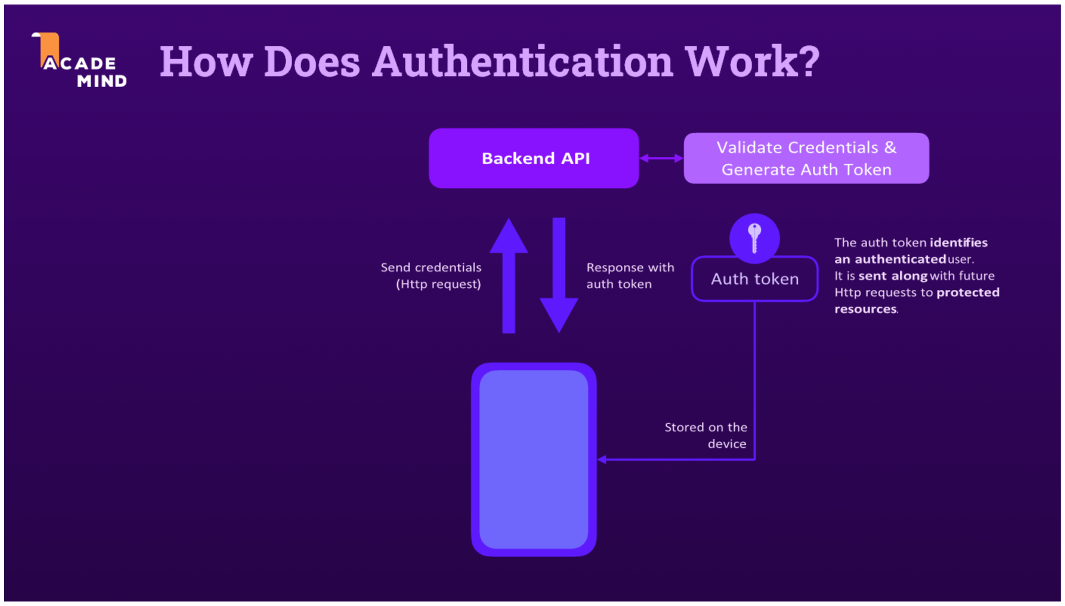
1.Authentication work

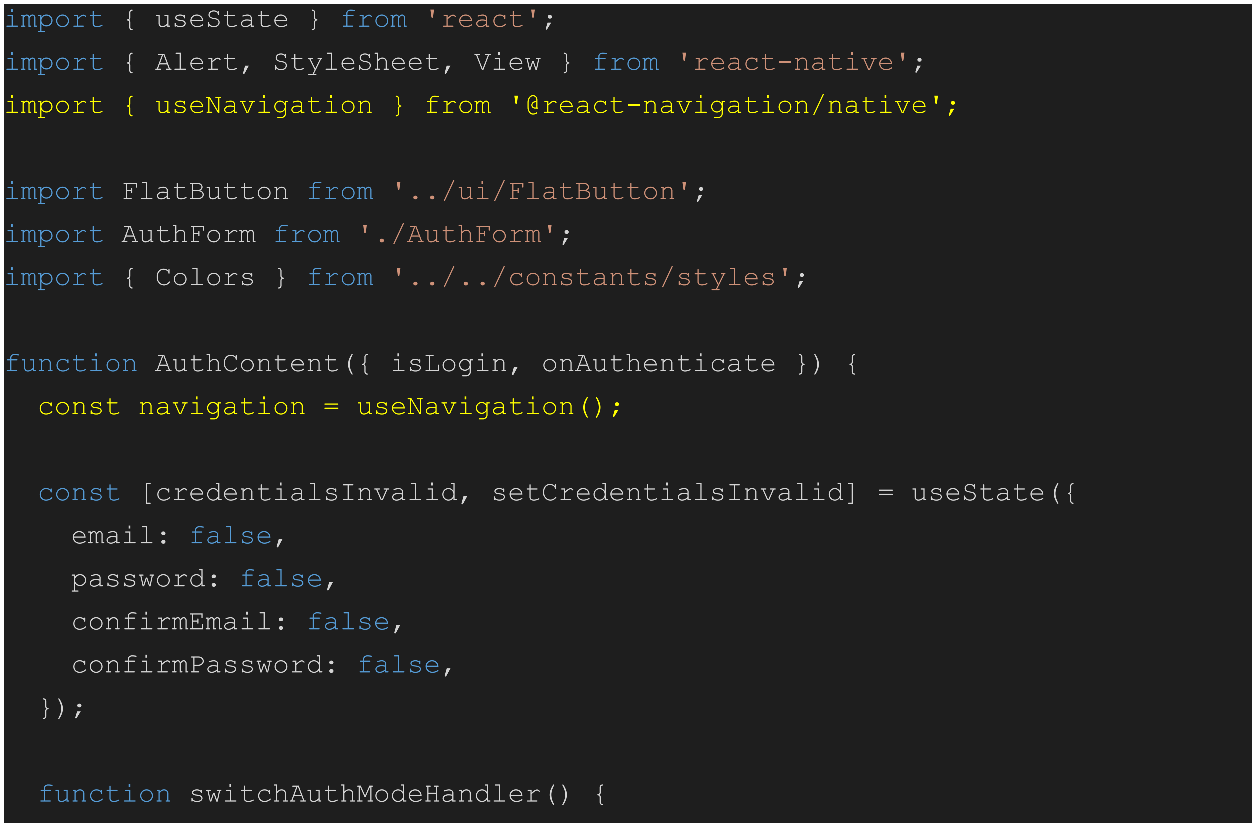

ACADEMIND
How Does Authentication Work?
Backend API
Validate Credentials & Generate Auth Token
Send credentials (Http request)
Response with auth token
Auth token
The auth token identifies an authenticated user. It is sent along with future Http requests to protected resources.
Stored on the device

2.backend setup

1.in firebase authentication select enable signin

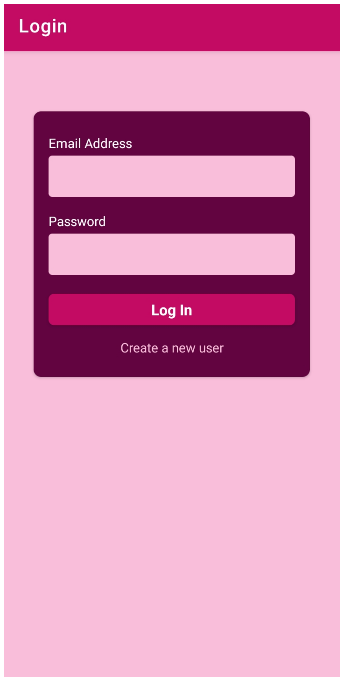
3)controlling signup and login screens

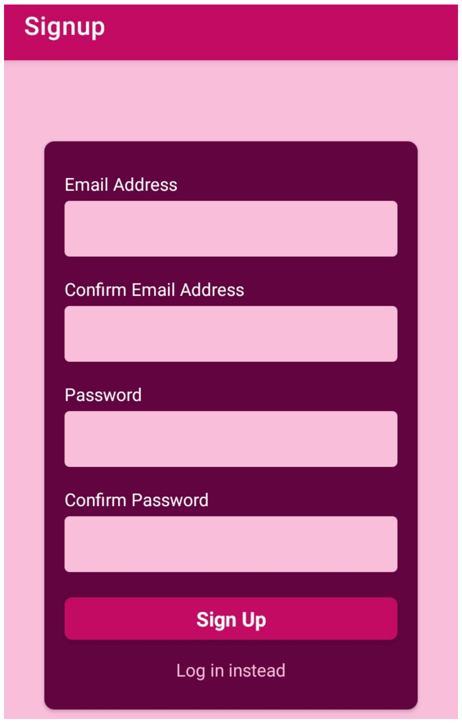
In authcontent.js(auth)



```
import { useState } from 'react';
import { Alert, StyleSheet, View } from 'react-native';
import { useNavigation } from '@react-navigation/native';

import FlatButton from '../ui/FlatButton';
import AuthForm from './AuthForm';
import { Colors } from '../../constants/styles';

function AuthContent({ isLogin, onAuthenticate }) {
  const navigation = useNavigation();

  const [credentialsInvalid, setCredentialsInvalid] = useState({
    email: false,
    password: false,
    confirmEmail: false,
    confirmPassword: false,
  });

  function switchAuthModeHandler() {
    if (isLogin) {
      navigation.replace('Signup');
    } else {
      navigation.replace('Login');
    }
  }

  function submitHandler(credentials) {
    let { email, confirmEmail, password, confirmPassword } = credentials;

    email = email.trim();
    password = password.trim();

