**A red and black logo

Description automatically generated**



Table of Contents

[Acknowledgment 3](#_Toc162031668)

[Abstract 4](#_Toc162031669)

[Introduction 5](#_Toc162031670)

[**Project Summary** 5](#_Toc162031671)

[**Objective** 6](#_Toc162031672)

[Project Description 8](#_Toc162031673)

[**Tools:** 8](#_Toc162031674)

[**Languages:** 9](#_Toc162031675)

[**How To Run the project?** 9](#_Toc162031676)

[Analysis and Design 10](#_Toc162031677)

[**ERD Diagram:** 11](#_Toc162031678)

[**Interface Design:** 12](#_Toc162031679)

[Implementation 16](#_Toc162031680)

[Future Work: 25](#_Toc162031681)

[Conclusion 25](#_Toc162031682)

# Acknowledgment

The success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

A special thank goes to our helpful supervisor, Eng: Mai Mostafa. And support she gave truly help in the progression of our graduation project. The co-operation is much indeed appreciated.

Our greatly appreciate to the contribution of Information Technology Institute,

Our grateful thanks also go to Eng. Yasser for his encouragement, patience, guidance, and

support when our project starts until we finish the project.

The project would be nothing without the enthusiasm, help and support from both of you. Besides, this project made us realized the value of working together as a team and as a new experience in working environment, which challenges us every minute.

# Abstract

In today's digital age, communication plays a vital role in connecting people across the globe. Real-time chat applications have become an integral part of our daily lives, enabling individuals to interact seamlessly and efficiently. This abstract outlines the design and implementation of a real-time chat application that facilitates instant messaging between users.

# Introduction

**GalaxyTalk** is a modern, real-time chat application built using Angular. It provides users with a seamless communication platform where they can engage in text-based conversations with friends, colleagues in real-time. With its intuitive user interface and robust features, GalaxyTalk aims to enhance the way people communicate and collaborate online.

**Project Summary:**

User Authentication: GalaxyTalk offers secure user authentication to ensure that only authorized users can access the chat platform. Users can sign up for an account, log in securely, and maintain their profile information.

Real-Time Messaging: Users can send and receive messages instantly, enabling fluid conversations without delays. The application leverages Angular's reactive capabilities and WebSockets to deliver messages in real-time.

Private Chats: GalaxyTalk supports both private, allowing users to have one-on-one conversations.

Responsive Design: GalaxyTalk is designed with responsiveness in mind, ensuring that the application adapts seamlessly to different screen sizes and devices. Whether users access the platform from a desktop, tablet, or smartphone, they can enjoy a consistent and optimized user experience.

## **Objective**

The objective of this project is to design, develop, and deploy a real-time chat application that facilitates seamless communication between users. The chat application aims to provide a user-friendly interface and robust features for instant messaging, and multimedia sharing. The primary objectives of the project include:

Seamless Communication: Create a platform that enables users to communicate in real-time, fostering instant and efficient communication across various devices and platforms.

1. User Authentication: Implement secure user authentication mechanisms to ensure the privacy and security of user data. Users should be able to sign up for new accounts and log in securely to access chat features.
2. Chat Features: Provide a comprehensive set of chat features, including one-on-one messaging, multimedia messaging and message search.
3. User Management: Enable users to manage their profiles, including updating personal information, changing profile pictures, and managing contact lists.
4. Scalability and Performance: Design the application to be scalable and performant, capable of handling a large number of concurrent users and messages without compromising performance.
5. Cross-Platform Compatibility: Ensure that the chat application is compatible with various devices and platforms, including web browser
6. User Experience: Prioritize user experience by designing an intuitive and user-friendly interface that simplifies navigation and encourages engagement.
7. Customization and Personalization: Allow users to customize their chat experience by choosing themes, setting notifications preferences, and organizing chats into categories.
8. Moderation and Security: Implement moderation tools and security measures to prevent spam, abuse, and unauthorized access to chats
9. Documentation and Support: Provide comprehensive documentation for developers, including API documentation, installation instructions, and usage guidelines. Offer responsive support channels to assist users and address any issues or inquiries promptly.

