A PROJECT REPORT

SUBMITTED BY

MADDINA KHUSHIRAM

IN

PRATHYUSHA ENGINEERING COLLEGE

TO

COINCENT INTRNSHIP

Project Title : Chat Application Development

Abstract :

For real-time communication and backend services, the project utilized **Firebase**, which provided essential features such as **Firebase Authentication** for user sign-in, **Firebase Realtime Database** for storing and synchronizing chat messages, and **Firebase Firestore** for managing user data and chat history. The integration of Firebase enabled seamless **real-time updates**, ensuring that messages are instantly delivered and visible to users.

The report outlines the **design and development phases**, detailing the creation of the **chat interface**, **message handling mechanisms**, and **user management functionalities**. It discusses the challenges encountered during development, such as optimizing **real-time database interactions** and ensuring **secure user authentication**, and describes the solutions implemented to address these challenges.

Performance evaluations focused on metrics such as **message latency**, **system responsiveness**, and **user experience**, demonstrating that the application effectively meets its goals. The report concludes with a discussion of potential future enhancements, including the integration of **message encryption**, **push notifications**, and **improved user interface elements**.

In an increasingly connected world, effective communication is crucial for both personal and professional interactions. Explore the possibilities of **chat application development**, where cutting-edge technology meets user-centric design to create engaging and functional messaging platforms.

In today's fast-paced environment, real-time communication tools are essential for effective personal connections, team collaboration, and customer interactions. By leveraging the latest technologies, we ensure your chat application facilitates smooth, instantaneous communication.

In the digital age, effective communication tools are more important than ever. Our expertise in **chat application development** helps you create innovative solutions that meet your communication needs.

Explore features such as **real-time messaging** all designed with a focus on **user experience** and . From initial design to final deployment, we ensure that your chat application is both functional and engaging.

Project : Chat Application Development

INDEX

1.Objectives:

1.1 Firebase :

Firebase provides a suite of cloud-based tools and services that facilitate the development of real-time applications. For a chat application, Firebase serves several critical objectives:

Objectives of Firebase:

- **Real-Time Data Synchronization:**
  - **Objective:** Ensure that chat messages and user interactions are instantly reflected across all connected clients.
  - **How:** Firebase's **Realtime Database** and **Cloud Firestore** enable automatic data synchronization. When a message is sent by a user, it is immediately pushed to all other users who are connected to the chat room.
- **User Authentication and Management:**
  - **Objective:** Provide a secure and user-friendly way for users to sign in and manage their accounts.
  - **How: Firebase Authentication** offers multiple sign-in methods (email/password, Google, Facebook, etc.) and handles user registration, login, and account management.
- **Scalable Backend Services:**
  - **Objective:** Support a growing number of users and messages without requiring extensive backend infrastructure.
  - **How:** Firebase's cloud-based services are designed to scale automatically based on usage, handling high volumes of simultaneous connections and data updates.
- **Message Storage and Retrieval:**
  - **Objective:** Store chat messages and retrieve them when needed, ensuring message history is accessible.
  - **How:** Firebase's **Realtime Database** or **Cloud Fire store** stores chat messages and can be queried to fetch past conversations.
- **Push Notifications:**

- **Objective:** Alert users about new messages or other important events even when they are not actively using the app.
- **How: Firebase Cloud Messaging (FCM)** allows you to send notifications to users about new messages or updates.

1.2 HTML

HTML (Hyper Text Markup Language) provides the structural foundation of the chat application. Its objectives in the development of a real-time chat application are:

Objectives of HTML:

- **Define Application Structure:**
  - **Objective:** Create the foundational layout of the chat application, including elements like chat windows, message input fields, and user lists.
  - **How:** HTML elements such as , <input>, and <button> are used to build the structure of the user interface, organizing content into a functional layout.
- **Form Elements for User Interaction:**
  - **Objective:** Provide forms and inputs for users to send messages and interact with the chat application.
  - **How:** HTML forms and input elements, like <textarea> for message input and <button> for sending messages, facilitate user interactions.
- **Embed and Organize Content:**
  - **Objective:** Display content such as chat messages, user profiles, and notifications.
  - **How:** HTML tags like <p>, <ul>, and are used to structure and present chat messages, user information, and notifications.

1.3. CSS

CSS (Cascading Style Sheets) is responsible for the presentation and visual design of the chat application. Its objectives are:

Objectives of CSS:

- **Design and Style the User Interface:**
  - **Objective:** Create an aesthetically pleasing and user-friendly design for the chat application.
  - **How:** CSS styles the HTML elements, defining properties like color, font, spacing, and layout. It ensures that the chat application has an attractive and intuitive interface.
- **Create a Responsive Layout:**
  - **Objective:** Ensure that the chat application works well on various devices and screen sizes.
  - **How:** CSS media queries and flexible layouts adapt the design for different screen sizes, ensuring that the application is usable on desktops, tablets, and smartphones.
- **Manage Layout and Positioning:**
  - **Objective:** Arrange elements on the page to create a functional and organized user interface.
  - **How:** CSS properties like display, flex, grid, and position are used to manage the layout of chat windows, message lists, and input areas.
- **Enhance User Experience with Animations and Transitions:**
  - **Objective:** Improve the user experience with smooth interactions and visual feedback.

    How: CSS animations and transitions provide visual effects for message notifications, user interactions, and UI changes.

1.4. JavaScript

JavaScript adds interactivity and dynamic behavior to the chat application. Its objectives are:

Objectives of JavaScript:

- **Implement Real-Time Messaging Features:**

    - **Objective:** Enable users to send and receive messages instantly.
    - **How:** JavaScript code handles the connection to Firebase's **Realtime Database** or **Fire store**, sending and receiving messages in real time using Firebase SDK methods.
- **Manage User Interactions:**
    - **Objective:** Handle user inputs, such as sending messages and joining chat rooms.
    - **How:** JavaScript code processes events like form submissions and button clicks to send messages and update the chat interface.
- **Update the User Interface Dynamically:**
    - **Objective:** Reflect real-time changes in the chat application's user interface.
    - **How:** JavaScript updates the DOM (Document Object Model) to display new messages, notifications, and other dynamic content as events occur.
- **Handle Authentication and Session Management:**
    - **Objective:** Manage user sign-ins and maintain user sessions.
    - **How:** JavaScript interacts with **Firebase Authentication** to handle user sign-ins, logouts, and session management, updating the UI based on the user's authentication state.
- **Fetch and Display Chat History:**
    - **Objective:** Retrieve and display past messages in the chat application.
    - **How:** JavaScript queries the Firebase database for message history and updates the chat window with the retrieved messages.

1.5. UI/UX Design with Figma:

While UI design focuses on visual and interactive elements, UX design is concerned with the overall experience and satisfaction of the user. A successful chat application requires both a well-designed interface and a smooth, enjoyable user experience. Good UI design enhances usability and aesthetic appeal, while effective UX design ensures that the application is user-friendly and meets the needs of its audience.

Figma is a popular web-based design tool used for creating user interfaces and prototypes. It offers a range of features that support both UI and UX design processes.

Features of Figma :

- **Design Tools:** Figma provides robust tools for creating designs, including vector graphics, typography, and layout controls.
    - **Example:** Tools for drawing shapes, creating icons, and setting up design grids.
- **Prototyping:** Allows designers to create interactive prototypes that demonstrate how users will navigate through the application.
    - **Example:** Linking screens to simulate user interactions like clicking buttons or navigating between chat rooms.
- **Collaboration:** Figma supports real-time collaboration, enabling multiple designers to work on the same project simultaneously.
    - **Example:** Team members can comment, share feedback, and make changes in real-time.
- **Design Systems:** Supports the creation of reusable components and design systems for consistency across different parts of the application.
    - **Example:** Designing a chat bubble component that can be used throughout the application.
- **Responsive Design:** Tools for designing layouts that adapt to different screen sizes and devices.
    - **Example:** Designing a responsive chat window that looks good on both mobile and desktop screens.

1.6. IBM CLOUD FOR HOSTING A WEBSITE :

IBM Cloud is a comprehensive cloud platform that offers a range of services and tools for developing, deploying, and managing applications. It provides solutions for businesses and developers looking to host websites, applications, and other digital services. This section will cover the key features of IBM Cloud, how it supports the hosting of application websites, and best practices for utilizing IBM Cloud effectively.

IBM Cloud is IBM's cloud computing platform that offers a suite of services including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and

Software as a Service (SaaS). It provides the infrastructure and tools needed to build, deploy, and manage applications across various environments, including public, private, and hybrid clouds.

Key Features of IBM Cloud

- **Flexible Infrastructure:** Offers scalable compute resources, storage, and networking options.
- **Wide Range of Services:** Includes computing power, database management, AI and machine learning tools, and developer services.
- **Global Reach:** Data centers around the world to support global application deployment and disaster recovery.
- **Security and Compliance:** Built-in security features and compliance certifications for various industries.

Benefits of Using IBM Cloud for Hosting

- **Scalability:** Easily scale resources up or down based on traffic and usage demands.
- **Reliability:** High availability with data replication and failover capabilities.
- **Performance:** Advanced networking and compute services for high-performance applications.
- **Cost Efficiency:** Pay-as-you-go pricing models and cost management tools.
- **Integration:** Integration with various IBM services and third-party tools.

2.Introduction :

In today's digital age, chat applications have become a cornerstone of personal and professional communication. These applications offer platforms for real-time interactions, allowing users to exchange messages instantly, manage conversations, and connect with others across the globe. This project focuses on the development of a **real-time chat application** designed to provide a seamless and efficient text-based communication experience.

Chat applications serve a wide range of purposes, from facilitating casual conversations to supporting complex group collaborations and professional dialogues. Their significance lies in their ability to enable users to communicate in real-time, manage group discussions, and access conversation history, all from a convenient web interface.

The objective of this project was to develop a chat application using **HTML**, **CSS**, **JavaScript**, and **Firebase**. The application aims to offer core features such as **real-time messaging**, **user authentication**, and **responsive design**. **HTML** was used to structure the content of the chat interface, **CSS** provided the visual styling and layout, and **JavaScript** handled the interactive elements and real-time data updates. **Firebase** was employed for backend services, including **real-time database management**, **user authentication**, and **application hosting**.

While UI design focuses on visual and interactive elements, UX design is concerned with the overall experience and satisfaction of the user. A successful chat application requires both a well-designed interface and a smooth, enjoyable user experience. Good UI design enhances usability and aesthetic appeal, while effective UX design ensures that the application is user-friendly and meets the needs of its audience.

The development process involved designing a user-friendly interface, implementing real-time messaging features, and ensuring data security and application performance. The report covers the design and implementation phases, addresses the challenges encountered, and presents solutions for optimizing the application's functionality.

Looking ahead, the project identifies opportunities for future enhancements, including the potential integration of **message encryption**, **push notifications**, and **multimedia communication features**. By achieving its objectives, this project lays the groundwork for future developments in web-based communication technologies

A chat application is a type of software that enables users to communicate with each other via text messages, either in real-time or through asynchronous exchanges. These applications serve as platforms for various forms of interaction, ranging from casual conversations between friends to professional discussions within organizations. This introduction explores the concept, significance, and technological foundation of chat applications, focusing on the development of a modern, real-time chat application using **HTML, CSS, JavaScript**, and **Firebase**.

In an era where digital communication is integral to both personal and professional interactions, **chat applications** have become ubiquitous tools for facilitating instant, text-based conversations across the globe. These applications cater to diverse needs, from simple one-on-one chats to complex group discussions, and have evolved from basic messaging platforms to sophisticated solutions with advanced features and capabilities.

2.1 Scope and Features of the Chat Application :

This project aims to develop a chat application with core features focused on essential communication functionalities:

- **Real-Time Messaging:** Enables users to send and receive messages instantly.