    const emailIsValid = email.includes('@');
    const passwordIsValid = password.length > 6;
    const emailsAreEqual = email === confirmEmail;
    const passwordsAreEqual = password === confirmPassword;

    if (
      !emailIsValid ||
      !passwordIsValid ||
      (!isLogin && (!emailsAreEqual || !passwordsAreEqual))
    ) {
      Alert.alert('Invalid input', 'Please check your entered
credentials.');
      setCredentialsInvalid({
        email: !emailIsValid,
        confirmEmail: !emailIsValid || !emailsAreEqual,
        password: !passwordIsValid,
        confirmPassword: !passwordIsValid || !passwordsAreEqual,
      });
      return;
    }
    onAuthenticate({ email, password });
  }

  return (
    <View style={styles.authContent}>
      <AuthForm
        isLogin={isLogin}
        onSubmit={submitHandler}
        credentialsInvalid={credentialsInvalid}
      />
      <View style={styles.buttons}>
        <FlatButton onPress={switchAuthModeHandler}>
          {isLogin ? 'Create a new user' : 'Log in instead'}
        </FlatButton>
      </View>
    </View>
  );
}

export default AuthContent;

const styles = StyleSheet.create({
  authContent: {
    marginTop: 64,
    marginHorizontal: 32,
    padding: 16,
    borderRadius: 8,
    backgroundColor: Colors.primary800,
    elevation: 2,
    shadowColor: 'black',
    shadowOffset: { width: 1, height: 1 },
    shadowOpacity: 0.35,
    shadowRadius: 4,
  },
  buttons: {
    marginTop: 8,
  },
});
```

