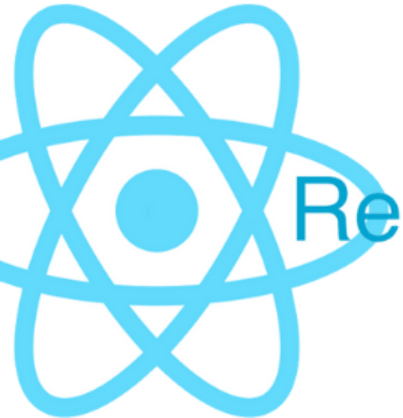
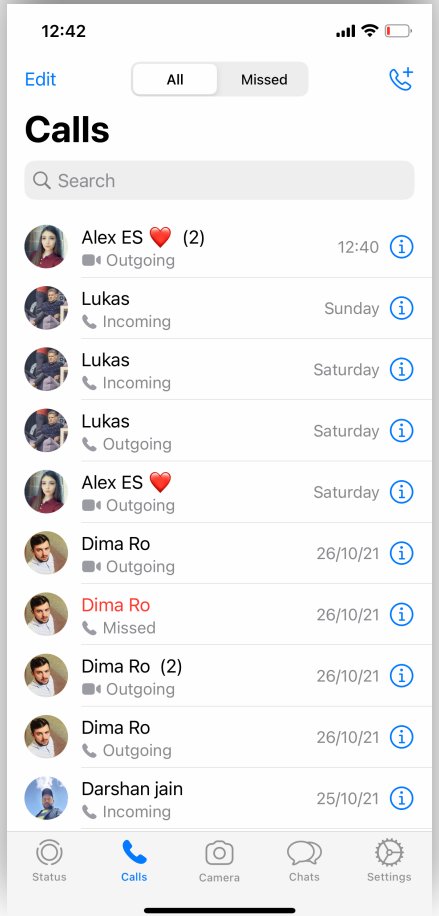
VIDEO

