CSCI 460 - Capstone

Upama Thapa Magar
Instructor - Prof. Darren Seifert
Spring 2023

bEvents

Your one stop app to All Beaver Events!

motivation

i wish more students show up to our event...

i really want to go to this event but i dont have a ride...

i wish i had a friend to go to this event with...

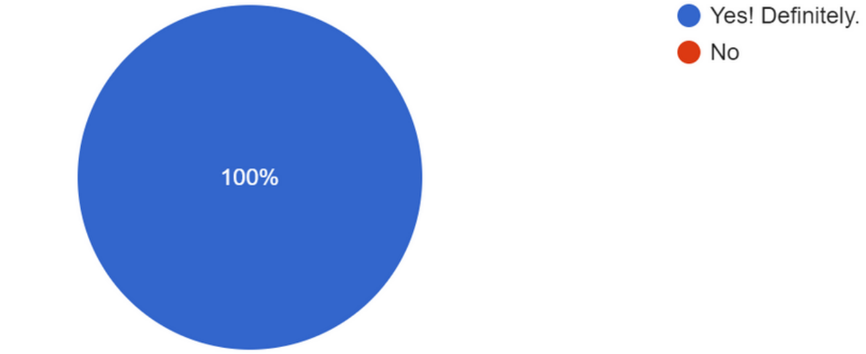
Do you guys think students would come to this event?...

market Study

Survey

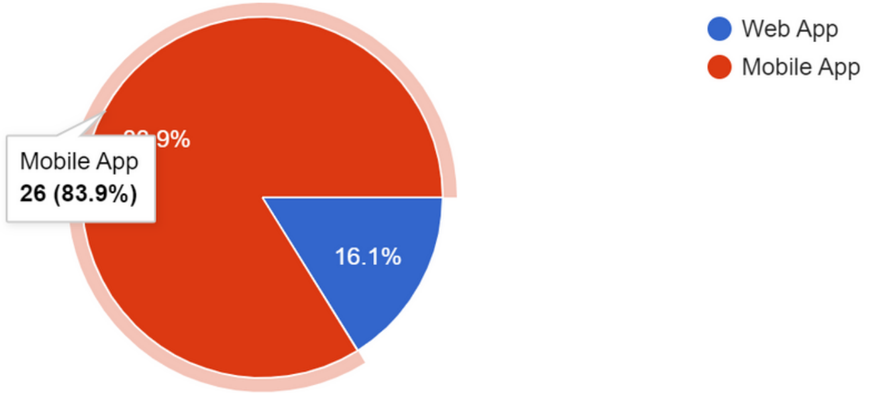
If we had an app like shown below, would you use it?

31 responses



Would you like it as a web app or a mobile app?

31 responses



Beaver Fever

5/41

THE LONG WAIT IS OVER.

Scroll down to see some of the coolest events that we have brought for you!

See more details!

1. Snow Tubing!

See other students review and ratings about this event.

Find a Friend

I need a ride!

Beaver Events

You are just one tap away from discovering amazing MSU Life Events!

Create new Beaver Account

Already Registered? Log in here.

STUDENT ID

Type your student ID

STUDENT EMAIL

Type your student email

PASSWORD

******

Sign up

Some motivating quote related to College Students that keeps changing every minute

Login

Sign in to continue.

STUDENT ID

Type your student ID

PASSWORD

******

Log in

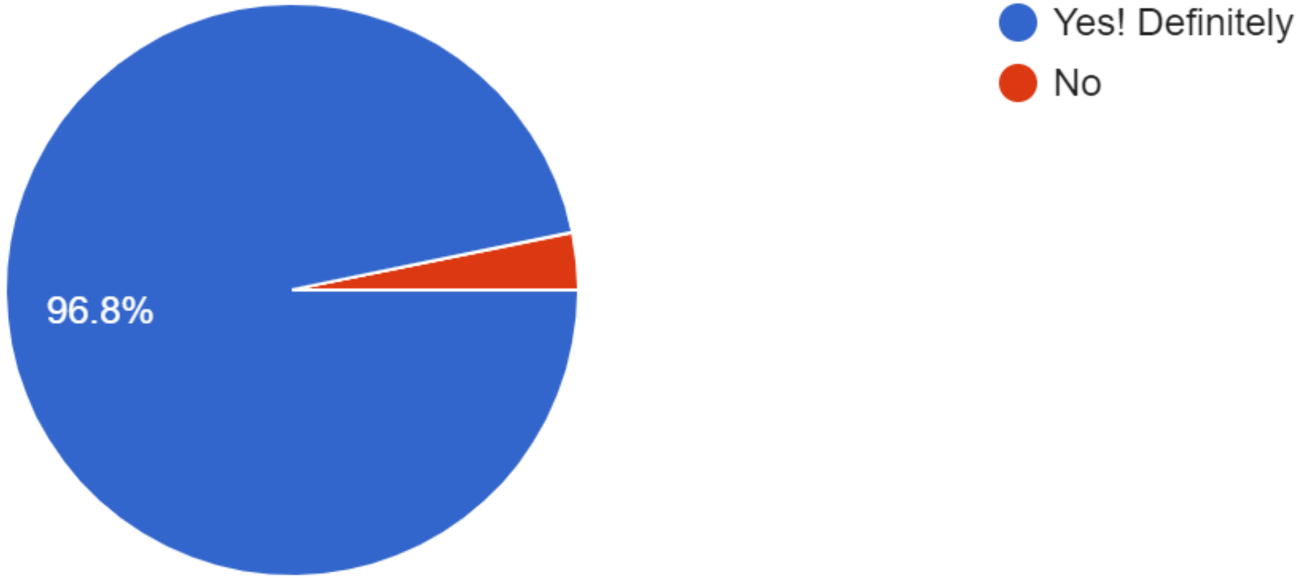
Forgot Password?

Signup !

Do you think having an app or a web app where you can login and view all events and even find a friend to go to an event with or a ride would promote students engagement at MSU?

Copy

31 responses



Current Apps in market -

1. Eventbrite
2. Campus Groups
3. Trello
4. Eventleaf
5. Bizzabo

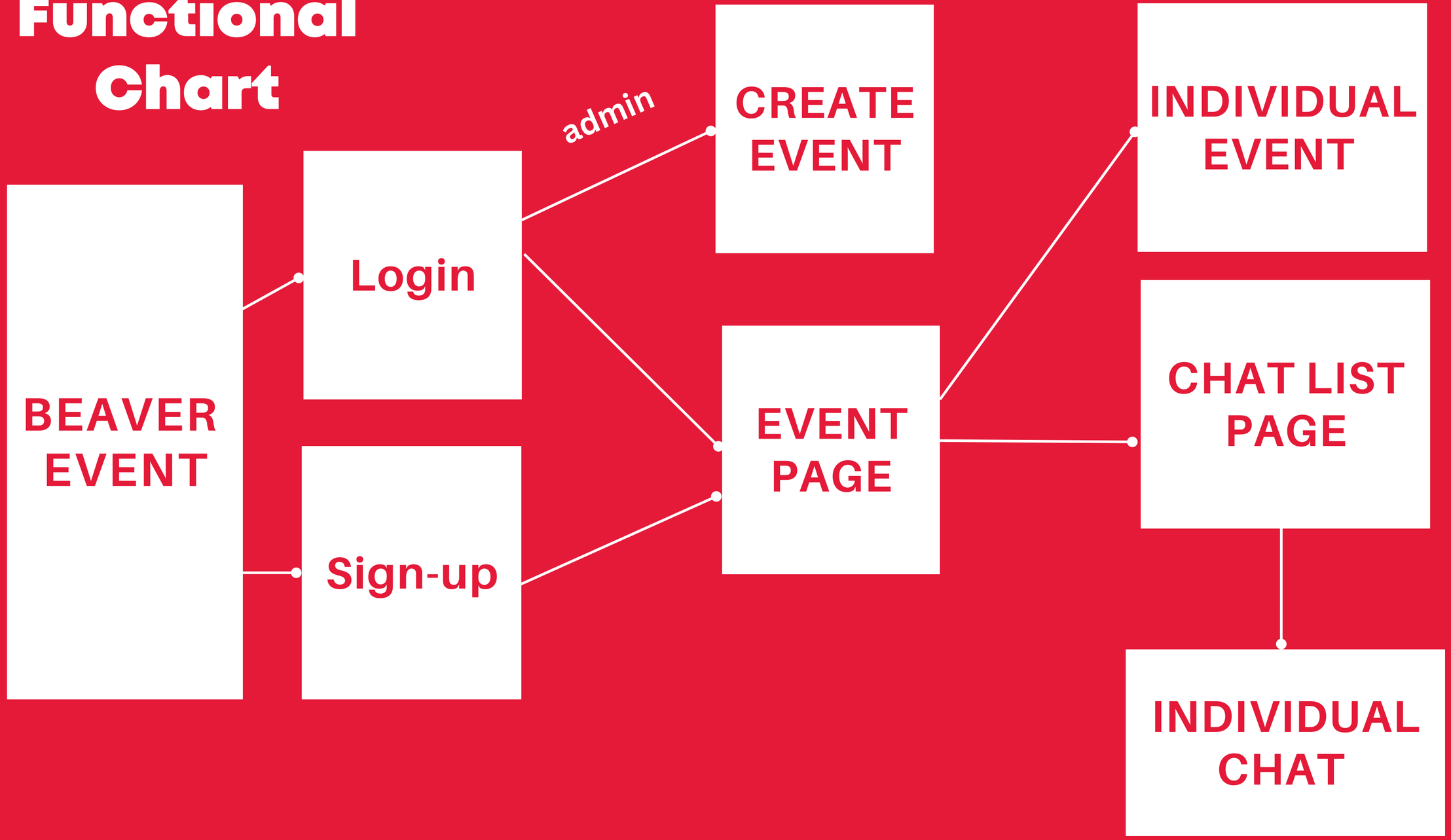
WHY BEAVER EVENTS?

1. **Economical**
2. **Customized feature set**
3. **School brand**

Key Functionalities of the App

1. User authentication -> LOGIN/ SIGNUP
2. Real-time messaging -> IN APP CHAT
3. Chat history -> SAVED PREVIOUS CHAT
4. Event Creation -> CREATE NEW EVENTS
5. Search for events -> EVENTS SEARCH

Functional Chart



Choosing the right programming language

What I chose & why?

