National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Swapnil S Vernekar
Student ID: 21174041

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Swapnil S Vernekar
Student ID:	21174041
Programme:	MSc. in Cloud Computing
Year:	2022-2023
Module:	MSc Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual
Word Count:	1732
Page Count:	12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Swapnil S Vernekar
Date:	17th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Swapnil S Vernekar
21174041

1 Android application development environment

For this research project, an android application is in use which is installed on the android mobile device. For the purpose of its development, android studio code is been used. Its and IDE which is developed by the google which enables developers to develop robust applications.
For this research project Android Studio Flamingo - 2022.2.1 Patch 2 is used which is improved by jetbrains industries.

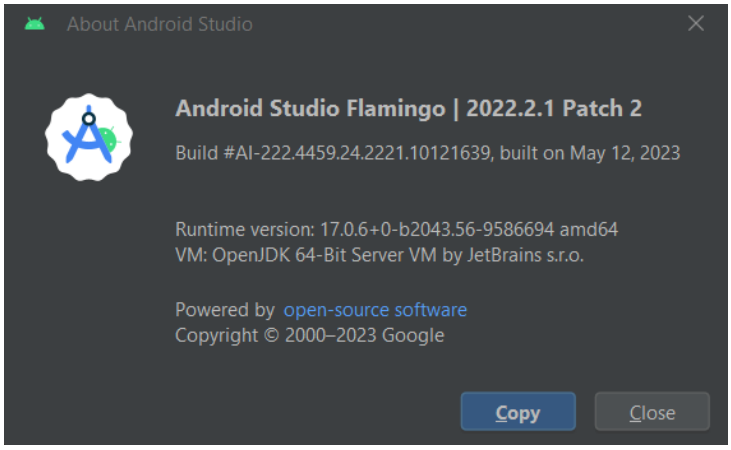

Figure 1: Android studio code

There are two most important dependencies that must be imported for this application to work which is the google's vision library and the firebase plugin. Google vision library is used for integration with cameras as this application has image processing functionality embedded in it and the firebase plugin is used in order to connect the application to the firebase console for data store and monitoring purposes. All of these dependencies must be added into the build.gradle file. Once these dependencies are added, the application will be connected to the firebase console and also able to use the camera.

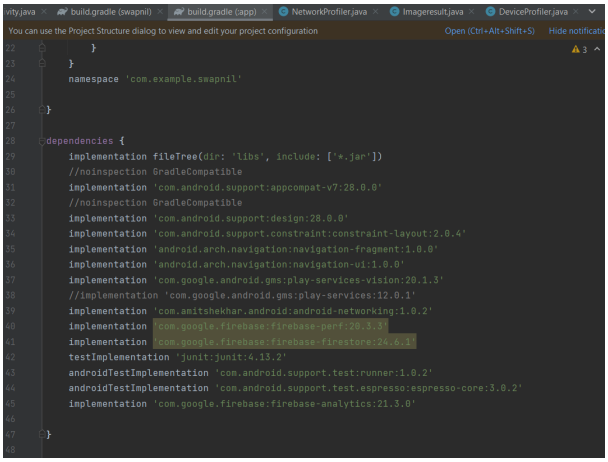

Figure 2: Dependencies on build.gradle file

After the dependencies are imported the respective class files in java can be created with their individual logic. After setting up all the class files, the structure of the application looks as follows.

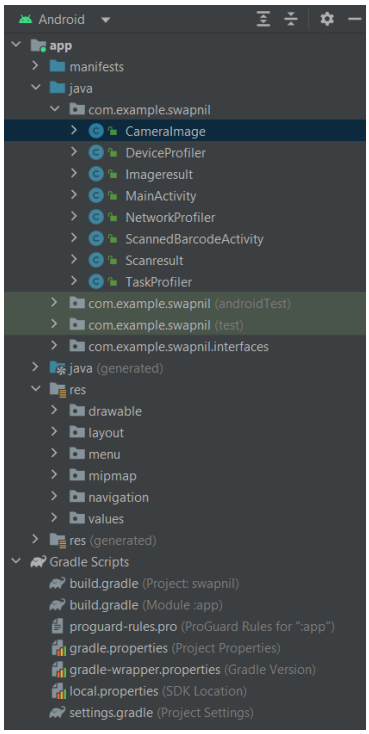

Figure 3: Structure of the project

There is a power equation used in this application logic which has a set of weights which give priority to the network parameters. The weights are W1, W2, W3 and the W4. These weights can be adjusted based on the priority to be given. This equation is very critical as it is used intensively by the decision making engine for offloading tasks. For this research the weights are given as follows:-

Performance Parameter weight	Value
Bandwidth W1	0.3
Latency W2	1
Signal Strength W3	0.2
Battery Level W4	0.1

However, these weights can be changes as per the use case.

```
    // Calculate performance
    performance = 0.3 * getBandwidth() + 1.0 * (1.0 / latency) + 0.2 * getSignalStrength() + 0.1 * getBatteryLevel();
    Log.d( tag: "Performance",  msg: "Performance: " + performance);

    verifyanduploadTofirestore();

}
```

Figure 4: Performance Equation with weights

After setting up the power equation with its respective weights, the application is now ready to be tested on the device. For this research, virtual device emulator is used and the application is installed on it and is tested. The android emulator device Nexus 5 API 29 is used which has android version 10 installed on it. For setting up a new emulator go to run and select the select device tab.