Calling App

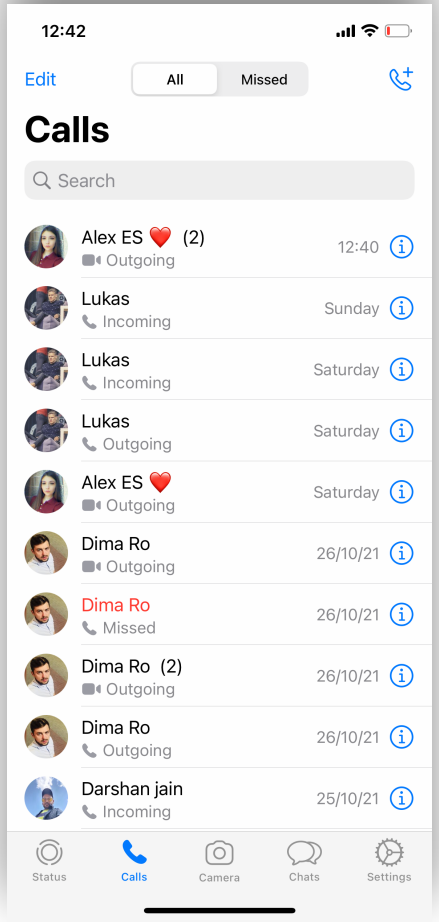


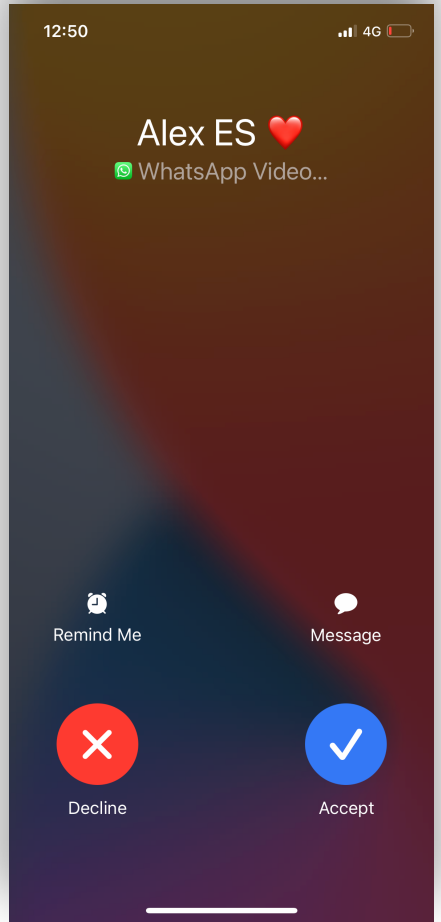
React Native

We are going to build

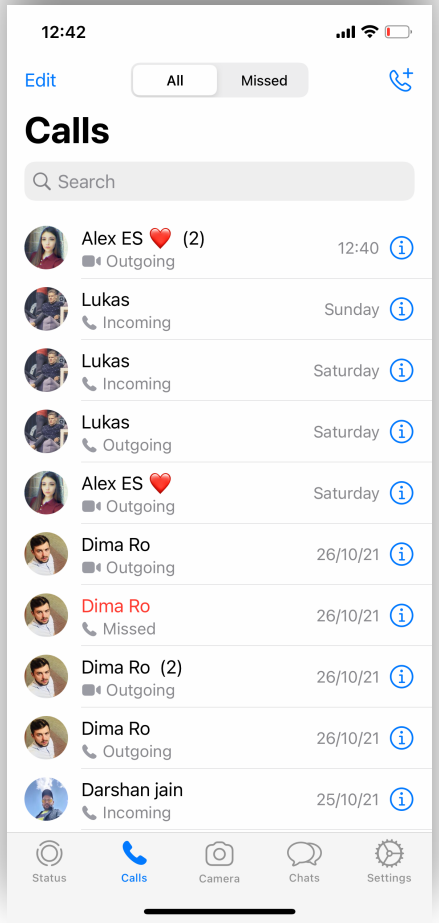


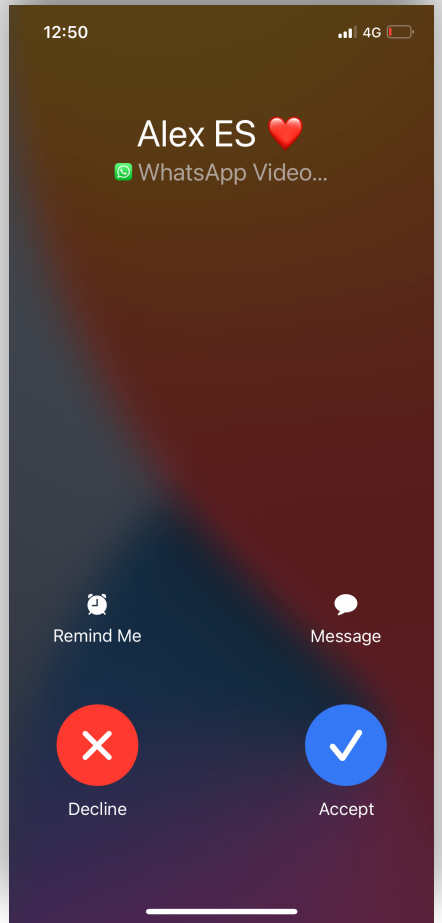
We are going to build

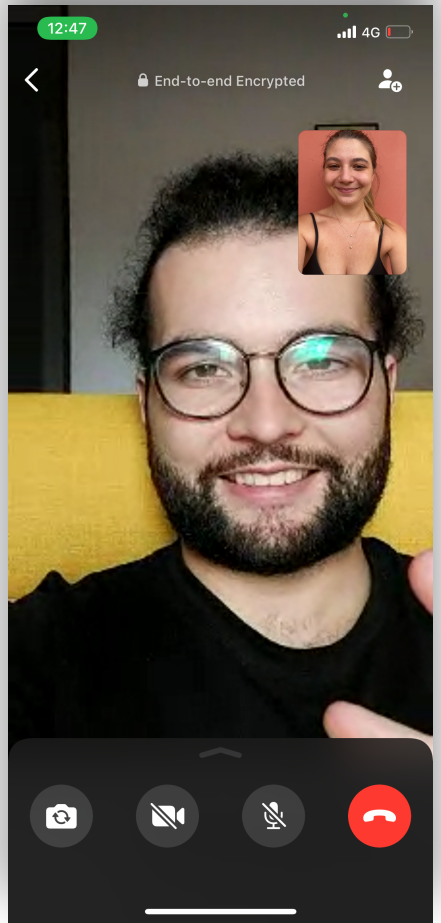




We are going to build





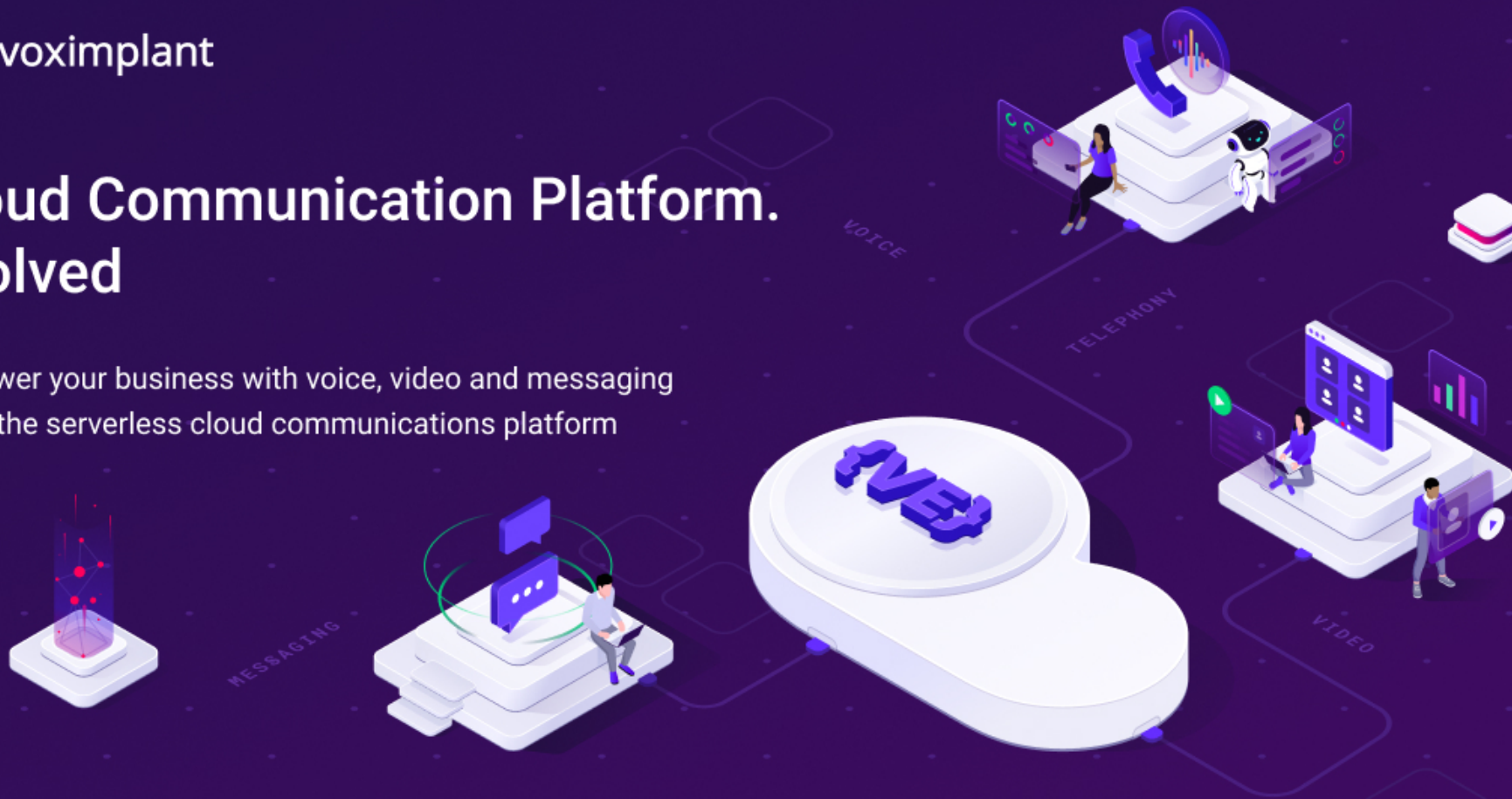


Video calling app Series: Fullstack tutorial

voximplant

Cloud Communication Platform. Evolved

Empower your business with voice, video and messaging using the serverless cloud communications platform

