```
/**
 * Sample React Native App
 * https://github.com/facebook/react-native
 * @flow
 */

import React, {Component} from ’react’;
import Firestack from ’react-native-firestack’
import moment from ’moment’;
import {
  AppRegistry,
  StyleSheet,
  Text,
  View,
  TextInput,
  TouchableOpacity,
  AppState,
  Modal
} from ’react-native’;
import BackgroundTimer from ’react-native-background-timer’;

import * as firebase from ’firebase’;
// Initialize Firebase
var config = {
  apiKey: "AIzaSyDg9Qy5W1YzNK1mmleIvzMBTxtQIub3kR8",
  authDomain: "myreactnative.firebaseapp.com",
  databaseURL: "https://myreactnative.firebaseio.com",
  storageBucket: "myreactnative.appspot.com",
  messagingSenderId: "480732941628"
};
firebase.initializeApp(config);
const firestack = new Firestack();

export default class l12_firebase extends Component {
  constructor(props) {
    super(props);
    this.database = firebase.database();
    this.state = {
      chat: ’’,
      chats: [],
      userOnline: 0,
      modalVisible: false,
      name: "Anonymous",
      isShowLogin: true,
      email: "",
      password: ""
    }
    this.userOnlineRef = this.database.ref(’userOnline’);
    this.chatsRef = this.database.ref(’chats’);
    this.handleAppStateChange = this.handleAppStateChange.bind(this);
    this.sendChat = this.sendChat.bind(this);
    this.register = this.register.bind(this);
    this.login = this.login.bind(this);
    this.listeningForAuthChange = this.listeningForAuthChange.bind(this);
    AppState.addEventListener(’change’, this.handleAppStateChange);
  }

  listeningForAuthChange() {
    firestack.auth.listenForAuth((evt) => {
      if (!evt.authenticated) {
        // There was an error or there is no user
        console.log(evt.error)
        this.setState({name: ’Anonymous’, modalVisible: true});
      } else {
        // evt.user contains the user details
        console.log(’User details’, evt.user);
        this.setState({name: evt.user.email});
      }
    }).then(() => console.log(’Listening for authentication changes’));
  }

  handleAppStateChange() {
    console.log("AppState", AppState.currentState);
    if (AppState.currentState == ’active’) {
      this.getNumberOfUserOnlineOnceAndIncreaseBy1ByTransaction();
    } else if (AppState.currentState == ’inactive’) {
      BackgroundTimer.setTimeout(() => this.decreaseNumberOfUserOnlineByTransaction(), 0);
    }
  }

  register() {
    console.log(this.state.email, this.state.password);
    firestack.auth.createUserWithEmail(this.state.email,
       this.state.password).then((user) => {
      this.setState({isShowLogin: true, modalVisible: false});
      console.log("Create user successfully");
    }).catch((err) => {
      alert("An error occured: " + err.description);
      console.log('An error occurred', err);
    })
  }

  login() {
    console.log(this.state.email, this.state.password);
    firestack.auth.signInWithEmail(this.state.email,
       this.state.password).then((user) => {
      this.setState({isShowLogin: true, modalVisible: false});
      console.log("Login user successfully");
    }).catch((err) => {
      alert("An error occured: " + err.rawDescription);
      console.log('An error occurred', err);
    })
  }

  signOut() {
    firestack.auth.signOut().then(res => {
      console.log('You have been signed out')
    }).catch(err => console.log('Uh oh... something weird happened'));
  }

  getNumberOfUserOnlineOnceAndIncreaseBy1ByTransaction() {
    this.userOnlineRef.transaction(function(currentUserOnline) {
      return currentUserOnline + 1;
    });
  }

  decreaseNumberOfUserOnlineByTransaction() {
    this.userOnlineRef.transaction(function(currentUserOnline) {
      return currentUserOnline > 0