Output:





2)sending authentication requests to backend

1)go to https://firebase.google.com/docs/reference/rest/auth

And select endpoint

In auth.js(util)

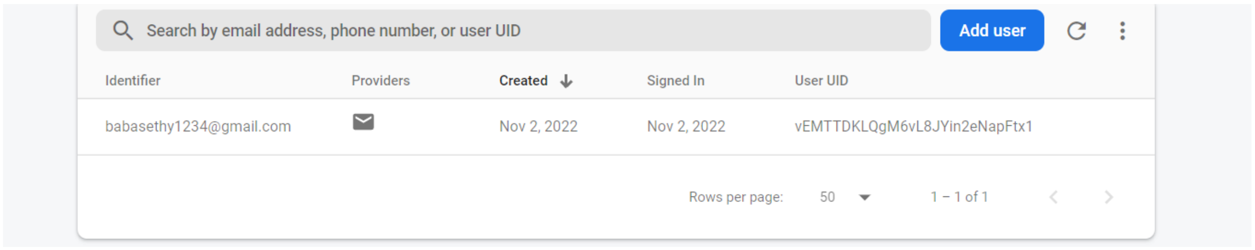
```
import axios from 'axios';

const API_KEY = 'AIzaSyCfBu4-Ja3kqQ878sy5ufeeP9-bLCDP5_Y'

export async function createUser(email, password) {
  const response = await axios.post(

'https://identitytoolkit.googleapis.com/v1/accounts:signInWithCustomToken?
key=' + API_KEY,
    {
      email: email,
      password: password,
      returnSecureToken: true
    }
  );
}
```

3)creating new users

In signupscreen.js(screens)

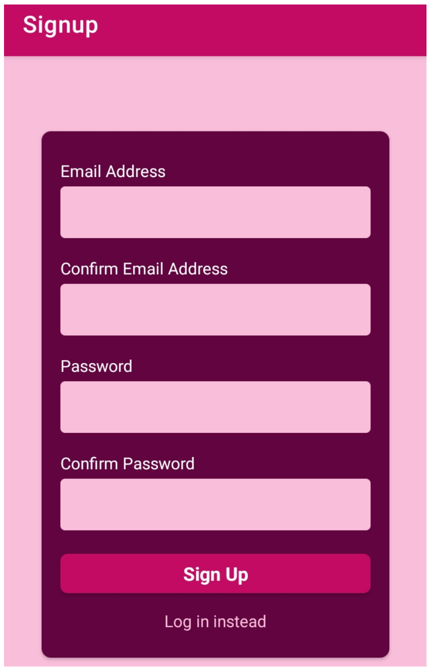
```
import { useState } from 'react';

import AuthContent from '../components/Auth/AuthContent';
import LoadingOverlay from '../components/ui/LoadingOverlay';
import { createUser } from '../util/auth';

function SignupScreen() {
  const [isAuthenticating, setIsAuthenticating] = useState(false);

  async function signupHandler({ email, password }) {
    setIsAuthenticating(true);
    await createUser(email, password);
    setIsAuthenticating(false);
  }

  if (isAuthenticating) {
    return <LoadingOverlay message="Creating user..." />;
  }

  return <AuthContent onAuthenticate={signupHandler} />;
}

export default SignupScreen;
```