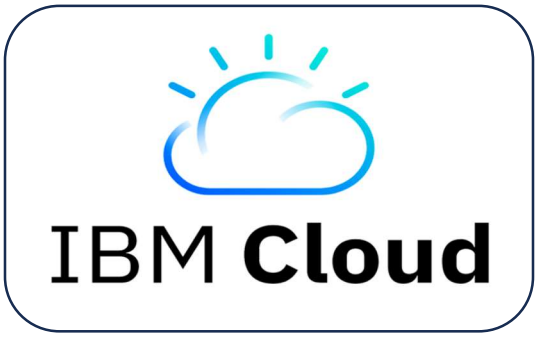
- **User Authentication:** Provides secure sign-in options using Firebase Authentication.
- **Message History:** Maintains a log of past conversations for users to reference.
- **Group Chats:** Allows multiple users to participate in a single chat room.
- **User Interface:** A clean, intuitive design that enhances user interactions.

2.2 Future Enhancements :

Future development opportunities for the chat application include:

- **Message Encryption:** Implementing end-to-end encryption to secure message contents against unauthorized access.
- **Push Notifications:** Adding notifications to alert users of new messages or updates.
- **Voice and Video Calls:** Introducing features for voice and video communication to expand the application's capabilities.
- **File Sharing:** Enabling users to share images, documents, and other files within the chat application.
- **Customizable User Profiles:** Adding features for user profile management, including profile pictures, status updates, and personal settings.

2.3 Logo of objectives :



3. Methodology :

Creating a chat application involves several steps, from designing the user interface to implementing the backend and real-time communication. Below is a comprehensive methodology to guide you through developing a chat application using HTML, CSS, JavaScript, UI/UX design from Figma, and Firebase for backend services.

3.1. Planning and Designing :

Requirements Gathering

- Define the features: user authentication, real-time messaging, group chats, media sharing, notifications, etc.
- Identify user personas and their needs.

UI/UX Design :

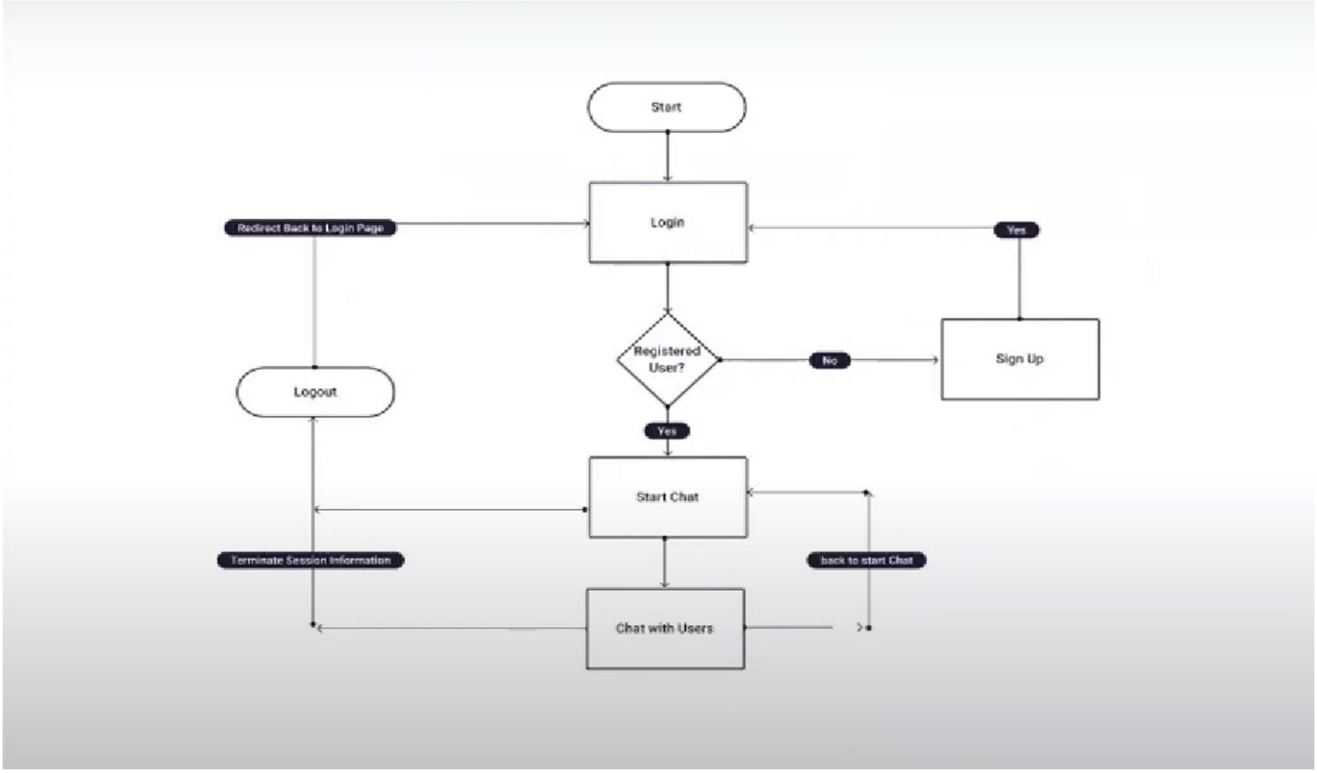
- Use Figma to create wireframes and prototypes.
- Design the user flow and interactions.
- Focus on creating an intuitive and responsive design.

3.1.1 Archetecture design :

The Archetecture design is represents how the website will flows step by step process and it is very important to making a professional websites.

UI (User Interface) design focuses on the visual aspects and interactive elements of a website or application, such as layout, color schemes, typography, buttons, and icons, ensuring that it is aesthetically pleasing and easy to navigate. In contrast, UX (User Experience) design emphasizes the overall feel and functionality, concentrating on user satisfaction, ease of use, and efficient task completion by considering user research, interaction design, information architecture, and usability. While UI is about the look and interactive experience, UX is about the user's journey and how the interface serves their needs.

Example for Archetecture of a website :

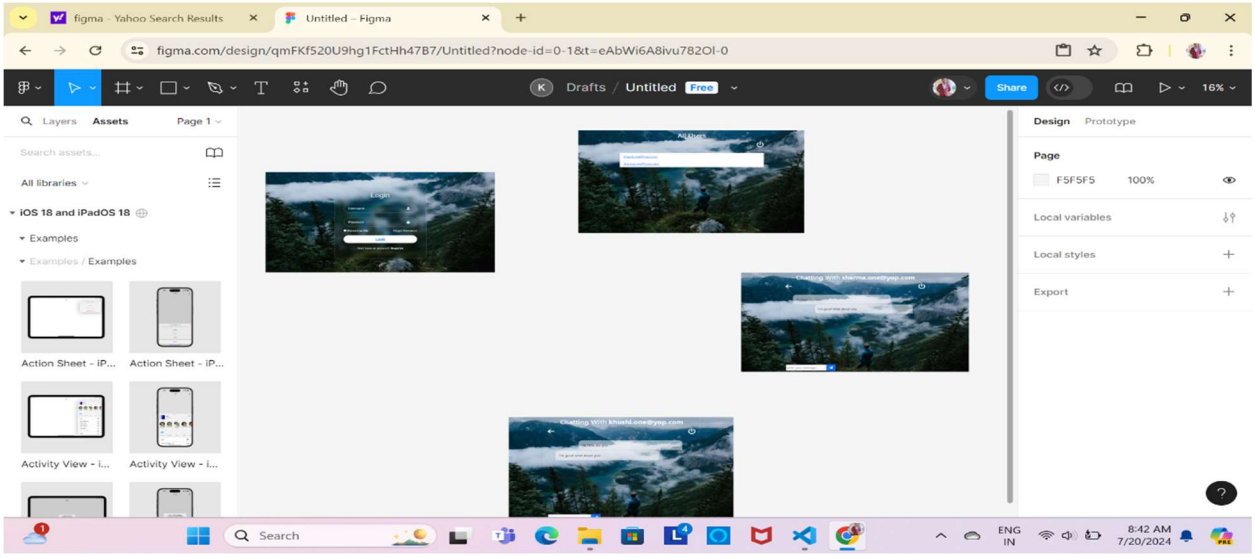


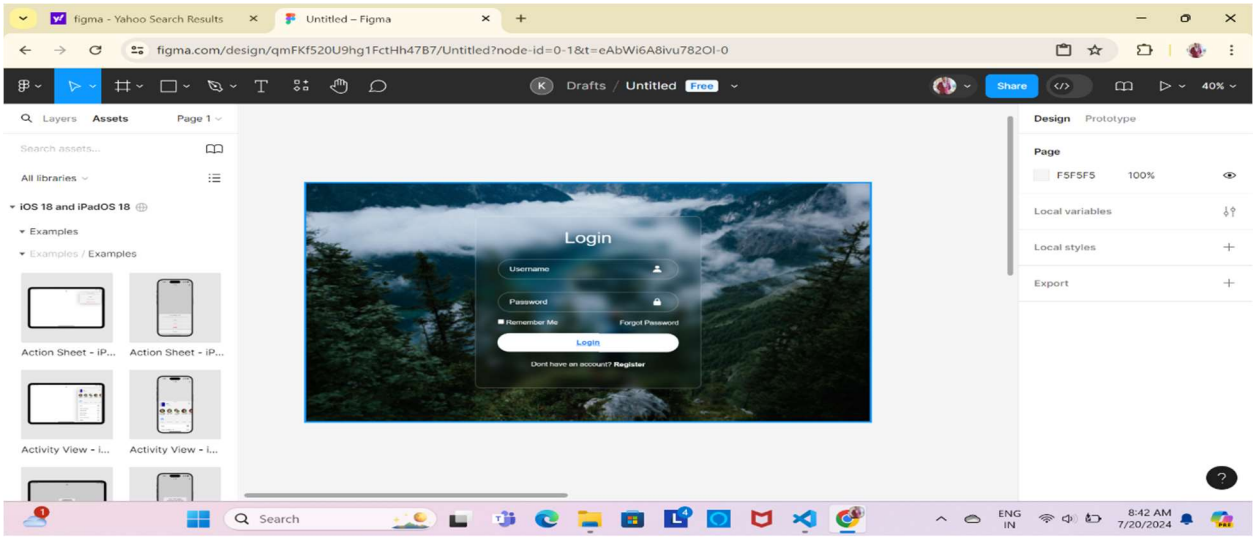
3.1.2 Design the UI :

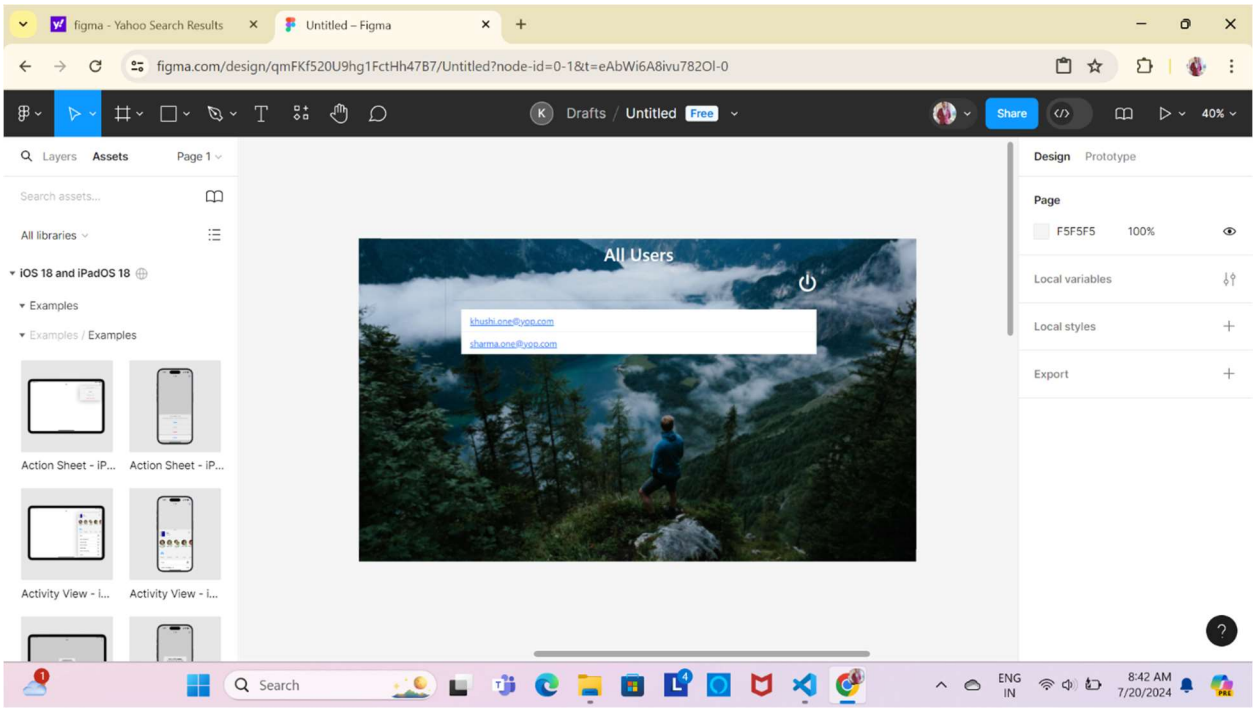
Here we can use the Figma for design the outline of the webpage. In this section we can able to assume and find the correct structure of a website .

