WhatsApp Web Clone

APROJECTREPORT
for
Mini Project (KCA353)
Session (2023-24)

Submitted by

Ayush Dwivedi
(2200290140045)
Dushyant Kaushik
(2200290140058)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Vipin Kumar
(Associate Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MARCH 2024)

CERTIFICATE

Certified that **Ayush Dwivedi 2200290140045, Dushyant Kaushik2200290140058**has/ have carried out the project work having "**WhatsApp Web Clone**" (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Ayush Dwivedi (2200290140045)

Dushyant Kaushik (2200290140058)

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

Date:

Dr. Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Ayush Dwivedi
Dushyant Kaushik

WhatsApp Web clone

ABSTRACT

The primary objective of the project is to create a user-friendly and feature-rich platform that mirrors the functionality of the popular messaging application. The MERN stack serves as the technological foundation, ensuring a robust and scalable architecture for efficient data handling and real-time communication. The application allows users to seamlessly connect to their WhatsApp accounts through a web interface, enabling them to send and receive messages, multimedia files, and engage in group conversations. The use of MongoDB ensures secure and scalable storage of user data, while Express.js and Node.js handle the server-side logic, facilitating smooth communication between the client and server. The user interface is designed to mimic the familiar WhatsApp Web layout, providing an intuitive and responsive experience for users. The report delves into the design principles, technical aspects, and challenges encountered during the development process. It highlights the successful replication of key features, such as end-to-end encryption and real-time message synchronization.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Ayush Dwivedi

Dushyant Kaushik

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The WhatsApp Clone application is a comprehensive and user-friendly messaging platform developed using the MERN tech stack, designed to emulate the functionality of WhatsApp. The app caters to individuals who seek a familiar and efficient messaging experience, allowing users to connect with friends, family, and colleagues seamlessly. What's Up Clone comes equipped with essential features, including user registration, account creation, one-on-one messaging, multimedia sharing, group chats, and real-time updates. Users can personalize their profiles, customize chat settings, and enjoy a smooth and reliable messaging experience. The MERN tech stack ensures a robust and scalable foundation for a secure and feature-rich WhatsApp clone.

1.1 Overview

The WhatsApp Messenger is a robust and user-friendly application developed with the MERN (MongoDB, Express.js, React, Node.js) tech stack. This cutting-edge messaging platform replicates the functionality of WhatsApp, providing a seamless communication experience for users. Tailored for individuals who seek efficient and secure messaging, What's Up Messenger offers essential features such as user registration, account creation, one-on-one messaging, multimedia sharing, and group chats. The MERN tech stack ensures a scalable and reliable foundation, delivering a WhatsApp clone that combines familiarity with enhanced performance for an optimal messaging experience.

1.2 Description

The WhatsApp website clone is a meticulously crafted replication of the popular messaging platform, designed to offer users a familiar and seamless communication experience on the web. Mirroring the iconic green and white interface, this clone provides a user-friendly environment for instant messaging, voice calls, and multimedia sharing.

With a responsive and intuitive design, the WhatsApp website clone ensures accessibility across various devices, enabling users to stay connected anytime, anywhere. The platform retains the core functionalities of the original app, allowing users to create individual and group chats, share photos, videos, and documents effortlessly.

Security is a top priority in this clone, incorporating end-to-end encryption to safeguard user privacy and confidential conversations. Users can log in securely, synchronize their contacts, and enjoy real-time communication without compromising data integrity.

The WhatsApp website clone brings innovation to the online messaging sphere, offering a reliable and efficient communication platform for both personal and professional use. Its familiar interface, coupled with robust security measures, makes it a compelling choice for those seeking a web-based messaging solution reminiscent of the widely embraced original application.

1.3 Key Features

User-Friendly Interface: The Whats App Website Clone is an intuitive and user-friendly interface, ensuring a seamless and enjoyable messaging experience for both senders and recipients.
Authentication and Authorization: Robust login and signup functionalities guarantee a personalized and secure environment for users, ensuring the confidentiality of their conversations.
Messaging Management: Users can effortlessly send, receive, and manage their messages, creating a diverse and dynamic platform for communication.
Chat Functionality: Users can easily initiate and engage in conversations, fostering a convenient and efficient messaging experience.
Message Details: Detailed message information empowers users with essential context, aiding them in making informed decisions during conversations.
Search and Categorization: The app facilitates a powerful search functionality, allowing users to find specific conversations swiftly, enhancing the overall messaging experience.
Secure Messaging: The incorporation of end-to-end encryption ensures safe and reliable communication, instilling confidence in users regarding the privacy and security of their messages.

1.4 Objectives

Effortless Messaging Experience: The core objective is to simplify the process of messaging, providing a convenient platform for users to engage in seamless conversations through the Whats App Web Clone.

User Engagement: Foster user engagement through an appealing and intuitive design, encouraging users to explore the messaging app and communicate effortlessly with friends, family, and colleagues.

Enhanced Connectivity Opportunities: Create a messaging platform that not only benefits users but also offers them the opportunity to connect effectively by providing a reliable space for secure and private communication.

Scalability and Performance: Develop a scalable architecture that can accommodate growth while ensuring optimal performance, even during peak usage periods on the Whats App Web Clone.

Personalizing Messaging Experiences: Users could have a customized messaging experience. Users can engage in conversations based on their preferences and needs without restriction. In the messaging app, users can connect with others based on their interests and contacts.

Elevating Communication Process: Messaging apps have accelerated the entire communication procedure for users. They can chat with others from the comfort of their own space, without the need for physical meetings. It saves enormous amounts of time and expedites conversations.

Retargeting Users: Messaging platforms have simplified the process of retargeting users for businesses. While users are messaging online, the messaging app collects information about

them. Periodically, users can be engaged by sending them personalized messages, promotions, and updates through Whats App Web Clone.

Access to User Data: Messaging platforms gather valuable data about user behavior, preferences, and demographics. This data can be analyzed to make informed decisions, refine features, and enhance the overall user experience on Whats App Web Clone.

Flexibility and Scalability: Messaging apps can easily adapt to changing user needs and scale their operations to accommodate growth. This flexibility is crucial for apps experiencing fluctuations in demand or those looking to expand their features within Whats App Web Clone.

CHAPTER 2

FEASIBILITY STUDY

The Whats App Web Clone project aims to assess the viability and practicality of developing a messaging application using MongoDB, Express.js, React, and Node.js. This feasibility study evaluates technical, operational, economic, and scheduling aspects to determine the project's viability.

2.1 Technical Feasibility

Technology Stack Suitability: Evaluate the specific technical requirements of the messaging application and ensure that the chosen technology stack, inspired by WhatsApp, aligns with these needs. Consider factors such as real-time message updates, scalability, and responsiveness to provide a seamless messaging experience for users.

Integration Capabilities: Assess the ease of integration with external services such as authentication systems, third-party APIs, and multimedia sharing functionalities, ensuring the WhatsApp clone supports seamless communication with these external components to enhance user engagement.

Scalability Testing: Conduct scalability tests to simulate increased user loads and message volumes. Analyze the performance of the chosen technology stack under varying conditions to ensure it can handle growth without compromising the speed and reliability of delivering messages promptly, just like WhatsApp.

Security Measures: Identify and implement robust security measures within the WhatsApp clone's technology stack to safeguard user conversations, multimedia sharing, and sensitive information. Prioritize end-to-end encryption and compliance with industry standards and regulations to ensure a secure messaging environment for users, mirroring the security features of WhatsApp.

2.2 Operational Feasibility

User Training and Adoption: Evaluate the ease with which end-users, including administrators, can adapt to the WhatsApp clone-based system. Plan for training programs or user guides to facilitate a smooth transition and optimal utilization of the messaging platform.

Maintenance Protocols: Define protocols for system maintenance, updates, and bug fixes in the WhatsApp clone. Ensure that the chosen technology stack's modular structure allows for efficient troubleshooting and updates without disrupting the overall operation of the messaging application.

Operational Scalability: Consider how well the operational aspects of the platform can scale as the user base and message volumes increase. Assess the chosen technology stack's ability to handle concurrent users, manage databases efficiently, and adapt to evolving operational demands in the WhatsApp clone.

2.3 Economic Feasibility

Development and Infrastructure Costs: Conduct a detailed analysis of development costs, including expenses related to hiring skilled developers for the Whats App Web Clone, infrastructure, and any licensing fees. Compare these costs against the potential benefits and revenue projections for the messaging application.

Return on Investment (ROI): Develop a comprehensive business model to estimate the ROI based on revenue streams, marketing strategies, and projected user acquisition for the Whats App Web Clone. Assess the time it will take for the project to break even and generate profits within the messaging app market.

Market Trends and Competitive Analysis: Stay abreast of market trends and conduct a thorough competitive analysis within the messaging application sector for the Whats App Web Clone. Adjust economic considerations based on the app's unique value proposition and its ability to meet current market demands for seamless communication.

Flexibility for Future Enhancements: Assess the economic feasibility of incorporating future enhancements and features into the Whats App Web Clone. Ensure that the initial investment allows for scalability and adaptability to evolving market trends in the messaging app landscape.

CHAPTER 3

REQUIREMENT ANALYSIS

The WhatsApp Website Clone project aims to replicate the functionality and user experience of the popular mobile messaging application, WhatsApp, for the web platform. This comprehensive requirement analysis categorizes the identified requirements into functional and non-functional categories, providing a structured approach to the development process.

3.1 Functional Requirements

The functional requirements for the WhatsApp website clone encompass essential features for seamless communication. This includes user registration and login, individual and group messaging, multimedia sharing, real-time communication through instant messaging and voice calls, contact synchronization, and device compatibility. The clone must faithfully replicate the intuitive design of the original app, ensuring an optimal user experience. Additionally, it should support optional features such as video calls. These functional requirements collectively define the core functionalities that users expect, mirroring the familiar and widely-used features of the mobile application on the web platform.

3.1.1 User Authentication and Registration

User Registration: Users should be able to create accounts by providing essential information, including a valid phone number.

Login: Secure authentication mechanisms should be implemented to allow users to log in to the website using their registered phone numbers.

Password Recovery: A mechanism for password recovery, such as email or SMS verification, should be in place.

3.1.2 Messaging Features

Individual Chats: Users should be able to initiate and participate in one-on-one conversations with their contacts.

Group Chats: The clone should support group messaging, allowing users to create, join, and manage group conversations.

Multimedia Sharing: Users must be able to share text, images, videos, documents, and voice messages within chats.

3.1.3 Real-Time Communication

Instant Messaging: The platform should provide real-time message delivery, ensuring quick and seamless communication.

Voice Calls: Implementation of voice calling functionality, allowing users to make and receive calls within the web interface.

Video Calls (Optional): If feasible, consider adding video calling functionality for a more comprehensive communication experience.

3.1.4 Synchronization and Connectivity

Contact Synchronization: Users should be able to sync their contacts to the website, facilitating easy communication with existing connections.

Device Compatibility: Ensure compatibility across various devices and web browsers for a consistent user experience.

3.2 Non-Functional Requirements

The non-functional requirement analysis for the WhatsApp website clone emphasizes key aspects such as security, usability, performance, and compliance. Security measures include end-to-end encryption and secure authentication. Usability considerations focus on an intuitive design and responsive layout. Performance optimization ensures fast loading times and smooth interactions, while scalability is crucial for accommodating a growing user base. Rigorous testing and quality assurance processes are in place. Technical specifications, project management milestones, and comprehensive documentation are outlined. Legal and compliance aspects, including a clear privacy policy and adherence to data protection regulations, are integral to the non-functional requirements.

3.2.1 Security Requirements

End-to-End Encryption: Implement robust encryption mechanisms to secure user messages and media, ensuring privacy.

Secure Authentication: Utilize secure login practices, such as two-factor authentication, to protect user accounts from unauthorized access

Data Protection: Implement measures to safeguard user data, ensuring compliance with privacy regulations.

3.2.2 Usability and User Interface

Intuitive Design: Create a user-friendly and visually appealing interface that mirrors the WhatsApp mobile app for a seamless transition.

Responsive Layout: Ensure the website is responsive, providing an optimal viewing experience across various screen sizes and devices.

3.2.3 Performance and Scalability

Performance Optimization: Optimize the website's performance to deliver fast loading times and smooth user interactions.

Scalability: Design the system to handle a growing user base and increased data load, ensuring scalability and responsiveness.

3.2.4 Testing and Quality Assurance

Test Cases: Develop comprehensive test cases to cover functional, security, and usability aspects of the website clone.

Quality Assurance: Implement a rigorous quality assurance process to identify and rectify any bugs or issues before deployment.

3.2.5 Technical Specifications

Frontend Technology: Utilize modern frontend technologies such as HTML5, CSS3, and JavaScript (React or Angular) for a dynamic and responsive user interface.

Backend Technology: Choose a scalable backend technology (Node.js) to handle user authentication, messaging, and data storage.

Database: Implement a secure and scalable database (MongoDB) to store user data and chat histories.

Real-Time Communication: Explore technologies like WebSocket for real-time communication between users.

3.2.6 Project Management

Project Phases: Divide the project into distinct phases, including planning, design, development, testing, and deployment.

Milestones: Set specific milestones for each phase to track progress and ensure timely completion.

3.2.7 Documentation

Technical Documentation: Create comprehensive technical documentation covering the system architecture, APIs, and data flow.

User Documentation: Develop user guides and documentation to assist users in navigating and utilizing the features of the WhatsApp website clone.

This structured categorization of requirements into functional and non-functional aspects provides a clear roadmap for the development of the WhatsApp Website Clone. By addressing these requirements, the project aims to deliver a secure, user-friendly, and functionally rich web-based messaging platform that replicates the seamless communication experience of the original WhatsApp Web Clone.