Output:



Identifier	Providers	Created	Signed In	User UID
babasethy1234@gmail.com	✉	Nov 2, 2022	Nov 2, 2022	vEMTTDKLQgM6vL8JYin2eNapFtx1

Rows per page: 50 1 – 1 of 1



4)logging users in

In auth.js

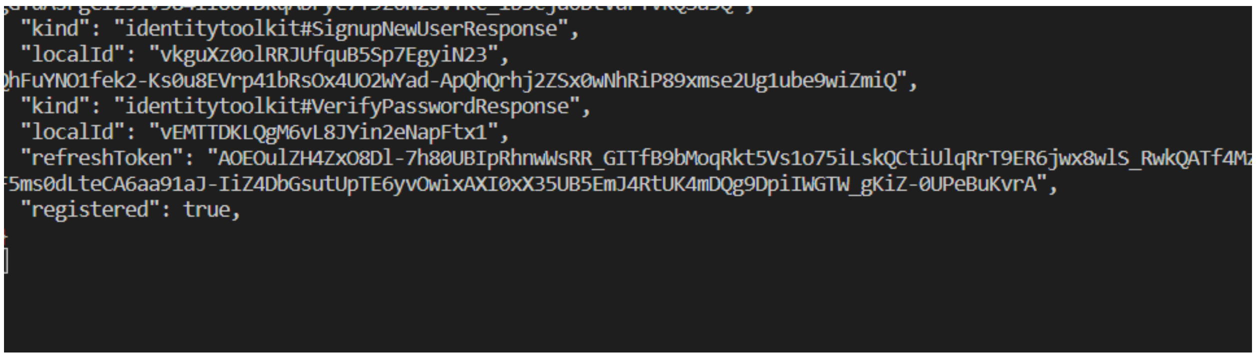
```
import axios from 'axios';

const API_KEY = 'AIzaSyDCYasArcOwcALFhIj2szug5aD2PgUQu1E';

async function authenticate(mode, email, password) {
  const url =
`https://identitytoolkit.googleapis.com/v1/accounts:${mode}?key=${API_KEY}
`;

  const response = await axios.post(url, {
    email: email,
    password: password,
    returnSecureToken: true,
  });

  console.log(response.data);
}

export async function createUser(email, password) {
  await authenticate('signUp', email, password);
}
```

```
export async function login(email, password) {
  await authenticate('signInWithPassword', email, password);
}
```

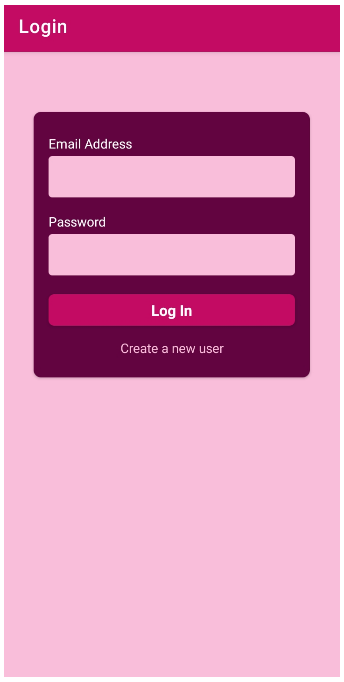
In loginscreen.js

```
import { useState } from 'react';

import AuthContent from '../components/Auth/AuthContent';
import LoadingOverlay from '../components/ui/LoadingOverlay';
import { login } from '../util/auth';

function LoginScreen() {
  const [isAuthenticating, setIsAuthenticating] = useState(false);

  async function loginHandler({ email, password }) {
    setIsAuthenticating(true);
    await login(email, password);
    setIsAuthenticating(false);
  }

  if (isAuthenticating) {
    return <LoadingOverlay message="Logging you in..." />;
  }

  return <AuthContent isLogin onAuthenticate={loginHandler} />;
}

export default LoginScreen;
```

Output:



```
  "kind": "identitytoolkit#SignupNewUserResponse",
  "localId": "vkguXz0olRRJUfquB5Sp7EgyiN23",
QhFuYNO1fek2-Ks0u8EVrp41bRsOx4UO2WYad-ApQhQrhj2ZSx0wNhRiP89xmse2Ug1ube9wiZmiQ",
  "kind": "identitytoolkit#VerifyPasswordResponse",
  "localId": "vEMTTDKLQgM6vL8JYin2eNapFtx1",
  "refreshToken": "AOEOulZH4ZxO8Dl-7h80UBIpRhnwWsRR_GITfB9bMoqRkt5Vs1o75iLskQCtiUlqRrT9ER6jwx8wlS_RwkQATf4Mz
f5ms0dLteCA6aa91aJ-IiZ4DbGsutUpTE6yvOwixAXI0xX35UB5EmJ4RtUK4mDQg9DpiIWGTW_gKiZ-0UPeBuKvrA",
  "registered": true,
```



