br></span></h1>
        <p>This platform is designed to provide users with a simple, secure, and fast way
to communicate in
            real-time.<br> Whether you're connecting with friends, family, or colleagues,
our app allows for
            seamless conversations.</p>
    </div>

```

<p>This chat application is a project developed as part of our 5th semester group minor project at

MAKUAT.
 Our team of five friends came together to build this platform, combining our skills in

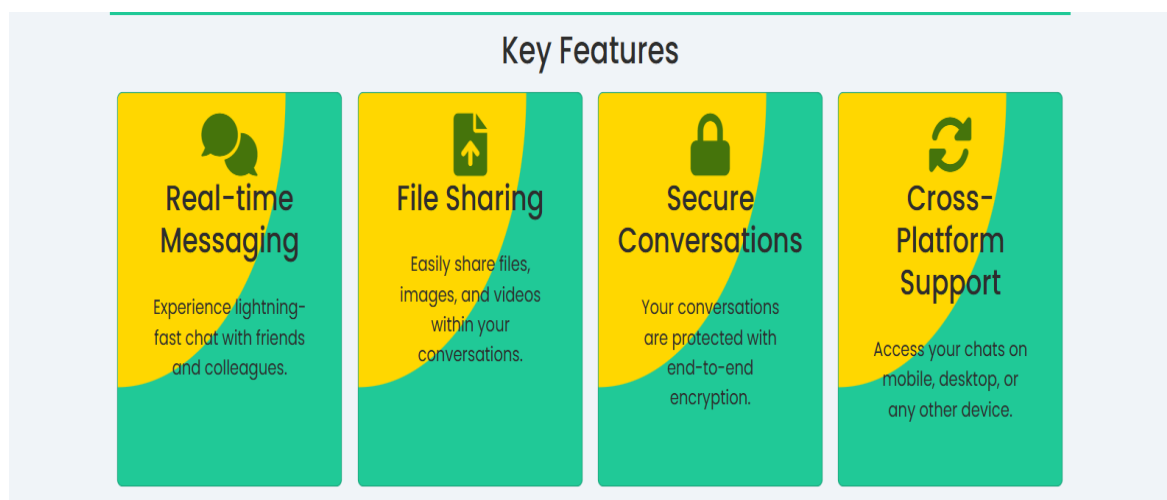
JavaScript, HTML, CSS, and PHP.
 We've worked hard to create an intuitive and user-friendly

experience, and we hope you enjoy using it as much as we enjoyed creating it!</p>

Get Started

</div>

</div>



<!-- features card -->

<section class="features" id="featuresSection">

<h2 id="featuresHeader">Key Features</h2>

<div class="featuresBox">

<div class="card featuresCard card-all">

<div class="card-body card-index">

<i class="fas fa-comments"></i>

<h3 class="card-title">Real-time Messaging</h3>

<p class="card-body">Experience lightning-fast chat with friends and colleagues.</p>

</div>

</div>

<div class="card featuresCard card-all">

<div class="card-body card-index">

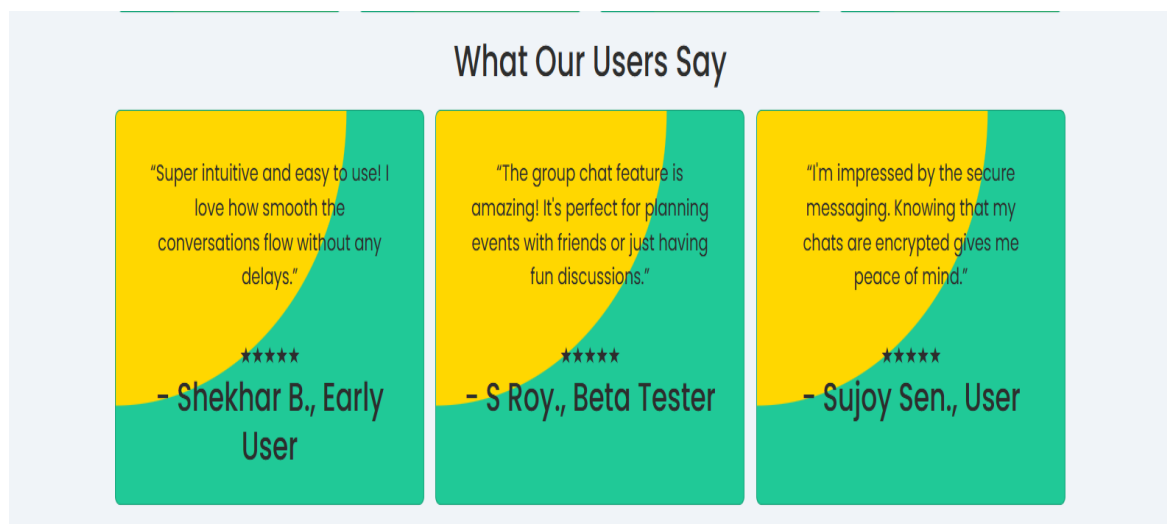
<i class="fas fa-file-upload"></i>

```

        <h3 class="card-title">File Sharing</h3>
        <p class="card-body">Easily share files, images, and videos within your
conversations.</p>
    </div>
</div>

<div class="card featuresCard card-all">
    <div class="card-body card-index">
        <i class="fas fa-lock"></i>
        <h3 class="card-title">Secure Conversations</h3>
        <p class="card-body">Your conversations are protected with end-to-end
encryption.</p>
    </div>
</div>
<div class="card featuresCard card-all">
    <div class="card-body card-index">
        <i class="fas fa-sync-alt"></i>
        <h3 class="card-title">Cross-Platform Support</h3>
        <p class="card-body">Access your chats on mobile, desktop, or any other
device.</p>
    </div>
</div>
</div>
</div>
</section>

```



```

<!-- Users Say -->
<section class="features" id="featuresSection">
    <h2 id="featuresHeader">What Our Users Say</h2>

```

```

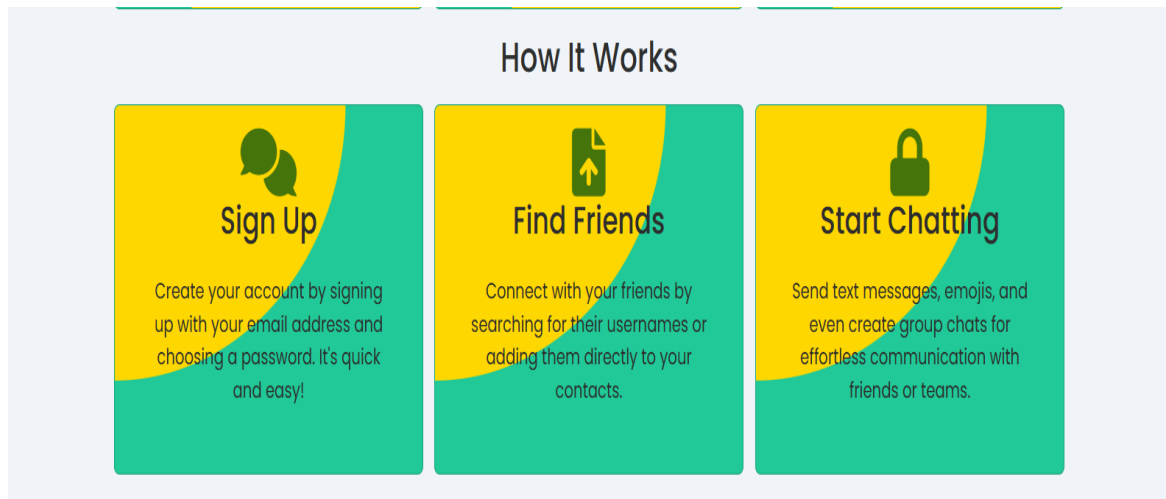
<div class="featuresBox">
  <div class="card featuresCard card-all">
    <div class="card-body card-index">
      <p class="card-body">“Super intuitive and easy to use! I love how smooth
the conversations flow
      without any delays.”</p>
      <div class="stars">
        <span>★</span><span>★</span><span>★</span><span>★</span><span>★</span>
</span><span>★</span></div>
      <h3 class="card-title">- Shekhar B., Early User</h3>
    </div>
  </div>

  <div class="card featuresCard card-all">
    <div class="card-body card-index">
      <p class="card-body">“The group chat feature is amazing! It's perfect for
planning events with
      friends or just having fun discussions.”</p>

      <div class="stars">
        <span>★</span><span>★</span><span>★</span><span>★</span><span>★</span>
</span><span>★</span></div>
      <h3 class="card-title">- S Roy., Beta Tester</h3>
    </div>
  </div>

  <div class="card featuresCard card-all">
    <div class="card-body card-index">
      <p class="card-body">“I'm impressed by the secure messaging. Knowing that
my chats are encrypted
      gives me peace of mind.”</p>
      <div class="stars">
        <span>★</span><span>★</span><span>★</span><span>★</span><span>★</span>
</span><span>★</span></div>
      <h3 class="card-title">- Sujoy Sen., User</h3>
    </div>
  </div>
</div>
</section>

```

```

<!-- How chat works cards -->
<section class="features" id="featuresSection">
  <h2 id="featuresHeader">How It Works</h2>
  <div class="featuresBox">
    <div class="card featuresCard card-all">
      <div class="card-body card-index">
        <i class="fas fa-comments"></i>
        <h3 class="card-title">Sign Up</h3>
        <p class="card-body">Create your account by signing up with your email
address and choosing a
          password. It's quick and easy!</p>
      </div>
    </div>
    <div class="card featuresCard card-all">
      <div class="card-body card-index">
        <i class="fas fa-file-upload"></i>
        <h3 class="card-title">Find Friends</h3>
        <p class="card-body">Connect with your friends by searching for their
usernames or adding them
          directly to your contacts.</p>
      </div>
    </div>
    <div class="card featuresCard card-all">
      <div class="card-body card-index">
        <i class="fas fa-lock"></i>
        <h3 class="card-title">Start Chatting</h3>

```

```

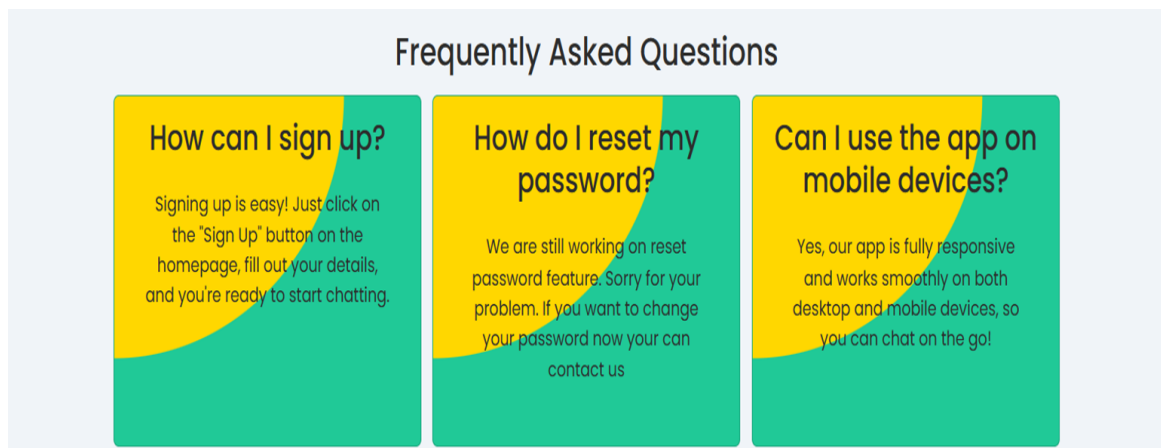
        <p class="card-body">Send text messages, emojis, and even create group
        chats for effortless
        communication with friends or teams.</p>
    </div>
</div>

```

```

    </div>
</section>

```



```

<!-- faq cards -->
<section class="features" id="featuresSection">
    <h2 id="featuresHeader">Frequently Asked Questions</h2>
    <div class="featuresBox">
        <div class="card featuresCard card-all">
            <div class="card-body card-index">
                <h3 class="card-title">How can I sign up?</h3>
                <p class="card-body">Signing up is easy! Just click on the "Sign Up" button
                on the homepage,
                fill out your details, and you're ready to start chatting.</p>
            </div>
        </div>
        <div class="card featuresCard card-all">
            <div class="card-body card-index">
                <h3 class="card-title">How do I reset my password?</h3>
                <p class="card-body">We are still working on reset password feature. Sorry
                for your problem. If
                you want to change your password now your can <a class="nav-link hover-
                underline-animation"
                href="#contactUs">contact us</a></p>
            </div>
        </div>
    </div>

```

```

    </div>
</div>

<div class="card featuresCard card-all">
  <div class="card-body card-index">
    <h3 class="card-title">Can I use the app on mobile devices?</h3>
    <p class="card-body">Yes, our app is fully responsive and works smoothly
on both desktop and
      mobile devices, so you can chat on the go!</p>
    </div>
  </div>

</div>
</section>

```

Please give your feedback here

First Name

Last Name

First name

Last name

Email Address

Enter your email

Message

Submit

Back to top

```

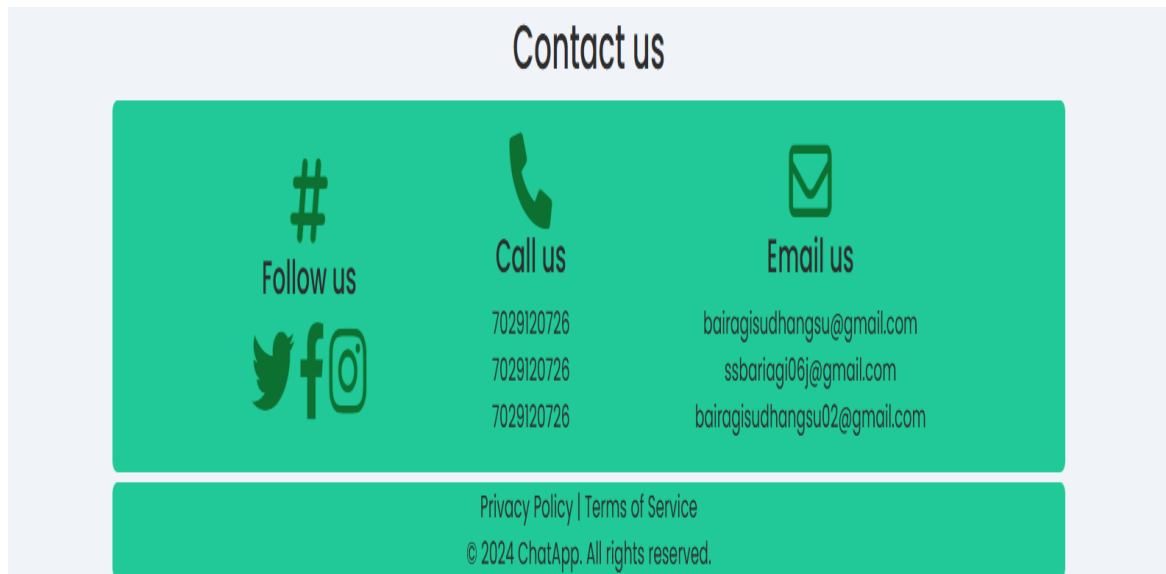
<section class="form feedbackForm" id = "feedbackForm">

