T.C.
MARMARA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

PROJECT NAME: *Instant Call*

Group Members

150115041 – Can Özgen

150115012 – Muhammet Mustafa Selimoğlu

150119938 – Muhammad Ismail

Supervised by

Prof.Dr. Ali Fuat Alkaya

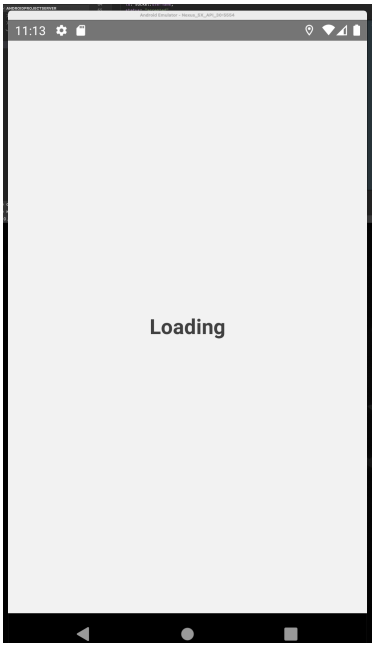
Abstract

In this project, we have created an application that enables the user to enter a username and once he/she logs in, there will be a map shown for him/her. Inside this map, the user can see the location that corresponds to him. He can see other users' locations as well. Thus, once the user enters his username, he will be added to the map and shown for other users that are already logged in. The location is instant, meaning that once the user moves around, its location in the app will be automatically updated. When the user chooses (clicks) one of them, a call screen will show up and the call to that person will be initiated (either video or voice). The user that should receive the call will see a pop-up call screen. He then can accept or reject the call. The technologies used in this application are sockets for communication between the two users, maps to show each user’s current location and other logged in users’ locations , WebRTC is the protocol that is used to issue the video call.

Mobile Application Structure

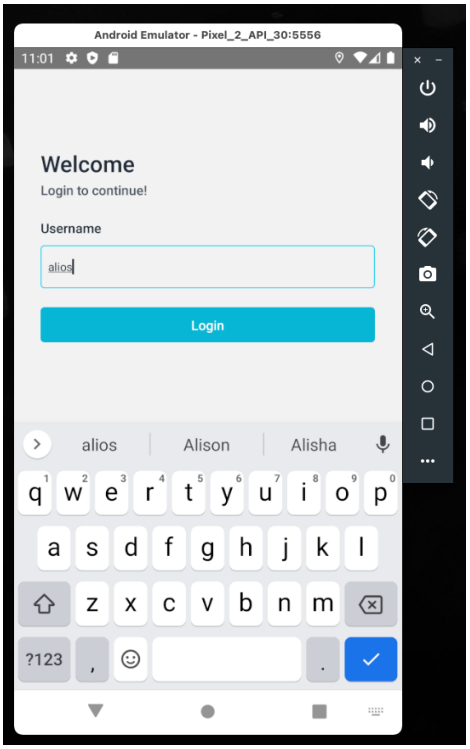
Application consists of 4 screens. These are Loading, Login, Match and Call screens.

Loading Screen



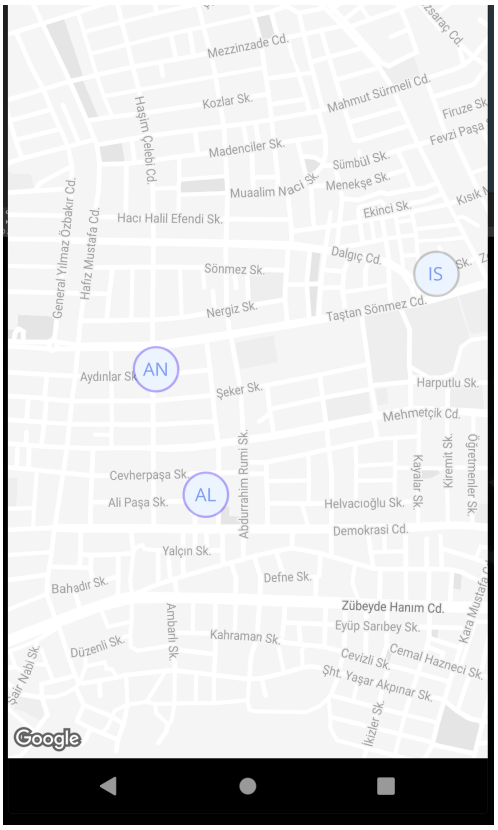
In this screen application establishes a socket connection with the server. After connection is created, app is redirecting itself to Login Screen.