5)auth-error-logging

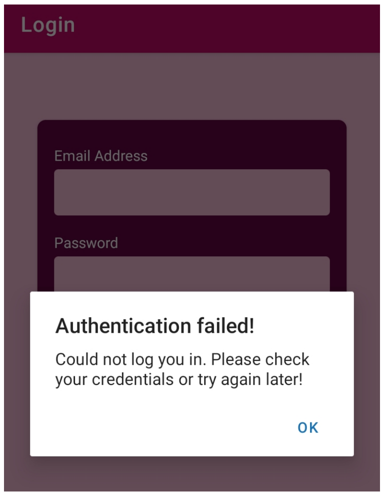
In loginscreen.js

```
import { useState } from 'react';
import { Alert } from 'react-native';

import AuthContent from '../components/Auth/AuthContent';
import LoadingOverlay from '../components/ui/LoadingOverlay';
import { login } from '../util/auth';

function LoginScreen() {
  const [isAuthenticating, setIsAuthenticating] = useState(false);
  async function loginHandler({ email, password }) {
    setIsAuthenticating(true);
    try {
      await login(email, password);
    } catch (error) {
      Alert.alert(
        'Authentication failed!',
        'Could not log you in. Please check your credentials or try again
later!'
      );
    }
    setIsAuthenticating(false);
  }

  if (isAuthenticating) {
    return <LoadingOverlay message="Logging you in..." />;
  }

  return <AuthContent isLogin onAuthenticate={loginHandler} />;
}

export default LoginScreen;
```

In signupscreen.js

```
import { useState } from 'react';
import { Alert } from 'react-native';

import AuthContent from '../components/Auth/AuthContent';
import LoadingOverlay from '../components/ui/LoadingOverlay';
import { createUser } from '../util/auth';

function SignupScreen() {
  const [isAuthenticating, setIsAuthenticating] = useState(false);

  async function signupHandler({ email, password }) {
    setIsAuthenticating(true);
    try {
      await createUser(email, password);
    } catch (error) {
      Alert.alert(
        'Authentication failed',
        'Could not create user, please check your input and try again
later.'
      );
    }
    setIsAuthenticating(false);
  }

  if (isAuthenticating) {
    return <LoadingOverlay message="Creating user..." />;
  }

  return <AuthContent onAuthenticate={signupHandler} />;
}

export default SignupScreen;
```

Output:



Authentication failed!

Could not log you in. Please check your credentials or try again later!

6)storing managing the user auth state(switching for welcome screen.js)

In auth-context.js(store folder)

```
import { createContext, useState } from 'react';

export const AuthContext = createContext({
  token: '',
  isAuthenticated: false,
  authenticate: (token) => {},
  logout: () => {},
});

function AuthContextProvider({ children }) {
  const [authToken, setAuthToken] = useState();

  function authenticate(token) {
    setAuthToken(token);
  }

  function logout() {
    setAuthToken(null);
  }

  const value = {
    token: authToken,
    isAuthenticated: !!authToken,
    authenticate: authenticate,
    logout: logout,
  };

  return <AuthContext.Provider
value={value}>{children}</AuthContext.Provider>;
}

export default AuthContextProvider;
```

In app.js

```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
'@react-navigation/native-stack';
import { StatusBar } from 'expo-status-bar';

import LoginScreen from './screens/LoginScreen';
import SignupScreen from './screens/SignupScreen';
import WelcomeScreen from './screens/WelcomeScreen';
import { Colors } from './constants/styles';
import AuthContextProvider from './store/auth-context';

```

```
const Stack = createNativeStackNavigator();

function AuthStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen name="Login" component={LoginScreen} />
      <Stack.Screen name="Signup" component={SignupScreen} />
    </Stack.Navigator>
  );
}

function AuthenticatedStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen name="Welcome" component={WelcomeScreen} />
    </Stack.Navigator>
  );
}

function Navigation() {
  return (
    <AuthContextProvider>
      <NavigationContainer>
        <AuthStack />
      </NavigationContainer>
    </AuthContextProvider>
  );
}
```

```
export default function App() {
  return (
    <>
      <StatusBar style="light" />

      <Navigation />
    </>
  );
}
```