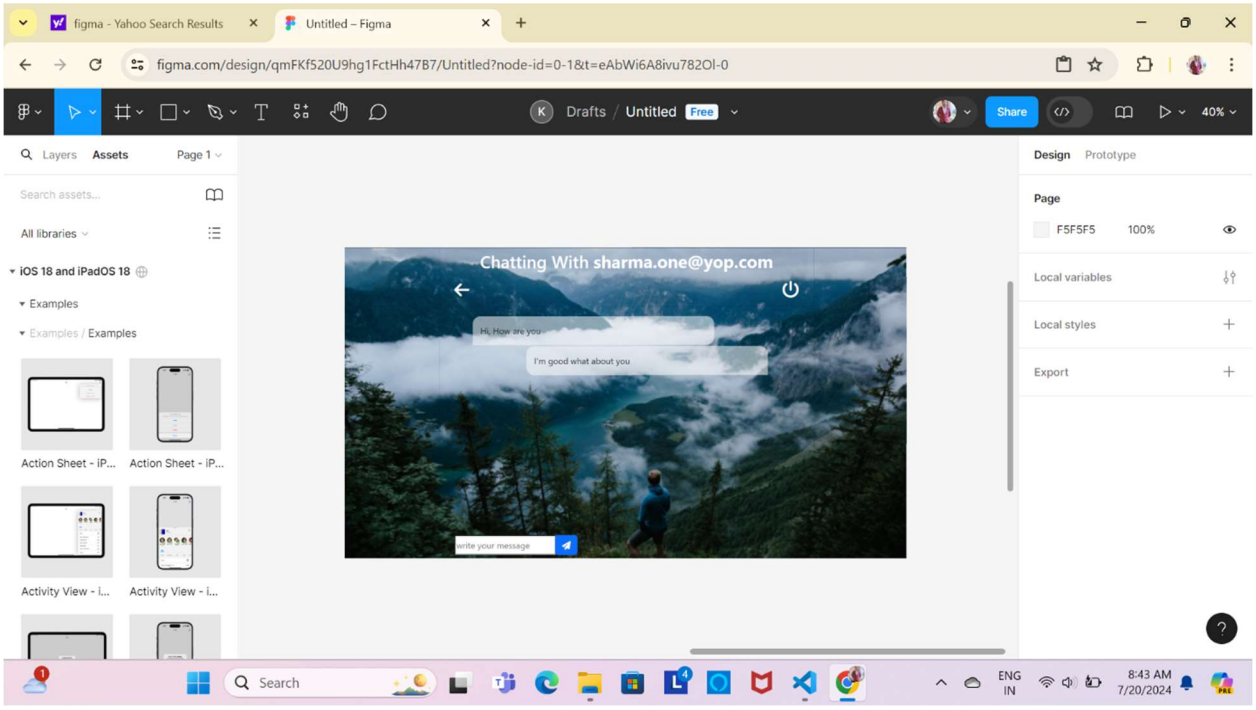
Designing the UI for a chat website involves creating an intuitive, visually appealing, and functional interface that enhances user experience. Key elements include a clear messaging interface with distinct message bubbles, timestamps, and typing indicators; easy navigation and search functionality; customizable user profiles and settings; real-time notifications; and robust security features like end-to-end encryption. The design should be accessible and responsive, adapting to various screen sizes, and include interactive features like emojis and file sharing. Continuous usability testing and iteration based on user feedback are crucial for ongoing improvement.

UI design Layouts :






figma - Yahoo Search Results
Untitled – Figma
figma.com/design/qmFKf520U9hg1FctHh47B7/Untitled?node-id=0-1&t=eAbWi6A8ivu782Ol-0
Drafts / Untitled Free
Share
40%
Layers Assets Page 1
Search assets...
All libraries
iOS 18 and iPadOS 18
Examples
Examples / Examples
Action Sheet - iP...
Action Sheet - iP...
Activity View - i...
Activity View - i...
All Users
khushi.one@yop.com
sharma.one@yop.com
Design Prototype
Page
F5F5F5
100%
Local variables
Local styles
Export
Search
ENG IN
8:42 AM 7/20/2024


figma - Yahoo Search Results
Untitled – Figma
figma.com/design/qmFKf520U9hg1FctHh47B7/Untitled?node-id=0-1&t=eAbWi6A8ivu782Ol-0
Drafts / Untitled Free
Share
40%
Layers Assets Page 1
Search assets...
All libraries
iOS 18 and iPadOS 18
Examples
Examples / Examples
Action Sheet - iP...
Action Sheet - iP...
Activity View - i...
Activity View - i...
Chatting With sharma.one@yop.com
Hi, How are you
I'm good what about you
write your message
Design Prototype
Page
F5F5F5
100%
Local variables
Local styles
Export
Search
ENG IN
8:43 AM 7/20/2024

3.2 Firebase setup :

Setting up Firebase for a website involves several steps

1.**Create a Firebase Project:**

- Go to the Firebase Console and click on "Add project".
- Follow the prompts to create a new project.
- Once created, you'll be taken to the project dashboard.

2.**Add Firebase to Your Web App:**

- In the Firebase Console, select the web icon to add Firebase to your web app.
- Register your app by giving it a nickname and optionally setting up Firebase Hosting.
- You will receive a configuration snippet containing your project's settings.

3.**Include Firebase SDK:**

- Include the Firebase SDKs (Software Development Kits) in your website by adding them to your HTML file. These SDKs enable your website to use Firebase services.

4.**Initialize Firebase:**

- Use the configuration snippet provided earlier to initialize Firebase in your website's JavaScript file. This step connects your website to your Firebase project.

5.**Configure Firebase Services:**

- Depending on the Firebase services you want to use (such as Firestore for databases, Authentication for user sign-in, etc.), include the specific SDKs and configure them in your JavaScript file.

6.**Set Up Firebase Authentication :**

- If you want to enable user authentication, set up Firebase Authentication by including its SDK and configuring it to handle sign-in methods like email/password or social logins.

7. **Deploy Your Website :**

- If you choose to use Firebase Hosting to deploy your website, install the Firebase CLI (Command Line Interface) tool, initialize your project for hosting, and deploy your site using Firebase Hosting.

3.3 Setup of the application :

- setup the basic html file in vscode editor
- setup the basic css file
- setup the basic javascript file
- setup bootstrap file in folder and in html code

After completion of the setup process keep it improve the development process and initialize the backend setup like firebase for real time process

3.4 Configuration of firebase in application :

1. **Create a Firebase Project**: Set up a new project in the Firebase Console.
2. **Register Your Web App**: Add your web app to the Firebase project and get the configuration snippet.
3. **Include Firebase SDKs**: Add Firebase SDK scripts to your HTML file and initialize Firebase with the configuration snippet.
4. **Initialize Firebase Services**: Include and initialize the SDKs for the Firebase services you need (Fire store, Authentication, Realtime Database, Cloud Storage).
5. **Optional: Set Up Firebase Hosting**: Use Firebase CLI to deploy your web application.

By following these steps, you can configure Firebase in your web application and leverage its backend services to build a robust and scalable web app.

4. CODING PART :

4.1 Firebase setup code in all the HTML files like index.html , chat.html , startchat.html, registration.html files:

```
<script type="module">

  import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-
app.js";
  import {getAuth,signInWithEmailAndPassword} from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-
auth.js";

   const firebaseConfig = {

    apiKey: "AIzaSyBpK5M7KUWPWlL8pWpOGLnRTm446KA6gD8",
    authDomain: "samplechatapp-eff22.firebaseapp.com",
    databaseURL: "https://samplechatapp-eff22-default-
rtdb.firebaseio.com",
    projectId: "samplechatapp-eff22",
    storageBucket: "samplechatapp-eff22.appspot.com",
    messagingSenderId: "662779901868",
    appId: "1:662779901868:web:7ca9e350a8f897334dec09",
    measurementId: "G-TCK9YE8DZC"
  };


</script>
```

4.2 Index.html (loginpage):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Form</title>
  <link rel="stylesheet" href="styles.css">
  <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css'
rel='stylesheet'>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css
" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.6.0/css/all.min.css" integrity="sha512-
Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve5UMg
5GT6L4JJg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
  <script>try{Typekit.load({async:true});}catch(e){}</script>
</head>
<body>
  <main>
  <div class="wrapper">
    <form action="">
      <h1>Login</h1>
      <div class="input-box" id="email_address">
        <input type="text" placeholder="Username" required>
        <i class='bx bxs-user'></i>
      </div>
      <div class="input-box" id="password">
        <input type="password" placeholder="Password" required>
        <i class='bx bxs-lock-alt' ></i>
      </div>
      <div class="remember-forgot">
        <label><input type="checkbox">Remember Me</label>
        <a href="#">Forgot Password</a>
      </div>
      <button type="submit" class="btn"> <a
href="startchart.html">Login</a></button>
      <div class="register-link">
        <p>Dont have an account? <a href="registration.html">Register</a></p>
      </div>
```

```
    </form>
  </div>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></scri
pt>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.j
s" integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-
0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
<script src="validation.js"></script>
<script type="module">

  import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-app.js";
  import {getAuth,signInWithEmailAndPassword} from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-auth.js";

   const firebaseConfig = {
    apiKey: "AIzaSyBpK5M7KUWPWlL8pWpOGLnRTm446KA6gD8",
    authDomain: "samplechatapp-eff22.firebaseapp.com",
    databaseURL: "https://samplechatapp-eff22-default-rtdb.firebaseio.com",
    projectId: "samplechatapp-eff22",
    storageBucket: "samplechatapp-eff22.appspot.com",
    messagingSenderId: "662779901868",
    appId: "1:662779901868:web:7ca9e350a8f897334dec09",
    measurementId: "G-TCK9YE8DZC"
  };

  const app = initializeApp(firebaseConfig);
      function authenticateFromFirebase(){
        if(validateLoginInputCredentials()){
          const auth = getAuth(app);
          createUserWithEmailAndPassword(auth,email,password)
          .then((userCredentials)=>{
            const user = userCredentials.user;
            localstorage.setItem('loggedin_user_id',user.uid)
            localstorage.setItem('logged_user_email',user.email)
            setTimeout(location.href="startchart.html",2000)
          }).catch((error)=>{
```

```
            const errorCode = error.code;
            const errorMessage = error.message;
          })
        }
      }
      $("authenticateUser").on('click',function(){
        saveUserToFirebase();
      })
</script>
</main>
</body>
</html>
```

Style.css (loginpage) :

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "Poppins", sans-serif;
  }
  body{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background: url(newimg.png) no-repeat;
    background-size: cover;
    background-position: center;
  }

  .input-box input::placeholder{
    color: #fff;
  }
  .input-box i{
    position: absolute;
    right: 20px;
    top: 30%;
    transform: translate(-50%);
    font-size: 20px;


  }
```