Login Screen

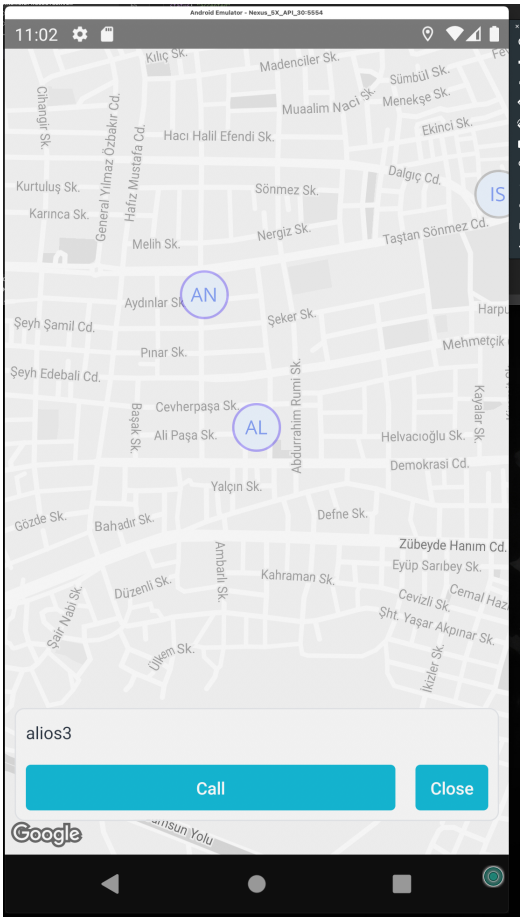


Login screen includes EditText and Login button to manage user identity. User should type him/her username to login to the system. Username should not exist on the system. Otherwise, user will face with an error “Username is already taken.”.