7)extracting the authentication token
In loginscreen.js

```
import { useContext, useState } from 'react';
import { Alert } from 'react-native';

import AuthContent from '../components/Auth/AuthContent';
import LoadingOverlay from '../components/ui/LoadingOverlay';
import { AuthContext } from '../store/auth-context';
import { login } from '../util/auth';

function LoginScreen() {
  const [isAuthenticating, setIsAuthenticating] = useState(false);

  const authCtx = useContext(AuthContext);

  async function loginHandler({ email, password }) {
    setIsAuthenticating(true);
    try {
      const token = await login(email, password);
      authCtx.authenticate(token);
    } catch (error) {
      Alert.alert(
        'Authentication failed!',
        'Could not log you in. Please check your credentials or try again
later!'
      );
      setIsAuthenticating(false);
    }
  }

  if (isAuthenticating) {
    return <LoadingOverlay message="Logging you in..." />;
  }

  return <AuthContent isLogin onAuthenticate={loginHandler} />;
}

export default LoginScreen;
```

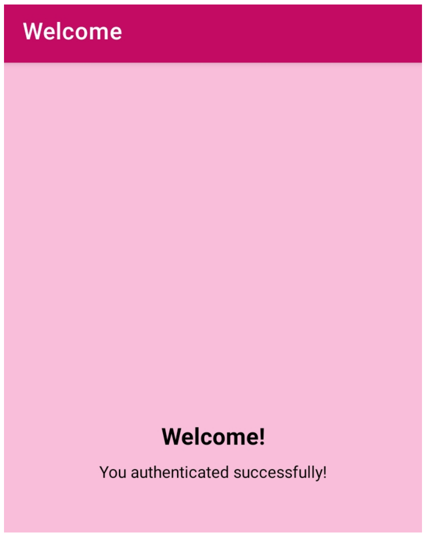
In auth.js

```
import axios from 'axios';

const API_KEY = 'AIzaSyDCYasArcOwcALFhIj2szug5aD2PgUQu1E';

async function authenticate(mode, email, password) {
  const url =
`https://identitytoolkit.googleapis.com/v1/accounts:${mode}?key=${API_KEY}
`;

  const response = await axios.post(url, {
    email: email,
    password: password,
    returnSecureToken: true,
  });

  const token = response.data.idToken;

  return token;
}

export function createUser(email, password) {
  return authenticate('signUp', email, password);
}

export function login(email, password) {
  return authenticate('signInWithPassword', email, password);
}
```

9)protecting screens
In app.js

```
import { useContext } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
'@react-navigation/native-stack';
import { StatusBar } from 'expo-status-bar';

import LoginScreen from './screens/LoginScreen';
import SignupScreen from './screens/SignupScreen';
import WelcomeScreen from './screens/WelcomeScreen';
import { Colors } from './constants/styles';
import AuthContextProvider, { AuthContext } from './store/auth-context';

const Stack = createNativeStackNavigator();

function AuthStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen name="Login" component={LoginScreen} />
      <Stack.Screen name="Signup" component={SignupScreen} />
    </Stack.Navigator>
  );
}

function AuthenticatedStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen name="Welcome" component={WelcomeScreen} />
    </Stack.Navigator>
  );
}

function Navigation() {
  const authCtx = useContext(AuthContext);

  return (
    <NavigationContainer>
      {!authCtx.isAuthenticated && <AuthStack />}
      {authCtx.isAuthenticated && <AuthenticatedStack />}
    </NavigationContainer>
  );
}

export default function App() {
  return (
    <>
      <StatusBar style="light" />
      <AuthContextProvider>
        <Navigation />
      </AuthContextProvider>
    </>
  );
}
```

Output:



Welcome!

You authenticated successfully!