4.3 registration.html :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Registration Form</title>
    <meta name="viewport" content="width=device-width,
      initial-scale=1.0"/>
    <link rel="stylesheet" href="registration.css" />
    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css'
rel='stylesheet'>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.6.0/css/all.min.css" integrity="sha512-
Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve5UMg5GT6
L4JJg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <script>try{Typekit.load({async:true});}catch(e){}</script>
  </head>
  <body>
    <div class="dude">
    <div class="container">
      <h1 class="form-title">Registration</h1>
      <form action="#">
        <div class="main-user-info">
          <div class="user-input-box">
            <label for="username">Username</label>
            <input type="text"
                    id="username"
                    name="username"
                    placeholder="Enter Username"/>
          </div>
          <div class="user-input-box">
            <label for="email">Email</label>
            <input type="email"
                    id="email_address"
                    name="email"
                    placeholder="Enter Email"/>
```

```
          </div>
          <div class="user-input-box">
            <label for="password">Password</label>
            <input type="password"
                    id="password"
                    name="password"
                    placeholder="Enter Password"/>
          </div>
          <div class="user-input-box">
            <label for="confirmPassword">Confirm Password</label>
            <input type="password"
                    id="confirm_password"
                    name="confirmPassword"
                    placeholder="Confirm Password"/>
          </div>
        </div>
        <div class="form-submit-btn" id="saveUser">
          <input type="submit" value="Register" >
        </div>
        <div class="back-to-login">
          <center>
            <font color="white" size="5">If you have already an account click on
</font>
            <a href="index.html">Click Me</a>
          </center>
        </div>
      </form>
    </div>
    </div>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-
0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
    <script src="validation.js"></script>
    <script type="module">
```

```
      import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-app.js";
      import { getAuth,createUserWithEmailAndPassword} from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-auth.js";
      import { getDatabase,ref,set,push} from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-database.js";
       const firebaseConfig = {
        apiKey: "AIzaSyBpK5M7KUWPWlL8pWpOGLnRTm446KA6gD8",
        authDomain: "samplechatapp-eff22.firebaseapp.com",
        databaseURL: "https://samplechatapp-eff22-default-rtdb.firebaseio.com",
        projectId: "samplechatapp-eff22",
        storageBucket: "samplechatapp-eff22.appspot.com",
        messagingSenderId: "662779901868",
        appId: "1:662779901868:web:7ca9e350a8f897334dec09",
        measurementId: "G-TCK9YE8DZC"
      };

      const app = initializeApp(firebaseConfig);
      const db= getDatabase();
      function saveUserToFirebase(){
        if(validateSignUpInputCredentials()){
          const auth = getAuth(app);
          createUserWithEmailAndPassword(auth,email,password)
          .then((userCredentials)=>{
            const user = userCredentials.user;
            push(ref(db,'/registered_users'),{
              userid : user.uid,
              email : user.email
            }).then(()=>{
              alert('Successfully Registered');
              location.href="index.html";
            })
          }).catch((error)=>{
            const errorCode = error.code;
            const errorMessage = error.message;
          })
        }
      }
      $("#saveUser").on('click',function(){
        saveUserToFirebase();
      })
    </script>
  </body>
</html>
```

Registration.css :

```
body{
    display: flex;
    height: 100vh;
    justify-content: center;
    align-items: center;
    background: url(newimg.png) no-repeat;
    background-size: cover;
}
.container{
    width: 100%;
    max-width: 650px;
    background: rgba(0, 0, 0, 0.5);
    padding: 28px;
    margin: 0 28px;
    border-radius: 10px;
    box-shadow: inset -2px 2px 2px white;
}
.form-title{
    font-size: 26px;
    font-weight: 600;
    text-align: center;
    padding-bottom: 6px;
    color: white;
    text-shadow: 2px 2px 2px black;
    border-bottom: solid 1px white;
}
.main-user-info{
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    padding: 20px 0;
}

.user-input-box:nth-child(2n){
    justify-content: end;
}

.user-input-box{
    display: flex;
    flex-wrap: wrap;
```

```
    width: 50%;
    padding-bottom: 15px;
}

.user-input-box label{
    width: 95%;
    color: white;
    font-size: 20px;
    font-weight: 400;
    margin: 5px 0;
}

.user-input-box input{
    height: 40px;
    width: 95%;
    border-radius: 7px;
    outline: none;
    border: 1px solid grey;
    padding: 0 10px;
}
.form-submit-btn{
    margin-top: 40px;
}
.form-submit-btn input{
    display: block;
    width: 100%;
    margin-top: 10px;
    font-size: 20px;
    padding: 10px;
    border:none;
    border-radius: 3px;
    color: rgb(209, 209, 209);
  background: rgba(63, 114, 76, 0.7);
}
```

4.4 startchart.html (All users) :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>All Users</title>
  <link rel="stylesheet" href="startchat.css">
```

```
  <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.6.0/css/all.min.css" integrity="sha512-
Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve5UMg5GT6L4JJg=="
crossorigin="anonymous" referrerpolicy="no-referrer" />
  <script>try{Typekit.load({async:true});}catch(e){}</script>
</head>
<body>
  <main>
      <div class="container">
        <div class="row justify-content-center">
          <div class="card" style="width: 700px; height: 600px; background: transparent;">
            <div class="card-header text-center">
            <span><h2><font color="white">All Users</font></h2></span>
            <font color="white" size="6"><span class="float-end" data-bs-toggle="tooltip"
data-bs-placement="right" title="Logout" ><i class="fa fa-power-off"></i></font></span>
            </div>
            <div class="card-body">
            <ul class="list-group list-group-flush" id="user-list" style="cursor:pointer;">
              <li class="list-group-item"><a href="chat.html">khushi.one@yop.com</a></li>
              <li class="list-group-item"><a href="chat copy.html">sharma.one@yop.com</a></li>
            </ul>
            </div>
          </div>
        </div>
      </div>
  </main>


<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
<script>
  const tooltipTriggerList = document.querySelectorAll('[data-bs-toggle="tooltip"]')
  const tooltipList = [...tooltipTriggerList].map(tooltipTriggerEl => new
bootstrap.Tooltip(tooltipTriggerEl))
</script>
<script type="module">
```

```
    import { initializeApp } from "https://www.gstatic.com/firebasejs/10.12.3/firebase-
app.js";
    import { getAuth} from "https://www.gstatic.com/firebasejs/10.12.3/firebase-auth.js";
    import { getDatabase,ref,onValue } from
"https://www.gstatic.com/firebasejs/10.12.3/firebase-database.js";

    const firebaseConfig = {
    apiKey: "AIzaSyBpK5M7KUWPWlL8pWpOGLnRTm446KA6gD8",
    authDomain: "samplechatapp-eff22.firebaseapp.com",
    databaseURL: "https://samplechatapp-eff22-default-rtdb.firebaseio.com",
    projectId: "samplechatapp-eff22",
    storageBucket: "samplechatapp-eff22.appspot.com",
    messagingSenderId: "662779901868",
    appId: "1:662779901868:web:7ca9e350a8f897334dec09",
    measurementId: "G-TCK9YE8DZC"
  };

  const app = initializeApp(firebaseConfig);
  var db = getDatabase();
  const userRef= ref(db,"/registered_users");

  function showUsers(Users){
    var html="";
    for(let user in Users){

      html=html+'<li class="list-group-item" data-user_id="'+users[user].userid+'"data-
user_email="'+Users[user].email+'</li>'
    }
    $("#user-list").html(html);
  }

$(document).on('click',".loadchat",function(e){
  localstorage.setItem('chat_user_id',$(this).attr('data-user_id'))
  localstorage.setItem('chat_user_email',$(this).attr('data-user_email'))
  setTimeout(location.href="chat.html",2000)

})

</script>
</body>
</html>
```

Startchart.css :

```
body{

  background: url(newimg.png) no-repeat;
  background-size: cover;
  background-position: center;
}
```

4.5 Chat.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>chat section</title>
  <link rel="stylesheet" href="chat.css">
  <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.6.0/css/all.min.css" integrity="sha512-
Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve5UMg5GT6L4JJg=="
crossorigin="anonymous" referrerpolicy="no-referrer" />
  <script>try{Typekit.load({async:true});}catch(e){}</script>

</head>
<body>
  <main>
    <div class="container">
      <div class="row justify-content-center">
        <div class="card" style="width: 700px; height: 600px; background: transparent;">
          <div class="card-header text-center">
          <span><h2><font color="white">Chatting With
<b>sharma.one@yop.com</b></font></h2></span>
          <font color="white" size="6"><span class="float-start" data-bs-toggle="tooltip"
data-bs-placement="right" title="Go back to chat section"><i class="fa fa-arrow-
left"></i></font></span>
```

```
          <font color="white" size="6"><span class="float-end" data-bs-toggle="tooltip" data-
bs-placement="right" title="Logout" ><i class="fa fa-power-off"></i></font></span>
          </div>
          <div class="card-body">
            <div class="chats" id="chatBox">
              <ul class="chat-bubble-container">
                <li class="sent" >
                  <p class="chat-bubble from-message">Hi, How are you</p>
                </li>
                <li class="reply">
                  <p class="chat-bubble to-message">I'm good what about you</p>
                </li>
              </ul>
            </div>
          </div>
          <center>
          <div class="card-footer">
            <div class="chat-input">
              <div class="input-group mb-3">
                <input type="text" class="chat from-control" placeholder="write your message">
                <div class="input-group-append">
                  <button class="btn btn-primary submit"><i class="fa fa-paper-
plane"></i></button>
                </div>
              </div>
            </div>
          </div>
          </center>
        </div>
      </div>
    </div>
</main>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
<script>
  if(localStorage.getItemById('loggedin_user_id')==""||
localStorage.getItem(loggedin_user_id)==undefined ||
localstorage.getItem('loggedin_user_id')==null){
  location.href="index.html";
```

```
}
document.getElementById('email-label').innerHTML="chatting with
<b>"+localStorage.getItem('chat_user_email')+"</b>";
const tooltipTriggerList = document.querySelectorAll('[data-bs-toggle="tooltip"]')
const tooltipList = [...tooltipTriggerList].map(tooltipTriggerEl => new
bootstrap.Tooltip(tooltipTriggerEl))
function triggerLogout(){
  localStorage.removeItem('loggedin_user_id')
  localStorage.removeItem('chat_user_id')
  localStorage.removeItem('chat_user_email')
  location.href = "index.html";
}

function goBackToStartPage(){
  localStorage.removeItem('chat_user_id')
  localStorage.removeItem('chat_user_email')
  location.href = "startchart.html";

}
</script>
<script type="module">

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.12.3/firebase-app.js";
import { getAuth} from "https://www.gstatic.com/firebasejs/10.12.3/firebase-auth.js";
import {  } from "https://www.gstatic.com/firebasejs/10.12.3/firebase-database.js";

const firebaseConfig = {
apiKey: "AIzaSyBpK5M7KUWPWlL8pWpOGLnRTm446KA6gD8",
authDomain: "samplechatapp-eff22.firebaseapp.com",
databaseURL: "https://samplechatapp-eff22-default-rtdb.firebaseio.com",
projectId: "samplechatapp-eff22",
storageBucket: "samplechatapp-eff22.appspot.com",
messagingSenderId: "662779901868",
appId: "1:662779901868:web:7ca9e350a8f897334dec09",
measurementId: "G-TCK9YE8DZC"
};

const app = initializeApp(firebaseConfig);
const db=getDatabase();