1

React Native

2

Firebase Firestore for data storage

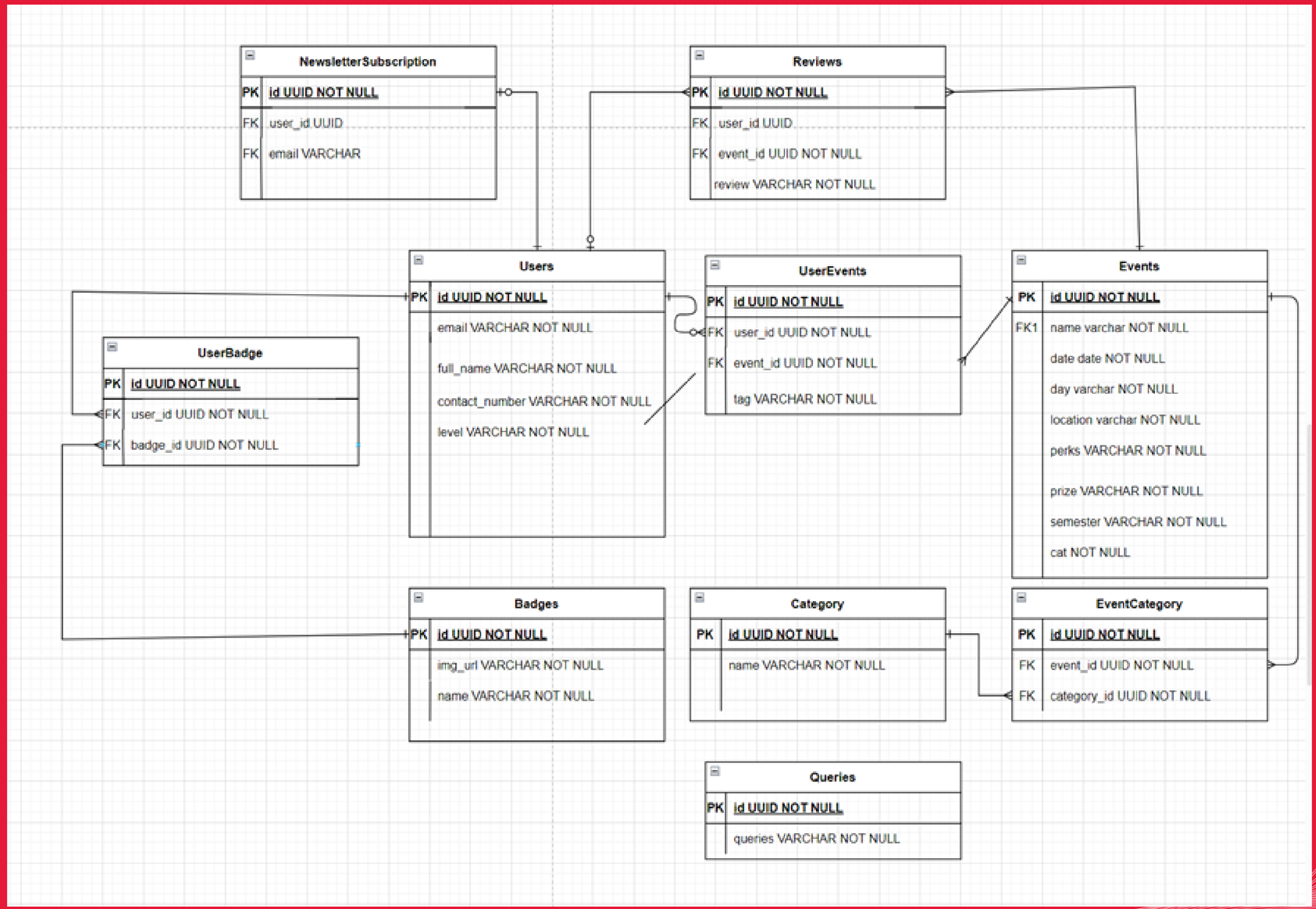
3

Firebase Storage for storing images

4

Gifted Chat messaging UI

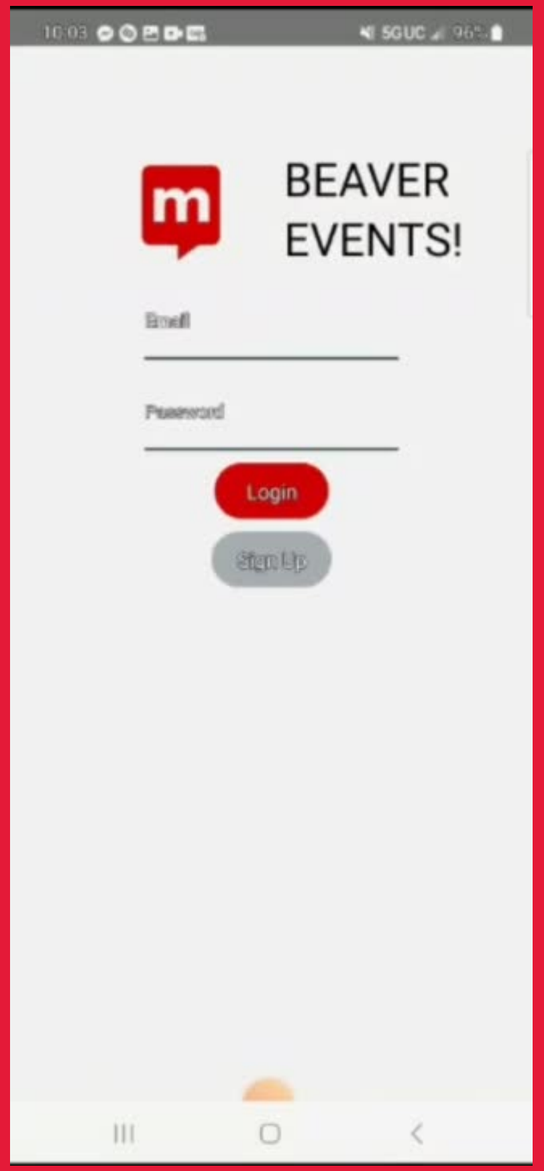
ER diagram-

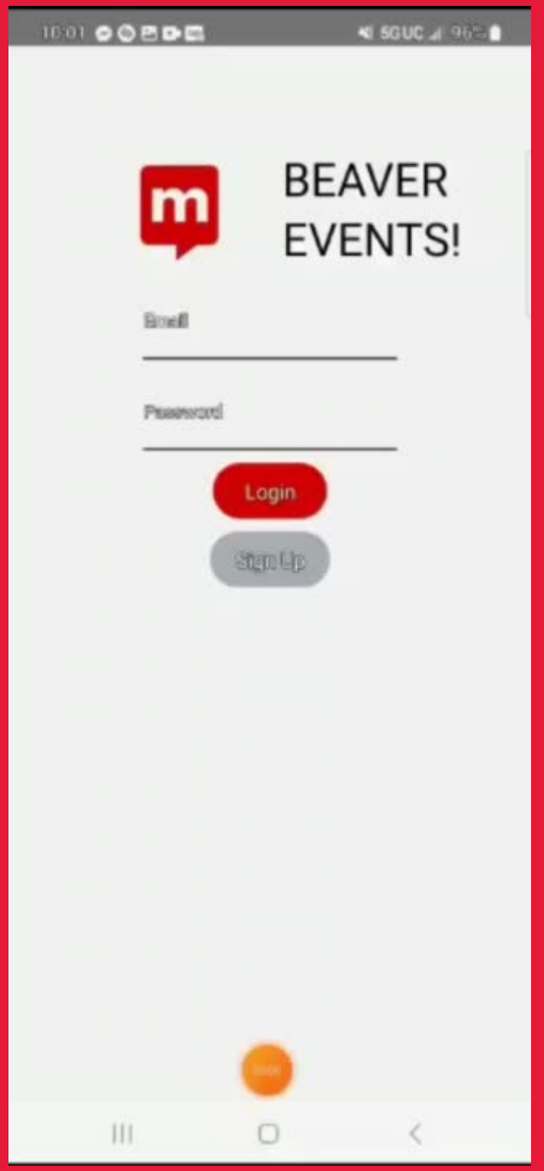

NewsletterSubscription
PK id UUID NOT NULL
FK user_id UUID
FK email VARCHAR
Reviews
PK id UUID NOT NULL
FK user_id UUID
FK event_id UUID NOT NULL
review VARCHAR NOT NULL
Users
PK id UUID NOT NULL
email VARCHAR NOT NULL
full_name VARCHAR NOT NULL
contact_number VARCHAR NOT NULL
level VARCHAR NOT NULL
UserEvents
PK id UUID NOT NULL
FK user_id UUID NOT NULL
FK event_id UUID NOT NULL
tag VARCHAR NOT NULL
Events
PK id UUID NOT NULL
FK1 name varchar NOT NULL
date date NOT NULL
day varchar NOT NULL
location varchar NOT NULL
perks VARCHAR NOT NULL
prize VARCHAR NOT NULL
semester VARCHAR NOT NULL
cat NOT NULL
UserBadge
PK id UUID NOT NULL
FK user_id UUID NOT NULL
FK badge_id UUID NOT NULL
Badges
PK id UUID NOT NULL
img_url VARCHAR NOT NULL
name VARCHAR NOT NULL
Category
PK id UUID NOT NULL
name VARCHAR NOT NULL
EventCategory
PK id UUID NOT NULL
FK event_id UUID NOT NULL
FK category_id UUID NOT NULL
Queries
PK id UUID NOT NULL
queries VARCHAR NOT NULL


life!
Run the app!

