**CLOUD TALK**

**Team Member’s Names and Roles:**

1. Venkata Sai Sriram Valluri – 16352224

**Roles and Responsibilities:**

**Web App Development**: Developed the Web Application for user to have an account and handled login screen and register screen.

**Model Design and Framework Selection**: React is used to build the frontend UI and manage the component-based architecture.

**Integration with the Firebase database**: Handled the integration of the application with the firebase database.

2. Narendra Mamilla – 16351892

**Roles and Responsibilities**:

**Web App Development**: Developed the Web Application for user to have an account and handled inside component of group screen.

**Authentication part for Firebase database:** Handled the authentication part for the database.

3. Venkata Subbarao Gunturu – 16355321

**Roles and Responsibilities**:

**Web App Development**: Developed the Web Application for user to have an account and handled the ﬂow between the users.

**Testing the entire application**: Responsible for checking the end-to-end ﬂow of the Application.

4. Siva Sai Maturi – 16341232

**Roles and Responsibilities**:

**Web App Development**: Developed the Web Application for user to have an account and handled groups related screen.

**Deploying in GitHub**: Deploying the project to GitHub.

**Group Member’s Picture:**



**Motivation / Purpose:**

The motivation on and purpose behind developing a cloud-based messaging service application lie in the ever-evolving landscape of modern Communication. With the increasing demand for real-me, eﬃcient, and ﬂexible interaction, such a platform becomes essential. It aims to provide users, whether individuals or organizations, with a seamless and accessible means of connecting, sharing information, and collaborating from anywhere in the world. This technology-driven solution seeks to transcend geographical boundaries, enhance convenience, and boost productivity, making it an indispensable tool for both personal and professional communication needs. By harnessing the power of cloud-based infrastructure, the application aims to deliver reliability, scalability, and a user-centric experience, ultimately bridging the gaps in our connected world.

**Architecture:**

React JS

UI Handler

Login Page

Chat Group Page

Sign up page

New Chat

Chat Screen

API Handler

User

Chat Group

Login

Signup

User Details

Data storage

Data Retrieve

**Features:**

Below are the main tasks need to be performed to achieve the final outcome:

* User Registration and Account Creation
* User Login
* Chat Group Creation
* Real-Time Messaging
* Firebase Database Integration

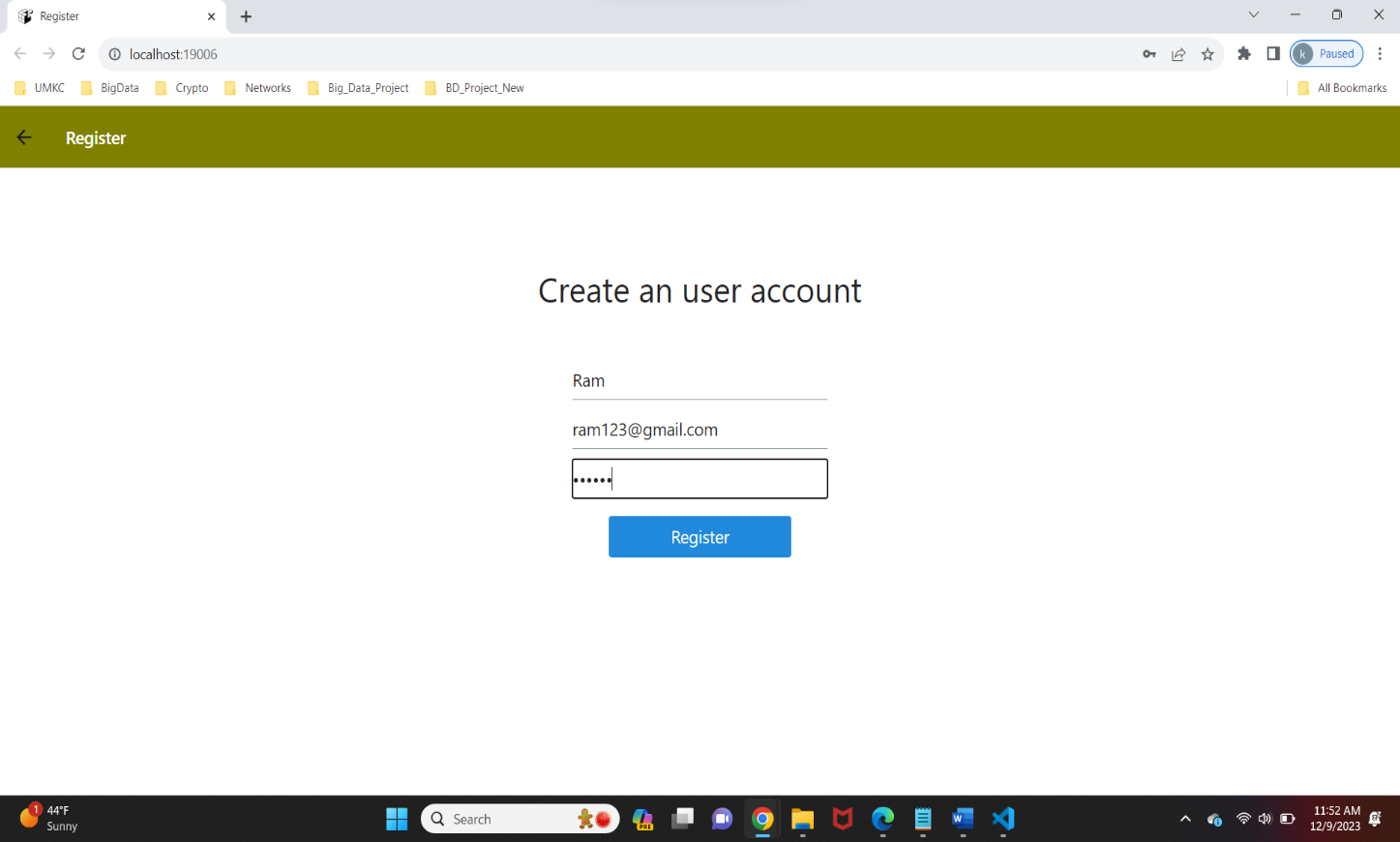
**Task A: User Registration and Account creation and User Login:**

In this task we have created two different screens by using React JS.

**1.Register Screen**:

* A web page was crafted with React JS, prompting users to input their username, email, and password.
* Upon clicking the sign-up (register) button, the provided values are securely stored in the Firebase Database, creating a new user profile.

