CometChat Internship Task– UI Kit Integration Assignment

Project Overview

I have developed a basic chat application by integrating the CometChat UI Kit using [React Native]. This assignment demonstrates the implementation of CometChat's real-time messaging features such as one-on-one chats, group conversations, user list, and message history

Steps Followed

- Signed up on CometChat Dashboard using the required format vmugeshvmugesh24+test@gmail.com.
- **Created a new app** in the CometChat dashboard and retrieved the required credentials (App ID, Auth Key, Region).
- **Installed the CometChat UI Kit SDK** into my application using:
  npm install @cometchat-pro/react-native-ui-kit –save
- **Configured the app** using CometChat.init() in the app's entry point
- **Integrated CometChat Components** like CometChatConversations, CometChatUsers, and CometChatMessages into the application screens.

Code Snippet

App.js

```
import { registerRootComponent } from 'expo';

import { ExpoRoot } from 'expo-router';

export function App() {

 const ctx = require.context('./app');

return <ExpoRoot context={ctx} />;

}

registerRootComponent(App);
```

Login.tsx

```
import { CometChat } from '@cometchat/chat-sdk-react-native';

import { router } from 'expo-router';

import React, { useState } from 'react';

import { ActivityIndicator, Alert, StyleSheet, TextInput, TouchableOpacity } from 'react-native';
```

```
import { ThemedText } from '@/components/ThemedText';
import { ThemedView } from '@/components/ThemedView';
import { COMETCHAT_CONSTANTS } from '@/constants/CometChatConfig';
export default function LoginScreen() {
const [uid, setUid] = useState('');
const [loading, setLoading] = useState(false);
const handleLogin = async () => {
if (!uid.trim()) {
Alert.alert('Error', 'Please enter a valid User ID');
return;}
setLoading(true);
try {
const user = await CometChat.login(uid, COMETCHAT_CONSTANTS.AUTH_KEY);
console.log('Login Successful:', user);
setLoading(false);
router.replace('/(tabs)');
} catch (error: any) {
console.log('Login failed with error:', error);
const errorMessage = error.message || 'Failed to log in. Please check your credentials.';
Alert.alert('Login Error', errorMessage);
setLoading(false);}}
return (
<ThemedView style={styles.container}>
<ThemedText type="title" style={styles.title}>CometChat</ThemedText>
<ThemedText style={styles.subtitle}>Chat Application</ThemedText>
<ThemedView style={styles.inputContainer}>
```

```
<TextInput

style={styles.input}

placeholder="Enter User ID (e.g. superhero1)"

placeholderTextColor="#888"

value={uid}

onChangeText={setUid}

autoCapitalize="none"/>

<TouchableOpacity

style={styles.button}

onPress={handleLogin}

disabled={loading}>

{loading ? (

<ActivityIndicator color="#fff" />) : (

<ThemedText style={styles.buttonText}>Login</ThemedText>)}

</TouchableOpacity>

<TouchableOpacity onPress={() => router.push('/signup')}>

<ThemedText style={styles.linkText}>

Don&apos;t have an account? Sign up

</ThemedText>

</TouchableOpacity></ThemedView>

<ThemedText style={styles.helpText}>

Use one of these test users: superhero1, superhero2, superhero3, superhero4

</ThemedText>

</ThemedView>);}

const styles = StyleSheet.create({

container: {
```

```
flex: 1,
justifyContent: 'center',
padding: 20,},
title: {
textAlign: 'center',
marginBottom: 10,
fontSize: 36,},
subtitle: {
textAlign: 'center',
marginBottom: 48,
fontSize: 18,
opacity: 0.7,},
inputContainer: {
width: '100%',
marginBottom: 20,},
input: {
backgroundColor: '#f0f0f0',
padding: 15,
borderRadius: 10,
marginBottom: 15,
fontSize: 16,},
button: {
backgroundColor: '#0a7ea4',
padding: 15,
borderRadius: 10,
alignItems: 'center',
```

```
marginBottom: 15,},

buttonText: {

color: 'white',

fontSize: 16,

fontWeight: 'bold',},

linkText: {

textAlign: 'center',

color: '#0a7ea4',

marginTop: 10,},

helpText: {

textAlign: 'center',

marginTop: 20,

fontSize: 14,

opacity: 0.7,},});
```

SignUp.tsx

```
import { CometChat } from '@cometchat/chat-sdk-react-native';

import { router } from 'expo-router';

import React, { useState } from 'react';

import { ActivityIndicator, Alert, StyleSheet, TextInput, TouchableOpacity } from 'react-native';



import { ThemedText } from '@/components/ThemedText';

import { ThemedView } from '@/components/ThemedView';

import { COMETCHAT_CONSTANTS } from '@/constants/CometChatConfig';



export default function SignupScreen() {

  const [uid, setUid] = useState('');
```

```
  const [name, setName] = useState('');

  const [loading, setLoading] = useState(false);


  const handleSignup = async () => {
   if (!uid.trim() || !name.trim()) {
     Alert.alert('Error', 'Please fill in all fields');
     return;
   }


   setLoading(true);


   try {


     const user = new CometChat.User(uid, name);


     const createdUser = await CometChat.createUser(user,
COMETCHAT_CONSTANTS.AUTH_KEY);
     console.log('User created successfully:', createdUser);
     try {
       // Fix: directly pass the uid and authKey as separate parameters
       const loggedInUser = await CometChat.login(uid,
COMETCHAT_CONSTANTS.AUTH_KEY)
       console.log('Login Successful:', loggedInUser);
       setLoading(false);
       router.replace('/(tabs)');
     } catch (loginError: any) {
       console.log('Login failed with error:', loginError);
```

```
      Alert.alert(
        'Login Error',
        'User created but login failed. Please try logging in manually.',
        [
          {
            text: 'OK',
            onPress: () => router.push('/login')
          }
        ]
      );
      setLoading(false);
    }
  } catch (error: any) {
    console.log('User creation failed with error:', error);
    if (error.code === 'ERR_UID_ALREADY_EXISTS') {
      Alert.alert(
        'Signup Error',
        'This User ID already exists. Please try logging in instead.',
        [
          {
            text: 'Go to Login',
            onPress: () => router.push('/login')
          },
          {
            text: 'Try Again',
            style: 'cancel'
```

```
          }
        ]
      );
    } else {
      Alert.alert('Signup Error', error.message || 'Failed to create user');
    }

    setLoading(false);
  }
};

return (
  <ThemedView style={styles.container}>
    <ThemedText type="title" style={styles.title}>Create Account</ThemedText>
    <ThemedText style={styles.subtitle}>Join CometChat</ThemedText>

    <ThemedView style={styles.inputContainer}>
      <TextInput
        style={styles.input}
        placeholder="Enter User ID"
        placeholderTextColor="#888"
        value={uid}
        onChangeText={setUid}
        autoCapitalize="none"
      />
```

```
  <TextInput
    style={styles.input}
    placeholder="Enter Name"
    placeholderTextColor="#888"
    value={name}
    onChangeText={setName}
  />

  <TouchableOpacity
    style={styles.button}
    onPress={handleSignup}
    disabled={loading}
  >
    {loading ? (
      <ActivityIndicator color="#fff" />
    ) : (
      <ThemedText style={styles.buttonText}>Sign Up</ThemedText>
    )}
  </TouchableOpacity>

  <TouchableOpacity onPress={() => router.push('/login')}>
    <ThemedText style={styles.linkText}>
      Already have an account? Login
    </ThemedText>
  </TouchableOpacity>
</ThemedView>
```

```
    </ThemedView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    padding: 20,
  },
  title: {
    textAlign: 'center',
    marginBottom: 10,
    fontSize: 36,
  },
  subtitle: {
    textAlign: 'center',
    marginBottom: 48,
    fontSize: 18,
    opacity: 0.7,
  },
  inputContainer: {
    width: '100%',
    marginBottom: 20,
  },
  input: {
```

```
    backgroundColor: '#f0f0f0',
    padding: 15,
    borderRadius: 10,
    marginBottom: 15,
    fontSize: 16,
  },
  button: {
    backgroundColor: '#0a7ea4',
    padding: 15,
    borderRadius: 10,
    alignItems: 'center',
    marginBottom: 15,
  },
  buttonText: {
    color: 'white',
    fontSize: 16,
    fontWeight: 'bold',
  },
  linkText: {
    textAlign: 'center',
    color: '#0a7ea4',
    marginTop: 10,
  },
});
```