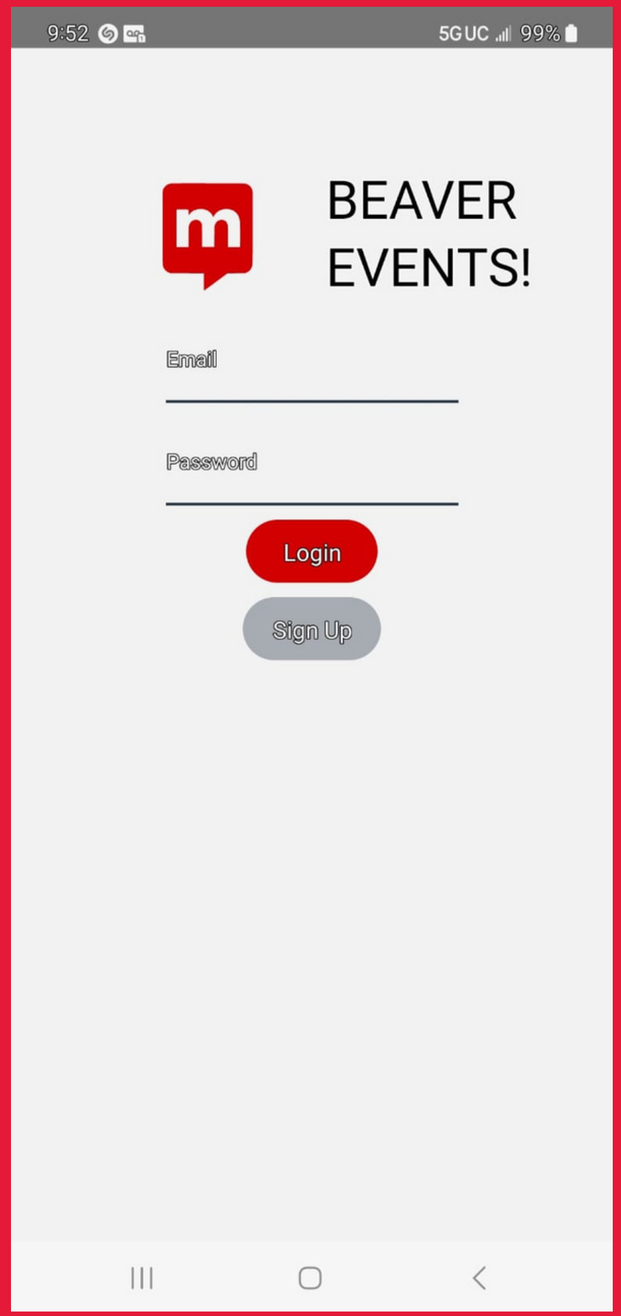
Video

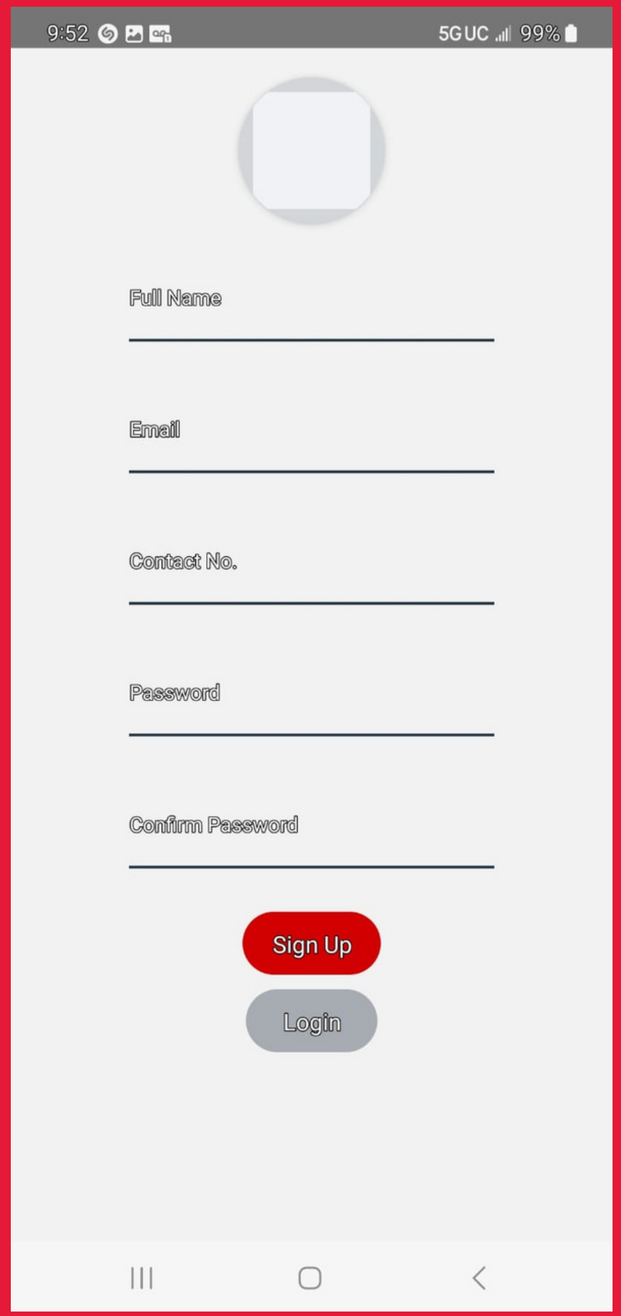

BEAVER
EVENTS!
Email
Password
Login
Sign Up

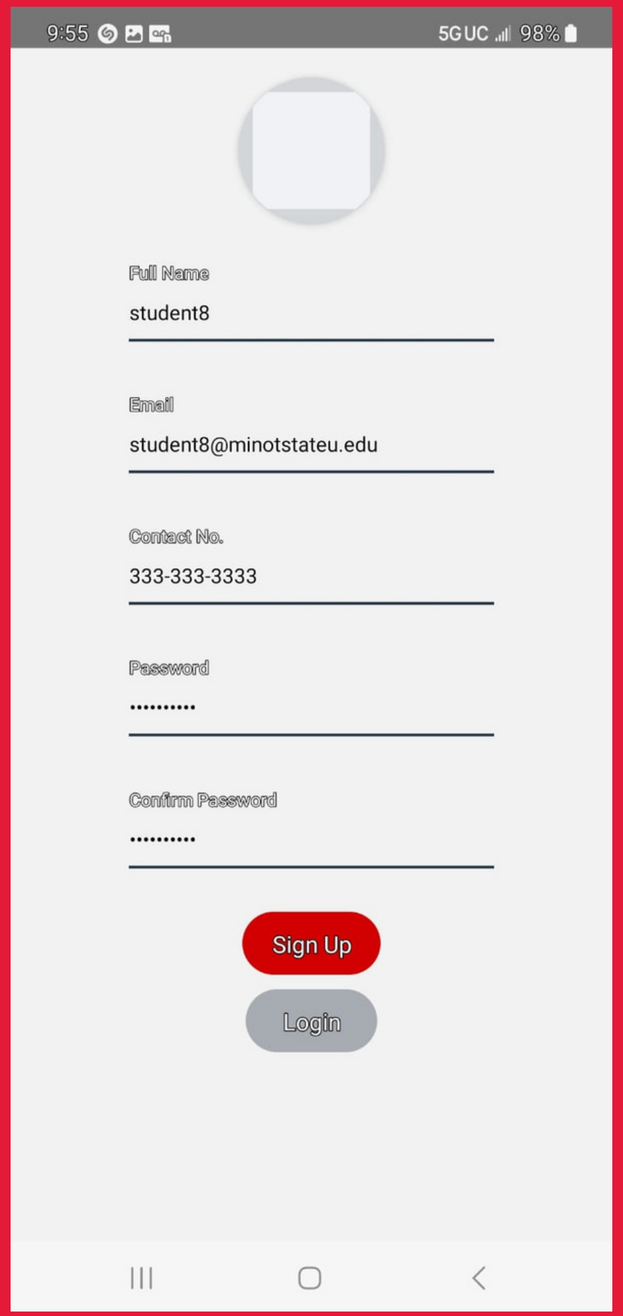

BEAVER
EVENTS!
Email
Password
Login
Sign Up

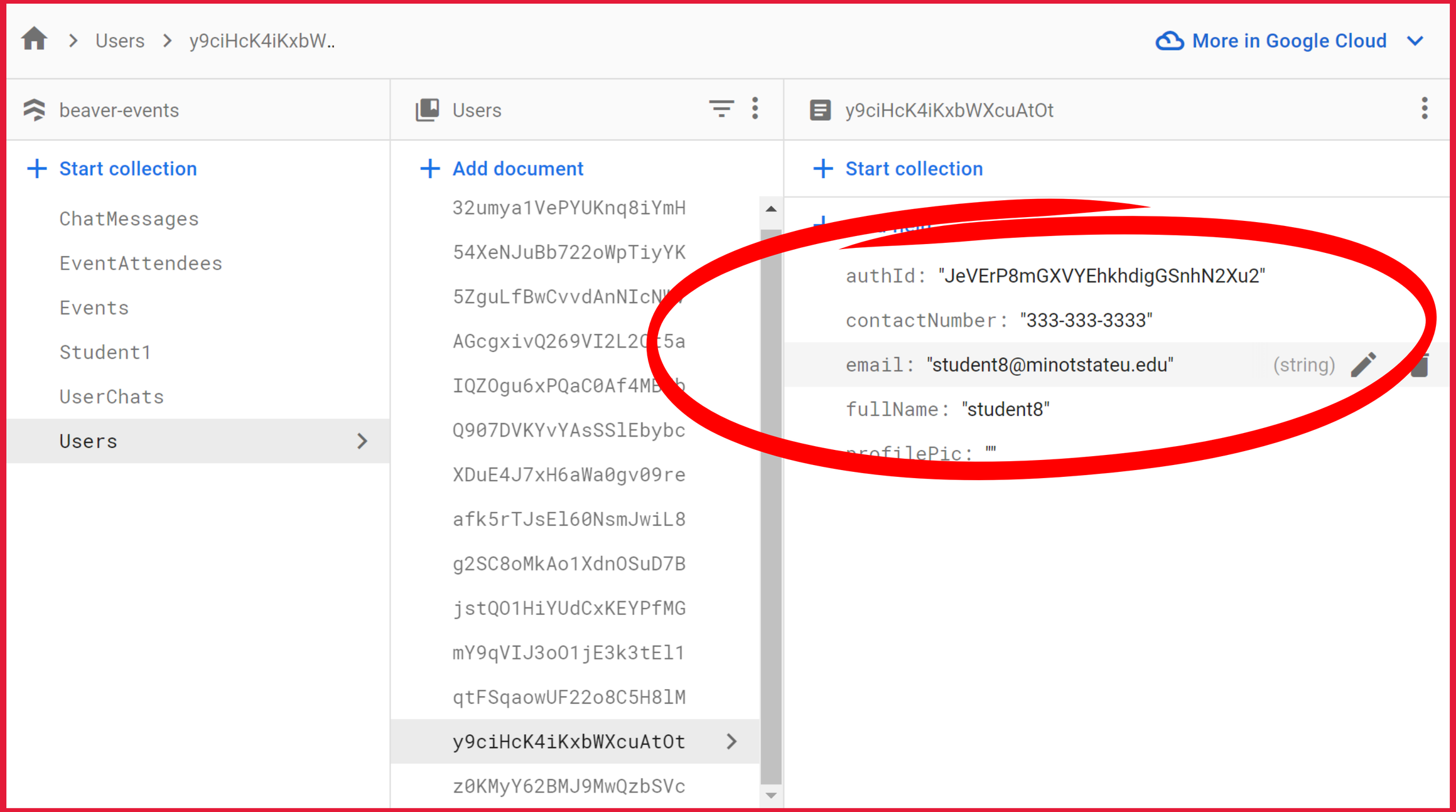

BEAVER
EVENTS!
Email
Password
Login
Sign Up

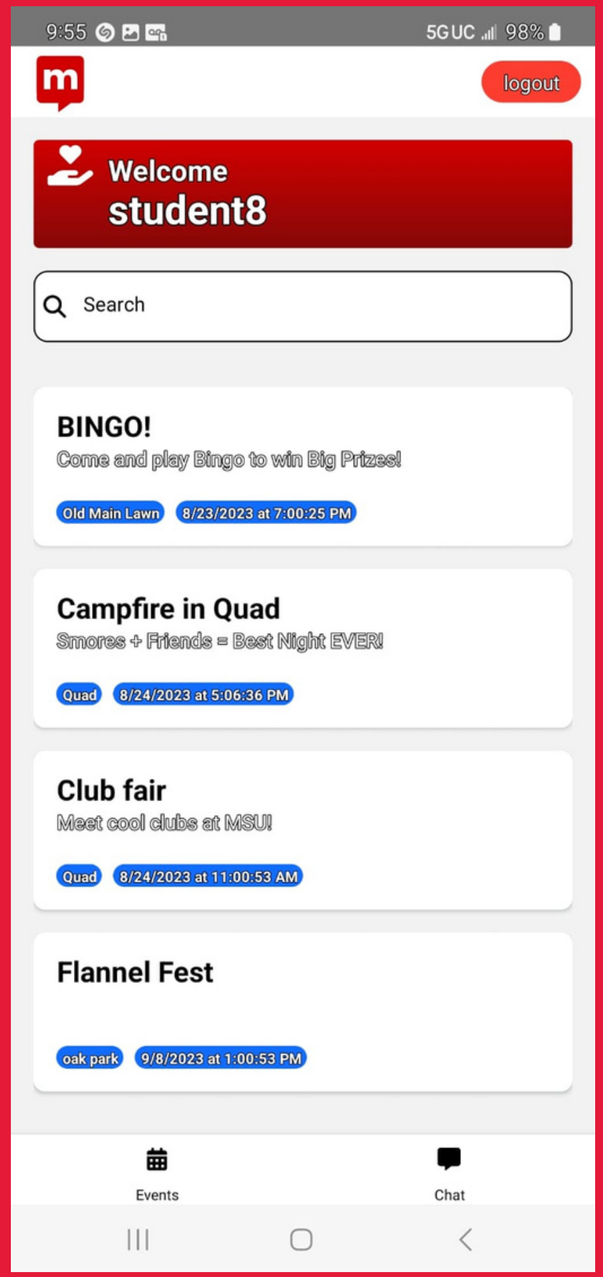
Video