$(window).on('keydown',function(e){
if(e.which==13){
newChat();
}
})
```

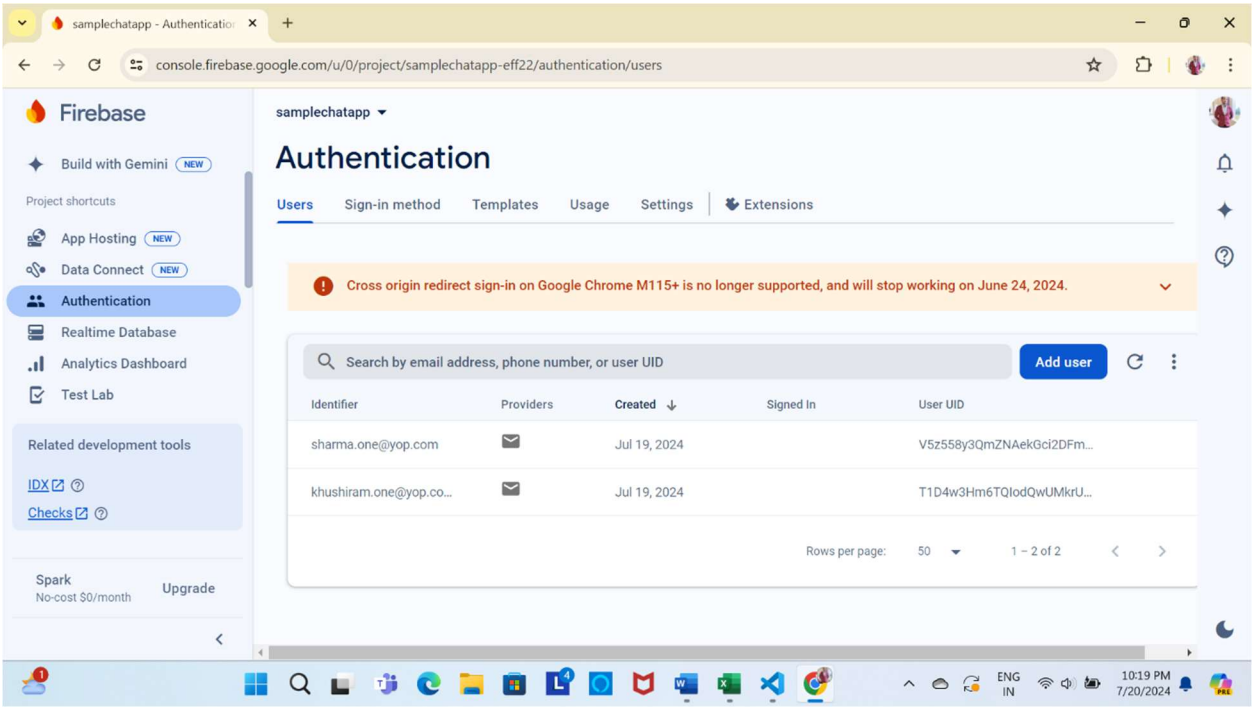
```
function newChat(){
var chat = $(".chat").val();
if($.trim(chat)==""){
return false
}
insertChatData(chat)
}
function insertChatData(chat){
push(ref(db,"/chats"),{
from:localStorage.getItem('loggedin_user_id'),
to:localStorage.getItem('chat_user_id'),
message : chat
})

}
const chatRef = ref(db,"/chats");
onValue(chatRef, (snapshot)=>{
var arr=[];
snapshot.forEach((childSnapshot)=>{
const childData = childSnapshot.val();
if((childData.from == localstorage.getItem('loggedin_user_id') || childData.to
==('child_user_id') || childData.to == localStorage.getItem('chat_user_id'))){
  arr.push(childData);
}
})
writeChatOnScreen(arr);

})

function writeChatsOnScreen(chats){
$(".chats ul").html("");
for(let chat in chats){
var from_user_id = chats[chat].from
var to_user_id = chats[chat].to
var stored_messages = chats[chat].message
var message_type;
var bubble;
if(from_user_id == localStorage.getItem('loggedin_user_id')){
  message_type = "sent ";
  bubble = "from-message"
}else{
  message_type='reply';
  bubble="to-message";
```

```
}
$('<li class="'+ message_type +'"><p class="'+bubble+'">'+stored_message+'</p></li>').
appendTo($(".chat ul"))
$(".chat").val(null)
}
}

</script>
</body>
</html>
```

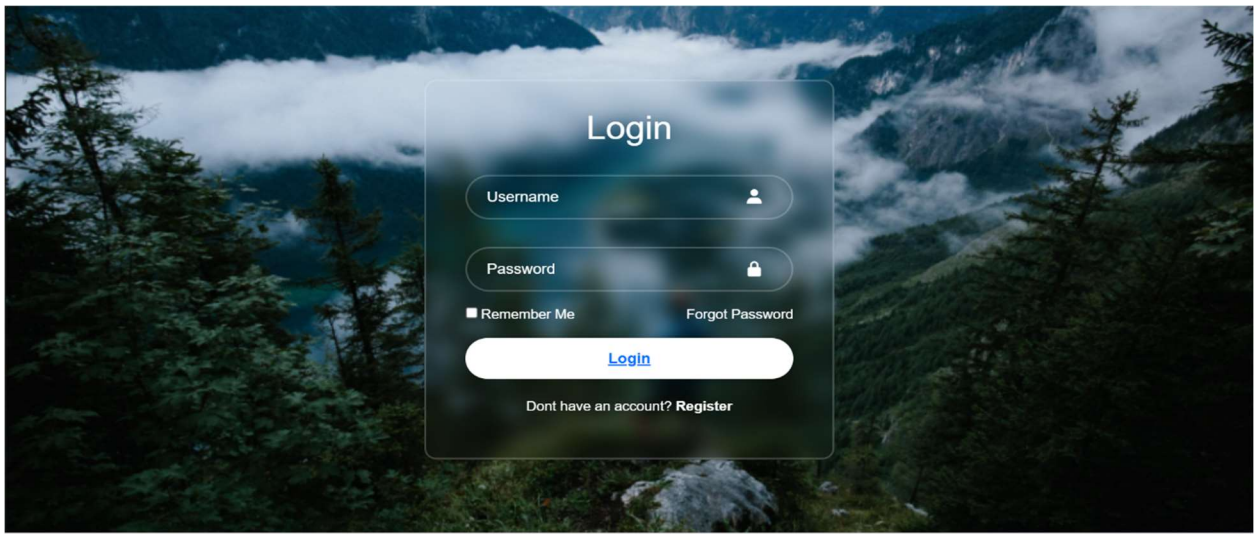
Chat.css :

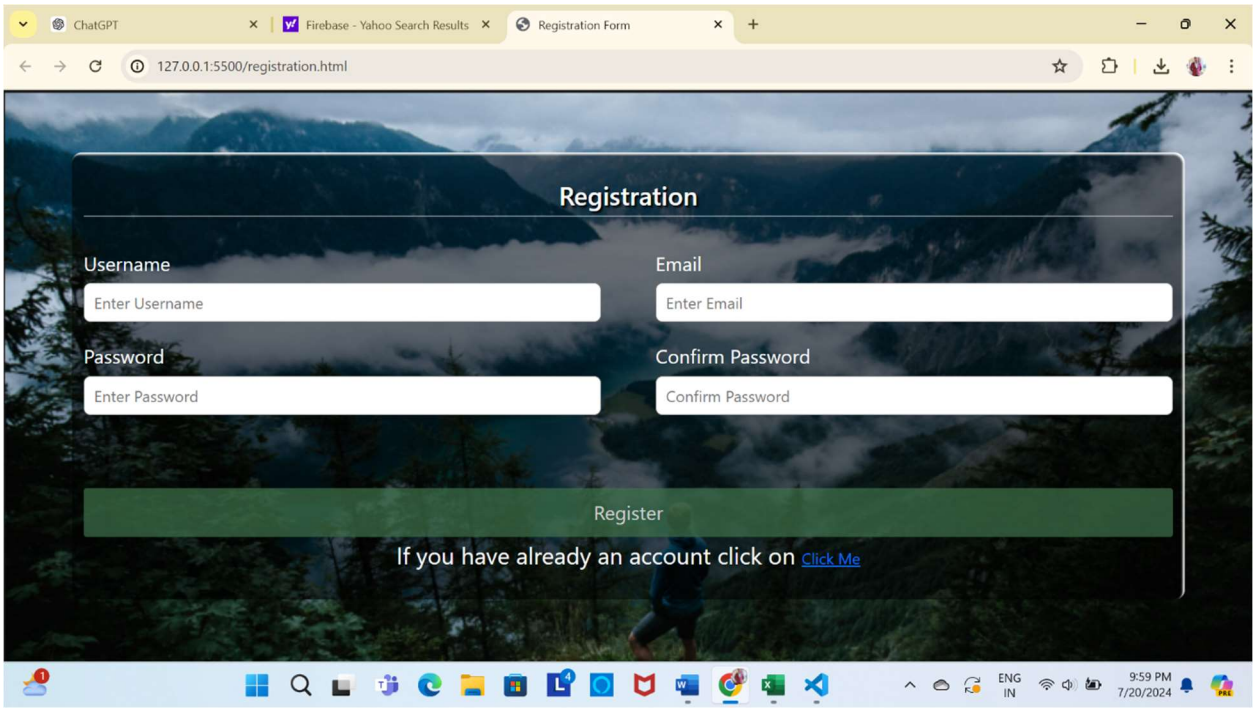
```
body{

    background: url(newimg.png) no-repeat;
    background-size:cover;