Profile.tsx

```
import { CometChat } from '@cometchat/chat-sdk-react-native';
```

```
import { Ionicons } from '@expo/vector-icons';
import { Image } from 'expo-image';
import { router } from 'expo-router';
import React, { useEffect, useState } from 'react';
import { ActivityIndicator, Alert, StyleSheet, TouchableOpacity } from 'react-native';

import { ThemedText } from '@/components/ThemedText';
import { ThemedView } from '@/components/ThemedView';

export default function ProfileScreen() {
  const [user, setUser] = useState<any>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    CometChat.getLoggedinUser().then(
      (user: any) => {
        console.log('User details:', user);
        setUser(user);
        setLoading(false);
      },
      (error: any) => {
        console.log('User details fetching failed with error:', error);
        setLoading(false);
      }
    );
  }, []);
```

```
const handleLogout = async () => {
  Alert.alert(
    'Logout',
    'Are you sure you want to logout?',
    [
      {
        text: 'Cancel',
        style: 'cancel',
      },
      {
        text: 'Logout',
        onPress: async () => {
          setLoading(true);
          try {
            const currentUser = await CometChat.getLoggedinUser();
            if (currentUser) {
              await CometChat.logout();
            }
            router.replace('/(auth)/login');
          } catch (error: any) {
            router.replace('/(auth)/login');
          }
        },
      },
    ],
```

```
      { cancelable: false }
    );
  };

  const showAccountInfo = () => {
    Alert.alert(
      'Account Information',
      `Name: ${user?.name || 'N/A'}\nUID: ${user?.uid}\nStatus: ${user?.status || 'N/A'}`,
      [{ text: 'OK' }]
    );
  };

  const showNotificationsSettings = () => {
    Alert.alert(
      'Notifications',
      'Notification settings would be displayed here',
      [{ text: 'OK' }]
    );
  };

  const showPrivacySecurity = () => {
    Alert.alert(
      'Privacy & Security',
      'When You login please Enter the UID properly',
      [{ text: 'OK' }]
    );
```

```
};

const showHelpSupport = () => {
  Alert.alert(
    'Help & Support',
    'For support, please contact us at careers.intern@cometchat.com',
    [{ text: 'OK' }]
  );
};

if (loading) {
  return (
    <ThemedView style={styles.loadingContainer}>
      <ActivityIndicator size="large" color="#0a7ea4" />
    </ThemedView>
  );
}

return (
  <ThemedView style={styles.container}>
    <ThemedView style={styles.header}>
      <ThemedText type="title">Profile</ThemedText>
    </ThemedView>

    <ThemedView style={styles.profileContainer}>
      <Image
```

```
            source={{ uri: user?.avatar ||
'https://www.gravatar.com/avatar/00000000000000000000000000000000?d=mp&f=y' }}
            style={styles.avatar}
          />
          <ThemedText type="title" style={styles.userName}>
            {user?.name || user?.uid}
          </ThemedText>
          <ThemedText style={styles.userUid}>
            @{user?.uid}
          </ThemedText>
        </ThemedView>

        <ThemedView style={styles.infoContainer}>
          <TouchableOpacity style={styles.infoRow} onPress={showAccountInfo}>
            <Ionicons name="person-outline" size={24} color="#666" />
            <ThemedText style={styles.infoText}>Account Information</ThemedText>
            <Ionicons name="chevron-forward" size={20} color="#ccc" />
          </TouchableOpacity>

          <TouchableOpacity style={styles.infoRow} onPress={showNotificationsSettings}>
            <Ionicons name="notifications-outline" size={24} color="#666" />
            <ThemedText style={styles.infoText}>Notifications</ThemedText>
            <Ionicons name="chevron-forward" size={20} color="#ccc" />
          </TouchableOpacity>

          <TouchableOpacity style={styles.infoRow} onPress={showPrivacySecurity}>
            <Ionicons name="lock-closed-outline" size={24} color="#666" />
```

```
          <ThemedText style={styles.infoText}>Privacy & Security</ThemedText>
          <Ionicons name="chevron-forward" size={20} color="#ccc" />
        </TouchableOpacity>

        <TouchableOpacity style={styles.infoRow} onPress={showHelpSupport}>
          <Ionicons name="help-circle-outline" size={24} color="#666" />
          <ThemedText style={styles.infoText}>Help & Support</ThemedText>
          <Ionicons name="chevron-forward" size={20} color="#ccc" />
        </TouchableOpacity>
      </ThemedView>

      <TouchableOpacity style={styles.logoutButton} onPress={handleLogout}>
        <Ionicons name="log-out-outline" size={24} color="#ff3b30" />
        <ThemedText style={styles.logoutText}>Logout</ThemedText>
      </TouchableOpacity>

      <ThemedText style={styles.versionText}>
        www.cometchat.com
      </ThemedText>
    </ThemedView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
```

```
  paddingTop: 60,
},
loadingContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
},
header: {
  paddingHorizontal: 20,
  paddingBottom: 20,
},
profileContainer: {
  alignItems: 'center',
  padding: 20,
  paddingBottom: 40,
},
avatar: {
  width: 100,
  height: 100,
  borderRadius: 50,
  marginBottom: 16,
},
userName: {
  fontSize: 24,
  marginBottom: 8,
},
```

```
userUid: {
  fontSize: 16,
  opacity: 0.6,
},
infoContainer: {
  borderTopWidth: 1,
  borderTopColor: '#f0f0f0',
  paddingTop: 20,
},
infoRow: {
  flexDirection: 'row',
  alignItems: 'center',
  paddingVertical: 16,
  paddingHorizontal: 20,
  borderBottomWidth: 1,
  borderBottomColor: '#f0f0f0',
},
infoText: {
  flex: 1,
  marginLeft: 16,
  fontSize: 16,
},
logoutButton: {
  flexDirection: 'row',
  alignItems: 'center',
  paddingVertical: 16,
```

```
    paddingHorizontal: 20,
    marginTop: 20,
  },
  logoutText: {
    color: '#ff3b30',
    marginLeft: 16,
    fontSize: 16,
    fontWeight: '600',
  },
  versionText: {
    textAlign: 'center',
    marginTop: 'auto',
    marginBottom: 20,
    fontSize: 14,
    opacity: 0.5,
  },
});
```