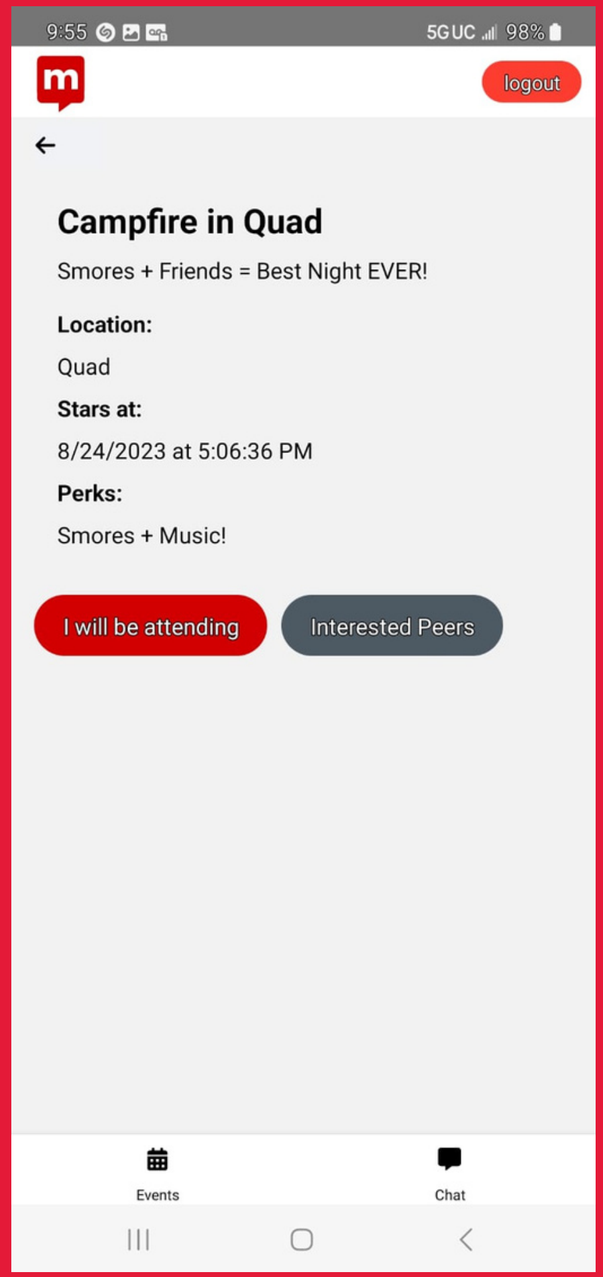

9:52
5GUC 99%
BEAVER EVENTS!
Email
Password
Login
Sign Up

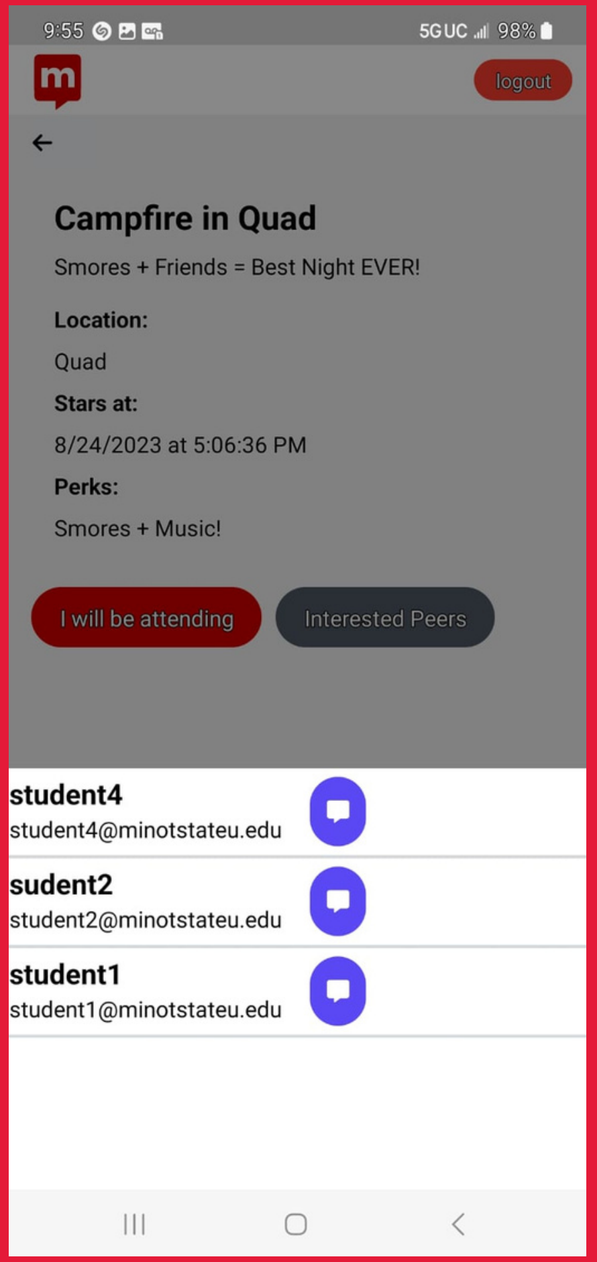

9:52
5GUC 99%
Full Name
Email
Contact No.
Password
Confirm Password
Sign Up
Login

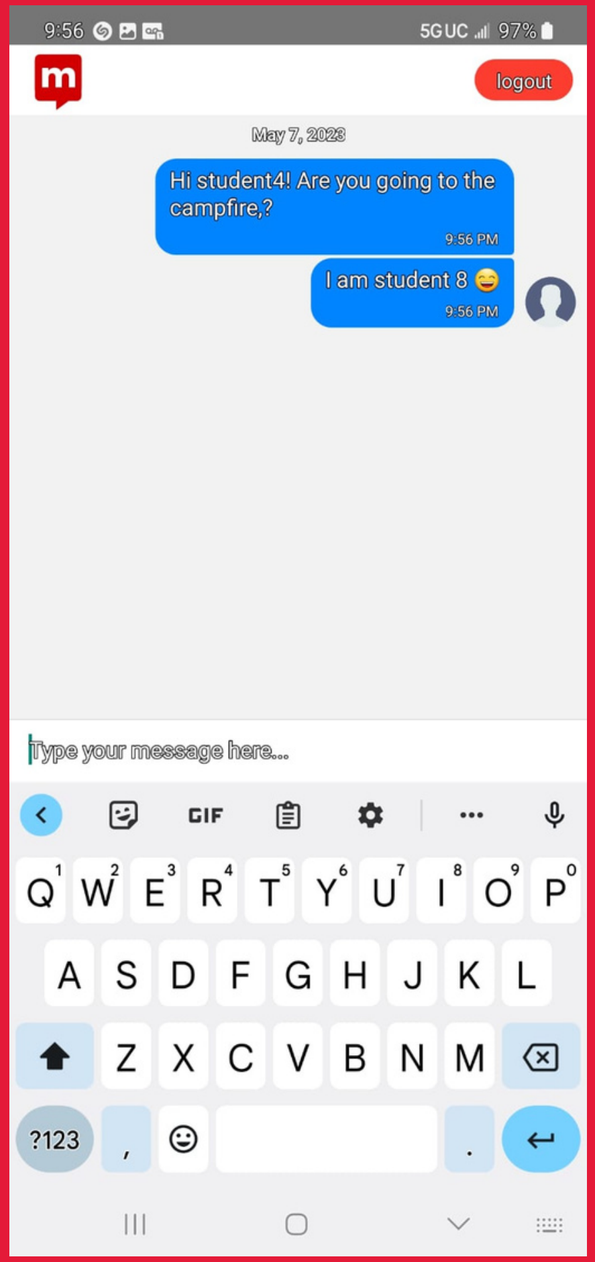

9:55
5GUC 98%
Full Name
student8
Email
student8@minotstateu.edu
Contact No.
333-333-3333
Password
Confirm Password
Sign Up
Login