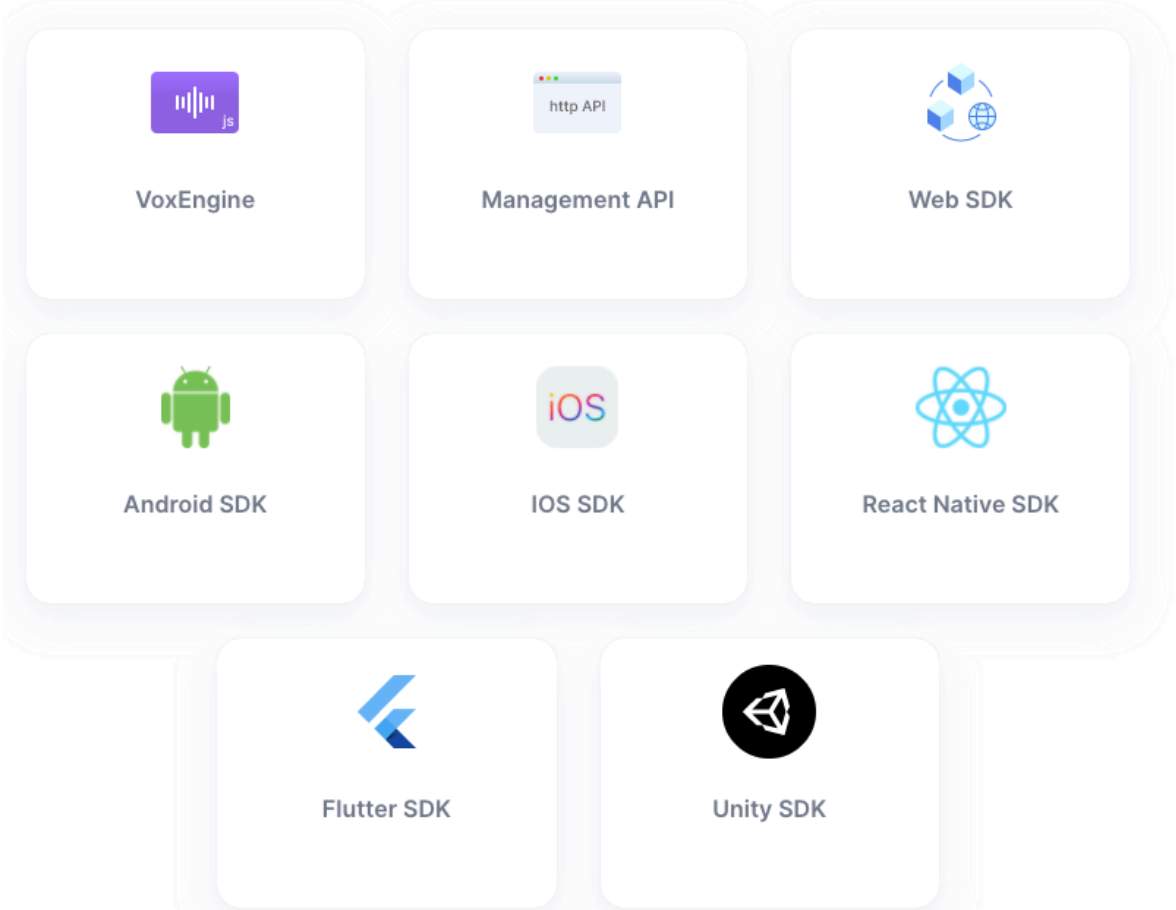


notJust
//development

What can be done with Voximplant?

- **virtual call center**
- **audio & video conferencing** apps like clubhouse, zoom, meets or teams.
- **peer-to-peer audio & video calling** apps like whatsapp, FaceTime, Skype, Viber.
- **Live streaming** apps like Twitch and Youtube

APIs and SDKs



notJust
//development

Prerequisites

1. Asset Bundle (dummy data, images, icons, PDF presentation, unlimited karma):
https://assets.notjust.dev/video-call

2. React Native environment setup
https://youtu.be/oorfevovPWw

notJust

//development

Let’s get started

Let’s get started

1. Initialise the React Native project:

```
$ npx react-native init VideoCall
```

2. Open project

```
$ cd VideoCall
```

```
$ code .
```

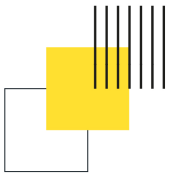
3. Run the development server

```
$ npm start
```