CHAPTER 4

ARCHITECTURAL DESIGN

The database design for the WhatsApp website clone employs a robust and scalable structure. Utilizing technologies like MongoDB or PostgreSQL, it efficiently stores user data, chat histories, and multimedia content. The design prioritizes data security, ensuring sensitive information is safeguarded. A well-organized schema facilitates seamless retrieval and management of user interactions, enabling quick access to messages and media. The database architecture supports the real-time communication demands of the platform, contributing to a smooth and responsive user experience. Overall, the database design is tailored to handle the dynamic and growing nature of a web-based messaging platform.

4.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a visual representation that illustrates how data moves within a system. The 0-level DFD, also known as the context diagram, provides a high-level view of the entire system, showing its interactions with external entities. For the WhatsApp website clone project, the 0-level DFD showcases the core components and their relationships.

4.1.1 0-level DFD

External Entities

Users: The primary external entity representing individuals using the WhatsApp website clone to send and receive messages, make calls, and share media.

Contacts: External entities representing the users' contact lists, involved in the synchronization process.

Processes

User Authentication and Registration: This process manages the creation of user accounts and their authentication during login. It interacts with the Users entity.

Messaging Engine: The core of the system, responsible for handling individual and group messaging, multimedia sharing, and real-time communication. It interacts with both Users and Contacts entities.

Contact Synchronization: Manages the synchronization of contacts from users' devices, ensuring up-to-date contact information is available within the system.

Media Storage: Responsible for storing and retrieving multimedia content such as images, videos, and documents shared within chats. It interfaces with the Messaging Engine.

Data Stores

User Database: Stores user profiles, authentication credentials, and account information.

Message Database: Contains message logs, storing the content, timestamps, and sender/receiver details.

Contact Database: Stores information about users' contacts, facilitating the synchronization process.

Media Repository: A storage system for multimedia content shared in chats.

Data Flows

User Registration Data Flow: From Users to User Authentication and Registration: User registration data, including name, phone number, and password, flows into the system for account creation.

Login Data Flow: From Users to User Authentication and Registration: User login credentials are provided for authentication.

Messaging Data Flow: Text messages, multimedia content, and other communication inputs flow from users to the Messaging Engine. Delivered messages and multimedia content flow back to users.

Contact Synchronization Data Flow: Requests to synchronize contacts are initiated by users. Updated contact information flows to the Contacts entity.

Media Storage Data Flow: Multimedia content shared in chats is stored in the Media Repository.

External Interfaces

User Interface: The interface through which users interact with the system, providing input and receiving output.

Device Interfaces: Interfaces with users' devices for contact synchronization and real-time communication.

The 0-level DFD for the WhatsApp website clone project provides a holistic view of the system's key components, data flows, and external interfaces. This diagram serves as a foundation for further development, aiding in the understanding of how data moves within the system and interactions with external entities. The identified processes and data stores lay the groundwork for a more detailed DFD at lower levels, facilitating a comprehensive understanding of the project's architecture and functionality.

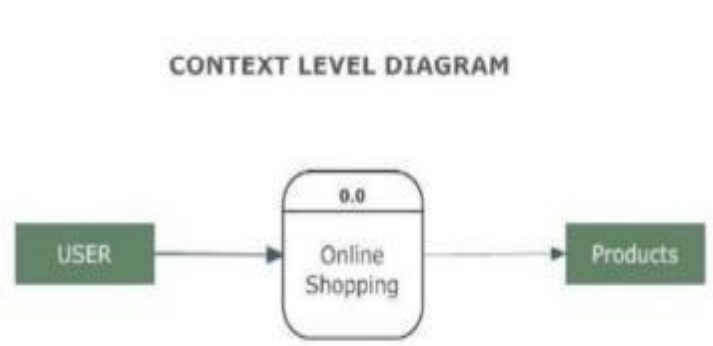


Fig 4.1.1 Level 0 DFD

4.1.2 Level 1 DFD:

A Level 1 Data Flow Diagram (DFD) provides a high-level overview of the data flow within a system, showcasing major processes, data sources, and interactions. In the context of the WhatsApp website clone project, the Level 1 DFD outlines the primary components and their interactions.

Components of the Level 1 DFD

User Interface:

The user interface represents the web-based platform where users interact with the WhatsApp website clone. It includes elements for registration, login, messaging, and multimedia sharing.

User Authentication and Registration:

This process handles user registration and authentication. When a user registers, the system validates and stores the user's information securely. For authentication, it checks user credentials against stored data.

Chat Management:

The Chat Management process oversees the creation and management of individual and group chats. It handles requests to create new chats, add users to existing chats, and manages the overall flow of messages within the chats.

Message Processing:

This component is responsible for processing incoming and outgoing messages. It ensures proper formatting, encryption, and decryption of messages. Multimedia files, such as images, videos, and documents, are also processed within this module.

Real-Time Communication:

The Real-Time Communication process manages the instant messaging, voice calls, and potentially video calls. It ensures the seamless and secure flow of real-time communication between users.

Contact Synchronization:

This process handles the synchronization of contacts for users. When a user logs in, the system syncs their contacts to facilitate easy communication with existing connections.

Database Management:

The Database Management process interacts with the database, storing and retrieving user data, chat histories, and multimedia content. It maintains the integrity and security of the data stored in the system.

Data Flow

User Registration and Authentication:

Data flows from the User Interface to the User Authentication and Registration process. During registration, user details are captured and sent for validation and storage. During login, the process validates user credentials and allows access to the user interface.

Chat Initialization and Management:

The User Interface initiates a request for creating a new chat or managing existing ones. This request is processed by the Chat Management process, which updates the chat details in the system.

Message Flow:

Messages initiated by users are sent from the User Interface to the Message Processing component. This component formats, encrypts, and sends the message to the intended recipient. Incoming messages are processed similarly, ensuring proper display on the User Interface.

Real-Time Communication:

Real-time communication data flows between the User Interface and the Real-Time Communication process. This includes data related to voice calls, video calls (if implemented), and instant messaging.

Contact Synchronization:

The User Interface triggers the Contact Synchronization process during login. The process fetches and updates the user's contact list, ensuring the latest contacts are available for communication.

Database Interaction:

Various processes interact with the Database Management component to store and retrieve data. For instance, during user registration, user details are stored. Messages and multimedia files are stored and retrieved during the messaging process.

The Level 1 DFD for the WhatsApp website clone project provides a structured visualization of major processes and data flows within the system. It lays the foundation for further detailing and understanding the intricacies of data movement, user interactions, and system functionalities in the development of this web-based messaging platform.

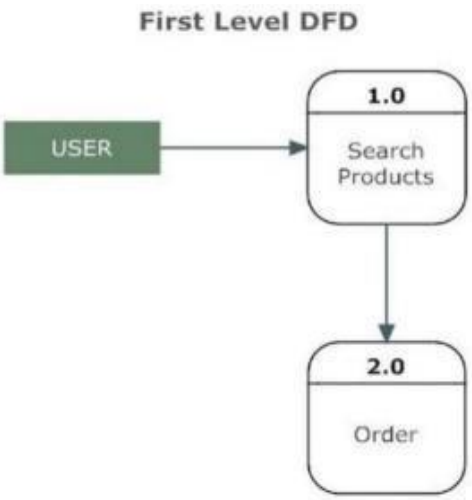


Fig 4.1.2 Level 1 DFD

4.1.3 2nd Level DFD

A Second Level Data Flow Diagram provides a detailed representation of the processes, data stores, and data flows within a system. In the context of the WhatsApp Website Clone project, this diagram elucidates the major components and their interactions, offering insights into the flow of information and functionalities.

User Module

The User Module is the primary entry point for individuals interacting with the WhatsApp website clone.

Process - User Authentication

The User Authentication process ensures secure login functionality. It takes user credentials, validates them against stored data, and grants access if authentication is successful.

Process - User Registration

This process handles user registration, capturing essential information such as phone numbers. Once verified, the system creates a new user account and updates the user database.

Chat Module

The Chat Module manages all aspects related to individual and group chats, enabling users to communicate seamlessly.

Process - Individual Chats

This process facilitates one-on-one messaging. It manages the creation, sending, and receiving of individual chat messages, updating the chat history accordingly.

Process - Group Chats

For group communication, this process allows users to create, join, and manage group chats. It handles group messages, ensuring proper distribution among members.

Process - Multimedia Sharing

Multimedia Sharing manages the exchange of various media formats within chats, including text, images, videos, documents, and voice messages.

Real-Time Communication Module

The Real-Time Communication Module governs the instantaneous exchange of messages and media.

Process - Instant Messaging

Ensuring real-time delivery, the Instant Messaging process handles the transmission of messages between users, keeping conversations synchronous.

Process - Voice Calls

For voice communication, this process manages the initiation, acceptance, and termination of voice calls within the web interface.

Process - Video Calls (Optional)

If implemented, the Video Calls process enables video communication, ensuring a seamless video calling experience.

Synchronization and Connectivity Module

This module ensures connectivity and synchronization of user data across devices.

Process - Contact Synchronization

Contact Synchronization allows users to sync their contacts to the website, facilitating easy communication with existing connections.

Process - Device Compatibility

Ensuring compatibility across devices, the Device Compatibility process makes certain that the website functions optimally on various browsers and screen sizes.

Security Module

The Security Module is crucial for safeguarding user data and ensuring secure interactions.

Process - End-to-End Encryption

This process implements robust encryption mechanisms to secure user messages and media, ensuring end-to-end privacy.

Process - Secure Authentication

Secure Authentication handles secure login practices, including two-factor authentication, protecting user accounts from unauthorized access.

Process - Data Protection

Implementing measures to safeguard user data, the Data Protection process ensures compliance with privacy regulations.

Usability and User Interface Module

This module focuses on creating a user-friendly and visually appealing interface.

Process - Intuitive Design

The Intuitive Design process ensures the website's user interface is user-friendly, mirroring the familiarity of the WhatsApp mobile app.

Process - Responsive Layout

The Responsive Layout process ensures the website adapts to various screen sizes and devices for an optimal user experience.

Performance and Scalability Module

Ensuring optimal performance and scalability to accommodate a growing user base.

Process - Performance Optimization

Performance Optimization focuses on enhancing the website's speed and responsiveness, delivering a seamless user experience.

Process - Scalability

The Scalability process designs the system to handle increased data load, ensuring responsiveness as the user base grows.

This Second Level Data Flow Diagram provides a detailed overview of the key modules and processes within the WhatsApp Website Clone project. Each process contributes to the overall functionality, security, and usability of the web-based messaging platform, replicating the seamless communication experience of the original WhatsApp mobile app.

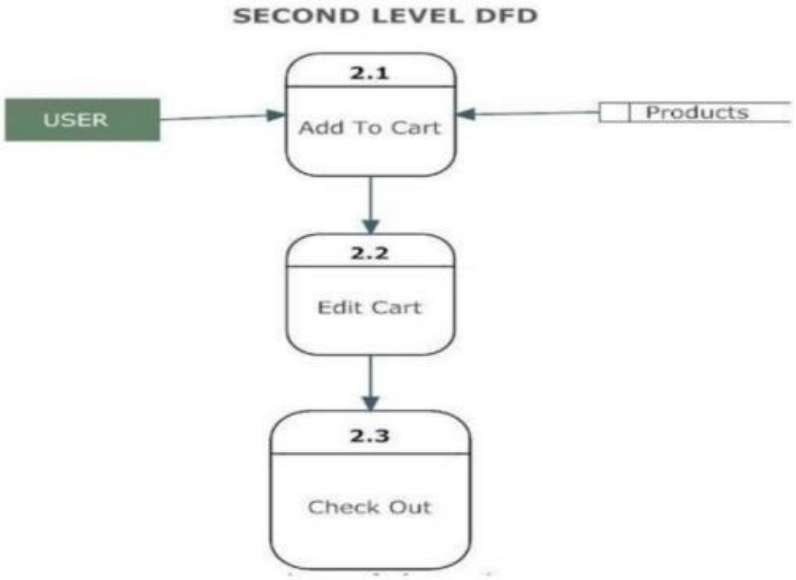


Fig 4.1.3 Level 2 DFD

4.2 Use Case diagram

A Use Case Diagram illustrates the various interactions between actors (users) and the system, showcasing the functionalities and actions the system performs. In the context of the WhatsApp Website Clone project, the diagram outlines the primary use cases and actors involved in the system.

Actors:

User: Represents individuals interacting with the WhatsApp website clone.

System: Encompasses all the functionalities and processes within the website clone.

Use Cases:

User Authentication:

Primary Actor: User

Description: Users initiate the authentication process by providing their credentials (phone number and password) to log in securely.

Trigger: User wants to access their account on the website.

User Registration:

Primary Actor: User

Description: New users go through the registration process, providing necessary information to create a new account.

Trigger: User is registering for the first time.

Individual Chats:

Primary Actor: User

Description: Users can initiate, participate, and manage one-on-one chat conversations with their contacts.

Trigger: User wants to have a private conversation with another user.

Group Chats:

Primary Actor: User

Description: Users can create, join, and manage group conversations with multiple participants.

Trigger: User wants to engage in a group discussion.

Multimedia Sharing:

Primary Actor: User

Description: Users can share various multimedia content, including text, images, videos, documents, and voice messages, within chats.

Trigger: User wants to share media content with a contact or group.

Instant Messaging:

Primary Actor: User

Description: Users can send and receive instant messages, ensuring real-time communication.

Trigger: User wants to communicate with another user in real-time.

Voice Calls:

Primary Actor: User

Description: Users can initiate, receive, and terminate voice calls within the web interface.

Trigger: User wants to make a voice call to another user.

Video Calls (Optional):

Primary Actor: User

Description: If implemented, users can engage in video calls for a more comprehensive communication experience.

Trigger: User wants to make a video call to another user.