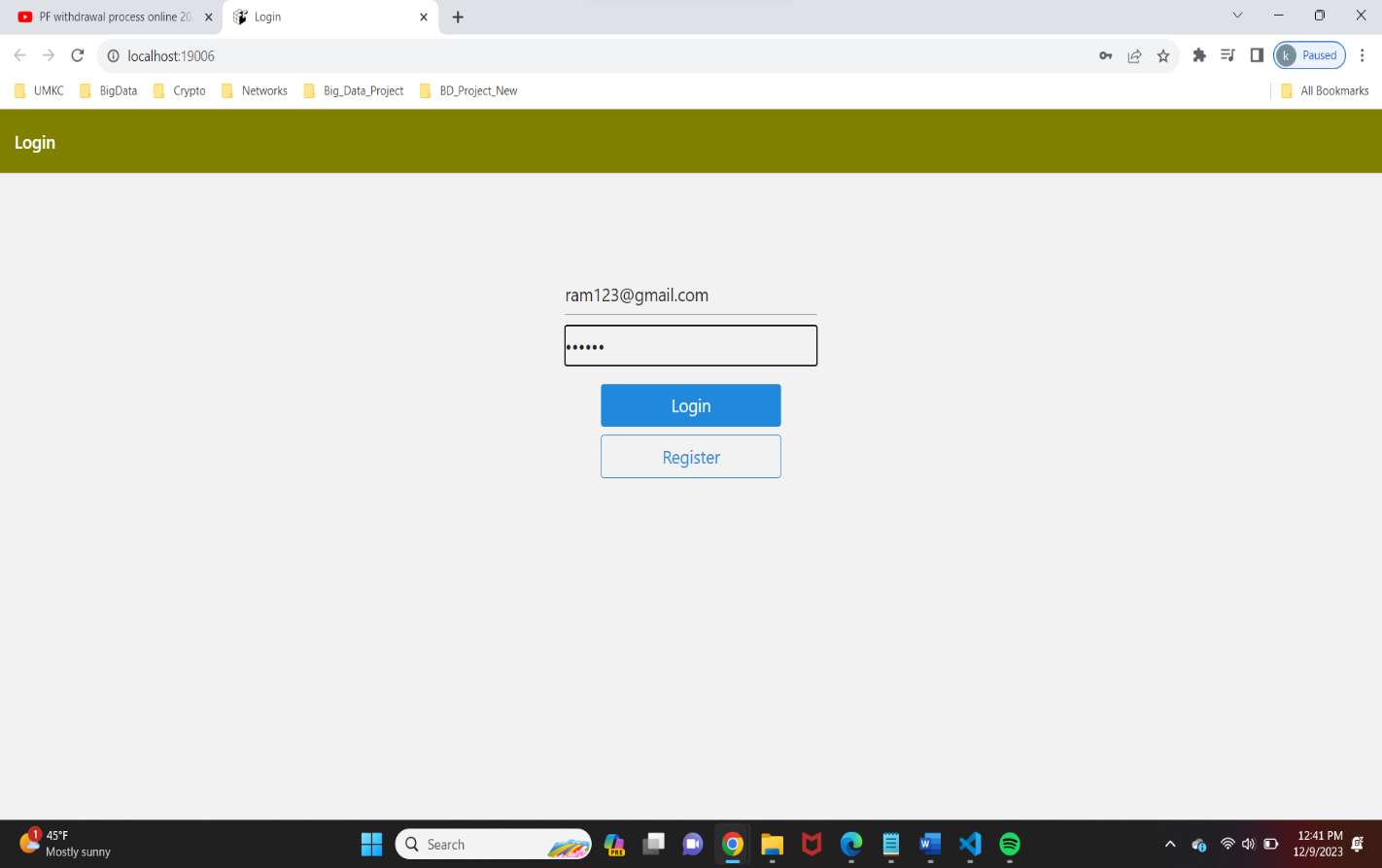
Here we have created an account with user “RAM”



**2.Login Screen:**

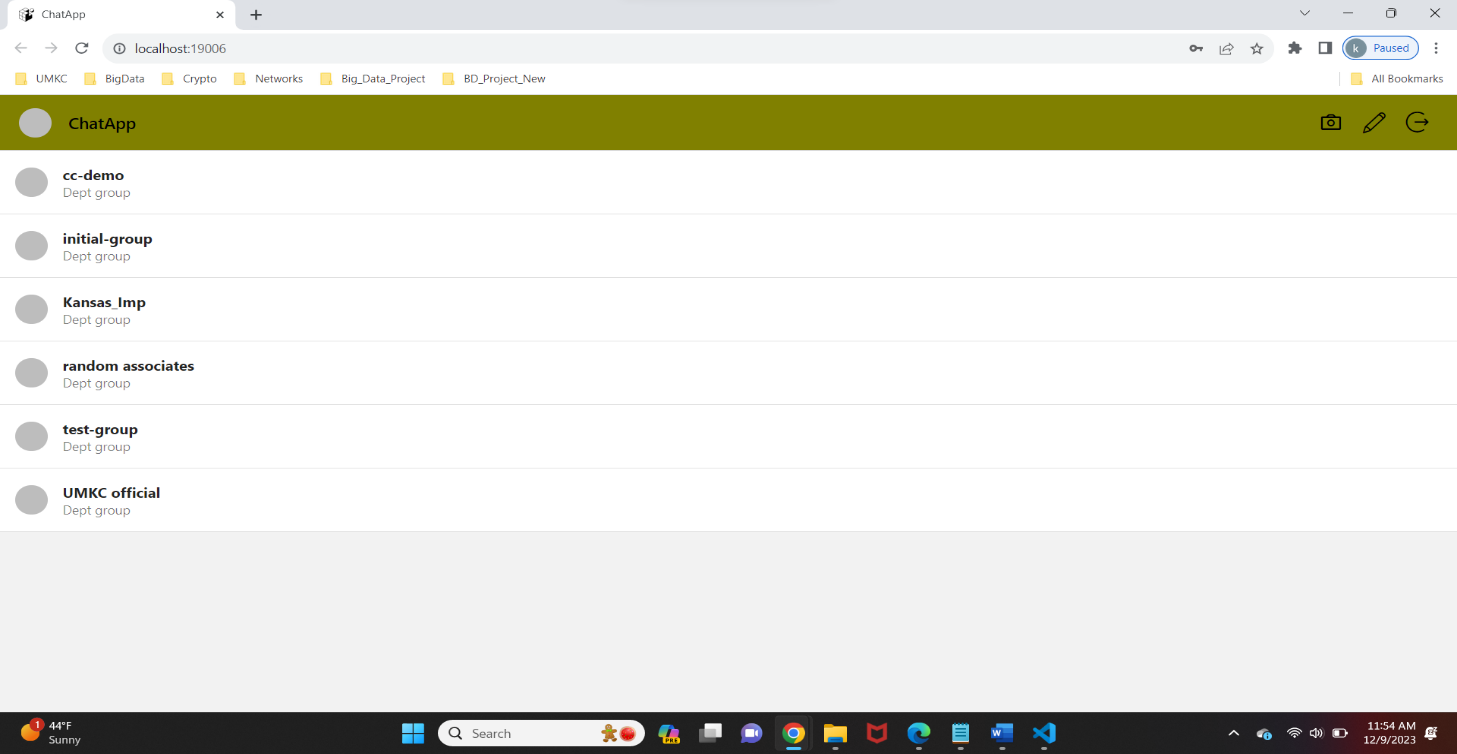
* Once users complete the account registration, they can utilize this screen to log in to the web application.
* To log in, users must provide their email and password, which undergo validation against the Firebase database.

