Terminologies:

- **Microservice:**
  - Microservice architecture is a method of developing system in small modules which communicates internally with other modules through lightweight mechanism like HTTP/REST with JSON.

  - **Advantages of using Microservice Architecture:**
    - Each module of application can be developed, tweaked and redeployed independently.
    - In Microservice architecture, different modules of the application can be built with different technologies.
    - In Microservice architecture, each modules usually manage its unique databases so decentralized data management can be easily done.
- **AWS Lambda:**
  - AWS Lambda is a service provided by Amazon to create Serverless application which executes code when there is request for processing and scales automatically when there are many request for processing.
- **AWS Cognito**
  - AWS Cognito service is used for user management of the application which allows to authenticate user in the application. It also allows user to authenticate user through social sites such as Facebook, Twitter or Amazon using SAML identity solution or using own identity system.
- **AWS DynamoDB**
  - AWS DynamoDB is NOSQL Database which provides good performance and is auto scalable so it can handle any level of traffic.

Project Technology Stack:

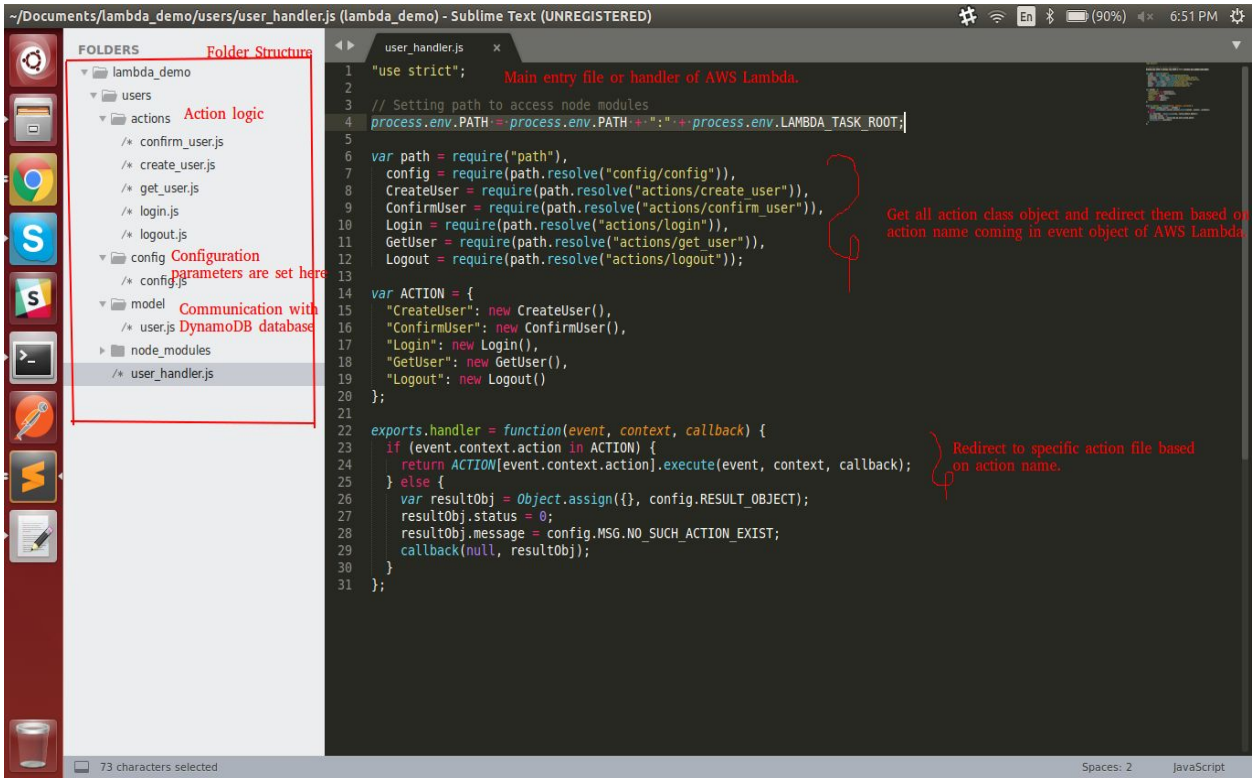
- **Code Language:** AWS Lambda in NodeJS environment
- **User Management:** AWS Cognito Service
- **Database:** AWS DynamoDB