Contact Synchronization:

Primary Actor: User

Description: Users can synchronize their contacts with the website, ensuring easy communication with existing connections.

Trigger: User wants to access their contacts on the website.

Device Compatibility:

Primary Actor: User

Description: Ensures the website functions optimally across various devices and web browsers.

Trigger: User accesses the website from different devices.

End-to-End Encryption:

Primary Actor: System

Description: The system ensures the implementation of robust encryption mechanisms to secure user messages and media.

Trigger: User initiates a secure conversation.

Secure Authentication:

Primary Actor: System

Description: The system implements secure login practices, such as two-factor authentication, to protect user accounts.

Trigger: User initiates the login process.

Data Protection:

Primary Actor: System

Description: The system implements measures to safeguard user data, ensuring compliance with privacy regulations.

Trigger: User interacts with the system.

Intuitive Design:

Primary Actor: System

Description: The system ensures the website's user interface is intuitive and user-friendly.

Trigger: User navigates through the website.

Responsive Layout:

Primary Actor: System

Description: The system ensures the website's layout is responsive, providing an optimal viewing experience across devices.

Trigger: User accesses the website from different devices.

Privacy Policy:

Primary Actor: System

Description: The system ensures the display of a clear privacy policy outlining how user data will be handled and protected.

Trigger: User accesses the website.

Relationships:

Association: Represents the relationship between users and specific functionalities.

Include: Represents functionalities that are included in other use cases.

Extend: Represents optional or extended functionalities.

The Use Case Diagram provides a visual representation of the primary interactions and functionalities within the WhatsApp Website Clone project, offering a comprehensive overview for both developers and stakeholders involved in the system.

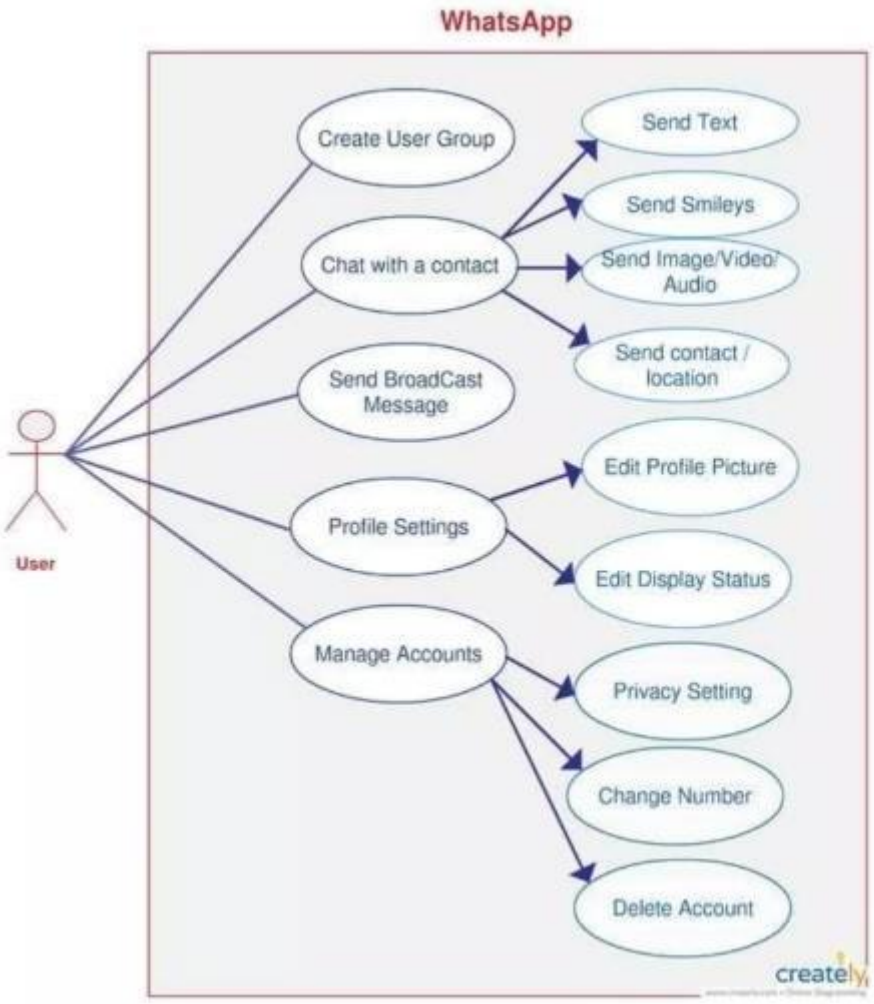


Fig. 4.2 Use case diagram

4.3 ER Diagram

The Entity-Relationship (E-R) diagram for the WhatsApp Website Clone project visualizes the structure of the database, illustrating entities, their relationships, and attributes. The E-R diagram serves as a blueprint for designing the underlying database architecture that supports the functionality of the web-based messaging platform.

Entities:

1. **User:**
    - *Attributes:* UserID (Primary Key), Phone Number, Password, First Name, Last Name, Registration Date.

- *Description:* Represents the individuals registered on the platform, with a unique UserID as the primary key.

2. **Chat:**
    - *Attributes:* ChatID (Primary Key), Type (Individual or Group), Creation Date.
    - *Description:* Represents individual and group chat conversations, uniquely identified by ChatID.
3. **Message:**
    - *Attributes:* MessageID (Primary Key), Content, Timestamp.
    - *Description:* Represents individual messages exchanged within chats, uniquely identified by MessageID.
4. **Group:**
    - *Attributes:* GroupID (Primary Key), Name, Creation Date.
    - *Description:* Represents the groups created by users for group chat functionality, uniquely identified by GroupID.
5. **Media:**
    - *Attributes:* MediaID (Primary Key), Type (Image, Video, Document, Voice), File URL, Timestamp.
    - *Description:* Represents multimedia content shared within messages, uniquely identified by MediaID.
6. **Contact:**
    - *Attributes:* ContactID (Primary Key), UserID1, UserID2.
    - *Description:* Represents the connections between users, uniquely identified by ContactID.

Relationships:

1. **User-Chat Relationship:**
    - *Type:* One-to-Many
    - *Description:* A user can be associated with multiple chats (both individual and group), but each chat is associated with only one user.
2. **Chat-Message Relationship:**
    - *Type:* One-to-Many
    - *Description:* Each chat can have multiple messages, but each message belongs to only one chat.
3. **User-Group Relationship:**
    - *Type:* Many-to-Many (via Contact entity)

- *Description:* Users can create multiple groups, and each group can have multiple users. The Contact entity helps represent the membership of users in groups.

4. **User-Message Relationship:**
    - *Type:* One-to-Many
    - *Description:* A user can send multiple messages, but each message is sent by only one user.
5. **Message-Media Relationship:**
    - *Type:* One-to-One
    - *Description:* Each message can have associated multimedia content (image, video, document, or voice), but each multimedia item corresponds to only one message.
6. **User-Contact Relationship:**
    - *Type:* Many-to-Many
    - *Description:* Users can have multiple contacts, and each contact can be connected to multiple users.

Attributes:

1. **User Attributes:**
    - *UserID:* Unique identifier for each user.
    - *Phone Number:* The contact number associated with the user.
    - *Password:* Securely stored user password.
    - *First Name, Last Name:* User's personal information.
    - *Registration Date:* Date when the user registered on the platform.
2. **Chat Attributes:**
    - *ChatID:* Unique identifier for each chat.
    - *Type:* Indicates whether the chat is individual or group.
    - *Creation Date:* Date when the chat was initiated.
3. **Message Attributes:**
    - *MessageID:* Unique identifier for each message.
    - *Content:* The actual text content of the message.
    - *Timestamp:* Time when the message was sent.
4. **Group Attributes:**
    - *GroupID:* Unique identifier for each group.
    - *Name:* Name of the group.

- *Creation Date:* Date when the group was created.

5. **Media Attributes:**
    - *MediaID:* Unique identifier for each media item.
    - *Type:* Indicates the type of media (image, video, document, or voice).
    - *File URL:* URL or reference to the actual media file.
    - *Timestamp:* Time when the media was shared.
6. **Contact Attributes:**
    - *ContactID:* Unique identifier for each contact.
    - *UserID1, UserID2:* Identifiers of the connected users.

Cardinalities:

1. **User-Chat Cardinality:**
    - *User:* One
    - *Chat:* Many
2. **Chat-Message Cardinality:**
    - *Chat:* One
    - *Message:* Many
3. **User-Group Cardinality:**
    - *User:* Many
    - *Group:* Many
4. **User-Message Cardinality:**
    - *User:* One
    - *Message:* Many
5. **Message-Media Cardinality:**
    - *Message:* One
    - *Media:* One
6. **User-Contact Cardinality:**
    - *User:* Many
    - *Contact:* Many

The E-R diagram encapsulates the essential entities, relationships, and attributes of the WhatsApp Website Clone project, serving as a foundation for designing an efficient and scalable database that supports the diverse functionalities of the web-based messaging platform.

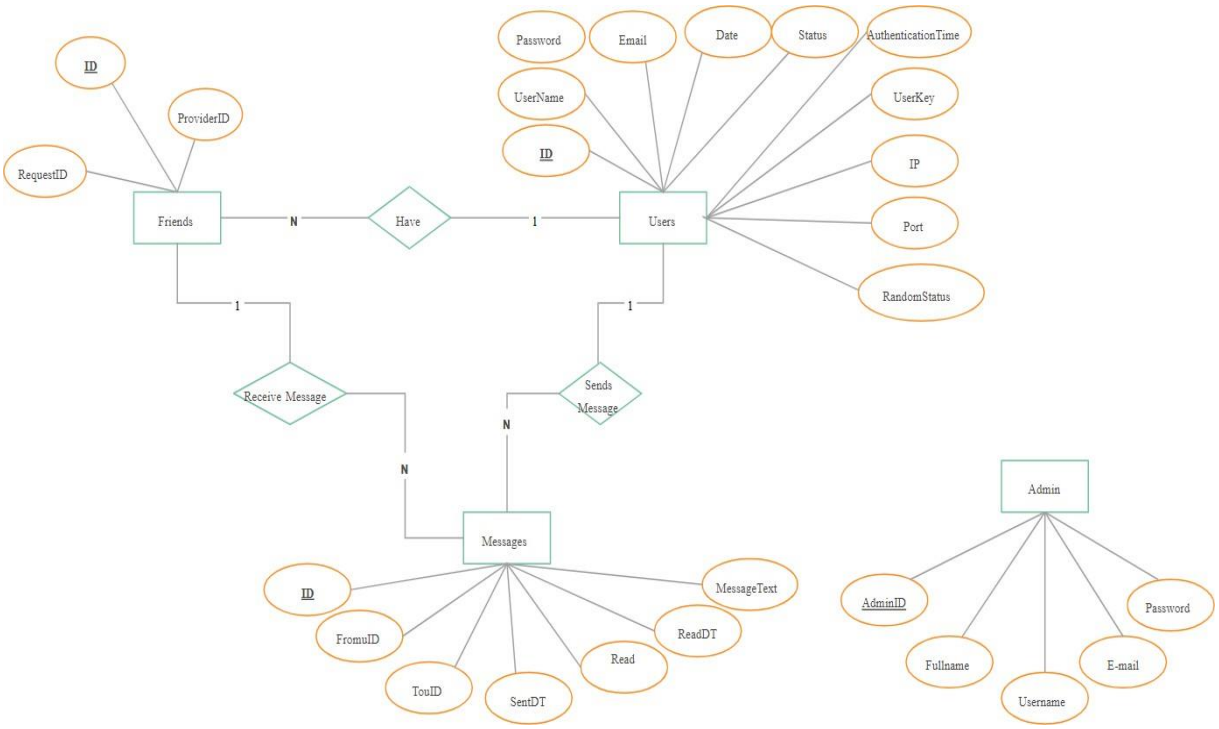


Fig. 4.3 ER diagram

CHAPTER 5

HARDWARE AND SOFTWARE SPECIFICATION

5.1 Hardware Specifications

Component	Minimum Requirement	Recommended Requirement
Processor	Dual-core processor, 2.0 GHz or higher	Quad-core processor, 2.5 GHz or higher
RAM	4 GB	8 GB or higher
Storage	128 GB SSD	256 GB SSD or higher
Browser	Latest version of popular web browsers	Latest version of Google Chrome or Firefox

Table 5.1 Hardware Specification

5.2 Software Specifications

Component	Minimum Requirement	Recommended Requirement
Operating System	Windows 10 or macOS Catalina (or latest versions)	Windows 11 or macOS Monterey (or latest versions)
Database	Mongo DB	Mongo DB
Server-Side Tech	Node.js 1(or latest versions)	Node.js 14 (or latest versions)
Frontend Framework	React Js (or latest versions)	React Js(or latest versions)
Backend Framework	Express.js 4 (or latest versions)	Express.js 5 (or latest versions)
Messaging Library	Socket.io 3.0 (or latest versions)	Socket.io 4.0 (or latest versions)
Encryption	OpenSSL 1.1 or equivalent	OpenSSL 1.3 or equivalent

Table 5.2 Software Specification

Explanation of Specifications

1. **Processor:** A dual-core processor is the minimum requirement, but a quad-core processor is recommended for improved performance, especially during real-time communication and media handling.
2. **RAM:** A minimum of 4 GB RAM is necessary, but 8 GB or more is recommended to ensure smooth multitasking and handling of large datasets.
3. **Storage:** A minimum of 128 GB SSD is recommended for faster data access and application loading times.