Now we are logging into application using RAM profile/credentials.

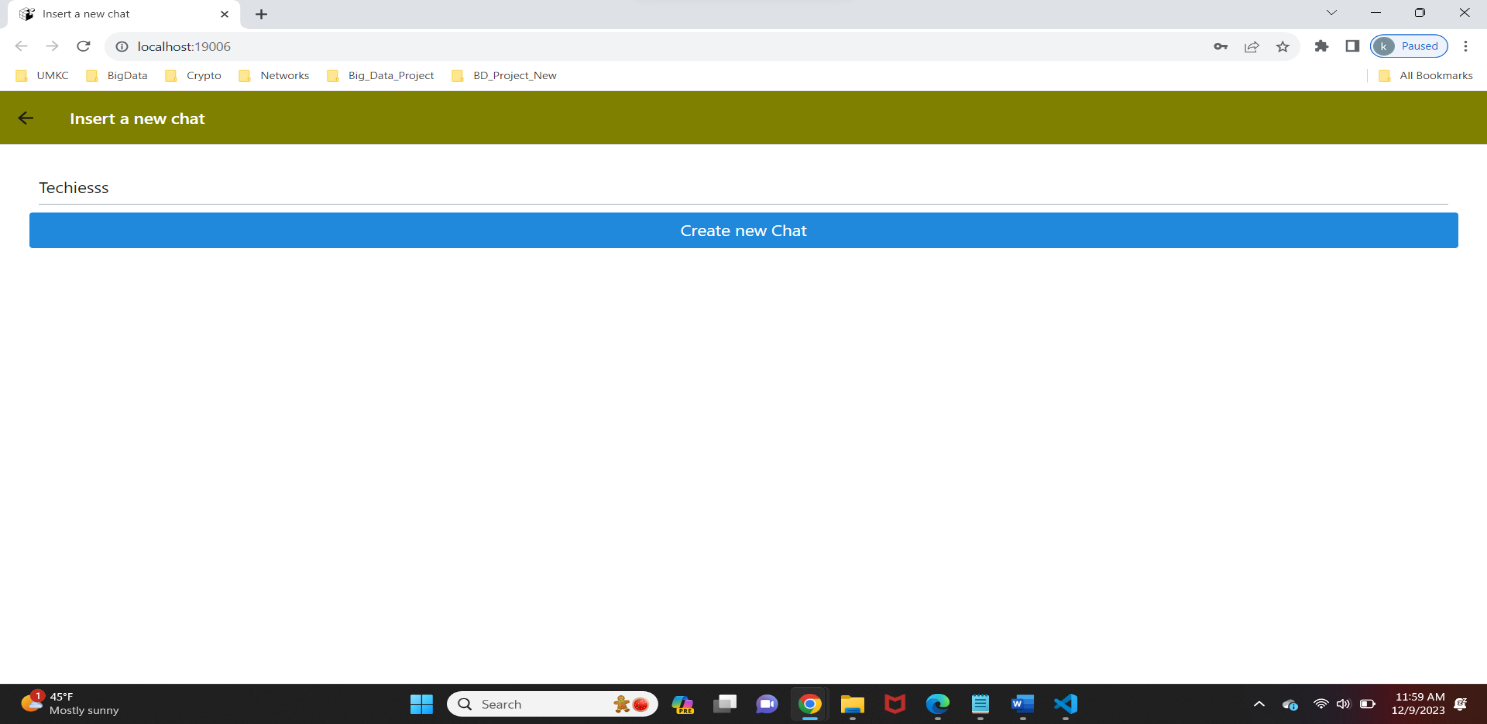


**Task B: Chat Group Creation:**

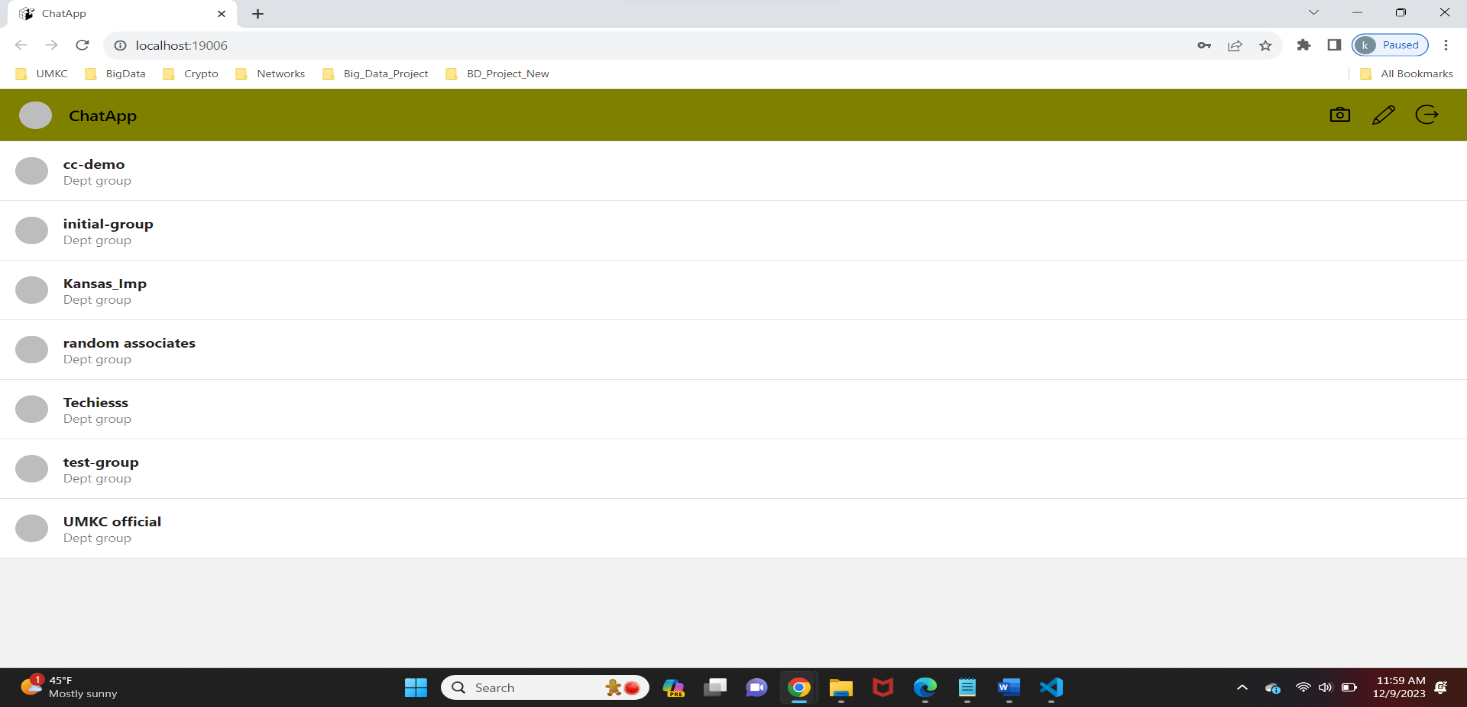
* Implementation: The home screen has been created, showcasing a list of public group names for sending and receiving messages.
* If the validation is successful, users are redirected to the home screen. Below is the Home Screen of our Chat Application, where we can see the List of All Public Groups.