    <div class="feedbackHeader">
        <h2>Please give your feedback here</h2>
    </div>

    <div class="feedbackFormAni">
        <form action="#" method="POST" id="feedbackForm" enctype="multipart/form-
data" autocomplete="off">
            <div class="error-text"></div>
            <div class="name-details">
                <div class="field input">
                    <label>First Name</label>
                    <input type="text" name="fname" placeholder="First name" required>
                </div>
                <div class="field input">
                    <label>Last Name</label>
                    <input type="text" name="lname" placeholder="Last name" required>
                </div>
            </div>
            <div class="field input">
                <label>Email Address</label>
                <input type="text" name="email" placeholder="Enter your email" required>
            </div>
            <div class="field input">
                <label for="message" class="form-label">Message</label>
                <textarea id="message" name="message" required></textarea>
            </div>
            <div class="field button">
                <input type="submit" name="submit" value="Submit">
            </div>
        </form>
    </div>
</section>

<footer>
    <div class="backToTop">
        <a class="hover-underline-animation" href="#navbarSection">Back to top</a>
    </div>

```



```

<h2 class="contactUs" id="contactUs">Contact us</h2>
<div class="contact">

  <div class="contactSection">
    <i class="fa-regular fa-hashtag"></i>
    <h4>Follow us</h4>
    <div class="mediaIcon">
      <a class="hover-underline-animation" href="https://twitter.com"
target="_blank"> <i class="fab fa-twitter"></i> </a>
      <a class="hover-underline-animation" href="https://facebook.com"
target="_blank"><i class="fab fa-facebook-f"></i> </a>
      <a class="hover-underline-animation" href="https://instagram.com"
target="_blank"><i class="fab fa-instagram"></i></a>
    </div>
  </div>

  <div class="contactSection">
    <i class="fa-solid fa-phone"></i>
    <h4>Call us</h4>
    <a>7029120726</a>
    <a>7029120726</a>
    <a>7029120726</a>
  </div>

  <div class="contactSection">
    <i class="fa-regular fa-envelope"></i>
    <h4>Email us</h4>

```

```

        <a class="hover-underline-animation" href="bairagisudhangsu@gmail.com"
          target="_blank">bairagisudhangsu@gmail.com</a>
        <a class="hover-underline-animation" href="ssbariagi06j@gmail.com"
          target="_blank">ssbariagi06j@gmail.com</a>
        <a class="hover-underline-animation" href="bairagisudhangsu02@gmail.com"
          target="_blank">bairagisudhangsu02@gmail.com</a>
      </div>
    </div>

    <div class="footerCopyRights">
      <p>
        <a class="hover-underline-animation" href="/privacy">Privacy Policy</a> |
        <a class="hover-underline-animation" href="/terms">Terms of Service</a>
      </p>
      <p>© 2024 ChatApp. All rights reserved.</p>
    </div>
  </div>
  <script src="javascript/feedback.js"></script>
  <script src="javascript/themeMode.js"></script>

</body>
</html>

```

javascript/feedback.js:

```

const form = document.querySelector(".feedbackForm form"),
  continueBtn = form.querySelector(".button input"),
  errorText = form.querySelector(".error-text");

form.onsubmit = (e) => {
  e.preventDefault();
}

continueBtn.onclick = () => {
  let xhr = new XMLHttpRequest();
  xhr.open("POST", "php/feedback.php", true);

```

```

xhr.onload = () => {
  if (xhr.readyState === XMLHttpRequest.DONE) {
    if (xhr.status === 200) {
      let data = xhr.response;
      if (data === "success") {
        location.href = "index.php";
        alert("Thank You for your feedback ! Your response was submitted successfully");
      } else {
        errorText.style.display = "block";
        errorText.textContent = data;
      }
    }
  }
}
let formData = new FormData(form);
xhr.send(formData);
}

```

javascript/themeMode.js:

```

document.addEventListener('DOMContentLoaded', () => {
  const themeToggle = document.getElementById('theme-toggle');
  const body = document.body;
  const themeIcon = document.querySelector('#theme-toggle-icon i');
  if (localStorage.getItem('dark-mode') === 'enabled') {
    body.classList.add('dark-mode');
    themeIcon.classList.replace('fa-moon', 'fa-sun');
  }
  themeToggle.addEventListener('change', () => {
    body.classList.toggle('dark-mode');

    if (body.classList.contains('dark-mode')) {
      localStorage.setItem('dark-mode', 'enabled');
      themeIcon.classList.replace('fa-moon', 'fa-sun');
    } else {
      localStorage.setItem('dark-mode', 'disabled');
      themeIcon.classList.replace('fa-sun', 'fa-moon');
    }
  });
});

```

animationCss.js:

```
document.addEventListener("DOMContentLoaded", () => {
  const cardRipple = document.querySelector(".Mybtn,.form form .button,.container .form
.feedbackFormAni form .button input");

  cardRipple.addEventListener("mouseover", (e) => {
    const x = e.pageX - cardRipple.offsetLeft;
    const y = e.pageY - cardRipple.offsetTop;

    cardRipple.style.setProperty("--posX", x + "px");
    cardRipple.style.setProperty("--posY", y + "px");
  });

  // ripple animation on card
  const cards = document.querySelectorAll('.featuresCard,.feedbackForm');
  let lastScrollY = window.scrollY;

  function isScrollingDown() {
    const currentScrollY = window.scrollY;
    const isScrollingDown = currentScrollY > lastScrollY;
    lastScrollY = currentScrollY;
    return isScrollingDown;
  }

  function isInViewport(element) {
    const rect = element.getBoundingClientRect();
    return rect.top < window.innerHeight && rect.bottom > 0;
  }

  function applyRippleEffect() {
    const scrollingDown = isScrollingDown();

    cards.forEach(card => {
      if (isInViewport(card)) {

        if (!card.classList.contains('scroll-ripple')) {
          const ripple = card.querySelector('::before');

          if (scrollingDown) {

            card.style.setProperty('--ripple-x', '0%');
            card.style.setProperty('--ripple-y', '0%');
          } else {

            card.style.setProperty('--ripple-x', '100%');
            card.style.setProperty('--ripple-y', '100%');
          }
        }
      }
    });
  }
});
```



```
        card.classList.add('scroll-ripple');
    }
} else {

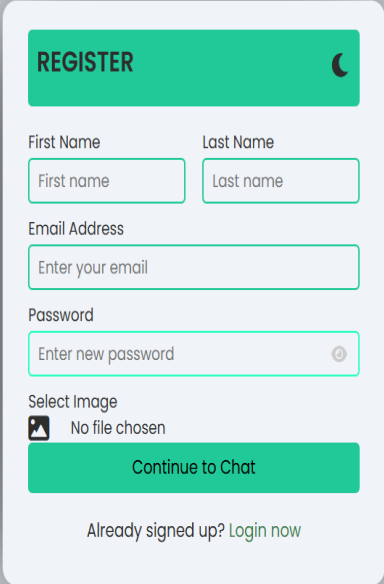
    card.classList.remove('scroll-ripple');
}
});
}

// Listen for the scroll event and apply the ripple effect
window.addEventListener('scroll', applyRippleEffect);

// Apply ripple effect on page load to handle already visible cards
applyRippleEffect();

});
```

Sign Up:



REGISTER

First Name Last Name

First name Last name

Email Address

Enter your email

Password

Enter new password

Select Image

No file chosen

Continue to Chat

Already signed up? [Login now](#)

Signup.php:

```
<?php
session_start();
if(isset($_SESSION['unique_id'])){
    header("location: users.php");
}
?>

<?php include_once "header.php"; ?>
<body>

<div class="wrapper">
    <section class="form signup">
        <div class="headerSection">
            <header>REGISTER</header>
            <label for="theme-toggle" id="theme-toggle-icon">
                <i class="fa-solid fa-moon"></i>
            </label>
            <input type="checkbox" class="theme-toggle" id="theme-toggle">
        </div>
```

```

<form action="#" method="POST" enctype="multipart/form-data" autocomplete="off">
  <div class="error-text"></div>
  <div class="name-details">
    <div class="field input">
      <label>First Name</label>
      <input type="text" name="fname" placeholder="First name" required>
    </div>
    <div class="field input">
      <label>Last Name</label>
      <input type="text" name="lname" placeholder="Last name" required>
    </div>
  </div>
  <div class="field input">
    <label>Email Address</label>
    <input type="text" name="email" placeholder="Enter your email" required>
  </div>
  <div class="field input">
    <label>Password</label>
    <input type="password" name="password" placeholder="Enter new password"
required>
    <i class="fa-solid fa-eye"></i>
  </div>
  <div class="field image">
    <label>Select Image <input type="file" name="image" class = "file-input"
accept="image/x-png,image/gif,image/jpeg,image/jpg" required><i class="fa-solid fa-
image"><span class="file-name">No file chosen</span></i></label>
  </div>
  <div class="field button">
    <input type="submit" name="submit" value="Continue to Chat">
  </div>
</form>
  <div class="link">Already signed up? <a class="hover-underline-animation"
href="login.php">Login now</a></div>
</section>
</div>

<script src = "javascript/fileInput.js"></script>
<script src="javascript/themeMode.js"></script>
<script src="javascript/pass-show-hide.js"></script>
<script src="javascript/signup.js"></script>

</body>
</html>

```

php/signup.php:

```
<?php
    session_start();
    include_once "config.php";
    $fname = mysqli_real_escape_string($conn, $_POST['fname']);
    $lname = mysqli_real_escape_string($conn, $_POST['lname']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $password = mysqli_real_escape_string($conn, $_POST['password']);
    if(!empty($fname) && !empty($lname) && !empty($email) && !empty($password)){
        if(filter_var($email, FILTER_VALIDATE_EMAIL)){
            $sql = mysqli_query($conn, "SELECT * FROM users WHERE email = '{$email}'");
            if(mysqli_num_rows($sql) > 0){
                echo "$email - This email already exist!";
            }else{
                if(isset($_FILES['image'])){
                    $img_name = $_FILES['image']['name'];
                    $img_type = $_FILES['image']['type'];
                    $tmp_name = $_FILES['image']['tmp_name'];

                    $img_explode = explode('.', $img_name);
                    $img_ext = end($img_explode);

                    $extensions = ["jpeg", "png", "jpg"];
                    if(in_array($img_ext, $extensions) === true){
                        $types = ["image/jpeg", "image/jpg", "image/png"];
                        if(in_array($img_type, $types) === true){
                            $time = time();
                            $new_img_name = $time.$img_name;
                            if(move_uploaded_file($tmp_name, "images/".$new_img_name)){
                                $ran_id = rand(time(), 100000000);
                                $status = "Active now";
                                $encrypt_pass = md5($password);
                                $insert_query = mysqli_query($conn, "INSERT INTO users (unique_id,
                                fname, lname, email, password, img, status)
                                VALUES ({ $ran_id }, '{$fname}', '{$lname}', '{$email}',
                                '{$encrypt_pass}', '{$new_img_name}', '{$status}')");
                                if($insert_query){
                                    $select_sql2 = mysqli_query($conn, "SELECT * FROM users
                                WHERE email = '{$email}'");
                                    if(mysqli_num_rows($select_sql2) > 0){
                                        $result = mysqli_fetch_assoc($select_sql2);
                                        $_SESSION['unique_id'] = $result['unique_id'];
                                        echo "success";
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```

        }else{
            echo "This email address not Exist!";
        }
    }else{
        echo "Something went wrong. Please try again!";
    }
}
}else{
    echo "Please upload an image file - jpeg, png, jpg";
}
}else{
    echo "Please upload an image file - jpeg, png, jpg";
}
}
}
}else{
    echo "$email is not a valid email!";
}
}
}else{
    echo "All input fields are required!";
}
?>

```

javascript/signup.js:

```

const form = document.querySelector(".signup form"),
continueBtn = form.querySelector(".button input"),
errorText = form.querySelector(".error-text");

form.onsubmit = (e)=>{
    e.preventDefault();
}

continueBtn.onclick = ()=>{
    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/signup.php", true);
    xhr.onload = ()=>{
        if(xhr.readyState === XMLHttpRequest.DONE){
            if(xhr.status === 200){
                let data = xhr.response;
                if(data === "success"){
                    location.href="users.php";
                }
            }
        }
    }
}

```

```

        }else{
            errorText.style.display = "block";
            errorText.textContent = data;
        }
    }
}
}
let formData = new FormData(form);
xhr.send(formData);
}

```

Javascript/fileInput.js:

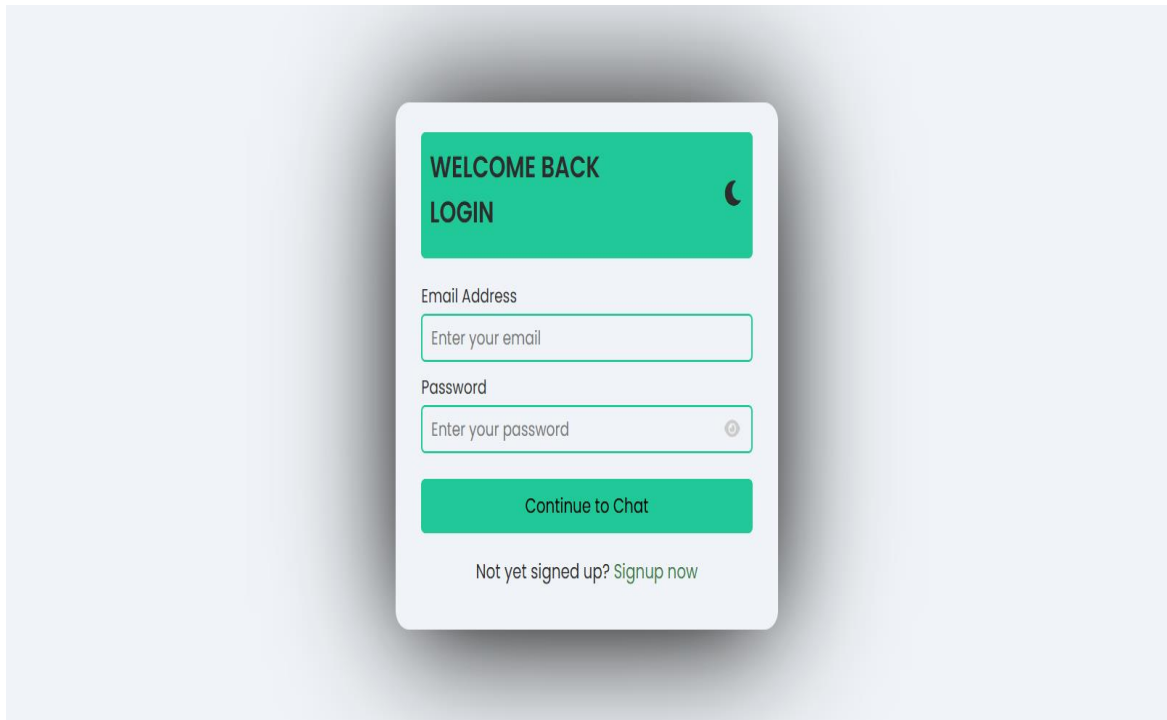
```

const fileInput = document.querySelector('.file-input');
const fileNameDisplay = document.querySelector('.file-name');

fileInput.addEventListener('change', function () {
    if (fileInput.files.length > 0) {
        fileNameDisplay.textContent = fileInput.files[0].name;
    } else {
        fileNameDisplay.textContent = 'No file chosen';
    }
});

```

Log In:



login.php:

```
<?php
    session_start();
    if(isset($_SESSION['unique_id'])){
        header("location: users.php");
    }
?>

<?php include_once "header.php"; ?>
<body>
    <div class="wrapper">
        <section class="form login">
            <div class="headerSection">
                <header>WELCOME BACK <BR>LOGIN</BR></header>
                <label for="theme-toggle" id="theme-toggle-icon">
                    <i class="fa-solid fa-moon"></i>
                </label>
                <input type="checkbox" class="theme-toggle" id="theme-toggle">
            </div>
            <form action="#" method="POST" enctype="multipart/form-data" autocomplete="off">
                <div class="error-text"></div>
```

```

<div class="field input">
  <label>Email Address</label>
  <input type="text" name="email" placeholder="Enter your email" required>
</div>
<div class="field input">
  <label>Password</label>
  <input type="password" name="password" placeholder="Enter your password"
required>
  <i id="themeMode" class="fa-solid fa-eye"></i>
</div>
<div class="field button">
  <input type="submit" name="submit" value="Continue to Chat">
</div>
</form>
<div class="link">Not yet signed up? <a class="hover-underline-animation"
href="signup.php">Signup now</a></div>
</section>
</div>

<script src="javascript/pass-show-hide.js"></script>
<script src="javascript/login.js"></script>
<script src="javascript/themeMode.js"></script>
</body>
</html>

```

php/login.php:

```

<?php
  session_start();
  include_once "config.php";
  $email = mysqli_real_escape_string($conn, $_POST['email']);
  $password = mysqli_real_escape_string($conn, $_POST['password']);
  if(!empty($email) && !empty($password)){
    $sql = mysqli_query($conn, "SELECT * FROM users WHERE email = '{ $email }'");
    if(mysqli_num_rows($sql) > 0){
      $row = mysqli_fetch_assoc($sql);
      $user_pass = md5($password);
      $enc_pass = $row['password'];
      if($user_pass === $enc_pass){
        $status = "Active now";
        $sql2 = mysqli_query($conn, "UPDATE users SET status = '{ $status }' WHERE
unique_id = { $row['unique_id'] }");

```



```

        if($sql2){
            $_SESSION['unique_id'] = $row['unique_id'];
            echo "success";
        }else{
            echo "Something went wrong. Please try again!";
        }
    }else{
        echo "Email or Password is Incorrect!";
    }
}
}else{
    echo "$email - This email not Exist!";
}
}
}else{
    echo "All input fields are required!";
}
}
?>

```

javascript/pass-show-hide.js:

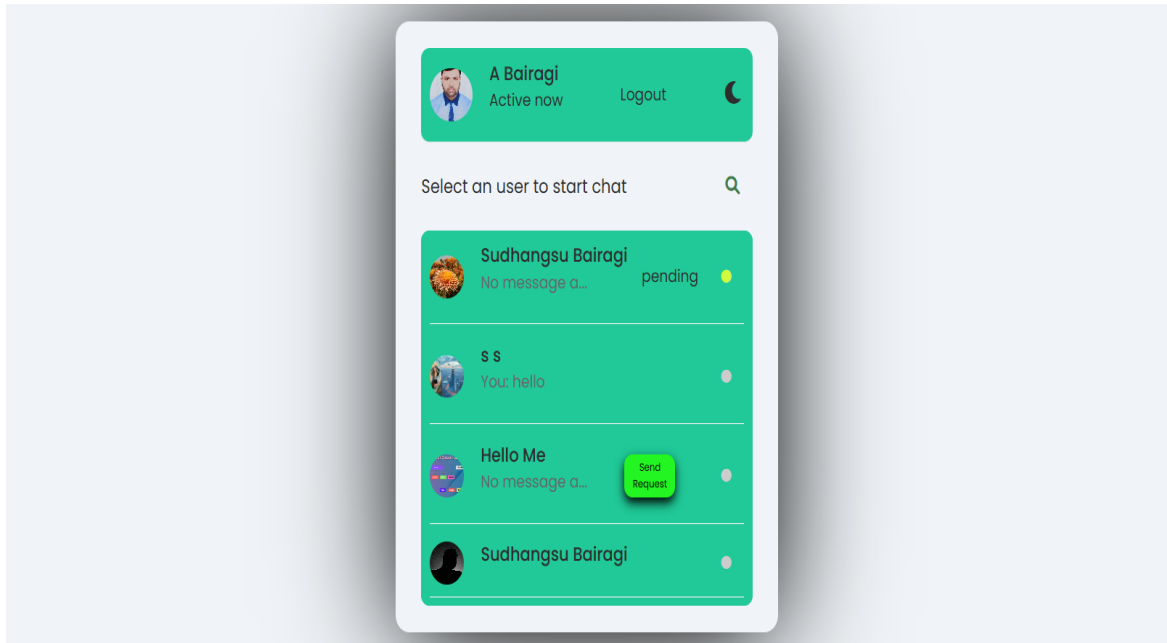
```

const pswrdField = document.querySelector(".form input[type='password']"),
toggleIcon = document.querySelector(".form .field i");

toggleIcon.onclick = () =>{
    if(pswrdField.type === "password"){
        pswrdField.type = "text";
        toggleIcon.classList.add("active");
    }else{
        pswrdField.type = "password";
        toggleIcon.classList.remove("active");
    }
}
}

```

Contract List:



users.php:

```
<?php
    session_start();
    include_once "php/config.php";
    if(!isset($_SESSION['unique_id'])){
        header("location: login.php");
    }
?>
<?php include_once "header.php"; ?>
<body>
    <div class="wrapper">
        <section class="users">
            <header>
                <div class="content">
                    <?php
                        $sql = mysqli_query($conn, "SELECT * FROM users WHERE unique_id =
{$_SESSION['unique_id']}");
                        if(mysqli_num_rows($sql) > 0){
                            $row = mysqli_fetch_assoc($sql);
                        }
                    ?>
```

```

         >
        <div class="details">
            <span><?php echo $row['fname']. " " . $row['lname'] ?></span>
            <p><?php echo $row['status']; ?></p>
        </div>
    </div>

    <a class="hover-underline-animation" href="php/logout.php?logout_id=<?php echo
$row['unique_id']; ?>" class="logout">Logout</a>
    <label for="theme-toggle" id="theme-toggle-icon">
        <i class="fa-solid fa-moon"></i>
    </label>
    <input type="checkbox" class="theme-toggle" id="theme-toggle">
</header>
<!-- <i id="themeMode" class="fa-solid fa-sun"></i> -->
<div class="search">
    <span class="text">Select an user to start chat</span>
    <input type="text" placeholder="Enter name to search...">
    <button><i id="themeMode" class="fas fa-search"></i></button>
</div>
<div class="users-list">

</div>
</section>
</div>

<script src="javascript/users.js"></script>
<script src="javascript/themeMode.js"></script>
</body>
</html>

```

php/users.php:

```

<?php

    session_start();
    include_once "config.php";
    $outgoing_id = $_SESSION['unique_id'];
    $sql = "SELECT * FROM users WHERE NOT unique_id = {$outgoing_id} ORDER BY
user_id DESC";
    $query = mysqli_query($conn, $sql);
    $output = "";

```

```

if(mysqli_num_rows($query) == 0){
    $output .= "No users are available to chat";
}elseif(mysqli_num_rows($query) > 0){
    include_once "data.php";
}
echo $output;

```

?>

javascript/users.js:

```

const searchBar = document.querySelector(".search input"),
    searchIcon = document.querySelector(".search button"),
    userList = document.querySelector(".users-list");

searchIcon.onclick = () => {
    searchBar.classList.toggle("show");
    searchIcon.classList.toggle("active");
    searchBar.focus();
    if (searchBar.classList.contains("active")) {
        searchBar.value = "";
        searchBar.classList.remove("active");
    }
};

searchBar.onkeyup = () => {
    let searchTerm = searchBar.value;
    if (searchTerm !== "") {
        searchBar.classList.add("active");
    } else {
        searchBar.classList.remove("active");
    }

    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/search.php", true);
    xhr.onload = () => {
        if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
            userList.innerHTML = xhr.response;
        }
    };
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.send("searchTerm=" + searchTerm);
};

```

```

setInterval(() => {
    let xhr = new XMLHttpRequest();
    xhr.open("GET", "php/users.php", true);
    xhr.onload = () => {
        if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
            if (!searchBar.classList.contains("active")) {
                usersList.innerHTML = xhr.response;
            }
        }
    };
    xhr.send();
}, 500);

// Function to send message request
function sendMessageRequest(receiverId) {
    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/send_request.php", true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.onload = () => {
        if (xhr.status === 200) {
            alert(xhr.responseText); // Notify user of request status
        }
    };
    xhr.send("receiver_id=" + receiverId);
}

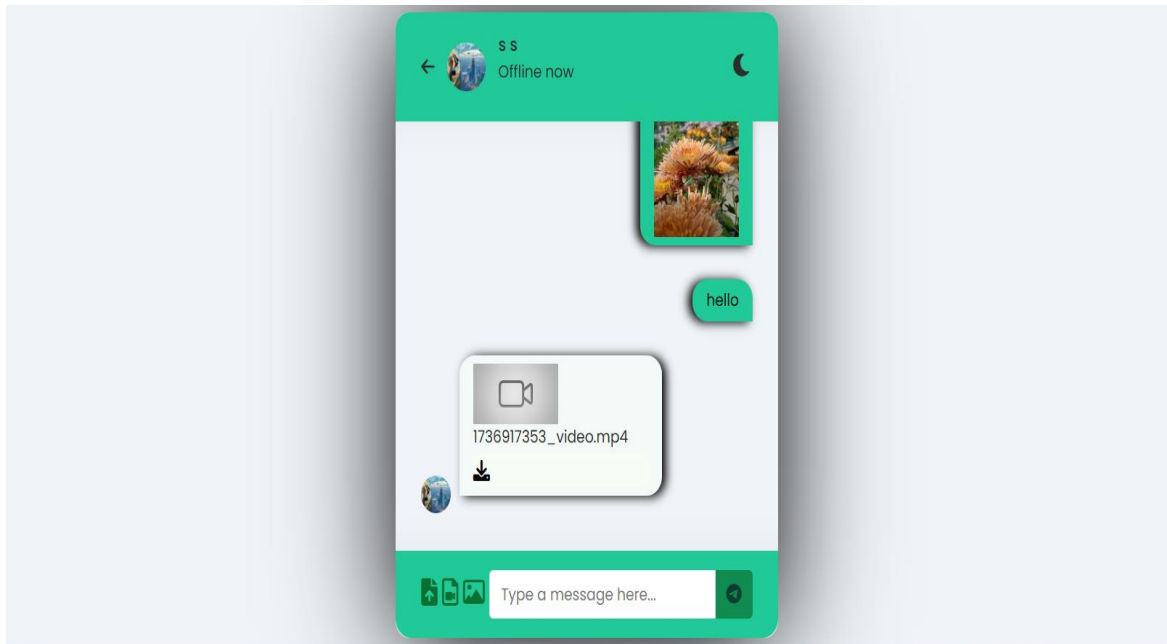
// Function to send a message request
function sendMessageRequest(receiverId) {
    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/send_request.php", true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.onload = () => {
        if (xhr.status === 200) {
            // alert(xhr.responseText); // Notify user of request status
        }
    };
    xhr.send("receiver_id=" + receiverId);
}

function respondRequest(senderId, action) {
    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/respond_request.php", true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

```

```
xhr.onload = () => {  
    if (xhr.status === 200) {  
        alert(xhr.responseText); // Notify user of acceptance or rejection  
        // location.reload(); // Reload users list  
    }  
};  
xhr.send("sender_id=" + senderId + "&action=" + action);  
}  
  
// Function to open chat if request is accepted  
function openChat(receiverId) {  
    window.location.href = "chat.php?user_id=" + receiverId;  
}
```

Chat :



chat.php:

```
<?php
session_start();
include_once "php/config.php";
if(!isset($_SESSION['unique_id'])){
    header("location: login.php");
}
?>
<?php include_once "header.php"; ?>
<body>
    <div class="wrapper">
        <section class="chat-area">
            <header>
                <?php
                $user_id = mysqli_real_escape_string($conn, $_GET['user_id']);
                $sql = mysqli_query($conn, "SELECT * FROM users WHERE unique_id =
                {$user_id}");
                if(mysqli_num_rows($sql) > 0){
                    $row = mysqli_fetch_assoc($sql);
                }else{
                    header("location: users.php");
                }
                ?>
```

```

<div id="user-status">
  <a href="users.php" class="back-icon"><i class="fas fa-arrow-left"></i></a>
  
  <div class="details">
    <span><?php echo $row['fname']. " " . $row['lname'] ?></span>
    <p><?php echo $row['status']; ?></p>
  </div>
</div>

<div id="moon-sun">
<label for="theme-toggle" id="theme-toggle-icon">
  <i class="fa-solid fa-moon"></i>
</label>
<input type="checkbox" class="theme-toggle" id="theme-toggle">
</div>
</header>

<div class="chat-box">

</div>

<form action="#" class="typing-area">
  <input type="text" class="incoming_id" name="incoming_id" value="<?php echo
$user_id; ?>" hidden>

  <input type="file" name="document" id="fileInput" class="fileInput"
accept=".doc,.docx,.pdf,.txt,.xlsx,.xls,.ppt,.pptx">
  <label for="fileInput" class="input-file uploadButton" ><i class="fa-solid fa-file-arrow-
up"></i></label>

  <input type="file" name="video" id="videoInput" class="fileInput" accept="video/*">
  <label for="videoInput" class="input-video uploadButton"><i class="fa-regular fa-file-
video"></i></label>

  <input type="file" name="image" id="imageInput" class="fileInput" accept="image/*">
  <label for="imageInput" class="input-image uploadButton"><i class="fa-regular fa-
image"></i></label>

  <input type="text" name="message" class="input-field" placeholder="Type a message
here..." autocomplete="off">
  <button><i class="fab fa-telegram-plane"></i></button>
</form>
</section>

```



```

</div>
<script src="javascript/themeMode.js"></script>
<script src="javascript/chat.js"></script>

</body>
</html>

```

javascript/chat.php:

```

const form = document.querySelector(".typing-area"),
incoming_id = form.querySelector(".incoming_id").value,
inputField = form.querySelector(".input-field"),
imageInputField = form.querySelector("#imageInput"),
videoInputField = form.querySelector("#videoInput"),
fileInputField = form.querySelector("#fileInput"),
sendBtn = form.querySelector("button"),
chatBox = document.querySelector(".chat-box");

```

```

form.onsubmit = (e) => {
  e.preventDefault();
};

```

```

inputField.focus();
inputField.onkeyup = () => {
  checkFields();
};

```

```

// Add listeners for both file inputs
imageInputField.onChange = () => {
  checkFields();
};
fileInputField.onChange = () => {
  checkFields();
};

```

```

videoInputField.onChange = () => {
  checkFields();
};

```

```

// Function to check all fields and activate the button
function checkFields() {

```