}
.chats{
    max-height: 450px;
    overflow: auto;
}
.chat-bubble-container{
    list-style: none;
}
.chat-bubble{
    max-width: 450px;
    padding: 15px;
    margin: 2px;
    background: rgba(252, 249, 249, 0.548);
    border-radius: 15px;
    text-align: left;
}
.from-message{
    border-bottom-left-radius:0;
    position: relative;
}
.to-message{
    border-bottom-right-radius:0;
    right: -100px !important;
    position: relative;
}
```

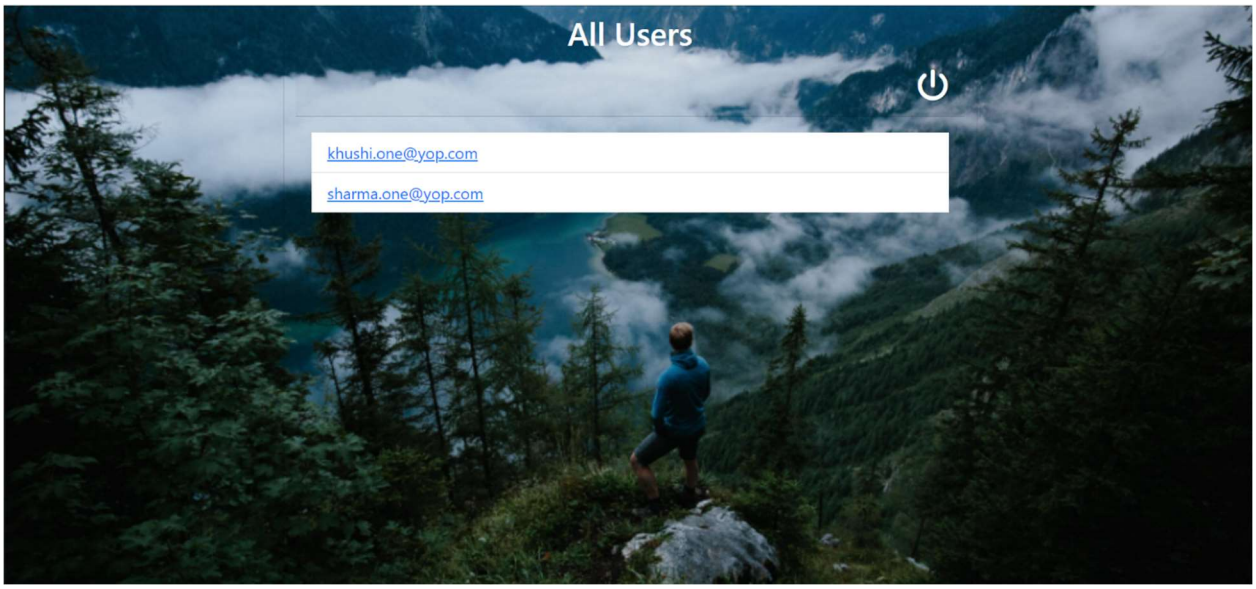
THE RESULT DATA WILL DISPLAY IN THE FIREBASE ON AUTHENTICATION SECTION OR REALTIMEDATABASE SECTION .

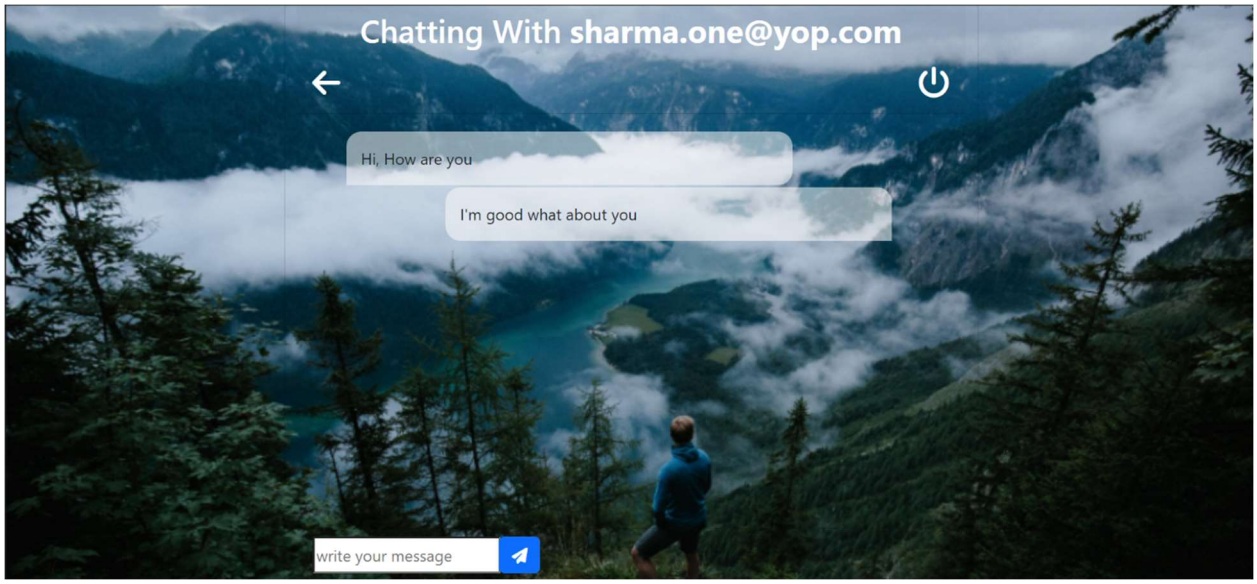


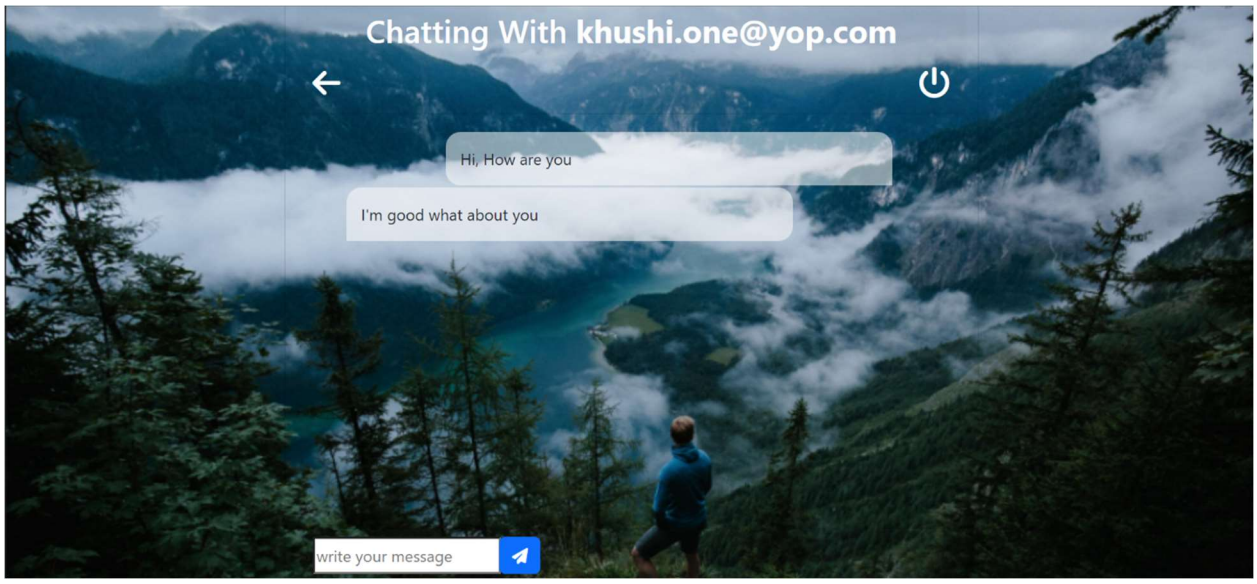
5.OUTPUT AND RESULT :




ChatGPT
Firebase - Yahoo Search Results
Registration Form
127.0.0.1:5500/registration.html
Registration
Username
Enter Username
Email
Enter Email
Password
Enter Password
Confirm Password
Confirm Password
Register
If you have already an account click on Click Me
ENG
IN
9:59 PM
7/20/2024


All Users
khushi.one@yop.com
sharma.one@yop.com

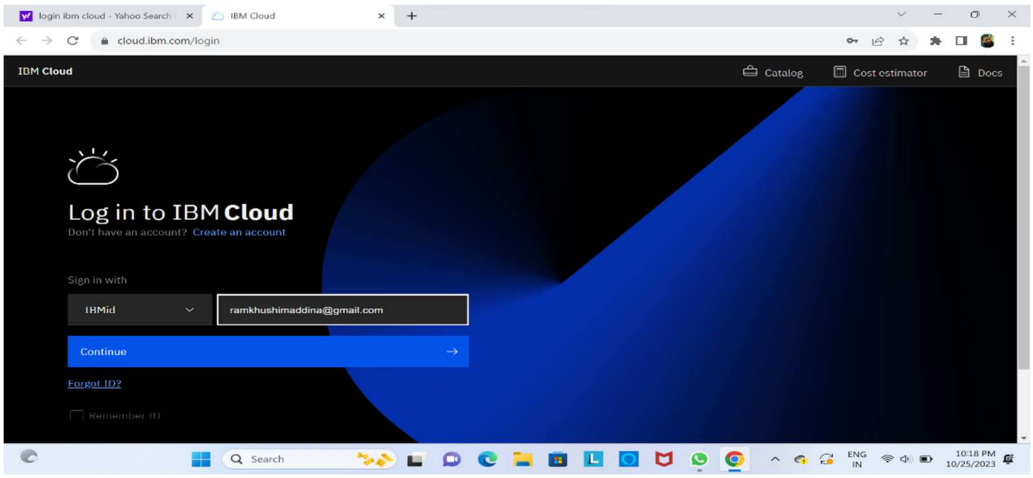

Chatting With sharma.one@yop.com
Hi, How are you
I'm good what about you
write your message


Chatting With khushi.one@yop.com
Hi, How are you
I'm good what about you
write your message