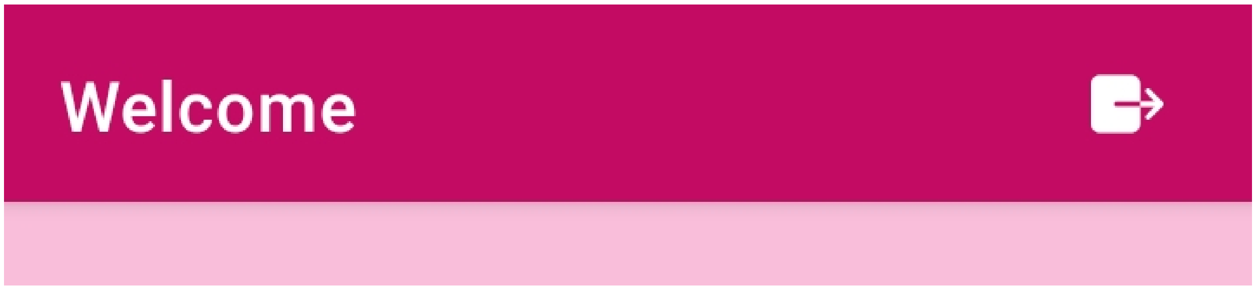
10)adding-logout

In app.js

```
import { useContext } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
'@react-navigation/native-stack';
import { StatusBar } from 'expo-status-bar';

import LoginScreen from './screens/LoginScreen';
import SignupScreen from './screens/SignupScreen';
import WelcomeScreen from './screens/WelcomeScreen';
import { Colors } from './constants/styles';
import AuthContextProvider, { AuthContext } from './store/auth-context';
import IconButton from './components/ui/IconButton';

const Stack = createNativeStackNavigator();

function AuthStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
```

```
      <Stack.Screen name="Login" component={LoginScreen} />
      <Stack.Screen name="Signup" component={SignupScreen} />
    </Stack.Navigator>
  );
}

function AuthenticatedStack() {
  const authCtx = useContext(AuthContext);
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen
        name="Welcome"
        component={WelcomeScreen}
        options={{
          headerRight: ({ tintColor }) => (
            <IconButton
              icon="exit"
              color={tintColor}
              size={24}
              onPress={authCtx.logout}
            />
          ),
        }}
      />
    </Stack.Navigator>
  );
}

function Navigation() {
  const authCtx = useContext(AuthContext);

  return (
    <NavigationContainer>
      {!authCtx.isAuthenticated && <AuthStack />}
      {authCtx.isAuthenticated && <AuthenticatedStack />}
    </NavigationContainer>
  );
}

export default function App() {
  return (
    <>
      <StatusBar style="light" />
      <AuthContextProvider>
        <Navigation />
      </AuthContextProvider>
    </>
  );
}
```

Output:

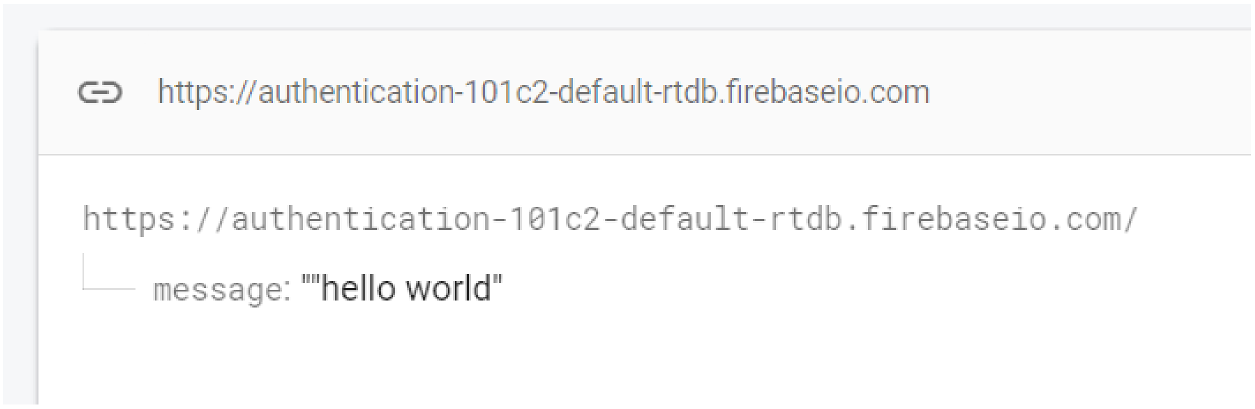


10.accesing proteted resources

In welcomescreen.js

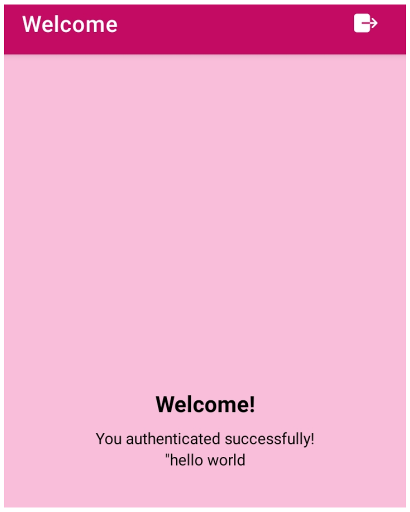
```
import axios from 'axios';
import { useContext, useEffect, useState } from 'react';

import { StyleSheet, Text, View } from 'react-native';
import { AuthContext } from '../store/auth-context';

function WelcomeScreen() {
  const [fetchedMessage, setFetchedMesssage] = useState('');

  const authCtx = useContext(AuthContext);
  const token = authCtx.token;
  useEffect(() => {
    axios
      .get(

'https://react-native-course-3cceb-default-rtdb.firebaseio.com/message.jso
n?auth=' +
          token
      )
      .then((response) => {
        setFetchedMesssage(response.data);
      });
  }, [token]);

  return (
    <View style={styles.rootContainer}>
      <Text style={styles.title}>Welcome!</Text>
      <Text>You authenticated successfully!</Text>
      <Text>{fetchedMessage}</Text>
    </View>
  );
}

export default WelcomeScreen;

const styles = StyleSheet.create({
  rootContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 32,
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 8,
  },
});
```

Output:



https://authentication-101c2-default-rtdb.firebaseio.com

```
https://authentication-101c2-default-rtdb.firebaseio.com/
└── message: ""hello world"
```



Welcome!

You authenticated successfully!
"hello world

13)storing auth tokens on device & logging users in automatically

Install async storage

a)npx expo install @react-native-async-storage/async-storage

b)npx expo install expo-app-loading

In app.js

```
import { useContext, useEffect, useState } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
'@react-navigation/native-stack';
import { StatusBar } from 'expo-status-bar';
import AsyncStorage from '@react-native-async-storage/async-storage';
import AppLoading from 'expo-app-loading';
import LoginScreen from './screens/LoginScreen';
import SignupScreen from './screens/SignupScreen';
import WelcomeScreen from './screens/WelcomeScreen';
import { Colors } from './constants/styles';
import AuthContextProvider, { AuthContext } from './store/auth-context';
import IconButton from './components/ui/IconButton';

const Stack = createNativeStackNavigator();

function AuthStack() {
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen name="Login" component={LoginScreen} />
      <Stack.Screen name="Signup" component={SignupScreen} />
    </Stack.Navigator>
  );
}

function AuthenticatedStack() {
  const authCtx = useContext(AuthContext);
  return (
    <Stack.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: Colors.primary500 },
        headerTintColor: 'white',
        contentStyle: { backgroundColor: Colors.primary100 },
      }}
    >
      <Stack.Screen
        name="Welcome"
        component={WelcomeScreen}
        options={{
          headerRight: ({ tintColor }) => (
            <IconButton
              icon="exit"
              color={tintColor}
              size={24}
              onPress={authCtx.logout}
            />
          ),
        }}
      />
    </Stack.Navigator>
  );
}

function Navigation() {
  const authCtx = useContext(AuthContext);

  return (
    <NavigationContainer>
      {!authCtx.isAuthenticated && <AuthStack />}
      {authCtx.isAuthenticated && <AuthenticatedStack />}
    </NavigationContainer>
  );
}

function Root() {
  const [isTryingLogin, setIsTryingLogin] = useState(true);

  const authCtx = useContext(AuthContext);

  useEffect(() => {
    async function fetchToken() {
      const storedToken = await AsyncStorage.getItem('token');

      if (storedToken) {
        authCtx.authenticate(storedToken);
      }

      setIsTryingLogin(false);
    }
    fetchToken();
  }, []);

  if (isTryingLogin) {
    return <AppLoading />;
  }

  return <Navigation />;
}

export default function App() {

  return (
    <>
      <StatusBar style="light" />
      <AuthContextProvider>
        <Root />
      </AuthContextProvider>
    </>
  );
}
```

In auth-context.js

```
import AsyncStorage from '@react-native-async-storage/async-storage';

import { createContext, useEffect, useState } from 'react';

export const AuthContext = createContext({
  token: '',
  isAuthenticated: false,
  authenticate: (token) => {},
  logout: () => {},
});

function AuthContextProvider({ children }) {
  const [authToken, setAuthToken] = useState();

  function authenticate(token) {
    setAuthToken(token);
    AsyncStorage.setItem('token', token);
  }

  function logout() {
    setAuthToken(null);
    AsyncStorage.removeItem('token');
  }

  const value = {
    token: authToken,
    isAuthenticated: !!authToken,
    authenticate: authenticate,
    logout: logout,
  };

  return <AuthContext.Provider
value={value}>{children}</AuthContext.Provider>;
}

export default AuthContextProvider;
```