4. Run the app on device or emulator

```
$ npm run android
```

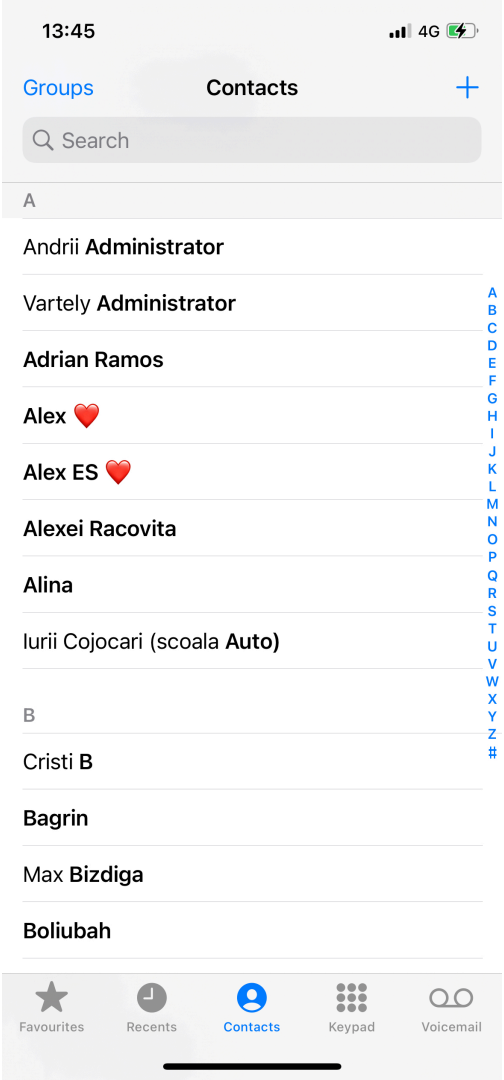
```
$ npm run iOS
```



notJust

//development

List of contacts



notJust

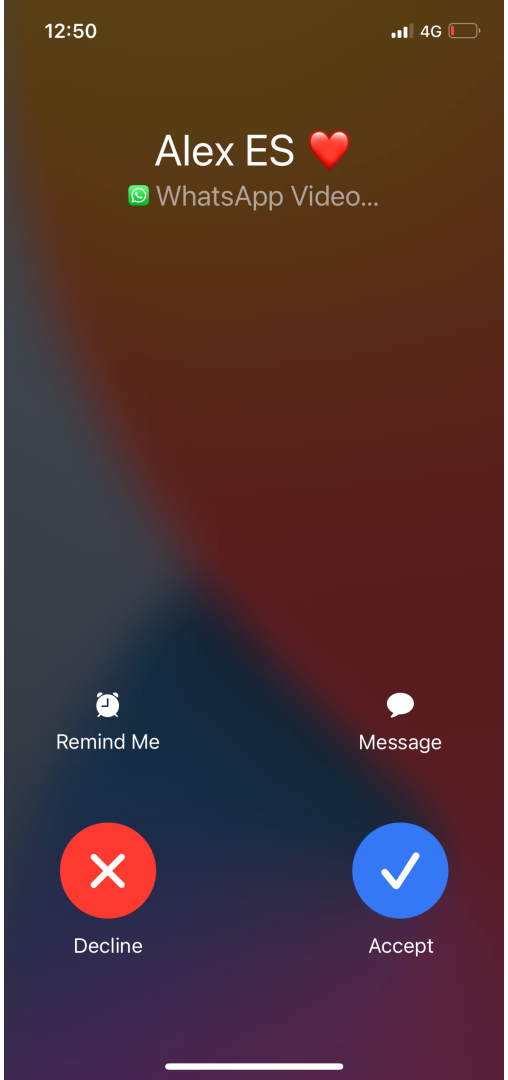
//development

Calling screen

notJust

//development

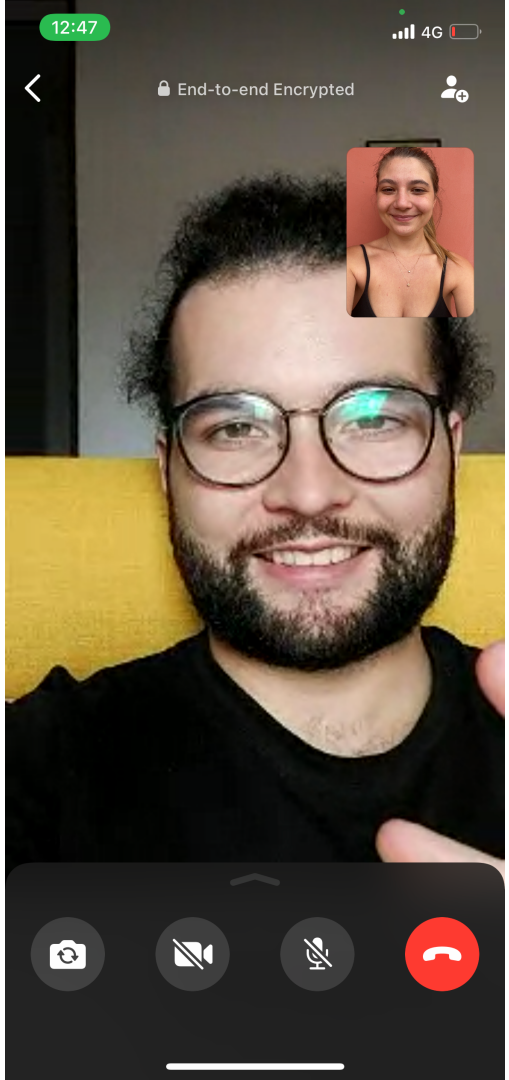
Incoming call screen



notJust

//development

Call screen



notJust

//development

Navigation

notJust

//development

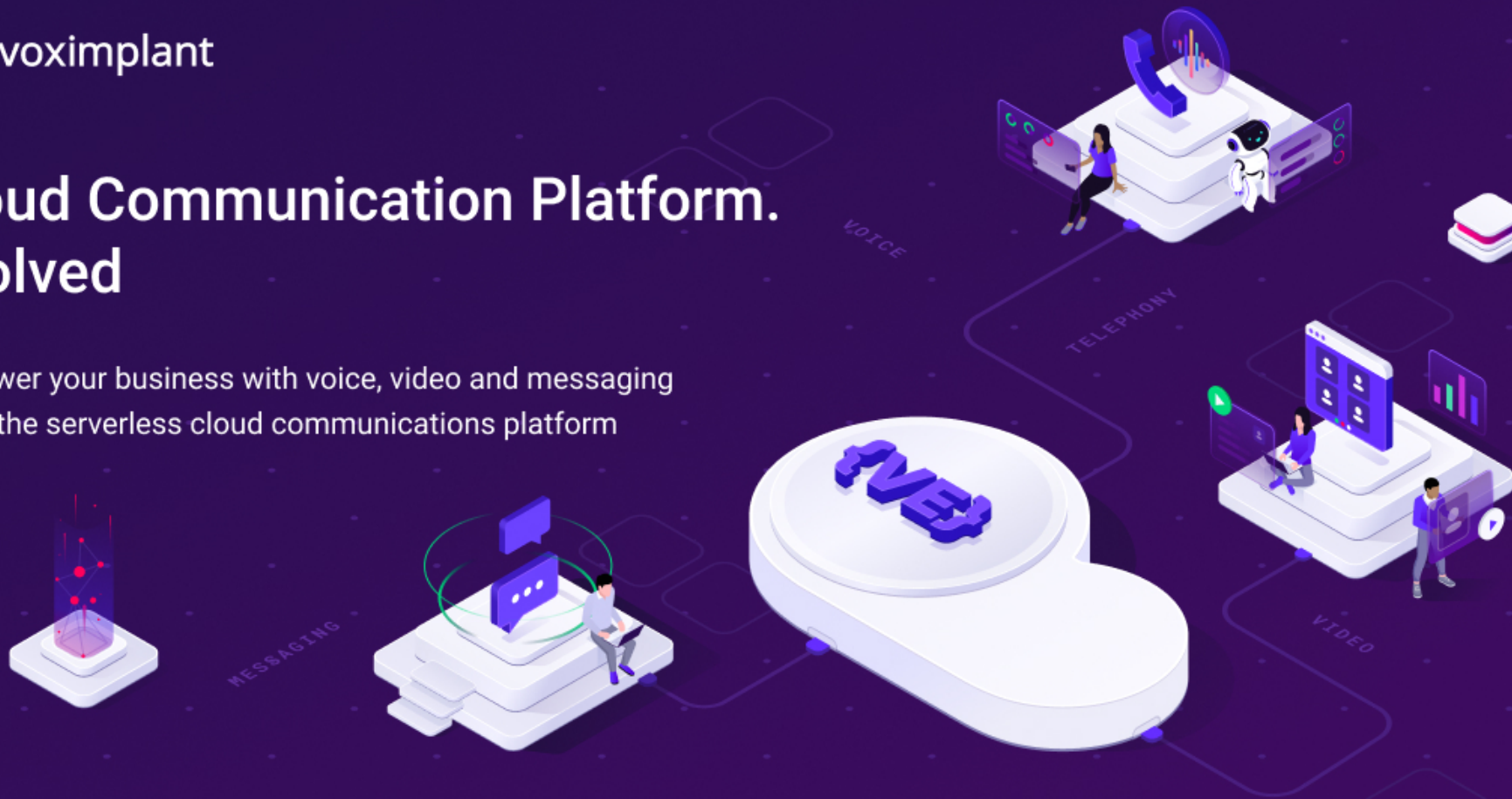
Q&A

Video calling app Series: Fullstack tutorial

voximplant

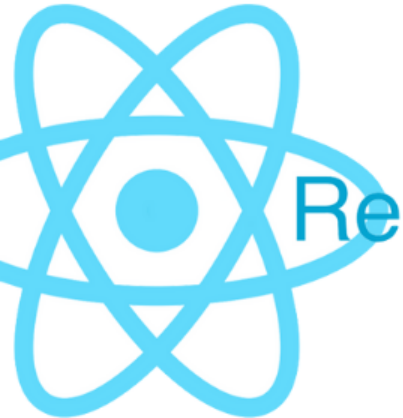
Cloud Communication Platform. Evolved

Empower your business with voice, video and messaging using the serverless cloud communications platform