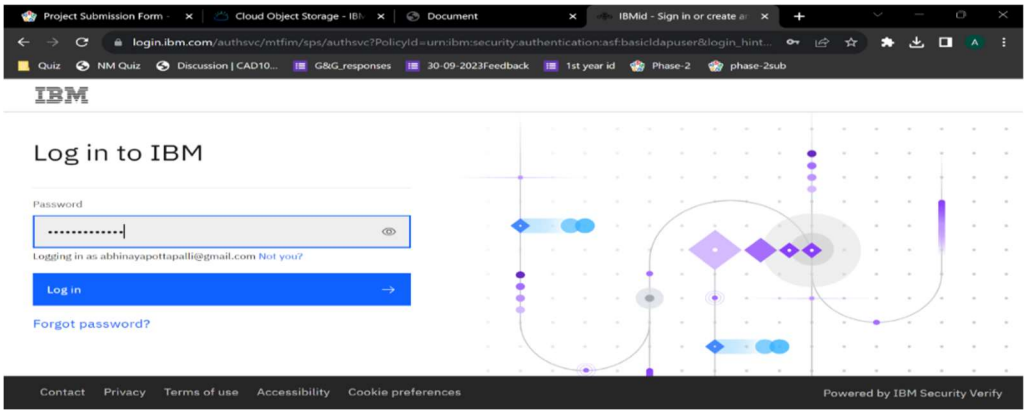
6 . DEPLOYMENT OF OUR WEBSITE IN IBM CLOUD :

Signup for IBM cloud account:

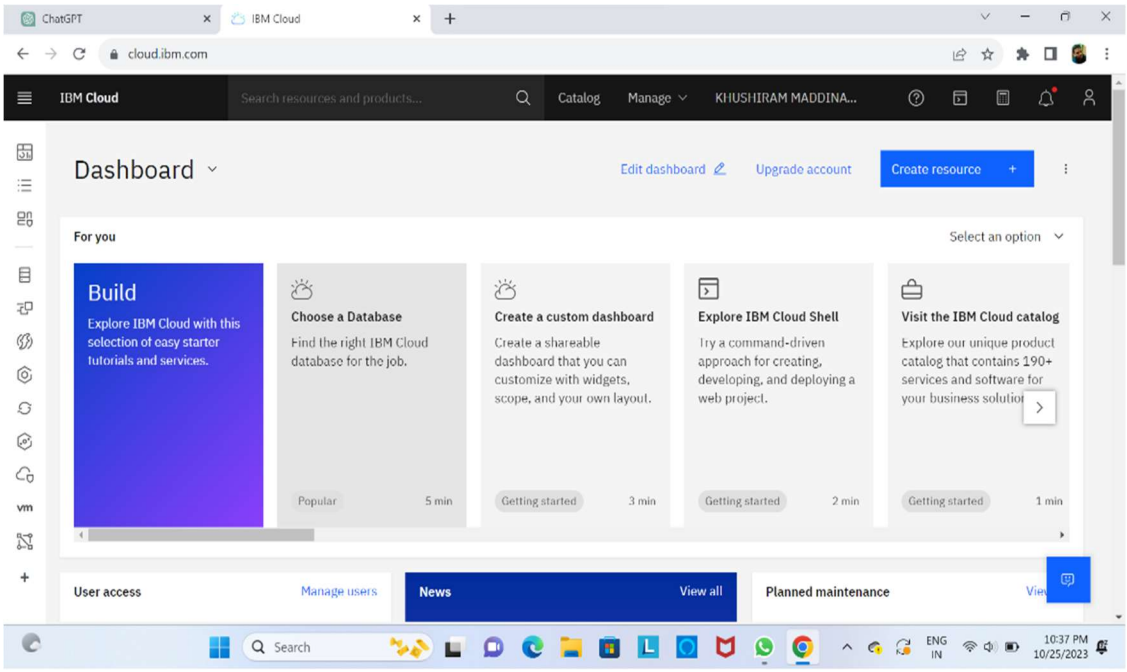
After creating the IBM Cloud Account, login with the IBMid . Enter your IBM id then click continue.



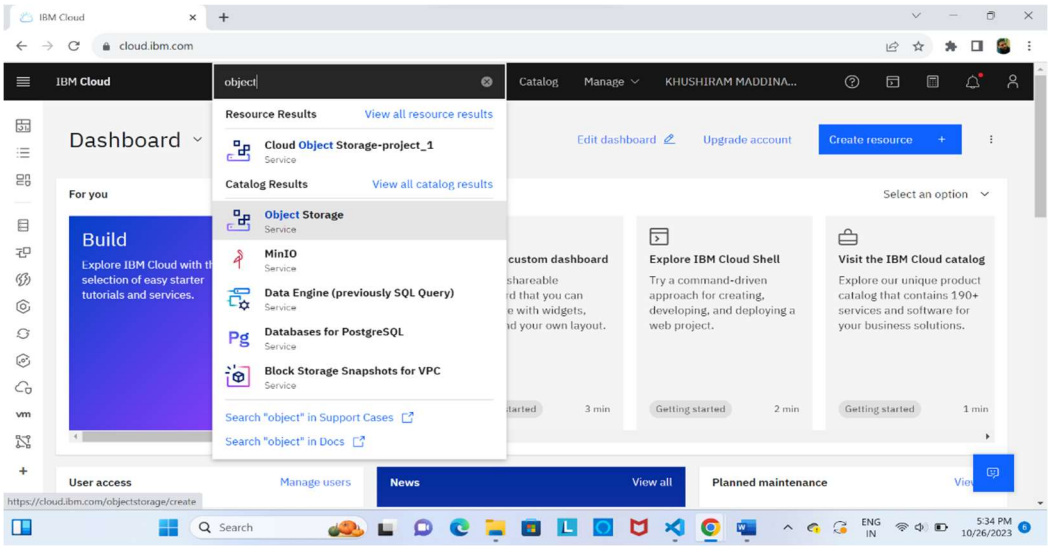
Enter your password, then click login :



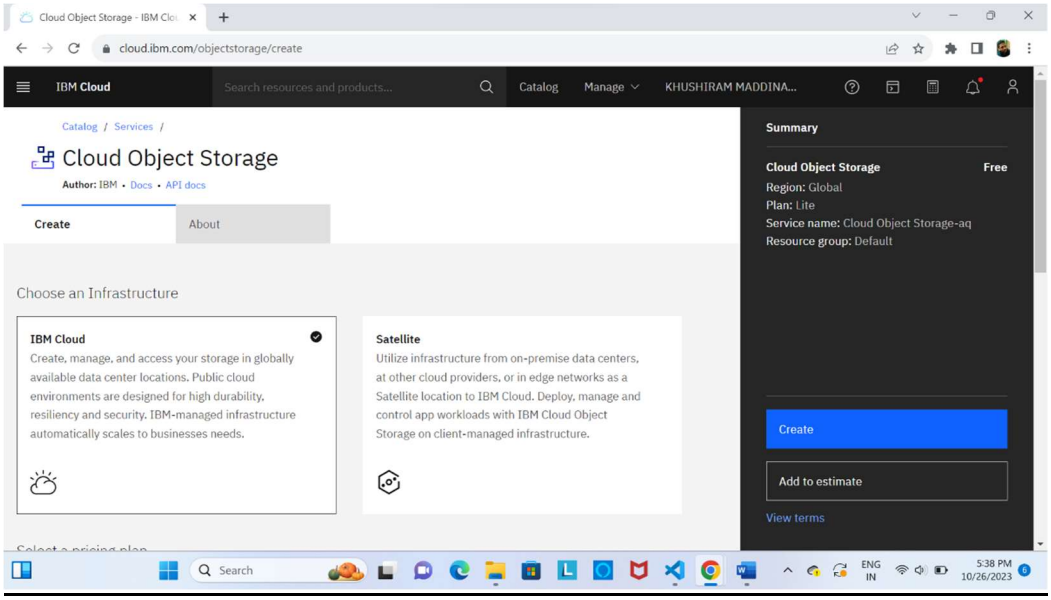
The homepage of the IBM account as follows:

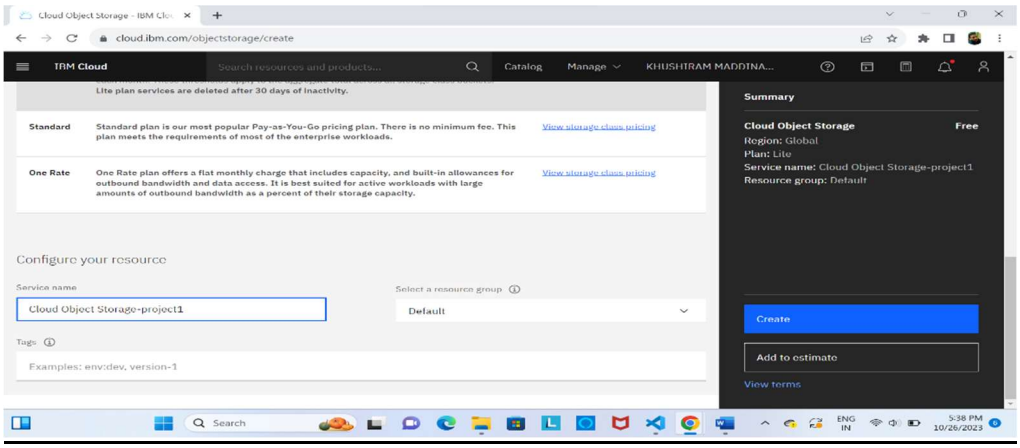


Click on the search bar , Object storage :

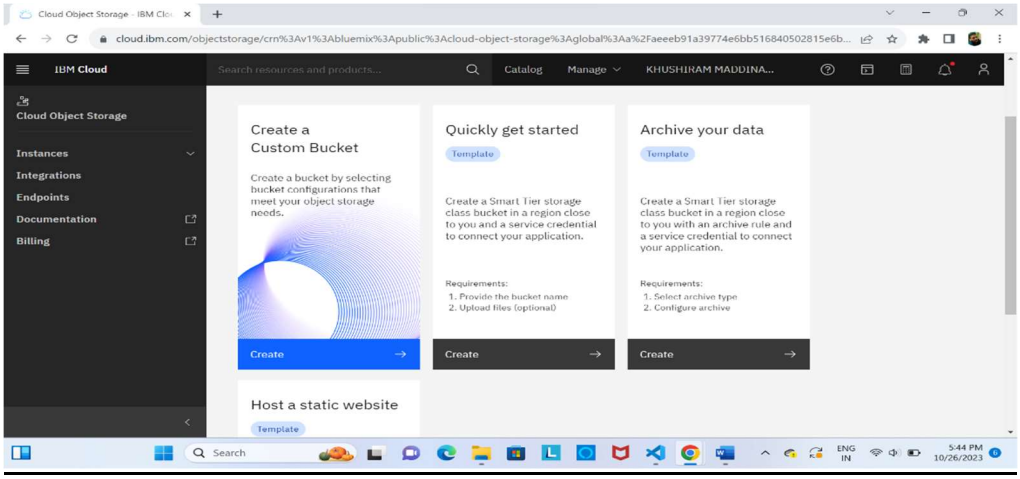


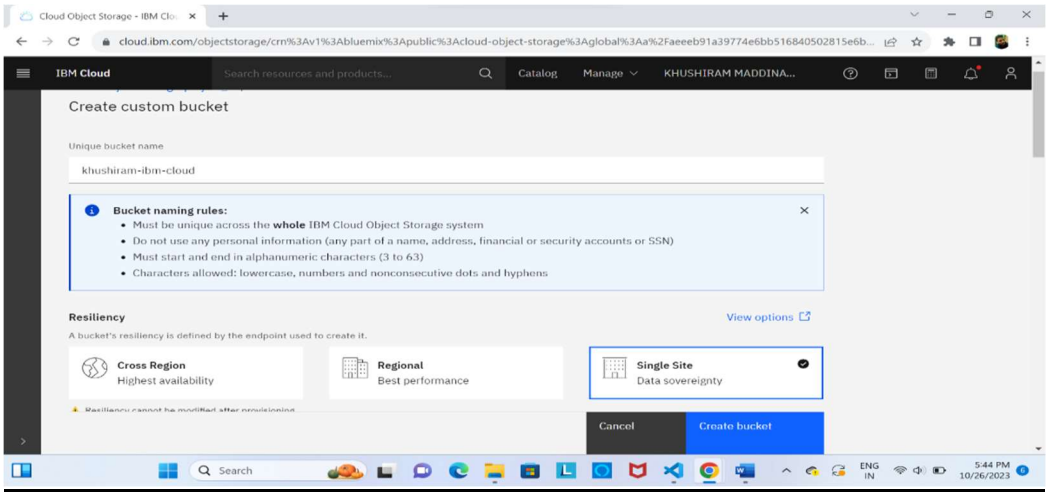
Scroll down give the service name and then click on create :





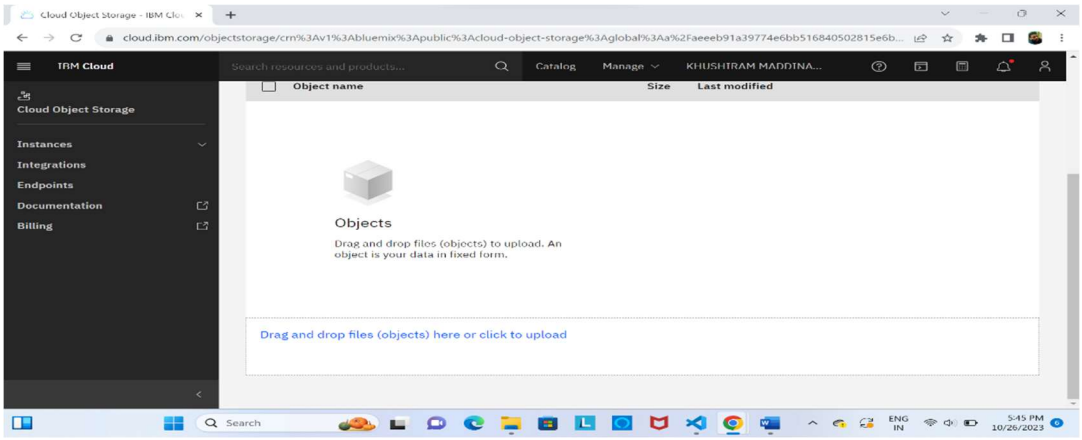
Click on create bucket and customize the bucket :



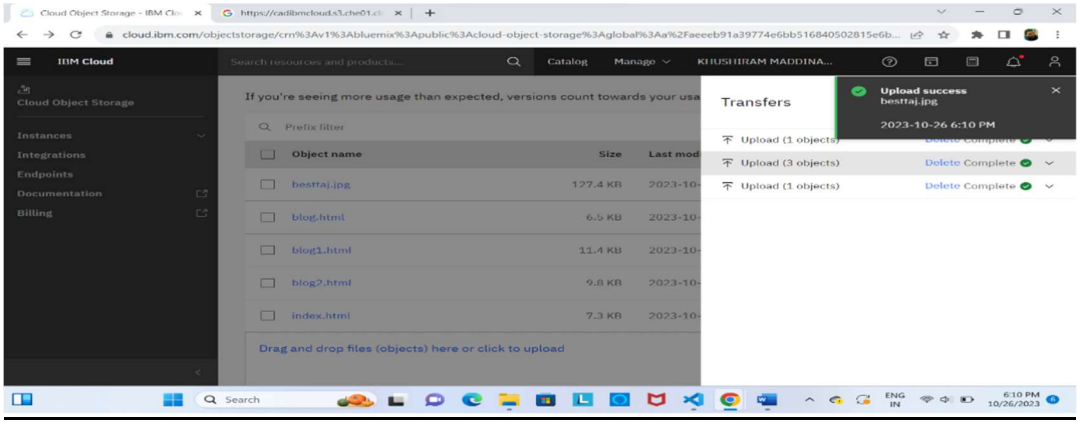


Give the unique bucket name and set as single site then click on create bucket

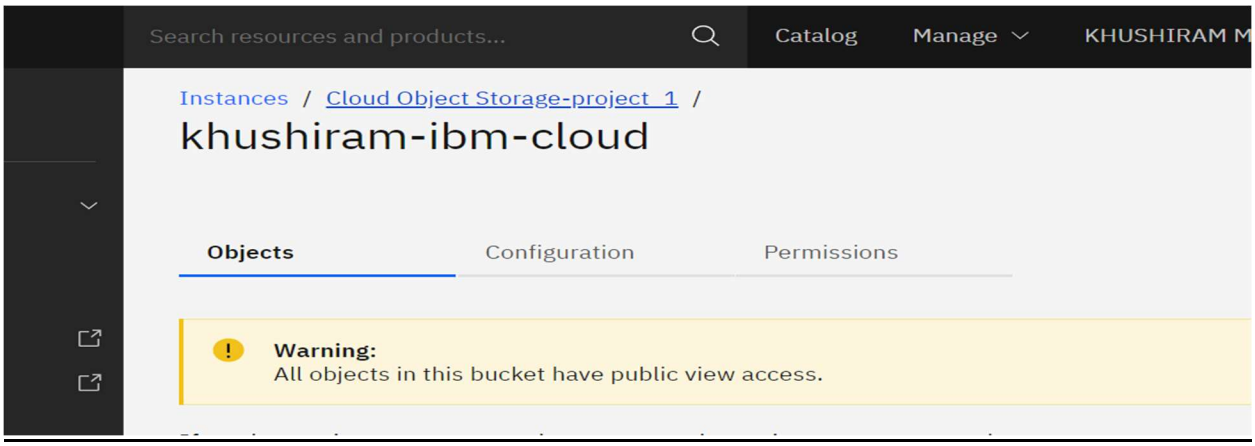
Then you get interface like this:



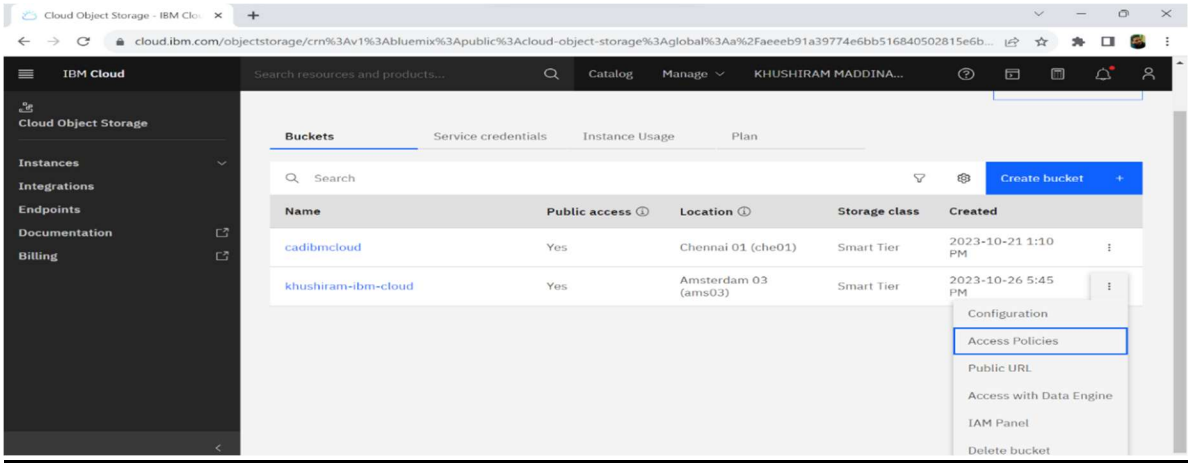
<u>Then you need to upload the html and css file in this objects :</u>



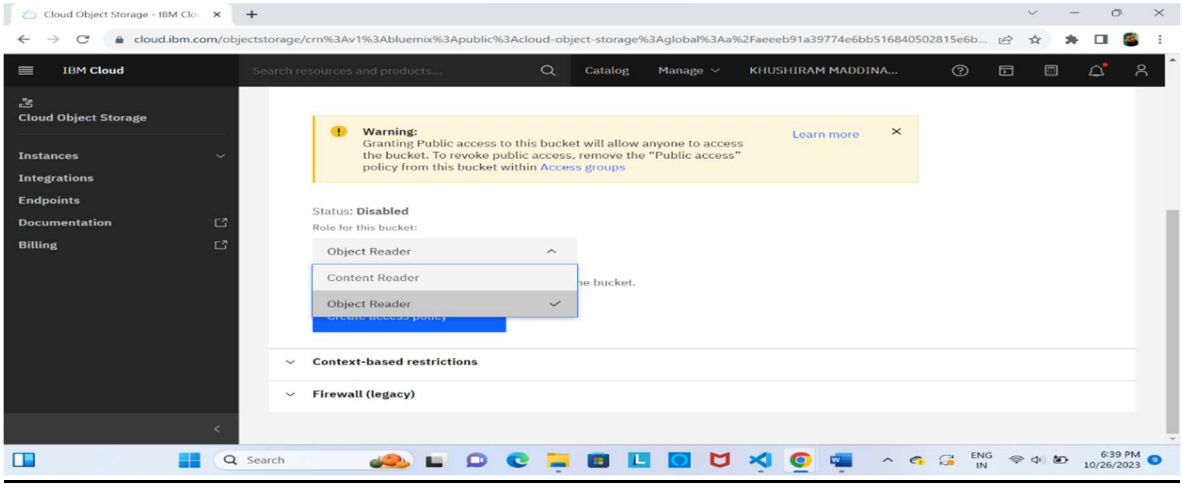
<u>After that you give public access for URL , click on cloud object storage :</u>



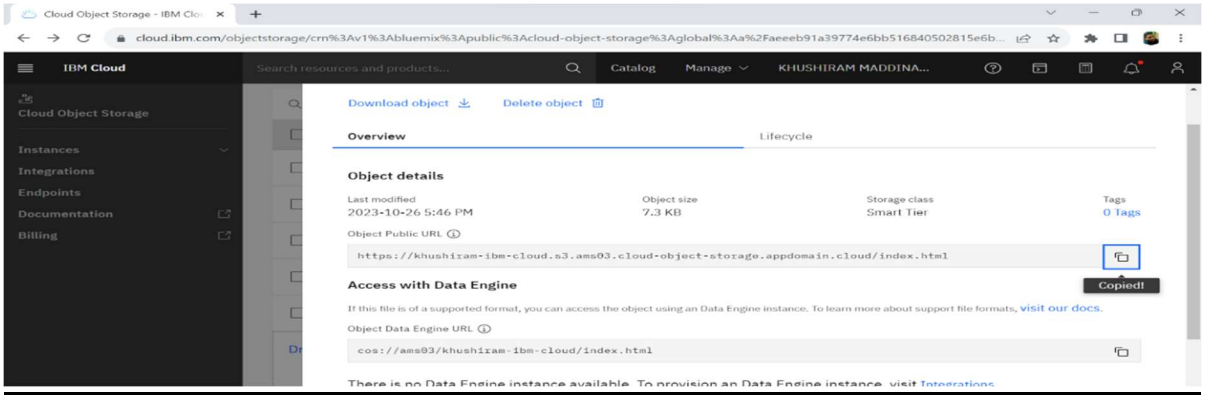