* To add new chat group, we can click on add symbol which is there at top right corner.



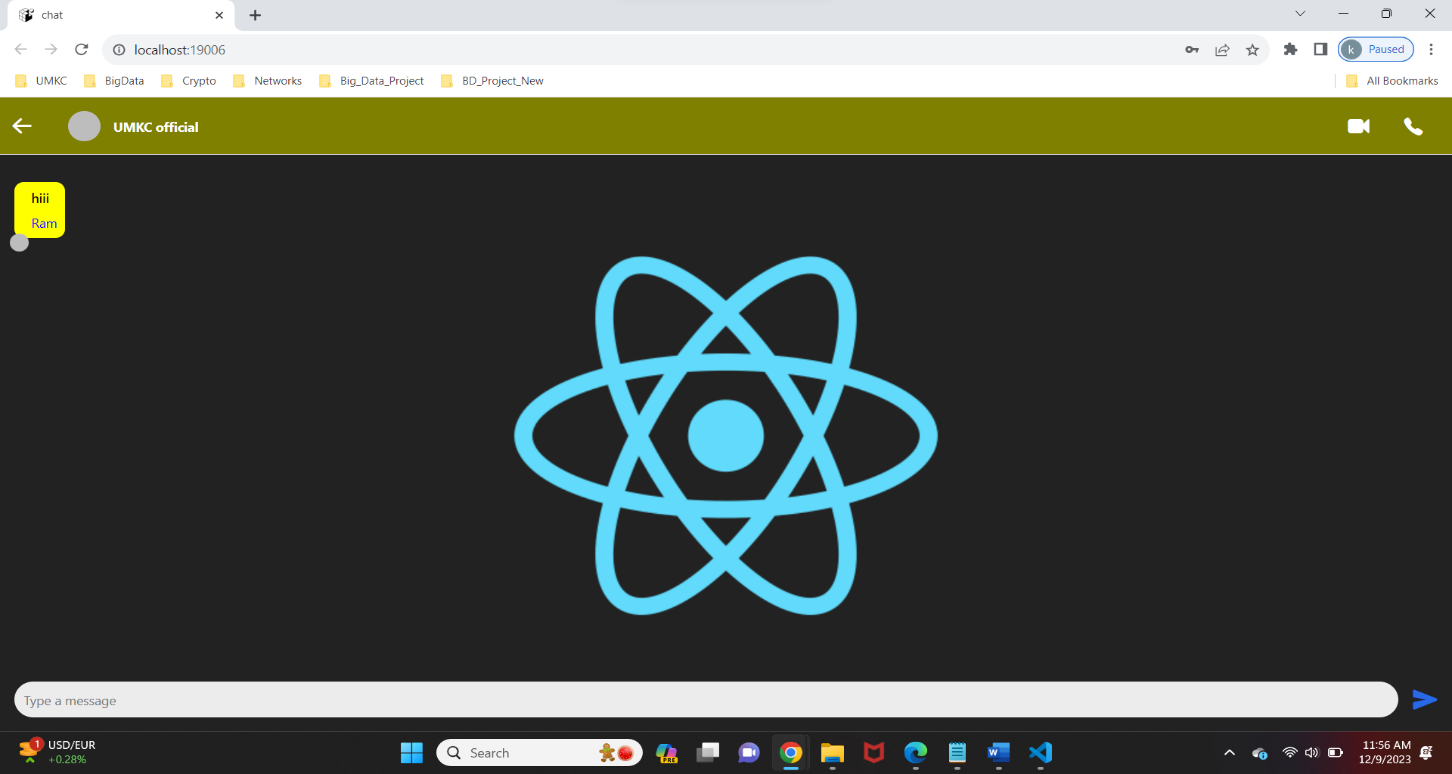
Now we have added the New Chat group “Techiesss”, which was added to the list of existing chat groups.



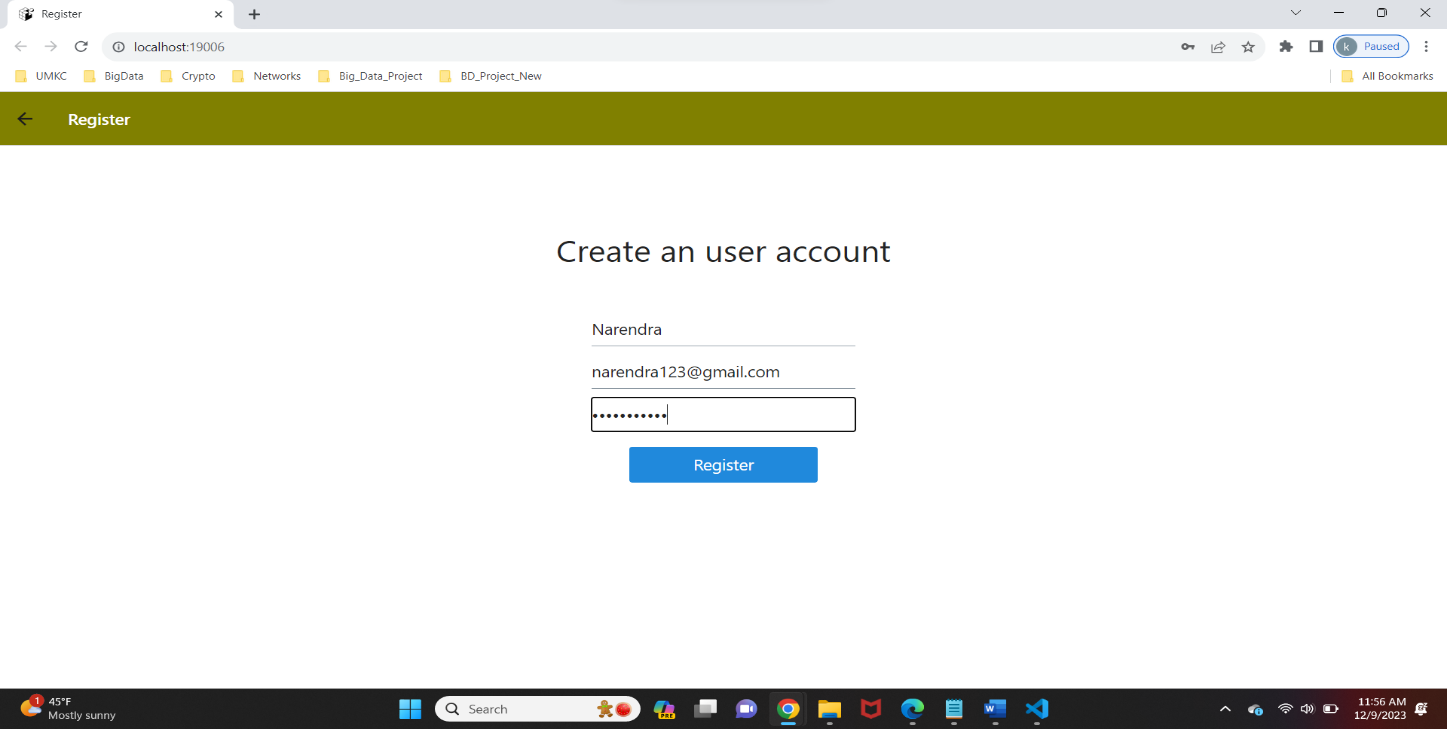
**Task C: Real-Time Messaging & Communication:**