```

    if (inputField.value !== "" || imageInputField.files.length > 0 || videoInputField.files.length
    > 0 || fileInputField.files.length > 0) {
        sendBtn.classList.add("active");
    } else {
        sendBtn.classList.remove("active");
    }
}

sendBtn.onclick = () => {
    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/insert-chat.php", true);
    xhr.onprogress = () => {
        console.log("OnProgress....");
        loader.style.display = 'block';
    };
    xhr.onload = () => {
        loader.style.display = 'none';
        if (xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                fileInputField.value = "";
                inputField.value = "";
                imageInputField.value = "";
                videoInputField.value = "";

                checkFields();
                let data = xhr.responseText;
                if (data !== "Success") {
                    alert(data);
                }
            }
        }
    };
    let formData = new FormData(form);
    xhr.send(formData);
};

chatBox.onmouseenter = () => {
    chatBox.classList.add("active");
};

chatBox.onmouseleave = () => {
    chatBox.classList.remove("active");
};

```

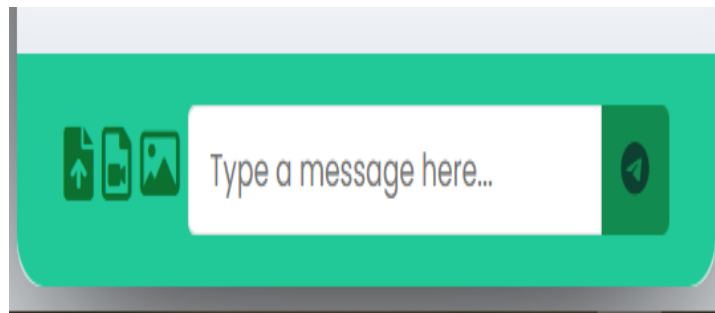
```

setInterval(() => {
    let xhr = new XMLHttpRequest();
    xhr.open("POST", "php/get-chat.php", true);
    xhr.onload = () => {
        if (xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                let data = xhr.response;
                chatBox.innerHTML = data;
                if (!chatBox.classList.contains("active")) {
                    scrollToBottom();
                }
            }
        }
    };
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.send("incoming_id=" + incoming_id);
}, 500);

function scrollToBottom() {
    chatBox.scrollTop = chatBox.scrollHeight;
}

```

Message:



php/get-chat.php:

<?php

```
session_start();
if(isset($_SESSION['unique_id'])){
    include_once "config.php";
    $outgoing_id = $_SESSION['unique_id'];
    $incoming_id = mysqli_real_escape_string($conn, $_POST['incoming_id']);
    $output = "";

    // Execute the first query

    $sql = "SELECT * FROM messages LEFT JOIN users ON users.unique_id =
messages.outgoing_msg_id
        WHERE (outgoing_msg_id = {$outgoing_id} AND incoming_msg_id =
{$incoming_id})
        OR (outgoing_msg_id = {$incoming_id} AND incoming_msg_id =
{$outgoing_id}) ORDER BY msg_id";
    $query = mysqli_query($conn, $sql);
    $check_There_Is_message_or_not = FALSE;

    // Check if there are results from either query

    if(mysqli_num_rows($query) > 0 ){

        $check_There_Is_message_or_not = TRUE;
        while($row = mysqli_fetch_assoc($query)) {
            if($row['outgoing_msg_id'] === $outgoing_id) {
```

```

        if(!empty(htmlspecialchars($row['msg']))) {
            $output .= '<div class="chat outgoing">
                <div class="details">
                    <p>'. $row['msg'] .'</p>

                </div>
            </div>';
        }
        if(!empty(htmlspecialchars($row['SendImg']))) {
            $output .= '<div class="chat outgoing">
                <div class="details">
                    <p class = "chatimage"></p>
                </div>
            </div>';

        }
        if(!empty(htmlspecialchars($row['SendVideo']))) {
            $output .= '<div class="chat outgoing">
                <div class="details">
                    <p class = "chatimage">'. $row['SendVideo'] .'</p>
                </div>
            </div>';

        }
        if(!empty(htmlspecialchars($row['Senddocument']))) {
            $output .= '<div class="chat outgoing">
                <div class="details">
                    <p class = "chatimage"><i class="fa-regular fa-file-
alt"></i>'. $row['Senddocument'] .'</p>
                </div>
            </div>';

        }
    } else {

        if(!empty(htmlspecialchars($row['msg']))) {
            $output .= '<div class="chat incoming">
                
                <div class="details">
                    <p>'. $row['msg'] .'</p>
                </div>
            </div>';

        }
    }

```

```

        if(!empty(htmlspecialchars($row['SendImg']))) {
            $output .= '<div class="chat incoming">
                
                <div class="details">
                    <p class = "chatimage"> <a href="php/Sendimages/'.
htmlspecialchars($row['SendImg']) .'" download="image.jpg" class="download-btn"><i
class="fa-solid fa-download"></i></a></p>

                    </div>
                </div>';

        }

        if(!empty(htmlspecialchars($row['SendVideo']))) {
            $output .= '<div class="chat incoming">
                
                <div class="details">
                    <p class = "chatimage">'. $row['SendVideo']. '
                    <a href="php/Sendimages/'. htmlspecialchars($row['SendVideo'])
.'" download="video.mp4" class="download-btn"><i class="fa-solid fa-
download"></i></a></p>

                    </div>
                </div>';

        }

        if (!empty(htmlspecialchars($row['Senddocument']))) {
            $output .= '<div class="chat incoming">
                
                <div class="details">
                    <p class="chatdocument">

                        <i class="fa-regular fa-file-alt"></i> ' .
htmlspecialchars($row['Senddocument']) . '

                        <!-- Download button for the document -->
                        <a href="php/Sendimages/' .
htmlspecialchars($row['Senddocument']) . '" download="" .
htmlspecialchars($row['Senddocument']) . '" class="download-btn">
                            <i class="fa-solid fa-download"></i>
                        </a>
                    </p>
                </div>
            </div>';
        }
    }

```

```

        }
    }
}

if($check_There_Is_message_or_not == FALSE){
    $output .= '<div class="text">No messages are available. Once you send a message
they will appear here.</div>';
}

echo $output;
} else {
    header("location: ../login.php");
}

?>

```

php/insert-chat.php

```

<?php
session_start();
if (isset($_SESSION['unique_id'])) {
    include_once "config.php";
    $outgoing_id = $_SESSION['unique_id'];
    $incoming_id = mysqli_real_escape_string($conn, $_POST['incoming_id']);
    $message = mysqli_real_escape_string($conn, $_POST['message']);
    // Check if a message is provided
    if (!empty($message)) {
        $sql = mysqli_query($conn, "INSERT INTO messages (incoming_msg_id,
outgoing_msg_id, msg)
VALUES ({ $incoming_id }, { $outgoing_id }, '{ $message }')") or
die(mysqli_error($conn));

        if ($sql) {
            echo "Success";
        } else {
            echo "Something went wrong. Please try again!";
        }
    }
}

```

```

// Handle image upload
elseif (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK)
{
    handleFileUpload($conn, $incoming_id, $outgoing_id, $_FILES['image'],
"SendImg");
}

// Handle video upload
elseif (isset($_FILES['video']) && $_FILES['video']['error'] === UPLOAD_ERR_OK)
{
    handleFileUpload($conn, $incoming_id, $outgoing_id, $_FILES['video'],
"SendVideo");
}
elseif (isset($_FILES['document']) && $_FILES['document']['error'] ===
UPLOAD_ERR_OK) {
    handleFileUpload($conn, $incoming_id, $outgoing_id, $_FILES['document'],
"Senddocument");
}

else {
    echo "No file uploaded or error occurred.";
}
} else {
    header("location: ../login.php");
}

// Function to handle file uploads for both images and videos
function handleFileUpload($conn, $incoming_id, $outgoing_id, $file, $db_column) {
    $file_name = $file['name'];
    $file_type = $file['type'];
    $tmp_name = $file['tmp_name'];

    // Get the file extension
    $file_explode = explode('.', $file_name);
    $file_ext = strtolower(end($file_explode)); // Make extension lowercase

    // Allowed extensions for images and videos
    $image_extensions = ["jpeg", "png", "jpg"];
    $video_extensions = ["mp4", "avi", "mov"];
    $document_extensions = ["doc", "docx", "pdf", "txt", "xls", "xlsx", "ppt", "pptx"];
    $all_extensions = array_merge($image_extensions, $video_extensions,
$document_extensions); // Combine both arrays

```



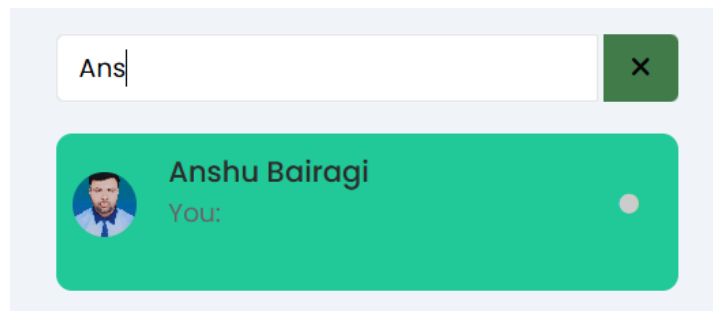
```

if (in_array($file_ext, $all_extensions)) {
    $image_types = ["image/jpeg", "image/jpg", "image/png"];
    $video_types = ["video/mp4", "video/x-msvideo", "video/quicktime"];
    $document_types = [
        "application/msword",           // .doc
        "application/vnd.openxmlformats-officedocument.wordprocessingml.document", //
.docx
        "application/pdf",              // .pdf
        "text/plain",                   // .txt
        "application/vnd.ms-excel",      // .xls
        "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", // .xlsx
        "application/vnd.ms-powerpoint", // .ppt
        "application/vnd.openxmlformats-officedocument.presentationml.presentation" //
.pptx
    ];
    $all_types = array_merge($image_types, $video_types, $document_types); //
Combine both MIME types

    if (in_array($file_type, $all_types)) {
        $time = time();
        $new_file_name = $time . "_" . $file_name; // Use underscore to avoid issues
        // Move the uploaded file to the appropriate directory
        if (move_uploaded_file($tmp_name, "Sendimages/" . $new_file_name)) {
            // Insert the file name into the correct database column (image or video)
            $sql = mysqli_query($conn, "INSERT INTO messages (incoming_msg_id,
outgoing_msg_id, {$db_column})
                                VALUES ({ $incoming_id}, { $outgoing_id},
'{$new_file_name}')") or die(mysqli_error($conn));
            if ($sql) {
                echo "Success";
            } else {
                echo "Something went wrong. Please try again!";
            }
        } else {
            echo "Failed to move uploaded file.";
        }
    } else {
        echo "File type not allowed.";
    }
} else {
    echo "Invalid file extension.";
}
}
?>

```

Search Bar:



php/search.php:

```
<?php
    session_start();
    include_once "config.php";

    $outgoing_id = $_SESSION['unique_id'];
    $searchTerm = mysqli_real_escape_string($conn, $_POST['searchTerm']);

    $sql = "SELECT * FROM users WHERE NOT unique_id = {$outgoing_id} AND (fname
    LIKE '% {$searchTerm}%' OR lname LIKE '% {$searchTerm}%') ";
    $output = "";
    $query = mysqli_query($conn, $sql);
    if(mysqli_num_rows($query) > 0){
        include_once "data.php";
    }else{
        $output .= 'No user found related to your search term';
    }
    echo $output;
?>
```

php/data.php:

<?php

// Check if users are available

```
while ($row = mysqli_fetch_assoc($query)) {

    $sql2 = "SELECT * FROM messages WHERE (incoming_msg_id =
    {$row['unique_id']}
        OR outgoing_msg_id = {$row['unique_id']}) AND (outgoing_msg_id =
    {$soutgoing_id}
        OR incoming_msg_id = {$soutgoing_id}) ORDER BY msg_id DESC LIMIT
    1";

    $query2 = mysqli_query($conn, $sql2);
    $row2 = mysqli_fetch_assoc($query2);
    (mysqli_num_rows($query2) > 0) ? $result = $row2['msg'] : $result = "No message
    available";
    (strlen($result) > 12) ? $msg = substr($result, 0, 12) . '...' : $msg = $result;

    if(isset($row2['outgoing_msg_id'])){
        ($soutgoing_id == $row2['outgoing_msg_id']) ? $you = "You: " : $you = "";
    }else{
        $you = "";
    }
    ($row['status'] == "Offline now") ? $offline = "offline" : $offline = "";
    ($soutgoing_id == $row['unique_id']) ? $hid_me = "hide" : $hid_me = "";

    $user_id = $row['unique_id'];

    // Check if there is a pending message request involving the logged-in user
    $checkRequest = mysqli_query($conn, "SELECT sender_id, receiver_id, status
        FROM message_requests
        WHERE (sender_id = {$soutgoing_id} AND receiver_id =
    {$user_id})
        OR (sender_id = {$user_id} AND receiver_id =
    {$soutgoing_id})");
    $request = mysqli_fetch_assoc($checkRequest);

    // Determine what button to display based on request status
    if ($request) {
        if ($request['status'] == 'accepted') {
            // $chat_button = '<button onclick="openChat(' . $user_id . ')">Chat</button>';
            $output .= '<a class="hover-underline-animation" href="chat.php?user_id=' .
    $row['unique_id'] . '">
```

```

        <div class="content">
        
        <div class="details">
            <span>'. $row['fname']. " " . $row['lname'] .'</span>
            <p>'. $you . $msg .'</p>
        </div>
        </div>
        <div class="status-dot ' . $offline .'"><i class="fas fa-circle"></i></div>
    </a>;
    } elseif ($request['status'] == 'pending' && $request['receiver_id'] == $outgoing_id)
    {
        $chat_button = '<button class="SendButton-Request" onclick="respondRequest(' .
        $request['sender_id'] . ', \'accept\')>Accept</button>
            <button class="SendButton-Request" onclick="respondRequest(' .
        $request['sender_id'] . ', \'reject\')>Reject</button>;
        $output .= '<a class="hover-underline-animation">
            <div class="content">
            
            <div class="details">
                <span>'. $row['fname']. " " . $row['lname'] .'</span>
                <p>'. $you . $msg .'</p>
            </div>
            </div>
            '. $chat_button . '
            <div class="status-dot ' . $offline .'"><i class="fas fa-circle"></i></div>
        </a>;
    } elseif ($request['status'] == 'rejected' && $request['sender_id'] == $outgoing_id) {
        $chat_button = '<button class="SendButton-Request"
        onclick="sendMessageRequest(' . $user_id . ')>Send Request</button>;
        $output .= '<a class="hover-underline-animation">
            <div class="content">
            
            <div class="details">
                <span>'. $row['fname']. " " . $row['lname'] .'</span>
                <p>'. $you . $msg .'</p>
            </div>
            </div>
            '. $chat_button . '
            <div class="status-dot ' . $offline .'"><i class="fas fa-circle"></i></div>
        </a>;
    } else {
        $chat_button = '<span>'. $request['status'] .'</span>;
        $output .= '<a class="hover-underline-animation">
            <div class="content">

```

```

        
```

php/respond_request.php:

```
<?php
session_start();
include_once "config.php";

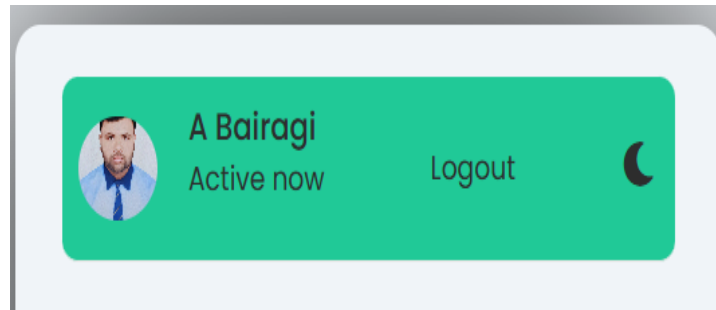
$receiver_id = $_SESSION['unique_id'];
$sender_id = $_POST['sender_id'];
$action = $_POST['action']; // 'accept' or 'reject'

if ($action == 'accept') {
    $status = 'accepted';
} else {
    $status = 'rejected';
}

$sql = "UPDATE message_requests
      SET status = '$status'
      WHERE sender_id = $sender_id AND receiver_id = $receiver_id";

if (mysqli_query($conn, $sql)) {
    echo "Request $status successfully!";
} else {
    echo "Failed to update request.";
}
?>
```

Logout:



php/ logout.php:

```
<?php
session_start();
if(isset($_SESSION['unique_id'])){
    include_once "config.php";
    $logout_id = mysqli_real_escape_string($conn, $_GET['logout_id']);
    if(isset($logout_id)){
        $status = "Offline now";
        $sql = mysqli_query($conn, "UPDATE users SET status = '{$status}' WHERE
unique_id={$_GET['logout_id']}");
        if($sql){
            session_unset();
            session_destroy();
            header("location: ../index.php#feedbackForm");
        }
    }else{
        header("location: ../users.php");
    }
}else{
    header("location: ../index.php");
}
?>
```


Database Connection:

php/config.php:

```
<?php
$hostname = "localhost";
$username = "root";
$password = "";
$dbname = "chatapp";

$conn = mysqli_connect($hostname, $username, $password, $dbname);
if(!$conn){
    echo "Database connection error".mysqli_connect_error();
}

?>
```

Header:

header.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Chat App</title>
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
    rel="stylesheet"
  />

  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<div id="loader"><div id="load"></div></div>
<script src="header.js"></script>
<script src="javascript\animationCss.js"></script>
```

Header.js:

```
var loader = document.querySelector("#loader");
window.addEventListener("load",function(){

  loader.style.display = 'none';

});
```

CSS:

style.css:

```
@import  
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700&di  
splay=swap');
```

```
#loader {  
  width: 100%;  
  height: 100%;  
  background-color: transparent;  
  z-index: 2;  
  position: fixed;  
  top: 0;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

```
#load {  
  width: 50px;  
  height: 50px;  
  background-color: transparent;  
  border-radius: 50%;  
  border: 5px solid transparent;  
  border-top: 5px solid rgb(26, 103, 7);  
  border-bottom: 5px solid rgb(34, 0, 255);  
  animation: myani 0.6s infinite;  
  
}
```

```
@keyframes myani {  
  from {  
    transform: rotate(0deg);  
  }  
  
  to {  
    transform: rotate(360deg);  
  }  
}
```

```

:root {
  --char-body: #20C997;
  --hover-color: #29ffbf;
  --status-dot: rgb(209, 250, 62);
  --background-header: #20C997;
  --bg-style: #23dba4;
  --bg-style1: #67f7cc;
  --bg-style2: #b1fde6;
  --bg-style3: #d7fdf1;
  --bg-style4: #e9f8f4;
  --bg-style5: #F0F4F8;
  --background-color: #F0F4F8;
  --text-color: #2D2D2D;
  --button-background: #06520fbe;
  --button-text-hover: #23fe01;
  --button-text: #060606;
  --incoming-bg: #f7fff583;
  --outgoing-bg: #20C997;
  --border-color: #E5E5E5;
  --shadow-color: rgb(0, 0, 0);
  --box-shadow-color: rgba(0, 0, 0, 0.5);
  --input-border-color: #CCC;
  --error-bg: #F8D7DA;
  --error-border: #F5C6CB;
  --ripple-color: gold;
}

```

/* Dark Mode */

```

body.dark-mode {
  --char-body: #0a4c38;
  --hover-color: #20C997;
  --status-dot: rgb(14, 174, 14);
  --background-header: #0a4c38;
  --bg-style: #093f30;
  --bg-style1: #083429;
  --bg-style2: #072b23;
  --bg-style3: #041612;
  --bg-style4: #030f0c;
  --bg-style5: #030f0c;
  --background-color: #1C1F24;
  --text-color: #ffffff;
  --button-background: #2aad4d;
  --button-text-hover: #93ff82;
  --button-text: #ffffff;
}

```

```

--incoming-bg: #3f3f3f;
--outgoing-bg: #0a4c38;
--border-color: #333;
--shadow-color: rgb(0, 0, 0);
--box-shadow-color: rgba(255, 255, 255, 0.3);
--input-border-color: #555;
--error-bg: #661d1d;
--error-border: #aa4444;
--ripple-color: rgb(248, 119, 13);
}

```

```

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  text-decoration: none;
  font-family: 'Poppins', sans-serif;
  scroll-behavior: smooth;
}

```

```

body {
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background: var(--background-color);
  padding: 0 10px;
  color: var(--text-color);
}

```

```

/* SendRequest Button */
.users-list .SendButton-Request {
  border: none;
  background-color: rgb(34, 245, 34);
  color: #060606;
  font-size: 10px;
  border-radius: 10px;
  width: 60px;
  height: 40px;
  box-shadow: 0px 5px 10px black;
  transition: 0.1s linear;
}

```

```

.users-list .SendButton-Request:hover {
  background-color: yellow;
}

.users-list .SendButton-Request:active {
  transform: translateY(10px);
}

/* button underline animation */
.hover-underline-animation {
  color: var(--text-color);
  position: relative;
  text-decoration: none;
}

.hover-underline-animation:hover {
  color: var(--button-text-hover);
}

.hover-underline-animation::after {
  content: "";
  position: absolute;
  width: 100%;
  height: 2px;
  bottom: 0;
  left: 0;
  background-color: var(--button-text-hover);
  transform-origin: bottom right;
  transition: transform 0.25s ease-out;
}

.hover-underline-animation::after {
  transform: scaleX(0);
}

.hover-underline-animation:hover::after {
  transform: scaleX(1);
  transform-origin: bottom left;
}

```

```

/* theme change code */
#theme-toggle {
  display: none;
}

#theme-toggle-icon i {
  cursor: pointer;
  font-size: 24px;
  color: var(--text-color);
}

body.dark-mode #theme-toggle-icon i.fa-moon {
  display: none;
}

body.dark-mode #theme-toggle-icon i.fa-sun {
  display: inline-block;
}

#theme-toggle-icon i.fa-sun {
  display: none;
}

/* navbar code */
.container .navbarDesign {
  background-color: var(--background-header);
  padding: 10px;
}

.container nav .containerdiv .brandName {
  color: var(--text-color);
  cursor: default;
}

/* hero (body) section */
.container .hero {
  width: 100%;
  background-color: var(--background-header);
  padding: 10px;
}

```

```
.container .hero .heroClass {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  border: none;
  color: var(--text-color);
  background: transparent;
  /* border: 10px solid rebeccapurple; */
  background-color: var(--background-color);
  padding: 35px;
  border-radius: 30% 70% 70% 30% / 30% 30% 50% 50%;
  animation: mybuttonAni 10s linear 2s infinite alternate;
}
```

```
.container .hero .heroClass #heroHiding {
  text-align: center;
  color: var(--button-background);
  font-weight: bolder;
  position: relative;
  overflow: hidden;
}
```

```
.container .hero .heroClass #heroHiding::after {
  content: "";
  width: 100%;
  height: 2px;
  position: absolute;
  background-color: var(--button-background);
  bottom: 0;
  left: 0;
  /* transform-origin: bottom right; */
  animation: textAni 5s linear infinite alternate-reverse;
}
```

```
@keyframes textAni {
  0% {
    transform: scaleX(0);
    transform-origin: left bottom;
  }
  100% {
    transform: scaleX(1);
    transform-origin: bottom left;
  }
}
```



```

}

.container .hero .heroClass #heroHiding span {
  font-size: 16px;
}

.container .hero .heroClass P {
  text-align: center;
}

@keyframes mybuttonAni {
  0% {
    border-radius: 30% 70% 70% 30% / 30% 30% 50% 50%;
  }

  25% {
    border-radius: 58% 42% 70% 25% / 70% 46% 54% 24%;
  }

  50% {
    border-radius: 50% 50% 33% 67% / 55% 27% 63% 45%;
  }

  75% {
    border-radius: 33% 67% 58% 42% / 63% 68% 32% 37%;
  }
}

.container .hero .heroClass a.Mybtn {
  background-color: #20C997;
  padding: 10px;
  border-radius: 10px;
  box-shadow: 5px 5px 10px var(--shadow-color);
  font-size: 26px;
  margin: 10px;
  position: relative;
  overflow: hidden;
  z-index: 1;
  text-decoration: none;
  color: black;
  transition: transform 0.1s ease-in-out;
}

```

```

.container .hero .heroClass a.Mybtn:active {
  transform: translateY(6px);
  color: rgb(13, 74, 18);
}

.container .hero .heroClass a.Mybtn::before {
  content: "";
  width: 0px;
  height: 0px;
  background-color: rgb(172, 255, 141);
  top: var(--posY);
  left: var(--posX);
  position: absolute;
  border-radius: 50%;
  transform: translate(-50%, -50%);
  transition: width 2s, height 2s;
  z-index: -1;
}

.container .hero .heroClass a.Mybtn: hover::before {
  width: 200%;
  height: 200%;
  z-index: -1;
}

/* features section */

.container .features {
  width: 100%;
  text-align: center;
  margin-top: 10px;
}

.container .features .featuresBox {
  width: auto;
  height: auto;
  display: flex;
  flex-wrap: wrap;
}

```

```

.container .features .featuresBox .featuresCard {
  min-height: 200px;
  margin: 15px;
  background-color: var(--background-header);
  color: var(--text-color);
  /* transition: transform 0.3s; */
  position: relative;
  overflow: hidden;
  transition: transform 0.3s ease-in-out;
}

.container .features .featuresBox .featuresCard .card-index {
  z-index: 1;
}

.container .features .featuresBox .card-all {
  min-width: 200px;
  flex: 1;
}

.container .features .featuresBox .card-4 {
  min-width: 220px;
  flex: 3;
}

.container .features .featuresBox .featuresCard .card-body i {
  font-size: 50px;
  color: var(--button-background);
}

.container .features .featuresBox .featuresCard::before,
.container .feedbackForm::before {
  content: "";
  width: 0;
  height: 0;
  background-color: var(--ripple-color);
  color: red;
  top: var(--ripple-y);
  left: var(--ripple-x);
  position: absolute;
  border-radius: 50%;
  transform: translate(-50%, -50%);
  transition: width 5s ease, height 4s ease;
}

```

```

.container .features .featuresBox .featuresCard.scroll-ripple::before,

.container .feedbackForm.scroll-ripple::before {
  width: 150%;
  height: 150%;
  z-index: -1;
}

.container .features .featuresBox .featuresCard.scroll-ripple,
.container .feedbackForm.scroll-ripple {
  transform: scale(1.05);
}

/* feedback */
.container .feedbackForm {
  display: flex;
  flex-direction: column;
  align-items: center;
  background-color: var(--background-header);
  margin: 30px;
  border-radius: 6px;
  position: relative;
  overflow: hidden;
  transition: transform 0.3s ease-in-out;
}

.container .feedbackForm .feedbackHeader {
  width: 100%;
  text-align: center;
}

.container .feedbackForm #feedbackForm {
  max-width: 450px;
}

.form form .input textarea {
  height: 40px;
  width: 100%;
  font-size: 16px;
  padding: 0 10px;
  border-radius: 5px;
  color: var(--text-color);
  border: 2px solid var(--background-header);
  background-color: transparent;
}

```

```

.form form .input textarea:hover {
  border: 2px solid var(--hover-color);
}

.container .feedbackForm .feedbackFormAni {
  border: none;
  color: var(--text-color);
  background-color: var(--background-color);
  padding: 20px 100px 20px 100px;
  border-radius: 30% 70% 70% 30% / 30% 30% 50% 50%;
  animation: mybuttonAni 10s linear 0s infinite alternate;
}

/* footer */

.container footer .backToTop {
  height: 50px;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  background-color: var(--background-header);
  border-radius: 6px;
  margin: 5px;
}

.container footer .contactUs {
  text-align: center;
}

.container footer .contact {
  display: flex;
  align-items: center;
  justify-content: space-evenly;
  flex-direction: row;
  padding: 16px;
  border-radius: 6px;
  margin: 5px;
  color: var(--text-color);
  background-color: var(--background-header);
}

```

```

.container footer .contact .contactSection {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

.container footer .contact .contactSection a {
  text-decoration: none;
}

.container footer .contact .contactSection i {
  font-size: 50px;
  color: var(--button-background);
}

.container footer .footerCopyRights {
  background-color: var(--background-header);
  border-radius: 6px;
  margin: 5px;
  padding: 1px;
  text-align: center;
  max-height: 50px;
}

.container footer .footerCopyRights p {
  margin: 0px;
  color: var(--text-color);
}

.container footer .footerCopyRights p a {
  text-decoration: none;
}

@media screen and (max-width: 500px) {
  .container footer .contact {
    flex-direction: column;
  }

  .container .feedbackForm {
    margin: 15px;
  }
}

```

```
.container .feedbackForm .feedbackFormAni {  
  animation: none;  
  padding: 20px 55px 20px 55px;  
  border-radius: 10px;  
  margin: 1px;  
}  
}
```

```
.wrapper {  
  background: var(--background-color);  
  max-width: 450px;  
  width: 100%;  
  border-radius: 16px;  
  box-shadow: 0 0 128px 0 var(--shadow-color),  
    0 32px 64px -48px var(--box-shadow-color);  
}
```

/* Login & Signup Form CSS Start */

```
.form .headerSection {  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
  background-color: var(--background-header);  
  padding: 10px;  
  border-radius: 5px;  
}
```

```
.form {  
  padding: 25px 30px;  
}
```

```
.form header {  
  font-size: 25px;  
  font-weight: 600;  
  padding-bottom: 10px;  
}
```

```
.form form {  
  margin: 20px 0;  
}
```

```

.form form .error-text {
  color: var(--text-color);
  padding: 8px 10px;
  text-align: center;
  border-radius: 5px;
  background: var(--error-bg);
  border: 1px solid var(--error-border);
  margin-bottom: 10px;
  display: none;
}

.form form .name-details {
  display: flex;
}

.form .name-details .field:first-child {
  margin-right: 10px;
}

.form .name-details .field:last-child {
  margin-left: 10px;
}

.form form .field {
  display: flex;
  margin-bottom: 10px;
  flex-direction: column;
  position: relative;
}