4. **Network Interface:** A 1 Gbps Ethernet connection is recommended to ensure fast and stable data transfer, crucial for real-time communication.
5. **Display Resolution:** The minimum display resolution of 1366 x 768 pixels is sufficient, but a higher resolution like 1920 x 1080 pixels is recommended for a better user interface.
6. **Browser:** The latest version of popular web browsers like Google Chrome or Firefox is the minimum requirement for ensuring compatibility with modern web technologies.
7. **Operating System:** The project is compatible with Windows 10 or macOS Catalina as a minimum requirement. However, using the latest operating systems, such as Windows 11 or macOS Monterey, is recommended for improved security and performance.
8. **Web Server:** Apache 2.4 or Nginx 1.16 is the minimum requirement, but using the latest versions is recommended for better security and performance.
9. **Database:** MySQL 5.7 or PostgreSQL 10 are the minimum requirements, but using the latest versions is recommended for enhanced features, performance, and security.
10. **Server-Side Tech:** Node.js 12 or Python 3.7 is the minimum requirement, but using the latest versions is recommended for better performance and support for new language features.
11. **Frontend Framework:** React 16 or Angular 8 is the minimum requirement, but using the latest versions is recommended for improved development efficiency and access to new features.
12. **Backend Framework:** Express.js 4 or Django 2 is the minimum requirement, but using the latest versions is recommended for better performance, security, and developer tools.
13. **Messaging Library:** Socket.io 3.0 or Django Channels 3 is the minimum requirement, but using the latest versions is recommended for improved real-time communication capabilities.
14. **Encryption:** OpenSSL 1.1 or an equivalent encryption library is the minimum requirement to ensure the security of user data during transmission. Using the latest encryption standards, such as OpenSSL 1.3, is recommended for enhanced security.

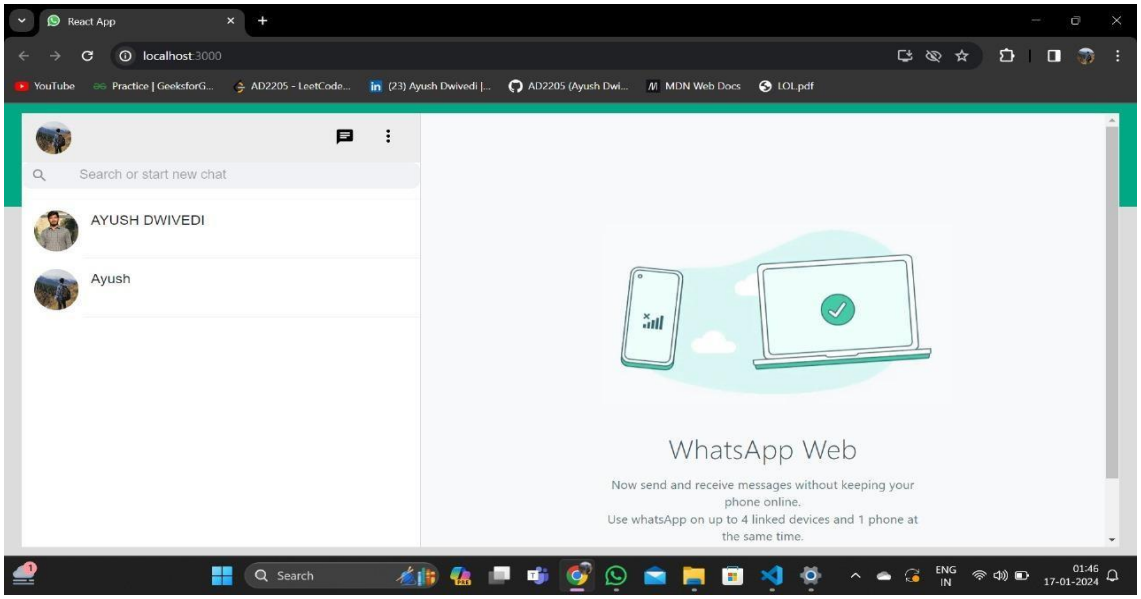
These hardware and software specifications are essential to ensure the WhatsApp Web Clone project's optimal performance, security, and compatibility with modern web technologies. Adhering to the recommended requirements will result in a more robust and scalable system, capable of handling the demands of a web-based messaging platform effectively.

CHAPTER 6

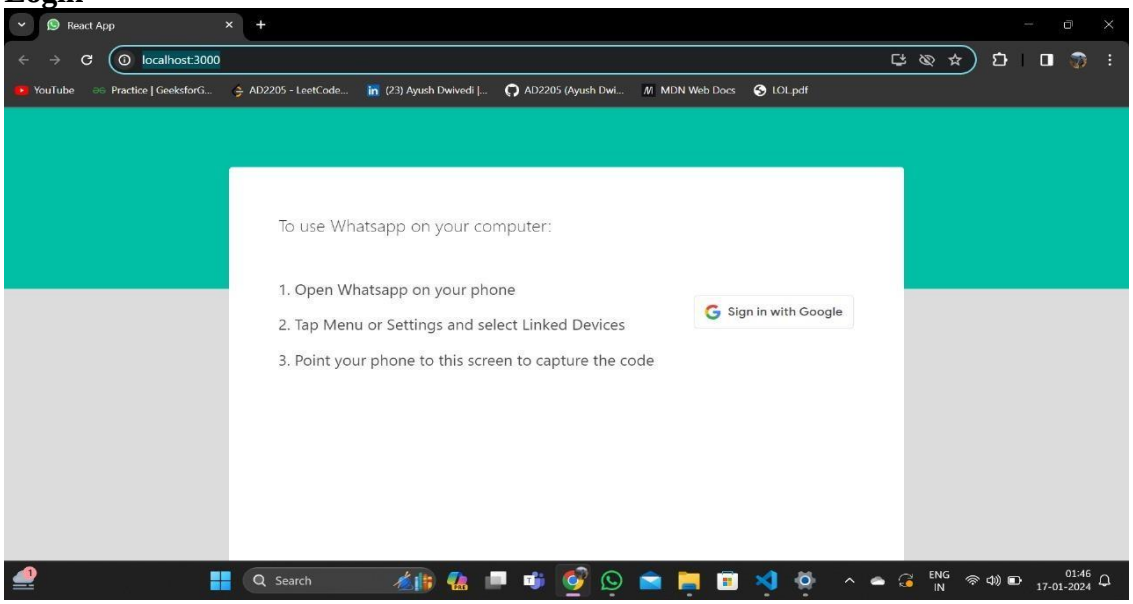
DESIGN

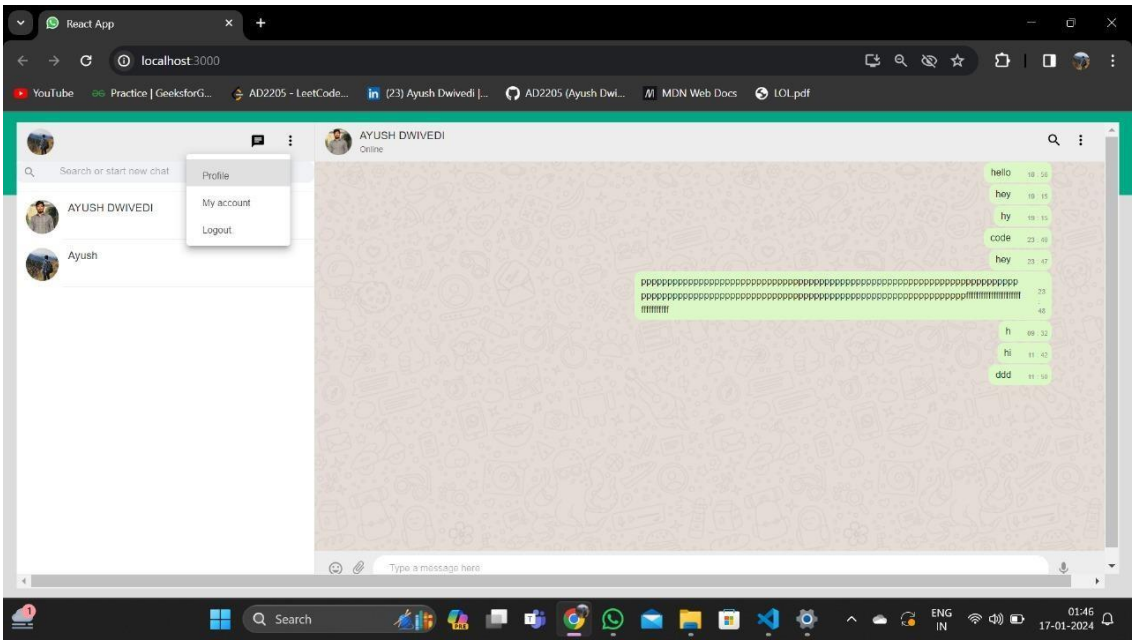
In the snapshot above, you'll find the main screen of the messaging app. It features a navigation bar resembling an online store, a search bar, and a list of ongoing conversations. Users exploring the app can search for specific chats and choose to navigate through categories to find and connect with their preferred contacts. This user-friendly interface allows for easy access to conversations and ensures a seamless experience for users to connect and communicate effortlessly.

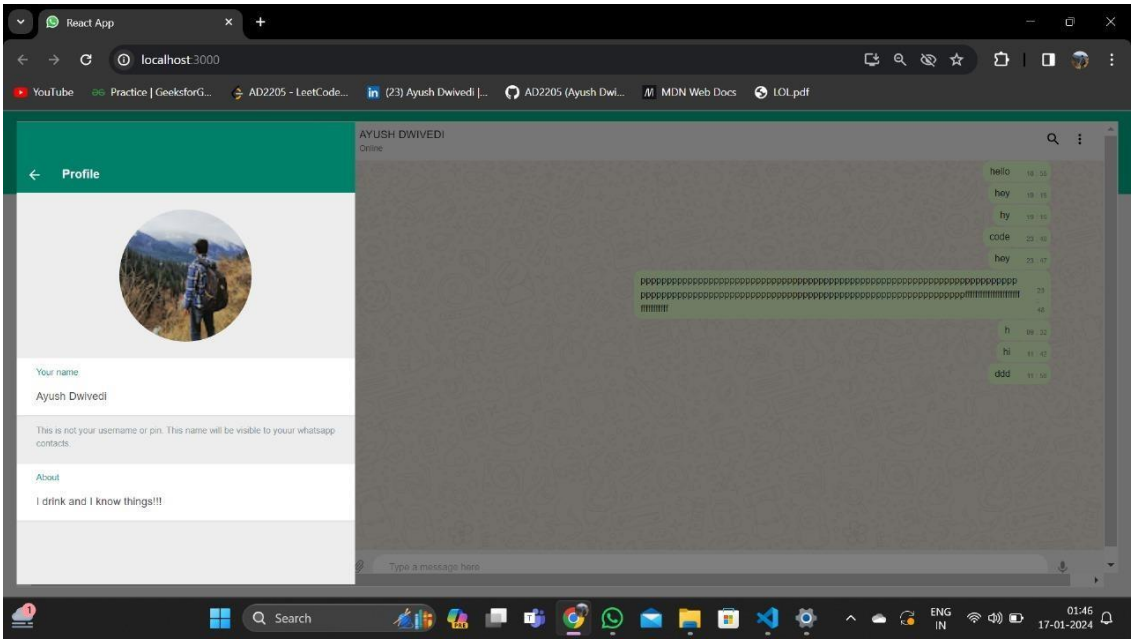
Chat Box

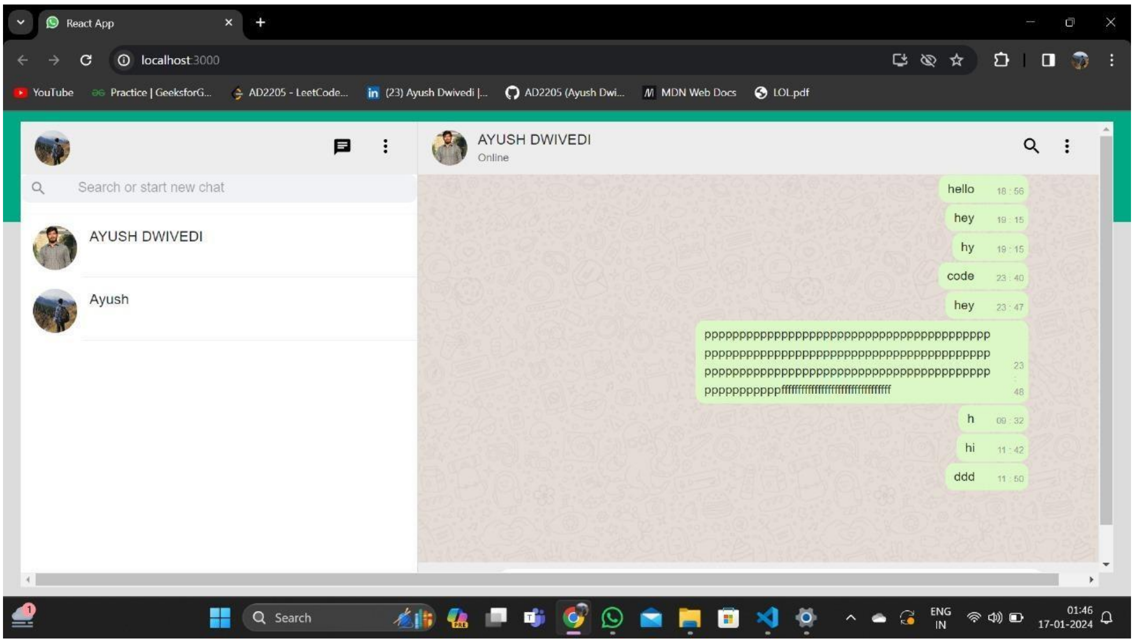


Login




React App
localhost:3000
YouTube
Practice | GeeksforG...
AD2205 - LeetCode...
(23) Ayush Dwivedi |...
AD2205 (Ayush Dwi...
MDN Web Docs
LOL.pdf
Search or start new chat
Profile
My account
Logout
AYUSH DWIVEDI
Ayush
AYUSH DWIVEDI
Online
hello
hey
hy
code
hey
h
hi
ddd
Type a message here
Search
ENG
IN
01:46
17-01-2024