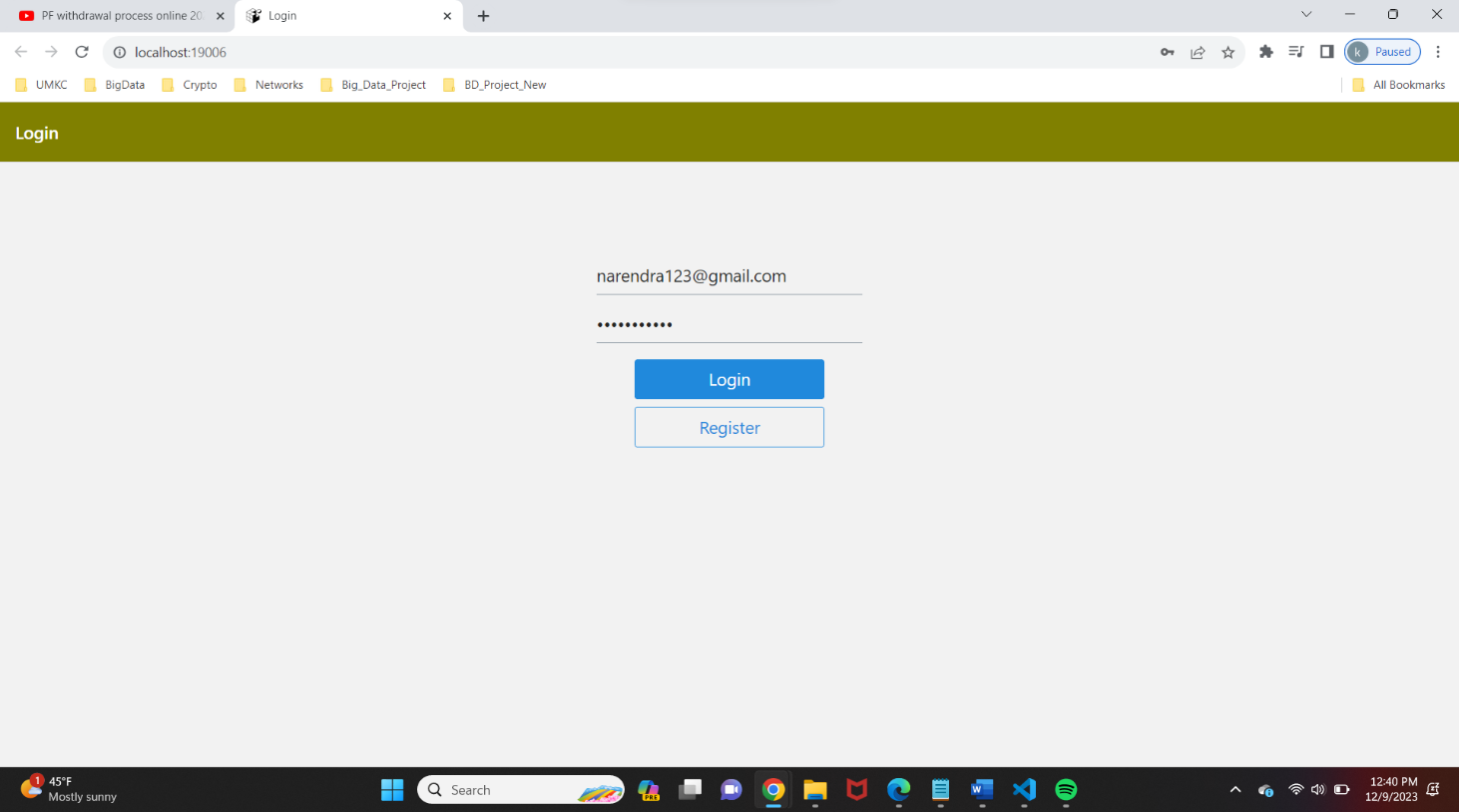
We have implemented a real-time messaging system, introducing a dedicated chat screen. This screen will empower users to engage in live conversations, sending and receiving messages in real-time. As users log into the web application, they will gain access to existing messages within the group, facilitating seamless communication. Users will also have the capability to contribute by sending their own messages, fostering an interactive and dynamic user experience.

1.Once after successful Login, RAM went to the Chat group “UMKC\_official” and entered a message “Hii”.