Chat.tsx

```
import { CometChat } from '@cometchat/chat-sdk-react-native';
import { Ionicons } from '@expo/vector-icons';
import { Image } from 'expo-image';
import { Stack, useLocalSearchParams } from 'expo-router';
import React, { useEffect, useRef, useState } from 'react';
import { ActivityIndicator, FlatList, KeyboardAvoidingView, Platform, StyleSheet,
TextInput, TouchableOpacity } from 'react-native';

import { ThemedText } from '@/components/ThemedText';
```

```
import { ThemedView } from '@/components/ThemedView';


export default function ChatScreen() {
  const { uid, name, avatar } = useLocalSearchParams();
  const [messages, setMessages] = useState<any[]>([]);
  const [inputText, setInputText] = useState('');
  const [loading, setLoading] = useState(true);
  const [sending, setSending] = useState(false);
  const flatListRef = useRef<FlatList | null>(null);
  const [currentUser, setCurrentUser] = useState<any>(null);


  useEffect(() => {
    CometChat.getLoggedinUser().then(
      (user: any) => {
        setCurrentUser(user);
      },
      (error: any) => {
        console.log('Current user fetching failed with error:', error);
      }
    );
  const messagesRequest = new CometChat.MessagesRequestBuilder()
      .setUID(uid as string)
      .setLimit(50)
      .build();


    messagesRequest.fetchPrevious().then(
```

```
    (messages: any) => {
      console.log('Message list fetched:', messages);
      setMessages(messages);
      setLoading(false);
    },
    (error: any) => {
      console.log('Message fetching failed with error:', error);
      setLoading(false);
    }
  );
  const listenerID = "CHAT_SCREEN_MESSAGE_LISTENER";
  CometChat.addMessageListener(
    listenerID,
    new CometChat.MessageListener({
      onTextMessageReceived: (message: any) => {
        console.log("Text message received:", message);
        if (message.sender.uid === uid) {
          setMessages(prevMessages => [...prevMessages, message]);
        }
      },
    })
  );
  return () => {
    CometChat.removeMessageListener(listenerID);
  };
}, [uid]);
```

```
const sendMessage = () => {
  if (!inputText.trim()) return;

  setSending(true);
  const receiverID = uid as string;
  const messageText = inputText;
  const receiverType = CometChat.RECEIVER_TYPE.USER;
  const textMessage = new CometChat.TextMessage(
    receiverID,
    messageText,
    receiverType
  );

  CometChat.sendMessage(textMessage).then(
    (message: any) => {
      console.log('Message sent successfully:', message);
      setMessages(prevMessages => [...prevMessages, message]);
      setInputText('');
      setSending(false);
    },
    (error: any) => {
      console.log('Message sending failed with error:', error);
      setSending(false);
    }
  );
```

```
};

const formatTime = (timestamp: any) => {
  if (!timestamp) return '';
  const date = new Date(timestamp * 1000);
  return date.toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' });
};
if (loading) {
  return (
    <ThemedView style={styles.loadingContainer}>
      <ActivityIndicator size="large" color="#0a7ea4" />
    </ThemedView>
  );
}
return (
  <>
    <Stack.Screen
      options={{
        headerTitle: () => (
          <ThemedView style={styles.headerTitleContainer}>
            <Image
              source={{ uri: avatar as string ||
'https://www.gravatar.com/avatar/00000000000000000000000000000000?d=mp&f=y' }}
              style={styles.headerAvatar}
            />
            <ThemedText type="defaultSemiBold">{name}</ThemedText>
          </ThemedView>
```

```
    ),
    headerShown: true,
    headerBackTitle: 'Back',
  }}
/>
<KeyboardAvoidingView
  style={{ flex: 1 }}
  behavior={Platform.OS === 'ios' ? 'padding' : undefined}
  keyboardVerticalOffset={Platform.OS === 'ios' ? 90 : 0}
>
  <ThemedView style={styles.containerStyle}>
    {messages.length === 0 ? (
      <ThemedView style={styles.emptyContainer}>
        <ThemedText style={styles.emptyText}>No messages yet</ThemedText>
        <ThemedText style={styles.emptySubText}>
          Send a message to start the conversation
        </ThemedText>
      </ThemedView>
    ) : (
      <FlatList
        ref={flatListRef}
        data={messages}
        keyExtractor={(item) => item.id.toString()}
        onContentSizeChange={() => flatListRef.current?.scrollToEnd({ animated: true })}
        renderItem={({ item }) => {
          const isSender = item.sender.uid === currentUser?.uid;
```

```
return (
  <ThemedView
    style={[
      styles.messageContainer,
      isSender ? styles.sentMessage : styles.receivedMessage
    ]}
  >
    <ThemedView
      style={[
        styles.messageBubble,
        isSender ? styles.sentBubble : styles.receivedBubble
      ]}
    >
      <ThemedText style={[
        styles.messageText,
        isSender ? { color: '#fff' } : {}
      ]}>
        {item.text}
      </ThemedText>
      <ThemedText style={[
        styles.timeText,
        isSender ? { color: '#fff' } : {}
      ]}>
        {formatTime(item.sentAt)}
      </ThemedText>
```

```
        </ThemedView>
      </ThemedView>
    );
  }}
 />
)}

<ThemedView style={styles.inputContainer}>
 <TextInput
  style={styles.input}
  placeholder="Type a message..."
  placeholderTextColor="#888"
  value={inputText}
  onChangeText={setInputText}
  multiline
 />
 <TouchableOpacity
  style={styles.sendButton}
  onPress={sendMessage}
  disabled={sending || !inputText.trim()}
 >
  {sending ? (
   <ActivityIndicator size="small" color="#fff" />
  ) : (
   <Ionicons name="send" size={20} color="#fff" />
  )}
```

```
          </TouchableOpacity>
        </ThemedView>
      </ThemedView>
    </KeyboardAvoidingView>
  </>
 );
}

const styles = StyleSheet.create({
 loadingContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
 },
 containerStyle: {
  flex: 1,
 },
 headerTitleContainer: {
  flexDirection: 'row',
  alignItems: 'center',
 },
 headerAvatar: {
  width: 30,
  height: 30,
  borderRadius: 15,
  marginRight: 10,
```

```
},
emptyContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  padding: 20,
},
emptyText: {
  fontSize: 18,
  fontWeight: '600',
},
emptySubText: {
  textAlign: 'center',
  marginTop: 10,
  opacity: 0.7,
},
messageContainer: {
  marginVertical: 4,
  paddingHorizontal: 16,
},
sentMessage: {
  alignItems: 'flex-end',
},
receivedMessage: {
  alignItems: 'flex-start',
},
```

```
messageBubble: {
  maxWidth: '80%',
  paddingHorizontal: 12,
  paddingVertical: 8,
  borderRadius: 16,
},
sentBubble: {
  backgroundColor: '#0a7ea4',
  borderBottomRightRadius: 4,
},
receivedBubble: {
  backgroundColor: '#f0f0f0',
  borderBottomLeftRadius: 4,
},
messageText: {
  fontSize: 16,
  color: '#000',
},
timeText: {
  fontSize: 12,
  opacity: 0.7,
  alignSelf: 'flex-end',
  marginTop: 4,
},
inputContainer: {
  flexDirection: 'row',
```

```
    padding: 10,
    borderTopWidth: 1,
    borderTopColor: '#f0f0f0',
    alignItems: 'center',
  },
  input: {
    flex: 1,
    backgroundColor: '#f5f5f5',
    borderRadius: 20,
    paddingHorizontal: 16,
    paddingVertical: 8,
    marginRight: 10,
    maxHeight: 100,
  },
  sendButton: {
    backgroundColor: '#0a7ea4',
    width: 40,
    height: 40,
    borderRadius: 20,
    justifyContent: 'center',
    alignItems: 'center',
  },
});
```