Match Screen

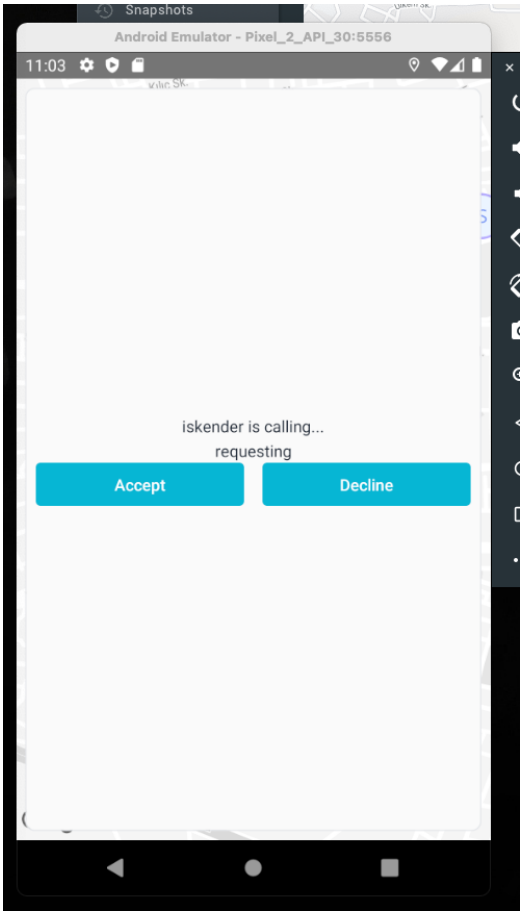


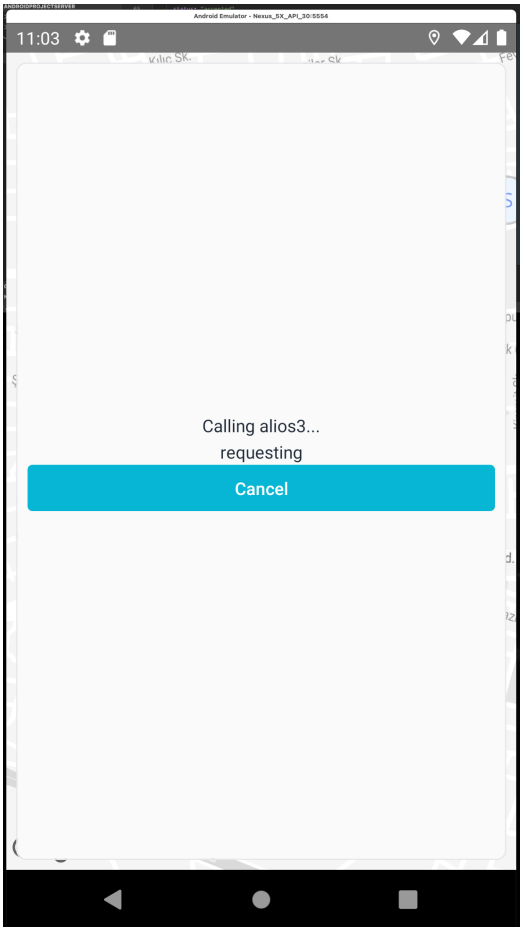
Match screen has a full screen Google Maps implementation. In this map, every online user is visible. Every user's coordinates are updated in real time with the help of the socket connection. Every user has an avatar which is generated from their usernames. The current user is indicated with a gray border. Every other user is indicated with a purple border.



If the user presses on the other users, he/she will get options popup to call other user.

If the user decides to initiate a call, the other user will see the screen on the bottom left. If the other user declines the call, both of them are notified about this action. If the requester cancels the call, both of them are notified too.