        ? currentUserOnline - 1
        : 0;
    });
  }
  listeningForNumberOfUserOnline() {
    this.userOnlineRef.on('value', (snapshot) => {
      console.log("UserOnline change", snapshot.val());
      this.setState({userOnline: snapshot.val()});
    });
  }

  listeningForChatChange() {
    this.chatsRef.on('value', (snapshot) => {
      console.log("Chats change:", snapshot.val());
      this.setState({
        chats: snapshot.val() || []
      });
    })
  }

  componentDidMount() {
    this.listeningForNumberOfUserOnline();
    this.getNumberOfUserOnlineOnceAndIncreaseBy1ByTransaction();
    this.listeningForChatChange();
    this.listeningForAuthChange();
  }

  sendChat() {
    console.log(this.state.chat);
    this.chatsRef.transaction((chats) => {
      if (!chats) {
        chats = [];
      }
      chats.push({name: this.state.name, chat: this.state.chat, when: new Date().getTime()})
;
```

```
      this.setState({chat: ""});
      return chats;
    });
  }

render() {
  let authUI;

  if (this.state.isShowLogin) {
    authUI = (
      <View>
        <Text style={styles.title}>Login</Text>
        <Text>E-mail</Text>
        <TextInput keyboardType="email-address" autoCapitalize="none"
          style={styles.textInput} value={this.state.email}
          onChangeText={(t) => this.setState({email: t})}></TextInput>
        <Text>Password</Text>
        <TextInput secureTextEntry={true} style={styles.textInput}
          value={this.state.password}
          onChangeText={(t) => this.setState({password: t})}></TextInput>
        <TouchableOpacity style={styles.submitButton} onPress={this.login}>
          <Text style={styles.buttonText}>Login</Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.secondButton}
          onPress={() => this.setState({isShowLogin: false})}>
          <Text style={styles.secondButtonText}>Don’t have any account?</Text>
        </TouchableOpacity>
      </View>
    )
  } else {
    authUI = (
      <View>
        <Text style={styles.title}>Register</Text>
        <Text>E-mail</Text>
        <TextInput keyboardType="email-address" autoCapitalize="none"
           style={styles.textInput} value={this.state.email}
           onChangeText={(t) => this.setState({email: t})}></TextInput>
        <Text>Password</Text>
        <TextInput secureTextEntry={true} style={styles.textInput}
          value={this.state.password}
          onChangeText={(t) => this.setState({password: t})}></TextInput>
        <TouchableOpacity style={styles.submitButton} onPress={this.register}>
          <Text style={styles.buttonText}>Register</Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.secondButton}
          onPress={() => this.setState({isShowLogin: true})}>
          <Text style={styles.secondButtonText}>Already have account?</Text>
        </TouchableOpacity>
      </View>
    )
  }

    let signOut;
    if (!this.state.modalVisible) {
      signOut = (
        <TouchableOpacity style={styles.signOutButton} onPress={this.signOut}>
          <Text style={styles.signOutText}>Sign Out</Text>
        </TouchableOpacity>
      )
    }

    return (
      <View style={styles.container}>
        <Modal animationType={"slide"} transparent={true} visible={this.state.modalVisible}>
          <View style={styles.modal}>
            <TouchableOpacity style={styles.closeButton} onPress={() => this.setState({modalVisible: false})}>
              <Text>X</Text>
            </TouchableOpacity>
            {authUI}
          </View>
        </Modal>
        <View style={styles.header}>
          <View style={{
            flexDirection: ’row’
          }}>
            <Text style={styles.labelText}>
```

```
              Hello {this.state.name}!