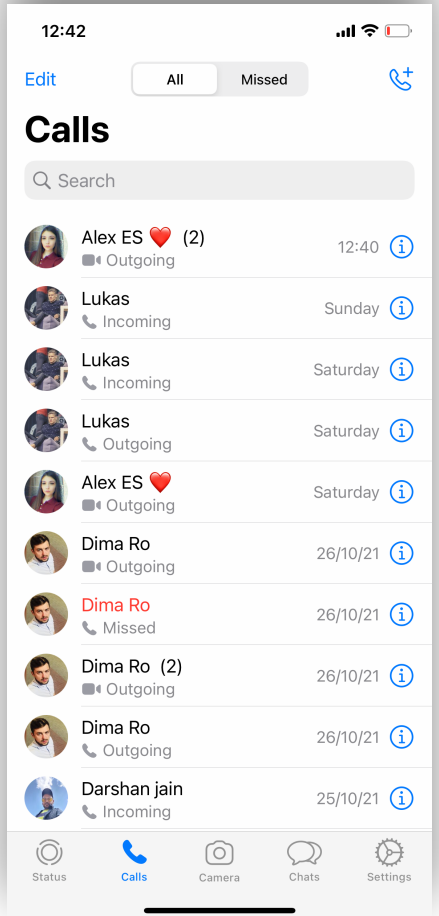
VIDEO

Calling App

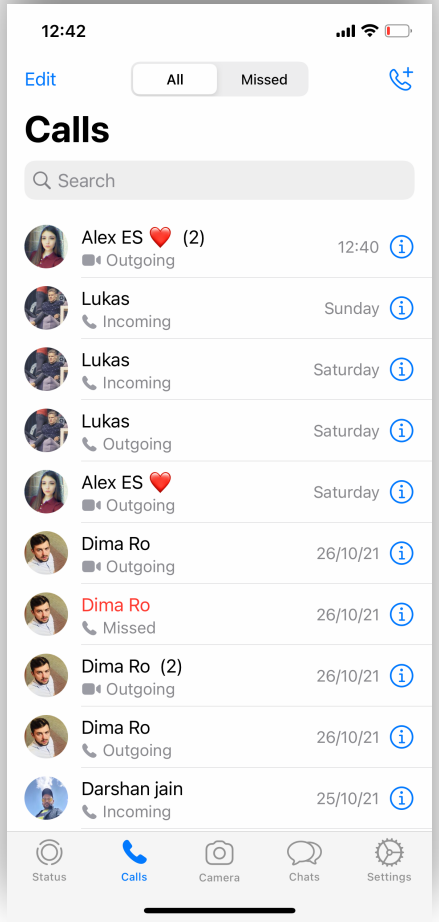


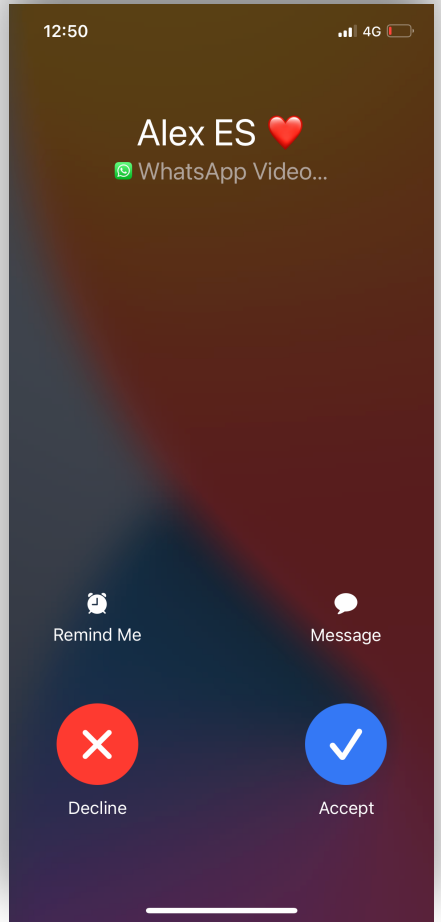
React Native

We are going to build

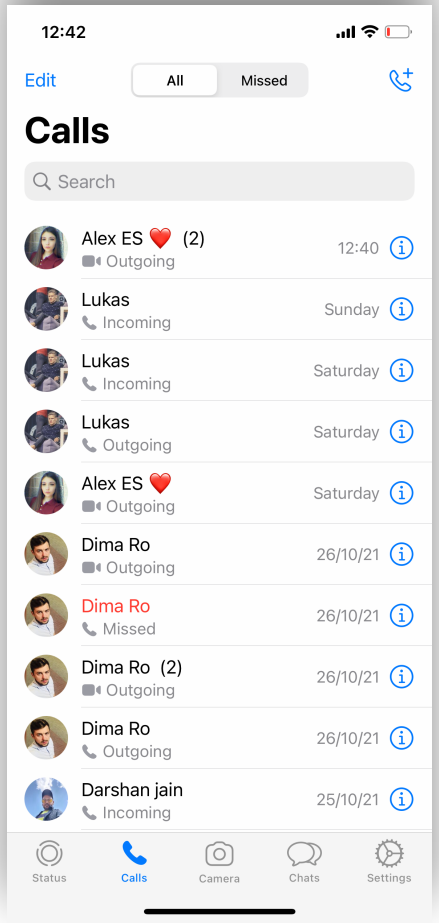


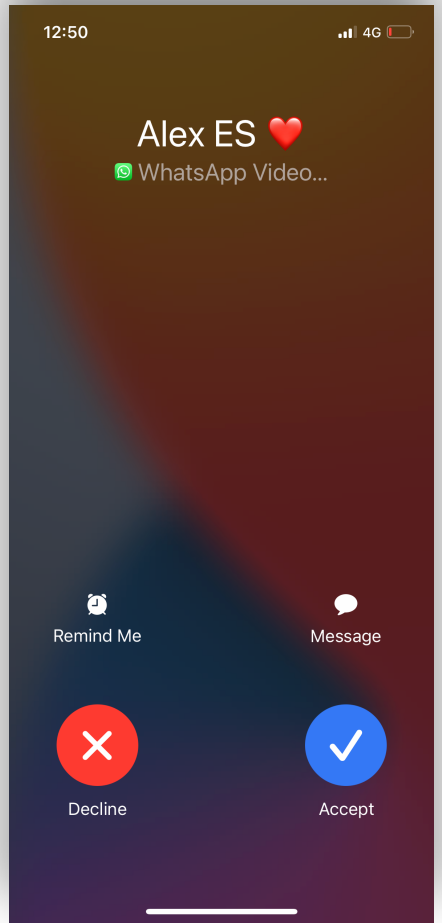
We are going to build