React App
localhost:3000
YouTube
Practice | GeeksforG...
AD2205 - LeetCode...
(23) Ayush Dwivedi |...
AD2205 (Ayush Dwi...
MDN Web Docs
LOL.pdf
Profile
Your name
Ayush Dwivedi
This is not your username or pin. This name will be visible to your whatsapp contacts.
About
I drink and I know things!!!
AYUSH DWIVEDI
Online
hello
hey
hy
code
hey
h
hi
ddd
Type a message here
Search
ENG
IN
01:46
17-01-2024


React App
localhost:3000
YouTube
Practice | GeeksforG...
AD2205 - LeetCode...
(23) Ayush Dwivedi |...
AD2205 (Ayush Dwi...
MDN Web Docs
LOL.pdf
Search or start new chat
AYUSH DWIVEDI
Ayush
AYUSH DWIVEDI
Online
hello 18 : 56
hey 19 : 15
hy 19 : 15
code 23 : 40
hey 23 : 47
h 00 : 32
hi 11 : 42
ddd 11 : 50
Search
ENG
IN
01:46
17-01-2024

CHAPTER 7

CODING

Package-lock.json


```
{
  "name": "Whatsapp_Clone",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "dependencies": {
        "multer": "^1.4.5-lts.1"
      }
    },
    "node_modules/append-field": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/append-field/-/append-field-1.0.0.tgz",
      "integrity": "sha512-klpgFSWLW1ZEs8svjfb7g4qWY0YS5imI82dTg+QahUvJ8YqAY0P10Uk8tTyh9ZGuYEZEMaeJYCF5BFuX552hsw=="
    },
    "node_modules/buffer-from": {
      "version": "1.1.2",
      "resolved": "https://registry.npmjs.org/buffer-from/-/buffer-from-1.1.2.tgz",
      "integrity": "sha512-E+XQCRwSbaaiChtv6k6Dwgc+bx+Bs6vuKJHHl5kox/BaKbhiXzqQOwK4cO22yElGp2OCmjwVhT3HmxgyPGnJfQ=="
    },
    "node_modules/busboy": {
      "version": "1.6.0",
      "resolved": "https://registry.npmjs.org/busboy/-/busboy-1.6.0.tgz",
```

"integrity": "sha512-8SFQbg/0hQ9xy3UNTB0YEnsNBbWfhf7RtnzpL7TkBiTBRfrQ9Fxcnz7VJsleJpyp6rVLvXiuORqjlHi5q+PYuA==",

"dependencies": {

"streamsearch": "^1.1.0"

},

"engines": {

"node": ">=10.16.0"

}

},

"node_modules/concat-stream": {

"version": "1.6.2",

"resolved": "https://registry.npmjs.org/concat-stream/-/concat-stream-1.6.2.tgz",

"integrity": "sha512-27HBghJxjiZtIk3Ycvn/4kbJk/1uZuJFfuPEns6LaEvpvG1f0hTea8lilrouyo9mVc2GWdcEZ8OLoGmSADlrCw==",

"engines": [

"node >= 0.8"

],

"dependencies": {

"buffer-from": "^1.0.0",

"inherits": "^2.0.3",

"readable-stream": "^2.2.2",

"typedarray": "^0.0.6"

}

},

"node_modules/core-util-is": {

"version": "1.0.3",

"resolved": "https://registry.npmjs.org/core-util-is/-/core-util-is-1.0.3.tgz",

"integrity": "sha512-ZQBvi1DcpJ4GDqanjucZ2Hj3wEO5pZDS89BWbkcrvdxksJorwUDDZamX9ldFkp9aw2lmBDLgkObEA4DWNJ9FYQ=="

},

"node_modules/inherits": {

```
    "version": "2.0.4",
    "resolved": "https://registry.npmjs.org/inherits/-/inherits-2.0.4.tgz",
    "integrity": "sha512-k/vGaX4/Yla3WzyMCvTQOXYeIHvqOKtnqBduzTHpzpQZzAskKMhZ2K+EnBiSM9zGSoIFeMpXKxa4dYeZIQqewQ=="
  },
  "node_modules/isarray": {
    "version": "1.0.0",
    "resolved": "https://registry.npmjs.org/isarray/-/isarray-1.0.0.tgz",
    "integrity": "sha512-VLghIWNM6ELQzo7zwmcg0NmTVyWKYjvIeM83yjp0wRDTmUnrM678fQbcKBo6n2CJEF0szoG//ytg+TKla89ALQ=="
  },
  "node_modules/media-typer": {
    "version": "0.3.0",
    "resolved": "https://registry.npmjs.org/media-typer/-/media-typer-0.3.0.tgz",
    "integrity": "sha512-dq+qelQ9akHpcOl/gUVRTxVIOkAJ1wR3QAvb4RsVjS8oVoFjDGTc679wJYmUmknUF5HwMLOgb5O+a3KxfWapPQ==",
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/mime-db": {
    "version": "1.52.0",
    "resolved": "https://registry.npmjs.org/mime-db/-/mime-db-1.52.0.tgz",
    "integrity": "sha512-sPU4uV7dYlvtWJxwwxHD0PuihVNiE7TyAbQ5SWxDCB9mUYvOgroQOwYQQOKPJ8CIbE+1ETVlOoK1UC2nU3gYvg==",
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/mime-types": {
    "version": "2.1.35",
```

```
    "resolved": "https://registry.npmjs.org/mime-types/-/mime-types-2.1.35.tgz",
    "integrity": "sha512-ZDY+bPm5zTTF+YpCrAU9nK0UgICYPT0QtT1NZWFv4s++TNkcgVaT0g6+4R2uI4MjQjzysHB1zxuWL50hzaeXiw==",
    "dependencies": {
      "mime-db": "1.52.0"
    },
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/minimist": {
    "version": "1.2.8",
    "resolved": "https://registry.npmjs.org/minimist/-/minimist-1.2.8.tgz",
    "integrity": "sha512-2yyAR8qBkN3YuheJanUpWC5U3bb5osDywNB8RzDVlDwDHbocAJveqqj1u8+SVD7jkWT4yvsHCpWqqWqAxb0zCA==",
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/mkdirp": {
    "version": "0.5.6",
    "resolved": "https://registry.npmjs.org/mkdirp/-/mkdirp-0.5.6.tgz",
    "integrity": "sha512-FP+p8RB8OWpF3YZBCrP5gtADmtXApB5AMLn+vdyA+PyxCjrCs00mjyUozssO33cwDeT3wNGdLxJ5M//YqtHAJw==",
    "dependencies": {
      "minimist": "^1.2.6"
    },
    "bin": {
      "mkdirp": "bin/cmd.js"
    }
  },
```

```
    "node_modules/multer": {
      "version": "1.4.5-lts.1",
      "resolved": "https://registry.npmjs.org/multer/-/multer-1.4.5-lts.1.tgz",
                                                    "integrity": "sha512-
ywPWvcDMeH+z9gQq5qYHCCy+ethsk4goepZ45GLD63fOu0YcNecQxi64nDs3qluZB+mur
G3/D4dJ7+dGctcCQQ==",
      "dependencies": {
        "append-field": "^1.0.0",
        "busboy": "^1.0.0",
        "concat-stream": "^1.5.2",
        "mkdirp": "^0.5.4",
        "object-assign": "^4.1.1",
        "type-is": "^1.6.4",
        "xtend": "^4.0.0"
      },
      "engines": {
        "node": ">= 6.0.0"
      }
    },
    "node_modules/object-assign": {
      "version": "4.1.1",
      "resolved": "https://registry.npmjs.org/object-assign/-/object-assign-4.1.1.tgz",
                                                    "integrity": "sha512-
rJgTQnkUnH1sFw8yT6VSU3zD3sWmu6sZhIseY8VX+GRu3P6F7Fu+JNDoXfklElbLJSnc3
FUQHVe4cU5hj+BcUg==",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/process-nextick-args": {
      "version": "2.0.1",
            "resolved":    "https://registry.npmjs.org/process-nextick-args/-/process-nextick-args-
2.0.1.tgz",
```

"integrity": "sha512-3ouUOpQhtgrbOa17J7+uxOTpITYWaGP7/AhoR3+A+/1e9skrzelGi/dXzEYyvbxubEF6Wn2ypscTKiKJFFn1ag=="

},

"node_modules/readable-stream": {

"version": "2.3.8",

"resolved": "https://registry.npmjs.org/readable-stream/-/readable-stream-2.3.8.tgz",

"integrity": "sha512-8p0AUk4XODgIewSi0l8Epjs+EVnWiK7NoDIEGU0HhE7+ZyY8D1IMY7odu5lRrFXGg71L15KG8QrPmum45RTtdA==",

"dependencies": {

"core-util-is": "~1.0.0",

"inherits": "~2.0.3",

"isarray": "~1.0.0",

"process-nextick-args": "~2.0.0",

"safe-buffer": "~5.1.1",

"string_decoder": "~1.1.1",

"util-deprecate": "~1.0.1"

}

},

"node_modules/safe-buffer": {

"version": "5.1.2",

"resolved": "https://registry.npmjs.org/safe-buffer/-/safe-buffer-5.1.2.tgz",

"integrity": "sha512-Gd2UZBJDkXlY7GbJxfsE8/nvKkUEU1G38c1siN6QP6a9PT9MmHB8GnpscSmMJSoF8LOIrt8ud/wPtojys4G6+g=="

},

"node_modules/streamsearch": {

"version": "1.1.0",

"resolved": "https://registry.npmjs.org/streamsearch/-/streamsearch-1.1.0.tgz",

"integrity": "sha512-Mcc5wHehp9aXz1ax6bZUyY5afg9u2rv5cqQI3mRrYkGC8rW2hM02jWuwjtL++LS5qinSyhj2QfLyNsuc+VsExg==",

"engines": {

"node": ">=10.0.0"

```
    }
  },
  "node_modules/string_decoder": {
    "version": "1.1.1",
    "resolved": "https://registry.npmjs.org/string_decoder/-/string_decoder-1.1.1.tgz",
    "integrity": "sha512-n/ShnvDi6FHbbVfviro+WojiFzv+s8MPMHBczVePfUpDJLwoLT0ht1l4YwBCbi8pJAveEEdnkHyPyTP/mzRfwg==",
    "dependencies": {
      "safe-buffer": "~5.1.0"
    }
  },
  "node_modules/type-is": {
    "version": "1.6.18",
    "resolved": "https://registry.npmjs.org/type-is/-/type-is-1.6.18.tgz",
    "integrity": "sha512-TkRKr9sUTxEH8MdfuCSP7VizJyzRNMjj2J2do2Jr3Kym598JVdEksuzPQCnlFPW4ky9Q+iA+ma9BGm06XQBy8g==",
    "dependencies": {
      "media-typer": "0.3.0",
      "mime-types": "~2.1.24"
    },
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/typedarray": {
    "version": "0.0.6",
    "resolved": "https://registry.npmjs.org/typedarray/-/typedarray-0.0.6.tgz",
    "integrity": "sha512-/aCDEGatGvZ2BIk+HmLf4ifCJFwvKFNb9/JeZPMulfgFracn9QFcAf5GO8B/mweUjSoblS5In0cWhqpfs/5PQA=="
  },
  "node_modules/util-deprecate": {
```

```
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/util-deprecate/-/util-deprecate-1.0.2.tgz",
                                                    "integrity": "sha512-
EPD5q1uXyFxJpCrLnCc1nHnq3gOa6DZBocAIiI2TaSCA7VCJ1UJDMagCzIkXNsUYfD1da
K//LTEQ8xiIbrHtcw=="
    },
    "node_modules/xtend": {
      "version": "4.0.2",
      "resolved": "https://registry.npmjs.org/xtend/-/xtend-4.0.2.tgz",
                                                    "integrity": "sha512-
LKYU1iAXJXUgAXn9URjiu+MWhyUXHsvfp7mcuYm9dSUKK0/CjtrUwFAxD82/mCWbt
LsGjFIad0wIsod4zrTAEQ==",
      "engines": {
        "node": ">=0.4"
      }
    }
  }
}
```


Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const container = document.getElementById('root');
const root = ReactDOM.createRoot(container);
root.render(

  <React.StrictMode>
    <App />
```

```
  </React.StrictMode>
);

reportWebVitals();
```

App.js

```
import Messenger from './Components/Messenger';
import { GoogleOAuthProvider } from '@react-oauth/google';
import AccountProvider from './context/AccountProvider';

function App() {

                const clientId = '488878505715-jrd2hvvkskdgq5ugu8q24bu746ui8b8k.apps.googleusercontent.com';
  return (
    <GoogleOAuthProvider clientId={clientId} >
      <AccountProvider>
    <Messenger/>
    </AccountProvider>
    </GoogleOAuthProvider>
  );
}

export default App;
```

packet.js

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
```

```
"dependencies": {
  "@emotion/react": "^11.11.1",
  "@emotion/styled": "^11.11.0",
  "@mui/icons-material": "^5.14.14",
  "@mui/material": "^5.14.8",
  "@react-oauth/google": "^0.11.1",
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "axios": "^1.6.0",
  "jwt-decode": "^3.1.2",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-scripts": "5.0.1",
  "socket.io-client": "^4.7.4",
  "web-vitals": "^2.1.4"
},
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"eslintConfig": {
  "extends": [
    "react-app",
    "react-app/jest"
  ]
},
"browserslist": {
  "production": [
  ">0.2%",
```

```
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
 }
}
```