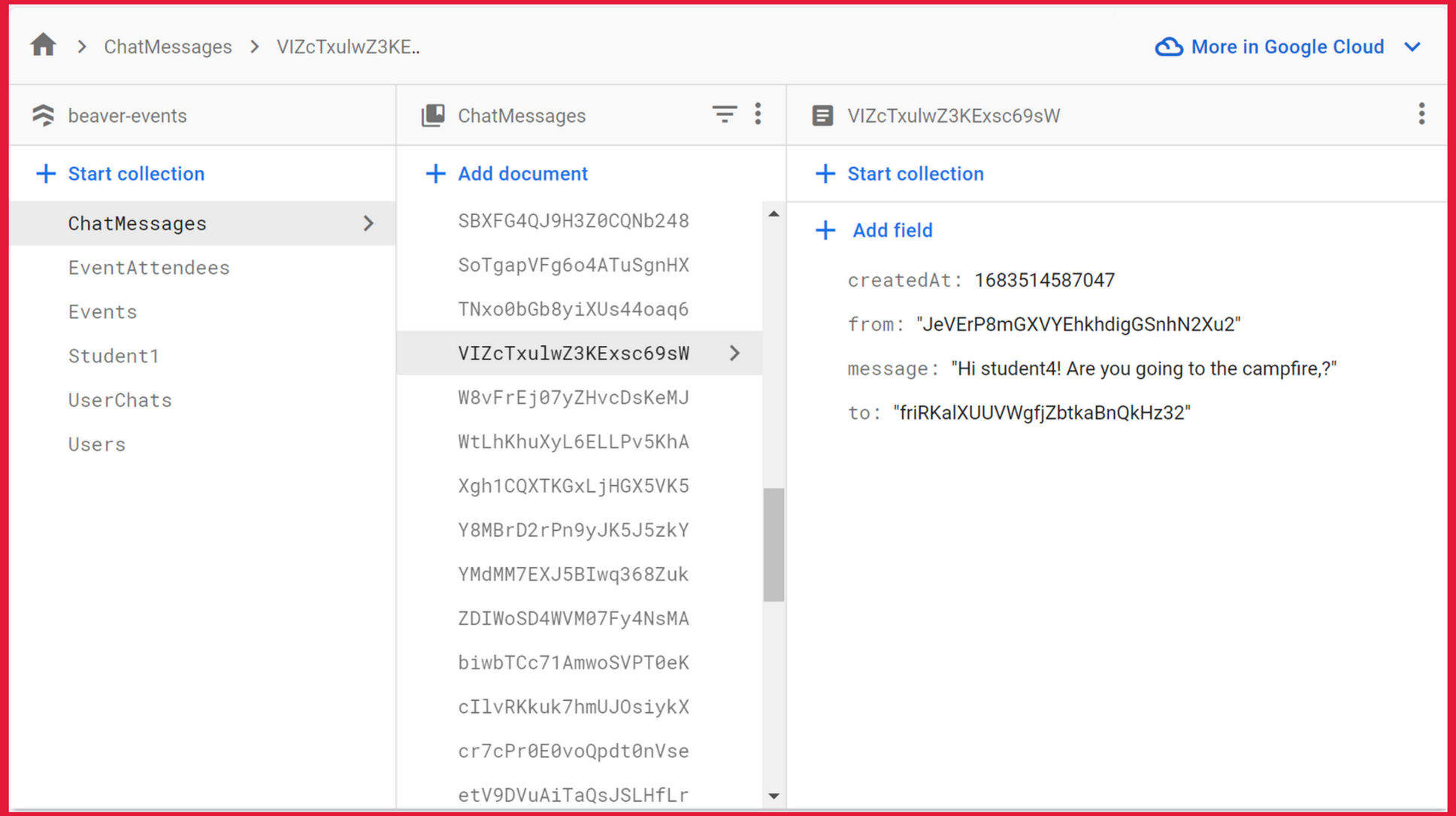

Users
y9ciHcK4iKxbW..
More in Google Cloud
beaver-events
Start collection
ChatMessages
EventAttendees
Events
Student1
UserChats
Users
Users
Add document
32umya1VePYUKnq8iYmH
54XeNJuBb722oWpTiyYK
5ZguLfBwCvvdAnNIcN
AGcgxivQ269VI2L2C
IQZOgu6xPQaC0Af4ME
Q907DVKYvYAsSSlEbybc
XDuE4J7xH6aWa0gv09re
afk5rTJsEl60NsmJwiL8
g2SC8oMkAo1XdnOSuD7B
jstQO1HiYUdCxKEYPfMG
mY9qVIJ3oO1jE3k3tEl1
qtFSqaowUF22o8C5H8lM
y9ciHcK4iKxbWXcuAtOt
z0KMyY62BMJ9MwQzbSVc
y9ciHcK4iKxbWXcuAtOt
Start collection
authId: "JeVErP8mGXVYEhkhdigGSnhN2Xu2"
contactNumber: "333-333-3333"
email: "student8@minotstateu.edu" (string)
fullName: "student8"
profilePic: ""

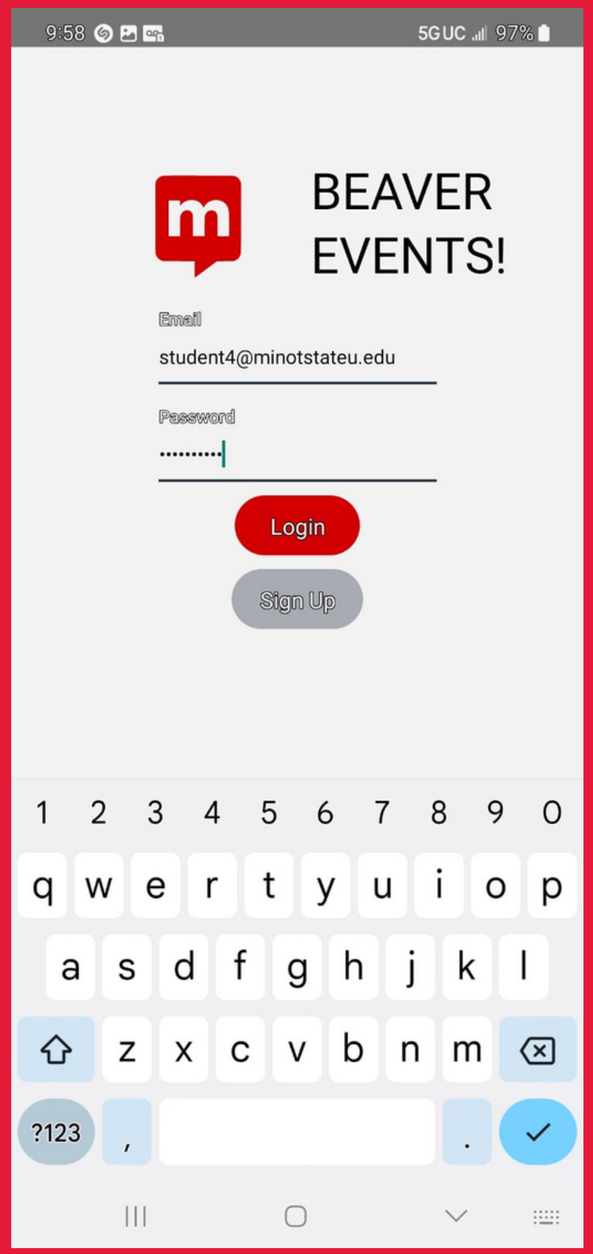

9:55
5GUC 98%
logout
Welcome
student8
Search
BINGO!
Come and play Bingo to win Big Prizes!
Old Main Lawn
8/23/2023 at 7:00:25 PM
Campfire in Quad
Smores + Friends = Best Night EVER!
Quad
8/24/2023 at 5:06:36 PM
Club fair
Meet cool clubs at MSU!
Quad
8/24/2023 at 11:00:53 AM
Flannel Fest
oak park
9/8/2023 at 1:00:53 PM
Events
Chat

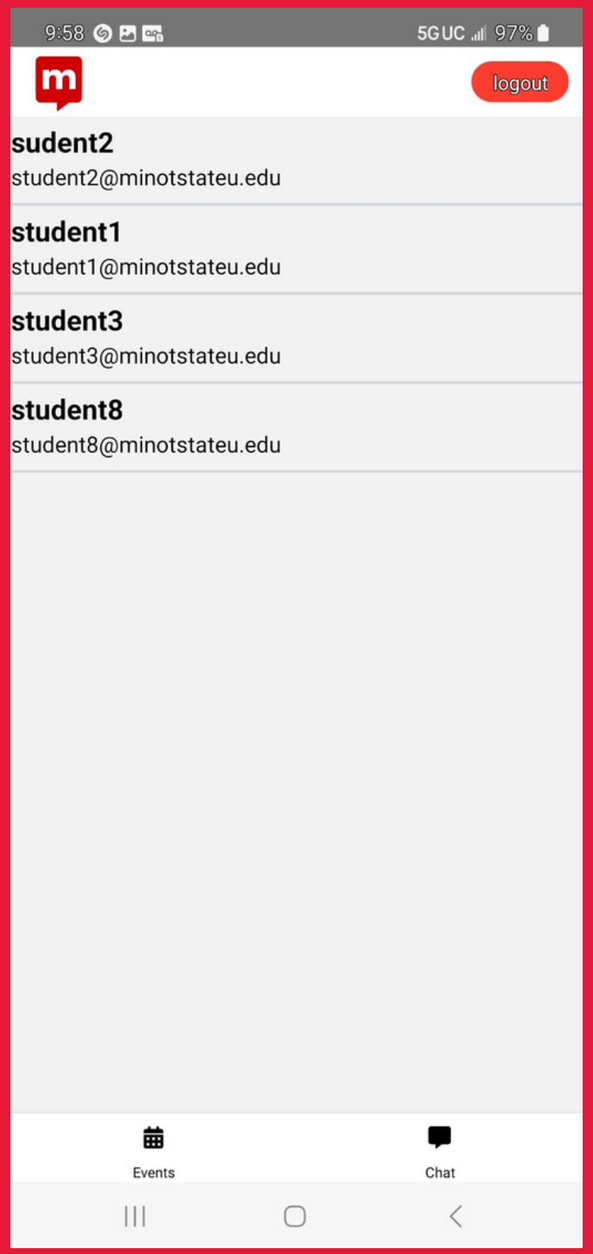

9:55
5GUC 98%
logout
Campfire in Quad
Smores + Friends = Best Night EVER!
Location:
Quad
Stars at:
8/24/2023 at 5:06:36 PM
Perks:
Smores + Music!
I will be attending
Interested Peers
Events
Chat