# Project Description

**Tools:**

**A logo of a company

Description automatically generated A blue ribbon with a cross

Description automatically generated**

Figma vs code

****

**Languages:**

|  |  |
| --- | --- |
| **Front-end:** | **Back-end :** |
| Angular  Tailwind  DaisyUi    CSS    Toastr | NodeJS  MondoDB  Express JS |

**How To Run the project?**

* Server Side
  + Install dependencies:

npm i cookie-parser mongoose bcryptjs validator dotenv cors jsonwebtoken express-async-handler

* + Start the server:

node index.js

* Client Side
  + Run ng e2e

to execute the end-to-end tests via a platform of your choice. To use this command, you need to first add a package that implements end-to-end testing capabilities.

* + Development server

ng serve for a dev server. Navigate to http://localhost:4200/. The application will automatically reload if you change any of the source files.

# Analysis and Design

1. Requirements Gathering:

* Define functional requirements (features, user roles, chat functionality) and non-functional requirements (performance, security, scalability)
* Prioritize requirements and establish a roadmap for development.

1. System Architecture:

* Define the high-level system architecture, including components, layers, and communication protocols.
* Choose appropriate technologies for frontend (Angular) and backend (Node.js, Express.js).
* Determine database requirements and select a suitable database system (MongoDB).
* Design the API architecture for communication between frontend and backend.

1. Database Design:

* Identify entities such as users, messages, chat rooms
* Design entity-relationship diagrams (ERDs) to visualize the relationships between entities.
* Normalize the database schema to minimize redundancy and ensure data integrity.

1. User Interface Design:

* Create wireframes to visualize the user interface (UI) design.
* Design intuitive UI components for features such as user authentication, chat windows, message input, and settings.
* Ensure consistency in UI elements, typography, colors, and layout across the application.
* Conduct usability testing to gather feedback and iterate on the design.

1. Authentication and Authorization:

* Implement user authentication using JWT (JSON Web Tokens)

1. Real-Time Communication:

* Set up WebSocket communication using libraries such as Socket.IO for real-time messaging.
* Handle WebSocket connections, messages, and events on the server-side and client-side.
* Implement features for sending and receiving real-time messages and updates.

**ERD Diagram:**

## **Interface Design:**

We started our design with

**1. Sign Up Page:**

It allows new users to create an account by providing necessary information like name, email, password, etc.

A screenshot of a computer

Description automatically generated

Figure 1 signup page.

2. Login Page:

A screenshot of a login page

Description automatically generatedIf user already has an account, he can log in easily using his Email and password

Figure 2Login Page

3. Welcome Chat Page :

A screenshot of a phone

Description automatically generatedWelcome serves as the initial screen users see when entering the chat after logging in . It also include a welcoming message

Figure 3 Welcome Chat Page

4. Chat Page :

It is designed to facilitate communication between users within the application.

It features chat bubbles or message threads where users can view and send text messages to each other.

A screenshot of a chat

Description automatically generated

Figure 4 Chat Page

Implementation:

**Frontend:**

Login Page:

The login page is a clean and modern user interface designed using Angular, a popular front-end framework for building dynamic web applications. Here is a description of the key features of the login page:

1. Header Section: The login page starts with a header section that typically includes the title of the application.

2. Login Form: The main component of the login page is the login form, which consists of input fields for the user to enter their email and password. The form also include additional element like checkbox for remember me

3. Validation: The login form includes validation to ensure that users provide the required information in the correct format. This helps prevent users from submitting incorrect or incomplete data.

4. Error Messages: If a user enters incorrect information an issue during the login process, error messages will be displayed to notify the user of the problem. These messages help users understand what went wrong and how to resolve it.