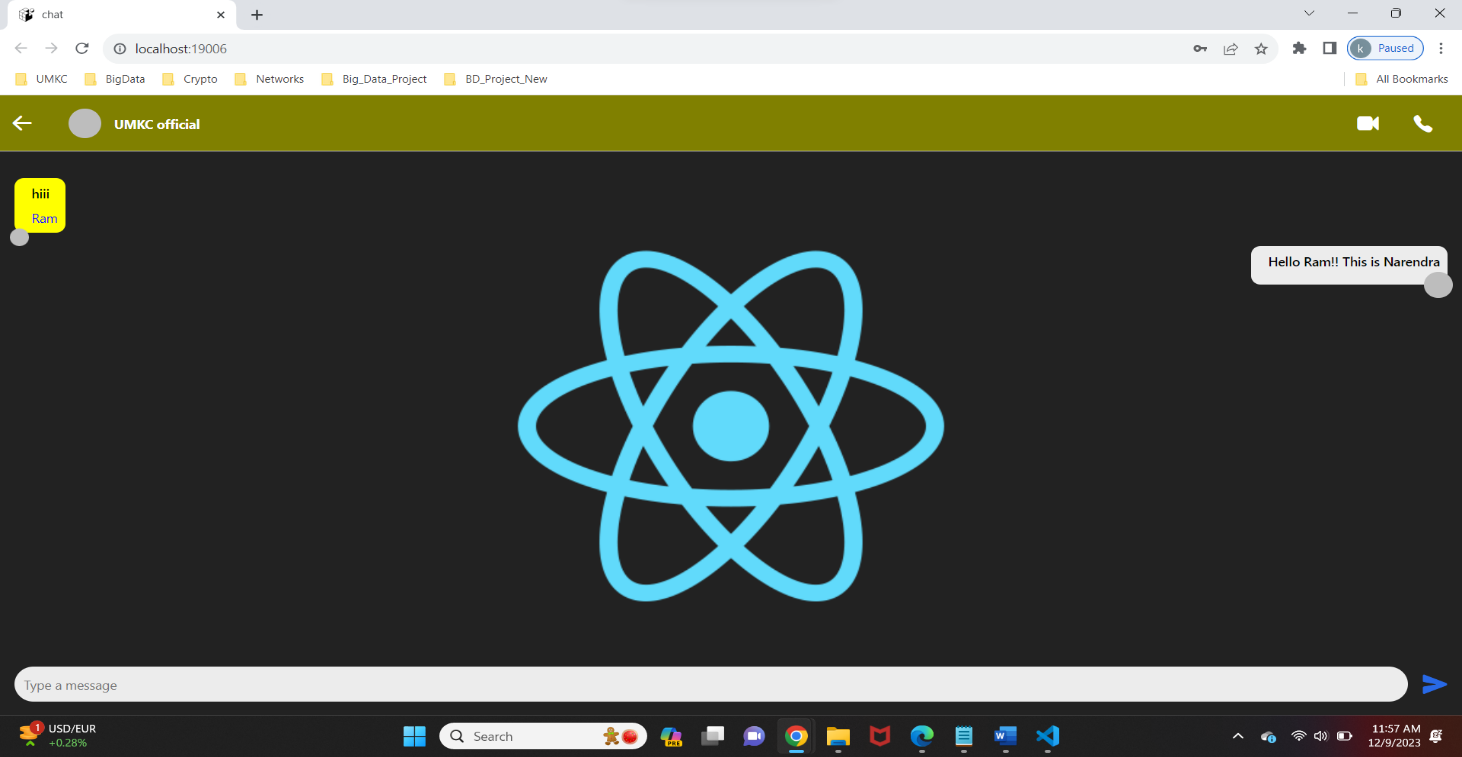
Now another User “Narendra” created the new account and Logged into the application.





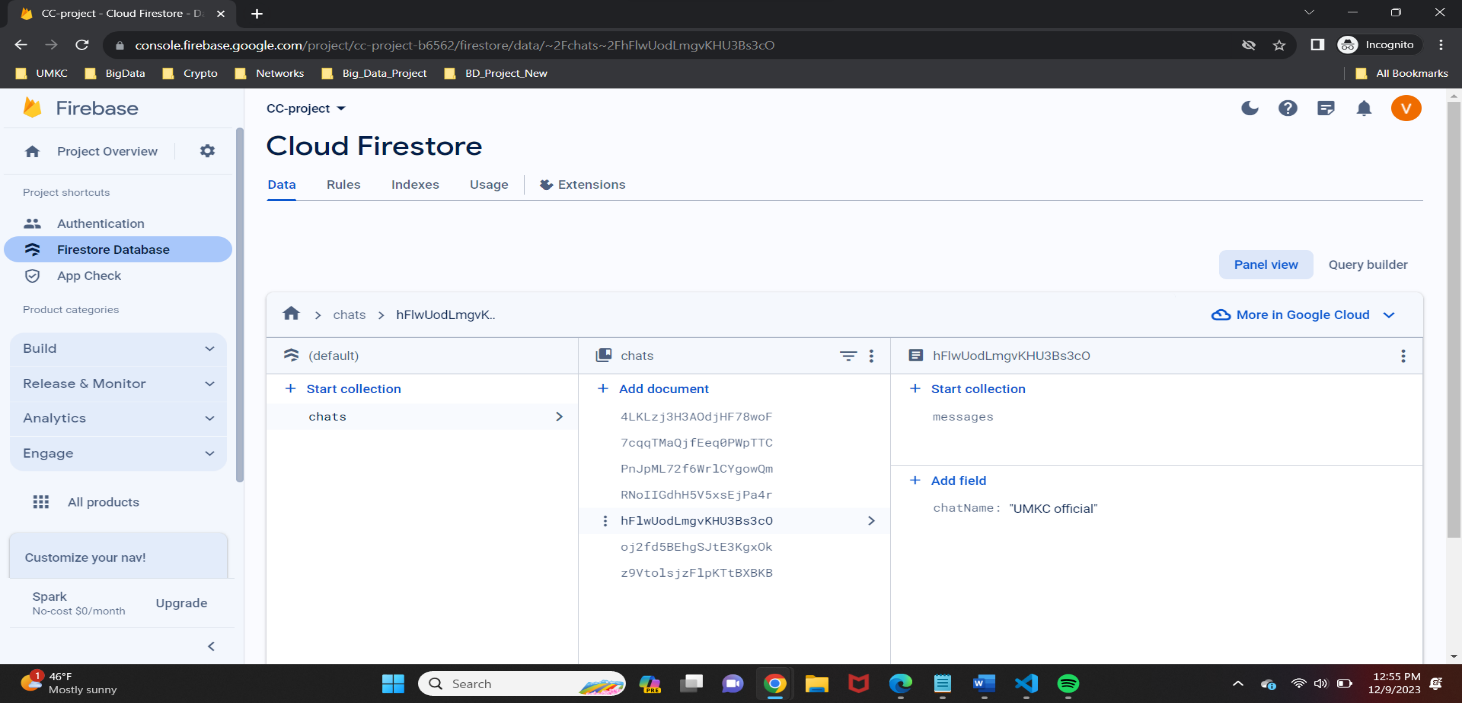
Now the User “Narendra” opened the public chat group “UMKC\_official” and received the message “Hii” published by user “RAM”.