Layout.tsx

```
import { CometChat } from '@cometchat/chat-sdk-react-native';
import { DarkTheme, DefaultTheme, ThemeProvider } from '@react-navigation/native';
```

```
import { useFonts } from 'expo-font';
import { Stack } from 'expo-router';
import { StatusBar } from 'expo-status-bar';
import { useEffect, useState } from 'react';
import 'react-native-reanimated';

import { COMETCHAT_CONSTANTS } from '@/constants/CometChatConfig';
import { useColorScheme } from '@/hooks/useColorScheme';

export default function RootLayout() {
  const colorScheme = useColorScheme();
  const [loaded] = useFonts({
    SpaceMono: require('../assets/fonts/SpaceMono-Regular.ttf'),
  });
  const [cometChatReady, setCometChatReady] = useState(false);

  useEffect(() => {
    const appSettings = new CometChat.AppSettingsBuilder()
      .subscribePresenceForAllUsers()
      .setRegion(COMETCHAT_CONSTANTS.REGION)
      .build();

    CometChat.init(COMETCHAT_CONSTANTS.APP_ID, appSettings).then(
      () => {
        console.log('CometChat initialization completed successfully');
        setCometChatReady(true);
```

```
    },
    (error: any) => {
      console.log('CometChat initialization failed with error:', error);
      setCometChatReady(true);
    }
  );
  return () => {
    CometChat.getLoggedinUser().then(
      (user) => {
        if (user) {
          CometChat.logout().then(
            () => console.log('Logout completed successfully'),
            (error: any) => console.log('Logout failed with error:', error)
          );
        }
      },
      () => {
        console.log('No logged in user to log out');
      }
    );
  };
}, []);
if (!loaded || !cometChatReady) {
  return null;
}
return (
```

```
    <ThemeProvider value={colorScheme === 'dark' ? DarkTheme : DefaultTheme}>
      <Stack>
        <Stack.Screen name="(auth)" options={{ headerShown: false }} />
        <Stack.Screen name="(tabs)" options={{ headerShown: false }} />
        <Stack.Screen name="+not-found" />
      </Stack>
      <StatusBar style="auto" />
    </ThemeProvider>
  );
}
```