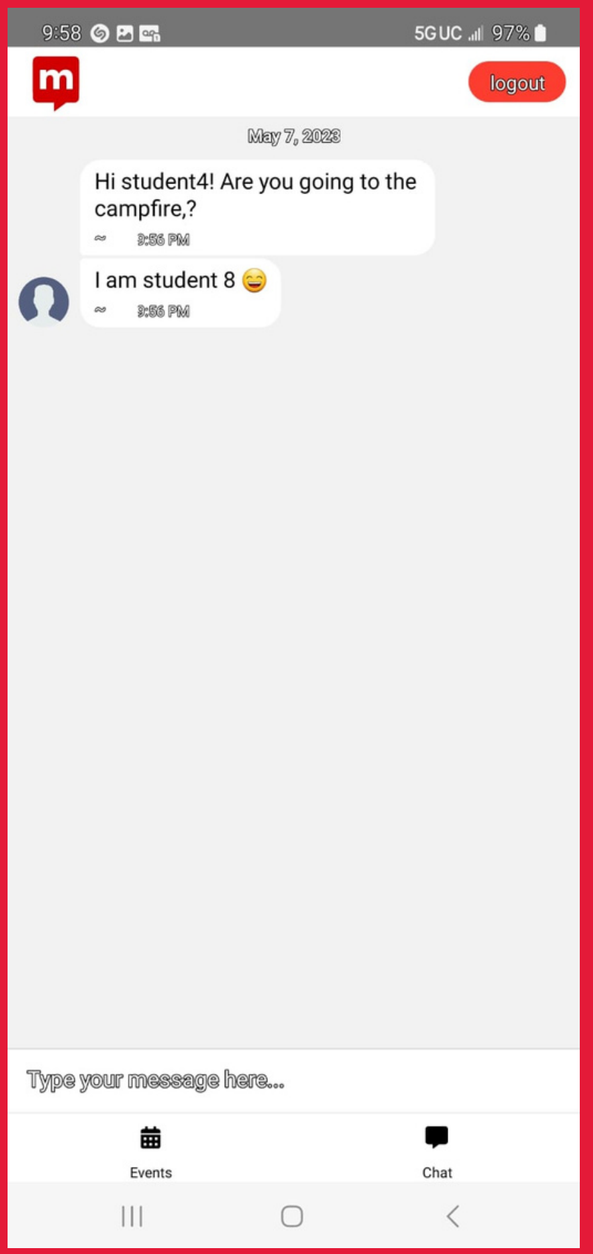

9:55
5GUC 98%
logout
Campfire in Quad
Smores + Friends = Best Night EVER!
Location:
Quad
Stars at:
8/24/2023 at 5:06:36 PM
Perks:
Smores + Music!
I will be attending
Interested Peers
student4
student4@minotstateu.edu
sudent2
student2@minotstateu.edu
student1
student1@minotstateu.edu

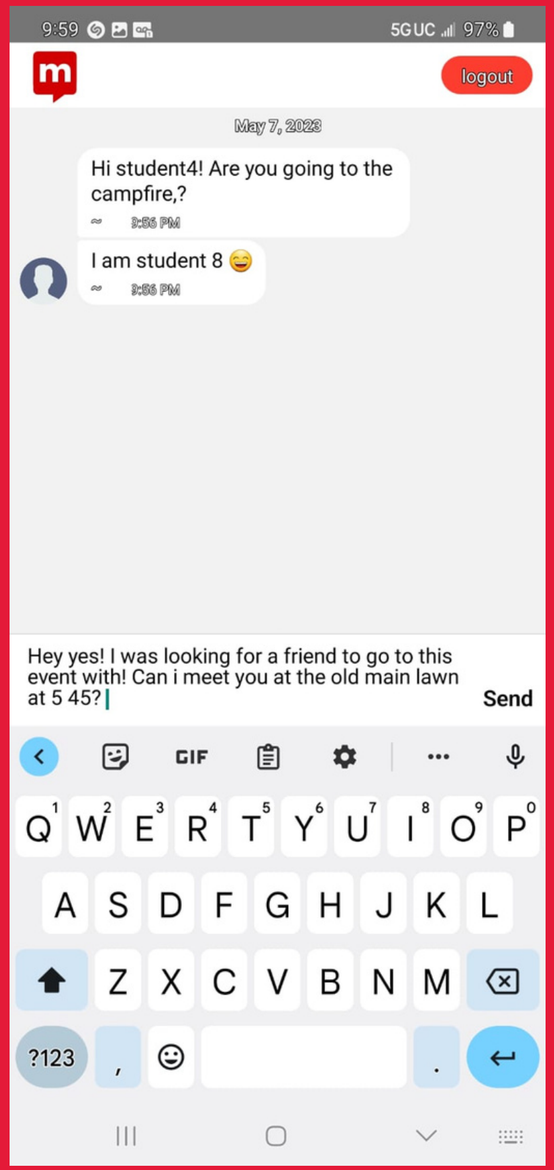

9:56
5GUC 97%
logout
May 7, 2023
Hi student4! Are you going to the campfire,?
9:56 PM
I am student 8 😄
9:56 PM
Type your message here...
GIF

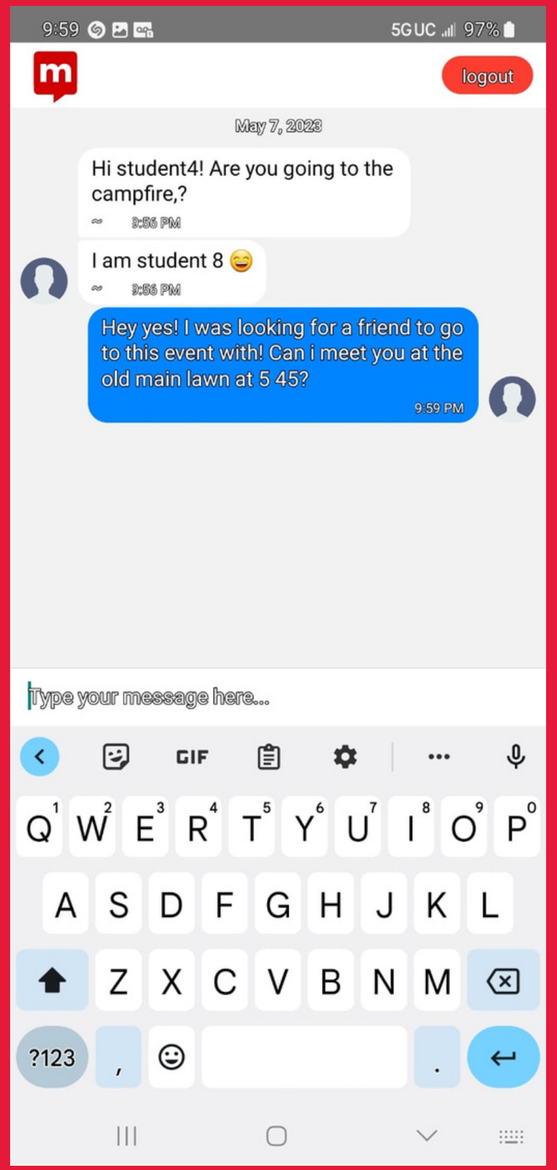

ChatMessages > VIZcTxulwZ3KE..
More in Google Cloud
beaver-events
Start collection
ChatMessages
EventAttendees
Events
Student1
UserChats
Users
ChatMessages
Add document
SBXFG4QJ9H3Z0CQNb248
SoTgapVFg6o4ATuSgnHX
TNxo0bGb8yiXUs44oaq6
VIZcTxulwZ3KExsc69sW
W8vFrEj07yZHvcDsKeMJ
WtLhKhuXyL6ELLPv5KhA
Xgh1CQXTKGxLjHGX5VK5
Y8MBrD2rPn9yJK5J5zkY
YMdMM7EXJ5BIwq368Zuk
ZDIWoSD4WVM07Fy4NsMA
biwbTCc71AmwoSVPT0eK
cIlvRKkuk7hmUJOsiykX
cr7cPr0E0voQpdt0nVse
etV9DVuAiTaQsJSLHfLr
VIZcTxulwZ3KExsc69sW
Start collection
Add field
createdAt: 1683514587047
from: "JeVErP8mGXVYEhkhdigGSnhN2Xu2"
message: "Hi student4! Are you going to the campfire,?"
to: "friRKalXUUVWgfjZbtkaBnQkHz32"


9:58
5GUC 97%
BEAVER EVENTS!
Email
student4@minotstateu.edu
Password
Login
Sign Up


9:58
5GUC 97%
logout
sudent2
student2@minotstateu.edu
student1
student1@minotstateu.edu
student3
student3@minotstateu.edu
student8
student8@minotstateu.edu
Events
Chat


9:58
5GUC 97%
logout
May 7, 2023
Hi student4! Are you going to the campfire,?
3:36 PM
I am student 8
3:36 PM
Type your message here...
Events
Chat


9:59
5GUC 97%
logout
May 7, 2023
Hi student4! Are you going to the campfire,?
3:56 PM
I am student 8
3:56 PM
Hey yes! I was looking for a friend to go to this event with! Can i meet you at the old main lawn at 5 45?
Send


9:59
5GUC 97%
logout
May 7, 2023
Hi student4! Are you going to the campfire,?
3:56 PM
I am student 8
3:56 PM
Hey yes! I was looking for a friend to go to this event with! Can i meet you at the old main lawn at 5 45?
9:59 PM
Type your message here...

Authentication components:

- **Login**
- **Signup**

Main components:

- **Home**
- **EventPage**
- **CreateEvent**

Chat components:

- **ChatList**
- **ChatMessages**

Contexts:

- **AuthContext: responsible for handling authentication state and user details.**

Firebase components:

- **Firestore: responsible for storing chat messages and user data.**
- **Firebase Storage: responsible for storing user profile images.**

UI Components:

- **Button**
- **Input**
- **ListItem**
- **Bubble**
- **GiftedChat**
- **SkeletonView**

Navigation components:

- **StackNavigator: responsible for navigating between screens.**

Helpers and Utils:

- **CustomBubble: Custom component to render chat bubble.**
- **getChatFriends: Helper function to retrieve the user's chat friends.**
- **formatDate: Utility function to format date in a readable way.**

- pages
  - chat
    - chatList.jsx
    - chatMessages.jsx
    - index.jsx
  - events
    - eventsDetailPage.jsx
    - eventsFeedPage.jsx
    - index.jsx
  - createEvent.jsx
  - login.jsx
  - signup.jsx

index.jsx

beaver_events-react > pages > events > index.jsx > ...

```jsx
import React from "react";
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import EventDetailScreen from "./eventsDetailPage";
import EventsFeedPage from "./eventsFeedPage";

const Stack = createNativeStackNavigator();

const Event = () => {
    return (
        <Stack.Navigator screenOptions={{ headerShown: false }}>
            <Stack.Screen name="Feed" component={EventsFeedPage} />
            <Stack.Screen name="Details" component={EventDetailScreen} />
        </Stack.Navigator>
    )
}

export default Event;
```

index.jsx

beaver_events-react > pages > chat > index.jsx > Chat

```jsx
        setOtherUser(data);
        navigation.navigate('ChatMessages');
    }

    return (
        <Stack.Navigator screenOptions={{ headerShown: false }} initialRouteName={"ChatList"}>
            <Stack.Screen name="ChatList">
                {(props) => <ChatList {...props} chatFriendsList={chatFriendsList} isChatFriendsLo
            </Stack.Screen>
            <Stack.Screen name="ChatMessages">
                {(props) => <ChatMessages {...props} otherUser={otherUser} />}
            </Stack.Screen>
        </Stack.Navigator>
    )
}

export default Chat;
```