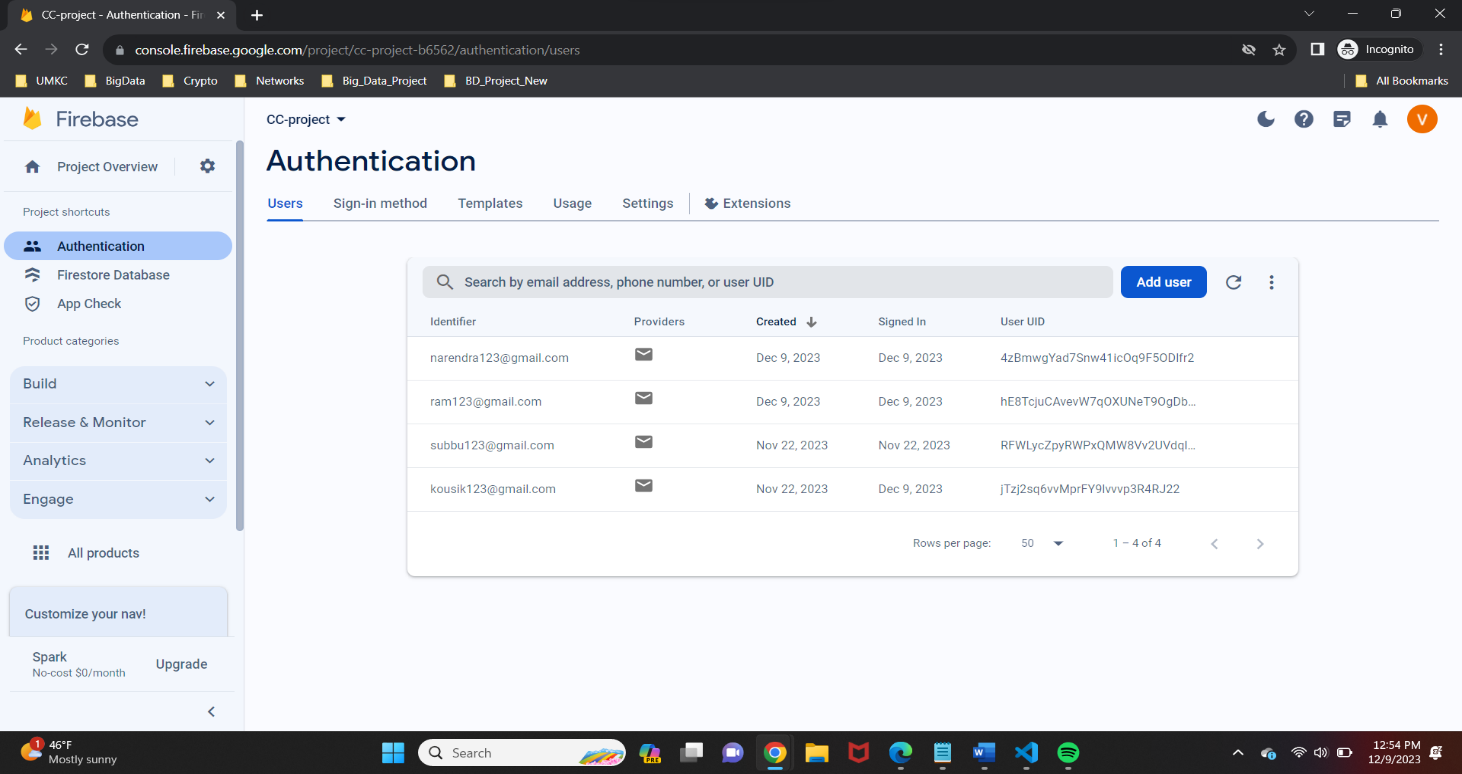
Then Replied back to the RAM’s Message.



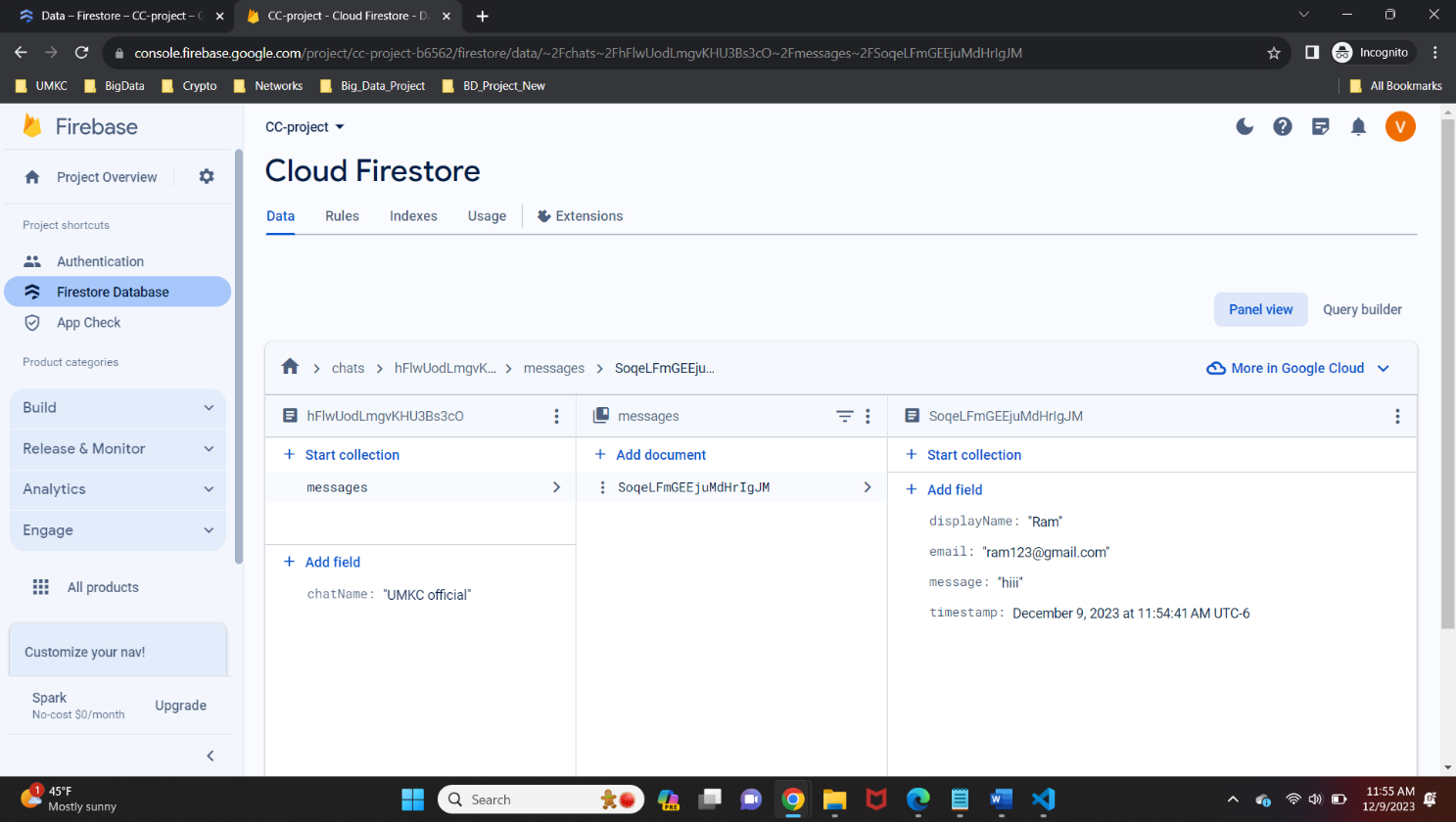
**Task D: Firebase Database Integration:**

This phase represents a pivotal step in our project as it involves integrating our web application with Firebase Database. To initiate this integration, we created a Firebase account and configured the settings to establish a connection. Subsequently, we executed CRUD operations, starting with user account registration (create) and progressing to user account validation (read). The crux of this step involves storing all messages and chat group data within the Firebase database, creating a foundation for real-time communication and data storage.