            </Text>
            {signOut}
          </View>
          <Text style={styles.labelText}>
            #User online: {this.state.userOnline}
          </Text>
        </View>
        <View style={styles.content}>
          {this.state.chats.map((obj, i) => <View key={i} style={styles.chatContainer}>
            <View style={styles.chatMeta}>
              <Text style={styles.bold}>{obj.name || "Anonymous"}</Text>
              <Text>
                ({moment(obj.when).fromNow()})</Text>
            </View>
            <View style={styles.chat}>
              <Text style={styles.chatText}>{obj.chat}</Text>
            </View>
          </View>)}
        </View>
        <View style={styles.footer}>
          <TextInput style={styles.textInput} value={this.state.chat} onChangeText={(t) => this.setState({chat: t})}></TextInput>
          <TouchableOpacity style={styles.button} onPress={this.sendChat}>
            <Text style={styles.buttonText}>Send</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  chatMeta: {
    flexDirection: ’row’
  },
  chat: {
    backgroundColor: ’grey’,
    borderRadius: 10,
    padding: 5
  },
  chatText: {
    color: ’white’
  },
  container: {
    flex: 1,
    paddingTop: 20
  },
  content: {
    padding: 10,
    flex: 1
  },
  labelText: {
    fontSize: 20
  },
  header: {
    justifyContent: ’center’,
    alignItems: ’center’,
    height: 50,
    padding: 5,
    backgroundColor: ’#dddddd’
  },
  footer: {
    height: 50,
    backgroundColor: ’yellow’,
    flexDirection: ’row’
  },
  textInput: {
    flex: 1,
    height: 50,
    width: 200,
    padding: 5,
    backgroundColor: ’white’,
    borderWidth: 1,
    borderRadius: 5
  },
  button: {
    width: 100,
    backgroundColor: ’darkblue’,
    padding: 10,
    borderWidth: 1,
    borderRadius: 5,
    justifyContent: ’center’,
    alignItems: ’center’
  },
  buttonText: {
    fontSize: 20,
    color: ’white’
  },
  modal: {
    height: 280,
    width: 300,
    marginTop: 150,
    padding: 10,
    alignSelf: ’center’,
    backgroundColor: ’lightblue’,
    margin: 10,
    borderRadius: 10,
    justifyContent: ’center’,
    alignItems: ’center’
  },
  closeButton: {
    alignSelf: ’flex-end’
  },
  secondButtonText: {
    color: ’white’
  },
  secondButton: {
    alignSelf: ’center’,
    backgroundColor: ’gray’,
    height: 44,
    borderRadius: 10,
    alignItems: ’center’,
    justifyContent: ’center’,
    margin: 10,
    padding: 5
  },
  submitButton: {
    alignSelf: ’center’,
    backgroundColor: ’darkblue’,
    width: 100,
    height: 44,
    borderRadius: 10,
    alignItems: ’center’,
    justifyContent: ’center’,
    margin: 10
  },
  title: {
    fontSize: 20,
    fontWeight: ’bold’
  },
  bold: {
    fontWeight: ’bold’
  },
  signOutButton: {
    padding: 5
  },
  signOutText: {
    color: ’red’
  }
});

AppRegistry.registerComponent(’l13_authentication’, () => l12_firebase);
```

```
/**
 * Sample React Native App
 * https://github.com/facebook/react-native
 * @flow
 */

import React, {Component} from ’react’;
import Firestack from ’react-native-firestack’
import moment from ’moment’;
import {
  AppRegistry,
  StyleSheet,
  Text,
  View,
  TextInput,
  TouchableOpacity,
  AppState,
  Modal
} from ’react-native’;
import BackgroundTimer from ’react-native-background-timer’;

import * as firebase from ’firebase’;
// Initialize Firebase
var config = {
  apiKey: "AIzaSyDg9Qy5W1YzNK1mmleIvzMBTxtQIub3kR8",
  authDomain: "myreactnative.firebaseapp.com",
  databaseURL: "https://myreactnative.firebaseio.com",
  storageBucket: "myreactnative.appspot.com",
  messagingSenderId: "480732941628"
};
firebase.initializeApp(config);
const firestack = new Firestack();

export default class l12_firebase extends Component {
  constructor(props) {
    super(props);
    this.database = firebase.database();
    this.state = {
      chat: ’’,
      chats: [],
      userOnline: 0,
      modalVisible: false,
      name: "Anonymous",
      isShowLogin: true,
      email: "",
      password: ""
    }
    this.userOnlineRef = this.database.ref(’userOnline’);
    this.chatsRef = this.database.ref(’chats’);
    this.handleAppStateChange = this.handleAppStateChange.bind(this);
    this.sendChat = this.sendChat.bind(this);
    this.register = this.register.bind(this);
    this.login = this.login.bind(this);
    this.listeningForAuthChange = this.listeningForAuthChange.bind(this);
    AppState.addEventListener(’change’, this.handleAppStateChange);
  }

  listeningForAuthChange() {
    firestack.auth.listenForAuth((evt) => {
      if (!evt.authenticated) {
        // There was an error or there is no user
        console.log(evt.error)
        this.setState({name: ’Anonymous’, modalVisible: true});
      } else {
        // evt.user contains the user details
        console.log(’User details’, evt.user);
        this.setState({name: evt.user.email});
      }
    }).then(() => console.log(’Listening for authentication changes’));
  }

  handleAppStateChange() {
    console.log("AppState", AppState.currentState);
    if (AppState.currentState == ’active’) {
      this.getNumberOfUserOnlineOnceAndIncreaseBy1ByTransaction();
    } else if (AppState.currentState == ’inactive’) {
      BackgroundTimer.setTimeout(() => this.decreaseNumberOfUserOnlineByTransaction(), 0);
    }
  }

  register() {
    console.log(this.state.email, this.state.password);