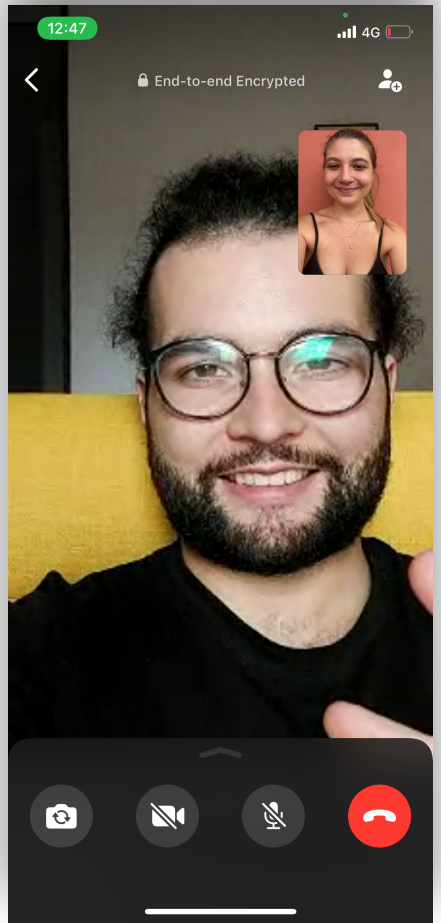




We are going to build



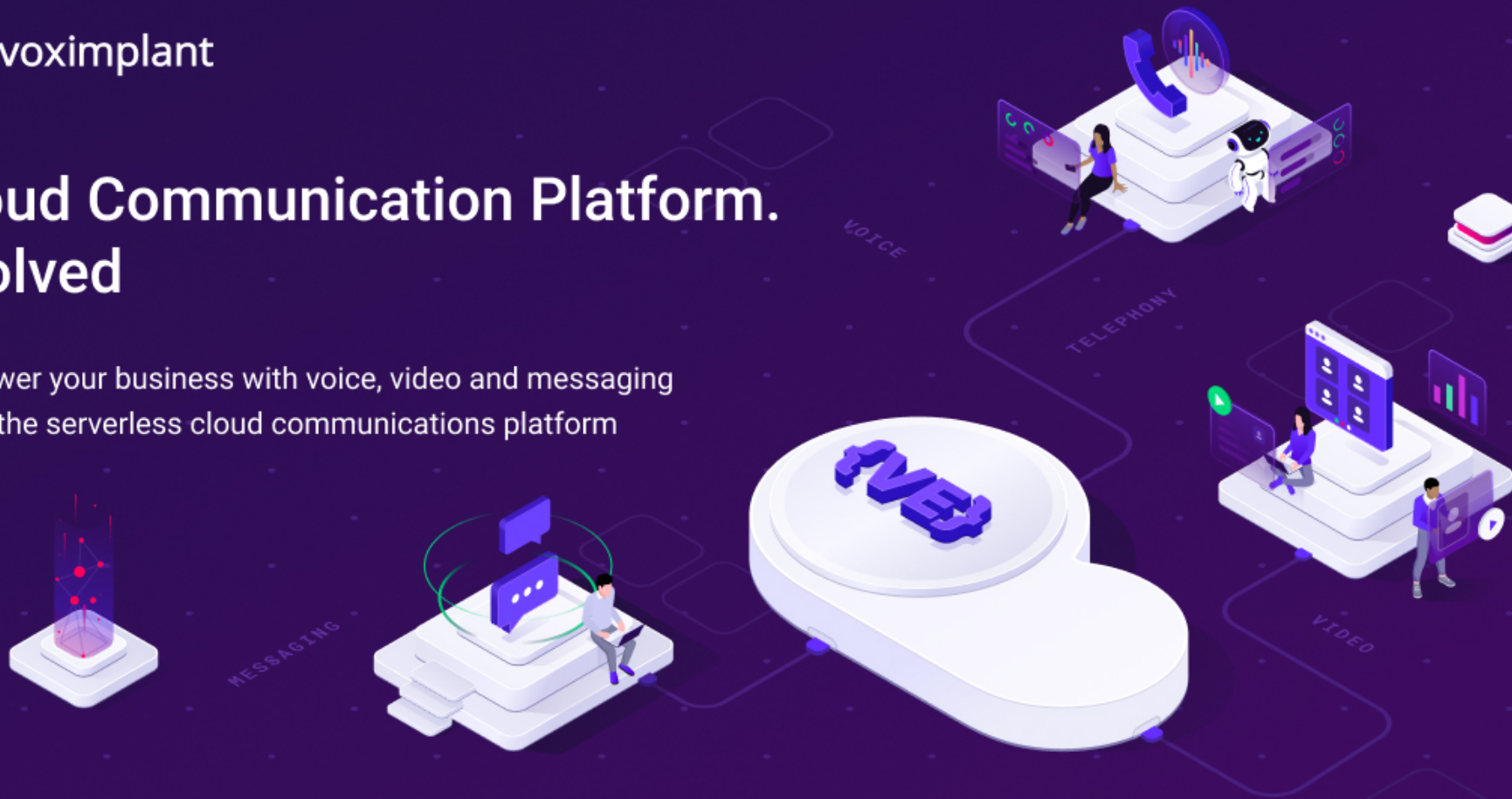




voximplant

Cloud Communication Platform. Evolved

Empower your business with voice, video and messaging using the serverless cloud communications platform