Project Description:

This project is build in AWS Lambda by using Microservice architecture. I have created one module of the application i.e. Users which supports basic functionality like creating user, confirm user by verifying email, login, get user details and logout. I have covered some concepts of AWS Lambda, DynamoDB and Cognito which are described below in "**Concepts covered in Project"** section.

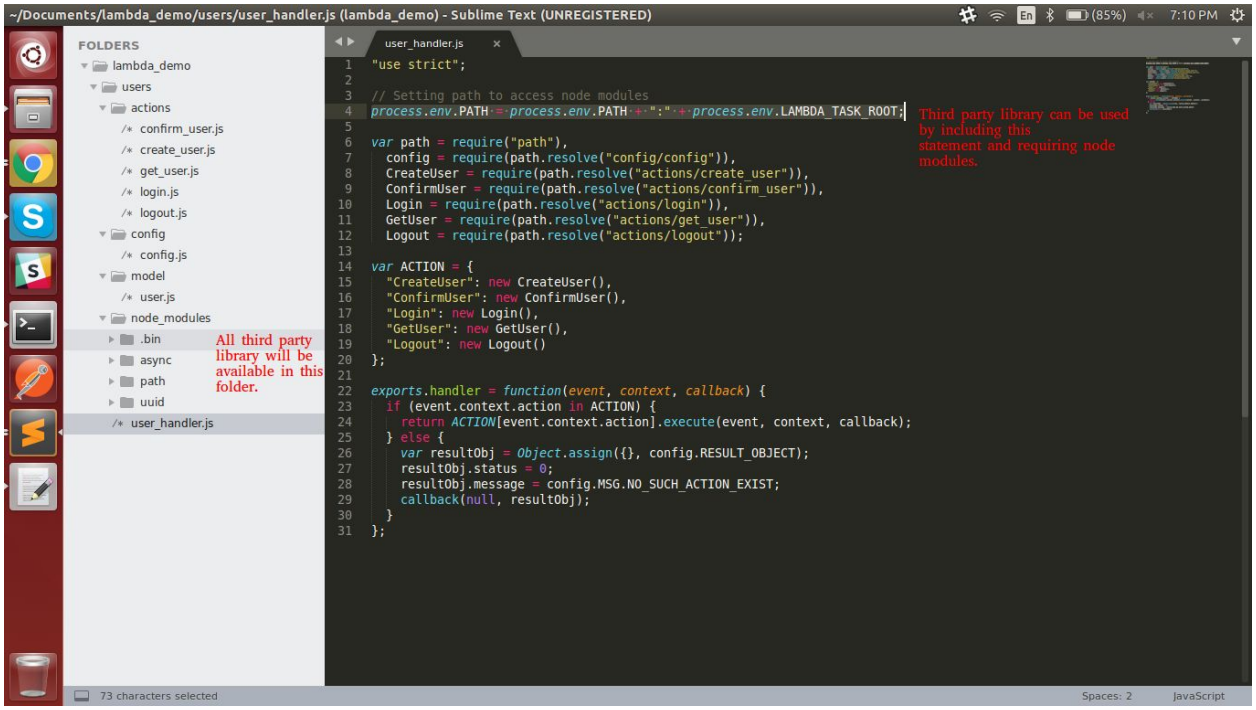
Concepts covered in Project:

- ➢ **Code Structure for AWS Lambda(Microservice)**



- “**user_handler.js**” is main entry file from where all request based on action name will be redirected to specific action file located in “users/actions” folder and do processing of request according to that.
- There are three folder which are actions, config, model.
- All action logic is present in “users/actions” folder.
- All communication with AWS DynamoDB database is done in “**user.js**” which is located in “users/model” folder.
- All configuration parameter are set in “**config.js**” which is located in “users/config” folder.

