STEPS TO BE FOLLOWED:

1)CREATE FRONT-END USING ANGULAR OR REACT

2)LOG IN TO AWS ACCOUNT

3)SEARCH FOR COGNITO AND CREATE USER POOL

4)REGISTER YOUR DOMAIN

5)INSIDE CONFIG.JSON FILE CREATE COGNITO OBJECT WITH FOLLOWING PROPERTIES

"cognito":{

    "REGION":"",

    "USER\_POOL\_ID":"",

    "APP\_CLIENT\_ID":""

  }

GET USER\_POOL\_ID AND APP\_CLIENT\_ID FROM AWS

6) INSIDE INDEX.JS FILE CREATE AMPLIFY.CONFIGURE FUNCTION LIKE THIS

Amplify.configure({

    Auth:{

        mandatorySignID:true,

        region:config.cognito.REGION,

        userPoolId:config.cognito.USER\_POOL\_ID,

        userPoolWebClientId:config.cognito.APP\_CLIENT\_ID

    }

})

AFTER THIS YOUR BASIC AWS USER POOL IS CONNECTED WITH YOUR DOMAIN.

NEXT WE ARE DOING RESITER AND LOGIN FUNCTIONALITY.

**REGISTER**

REGISTRATION STEPS

1)INSIDE REGISTER.JS FILE IMPORT {AUTH} FROM AWS-AMPLIFY

2)INSIDE CLASS CREATE STATE OBJECT LIKE THIS

state = {

    username: "",

    email: "",

    password: "",

    confirmpassword: "",

    errors: {

      cognito: null,

      blankfield: false,

      passwordmatch: false

    }

  }

3)AFTER FORM VALIDATION DO THIS

const { username,email,password }=this.state;

    try{

      const signUpResponse=await Auth.signUp({

        username,

        password,

        attributes:{

          email:email

        }

      });

      console.log(signUpResponse);

      this.props.history.push("/welcome");

    }catch(error){

      let err=null;

      !error.message?err={"message":error}:err=error;

      this.setState({

        errors:{

          ...this.state.errors,

          cognito:error

        }

      })

    }

  };

**LOGIN**

Inder login.js do same as in register without email

Like this:

 try{

      const user=await Auth.signIn(thsi.state.username,this.state.password);

      console.log(user);

      this.props.history.push("/");

    }catch(error){

      let err=null;

      !error.message?err={"message":error}:err=error;

      this.setState({

        errors:{

          ...this.state.errors,

          cognito:error

        }

      })

    }

  };

  };

After this open app.js for routing

state={

    isAuthenticated=false,

    user:null

  }

  setAuthStatus=authenticated=>{

    this.setState({isAuthenticated:authenticated})

  }

  setUser=user=>{

    this.setUser({user:user})

  }

  render() {

    const authProps={

      isAuthenticated:this.state.isAuthenticated,

      user:this.state.user,

      setAuthStatus:this.setAuthStatus,

      setUser:this.setUser

    }

    return (

      <div className="App">

        <Router>

          <div>

            <Navbar auth={authProps} />

            <Switch>

    <Route exact path="/" render={(props)=><Home auth={authProps}/>} />

              <Route exact path="/products" render={(props)=><Products auth={authProps}/>} />

              <Route exact path="/admin" render={(props)=><ProductAdmin auth={authProps}/>} />

              <Route exact path="/login" render={(props)=><LogIn auth={authProps}/>} />

              <Route exact path="/register" render={(props)=><Register auth={authProps}/>} />

              <Route exact path="/forgotpassword" render={(props)=><ForgotPassword auth={authProps}/>} />

              <Route exact path="/forgotpasswordverification" render={(props)=><ForgotPasswordVerification auth={authProps}/>} />

              <Route exact path="/changepassword" render={(props)=><ChangePassword auth={authProps}/>} />

              <Route exact path="/changepasswordconfirmation" render={(props)=><ChangePasswordConfirm auth={authProps}/>} />

              <Route exact path="/welcome" component={Welcome} />

            </Switch>

            <Footer />

          </div>

        </Router>

After this do logic for login and log out button

Logic is if user is authenticated then username and logout button should be displayed and if user is not authenticated then register and login button should be displayed.