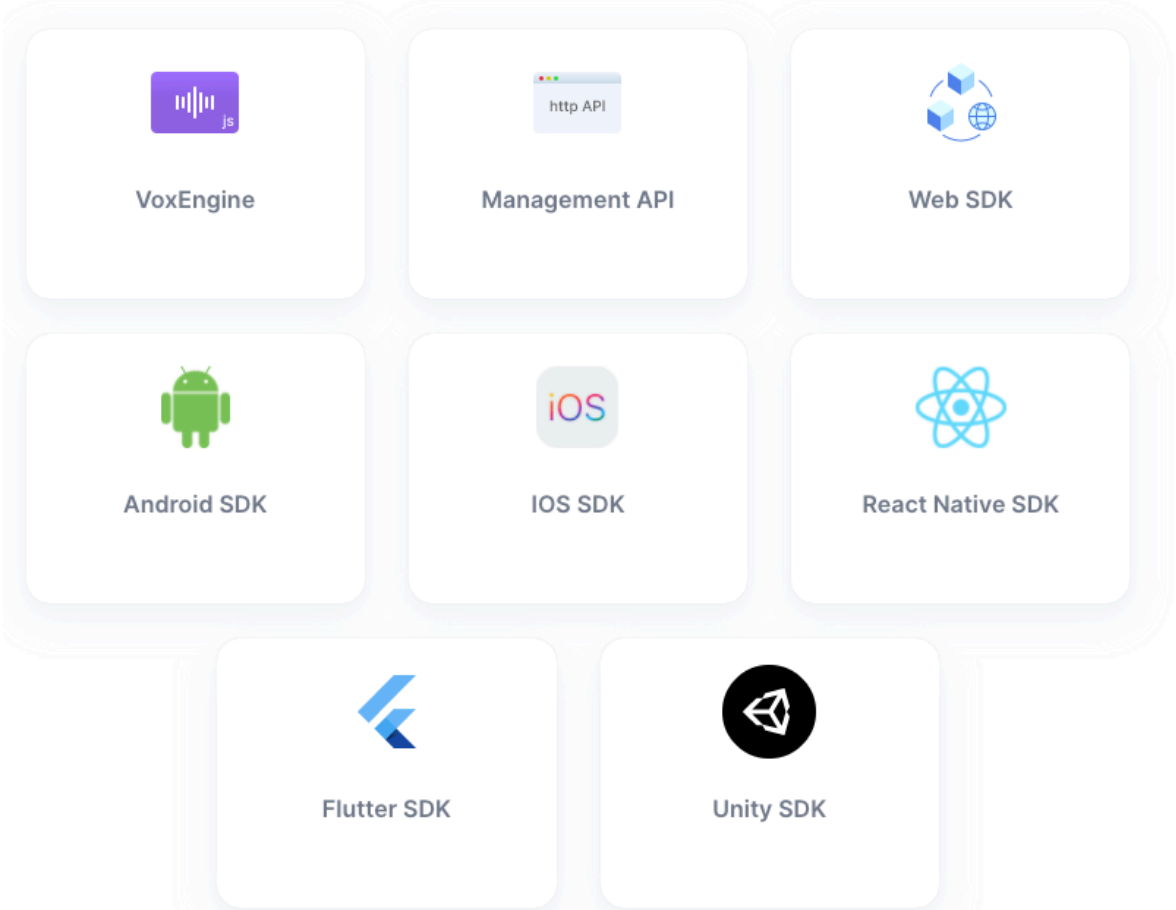


notJust
//development

What can be done with Voximplant?

- **virtual call center**
- **audio & video conferencing** apps like clubhouse, zoom, meets or teams.
- **peer-to-peer audio & video calling** apps like whatsapp, FaceTime, Skype, Viber.
- **Live streaming** apps like Twitch and Youtube

APIs and SDKs



Pricing

	VOICE, PRICE PER MINUTE	+ VIDEO, PRICE PER MIB
SDK to Platform	$0.002	$0.00005
Platform to SDK	$0.002	$0.00005
SDK to SDK Peer-to-peer calls	Free*	Free*
TURN Peer-to-peer calls	–	$0.0005

Monthly Active Users

Free

up to 1000 MAU**

FREE

Additional users are not available

Small

up to 5K MAU**

$100 / month

+$0.03 / for each additional user

Medium

up to 80K MAU**

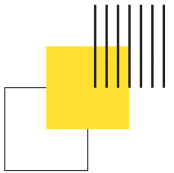
$1000 / month

+$0.02 / for each additional user

Custom

More than 1M MAU**

CONTACT US



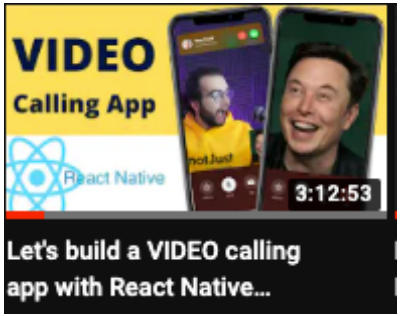
notJust
//development

Prerequisites

1. Asset Bundle (dummy data, images, icons, PDF presentation, unlimited karma):
https://assets.notjust.dev/video-call

2. React Native environment setup
https://youtu.be/oorfevovPWw

notJust
//development



https://youtu.be/rb70_TXRQNE

Starting point

1. Part 1 (UI)

Didn't follow part 1, but want to follow along now?

2. Clone the repo (UI branch): https://github.com/Savinvadim1312/VideoCallVoximplant/tree/ui
3. Install dependencies (npm install, npx pod-install)
4. Run the project (npm start)

notJust

//development

Let’s get started

notJust
//development

Setup Voximplant application

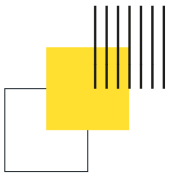
1. Sign up for a free account here
2. Create an Application
3. Add 2 test Users

Voximplant scenario

“A scenario is a JavaScript code document that manages calls within the cloud platform.

The code is executed automatically upon receiving an incoming call.“

```
VoxEngine.addEventListener(AppEvents.CallAlerting, (e) => {
  const newCall = VoxEngine.callUserDirect(
    e.call,
    e.destination,
    e.callerid,
    e.displayName,
    null
  );
  VoxEngine.easyProcess(e.call, newCall, ()=>{}, true);
});
```



notJust
//development

Routing rule

Create rule

Name

intercom

Video conference

Pattern

.*

Available Scenarios (optional)

Choose scenario from the list to assign scenario

Assigned Scenarios (optional)