Index.tsx

```
import { CometChat } from '@cometchat/chat-sdk-react-native';
import { Ionicons } from '@expo/vector-icons';
import { Image } from 'expo-image';
import { router } from 'expo-router';
import React, { useEffect, useState } from 'react';
import { ActivityIndicator, FlatList, StyleSheet, TouchableOpacity } from 'react-native';

import { HelloWave } from '@/components/HelloWave';
import ParallaxScrollView from '@/components/ParallaxScrollView';
import { ThemedText } from '@/components/ThemedText';
import { ThemedView } from '@/components/ThemedView';
import { SafeAreaView } from 'react-native-safe-area-context';

export default function HomeScreen() {
  const [conversations, setConversations] = useState<any[]>([]);
```

```
const [loading, setLoading] = useState(true);
useEffect(() => {
const conversationsRequest = new CometChat.ConversationsRequestBuilder()
    .setLimit(30)
    .build();
  conversationsRequest.fetchNext().then(
    (conversationList: any) => {
      console.log('Conversations list fetched successfully', conversationList);
      setConversations(conversationList);
      setLoading(false);},
    (error: any) => {
      console.log('Conversations list fetching failed with error:', error);
      setLoading(false);
    }
  );
}, []);
const navigateToChat = (conversation: any) => {
  const user = conversation.conversationWith;
  router.push({
    pathname: '/chat',
    params: {
      uid: user.uid,
      name: user.name,
      avatar: user.avatar || ''
    }
  });
```

```
};
 const formatTime = (timestamp: any) => {
  if (!timestamp) return '';
  const date = new Date(timestamp * 1000);
  const now = new Date();
  if (date.toDateString() === now.toDateString()) {
    return date.toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' });
  }
  const diffDays = Math.floor((now.getTime() - date.getTime()) / (1000 * 60 * 60 * 24));
  if (diffDays < 7) {
    return date.toLocaleDateString([], { weekday: 'short' });
  }
  return date.toLocaleDateString([], { month: 'short', day: 'numeric' });
};
if (loading) {
  return (
    <ThemedView style={styles.loadingContainer}>
      <ActivityIndicator size="large" color="#0a7ea4" />
    </ThemedView>
  );
}
return (
 <SafeAreaView
  style={{ flex: 1 }}>
    <ThemedView style={styles.titleContainer}>
      <ThemedText type="title">Welcome!</ThemedText>
```

```
          <HelloWave />
        </ThemedView>
        <ThemedView style={styles.container}>
          <ThemedView style={styles.header}>
            <ThemedText type="title">Chats</ThemedText>
            <TouchableOpacity style={styles.newChatButton} onPress={() =>
router.push('/contacts')}>
              <Ionicons name="create-outline" size={24} color="#0a7ea4" />
            </TouchableOpacity>
          </ThemedView>
          {conversations.length === 0 ? (
            <ThemedView style={styles.emptyContainer}>
              <Ionicons name="chatbubble-outline" size={80} color="#ccc" />
              <ThemedText style={styles.emptyText}>No conversations yet</ThemedText>
              <ThemedText style={styles.emptySubText}>
                Start a chat with someone from your contacts
              </ThemedText>
              <TouchableOpacity
                style={styles.newChatButtonLarge}
                onPress={() => router.push('/contacts')}>
                <ThemedText style={styles.newChatButtonText}>New Chat</ThemedText>
              </TouchableOpacity>
            </ThemedView>) : (
            <FlatList
              data={conversations}
              keyExtractor={(item) => item.conversationId}
              renderItem={({ item }) => {
```

```
                const user = item.conversationWith;
                const lastMessage = item.lastMessage?.text || 'New conversation'
                return (
                  <TouchableOpacity
                    style={styles.conversationItem}
                    onPress={() => navigateToChat(item)}
                    >
                    <Image
                      source={{ uri: user.avatar ||
'https://www.gravatar.com/avatar/00000000000000000000000000000000?d=mp&f=y' }}
                      style={styles.avatar}
                      />
                    <ThemedView style={styles.conversationContent}>
                      <ThemedView style={styles.conversationHeader}>
                        <ThemedText type="defaultSemiBold" numberOfLines={1}>
                          {user.name || user.uid}
                        </ThemedText>
                        <ThemedText style={styles.timeText}>
                          {formatTime(item.lastMessage?.sentAt)}
                        </ThemedText>
                      </ThemedView>
                      <ThemedText
                        numberOfLines={1}
                        style={styles.lastMessage}
                      >
                        {lastMessage}
                      </ThemedText>
```

```
                </ThemedView>
              </TouchableOpacity>
            );
          }}
          />
        )}
      </ThemedView>
    </SafeAreaView>
  );
}
const styles = StyleSheet.create({
  titleContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    gap: 8,
  },
  stepContainer: {
    gap: 8,
    marginBottom: 8,
  },
  reactLogo: {
    height: 178,
    width: 290,
    bottom: 0,
    left: 0,
    position: 'absolute',
```

```
},
container: {
  flex: 1,
  paddingTop: 20,
},
loadingContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
},
header: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  alignItems: 'center',
  paddingHorizontal: 0,
  paddingBottom: 20,
},
newChatButton: {
  padding: 10,
},
emptyContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  padding: 20,
},
```

```
emptyText: {
  fontSize: 18,
  marginTop: 20,
  fontWeight: '600',
},
emptySubText: {
  textAlign: 'center',
  marginTop: 10,
  opacity: 0.7,
},
newChatButtonLarge: {
  backgroundColor: '#0a7ea4',
  paddingVertical: 12,
  paddingHorizontal: 20,
  borderRadius: 8,
  marginTop: 20,
},
newChatButtonText: {
  color: 'white',
  fontWeight: '600',
},
conversationItem: {
  flexDirection: 'row',
  padding: 16,
  borderBottomWidth: 1,
  borderBottomColor: '#f0f0f0',
```

```
  },
  avatar: {
    width: 50,
    height: 50,
    borderRadius: 25,
    marginRight: 16,
  },
  conversationContent: {
    flex: 1,
    justifyContent: 'center',
  },
  conversationHeader: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    marginBottom: 6,
  },
  timeText: {
    fontSize: 12,
    opacity: 0.6,
  },
  lastMessage: {
    opacity: 0.7,
  },
});
```

Cometchat-config.ts

```
export const COMETCHAT_CONSTANTS = {
```

```
  APP_ID: '27792470fba3a203',
  REGION: 'IN',
  AUTH_KEY: 'bb7914b6f528f8f9a21763364e1f7992b9e87130',
  REST_API_KEY: 'b10604e0ea3cbdc7ce56e0cbdcc671565a2e6fc0',
};
export const SAMPLE_USERS = [
  {
    uid: 'superhero1',
    name: 'Iron Man',
    avatar: 'https://data-us.cometchat.io/assets/images/avatars/ironman.png',
  },
  {
    uid: 'superhero2',
    name: 'Captain America',
    avatar: 'https://data-us.cometchat.io/assets/images/avatars/captainamerica.png',
  },
  {
    uid: 'superhero3',
    name: 'Spiderman',
    avatar: 'https://data-us.cometchat.io/assets/images/avatars/spiderman.png',
  },
  {
    uid: 'superhero4',
    name: 'Wolverine',
    avatar: 'https://data-us.cometchat.io/assets/images/avatars/wolverine.png',
  },
```

```
];
```