.form form .field label {
  margin-bottom: 2px;
}

.form form .input input {
  height: 40px;
  width: 100%;
  font-size: 16px;
  padding: 0 10px;
  border-radius: 5px;
  color: var(--text-color);
  border: 2px solid var(--background-header);
  background-color: transparent;
}

```



```
.form form .input input:hover {  
  border: 2px solid var(--hover-color);  
}
```

```
.form form .field input {  
  outline: none;  
}
```

```
.form form .image label input {  
  display: none;  
}
```

```
.form form .image label {  
  margin-bottom: 2px;  
  border-radius: 5px;  
}
```

```
.form form .image label i {  
  display: flex;  
  align-items: center;  
  height: 40px;  
  margin-top: 17px;  
  color: var(--text-color);  
  font-size: 25px;  
  width: 100%;  
  padding: 0 15px;  
}
```

```
.form form .image label i:hover {  
  color: var(--button-background);  
}
```

```
.form form .image label i span {  
  margin-left: 25px;  
  font-size: 16px;  
  font-weight: 400;  
  color: var(--text-color);  
}
```

```
.form form .button input {  
  background: transparent;  
  border: none;  
  width: 100%;  
  height: 100%;  
}
```

```

.form form .button {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 45px;
  border: none;

  color: var(--button-text);
  font-size: 17px;
  background-color: var(--background-header);
  border-radius: 5px;
  cursor: pointer;
  margin-top: 22px;
  overflow: hidden;
  z-index: 1;
  transition: transform 0.1s ease-in-out;
}

.form form .button::before {
  content: "";
  width: 0px;
  height: 0px;
  background-color: rgb(172, 255, 141);
  top: var(--posY);
  left: var(--posX);
  position: absolute;
  border-radius: 50%;
  transform: translate(-50%, -50%);
  transition: width 2s, height 2s;
  z-index: -1;
}

.form form .button:hover::before {
  width: 200%;
  height: 200%;
  z-index: -1;
}

.form form .button:active {
  transform: translateY(6px);
  color: rgb(13, 74, 18);
}

```

```

.form form .field i {
  position: absolute;
  right: 15px;
  top: 70%;
  color: #ccc;
  cursor: pointer;
  transform: translateY(-50%);
}

.form form .field i.active::before {
  color: var(--button-background);
  content: "\f070";
}

.form .link {
  text-align: center;
  margin: 10px 0;
  font-size: 17px;
}

.form .link a {
  color: var(--button-background);
}

.form .link a:hover {
  color: var(--button-text-hover);
}

.users {
  padding: 25px 30px;
  min-height: 500px;
  max-height: 100vh;
  overflow: hidden;
}

.users header,
.users-list a {
  display: flex;
  align-items: center;
  padding-bottom: 20px;
  border-bottom: 1px solid var(--border-color);
  justify-content: space-between;
}

```

```
.users-list a button {
  width: 60px;
  height: 40px;
  background-color: #a2e9da;
  border: 5px solid black;
  color: red;
}

.users header {
  background-color: var(--background-header);
  padding: 10px;
  border-radius: 10px;
}

.wrapper img {
  object-fit: cover;
  border-radius: 50%;
}

.users header img {
  height: 50px;
  width: 50px;
}

:is(.users, .users-list) .content {
  display: flex;
  align-items: center;
}

:is(.users, .users-list) .content .details {
  color: var(--text-color);
  margin-left: 20px;
}

:is(.users, .users-list) .details span {
  font-size: 18px;
  font-weight: 500;
}

.users header .logout {
  display: block;
  background: transparent;
  color: var(--button-text);
  outline: none;
  border: none;
```

```

padding: 7px 15px;
text-decoration: none;
border-radius: 5px;
font-size: 17px;
}

.users .search {
margin: 20px 0;
display: flex;
position: relative;
align-items: center;
justify-content: space-between;
}

.users .search .text {
font-size: 18px;
}

.users .search input {
position: absolute;
height: 42px;
width: calc(100% - 50px);
font-size: 16px;
padding: 0 13px;
border: 1px solid var(--border-color);
outline: none;
border-radius: 5px 0 0 5px;
opacity: 0;
pointer-events: none;
transition: all 0.2s ease;
}

.users .search input.show {
opacity: 1;
pointer-events: auto;
}

.users .search button {
position: relative;
z-index: 1;
width: 47px;
height: 42px;
font-size: 17px;
cursor: pointer;
border: none;

```

```

background: var(--background-color);
color: var(--button-background);
outline: none;
border-radius: 0 5px 5px 0;
transition: all 0.2s ease;
}

.users .search button.active {
background: var(--button-background);
color: var(--button-text);
}

.search button.active i::before {
content: '\f00d';
}

.users-list {
background-color: var(--char-body);
max-height: 350px;
overflow-y: auto;
padding: 10px;
border-radius: 10px;
}

:is(.users-list, .chat-box)::--webkit-scrollbar {
width: 0px;
}

.users-list a {
padding-bottom: 10px;
margin-bottom: 15px;
padding-right: 15px;
border-bottom-color: #f1f1f1;
}

.users-list a:last-child {
margin-bottom: 0px;
border-bottom: none;
}

.users-list a img {
height: 40px;
width: 40px;
}

```

```

.users-list a .details p {
  color: #67676a;
}

.users-list a .status-dot {
  font-size: 12px;
  color: var(--status-dot);
  padding-left: 10px;
}

.users-list a .status-dot.offline {
  color: #ccc;
}

.chat-area header {
  background-color: var(--background-header);
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 18px 30px;
  border-radius: 16px 16px 0 0;
}

.chat-area header #user-status {
  display: flex;
  align-items: center;
}

.chat-area header #user-status .back-icon {
  color: var(--text-color);
  font-size: 18px;
}

.chat-area header img {
  height: 45px;
  width: 45px;
  margin: 0 15px;
}

.chat-area header #user-status .details span {
  font-size: 17px;
  font-weight: 500;
}

```

```
.chat-box {  
  position: relative;  
  min-height: 400px;  
  max-height: 400px;  
  
  overflow-y: auto;  
  padding: 10px 30px 20px 30px;  
  background: var(--background-color);  
  box-shadow: inset 0 32px 32px -32px rgb(0 0 0 / 5%),  
    inset 0 -32px 32px -32px rgb(0 0 0 / 5%);  
}
```

```
.chat-box .text {  
  position: absolute;  
  top: 45%;  
  left: 50%;  
  width: calc(100% - 50px);  
  text-align: center;  
  transform: translate(-50%, -50%);  
}
```

```
.chat-box .chat {  
  margin: 15px 0;  
}
```

```
.chat-box .chat p {  
  word-wrap: break-word;  
  padding: 8px 16px;  
  box-shadow: 0 0 32px rgb(0 0 0 / 8%),  
    0 16px 16px -16px rgb(0 0 0 / 10%);  
}
```

```
.chat-box .outgoing {  
  display: flex;  
}
```

```
.chat-box .outgoing .details {  
  margin-left: auto;  
  max-width: calc(100% - 130px);  
}
```

```
.outgoing .details p {  
  background: var(--outgoing-bg);  
  color: var(--button-text);  
  border-radius: 18px 18px 0 18px;
```



```

    box-shadow: -5px 1px 10px 0px rgb(0, 0, 0);
}

.chat-box .incoming {
    display: flex;
    align-items: flex-end;
}

.chat-box .incoming img {
    height: 35px;
    width: 35px;
}

.chat-box .incoming .details {
    margin-right: auto;
    margin-left: 10px;
    max-width: calc(100% - 130px);
}

.incoming .details p {
    background: var(--incoming-bg);
    color: var(--text-color);
    border-radius: 18px 18px 18px 0;
    box-shadow: 5px 1px 10px 0px rgb(0, 0, 0);
}

.chat-box .outgoing .details p.chatimage img,
.chat-box .incoming .details p.chatimage img {
    width: 100%;
    max-width: 100px;
    height: auto;
    margin: 0;
    padding: 0;
    border-radius: 1px;
    overflow: hidden;
    display: flex;
    align-items: center;
    justify-content: center;
}

```

```

/* Image style within the chat box*/
.chat-box .outgoing .details p.chatimage img,
.chat-box .incoming .details p.chatimage img {
  width: 100%;
  height: auto;
  border-radius: 1px;
  object-fit: cover;
  max-height: 200px;
}

/* chat box download button */
.chat-box .incoming .details p a.download-btn {
  display: inline-block;
  margin-top: 5px;
  background-color: var(--incoming-bg);
  color: var(--button-text);
  text-decoration: none;
  font-size: 20px;
  cursor: pointer;
  width: 100%;
}

.chat-box .incoming .details p a.download-btn:hover {
  color: #1daa15;
}

.wrapper .chat-area .typing-area {
  padding: 18px 30px;
  display: flex;
  justify-content: space-between;
}

.wrapper .chat-area .typing-area input {
  height: 45px;
  width: calc(100% - 58px);
  font-size: 16px;
  padding: 0 13px;
  border: 1px solid var(--border-color);
  outline: none;
  border-radius: 5px 0 0 5px;
}

.wrapper .chat-area .typing-area button {
  height: 45px;

```

```

color: var(--button-text);
width: 55px;
border: none;
outline: none;
background: var(--button-background);
font-size: 19px;
cursor: pointer;
opacity: 0.7;
pointer-events: none;
border-radius: 0 5px 5px 0;
transition: all 0.3s ease;

}

.wrapper .chat-area .typing-area button.active {
  opacity: 1;
  pointer-events: auto;
}

.wrapper .chat-area .typing-area .fileInput {
  display: none;
}

.wrapper .chat-area .typing-area {
  display: flex;
  align-items: center;
  background-color: var(--background-header);
  border-radius: 0px 0px 18px 18px;
}

.wrapper .chat-area .typing-area .uploadButton i {
  margin-right: 5px;
  cursor: pointer;
  color: var(--button-background);
  font-size: 26px;
}

.wrapper .chat-area .typing-area .uploadButton i:hover {
  color: var(--hover-color);
}