Login-

login.jsx

beaver_events-react > pages > login.jsx > ...

```jsx
import React, { useContext, useState } from 'react';
import { StyleSheet, View, ActivityIndicator, Alert } from 'react-native';
import { Text, TextField, Button, Colors } from 'react-native-ui-lib';
import auth from '@react-native-firebase/auth';
import firestore from '@react-native-firebase/firestore';
import { AuthContext } from '../contexts/AuthContext';
import { Logo } from '../assets/svgs';

const Login = ({ navigation }) => {
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [isLoginLoading, setIsLoginLoading] = useState(false);
    const { setUserAuthId } = useContext(AuthContext);
```

```jsx
const handleLogin = async () => {
    if (!email || !password) return;
    setIsLoginLoading(true)
    try {
        const res = await auth().signInWithEmailAndPassword(email, password);
        const authId = res.user.uid;
        setUserAuthId(authId)
    } catch (err) {
        console.log(err);
        const code = err.code;
        if (code === "auth/user-not-found") {
            Alert.alert('Incorrect email or password', 'The entered email or password is incorrect.', [
                { text: 'OK' },
            ])
        }
    } finally {
        setIsLoginLoading(false)
    }
};
```

```
    return (
        <View style={styles.container}>
            <View style={styles.logoContainer}>
                <Logo style={styles.logo} />
                <View style={styles.nameContainer}>
                    <Text style={styles.title} text30>BEAVER</Text>
                    <Text style={styles.title} text30>EVENTS!</Text>
                </View>
            </View>
            <TextField
                placeholder="Email"
                floatingPlaceholder
                value={email}
                onChangeText={setEmail}
                keyboardType="email-address"
                style={styles.input}
            />
            <TextField
                placeholder="Password"
                floatingPlaceholder
                value={password}
                onChangeText={setPassword}
                secureTextEntry
                style={styles.input}
            />
            <Button label="Login" onPress={handleLogin} disabled={isLoginLoading}>
                {
                    isLoginLoading &&
                    <ActivityIndicator
                        size="small"
                        color="white"
                        style={{ marginRight: 10 }}
                    />
                }
            </Button>
            <Button label="Sign Up" style={{ backgroundColor: Colors.$backgroundNeutralIdle }} onPress={() => { navigation.navigate("Signup") }} />
        </View>
    );
};
```

```
const styles = StyleSheet.create({
    container: {
        display: "flex",
        alignItems: 'center',
        paddingTop: 80,
        height: "100%",
        gap: 10,
    },
    logoContainer: {
        width: "90%",
        display: "flex",
        flexDirection: "row",
        justifyContent: "center",
        alignItems: "center"
    },
    nameContainer: {
        display: "flex",
        flexDirection: "column",
        justifyContent: "center"
    },
    logo: {
        height: "100%",
        width: "160px",
    },
    input: {
        borderBottomColor: Colors.grey10,
        borderBottomWidth: 2,
        width: 200,
        height: 40,
    }
});

export default Login;
```

Sign-UP

signup.jsx

beaver_events-react > pages > signup.jsx > styles > ProfilePicSelector > shadowOffset

```jsx
import React, { useContext, useState } from 'react';
import { StyleSheet, View, ScrollView, Alert, TouchableOpacity, ActivityIndicator } from 'react-native';
import { Text, TextField, Button, Colors, Image, ActionSheet } from 'react-native-ui-lib';
import auth from '@react-native-firebase/auth';
import storage from '@react-native-firebase/storage';
import firestore from '@react-native-firebase/firestore';
import { AuthContext } from '../contexts/AuthContext';
import ImagePicker from "react-native-image-crop-picker";

```

```jsx
const validateFields = () => {
    const errs = {};
    if (!fullName) {
        errs.fullName = "Full Name is required"
    } else if (fullName.length < 6) {
        errs.fullName = "Full Name must be 6 characters long"
    }

    if (!email) {
        errs.email = "Email is required"
    } else if (/\b[A-Z0-9._%+-]+@minotstateu\.com\b/i.test(email)) {
        errs.email = "Email is invalid";
    }

    if (!password) {
        errs.password = "Password is required"
    } else if (password.length < 6) {
        errs.password = "Password must be 6 characters long"
    }

    if (!confirmPassword) {
        errs.confirmPassword = "Password is required"
    } else if (confirmPassword !== password) {
        errs.confirmPassword = "Please make sure your passwords match"
    }

    setErrMsgs(errs);
    return Object.keys(errs).length === 0;
}
```

```jsx
        // Storing users info
        await firestore()
            .collection('Users')
            .add(userDetail)

        setUserDetails(userDetail);
        setIsLoggedIn(true);
    } catch (err) {
        console.log(err);
        if (err.message) {
            Alert.alert('Signup Error', err.message, [
                { text: 'OK' },
            ]);
        }
    } finally {
        setIsSignupLoading(false);
    }
};
```

Create event

```
const handleCreateEvent = async () => {
    const validationSuccessful = validateFields()
    if (!validationSuccessful) return;

    setIsCreateLoading(true);
    try {
        const eventDetails = {
            name,
            description,
            eventDateTime: eventDate.toUTCString(),
            location,
            prizes,
            perks
        }

        const newEvent = await firestore()
            .collection('Events')
            .add(eventDetails)

        if (eventThumbnail) {
            // Uploading user profile pic to cloud storage
            const eventThumbnailBucketRef = storage().ref(`eventThumbnails/${newEvent.id}`);
            await eventThumbnailBucketRef.putFile(eventThumbnail.path);
            const uploadedEventThumbnail = await eventThumbnailBucketRef.getDownloadURL();
            await newEvent.update({
                thumbnail: uploadedEventThumbnail
            })
        }

        Alert.alert("Created", `Event ${name} created!`, [
            {
                text: 'Done', onPress: () => {
                    resetForm();
                    setShouldLoadEvents(true);
                    navigation.navigate('Events');
                }
            },
            { text: 'Create Another', onPress: () => resetForm() },
        ]);
    } catch (err) {
        console.log(err);
        if (err.message) {
            Alert.alert('Signup Error', err.message, [
                { text: 'OK' },
            ]);
        }
    } finally {
        setIsCreateLoading(false)
    }
};
```

```
const validateFields = () => {
    const errs = {};
    if (!name) {
        errs.name = "Event Name is required"
    } else if (name.length < 6) {
        errs.name = "Event Name must be 6 characters long"
    }

    if (!location) {
        errs.location = "Location is required"
    }

    setErrMsgs(errs);
    return Object.keys(errs).length === 0;
}

const resetForm = () => {
    setName("");
    setEventDate(new Date());
    setEventThumbnail(null);
    setLocation("");
    setPerks("");
    setPrizes("");
    setErrMsgs({})
}
```

```
const selectEventThumbnailFromGallery = () => {
    ImagePicker.openPicker({
        width: 300,
        height: 400,
        cropping: true
    }).then((image) => {
        setEventThumbnail(image);
    }).catch((err) => {
        console.log(err);
        if (err.code === "E_NO_LIBRARY_PERMISSION") {
            Alert.alert('Permission Denied', 'You need to allow this app to
                { text: "OK" },
            ]);
        }
    });
};
```

```
<TextField
    placeholder="Event Name"
    floatingPlaceholder
    value={name}
    onChangeText={setName}
    style={styles.input}
    enableErrors
    validationMessage={errMsgs.name}
/>
<TextField
    placeholder="Event Description"
    multiline={true}
    floatingPlaceholder
    numberOfLines={4}
    value={description}
    onChangeText={setDescription}
    style={styles.multiplineInput}
    enableErrors
    validationMessage={errMsgs.description}
/>
```

```
const onDateTimeChange = (event, selectedDate) => {
    const currentDate = selectedDate;
    setShowDateTimePicker(false);
    setEventDate(currentDate);
};

const showDateTimeMode = (currentMode) => {
    setShowDateTimePicker(true);
    setDateTimeMode(currentMode);
};

const showDatepicker = () => {
    showDateTimeMode('date');
};

const showTimepicker = () => {
    showDateTimeMode('time');
};
```

```
<ActionSheet
    visible={isBottomSheetVisible}
    useNativeIOS={true}
    options={
        [{
            label: 'Open gallery',
            onPress: () => {
                setIsBottomSheetVisible(false);
                selectEventThumbnailFromGallery();
            }
        },
        {
            label: 'Open camera',
            onPress: () => {
                setIsBottomSheetVisible(false);
                captureEventThumbnail();
            }
        },
        { label: 'Cancel', onPress: () => setIsBottomSheetVisible(false) }
        ]
    }
    onDismiss={() => setIsBottomSheetVisible(false)}
    onModalDismissed={() => setIsBottomSheetVisible(false)}
    cancelButtonIndex={2}
/>
```

Event Feed and Event Detail Page

```jsx
const EventCard = ({ event, allowDelete, onDelete }) => (
    <Card style={styles.card}>
        {
            allowDelete &&
            <Button style={styles.eventDeleteButton} avoidMinWidth size={Button.sizes.xSmall} onPress={() => onDelete(event)} round
                <FontAwesomeIcon icon={faXmark} color={"white"} />
            </Button>
        }
        {
            event.thumbnail ?
                <Image source={{ uri: event.thumbnail }} style={styles.thumbnail} />
                :
                <></>
        }
        <View style={styles.textContainer}>
            <Text style={styles.name} text60>{event.name}</Text>
            <Text style={styles.description}>{event.description}</Text>
            <View style={styles.locationContainer}>
                <Badge style={styles.location} label={event.location} size={16} />
                {
                    event.eventDateTime ?
                        <Badge
                            style={styles.eventDateTime}
                            label={
                                event.eventDateTime &&
                                `${new Date(event.eventDateTime).toLocaleDateString()} at ${new Date(event.eventDateTime).toLocaleT
                            }
                            size={16} />
                        :
                        <></>
                }
            </View>
        </View>
    </Card>
)
```

```jsx
<View style={styles.card}>
    {
        event?.thumbnail &&
        <Image source={{ uri: event.thumbnail }} style={styles.image} />
    }
    <View style={styles.textContainer}>
        <Text style={styles.title}>{event.name}</Text>
        <Text style={styles.description}>{event.description}</Text>
        {
            event.location &&
            <>
                <Text style={styles.label}>Location:</Text>
                <Text style={styles.subtitle}>{event.location}</Text>
            </>
        }
        {
            event.eventDateTime &&
            <>
                <Text style={styles.label}>Stars at:</Text>
                <Text style={styles.subtitle}>
                    {new Date(event.eventDateTime).toLocaleDateString()} at{' '}
                    {new Date(event.eventDateTime).toLocaleTimeString()}
                </Text>
            </>
        }
        {
            event.perks &&
            <>
                <Text style={styles.label}>Perks:</Text>
                <Text style={styles.perks}>{event.perks}</Text>
            </>
        }
        {
            event.prizes &&
            <>
                <Text style={styles.label}>Prizes:</Text>
                <Text style={styles.prize}>
                    {event.prizes}
                </Text>
            </>
        }
    </View>
```