Route.js

```
import express from 'express';

import { addUser, getUsers } from '../controller/user-controller.js';
import { newConversation , getConversation} from '../controller/conversation-controller.js';
import { newMessage , getMessages } from '../controller/message-controller.js';
import { uploadFile , getImage } from '../controller/image-controller.js';

import upload from '../utils/upload.js';


const route = express.Router();

route.post('/add',addUser);
route.get('/users',getUsers);

route.post('/conversation/add' , newConversation);
route.post('/conversation/get',getConversation);

route.post('/message/add',newMessage);
```

```
route.get('/message/get/:id',getMessages);

route.post('/file/upload/',upload.single("file"), uploadFile);
route.get('/file/:filename' , getImage);

export default route;
```

upload.js

```
import multer from 'multer';
import {GridFsStorage} from 'multer-gridfs-storage';
import dotenv from 'dotenv';

dotenv.config();

const USERNAME = process.env.DB_USERNAME;
const PASSWORD = process.env.DB_PASSWORD;

const storage = new GridFsStorage({
                        url:`mongodb://${USERNAME}:${PASSWORD}@ac-fw9kttz-shard-00-00.egvsrje.mongodb.net:27017,ac-fw9kttz-shard-00-01.egvsrje.mongodb.net:27017,ac-fw9kttz-shard-00-02.egvsrje.mongodb.net:27017/?ssl=true&replicaSet=atlas-hxalcn-shard-0&authSource=admin&retryWrites=true&w=majority`,
    options: {useUnifiedTopology: true , useNewUrlParser: true},
    file: (request , file) => {
        const match = ["image/png" , "image/jpg"];

        if(match.indexOf(file.mimeType) === -1){
            return `${Date.now()}-file-${file.originalname}`;
        }

        return {
```

```
        bucketName:"photos",
        filename: `${Date.now()}-file-${file.originalname}`
      }
    }
});

export default multer({storage});
```

user.js

```
import mongoose from "mongoose";

const userSchema = mongoose.Schema({
    iss:{
        type: String
    },
    nbf:{
        type: Number
    },
    aud:{
        type: String
    },
    sub:{
        type: String,
        required: true
    },
    email:{
        type: String
    },
    email_verified:{
        type: Boolean
    },
```

```
    azp:{
        type: String
    },
    name:{
        type: String,
        required: true
    },
    picture:{
        type: String,
        required: true
    },
    given_name:{
        type: String
    },
    family_name:{
        type: String
    },
    iat:{
        type: Number
    },
    exp:{
        type: Number
    },
    jti:{
        type: String
    }

})

const user = mongoose.model('user', userSchema);

export default user;
```

message.js

```
import mongoose from "mongoose";

const MessageSchema = new mongoose.Schema({
    conversationId:{
        type: String
    },
    senderId:{
        type: String
    },
    receiverId:{
        tpye: String
    },
    text:{
        type: String
    },
    type: {
        type: String
    }
},
{
    timestamps: true
});

const message = mongoose.model('Message',MessageSchema);

export default message;
```

Socket - index.js

```
import { Server } from "socket.io";

const io = new Server(9000 , {
    cors: {
        origin: 'http://localhost:3000'
    },
})

let users = [];

const addUser = (userData , socketId) => {
    !users.some(user => user.sub === userData.sub) && users.push({...userData , socketId});
}
const removeUser = (socketId) => {
    users = users.filter(user => user.socketId !== socketId);
}

const getUser = (userId) => {
    return users.find(user => user.sub === userId);
}

io.on('connection' , (socket) => {
    console.log('User connected');

    socket.on("addUser" , userData => {
        addUser(userData , socket.id)
        io.emit("getUsers" , users);

    })

    socket.on('sendMessage' , (data) => {
        const user = getUser(data.receiverId);
```

```
        io.to(user.socketId).emit('getMessage' , data);
    })

    socket.on('disconnect', () => {
        console.log('user disconnected');
        removeUser(socket.id);
        io.emit('getUsers', users);
    })
})
```

Socket – packet-lock.json

```
{
  "name": "socket",
  "version": "1.0.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "socket",
      "version": "1.0.0",
      "license": "ISC",
      "dependencies": {
      "nodemon": "^3.0.3",
        "socket.io": "^4.7.4",
        "socket.io-client": "^4.7.4"
      }
    },
    "node_modules/@socket.io/component-emitter": {
      "version": "3.1.0",
            "resolved":    "https://registry.npmjs.org/@socket.io/component-emitter/-/component-emitter-3.1.0.tgz",
```

"integrity": "sha512-+9jVqKhRSpsc591z5vX+X5Yyw+he/HCB4iQ/RYxw35CEPaY1gnsNE43nf9n9AaYjAQrTiI/mOwKUKdUs9vf7Xg=="

},

"node_modules/@types/cookie": {

"version": "0.4.1",

"resolved": "https://registry.npmjs.org/@types/cookie/-/cookie-0.4.1.tgz",

"integrity": "sha512-XW/Aa8APYr6jSVVA1y/DEIZX0/GMKLEVekNG727R8cs56ahETkRAy/3DR7+fJyh7oUgGwNQaRfXCun0+KbWY7Q=="

},

"node_modules/@types/cors": {

"version": "2.8.17",

"resolved": "https://registry.npmjs.org/@types/cors/-/cors-2.8.17.tgz",

"integrity": "sha512-8CGDvrBj1zgo2qE+oS3pOCyYNqCPryMWY2bGfwA0dcfopWGgxs+78df0Rs3rc9THP4JkOhLsAa+15VdpAqkcUA==",

"dependencies": {

"@types/node": "*"

}

},

"node_modules/@types/node": {

"version": "20.11.4",

"resolved": "https://registry.npmjs.org/@types/node/-/node-20.11.4.tgz",

"integrity": "sha512-6I0fMH8Aoy2lOejL3s4LhyIYX34DPwY8bl5xlNjBvUEk8OHrcuzsFt+Ied4LvJihbtXPM+8zUqdydfIti86v9g==",

"dependencies": {

"undici-types": "~5.26.4"

}

},

"node_modules/abbrev": {

"version": "1.1.1",

"resolved": "https://registry.npmjs.org/abbrev/-/abbrev-1.1.1.tgz",

"integrity": "sha512-nne9/IiQ/hzIhY6pdDnbBtz7DjPTKrY00P/zvPSm5pOFkl6xuGrGnXn/VtTNNfNtAfZ9/1RtehkszU9qcTii0Q=="

},

"node_modules/accepts": {

"version": "1.3.8",

"resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.8.tgz",

"integrity": "sha512-PYAthTa2m2VKxuvSD3DPC/Gy+U+sOA1LAuT8mkmRuvw+NACSaeXEQ+NHcVF7rONl6qcaxV3Uuemwawk+7+SJLw==",

"dependencies": {

"mime-types": "~2.1.34",

"negotiator": "0.6.3"

},

"engines": {

"node": ">= 0.6"

}

},

"node_modules/anymatch": {

"version": "3.1.3",

"resolved": "https://registry.npmjs.org/anymatch/-/anymatch-3.1.3.tgz",

"integrity": "sha512-KMReFUr0B4t+D+OBkjR3KYqvocp2XaSzO55UcB6mgQMd3KbcE+mWTyvVV7D/zsdEbNnV6acZUutkiHQXvTr1Rw==",

"dependencies": {

"normalize-path": "^3.0.0",

"picomatch": "^2.0.4"

},

"engines": {

"node": ">= 8"

}

},

"node_modules/balanced-match": {

"version": "1.0.2",

```
    "resolved": "https://registry.npmjs.org/balanced-match/-/balanced-match-1.0.2.tgz",
    "integrity": "sha512-3oSeUO0TMV67hN1AmbXsK4yaqU7tjiHlbxRDZOpH0KW9+CeX4bRAaX0Anxt0tx2MrpRpWwQaPwIlISEJhYU5Pw=="
  },
  "node_modules/base64id": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/base64id/-/base64id-2.0.0.tgz",
    "integrity": "sha512-lGe34o6EHj9y3Kts9R4ZYs/Gr+6N7MCaMlIFA3F1R2O5/m7K06AxfSeO5530PEERE6/WyEg3lsuyw4GHlPZHog==",
    "engines": {
      "node": "^4.5.0 || >= 5.9"
    }
  },
  "node_modules/binary-extensions": {
    "version": "2.2.0",
    "resolved": "https://registry.npmjs.org/binary-extensions/-/binary-extensions-2.2.0.tgz",
    "integrity": "sha512-jDctJ/IVQbZoJykoeHbhXpOlNBqGNcwXJKJog42E5HDPUwQTSdjCHdihjj0DlnheQ7blbT6dHOafNAiS8ooQKA==",
    "engines": {
      "node": ">=8"
    }
  },
  "node_modules/brace-expansion": {
    "version": "1.1.11",
    "resolved": "https://registry.npmjs.org/brace-expansion/-/brace-expansion-1.1.11.tgz",
    "integrity": "sha512-iCuPHDFgrHX7H2vEI/5xpz07zSHB00TpugqhmYtVmMO6518mCuRMoOYFldEBl0g187ufozdaHgWKcYFb61qGiA==",
    "dependencies": {
      "balanced-match": "^1.0.0",
      "concat-map": "0.0.1"
    }
```

```
  },
  "node_modules/braces": {
    "version": "3.0.2",
    "resolved": "https://registry.npmjs.org/braces/-/braces-3.0.2.tgz",
    "integrity": "sha512-b8um+L1RzM3WDSzvhm6gIz1yfTbBt6YTlcEKAvsmqCZZFw46z626lVj9j1yEPW33H5H+lBQpZMP1k8l+78Ha0A==",
    "dependencies": {
      "fill-range": "^7.0.1"
    },
    "engines": {
      "node": ">=8"
    }
  },
  "node_modules/chokidar": {
    "version": "3.5.3",
    "resolved": "https://registry.npmjs.org/chokidar/-/chokidar-3.5.3.tgz",
    "integrity": "sha512-Dr3sfKRP6oTcjf2JmUmFJfeVMvXBdegxB0iVQ5eb2V10uFJUCAS8OByZdVAyVb8xXNz3GjjTgj9kLWsZTqE6kw==",
    "funding": [
      {
        "type": "individual",
        "url": "https://paulmillr.com/funding/"
      }
    ],
    "dependencies": {
      "anymatch": "~3.1.2",
      "braces": "~3.0.2",
      "glob-parent": "~5.1.2",
      "is-binary-path": "~2.1.0",
      "is-glob": "~4.0.1",
      "normalize-path": "~3.0.0",
```

```
      "readdirp": "~3.6.0"
    },
    "engines": {
      "node": ">= 8.10.0"
    },
    "optionalDependencies": {
      "fsevents": "~2.3.2"
    }
  },
  "node_modules/concat-map": {
    "version": "0.0.1",
    "resolved": "https://registry.npmjs.org/concat-map/-/concat-map-0.0.1.tgz",
    "integrity": "sha512-/Srv4dswyQNBfohGpz9o6Yb3Gz3SrUDqBH5rTuhGR7ahtlbYKnVxw2bCFMRljaA7EXHaXZ8wsHdodFvbkhKmqg=="
  },
  "node_modules/cookie": {
    "version": "0.4.2",
    "resolved": "https://registry.npmjs.org/cookie/-/cookie-0.4.2.tgz",
    "integrity": "sha512-aSWTXFzaKWkvHO1Ny/s+ePFpvKsPnjc551iI41v3ny/ow6tBG5Vd+FuqGNhh1LxOmVzOlGUriIlOaokOvhaStA==",
    "engines": {
      "node": ">= 0.6"
    }
  },
  "node_modules/cors": {
    "version": "2.8.5",
    "resolved": "https://registry.npmjs.org/cors/-/cors-2.8.5.tgz",
    "integrity": "sha512-KIHbLJqu73RGr/hnbrO9uBeixNGuvSQjul/jdFvS/KFSIH1hWVd1ng7zOHx+YrEfInLG7q4n6GHQ9cDtxv/P6g==",
    "dependencies": {
      "object-assign": "^4",
```

```
      "vary": "^1"
    },
    "engines": {
      "node": ">= 0.10"
    }
  },
  "node_modules/debug": {
    "version": "4.3.4",
    "resolved": "https://registry.npmjs.org/debug/-/debug-4.3.4.tgz",
    "integrity": "sha512-PRWFHuSU3eDtQJPvnNY7Jcket1j0t5OuOsFzPPzsekD52Zl8qUfFIPEiswXqIvHWGVHOgX+7G/vCNNhehwxfkQ==",
    "dependencies": {
      "ms": "2.1.2"
    },
    "engines": {
      "node": ">=6.0"
    },
    "peerDependenciesMeta": {
      "supports-color": {
        "optional": true
      }
    }
  },
  "node_modules/engine.io": {
    "version": "6.5.4",
    "resolved": "https://registry.npmjs.org/engine.io/-/engine.io-6.5.4.tgz",
    "integrity": "sha512-KdVSDKhVKyOi+r5uEabrDLZw2qXStVvCsEB/LN3mw4WFi6Gx50jTyuxYVCwAAC0U46FdnzP/ScKRBTXb/NiEOg==",
    "dependencies": {
      "@types/cookie": "^0.4.1",
      "@types/cors": "^2.8.12",
```

```
      "@types/node": ">=10.0.0",
      "accepts": "~1.3.4",
      "base64id": "2.0.0",
      "cookie": "~0.4.1",
      "cors": "~2.8.5",
      "debug": "~4.3.1",
      "engine.io-parser": "~5.2.1",
      "ws": "~8.11.0"
    },
    "engines": {
      "node": ">=10.2.0"
    }
  },
  "node_modules/engine.io-client": {
    "version": "6.5.3",
    "resolved": "https://registry.npmjs.org/engine.io-client/-/engine.io-client-6.5.3.tgz",
    "integrity": "sha512-
9Z0qLB0NIisTRt1DZ/8U2k12RJn8yls/nXMZLn+/N8hANT3TcYjKFKcwbw5zFQiN4NTde3
TSY9zb79e1ij6j9Q==",
    "dependencies": {
      "@socket.io/component-emitter": "~3.1.0",
      "debug": "~4.3.1",
      "engine.io-parser": "~5.2.1",
      "ws": "~8.11.0",
      "xmlhttprequest-ssl": "~2.0.0"
    }
  },
  "node_modules/engine.io-parser": {
    "version": "5.2.1",
    "resolved": "https://registry.npmjs.org/engine.io-parser/-/engine.io-parser-5.2.1.tgz",
    "integrity": "sha512-
9JktcM3u18nU9N2Lz3bWeBgxVgOKpw7yhRaoxQA3FUDZzzw+9WlA6p4G4u0RixNkg14f
H7EfEc/RhpurtiROTQ==",
```

```
    "engines": {
      "node": ">=10.0.0"
    }
  },
  "node_modules/fill-range": {
    "version": "7.0.1",
    "resolved": "https://registry.npmjs.org/fill-range/-/fill-range-7.0.1.tgz",
    "integrity": "sha512-qOo9F+dMUmC2Lcb4BbVvnKJxTPjCm+RRpe4gDuGrzkL7mEVl/djYSu2OdQ2Pa302N4oqkSg9ir6jaLWJ2USVpQ==",
    "dependencies": {
      "to-regex-range": "^5.0.1"
    },
    "engines": {
      "node": ">=8"
    }
  },
  "node_modules/fsevents": {
    "version": "2.3.3",
    "resolved": "https://registry.npmjs.org/fsevents/-/fsevents-2.3.3.tgz",
    "integrity": "sha512-5xoDfX+fL7faATnagmWPpbFtwh/R77WmMMqqHGS65C3vvB0YHrgF+B1YmZ3441tMj5n63k0212XNoJwzlhffQw==",
    "hasInstallScript": true,
    "optional": true,
    "os": [
      "darwin"
    ],
    "engines": {
      "node": "^8.16.0 || ^10.6.0 || >=11.0.0"
    }
  },
  "node_modules/glob-parent": {
```

```
    "version": "5.1.2",
    "resolved": "https://registry.npmjs.org/glob-parent/-/glob-parent-5.1.2.tgz",
    "integrity": "sha512-AOIgSQCepiJYwP3ARnGx+5VnTu2HBYdzbGP45eLw1vr3zB3vZLeyed1sC9hnbcOc9/SrMyM5RPQrkGz4aS9Zow==",
    "dependencies": {
      "is-glob": "^4.0.1"
    },
    "engines": {
      "node": ">= 6"
    }
  },
  "node_modules/has-flag": {
    "version": "3.0.0",
    "resolved": "https://registry.npmjs.org/has-flag/-/has-flag-3.0.0.tgz",
    "integrity": "sha512-sKJf1+ceQBr4SMkvQnBDNDtf4TXpVhVGateu0t918bl30FnbE2m4vNLX+VWe/dpjlb+HugGYzW7uQXH98HPEYw==",
    "engines": {
      "node": ">=4"
    }
  },
  "node_modules/ignore-by-default": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/ignore-by-default/-/ignore-by-default-1.0.1.tgz",
    "integrity": "sha512-Ius2VYcGNk7T90CppJqcIkS5ooHUZyIQK+ClZfMfMNFEF9VSE73Fq+906u/CWu92x4gzZMWOwfFYckPObzdEbA=="
  },
  "node_modules/is-binary-path": {
    "version": "2.1.0",
    "resolved": "https://registry.npmjs.org/is-binary-path/-/is-binary-path-2.1.0.tgz",
    "integrity": "sha512-ZMERYes6pDydyuGidse7OsHxtbI7WVeUEozgR/g7rd0xUimYNlvZRE/K2MgZTjWy725IfelLeVcEM97mmtRGXw==",
```

```
    "dependencies": {
      "binary-extensions": "^2.0.0"
    },
    "engines": {
      "node": ">=8"
    }
  },
  "node_modules/is-extglob": {
    "version": "2.1.1",
    "resolved": "https://registry.npmjs.org/is-extglob/-/is-extglob-2.1.1.tgz",
    "integrity": "sha512-SbKbANkN603Vi4jEZv49LeVJMn4yGwsbzZworEoyEiutsN3nJYdbO36zfhGJ6QEDpOZIFkDtnq5JRxmvl3jsoQ==",
    "engines": {
      "node": ">=0.10.0"
    }
  },
  "node_modules/is-glob": {
    "version": "4.0.3",
    "resolved": "https://registry.npmjs.org/is-glob/-/is-glob-4.0.3.tgz",
    "integrity": "sha512-xelSayHH36ZgE7ZWhli7pW34hNbNl8Ojv5KVmkJD4hBdD3th8Tfk9vYasLM+mXWOZhFkgZfxhLSnrwRr4elSSg==",
    "dependencies": {
      "is-extglob": "^2.1.1"
    },
    "engines": {
      "node": ">=0.10.0"
    }
  },
  "node_modules/is-number": {
    "version": "7.0.0",
    "resolved": "https://registry.npmjs.org/is-number/-/is-number-7.0.0.tgz",
```