    firestack.auth.createUserWithEmail(this.state.email,
       this.state.password).then((user) => {
      this.setState({isShowLogin: true, modalVisible: false});
      console.log("Create user successfully");
    }).catch((err) => {
      alert("An error occured: " + err.description);
      console.log('An error occurred', err);
    })
  }

  login() {
    console.log(this.state.email, this.state.password);
    firestack.auth.signInWithEmail(this.state.email,
       this.state.password).then((user) => {
      this.setState({isShowLogin: true, modalVisible: false});
      console.log("Login user successfully");
    }).catch((err) => {
      alert("An error occured: " + err.rawDescription);
      console.log('An error occurred', err);
    })
  }

  signOut() {
    firestack.auth.signOut().then(res => {
      console.log('You have been signed out')
    }).catch(err => console.log('Uh oh... something weird happened'));
  }

  getNumberOfUserOnlineOnceAndIncreaseBy1ByTransaction() {
    this.userOnlineRef.transaction(function(currentUserOnline) {
      return currentUserOnline + 1;
    });
  }

  decreaseNumberOfUserOnlineByTransaction() {
    this.userOnlineRef.transaction(function(currentUserOnline) {
      return currentUserOnline > 0
        ? currentUserOnline - 1
        : 0;
    });
  }
  listeningForNumberOfUserOnline() {
    this.userOnlineRef.on('value', (snapshot) => {
      console.log("UserOnline change", snapshot.val());
      this.setState({userOnline: snapshot.val()});
    });
  }

  listeningForChatChange() {
    this.chatsRef.on('value', (snapshot) => {
      console.log("Chats change:", snapshot.val());
      this.setState({
        chats: snapshot.val() || []
      });
      this.markAsRead();
    })
  }

  componentDidMount() {
    this.listeningForNumberOfUserOnline();
    this.getNumberOfUserOnlineOnceAndIncreaseBy1ByTransaction();
    this.listeningForChatChange();
    this.listeningForAuthChange();
  }

  sendChat() {
    console.log(this.state.chat);
    this.chatsRef.transaction((chats) => {
      if (!chats) {
        chats = [];
      }
      chats.push({name: this.state.name, chat: this.state.chat, when: new Date().getTime()})
```

```
;
      this.setState({chat: ""});
      return chats;
    });
  }

  markAsRead(){
    this.chatsRef.transaction((chats) => {
      if (!chats) {
        chats = [];
      }
      for (let i in chats){
        if (!chats[i].read){
          chats[i].read = [];
        }
        if (chats[i].read.indexOf(this.state.name)<0){
          chats[i].read.push(this.state.name);
        }
      }
      this.setState({chat: ""});
      return chats;
    });
  }

render() {
  let authUI;

  if (this.state.isShowLogin) {
    authUI = (
      <View>
        <Text style={styles.title}>Login</Text>
        <Text>E-mail</Text>
        <TextInput keyboardType="email-address" autoCapitalize="none"
          style={styles.textInput} value={this.state.email}
          onChangeText={(t) => this.setState({email: t})}></TextInput>
        <Text>Password</Text>
        <TextInput secureTextEntry={true} style={styles.textInput}
          value={this.state.password}
          onChangeText={(t) => this.setState({password: t})}></TextInput>
        <TouchableOpacity style={styles.submitButton} onPress={this.login}>
          <Text style={styles.buttonText}>Login</Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.secondButton}
          onPress={() => this.setState({isShowLogin: false})}>
          <Text style={styles.secondButtonText}>Don't have any account?</Text>
        </TouchableOpacity>
      </View>
    )
  } else {
    authUI = (
      <View>
        <Text style={styles.title}>Register</Text>
        <Text>E-mail</Text>
        <TextInput keyboardType="email-address" autoCapitalize="none"
           style={styles.textInput} value={this.state.email}
           onChangeText={(t) => this.setState({email: t})}></TextInput>
        <Text>Password</Text>
        <TextInput secureTextEntry={true} style={styles.textInput}
          value={this.state.password}
          onChangeText={(t) => this.setState({password: t})}></TextInput>
        <TouchableOpacity style={styles.submitButton} onPress={this.register}>
          <Text style={styles.buttonText}>Register</Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.secondButton}
          onPress={() => this.setState({isShowLogin: true})}>
          <Text style={styles.secondButtonText}>Already have account?</Text>
        </TouchableOpacity>
      </View>
    )
  }

    let signOut;
    if (!this.state.modalVisible) {
      signOut = (
        <TouchableOpacity style={styles.signOutButton} onPress={this.signOut}>
          <Text style={styles.signOutText}>Sign Out</Text>
        </TouchableOpacity>
      )
    }

    return (
      <View style={styles.container}>
        <Modal animationType={"slide"} transparent={true} visible={this.state.modalVisible}>
          <View style={styles.modal}>
            <TouchableOpacity style={styles.closeButton} onPress={() => this.setState({modalVisible: false})}>
              <Text>X</Text>
            </TouchableOpacity>
            {authUI}
          </View>
        </Modal>
        <View style={styles.header}>
          <View style={{
            flexDirection: ’row’
          }}>
            <Text style={styles.labelText}>
              Hello {this.state.name}!