/* Responsive media query */
@media screen and (max-width: 450px) {

```

```

.form,
.users {
  padding: 20px;
}

.form header {
  text-align: center;
}

.form form .name-details {
  flex-direction: column;
}

.form .name-details .field:first-child {
  margin-right: 0px;
}

.form .name-details .field:last-child {
  margin-left: 0px;
}

.users header img {
  height: 45px;
  width: 45px;
}

.users header .logout {
  padding: 6px 10px;
  font-size: 16px;
}

:is(.users, .users-list) .content .details {
  margin-left: 15px;
}

.users-list a {
  padding-right: 10px;
}

.chat-area header {
  padding: 15px 20px;
}

```

```
.chat-box {  
  min-height: 400px;  
  padding: 10px 15px 15px 20px;  
}  
  
.chat-box .chat p {  
  font-size: 15px;  
}  
  
.chat-box .outgoing .details {  
  max-width: 230px;  
}  
  
.chat-box .incoming .details {  
  max-width: 265px;  
}  
  
.incoming .details img {  
  height: 30px;  
  width: 30px;  
}  
  
.chat-area form {  
  padding: 20px;  
}  
  
.chat-area form input {  
  height: 40px;  
  width: calc(100% - 48px);  
}  
  
.chat-area form button {  
  width: 45px;  
}  
}
```

9 Case Studies

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order . Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview:

The following represents the overall flow of the testing process:

1. Identify the requirement to be tested. Allstate cases shall be derived using the current Program Specification.
2. Identify which particular test will be used to test each model.
3. Identify the expected results for each test.
4. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report(STR)
5. Perform the test.
6. Successful unit testing is required before the unit is eligible for component integration/System testing.
7. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
8. Test documents and reports shall be submitted. Any specification to be reviewed, received or updated shall be handled immediately.

10.1 Testing Methodology

UNIT TESTING:

Testing is done at the source code level for language specific programming errors such as bad syntax, logic errors or to test particular functions or code modules. The unit test cases shall be designed test the validity of the programs correctness.

Unit testing has the greatest effect in the quality of our code when it's an integral part of our software development worth flow. In our project we used this testing by ensuring whether the output is correct with respect to the output.

ALPHA TESTING:

This is a form of internal acceptance testing performed mainly by the in-house software QA and testing teams. Alpha testing is the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for beta test.

Alpha testing can also be done by the potential users or customers of the application. But still, this is a form of in-house acceptance testing.

BETA TESTING:

This is a testing stage followed by the internal full alpha test cycle. This is the final testing phase where the companies release the software to few external user groups outside the company test teams or employees. This initial software version is known as the beta version. Most companies gather user feedback in this release.

In short, beta testing can be defined as– the testing carried out by real users in a real environment.

WHITE BOX TESTING:

In White box testing the UI is by passed, inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code.

10.2 Test Cases

The Following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The test cases that will be used for designing use cases in shown and onward.

Tested By:	SAYAN ADHIKARY
Test Type	Unit Testing
Test Case Number	1
Test Case Name	Registration Testing
Test Case Description	The Users are required to enter their name, password, userName and add a photo in order to login. There can't be 2 similar userName, it has to be unique.
Item(s) to be tested	
1	Verification of unique userName with the record in the database.
Specifications	
Input	Expected Output/Result
1) All fields are required and Unique user	1) Successful registration
2) duplicate username	2) use Message

Tested By:		SAYAN ADHIKARY	
Test Type		Integration Testing	
Test Case Number		2	
Test Case Name		User Interface	
Test Case Description		Display all the details properly or not.	
Item(s) to be tested			
1	Check whether the details are showed or not.		
Specifications			
Input		Expected Output/Result	
1) Check whether all the unit is coupled & perfect or not		1) All the units are coupled & perfect	

Tested By:		SAYAN ADHIKARY	
Test Type		Black Box Testing	
Test Case Number		3	
Test Case Name		Messaging and chat creation	
Test Case Description		Users can search for people send request and start sending and receiving messages	
Item(s) to be tested			
1	Verification of the user can properly chat with friends or not.		
Specifications			
Input		Expected Output/Result	
1) Friend request send or not		1) Request send successfully	
2) Chat creation successful		2) messaging feature working flawless	

Tested By:	SAYAN ADHIKARY		
Test Type	Unit Testing		
Test Case Number	4		
Test Case Name	login Testing		
Test Case Description	The user should enter username and password so that member can able to go for the further options.		
Item(s) to be tested			
1	Verification all of the codes are executed or not.		
Specifications			
Input		Expected Output/Result	
1) Correct username and password		1) Successful login	
2) Incorrect username or Password		2) Failure Message	

Tested By:		SAYAN ADHIKARY
Test Type		White box testing
Test Case Number		5
Test Case Name		Coding Test
Test Case Description		All the code is executed or not
Item(s) to be tested		
1	Verification all of the codes are executed or not.	
Specifications		
Input		Expected Output/Result
1) Verify all the codes are properly worked or not.		1) All the codes are verified and properly worked.

10.3 Validation

The validation of the database design ensures that the chosen schema meets the functional and non-functional requirements of the chat application. By implementing proper validation techniques and principles, the system ensures data consistency, integrity, security, and optimal performance. Below is the detailed validation of the database design.

1. Validation of the Users Table:

The Users table is the foundation of the chat application, as it stores the core information about each user. Validation for this table ensures proper user registration, authentication, and management.

Primary Key Validation:

The user_id column serves as the primary key, ensuring each user has a unique identifier. Auto-increment is used to avoid manual input errors.

Unique Constraints:

The email and username columns are validated for uniqueness to prevent duplicate accounts. Regular expressions (Regex) are used to validate email format during user registration.

Password Security:

Passwords are hashed using secure algorithms like bcrypt before storage.

Input validation ensures password strength, requiring a mix of uppercase, lowercase, numbers, and special characters.

Status Validation:

The status column ensures values are limited to predefined options (e.g., "online," "offline," or "busy").

2. Validation of the Message Table:

The Message table handles the storage of real-time communication between users. Proper validation ensures accurate data storage and message delivery.

Foreign Key Validation:

The sender_id and receiver_id columns reference the user_id in the Users table, ensuring that only valid users can send or receive messages.

Foreign key constraints enforce referential integrity.

Content Validation:

The content column is validated to ensure that messages are not empty.

File uploads, such as images or videos, are validated for allowed formats and size restrictions.

3. Validation of the Message Requests Table:

The Message_Requests table facilitates communication permissions between users. Validation ensures proper tracking and management of message requests.

Primary Key and Foreign Keys:

The request_id is the primary key, providing a unique identifier for each request.

The sender_id and receiver_id reference the user_id in the Users table, ensuring that both sender and receiver exist in the database.

Status Validation:

The status column is limited to predefined values, such as "pending," "accepted," or "rejected."

This ensures the status of message requests is accurately represented at all times.

Request Date Validation:

The request_date column ensures that the date and time of the request are recorded automatically and accurately.

4. Validation of the Feedback Table

The Feedback table collects and stores user feedback about the application. Validation ensures feedback data is reliable and meaningful.

Foreign Key Validation:

The user_id column references the Users table, ensuring that feedback is associated with a valid user.

Content Validation:

The feedback_content column is validated to ensure that feedback is not empty or excessively lengthy.

Feedback is checked to prevent the inclusion of harmful or inappropriate content.

Submission Date Validation:

The submission_date column automatically records the date and time of feedback submission.

Additional Validation Techniques:

Input Sanitization:

All user inputs are sanitized to prevent SQL injection and cross-site scripting (XSS) attacks.

Data Integrity Rules:

Cascading updates and deletes are implemented in foreign key relationships to maintain data consistency.

For instance, when a user account is deleted, associated messages, message requests, and feedback entries are also removed.

Error Handling and Logging:

Validation errors, such as duplicate email addresses or invalid file formats, trigger clear error messages for users.

Logs of validation failures are maintained for debugging and system improvement.

Performance Optimization:

Indexing is applied to frequently queried columns like user_id, email, and timestamp to ensure quick data retrieval.

The database design is normalized to reduce redundancy and improve efficiency.

Scalability Validation:

Load testing ensures the database can handle increasing volumes of users, messages, and feedback without performance degradation.

Partitioning strategies may be adopted to manage large datasets effectively.

10.4. System Security Measures

Ensuring the security of a chat application is paramount to safeguarding user data, maintaining system integrity, and preventing unauthorized access. Here are essential security measures to consider implementing:

End-to-End Encryption:

Implement end-to-end encryption to secure communications between users. This ensures that messages are encrypted on the sender's device and decrypted only on the recipient's device, preventing unauthorized interception.

Secure Authentication:

Enforce strong authentication mechanisms, such as multi-factor authentication (MFA) or biometric authentication, to verify the identity of users. Use secure password hashing algorithms and enforce password complexity requirements to mitigate the risk of unauthorized access.

Data Validation and Sanitization:

Validate and sanitize user input to prevent common vulnerabilities like cross-site scripting (XSS) and SQL injection attacks. Implement server-side input validation and use parameterized queries to sanitize database inputs effectively.

Session Management:

Implement secure session management practices to prevent session hijacking and fixation attacks. Use secure, randomly generated session identifiers, and enforce session timeouts to limit the duration of active sessions.

Content Security Policy (CSP):

Implement CSP headers to control which external resources can be loaded and executed within the chat application. This helps mitigate the risk of XSS attacks by restricting the sources of executable content.

File Upload Security:

If the chat application allows file uploads, validate file types, limit file sizes, and store uploaded files in a secure location outside the web root directory. Perform thorough security checks on uploaded files to prevent malicious file execution.

Firewalls and Intrusion Detection Systems (IDS):

Deploy firewalls and IDS to monitor and filter incoming and outgoing network traffic. This helps detect and prevent unauthorized access attempts and malicious activities targeting the chat application.

Regular Software Updates:

Keep the chat application, underlying frameworks, and dependencies up to date with the latest security patches. Regularly apply updates to address known vulnerabilities and mitigate the risk of exploitation.

User Permissions and Access Control:

Implement granular user permissions and role-based access control (RBAC) to restrict access to sensitive features and data. Ensure that users only have access to the functionalities necessary for their roles.

Logging and Monitoring:

Implement comprehensive logging mechanisms to record user activities, system events, and security-related incidents. Monitor logs regularly for signs of suspicious behavior or unauthorized access attempts, and set up alerts for anomalous activities.

Security Training and Awareness:

Educate developers, administrators, and users about security best practices and potential threats. Provide training on secure coding practices, phishing awareness, and incident response procedures to enhance security awareness across the organization.

By implementing these security measures, the real-time chat application can mitigate risks, protect user privacy, and ensure a secure and reliable communication platform for users.

11. Maintenance

The maintenance of a chat application is a critical phase in its lifecycle to ensure consistent performance, security, and user satisfaction. This phase involves regular updates, bug fixes, performance optimizations, and system monitoring to maintain the application's reliability and functionality over time. Below are the key aspects of maintenance required for the chat application.

1. Corrective Maintenance:

This involves identifying and resolving errors or bugs in the application after deployment.

Bug Fixes:

Fixing any functional issues, such as errors in message delivery, incorrect file uploads, or UI glitches.

Addressing compatibility issues that may arise due to updates in browsers or devices.

Error Logging and Monitoring:

Implementing logging mechanisms to capture and analyze errors in real-time.

Regularly reviewing logs to identify recurring issues and address root causes.

Testing Updates:

Conducting regression testing to ensure that bug fixes do not introduce new issues.

2. Adaptive Maintenance

This involves modifying the application to adapt to changes in the environment, user needs, or technological advancements.

Environment Updates:

Updating server software, frameworks, and libraries (e.g., PHP, MySQL) to the latest stable versions to enhance performance and security.

Ensuring compatibility with new operating systems, browsers, and devices.

Feature Enhancements:

Adding new functionalities based on user feedback, such as group chats, video calling, or enhanced file-sharing options.

Updating the user interface (UI) to keep it modern and user-friendly.

Integration Updates:

Adapting to changes in third-party APIs or services integrated into the application, such as cloud storage or notification services.

3. Preventive Maintenance

Preventive maintenance focuses on minimizing potential issues before they occur to ensure smooth application operation.

Performance Optimization:

Monitoring database queries and optimizing them to improve response times.

Compressing images and videos to reduce storage and bandwidth usage.

Code Review and Refactoring:

Periodically reviewing and improving the codebase to ensure it remains clean, modular, and efficient.

Removing unused code and libraries to reduce vulnerabilities and improve performance.

Security Updates:

Regularly updating encryption algorithms and security protocols to combat evolving threats.

Conducting vulnerability assessments and penetration testing to identify and mitigate security risks.

Backup Management:

Automating backup processes and testing restoration procedures to ensure data integrity and availability during failures.

4. Perfective Maintenance

Perfective maintenance involves enhancing the system to improve user experience and meet new user requirements.

User Feedback Implementation:

Collecting and analyzing feedback submitted through the feedback module.

Prioritizing and implementing suggestions, such as improved message search functionality or personalized themes.

User Experience (UX) Improvements:

Updating the application design to align with current trends and improve accessibility.

Enhancing navigation and reducing the number of steps required for common tasks like sending messages or files.

Performance Improvements:

Scaling server resources to handle increased traffic as the user base grows.

Implementing caching mechanisms to speed up content delivery.

5. Monitoring and Reporting

Continuous monitoring ensures that the application runs smoothly and issues are identified proactively.

System Monitoring:

Using tools like Nagios or New Relic to monitor server health, database performance, and application uptime.

Setting up alerts for unusual activities, such as high CPU usage or excessive database queries.

Usage Analytics:

Tracking user activity to identify popular features and areas needing improvement.

Analyzing peak usage times to plan for resource scaling.

Report Generation:

Creating detailed reports on system performance, user engagement, and error trends for review by the development team.

6. Documentation Maintenance

Keeping all project documentation up to date is vital for ensuring the maintainability of the application.

Technical Documentation:

Updating API documentation, database schema diagrams, and system architecture documents after every significant change.

User Documentation:

Revising user guides and FAQs to reflect new features or changes in the application.

Maintenance Logs:

Maintaining detailed logs of updates, bug fixes, and feature additions for future reference.

7. Long-Term Scalability

Planning for the long-term growth of the application ensures it remains functional as the user base expands.

Database Scaling:

Implementing partitioning or sharding for large datasets to ensure efficient querying.

Migrating to distributed database systems if needed to handle increased traffic.

Infrastructure Upgrades:

Moving to cloud-based solutions for dynamic scaling of server resources.

Using content delivery networks (CDNs) to reduce latency and improve global access.

Feature Scalability:

Designing new features with modularity to facilitate easy integration without disrupting existing functionalities.

8. Cost Management

Maintaining an application also involves managing costs to ensure sustainability.

Resource Optimization:

Regularly reviewing and optimizing server resources to avoid unnecessary expenses.

Decommissioning unused services or infrastructure components.

Budget Planning:

Allocating budgets for system updates, maintenance tools, and team training.

12. Limitation and Future Scope

Limitations:

While the chat application is designed to offer robust functionality, it is essential to acknowledge its limitations to ensure realistic expectations and identify areas for improvement.

1. Real-Time Scalability:

The current implementation relies on basic server infrastructure, which may face performance issues with a rapidly growing user base or high traffic loads.

Advanced scaling solutions like WebSocket-based real-time communication or cloud-based services are not yet integrated.

2. Limited File Size Support:

File uploads are restricted to a predefined size limit to prevent server overload and ensure smooth performance. Larger files cannot be uploaded.

3. No Group Chat Functionality:

The application currently supports only one-to-one communication. Group chats or multi-user interactions are not implemented.

4. Platform Dependency:

The application is optimized for web browsers and lacks dedicated mobile or desktop app versions, limiting accessibility on certain devices.

5. Basic Message Requests:

Message request handling is functional but lacks advanced features like request filtering, blocking, or automated approvals.

6. Data Backup and Recovery:

While basic backup mechanisms are in place, comprehensive disaster recovery features, such as real-time replication, are not fully implemented.

7. Limited Personalization Options:

User customization options, such as themes or layouts, are minimal, potentially affecting the user experience.

8. Security Enhancements Needed:

Although security measures are implemented, advanced features like two-factor authentication, biometric login, or advanced encryption algorithms are not included.

Feature Scope of the Project

The feature scope outlines potential enhancements and future directions for the project to expand its functionality and address existing limitations.

1. Real-Time Messaging with WebSockets:

Implementing WebSocket technology will enable seamless, real-time communication, significantly improving performance and responsiveness.

2. Group Chat Support:

Adding group chat functionality will enhance collaboration and engagement by allowing multiple users to communicate in a shared space.

3. Mobile and Desktop Applications:

Developing native mobile apps for Android and iOS, along with desktop versions, will make the application accessible across all platforms.

4. Advanced File Sharing:

Introducing support for larger files, file previews, and multiple file uploads will enhance the user experience. Integration with cloud storage services like Google Drive or Dropbox can also be considered.

5. Enhanced Security Features:

Features like two-factor authentication (2FA), biometric login, and advanced encryption standards will strengthen data protection and user trust.

6. Customizable User Profiles:

Enabling users to customize their profiles with themes, avatars, and statuses will improve personalization and engagement.

7. AI-Powered Features:

Integrating AI for spam detection, automated replies, and sentiment analysis will make the application smarter and more efficient.

8. Integration with Third-Party Services:

Adding support for external APIs, such as payment gateways or social media platforms, will expand the application's utility and appeal.

9. Analytics Dashboard:

Providing administrators with an analytics dashboard to monitor user activity, message trends, and application performance will aid in data-driven decision-making.

10. Scalable Infrastructure:

Migrating to cloud-based infrastructure will enhance scalability, allowing the application to handle a larger user base and higher traffic efficiently.

11. Offline Messaging:

Implementing offline messaging will enable users to send messages even when one party is offline, with messages delivered upon reconnection.

12. Localization and Multilingual Support:

Adding support for multiple languages will broaden the application's reach and usability for diverse audiences.

13. Advanced Message Management:

Features like message editing, deletion, archiving, and search filters will improve usability and functionality.

13. Conclusion:

The development of this chat application marks a significant achievement in creating a secure, user-friendly, and efficient communication platform. By leveraging modern technologies such as HTML, CSS, JavaScript, PHP, and MySQL, the application provides real-time messaging capabilities, file sharing, and personalized user experiences. The carefully designed database with tables for users, messages, message requests, and feedback ensures data integrity and efficient management of user interactions.

Security has been prioritized at every stage of development, with features such as encrypted communications, secure authentication, input validation, and proactive monitoring mechanisms. These measures safeguard user data and protect the application from vulnerabilities, fostering trust among users. Furthermore, the incorporation of adaptive and scalable design principles ensures the application remains responsive to future technological advancements and growing user demands.

The project also emphasizes user-centric design, offering an intuitive interface and seamless navigation. Feedback mechanisms allow users to share their opinions, contributing to the continuous improvement of the platform. This approach aligns the application with modern standards of usability, accessibility, and engagement, enhancing its appeal and functionality.

In conclusion, this chat application is a robust, scalable, and secure solution that addresses the evolving needs of communication in a digital world. It serves as a versatile platform for users to connect and collaborate while setting the foundation for future enhancements. Through rigorous maintenance and regular updates, the application will continue to adapt to user expectations and technological advancements, ensuring long-term relevance and success.

BIBLIOGRAPHY

1. W3Schools. (n.d.). HTML Basic. W3Schools.
https://www.w3schools.com/html/html_basic.asp
2. Mozilla Contributors. (n.d.). <form>. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form>
3. W3Schools. (n.d.). HTML Media. W3Schools.
https://www.w3schools.com/html/html_media.asp
4. Tutorial Republic. (n.d.). HTML5 Semantic Elements. Tutorial Republic.
<https://www.tutorialrepublic.com/html-tutorial/html5-semantic-elements.php>
5. freeCodeCamp. (2021, May 7). What are Meta Tags? HTML Examples. freeCodeCamp.
<https://www.freecodecamp.org/news/what-are-meta-tags-html-examples/>
6. CodePen Contributors. (n.d.). Chat UI Examples. CodePen.
<https://codepen.io/collections/tag/chat>
7. Tutorial Republic. (n.d.). HTML Responsive Web Design. Tutorial Republic.
https://www.tutorialrepublic.com/html/html_responsive_web_design.php
8. W3Schools. (n.d.). HTML Tables. W3Schools.
https://www.w3schools.com/html/html_tables.asp
9. Mozilla Contributors. (n.d.). Accessibility in HTML. MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/Accessibility/HTML>
10. HTML CheatSheet. (n.d.). HTML Reference Guide. HTML CheatSheet.
<https://htmlcheatsheet.com/>
11. Mozilla Contributors. (n.d.). CSS Basics. MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/CSS>
12. CSS-Tricks. (n.d.). A Complete Guide to Flexbox. CSS-Tricks. <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
13. CSS-Tricks. (n.d.). A Complete Guide to Grid. CSS-Tricks. <https://css-tricks.com/snippets/css/complete-guide-grid/>
14. CodePen Contributors. (n.d.). Chat Bubble Design. CodePen. <https://codepen.io/tag/chat-bubble>
15. Mozilla Contributors. (n.d.). Using CSS Animations. MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations
16. Mozilla Contributors. (n.d.). CSS Pseudo-elements. MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>
17. Smashing Magazine. (n.d.). Guidelines for Responsive Web Design. Smashing Magazine. <https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/>
18. Mozilla Contributors. (n.d.). Using CSS Custom Properties (Variables). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties
19. W3Schools. (n.d.). CSS Media Queries. W3Schools.
https://www.w3schools.com/css/css_rwd_mediaqueries.asp
20. Mozilla Contributors. (n.d.). JavaScript Basics. MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

21. Mozilla Contributors. (n.d.). WebSockets API. MDN Web Docs.
22. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
23. Socket.IO Contributors. (n.d.). Socket.IO Documentation. Socket.IO.
<https://socket.io/docs/v4/>
24. Mozilla Contributors. (n.d.). Introduction to Events. MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/Events>
25. Mozilla Contributors. (n.d.). Promises in JavaScript. MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
26. Mozilla Contributors. (n.d.). Fetch API. MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
27. Mozilla Contributors. (n.d.). Local Storage in JavaScript. MDN Web Docs.
<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>
28. Mozilla Contributors. (n.d.). Debugging JavaScript. MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_went_wrong
29. Tutorial Republic. (n.d.). Creating Notifications in JavaScript. Tutorial Republic.
<https://www.tutorialspoint.com/how-to-create-desktop-notifications-using-javascript>
30. Mozilla Contributors. (n.d.). Regular Expressions in JavaScript. MDN Web Docs.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
31. Eric Meyer - Eric Meyer on CSS: Mastering the Language of Web Design - New Riders
32. Shay Howe - Learn to Code HTML and CSS: Develop and Style Websites - New Riders
33. Jennifer Niederst Robbins - HTML and XHTML Pocket Reference - O'Reilly Media
34. Dave Raggett, Jenny Lam, Ian Alexander - HTML 4 for the World Wide Web: VQS (Visual QuickStart Guides) - Peachpit Press
35. Chuck Easttom - HTML, CSS, and JavaScript All in One: Covering HTML5, CSS3, and ES6 - McGraw-Hill Education
36. Jeremy Keith, Jeffrey Sambells - DOM Scripting: Web Design with JavaScript and the Document Object Model - Apress
37. Jason Garber - Head First HTML and CSS - O'Reilly Media
38. Chuck Musciano, Bill Kennedy - HTML & XHTML: The Definitive Guide - O'Reilly Media
39. Chuck Easttom - HTML, CSS, and JavaScript All in One: Covering HTML5, CSS3, and ES6 - McGraw-Hill Education
40. Mark Pilgrim - HTML5: Up and Running - O'Reilly Media
41. Jennifer Niederst Robbins - Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics - O'Reilly Media
42. Matt West - HTML5 Foundations - Treehouse
43. Bruce Lawson, Remy Sharp - Introducing HTML5 - New Riders
44. Chuck Easttom - HTML, CSS, and JavaScript All in One: Covering HTML5, CSS3, and ES6 - McGraw-Hill Education
45. Elizabeth Castro - HTML, XHTML, and CSS, Sixth Edition: Visual QuickStart Guide -

46. Certainly! Here's the list without the headings:
47. Eric Meyer - Cascading Style Sheets: The Definitive Guide - O'Reilly Media
48. Jon Duckett - CSS3: Designing for the Web - Wiley
49. Rachel Andrew - The New CSS Layout - Smashing Magazine
50. David McFarland - CSS: The Missing Manual - O'Reilly Media
51. Chris Coyier - CSS Secrets: Better Solutions to Everyday Web Design Problems - O'Reilly Media
52. Louis Lazaris - CSS3 Foundations – Treehouse
53. Tiffany B. Brown, Kerry Butters, Sandeep Panda - Jump Start CSS – SitePoint
54. David Powers - React Solutions: Dynamic Web Design Made Easy - Friends of Ed
55. Christian Wenz - React Phrasebook - Addison-Wesley
56. W. Jason Gilmore - Easy React Websites with the Zend Framework - Springer
57. David Sklar, Adam Trachtenberg - React Cookbook - O'Reilly Media
58. Matthew MacDonald - React: The Missing Manual - O'Reilly Media
59. David Powers - React 7 Solutions: Dynamic Web Design Made Easy - Friends of Ed
60. David Sklar, Adam Trachtenberg - React Cookbook - O'Reilly Media
61. Dan Cederholm - CSS3 for Web Designers - A Book Apart
62. Andy Clarke - Transcending CSS: The Fine Art of Web Design - New Riders
63. Zoe Mickley Gillenwater - Flexible Web Design: Creating Liquid and Elastic Layouts with CSS - New Riders
64. Dan Cederholm - Bulletproof Web Design: Improving flexibility and protecting against worst-case scenarios with HTML5 and CSS3 - New Riders
65. Jennifer Niederst Robbins - CSS Cookbook - O'Reilly Media
66. Rasmus Lerdorf, Kevin Tatroe, Peter MacIntyre - Programming React - O'Reilly Media
67. Matt Zandstra - React Objects, Patterns, and Practice - Apress
68. Paul Redmond - Docker for React Developers - Leanpub
69. Estelle Weyl - CSS: The Definitive Guide - O'Reilly Media

70. Tiffany B. Brown, Kerry Butters, Sandeep Panda - Jump Start CSS – SitePoint
71. David Powers - React Solutions: Dynamic Web Design Made Easy - Friends of Ed
72. Christian Wenz - React Phrasebook - Addison-Wesley
73. W. Jason Gilmore - Easy React Websites with the Zend Framework - Springer
74. David Sklar, Adam Trachtenberg - React Cookbook - O'Reilly Media
75. Matthew MacDonald - React: The Missing Manual - O'Reilly Media
76. David Powers - React 7 Solutions: Dynamic Web Design Made Easy - Friends of Ed
77. David Sklar, Adam Trachtenberg - React Cookbook - O'Reilly Media
78. Dan Cederholm - CSS3 for Web Designers - A Book Apart
79. Andy Clarke - Transcending CSS: The Fine Art of Web Design - New Riders
80. Zoe Mickley Gillenwater - Flexible Web Design: Creating Liquid and Elastic Layouts with CSS - New Riders
81. Dan Cederholm - Bulletproof Web Design: Improving flexibility and protecting against worst-case scenarios with HTML5 and CSS3 - New Riders
82. Jennifer Niederst Robbins - CSS Cookbook - O'Reilly Media
83. Rasmus Lerdorf, Kevin Tatroe, Peter MacIntyre - Programming React - O'Reilly Media
84. Matt Zandstra - React Objects, Patterns, and Practice - Apress
85. Paul Redmond - Docker for React Developers - Leanpub
86. Julie C. Meloni - Apache: The Definitive Guide - O'Reilly Media
87. David Sklar, Adam Trachtenberg - React Cookbook - O'Reilly Media
88. Matt Zandstra - React Objects, Patterns, and Practice - Apress
89. Michael McLaughlin - Oracle React Cookbook - O'Reilly Media
90. David Powers - React Solutions: Dynamic Web Design Made Easy - Friends of Ed
91. Tricia Ballard, Colin Bell, Justin Morford - Pro React and jQuery - Apress
92. Tim Converse, Joyce Park - React 5 Recipes – Apress
93. John Coggeshall - Zend React 5 Certification Study Guide - Sams Publishing
94. Douglas Crockford - JavaScript: The Good Parts - O'Reilly Media
95. Kyle Simpson - You Don't Know JS (book series) - O'Reilly Media
96. Marijn Haverbeke - Eloquent JavaScript: A Modern Introduction to Programming - No Starch Press
97. David Flanagan - JavaScript: The Definitive Guide - O'Reilly Media
98. Eric Elliott - Programming JavaScript Applications - O'Reilly Media
99. Nicholas C. Zakas - Maintainable JavaScript - O'Reilly Media
100. Addy Osmani - Learning JavaScript Design Patterns - O'Reilly Media
101. Stoyan tefanov - JavaScript Patterns - O'Reilly Media
102. Axel Rauschmayer - Speaking JavaScript - O'Reilly Media
103. Cody Lindley - JavaScript Enlightenment - O'Reilly Media
104. Nicholas C. Zakas - Professional JavaScript for Web Developers - Wrox

REFERENCES:

1. https://www.w3schools.com/html/html_basic.asp
2. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form>
3. https://www.w3schools.com/html/html_media.asp
4. <https://www.tutorialrepublic.com/html-tutorial/html5-semantic-elements.php>
5. <https://www.freecodecamp.org/news/what-are-meta-tags-html-examples/>
6. <https://codepen.io/collections/tag/chat>
7. https://www.tutorialspoint.com/html/html_responsive_web_design.htm
8. https://www.w3schools.com/html/html_tables.asp
9. <https://developer.mozilla.org/en-US/docs/Web/Accessibility/HTML>
10. <https://htmlcheatsheet.com/>
11. <https://developer.mozilla.org/en-US/docs/Web/CSS>
12. <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
13. <https://css-tricks.com/snippets/css/complete-guide-grid/>
14. <https://codepen.io/tag/chat-bubble>
15. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations
16. <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>
17. <https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/>
18. https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties
19. https://www.w3schools.com/css/css_rwd_mediaqueries.asp
20. <https://css-tricks.com/almanac/selectors/h/hover/>
21. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics
22. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
23. <https://socket.io/docs/v4/>
24. <https://developer.mozilla.org/en-US/docs/Web/Events>
25. https://www.w3schools.com/xml/ajax_intro.asp
26. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
27. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
28. <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

29. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_went_wrong
30. <https://www.tutorialspoint.com/how-to-create-desktop-notifications-using-javascript>
31. <https://www.php.net/manual/en/tutorial.php>
32. <https://www.php.net/manual/en/book.session.php>
33. <https://www.php.net/manual/en/websocket.examples.php>
34. <https://code.tutsplus.com/tutorials/how-to-create-a-simple-web-based-chat-application--cms-26745>
35. <https://www.php.net/manual/en/book.errorfunc.php>
36. <https://www.tutorialrepublic.com/php-tutorial/php-ajax-example.php>
37. <https://www.php.net/manual/en/book.pdo.php>
38. <https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php>
39. https://www.w3schools.com/php/php_file_upload.asp
40. <https://www.tutorialrepublic.com/php-tutorial/php-mysql-crud-application.php>
41. <https://dev.mysql.com/doc/refman/8.0/en/tutorial.html>
42. <https://www.mysqltutorial.org/mysql-create-table.aspx>
43. <https://www.mysqltutorial.org/mysql-data-types.aspx>
44. https://www.w3schools.com/sql/sql_join.asp
45. <https://www.essentialsql.com/get-ready-to-learn-sql-database-normalization-explained-in-simple-english/>
46. <https://dev.mysql.com/doc/refman/8.0/en/optimization.html>
47. <https://www.mysqltutorial.org/mysql-index.aspx>
48. <https://www.mysqltutorial.org/mysql-foreign-key.aspx>
49. <https://www.mysql.com/products/workbench/backup-recovery/>
50. <https://www.mysqltutorial.org/mysql-triggers.aspx>
51. <https://www.tutorialrepublic.com/php-tutorial/php-mysql-chat-application.php>
52. <https://code.tutsplus.com/tutorials/build-a-real-time-chat-system-with-php-and-websockets--cms-34369>
53. <https://www.phpzag.com/build-live-chat-system-with-ajax-php-mysql/>
54. https://www.youtube.com/watch?v=CdvNS_nkGfQ
55. <https://www.tutsmake.com/php-ajax-chat-application/>
56. <https://phppot.com/php/simple-php-chat-application-using-websocket/>
57. <https://www.codexworld.com/create-simple-php-chat-application/>
58. <https://www.section.io/engineering-education/webrtc-in-php/>
59. <https://www.phpflow.com/php/php-simple-chat-application-using-php-and-mysql/>
60. <https://html.spec.whatwg.org/multipage/>
61. https://www.w3schools.com/html/html_images.asp
62. <https://www.freecodecamp.org/news/how-to-use-html-input-types/>
63. <https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/autocomplete>
64. https://www.tutorialspoint.com/html/html_layout.htm
65. <https://html5doctor.com/element-index/>
66. <https://www.w3docs.com/learn-html/html-audio-and-video.html>
67. https://www.w3schools.com/html/html_favicon.asp
68. <https://www.htmlgoodies.com/tutorials/adding-links-in-html/>
69. <https://www.tutorialsteacher.com/html/html-input-element>

70. https://developer.mozilla.org/en-US/docs/Web/CSS/Backgrounds_and_borders
71. https://www.w3schools.com/css/css3_borders.as
72. <https://css-tricks.com/snippets/css/using-nth-child-pseudo-class/>
73. <https://developer.mozilla.org/en-US/docs/Web/CSS/pointer-events>
74. <https://www.smashingmagazine.com/2020/06/designing-user-friendly-chatbots/>
75. <https://css-tricks.com/how-css-perspective-works/>
76. <https://developer.mozilla.org/en-US/docs/Web/CSS/white-space>
77. https://www.w3schools.com/css/css3_transitions.asp
78. <https://codepen.io/chriscoyier/pen/NaePRe>
79. <https://css-tricks.com/building-a-simple-chat-ui-with-flexbox/>
80. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
81. <https://developer.mozilla.org/en-US/docs/Web/API/EventSource>
82. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
83. <https://getcomposer.org/doc/00-intro.md>
84. <https://www.php.net/manual/en/function.password-hash.php>
85. <https://www.php.net/manual/en/filter.filters.sanitize.php>
86. <https://phphtherightway.com/#sessions>
87. <https://www.php.net/manual/en/pdo.prepare.php>
88. <https://www.tutorialrepublic.com/php-tutorial/php-file-handling.php>
89. <https://www.php.net/manual/en/features.http-auth.php>
90. <https://www.php.net/manual/en/function.json-encode.php>
91. https://www.tutorialspoint.com/php/php_file_handling.htm
92. <https://www.javatpoint.com/javascript-closest>
93. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs
94. <https://javascript.info/intro#what-can-t-in-browser-javascript-do>
95. <https://www.geeksforgeeks.org/how-to-create-a-form-dynamically-with-the-javascript/>
96. <https://www.geeksforgeeks.org/javascript-string-reference/?ref=shm>
97. <https://www.geeksforgeeks.org/javascript-math-reference/?ref=shm>
98. <https://www.w3schools.com/css/>
99. <https://drfurithemes.com/farmart2/>
100. <https://designreset.com/preview-equation/default/index.html>