Index for event-

index.jsx

beaver_events-react > pages > events > index.jsx > ...

```jsx
import React from "react";
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import EventDetailScreen from "./eventsDetailPage";
import EventsFeedPage from "./eventsFeedPage";

const Stack = createNativeStackNavigator();

const Event = () => {
    return (
        <Stack.Navigator screenOptions={{ headerShown: false }}>
            <Stack.Screen name="Feed" component={EventsFeedPage} />
            <Stack.Screen name="Details" component={EventDetailScreen} />
        </Stack.Navigator>
    )
}

export default Event;
```

Chat list

```
const renderRow = (row, idx, onPress) => {
    return (
        <View key={`row-${idx}`}>
            <ListItem
                activeBackgroundColor={Colors.grey60}
                activeOpacity={0.3}
                style={styles.listItem}
                onPress={() => { onPress(row) }}
            >
                {
                    row.profilePic ?
                        <ListItem.Part left>
                            <Image source={{ uri: row.profilePic }} style={styles.image} />
                        </ListItem.Part> : <></>
                }
                <ListItem.Part style={styles.listItemText}>
                    <ListItem.Part>
                        <Text text60 numberOfLines={1}>
                            {row.fullName}
                        </Text>
                    </ListItem.Part>
                    <ListItem.Part>
                        <Text text70 numberOfLines={1}>
                            {row.email}
                        </Text>
                    </ListItem.Part>
                </ListItem.Part>
            </ListItem>
        </View>
    );
}
```

```
const ChatList = ({ chatFriendsList, isChatFriendsLoading, onPressChatList }) => {
    return (
        isChatFriendsLoading ?
            <SkeletonView
                template={SkeletonView.templates.LIST_ITEM}
                times={6}
            />
            :
            chatFriendsList.length ?
                <FlatList
                    data={chatFriendsList}
                    renderItem={({ item, index }) => renderRow(item, index, onPressChatList)}
                /> :
                <View style={styles.noPeersContainer}>
                    <Text>
                        <FontAwesomeIcon icon={faFaceFrown} size={32} />
                    </Text>
                    <Text grey40>
                        No chats
                    </Text>
                </View>
    )
}
```

Chat messages

```
const ChatMessage = ({ otherUser }) => {
    const [messages, setMessages] = useState([]);
    const { userAuthId, userDetails } = useContext(AuthContext);

    const currentUser = useMemo(() => ({ ...userDetails, _id: userDetails.authId, avatar: (userDetails?.profilePic || defaultAvatar) }));

    useEffect(() => {
        if (!currentUser && !otherUser) return;
        const unsubscribe = firestore()
            .collection('UserChats')
            .doc(otherUser.userChatId)
            .onSnapshot(async (docSnapshot) => {
                if (docSnapshot.exists) {
                    const messagesRef = docSnapshot.data()?.messages;
                    const formattedMsgs = [];
                    for (let msgRef of messagesRef) {
                        const msgSnap = await msgRef.get();
                        const msgSnapData = msgSnap.data();
                        const formattedMsg = {
                            _id: Math.round(Math.random() * 1000000),
                            text: msgSnapData.message,
                            createdAt: msgSnapData.createdAt,
                            // Adding _id, because by default gifted chat packaged using _id to differentiate users
                            user: msgSnapData.from === userAuthId ? { ...currentUser, _id: currentUser.authId } : { ...otherUser, _id: otherUser.authId },
                        }
                        formattedMsg.user.avatar = formattedMsg?.user?.profilePic || defaultAvatar
                        formattedMsgs.push(formattedMsg);
                    }
                    setMessages(formattedMsgs);
                }
            });

        return () => unsubscribe();
    }, [currentUser, otherUser])
```

```
    return (
        <View style={styles.container}>
            <GiftedChat
                renderUsernameOnMessage
                style={styles.GiftedChat}
                messages={messages}
                user={currentUser}
                placeholder="Type your message here..."
                onSend={handleSend}
                inverted={false}
                textInputStyle={{ color: Colors.$textDefault }}
                renderBubble={CustomBubble}
                showUserAvatar={true}
            />
        </View>
    );
};

const styles = StyleSheet.create({
    container: {
        flex: 1,
    },
    GiftedChat: {
        backgroundColor: "red",
        height: "100%",
        color: "black"
    },
    messageContainer: {
        flexDirection: 'row',
        alignItems: 'center',
        marginVertical: 10,
        paddingHorizontal: 10,
    },
    leftMessage: {
        backgroundColor: '#f0f0f0',
        padding: 10,
        borderRadius: 10,
        maxWidth: '80%',
        alignSelf: 'flex-start',
    },
    rightMessage: {
        backgroundColor: '#0084ff',
        padding: 10,
        borderRadius: 10,
        maxWidth: '80%',
        alignSelf: 'flex-end',
    },
});

export default ChatMessages;
```

```
const handleSend = useCallback((messages) => {
    for (let messageObj of messages) {
        const message = messageObj.text;
        const [illegible] = messageObj.user.authId;
        const to = otherUser.authId;
        const createdAt = Date.now();
        const newMessage = {
            from,
            to,
            message,
            createdAt
        };
        (async () => {
            try {
                const createMessage = await firestore()
                    .collection('ChatMessages')
                    .add(newMessage);

                await firestore()
                    .collection('UserChats')
                    .doc(otherUser.userChatId)
                    .update({
                        messages: firebase.firestore.FieldValue.arrayUnion(createMessage)
                    })

                setMessages(prev => [...prev, { ...newMessage, _id: createMessage?.id }])
            } catch (err) {
                console.log(err)
            }
        })();
    }
}, [])
```

Index for chat-

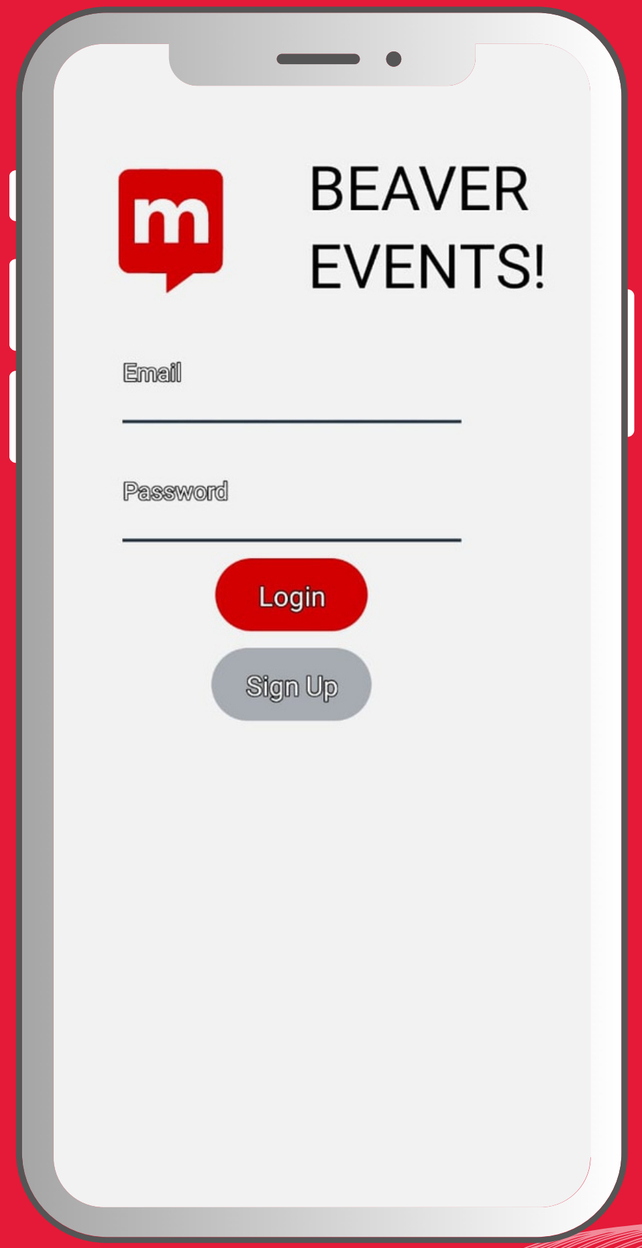

```
const Stack = createNativeStackNavigator();

const getChatFriends = async (userAuthId) => {
    console.log("(((((((( ", userAuthId)
    const chatsSnapshot = await firestore()
        .collection('UserChats')
        .where('authIds', 'array-contains', userAuthId)
        .get();

    const chatFriendIdsWithMessageRef = [];
    chatsSnapshot.forEach((docSnap) => {
        const authIds = docSnap.data().authIds;
        const messagesRef = docSnap.data().messages;
        const id = docSnap.id;
        for (let authId of authIds) {
            if (authId !== userAuthId) {
                chatFriendIdsWithMessageRef.push({ authId, messagesRef, id })
            };
        }
    })

    const chatFriends = []
    for (let chatItem of chatFriendIdsWithMessageRef) {
        const friendId = chatItem.authId
        if (userAuthId === friendId) continue;
        const peersSnapshot = await firestore().collection('Users').where('authId', '==', friendId).get()
        peersSnapshot.forEach((docSnap) => {
            chatFriends.push({ ...docSnap.data(), userChatId: chatItem.id });
        })
    }

    return chatFriends;
}
```


BEAVER
EVENTS!
Email
Password
Login
Sign Up

Challenges I faced :

1. Choosing the right software
2. Real-time messaging
2. Handling User Authentication
3. Chat history
4. Optimizing performance
5. UI/UX Design
6. Debugging and Testing

Immediate Improvements

1. Firebase storage authorization for images
2. 2 factor authentication
3. Message encryption
4. Group chat

Future Plans

1. Scalability
2. Server
3. Leverage the large language models and NLP tools to have in-app recommender system.
4. Deploy in Minot State University??

Thank You!