5. Submit Button: The submit button allows users to log in after entering their credentials. Clicking this button triggers the authentication process, where the user's credentials are verified against the database or authentication service.

A screenshot of a log in page

Description automatically generated

A screenshot of a computer

Description automatically generated6. Toastr: It provides a simple way to show warning messages, success notifications, error alerts to users if the data was wrong or he/she logged successfully

A screenshot of a login screen

Description automatically generated

7. Footer Section: The login page include a footer section with (SignUp ) link to click on it if the user don’t have account.

8. HTTP Service: Angular's HTTP service enables communication with the backend server to handle login requests. It facilitates sending user data, such as email, and password.

9.Router Service: After a successful login process, The Router service simplifies user navigation by redirecting them to the chat page . This service efficiently handles application routing and facilitates smooth transitions between various pages.

A screen shot of a computer program

Description automatically generatedThese two services, HTTP and Router, play integral roles in the Login functionality of GalaxyTalk, facilitating communication with the server and ensuring smooth navigation for users.

Sign up Page:

The signup page of GalaxyTalk allows new users to create an account on the platform. Users are prompted to enter their full name, email address, and password. The page includes form validation to ensure that the entered information meets the required criteria, such as minimum length for the name and password, and valid email format. Once the user submits the signup form, the provided information is sent to the server for account creation. Upon successful signup, users receive a confirmation message, and their account credentials are stored securely. Additionally, error handling is implemented to provide feedback in case of any issues during the signup process, such as invalid inputs or server errors.

- Services used

HTTP Service: Angular's HTTP service enables communication with the backend server to handle signup requests. It facilitates sending user data, such as name, email, and password, to the server for account creation. This service is crucial for interacting with the server-side application logic and databases.

Router Service: The Router service provided by Angular is utilized for navigation within the application. After a successful signup process, the Router service redirects the user to the chat page, ensuring seamless navigation and enhancing the overall user experience. This service manages the application's routing and navigation features, ensuring that users can easily move between different pages and components of the application.

These two services, HTTP and Router, play integral roles in the signup functionality of GalaxyTalk, facilitating communication with the server and ensuring smooth navigation for users.

In addition to the HTTP and Router services, GalaxyTalk's signup page also employs Angular's Reactive Forms module for validation purposes. This module provides a robust mechanism for handling form inputs and validating user-provided data before submission. By utilizing Reactive Forms, the signup page ensures that users enter valid information such as name, email, and password according to predefined criteria.

Reactive Forms offer features like built-in validators such as required fields, minimum length for names and passwords, and email format validation. These validators help maintain data integrity and enhance the overall user experience by providing meaningful error messages when users input incorrect or incomplete information.

A screenshot of a computer

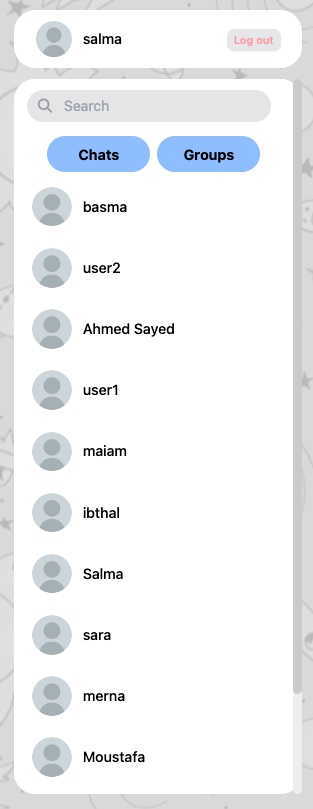
Description automatically generated

A screenshot of a login screen

Description automatically generated

A grey background with black text

Description automatically generatedChat Page:



Chat page Component:

Users Panels Component:

This component provides functionality for managing user interactions such as logging out and searching for other users. It typically contains a logout button and a search bar where users can search for specific individuals to chat with

Message Component: The Message component is responsible for displaying the messages exchanged between users. It dynamically renders incoming and outgoing messages in a chat-like format, typically showing the sender's name or avatar along with the message content and timestamp.