Cometchat-d.ts

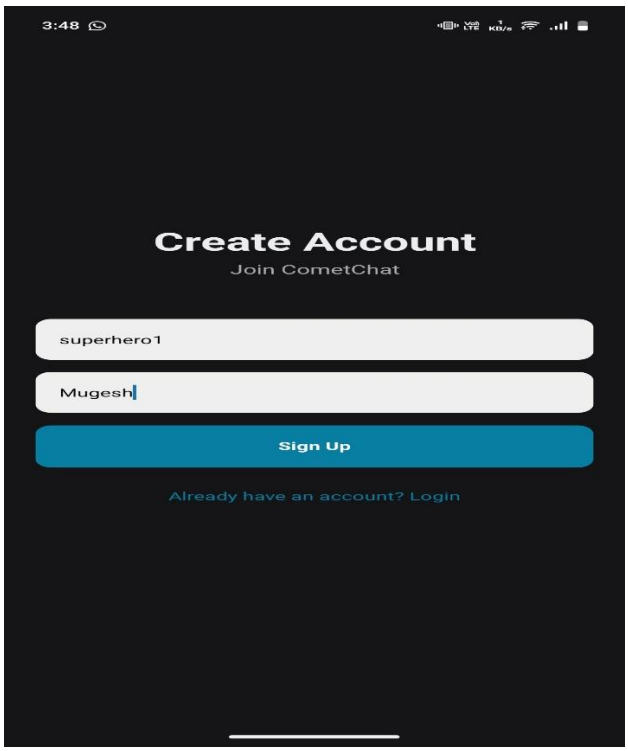
```
declare module '@cometchat/chat-sdk-react-native' {
 export namespace CometChat {
 export function init(appID: string, appSettings: any): Promise<string>;
 export function login(uid: string, authKey: string): Promise<User>;
 export function logout(): Promise<string>;
 export function getLoggedinUser(): Promise<User>;
 export function createUser(user: User, authKey: string): Promise<User>;
 export class User {
  constructor(uid: string, name?: string);
  uid: string;
  name?: string;
  avatar?: string;
  status?: string;
  statusMessage?: string;
  role?: string;
  static blockedUsersRequestBuilder(): BlockedUsersRequestBuilder;
 }
 export class TextMessage {
  constructor(receiverId: string, text: string, receiverType: string);
 }
 export class BlockedUsersRequestBuilder {
  build(): BlockedUsersRequest;
  setLimit(limit: number): BlockedUsersRequestBuilder;
  setSearchKeyword(keyword: string): BlockedUsersRequestBuilder;
```

```
}
export class BlockedUsersRequest {
  fetchNext(): Promise<User[]>;
}
export class UsersRequestBuilder {
  build(): UsersRequest;
  setLimit(limit: number): UsersRequestBuilder;
  setSearchKeyword(keyword: string): UsersRequestBuilder;
}
export class UsersRequest {
  fetchNext(): Promise<User[]>;
}
export class ConversationsRequestBuilder {
  build(): ConversationsRequest;
  setLimit(limit: number): ConversationsRequestBuilder;
  setConversationType(type: string): ConversationsRequestBuilder;
}
export class ConversationsRequest {
  fetchNext(): Promise<Conversation[]>;
}
export class Conversation {
  conversationId: string;
  conversationType: string;
  conversationWith: User | Group;
  lastMessage: BaseMessage;
  unreadMessageCount: number;
```

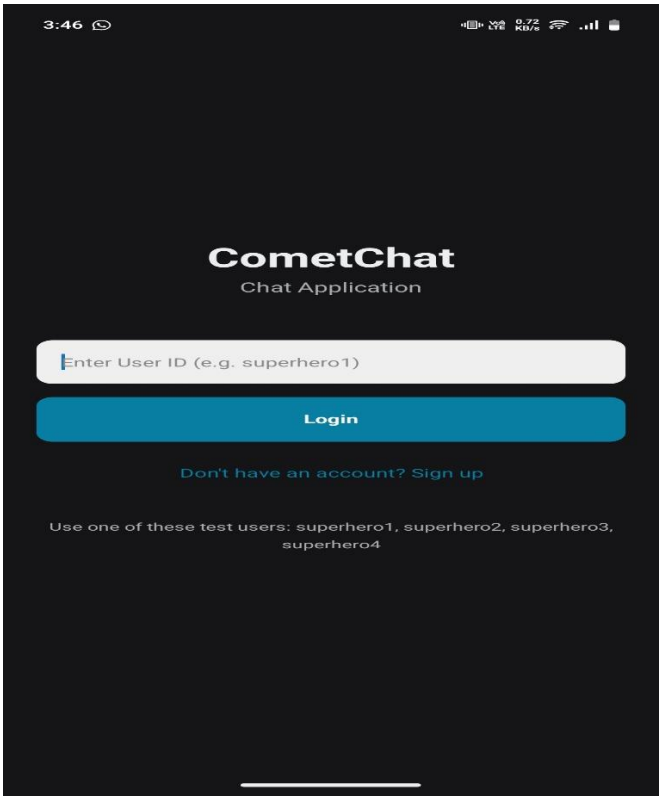
```
    }
    export class BaseMessage {
      id: string;
      sender: User;
      receiverId: string;
      receiverType: string;
      sentAt: number;
      deliveredAt: number;
      readAt: number;
    }
    export const RECEIVER_TYPE: {
      USER: string;
      GROUP: string;
    };
  }
}
declare module '@cometchat/chat-uikit-react-native' {
  export namespace CometChatUIKit {
    export class UIKitSettings {
      constructor(config: {
        appId: string;
        region: string;
        authKey: string;
      });
      subscribePresenceForAllUsers(): UIKitSettings;
      setAutoEstablishSocketConnection(flag: boolean): UIKitSettings;
```

```
    }
    export function init(config: UIKitSettings): Promise<string>
    export class CometChatConversationList extends React.Component<any> {}
    export class CometChatConversations extends React.Component<any> {}
    export class CometChatMessages extends React.Component<any> {}
    export class CometChatUserList extends React.Component<any> {}
    export class CometChatUI extends React.Component<any> {}
  }
}
```

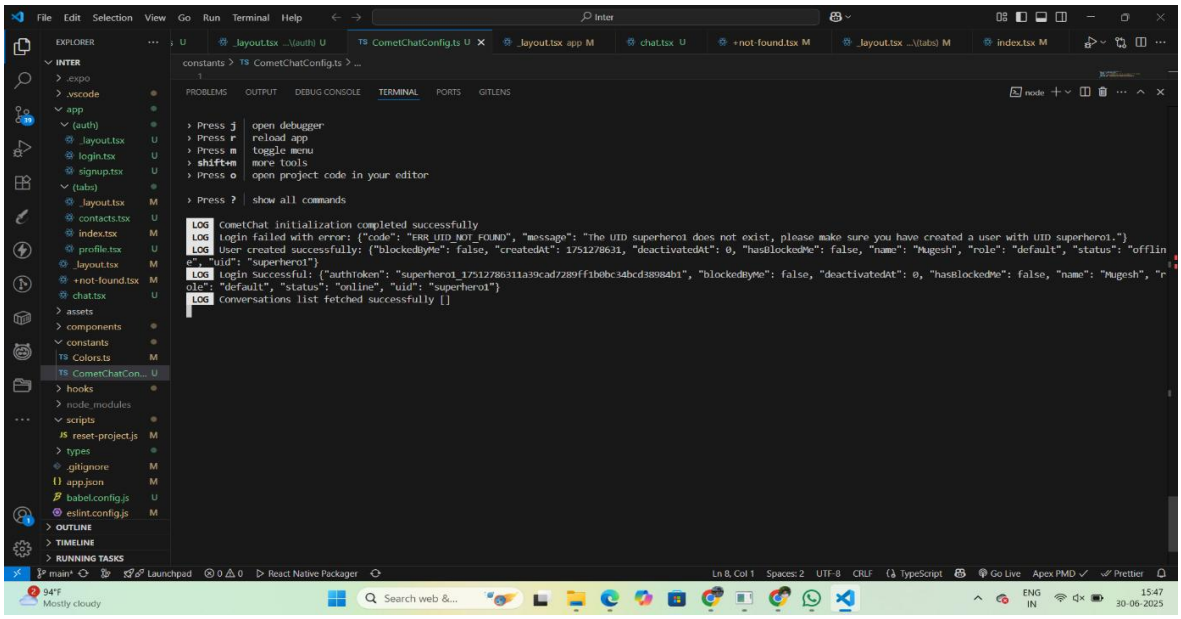
Output(screenshots)