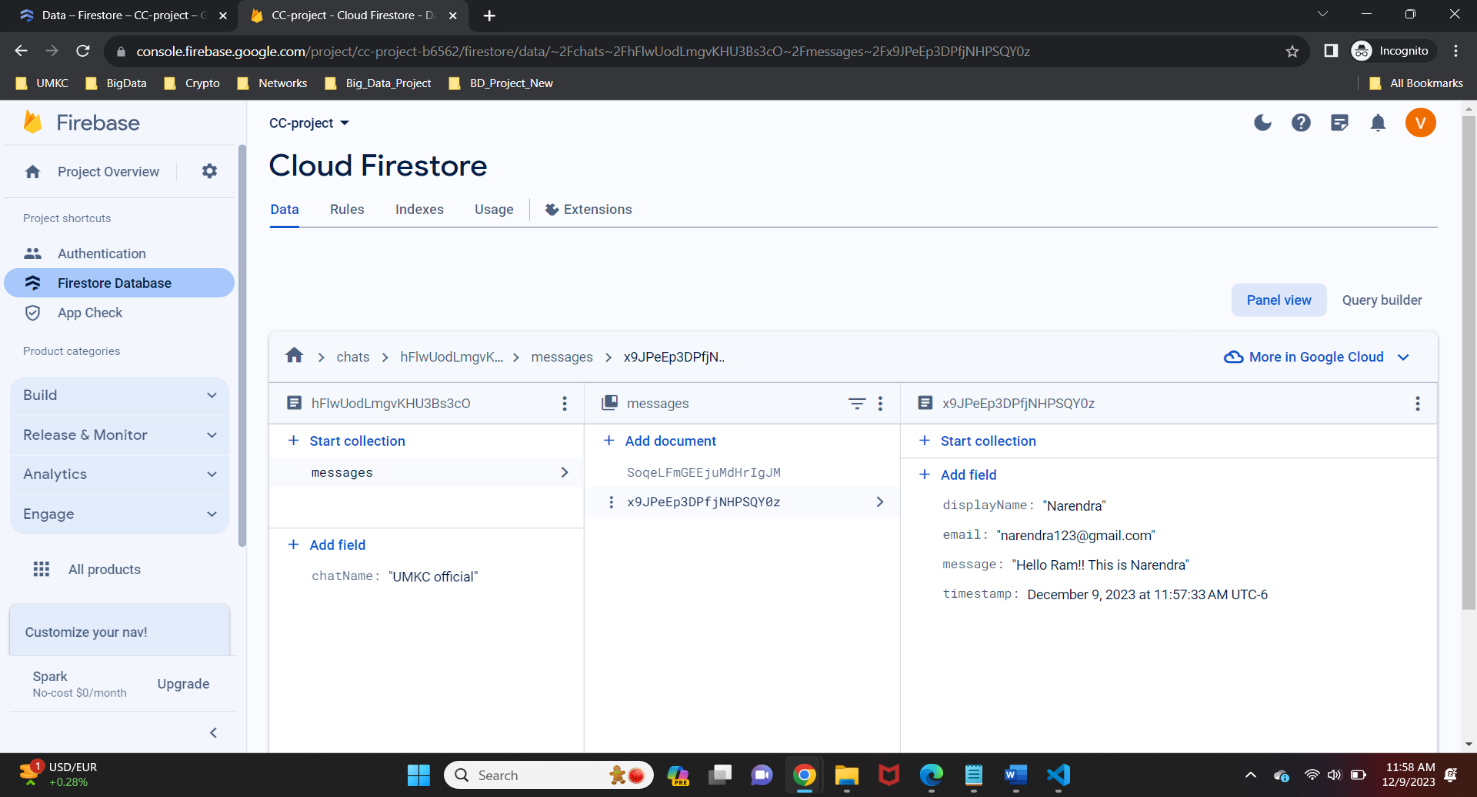




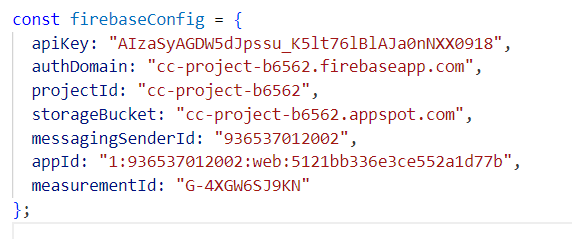
When RAM sends a message “Hii”, It will be stored in the Firebase Database shown as below.



When New user opens this chat group, all the previous conversations are retrieved from the database.



Below configuration file is required to integrate front-end react application with firebase database.



Here we have configured the firebase with corresponding API Key, Auth Domain which we will get while creating the new Firebase Project.

**Conclusion:**Similarly, the conceptual execution of the Chat App demonstrates the effectiveness of cloud-based technologies, specifically Firebase and React Native, in constructing a contemporary chat platform. Through the utilization of real-time data synchronization, user verification, and cloud storage, the application facilitates smooth communication and teamwork within entities such as UMKC school. Incorporating Firebase's Firestore database ensures expandable and efficient data storage, guaranteeing dependable access to chat conversations, user particulars, and group information. The future prospects for the app encompass heightened security measures, the introduction of push notifications, and integration with additional cloud services. Overall, the Chat App stands as a valuable resource for organizations seeking to streamline communication channels and cultivate collaborative environments.

**Future Work:**The potential avenues for advancing and broadening the Chat App implementation include:

• Enhanced Search Functionality: Introducing advanced search options in chat conversations, allowing users to easily find specific messages or information.

• Performance Enhancement: Continuously improving the app's performance to handle increased user demand, ensuring smooth and responsive communication, even with a large user base.

• Analytics and Insights Implementation: Installing analytics tools to understand user behavior, usage trends, and popular features, enabling data-driven decisions for further enhancements and improvements.

• Strengthening Security: Adding extra features, like encrypting messages, to guarantee secure communication within the app.

• User Photo Integration: Provide the feature for the users to upload and change their photos of the respective accounts.

• Audio and Video Calls Integration: Introducing the improvement like addition of audio and video calls integration for the users in order to provide effective communication.

**Video Describing Outputs:**<https://drive.google.com/file/d/1sEIKTgEpmuBHnZEdQNDLpkGG__EzZPqJ/view?usp=drive_link>

**Codes:**

<https://github.com/maturisivasai/CloudTalk_CCproject>