➢ How to use third party library in AWS Lambda?



- Install third party library in Lambda code folder using “npm install <library_name>”. After successful installation, library folder will be available in “node_modules” folder.
- Include “process.env.PATH = process.env.PATH + ":" + process.env.LAMBDA_TASK_ROOT;” line in main handler file of AWS Lambda. This will set path to root folder of AWS Lambda code folder for accessing third party library located in zipped code folder.
- Zip your Lambda folder code and upload zip on AWS Lambda by selecting “Code Entry type” dropdown as “Upload a .zip file”.

➢ How to use AWS Cognito service in AWS Lambda?

~/Documents/lambda_demo/users/actions/logout.js (lambda_demo) - Sublime Text (UNREGISTERED)

```
"use strict";

// requiring node modules which will be used in logic of action.

var aws =  require("aws-sdk"),
  path = require("path"),
  config = require(path.resolve("config/config")),
  cognitoIdentityServiceProvider = new aws.CognitoIdentityServiceProvider();

// setting AWS region
aws.config.update({
  region: config.AWS_REGION
});

module.exports = class Logout {
  constructor() {}

  execute (event, context, callback) {

    var params = {
      AccessToken: event.context.accessToken
    };

    // Sign out user from all devices
    cognitoIdentityServiceProvider.globalSignOut(params, function (err, result) {
      if (err) {
        console.error(err);
        var resultObj = Object.assign({}, config.RESULT_OBJECT);
        resultObj.status = 0;
        resultObj.message = err.message;
        resultObj.error = err;
        callback(null, resultObj);
      } else {
        var resultObj = Object.assign({}, config.RESULT_OBJECT);
        resultObj.message = config.MSG.SUCCESS;
        callback(null, resultObj);
      }
    });
  }
}
```

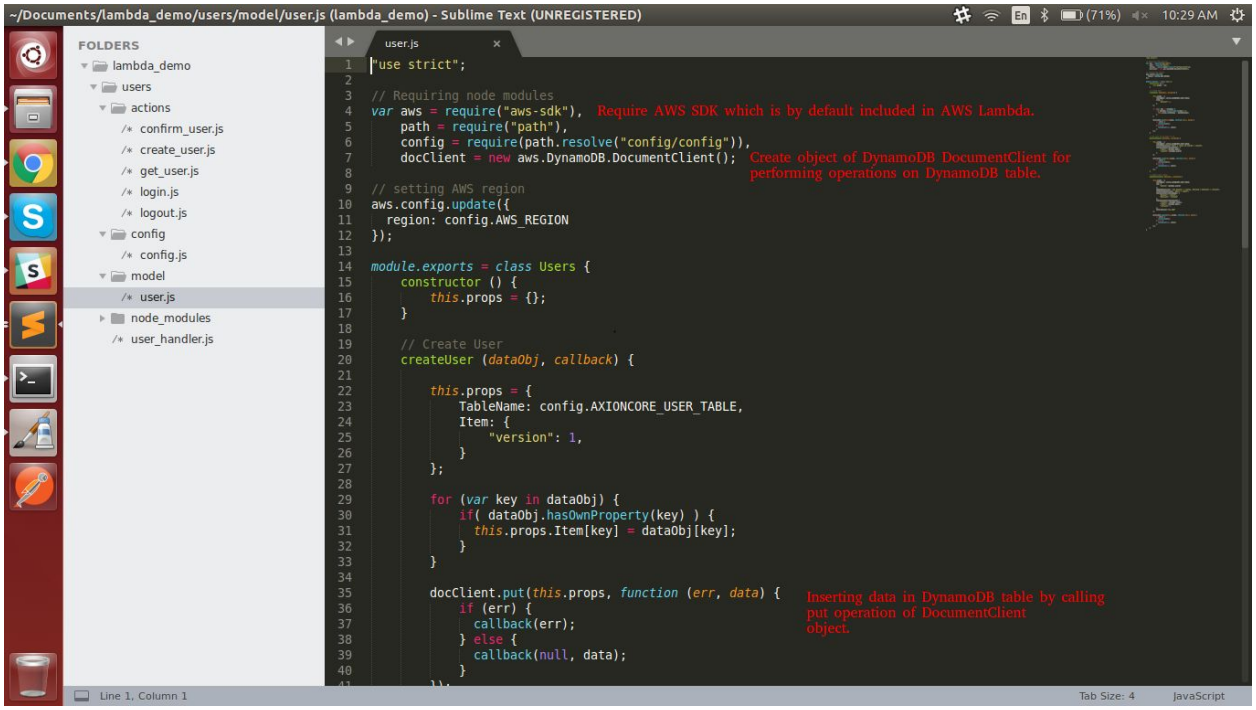
Require AWS SDK which is by default included in AWS Lambda.