"integrity": "sha512-41Cifkg6e8TylSpdtTpeLVMqvSBEVzTttHvERD741+pnZ8ANv0004MRL43QKPDlK9cGvNp6NZWZUBlbGXYxxng==",
"engines": {
"node": ">=0.12.0"
}
},
"node_modules/lru-cache": {
"version": "6.0.0",
"resolved": "https://registry.npmjs.org/lru-cache/-/lru-cache-6.0.0.tgz",
"integrity": "sha512-Jo6dJ04CmSjuznwJSS3pUeWmd/H0ffTlkXXgwZi+eq1UCmqQwCh+eLsYOYCwY991i2Fah4h1BEMCx4qThGbsiA==",
"dependencies": {
"yallist": "^4.0.0"
},
"engines": {
"node": ">=10"
}
},
"node_modules/mime-db": {
"version": "1.52.0",
"resolved": "https://registry.npmjs.org/mime-db/-/mime-db-1.52.0.tgz",
"integrity": "sha512-sPU4uV7dYlvtWJxwwxHD0PuihVNiE7TyAbQ5SWxDCB9mUYvOgroQOwYQQOKPJ8CIbE+1ETVlOoK1UC2nU3gYvg==",
"engines": {
"node": ">= 0.6"
}
},
"node_modules/mime-types": {
"version": "2.1.35",
"resolved": "https://registry.npmjs.org/mime-types/-/mime-types-2.1.35.tgz",

"integrity": "sha512-ZDY+bPm5zTTF+YpCrAU9nK0UgICYPT0QtT1NZWFv4s++TNkcgVaT0g6+4R2uI4MjQjzysHB1zxuWL50hzaeXiw==",
"dependencies": {
"mime-db": "1.52.0"
},
"engines": {
"node": ">= 0.6"
}
},
"node_modules/minimatch": {
"version": "3.1.2",
"resolved": "https://registry.npmjs.org/minimatch/-/minimatch-3.1.2.tgz",
"integrity": "sha512-J7p63hRiAjw1NDEww1W7i37+ByIrOWO5XQQAzZ3VOcL0PNybwpfmV/N05zFAzwQ9USyEcX6t3UO+K5aqBQOIHw==",
"dependencies": {
"brace-expansion": "^1.1.7"
},
"engines": {
"node": "*"
}
},
"node_modules/ms": {
"version": "2.1.2",
"resolved": "https://registry.npmjs.org/ms/-/ms-2.1.2.tgz",
"integrity": "sha512-sGkPx+VjMtmA6MX27oA4FBFELFCZZ4S4XqeGOXCv68tT+jb3vk/RyaKWP0PTKyWtmLSM0b+adUTEvbs1PEaH2w=="
},
"node_modules/negotiator": {
"version": "0.6.3",
"resolved": "https://registry.npmjs.org/negotiator/-/negotiator-0.6.3.tgz",

"integrity": "sha512-+EUsqGPLsM+j/zdChZjsnX51g4XrHFOIXwfnCVPGlQk/k5giakcKsuxCObBRu6DSm9opw/O6slWbJdghQM4bBg==",

"engines": {

"node": ">= 0.6"

}

},

"node_modules/nodemon": {

"version": "3.0.3",

"resolved": "https://registry.npmjs.org/nodemon/-/nodemon-3.0.3.tgz",

"integrity": "sha512-7jH/NXbFPxVaMwmBCC2B9F/V6X1VkEdNgx3iu9jji8WxWcvhMWkmhNWhI5077zknOnZnBzba9hZP6bCPJLSReQ==",

"dependencies": {

"chokidar": "^3.5.2",

"debug": "^4",

"ignore-by-default": "^1.0.1",

"minimatch": "^3.1.2",

"pstree.remy": "^1.1.8",

"semver": "^7.5.3",

"simple-update-notifier": "^2.0.0",

"supports-color": "^5.5.0",

"touch": "^3.1.0",

"undefsafe": "^2.0.5"

},

"bin": {

"nodemon": "bin/nodemon.js"

},

"engines": {

"node": ">=10"

},

"funding": {

"type": "opencollective",

```
      "url": "https://opencollective.com/nodemon"
    }
  },
  "node_modules/nopt": {
    "version": "1.0.10",
    "resolved": "https://registry.npmjs.org/nopt/-/nopt-1.0.10.tgz",
    "integrity": "sha512-NWmpvLSqUrgrAC9HCuxEvb+PSloHpqVu+FqcO4eeF2h5qYRhA7ev6KvelyQAKtegUbC6RypJnlEOhd8vloNKYg==",
    "dependencies": {
      "abbrev": "1"
    },
    "bin": {
      "nopt": "bin/nopt.js"
    },
    "engines": {
      "node": "*"
    }
  },
  "node_modules/normalize-path": {
    "version": "3.0.0",
    "resolved": "https://registry.npmjs.org/normalize-path/-/normalize-path-3.0.0.tgz",
    "integrity": "sha512-6eZs5Ls3WtCisHWp9S2GUy8dqkpGi4BVSz3GaqiE6ezub0512ESztXUwUB6C6IKbQkY2Pnb/mD4WYojCRwcwLA==",
    "engines": {
      "node": ">=0.10.0"
    }
  },
  "node_modules/object-assign": {
    "version": "4.1.1",
    "resolved": "https://registry.npmjs.org/object-assign/-/object-assign-4.1.1.tgz",
```

"integrity": "sha512-rJgTQnkUnH1sFw8yT6VSU3zD3sWmu6sZhIseY8VX+GRu3P6F7Fu+JNDoXfklElbLJSnc3FUQHVe4cU5hj+BcUg==",

"engines": {

"node": ">=0.10.0"

}

},

"node_modules/picomatch": {

"version": "2.3.1",

"resolved": "https://registry.npmjs.org/picomatch/-/picomatch-2.3.1.tgz",

"integrity": "sha512-JU3teHTNjmE2VCGFzuY8EXzCDVwEqB2a8fsIvwaStHhAWJEeVd1o1QD80CU6+ZdEXXSLbSsuLwJjkCBWqRQUVA==",

"engines": {

"node": ">=8.6"

},

"funding": {

"url": "https://github.com/sponsors/jonschlinkert"

}

},

"node_modules/pstree.remy": {

"version": "1.1.8",

"resolved": "https://registry.npmjs.org/pstree.remy/-/pstree.remy-1.1.8.tgz",

"integrity": "sha512-77DZwxQmxKnu3aR542U+X8FypNzbfJ+C5XQDk3uWjWxn6151aIMGthWYRXTqT1E5oJvg+ljaa2OJi+VfvCOQ8w=="

},

"node_modules/readdirp": {

"version": "3.6.0",

"resolved": "https://registry.npmjs.org/readdirp/-/readdirp-3.6.0.tgz",

"integrity": "sha512-hOS089on8RduqdbhvQ5Z37A0ESjsqz6qnRcffsMU3495FuTdqSm+7bhJ29JvIOsBDEEnan5DPu9t3To9VRlMzA==",

"dependencies": {

"picomatch": "^2.2.1"

```
    },
    "engines": {
      "node": ">=8.10.0"
    }
  },
  "node_modules/semver": {
    "version": "7.5.4",
    "resolved": "https://registry.npmjs.org/semver/-/semver-7.5.4.tgz",
    "integrity": "sha512-1bCSESV6Pv+i21Hvpxp3Dx+pSD8lIPt8uVjRrxAUt/nbswYc+tK6Y2btiULjd4+fnq15PX+nqQDC7Oft7WkwcA==",
    "dependencies": {
      "lru-cache": "^6.0.0"
    },
    "bin": {
      "semver": "bin/semver.js"
    },
    "engines": {
      "node": ">=10"
    }
  },
  "node_modules/simple-update-notifier": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/simple-update-notifier/-/simple-update-notifier-2.0.0.tgz",
    "integrity": "sha512-a2B9Y0KlNXl9u/vsW6sTIu9vGEpfKu2wRV6l1H3XEas/0gUIzGzBoP/IouTcUQbm9JWZLH3COxyn03TYlFax6w==",
    "dependencies": {
      "semver": "^7.5.3"
    },
    "engines": {
      "node": ">=10"
    }
```

```
  },
  "node_modules/socket.io": {
    "version": "4.7.4",
    "resolved": "https://registry.npmjs.org/socket.io/-/socket.io-4.7.4.tgz",
    "integrity": "sha512-DcotgfP1Zg9iP/dH9zvAQcWrE0TtbMVwXmlV4T4mqsvY+gw+LqUGPfx2AoVyRk0FLME+GQhufDMyacFmw7ksqw==",
    "dependencies": {
    "accepts": "~1.3.4",
      "base64id": "~2.0.0",
      "cors": "~2.8.5",
      "debug": "~4.3.2",
      "engine.io": "~6.5.2",
      "socket.io-adapter": "~2.5.2",
      "socket.io-parser": "~4.2.4"
    },
    "engines": {
      "node": ">=10.2.0"
    }
  },
  "node_modules/socket.io-adapter": {
    "version": "2.5.2",
    "resolved": "https://registry.npmjs.org/socket.io-adapter/-/socket.io-adapter-2.5.2.tgz",
    "integrity": "sha512-87C3LO/NOMc+eMcpcxUBebGjkpMDkNBS9tf7KJqcDsmL936EChtVva71Dw2q4tQcuVC+hAUy4an2NO/sYXmwRA==",
    "dependencies": {
      "ws": "~8.11.0"
    }
  },
  "node_modules/socket.io-client": {
    "version": "4.7.4",
    "resolved": "https://registry.npmjs.org/socket.io-client/-/socket.io-client-4.7.4.tgz",
```