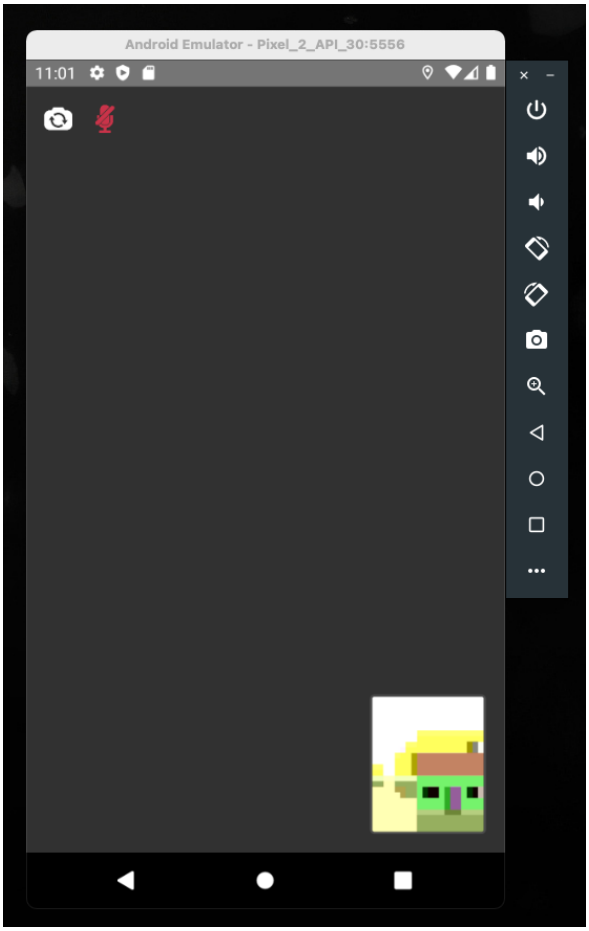


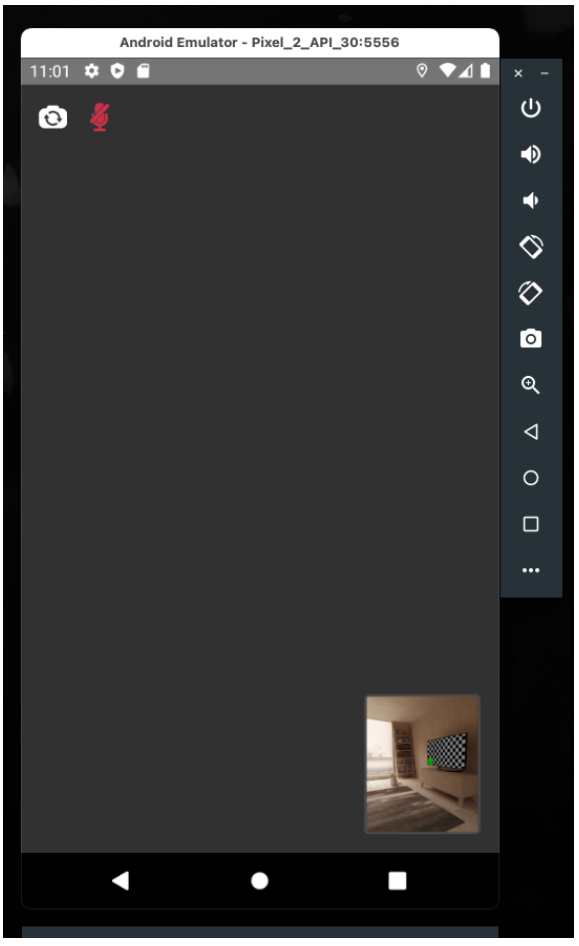


After the call is accepted both users are redirected to the call screen.

Call Screen:

In this screen, both users are creating their own peers. After creating their peers, they connect them together using our socket server. They send each other IceCandidates, Session Description Protocols. These session description protocols include the metadata of their local streams like width, height, supported codecs, opened udp ports etc. After the handshake is established they both start to see each other. On the top left corner, there is a camera switch button. If a device has 2 cameras, users can switch back and front cameras. There is a mute button on the right of the camera switch button. Users can mute their audio.





Permissions

These permissions are written in the AndroidManifest.xml. Camera, Microphone and Location permissions are asked in runtime for supporting android 16+.

1. ***android.permission.INTERNET*, *android.permission.ACCESS_NETWORK_STATE***

   This permission is required for establishing socket connection with the server. Network state is needed for understanding whether the device is connected to the internet or not.

2. ***android.permission.CAMERA*, *android.permission.RECORD_AUDIO*, *android.permission.MODIFY_AUDIO_SETTINGS***

   These permissions are required for accessing user's audio and video inputs.

3. ***android.permission.ACCESS_FINE_LOCATION*, *android.permission.ACCESS_COARSE_LOCATION***

   These permissions are required to access the device's current location and location changes.

4. ***android.permission.WAKE_LOCK***

   This permission is required for keeping the application awake. (Disables screen locks etc.)

React Native

By using RN, the app can run on the iOS platform too. We used Redux and ReduxThunk to manage application state. We have a Main Reducer and Socket Middleware.

Main Reducer keeps the state of current user, all users, focused user (when clicked on the map), call (call related information).

When a user moves we get location changes from GeoLocation Api and update the server about this change. Server emits this change to all connected sockets.

App handles video operations using the “react-native-webrtc” package, socket connections using the “socket.io-client” package, navigation using the “react-navigation” package, state using the “redux”, “redux-thunk” package, maps using the “react-native-maps” and “react-native-geolocation-service” packages, images using the “react-native-svg” package.

Backend Structure

Backend server is written in NodeJS. SocketIO and Express modules used for socket connections, user management and signaling.

Conclusion

In this project, We have created an application that can find each other when they are logged in using their current locations. In addition, they can initiate a call for the user they want once they click on it.