Create AWS Cognito Identity Service Provider object to perform operations on Cognito User Pool.

Calling globalSignOut operation of CognitoIdentityServiceProvider object to perform action on Cognito User Pool.

- AWS SDK provides CognitoIdentityServiceProvider service which helps to communicate with Cognito User Pool. AWS SDK is by default included by AWS Lambda.
- You can create object of CognitoIdentityServiceProvider service and can call various operation of that. All operations of CognitoIdentityServiceProvider are mentioned on this link.

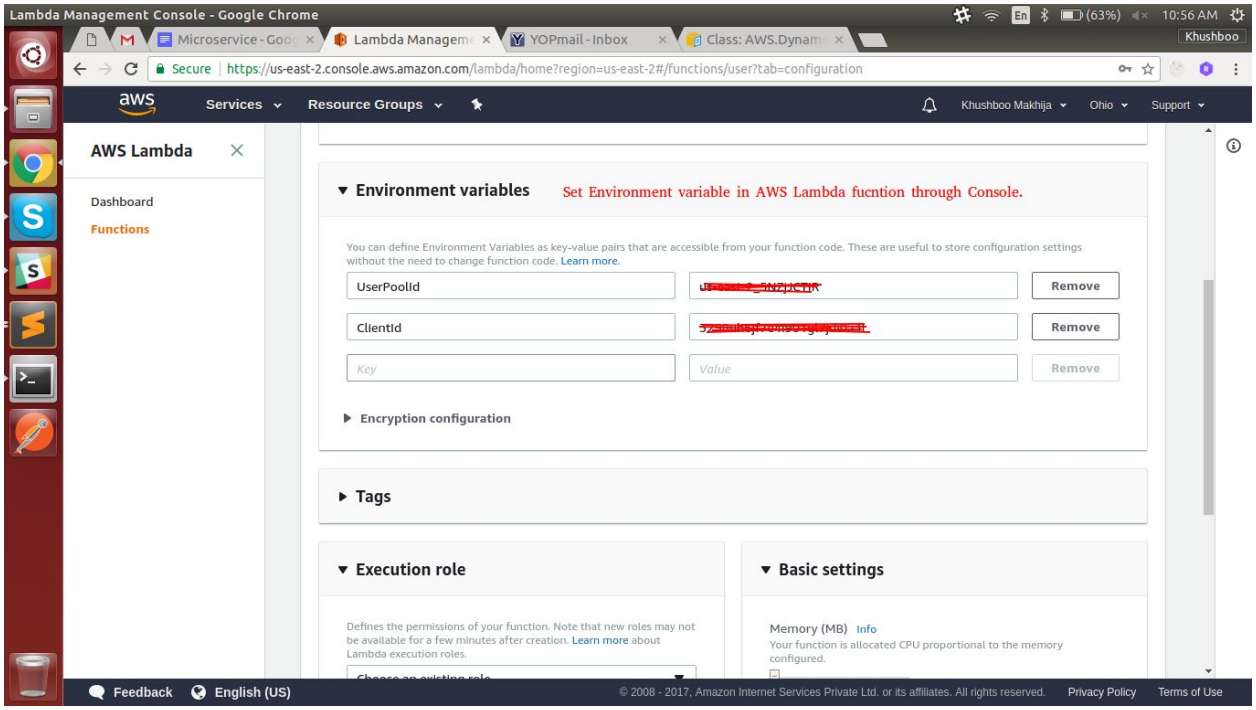
➢ How to use AWS DynamoDB in AWS Lambda?