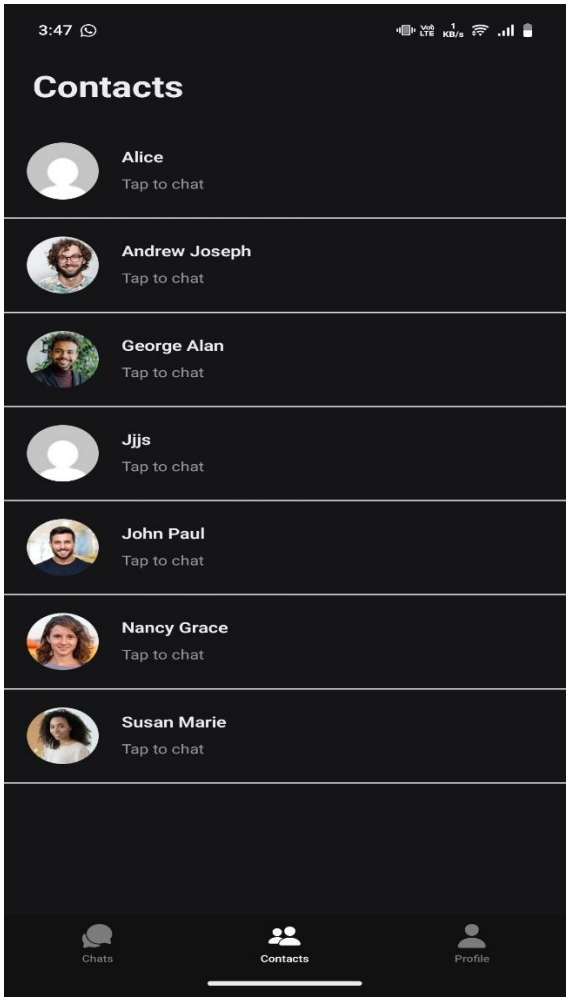
1.Signup Screen



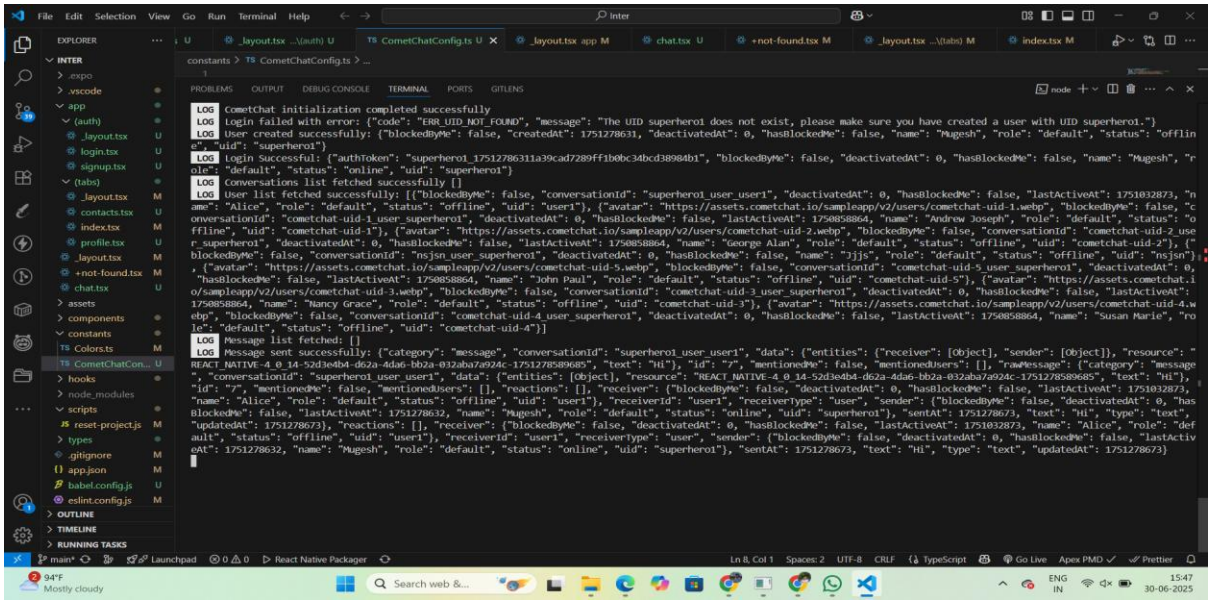
2.Login Screen



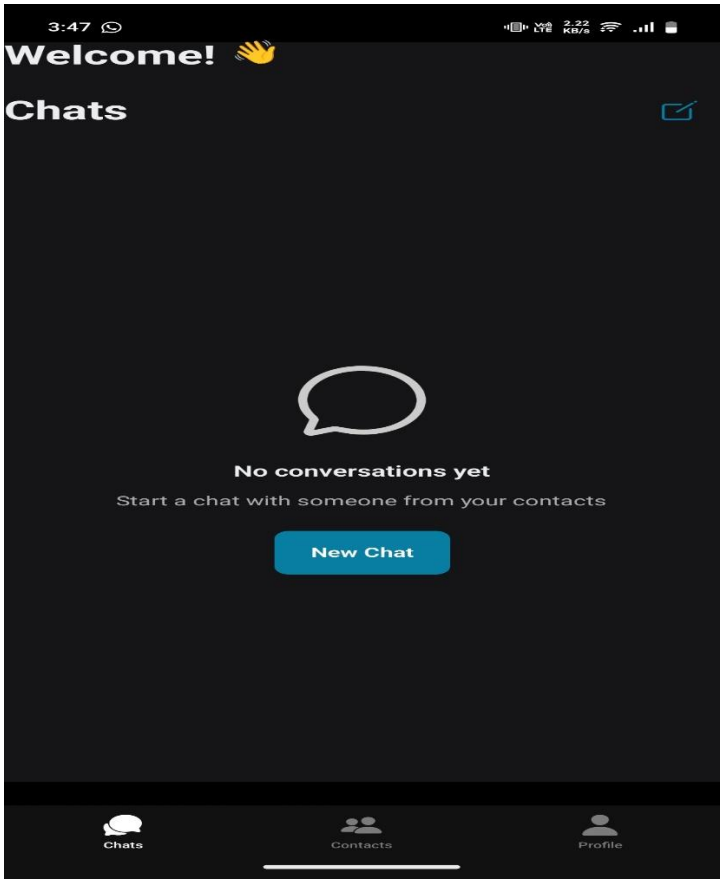
3.Console Output Screen



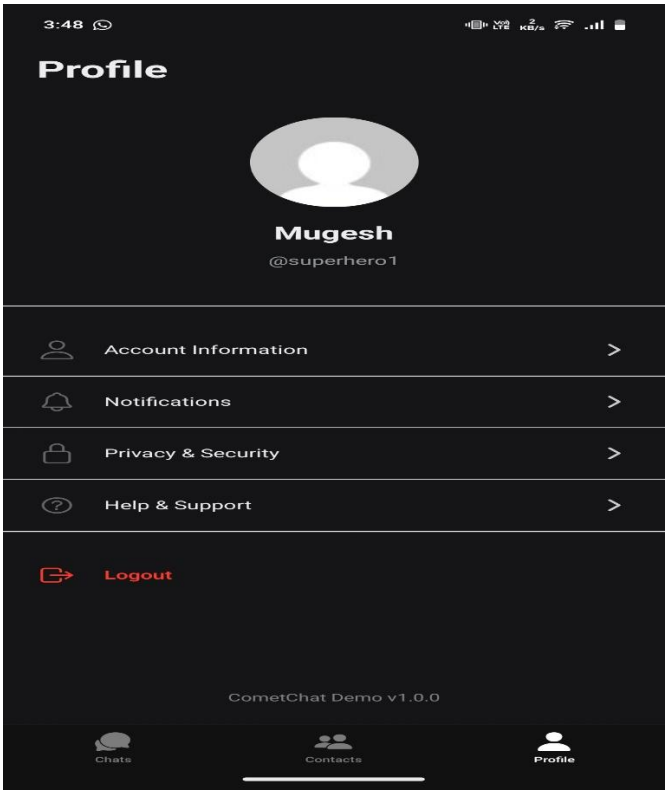
4.Contacts Screen



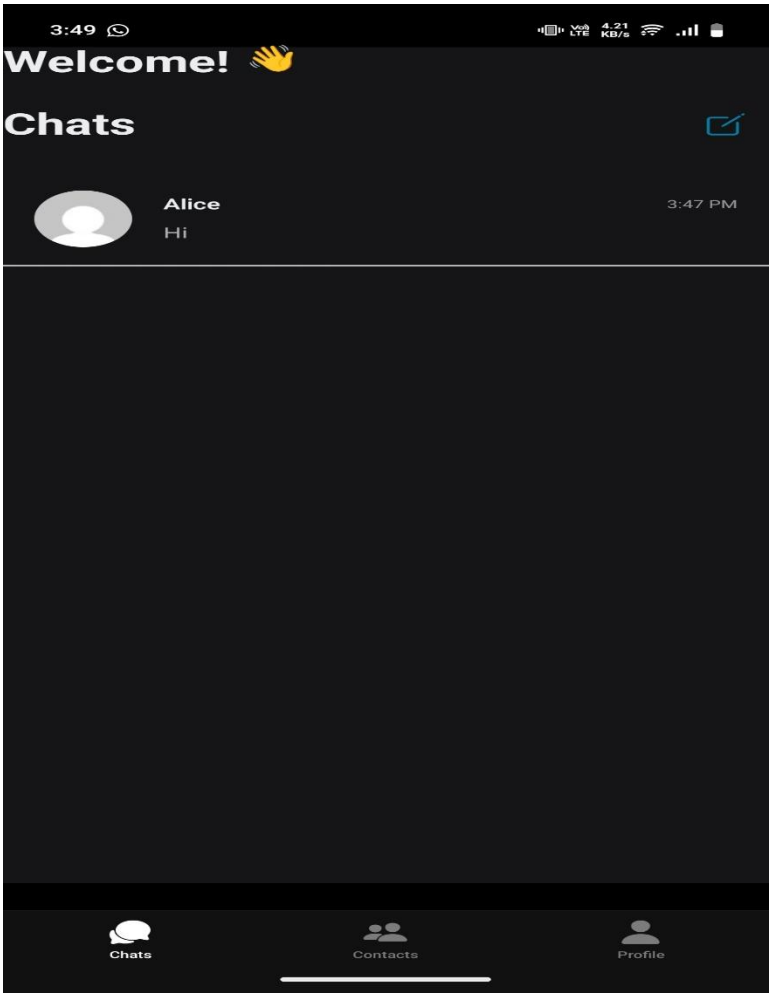
5.When message sent – Console Output



6.Chat Screen



7.Profile



8.Chat screen-After message

Issues Faced

- Issue: Initialization failed due to incorrect Region or App ID.
  Resolution: Double-checked the CometChat dashboard and corrected the region value as per my app's configuration.
- Issue: UI Kit styling overlap with my app theme.
  Resolution: Customized styles using available props and applied wrapper styles to harmonize appearance.
- Issue: Push notifications not working.
  Resolution: Not implemented due to time constraints, but explored Firebase setup for future integration.
- Lack of Prior Knowledge About CometChat
- At the beginning of this task, I had no prior experience with CometChat or real-time chat SDKs. I faced difficulty understanding the setup, integration process, and how the CometChat UI Kit works.
- However, by carefully going through the official documentation, watching tutorials, and experimenting with code, I was able to explore the features step by step and successfully implement the UI Kit in my application.
- This challenge helped me build confidence in integrating third-party SDKs and strengthened my ability to learn new technologies independently.

Applicant Details

Name: Mugesh

Email: vmugeshvmugesh24+test@gmail.com

Framework: React Native(TypeScript)

Final Output

- Login and start chatting
- View conversations in real-time
- Send/receive messages instantly

Conclusion

This internship task gave me a solid understanding of how to work with third-party SDKs like CometChat. It also helped me improve my skills in real-time communication app development and SDK integration best practices.

I look forward to applying these learnings in real-world projects and contributing to CometChat's mission.

GitHub Repository

https://github.com/mugesh713/CometChat-Intern.git