Click on the three dots beside of your website object then click on Access Policies:



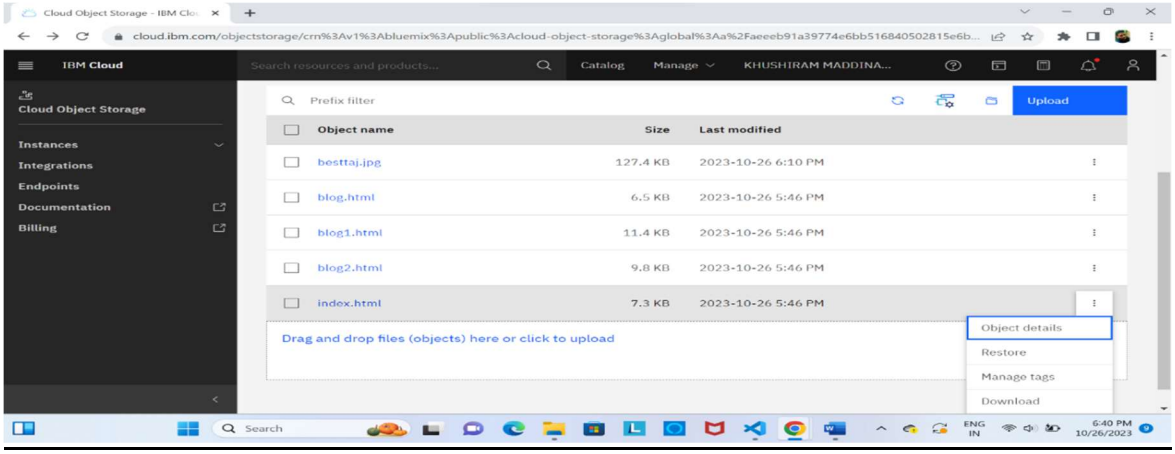
Scroll down click on public access set as object reader :



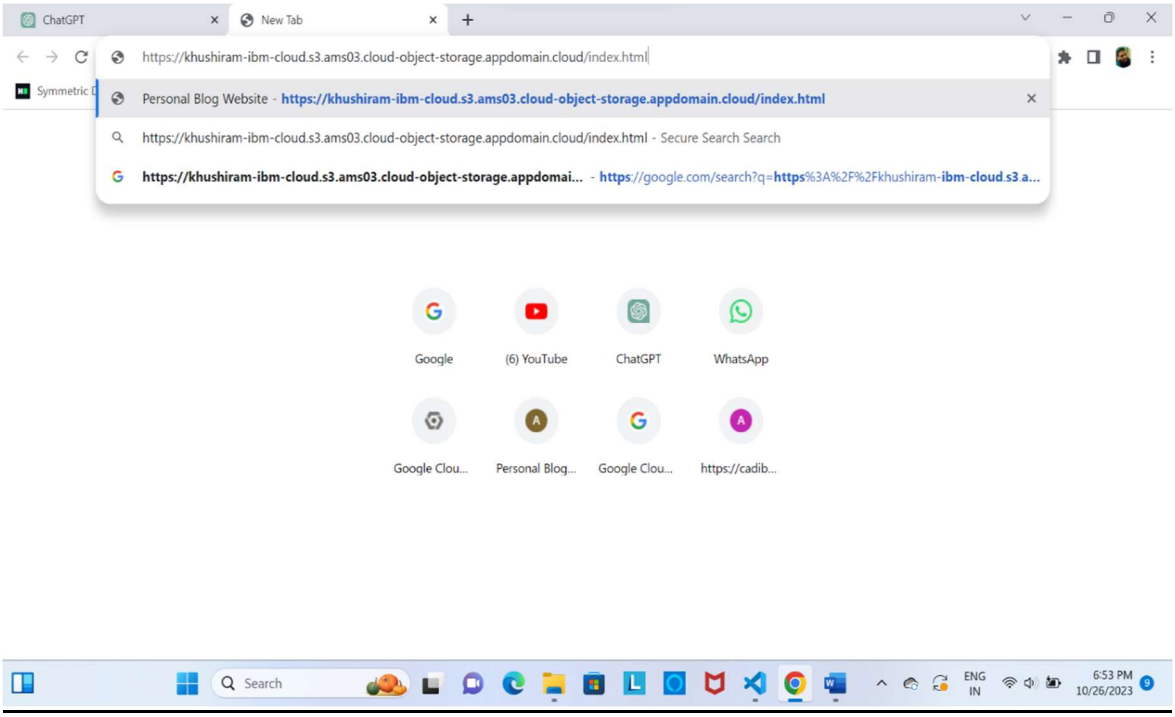
Go to object files click on three dots and click on object details :



You get object public link for public access :



Copy the link and paste it into new tab then click enter :



Finally your website display on browser online.

Conclusion :

Creating a chat application using HTML, CSS, JavaScript, Firebase, and focusing on UI/UX design results in a robust and user-friendly communication tool. By integrating these technologies, the application ensures real-time communication, scalability, and security while providing a clean, intuitive interface that enhances user experience. The use of Firebase allows for seamless data management and instant messaging, making the application reliable and efficient. Future enhancements could include advanced features like multimedia messaging, AI integration, and continuous design optimization, ensuring the application remains modern and user-centric. This project exemplifies the successful combination of technology and design to deliver an effective and engaging digital solution.

-x-