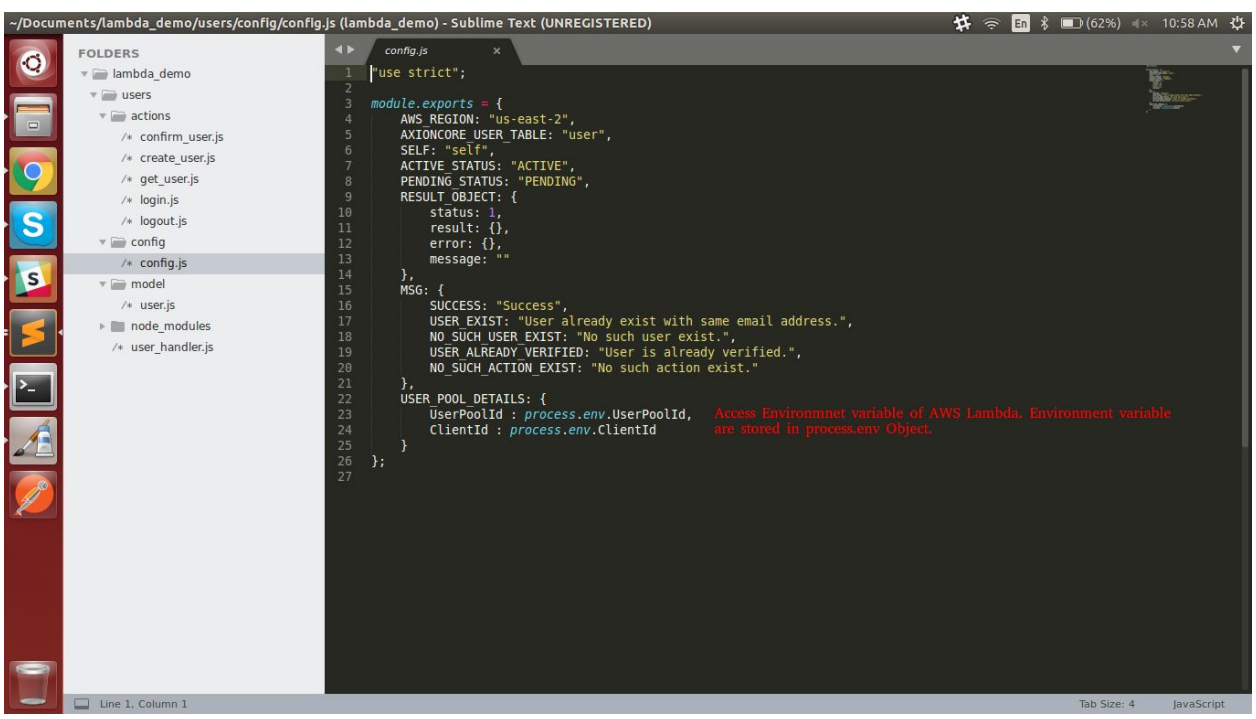
- To use AWS DynamoDB in AWS Lambda you have to create object of AWS DynamoDB DocumentClient. You can create object using AWS SDK Library which is by default available in AWS Lambda.
- Various operation on DynamoDB can be done. List of operations which can be done on DynamoDB are mentioned on this link.

➢ How to set and use environment variables in Lambda?

Set Environment Variable



Get Environment Variable:



- Environment variable can be set in AWS Lambda Console in “**Environment**” tab as key and value.
- You can use environment variable in your code by “process.env.<variable_name>”