"integrity": "sha512-wh+OkeF0rAVCrABWQBaEjLfb7DVPotMbu0cgWgyR0v6eA4EoVnAwcIeIbcdTE3GT/H3kbdLl7OoH2+asoDRIIg==",
"dependencies": {
"@socket.io/component-emitter": "~3.1.0",
"debug": "~4.3.2",
"engine.io-client": "~6.5.2",
"socket.io-parser": "~4.2.4"
},
"engines": {
"node": ">=10.0.0"
}
},
"node_modules/socket.io-parser": {
"version": "4.2.4",
"resolved": "https://registry.npmjs.org/socket.io-parser/-/socket.io-parser-4.2.4.tgz",
"integrity": "sha512-/GbIKmo8ioc+NIWIhwdecY0ge+qVBSMdgxGygevmdHj24bsfgtCmcUUcQ5ZzcylGFHsN3k4HB4Cgkl96KVnuew==",
"dependencies": {
"@socket.io/component-emitter": "~3.1.0",
"debug": "~4.3.1"
},
"engines": {
"node": ">=10.0.0"
}
},
"node_modules/supports-color": {
"version": "5.5.0",
"resolved": "https://registry.npmjs.org/supports-color/-/supports-color-5.5.0.tgz",
"integrity": "sha512-QjVjwdXIt408MIiAqCX4oUKsgU2EqAGzs2Ppkm4aQYbjm+ZEWEcW4SfFNTr4uMNZma0ey4f5lgLrkB0aX0QMow==",
"dependencies": {

```
      "has-flag": "^3.0.0"
    },
    "engines": {
      "node": ">=4"
    }
  },
  "node_modules/to-regex-range": {
    "version": "5.0.1",
    "resolved": "https://registry.npmjs.org/to-regex-range/-/to-regex-range-5.0.1.tgz",
    "integrity": "sha512-65P7iz6X5yEr1cwcgvQxbbIw7Uk3gOy5dIdtZ4rDveLqhrdJP+Li/Hx6tyK0NEb+2GCyneCMJiGqrADCSNk8sQ==",
    "dependencies": {
      "is-number": "^7.0.0"
    },
    "engines": {
      "node": ">=8.0"
    }
  },
  "node_modules/touch": {
    "version": "3.1.0",
    "resolved": "https://registry.npmjs.org/touch/-/touch-3.1.0.tgz",
    "integrity": "sha512-WBx8Uy5TLtOSRtIq+M03/sKDrXCLHxwDcquSP2c43Le03/9serjQBIztjRz6FkJez9D/hleyAXTBGLwwZUw9lA==",
    "dependencies": {
      "nopt": "~1.0.10"
    },
    "bin": {
      "nodetouch": "bin/nodetouch.js"
    }
  },
  "node_modules/undefsafe": {
```

```
    "version": "2.0.5",
    "resolved": "https://registry.npmjs.org/undefsafe/-/undefsafe-2.0.5.tgz",
    "integrity": "sha512-WxONCrssBM8TSPRqN5EmsjVrsv4A8X12J4ArBiiayv3DyyG3ZlIg6yysuuSYdZsVz3TKcTg2fd//Ujd4CHV1iA=="
  },
  "node_modules/undici-types": {
    "version": "5.26.5",
    "resolved": "https://registry.npmjs.org/undici-types/-/undici-types-5.26.5.tgz",
    "integrity": "sha512-JlCMO+ehdEIKqlFxk6IfVoAUVmgz7cU7zD/h9XZ0qzeosSHmUJVOzSQvvYSYWXkFXC+IfLKSIffhv0sVZup6pA=="
  },
  "node_modules/vary": {
    "version": "1.1.2",
    "resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",
    "integrity": "sha512-BNGbWLfd0eUPabhkXUVm0j8uuvREyTh5ovRa/dyow/BqAbZJyC+5fU+IzQOzmAKzYqYRAISoRhdQr3eIZ/PXqg==",
    "engines": {
      "node": ">= 0.8"
    }
  },
  "node_modules/ws": {
    "version": "8.11.0",
    "resolved": "https://registry.npmjs.org/ws/-/ws-8.11.0.tgz",
    "integrity": "sha512-HPG3wQd9sNQoT9xHyNCXoDUa+Xw/VevmY9FoHyQ+g+rrMn4j6FB4np7Z0OhdTgjx6MgQLK7jwSy1YecU1+4Asg==",
    "engines": {
      "node": ">=10.0.0"
    },
    "peerDependencies": {
      "bufferutil": "^4.0.1",
      "utf-8-validate": "^5.0.2"
```

```
    },
    "peerDependenciesMeta": {
      "bufferutil": {
        "optional": true
      },
      "utf-8-validate": {
        "optional": true
      }
    }
  },
  "node_modules/xmlhttprequest-ssl": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/xmlhttprequest-ssl/-/xmlhttprequest-ssl-2.0.0.tgz",
    "integrity": "sha512-QKxVRxiRACQcVuQEYFsI1hhkrMlrXHPegbbd1yn9UHOmRxY+si12nQYzri3vbzt8VdTTRviqcKxcyllFas5z2A==",
    "engines": {
      "node": ">=0.4.0"
    }
  },
  "node_modules/yallist": {
    "version": "4.0.0",
    "resolved": "https://registry.npmjs.org/yallist/-/yallist-4.0.0.tgz",
    "integrity": "sha512-3wdGidZyq5PB084XLES5TpOSRA3wjXAlIWMhum2kRcv/41Sn2emQ0dycQW4uZXLejwKvg6EsvbdlVL+FYEct7A=="
  }
 }
}
```


LoginDialog.jsx

```
import { useContext } from "react";
```

```
import { Dialog , Box, Typography , List, ListItem , styled} from "@mui/material";
import {GoogleLogin} from '@react-oauth/google';
import jwt_decode from 'jwt-decode';
import { AccountContext } from "../../context/AccountProvider";
import { addUser } from "../../service/api";

const Component = styled(Box)`
    display: flex;
`;

const Container =styled(Box)`
    padding: 56px 0 56px 56px;
`;

const Title = styled(Typography)`
    font-size:26px;
    margin-bottom:25px;
    color: #525252;
     font-family:Segoe UI,Helvetica Neue,Helvetica,Lucida Grande,Arial,Ubuntu,Cantarell,Fira Sans,sans-serif;
    font-weight:300;
`;

const StyledList = styled(List)`
    & > li  {
        padding: 0;
        margin-top:15px;
        font-size:18px;
        line-height:28px;
        color: #4a4a4a;
    }
`;
```

```
const dialogStyle ={
    marginTop: '12%',
    height: '96%',
    width: '60%',
    maxWidth: '100%',
    maxHeight:'100%',
    borderRadius: 0 ,
    boxShadow:'none',
    overFlow:'hidden'

}

const LoginDialog = () => {

    const{setAccount} = useContext(AccountContext);

    const onLoginError = (res) => {

        console.log('Login failed' , res);
    }

    const onLoginSuccess = async (res) => {

      const decoded = jwt_decode(res.credential);
      setAccount(decoded);
      await addUser(decoded);
    }

    return(
        <Dialog open = {true}
```

```
        BackdropProps={{style: {backgroundColor: 'unset'}}}
        maxWidth={'md'}
        PaperProps={{sx: dialogStyle}}
        // hideBackdrop={true}
            >
                <Component>
                    <Container>
                        <Title>To use Whatsapp on your computer:</Title>
                        <StyledList>
                            <ListItem>1. Open Whatsapp on your phone</ListItem>
                            <ListItem>2. Tap Menu or Settings and select Linked Devices</ListItem>
                            <ListItem>3. Point your phone to this screen to capture the code</ListItem>
                        </StyledList>
                    </Container>
                    <Box style={{position: 'relative'}}>
                            <Box style={{position: 'absolute' , top:'50%' , transform:'translateX(25%) translateY(-25%)' }}>
                        <GoogleLogin
                            buttonText = ""
                            onSuccess={onLoginSuccess}
                            onError={onLoginError}
                        />
                        </Box>
                    </Box>
                </Component>

        </Dialog>
    )
}

export default LoginDialog;
```

CHAPTER 8

TESTING

Testing is an integral part of the software development lifecycle, ensuring the reliability and functionality of the WhatsApp Web Clone project. The testing process includes three essential modules: Unit Testing, Integration Testing, and System Testing.

8.1 Unit Testing

Objective: Validate the functionality of individual components or units in isolation to ensure they perform as intended.
Testing Scope: Focuses on verifying the correctness of functions, methods, and modules at the smallest level.
Tools/Methodologies:

- **Testing Frameworks:** Jest for JavaScript-based components, Pytest for Python components.
- **Mocking Frameworks:** Sinon.js for JavaScript, unittest.mock for Python.
- **Code Coverage:** Istanbul for JavaScript, coverage.py for Python.

Key Activities:

1. Test each function or method independently to ensure they produce the expected output.
2. Isolate and mock external dependencies to create controlled testing environments.
3. Verify edge cases and handle potential error scenarios within individual units.
4. Achieve high code coverage to ensure most parts of the codebase are tested.

8.2 Integration Testing

Objective: Confirm that the interactions between different units or modules work as intended when combined.
Testing Scope: Focuses on the communication and data flow between various components to identify issues in their collaboration.
Tools/Methodologies:

- **Testing Frameworks:** Mocha-Chai for JavaScript, Pytest for Python.
- **API Testing:** Postman, Swagger for testing API endpoints.

- **Database Testing:** Utilize testing databases or frameworks like SQLite for isolation.

Key Activities:

1. Test the integration points where different modules interact, ensuring data exchange is accurate.
2. Confirm that APIs and endpoints communicate correctly and handle requests and responses appropriately.
3. Validate database interactions, ensuring data integrity and consistency during transactions.
4. Identify and rectify issues arising from the integration of various components.

8.3 System Testing

Objective: Validate the system as a whole, ensuring that all integrated components function seamlessly and meet the specified requirements.

Testing Scope: Involves testing the entire application, including user interfaces, databases, and external dependencies.

Tools/Methodologies:

- **End-to-End Testing:** Selenium for web applications, Cypress for JavaScript applications.
- **Performance Testing:** JMeter for load testing, Apache Bench for stress testing.
- **Security Testing:** OWASP ZAP for security scanning, penetration testing tools.

Key Activities:

1. Execute end-to-end tests to simulate user interactions and verify the system's overall functionality.
2. Conduct performance tests to assess how the system behaves under various loads and stress conditions.
3. Perform security testing to identify and address potential vulnerabilities in the system.
4. Validate that the system complies with functional requirements and delivers a seamless user experience.

The combination of Unit Testing, Integration Testing, and System Testing forms a comprehensive testing strategy for the WhatsApp Web Clone project. By systematically validating individual units, their interactions, and the system as a whole, the testing modules ensure the development of a robust and reliable web-based messaging platform. These testing efforts contribute to the overall quality and performance of the application, providing a positive user experience and mitigating potential issues before deployment.

CHAPTER 9

CONCLUSION

In conclusion, the development and implementation of the WhatsApp Web Clone project have been a comprehensive undertaking, aiming to replicate the core functionalities and user experience of the popular mobile messaging application on the web platform. Throughout the project lifecycle, various aspects, including requirement analysis, system design, hardware and software specifications, and testing modules, were meticulously addressed to ensure the creation of a robust and user-friendly web-based messaging platform.

The requirement analysis provided a clear roadmap, categorizing functional and non-functional aspects essential for the successful development of the project. The system design, exemplified by the Entity-Relationship diagram, detailed the database architecture, offering a structured foundation for data storage and retrieval.

Hardware and software specifications outlined the necessary components and technologies, ensuring optimal performance, security, and compatibility with modern web standards. The testing modules, including Unit Testing, Integration Testing, and System Testing, were crucial in identifying and rectifying issues throughout the development process, contributing to the reliability and functionality of the final product.

The project's success is contingent on adherence to these rigorous standards, ensuring a secure, scalable, and responsive web-based messaging platform. The development team's commitment to best practices, user-centric design, and thorough testing has resulted in a project that mirrors the seamless communication experience of the original WhatsApp mobile app.

As technology evolves, the WhatsApp Web Clone project remains adaptable, with a solid foundation for potential future enhancements and features. User feedback and continuous improvement will play pivotal roles in refining the platform and addressing emerging trends in web development and messaging applications.

In conclusion, the WhatsApp Web Clone project represents a successful endeavor in creating a user-friendly and feature-rich web-based messaging platform, providing a valuable communication tool for users across different devices. The project's journey, from conception to implementation, has been marked by attention to detail, adherence to best practices, and a commitment to delivering a high-quality user experience.

REFERENCES

The following are the websites that we had analysed for ours Whats App website clone projects:

- Amazon official website link: https://www.amazon.in/
- W3School
- React Js Official website link: https://react.dev/
- Mongo DB official website link: https://www.mongodb.com/