Input Field Component: The Input Field component enables users to compose and send messages. It includes a text input field where users can type their messages and a send button to submit them. Additionally, it may offer features like emoji support, file attachments, or message formatting options.

Chatbox Component:

The Chatbox component serves as the main container for the Message, and Input Field components. It provides the layout and structure for the chat interface, ensuring that all components are properly positioned and displayed within the chat window. This component may also handle additional functionality such as scrolling behavior to ensure that new messages are always visible.

chat page logic and services used:

In developing the chat application, robust real-time communication using Socket.IO was implemented. Socket.IO is a powerful library facilitating seamless, bidirectional communication between clients and servers. Leveraging Socket.IO has enabled instant messaging functionality, ensuring that messages are delivered and updated in real-time across users. This integration has significantly enhanced the responsiveness and interactivity of the chat interface, allowing for dynamic updates without the need for page reloads. Moreover, Socket.IO's scalability features have ensured that the application can seamlessly handle large volumes of concurrent connections, providing a smooth and reliable chat experience for users across A screenshot of a chat

Description automatically generateddifferent platforms and devices.

**Backend:**

implement the backend with Express.js, Node.js, and MongoDB for the chat application, we'll create endpoints for user authentication (login and signup), fetching all users, creating a chat between two users, getting chat messages and sending messages.

**User Authentication Endpoint:**

POST http://localhost:8000/user/login: Endpoint for user login. It accepts the user's credentials (email and password) in the request body and verifies them against the database. If the credentials are valid, it generates a JWT token and sends it back to the client.

POST http://localhost:8000/user/signup: Endpoint for user signup. It accepts the user's information (name, email, password) in the request body, creates a new user record in the database, and sends back a success message.

**Display All Users Endpoint:**

Get http://localhost:8000/user: Endpoint to fetch all users from the database. It retrieves all user records from the "users" collection and sends them back to the client.

**Create Chat Between Two Users Endpoint**:

POST http://localhost:8000/chat: Endpoint to create a chat between two users. It accepts the IDs of the two users involved in the chat in the request body, creates a new chat record in the "chats" collection, and sends back the details of the created chat.

**Get Chat Messages Endpoint:**

GET http://localhost:8000/message/chatId: Endpoint to fetch messages for a particular chat. It accepts the ID of the chat in the request parameters, retrieves all message records associated with the chat from the "messages" collection, and sends them back to the client.

**Send Message Endpoint:**

GET http://localhost:8000/message/chatId: Endpoint to send a message. It accepts the ID of the chat , the sender's ID, and the message content in the request body, creates a new message record in the "messages" collection, and sends back the details of the created message.

With these endpoints implemented, the backend will provide the necessary functionality for user authentication, managing users and chats, and sending messages within the chat application.

# Future Work:

User Profile Customization: Implement a feature that allows users to customize their profiles by adding profile pictures, setting status messages, and updating personal information.

Social Authentication: Integrate social authentication options (e.g., using Google or Facebook accounts) to streamline the signup process and provide users with additional login options.

Email Verification: Enhance account security by implementing email verification for newly registered users. This ensures that users provide valid email addresses and helps prevent spam or unauthorized account creations.

Group Chat Support: Extend the chat application to support group chats, allowing users to create and participate in conversations with multiple participants simultaneously.

Conclusion:

In conclusion, GalaxyTalk is a real-time chat application that provides users with a seamless and secure platform for communication. The signup page allows new users to create accounts easily, with robust form validation and error handling ensuring a smooth signup process. Looking ahead, there are several avenues for future work to enhance the application further, including user profile customization, social authentication, and advanced security features like two-factor authentication and message encryption. By continuing to iterate and improve upon the existing features while introducing new functionalities, GalaxyTalk aims to provide users with a modern and feature-rich chat experience while prioritizing security and usability.