            </Text>
            {signOut}
          </View>
          <Text style={styles.labelText}>
            #User online: {this.state.userOnline}
          </Text>
        </View>
        <View style={styles.content}>
          {this.state.chats.map((obj, i) => <View key={i} style={styles.chatContainer}>
            <View style={styles.chatMeta}>
              <Text style={styles.bold}>{obj.name || "Anonymous"}</Text>
              <Text>
                ({moment(obj.when).fromNow()})</Text>
              <Text>
                Read by: {(obj.read || []).length}
              </Text>
            </View>
            <View style={styles.chat}>
              <Text style={styles.chatText}>{obj.chat}</Text>
            </View>
          </View>)}
        </View>
        <View style={styles.footer}>
          <TextInput style={styles.textInput} value={this.state.chat} onChangeText={(t) => this.setState({chat: t})}></TextInput>
          <TouchableOpacity style={styles.button} onPress={this.sendChat}>
            <Text style={styles.buttonText}>Send</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  chatMeta: {
    flexDirection: ’row’
  },
  chat: {
    backgroundColor: ’grey’,
    borderRadius: 10,
    padding: 5
  },
  chatText: {
    color: ’white’
  },
  container: {
    flex: 1,
    paddingTop: 20
  },
  content: {
    padding: 10,
    flex: 1
  },
  labelText: {
    fontSize: 20
  },
  header: {
    justifyContent: ’center’,
    alignItems: ’center’,
    height: 50,
    padding: 5,
    backgroundColor: ’#dddddd’
  },
  footer: {
    height: 50,
    backgroundColor: ’yellow’,
    flexDirection: ’row’
  },
  textInput: {
    flex: 1,
    height: 50,
    width: 200,
    padding: 5,
    backgroundColor: ’white’,
    borderWidth: 1,
    borderRadius: 5
  },
  button: {
    width: 100,
    backgroundColor: ’darkblue’,
    padding: 10,
    borderWidth: 1,
    borderRadius: 5,
    justifyContent: ’center’,
    alignItems: ’center’
  },
  buttonText: {
    fontSize: 20,
    color: ’white’
  },
  modal: {
    height: 280,
    width: 300,
    marginTop: 150,
    padding: 10,
    alignSelf: ’center’,
    backgroundColor: ’lightblue’,
    margin: 10,
    borderRadius: 10,
    justifyContent: ’center’,
    alignItems: ’center’
  },
  closeButton: {
    alignSelf: ’flex-end’
  },
  secondButtonText: {
    color: ’white’
  },
  secondButton: {
    alignSelf: ’center’,
    backgroundColor: ’gray’,
    height: 44,
    borderRadius: 10,
    alignItems: ’center’,
    justifyContent: ’center’,
    margin: 10,
    padding: 5
  },
  submitButton: {
    alignSelf: ’center’,
    backgroundColor: ’darkblue’,
    width: 100,
    height: 44,
    borderRadius: 10,
    alignItems: ’center’,
    justifyContent: ’center’,
    margin: 10
  },
  title: {
    fontSize: 20,
    fontWeight: ’bold’
  },
  bold: {
    fontWeight: ’bold’
  },
  signOutButton: {
    padding: 5
  },
  signOutText: {
    color: ’red’
  }
});

AppRegistry.registerComponent(’l13_authentication’, () => l12_firebase);
```