User2User

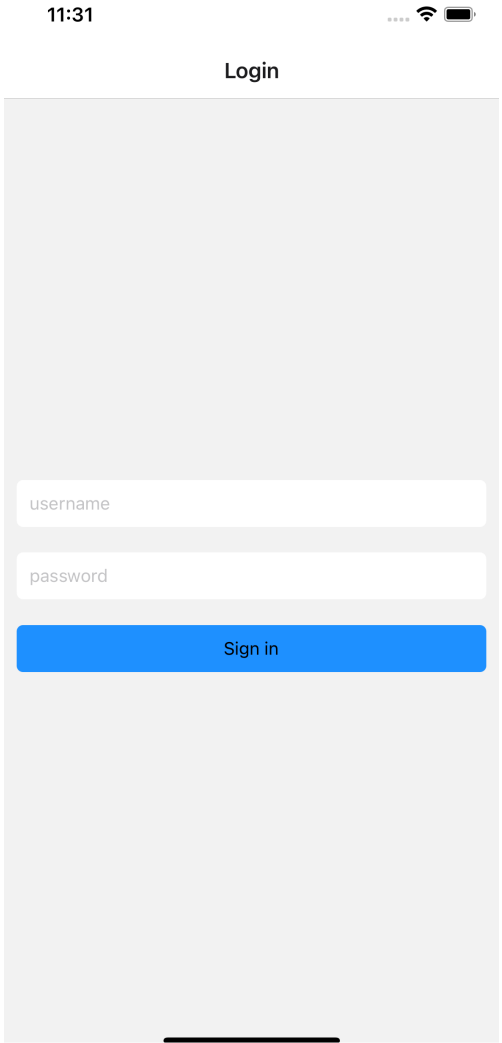
Create rule

notJust

//development

Login UI

1. Username Input
2. Password Input
3. Button
4. Navigate on press



notJust
//development

Install Voximplant

https://www.npmjs.com/package/react-native-voximplant

1. Install package

```
$ npm install react-native-voximplant
```

2. Install pods (macOS only)

```
$ npx pod-install
```

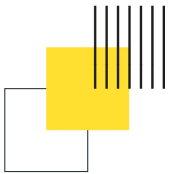
Connect Voximplant client

1. Get the Voximplant instance

```
const voximplant = Voximplant.getInstance();
```

2. Connect the client

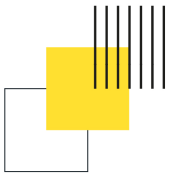
```
const connectVoximplant = async () => {
    let clientState = await voximplant.getClientState();
    if (clientState === Voximplant.ClientState.DISCONNECTED) {
      await voximplant.connect();
    } else if (clientState === Voximplant.ClientState.LOGGED_IN) {
      navigation.reset({index: 0, routes: [{name: 'Contacts'}]});
      return;
    }
  };
```



Voximplant Login

1. Login with the fully-qualified username and password.
   Fully-qualified username: username@appname.accname.voximplant.com

```
const login = async () => {
  try {
    await voximplant.login(
      `${username}@test2.savinvadim.voximplant.com`,
      password,
    );
    navigation.reset({index: 0, routes: [{name: 'Contacts'}]});
  } catch (e) {
    Alert.alert(e.name, `Error code: ${e.code}`);
  }
};
```



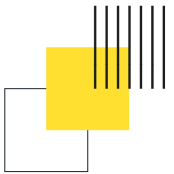
notJust

//development

Calls

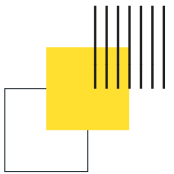
Android permissions

```
const requestPermissions = async () => {
    const granted = await PermissionsAndroid.requestMultiple(permissions);
    const recordAudioGranted =
      granted[PermissionsAndroid.PERMISSIONS.RECORD_AUDIO] === 'granted';
    const cameraGranted =
      granted[PermissionsAndroid.PERMISSIONS.CAMERA] === 'granted';
    if (!cameraGranted || !recordAudioGranted) {
      Alert.alert('Permissions not granted');
    } else {
      setPermissionGranted(true);
    }
  };

  useEffect(() => {
    if (Platform.OS === 'android') {requestPermissions()}
    else {setPermissionGranted(true);}
  }, []);
```



Make a call

```
const callSettings = {
      video: {
        sendVideo: true,
        receiveVideo: true,
      },
    };

const makeCall = async () => {
  call.current = await voximplant.call(user.user_name, callSettings);
};
```



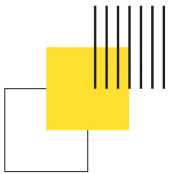
Call events

1. Call Failed

```
call.on(Voximplant.CallEvents.Failed, callEvent => {
        showCallError(callEvent.reason);
    });
```

2. Call Progress started

```
call.on(Voximplant.CallEvents.ProgressToneStart, callEvent => {
        setCallState('Ringing...');
    });
```

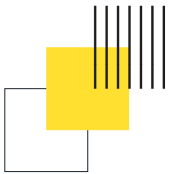


3. Call Connected

```
call.on(Voximplant.CallEvents.Connected, callEvent => {
    setCallState('Call connected');
});
```

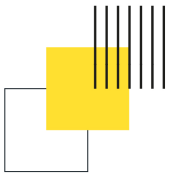
4. Call Disconnected

```
call.on(Voximplant.CallEvents.Disconnected, callEvent => {
 navigation.navigate('Contacts');
});
```



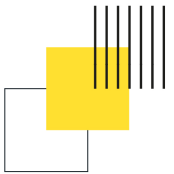
Incoming Call event

```
useEffect(() => {
    voximplant.on(Voximplant.ClientEvents.IncomingCall, incomingCallEvent => {
      navigation.navigate('IncomingCall', {
        call: incomingCallEvent.call,
      });
    });
    return () => {
      voximplant.off(Voximplant.ClientEvents.IncomingCall);
    };
  });
```



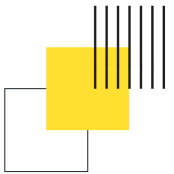
Incoming Call Screen

```
useEffect(() => {
    setCaller(call.getEndpoints()[0].displayName);
    call.on(Voximplant.CallEvents.Disconnected, callEvent => {
      navigation.navigate('Contacts');
    });
    return () => {
      call.off(Voximplant.CallEvents.Disconnected);
    };
  }, [call]);

  const onDecline = () => { call.decline() };

  const onAccept = () => {
    navigation.navigate('Calling', {call: call, isIncomingCall: true});
  };
```



Answer call

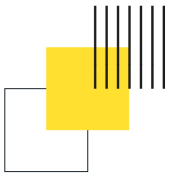
```
const answerCall = async () => {
      subscribeToCallEvents();
      await call.current.answer(callSettings);
    };
```



Your video

```
call.current.on(Voximplant.CallEvents.LocalVideoStreamAdded, callEvent => {
        setLocalVideoStreamId(callEvent.videoStream.id);
      });
```

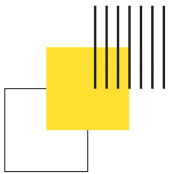
```
<Voximplant.VideoView
   style={styles.selfView}
   videoStreamId={localVideoStreamId}
/>
```



Remote Video

```
call.current.on(Voximplant.CallEvents.EndpointAdded, callEvent => {
        endpoint.current = callEvent.endpoint;
        subscribeToEndpointEvents();
      });
```

```
function subscribeToEndpointEvents() {
    endpoint.current.on(
      Voximplant.EndpointEvents.RemoteVideoStreamAdded,
      endpointEvent => {
        setRemoteVideoStreamId(endpointEvent.videoStream.id);
      },
    );
  }
```



notJust

//development

Q&A