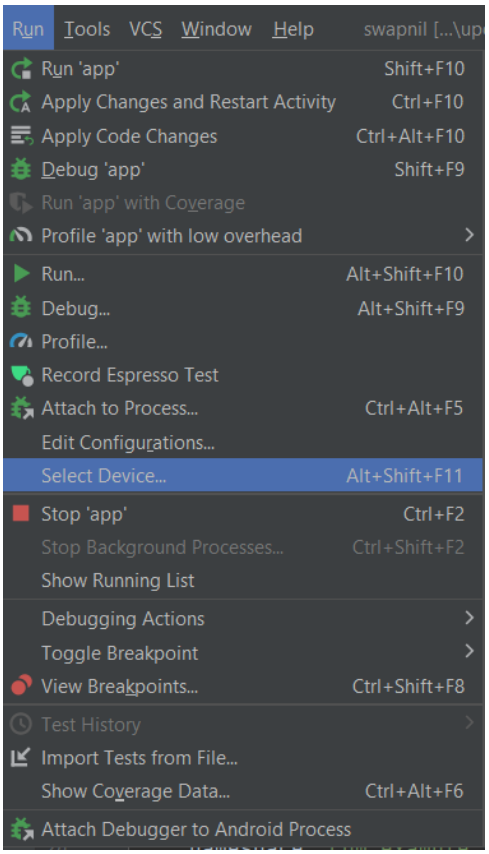

Figure 5: Emulator setup 1

Then click on device manager tab then select create device. There will be a set of virtual devices displayed, select on one of those and choose on select operating system and click on create. The operating system will be then downloaded and emulator will start.

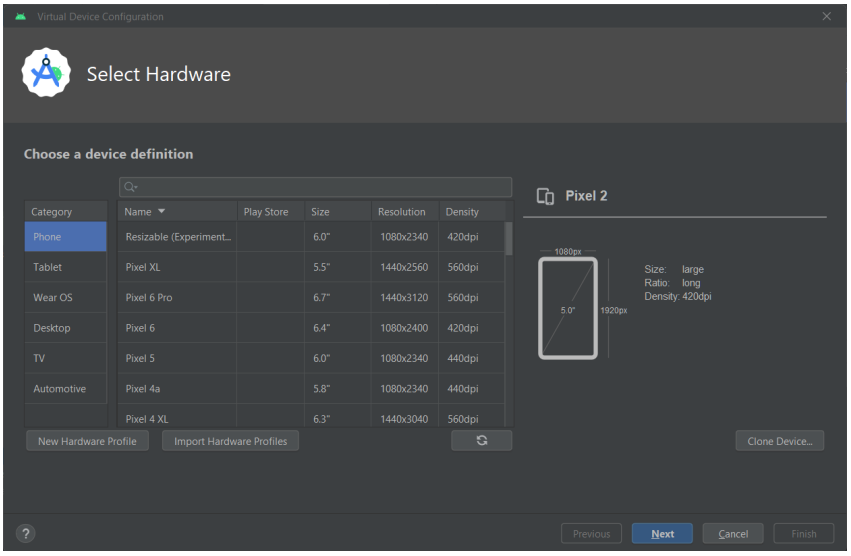

Figure 6: Select device

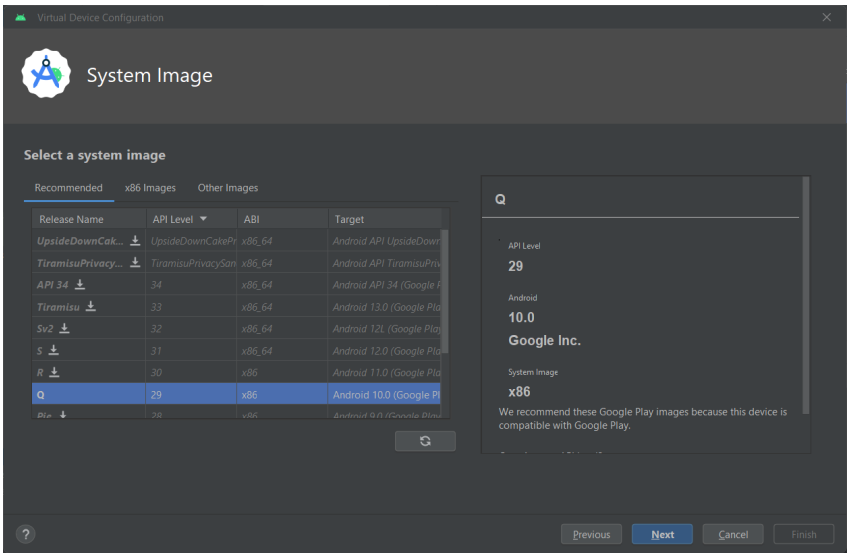

Figure 7: Select Operating System

After the emulator is started, the application can then be built and the system will start installing the apk on the installed virtual machine device in the android studio code.

2 Firebase Console

The application is connected with the firebase console which provides an insights to all the application level metrics such as the performance, start time, CPU utilization. Using

this console, lot of application level information can gathered and can be used to improve the logic even further.

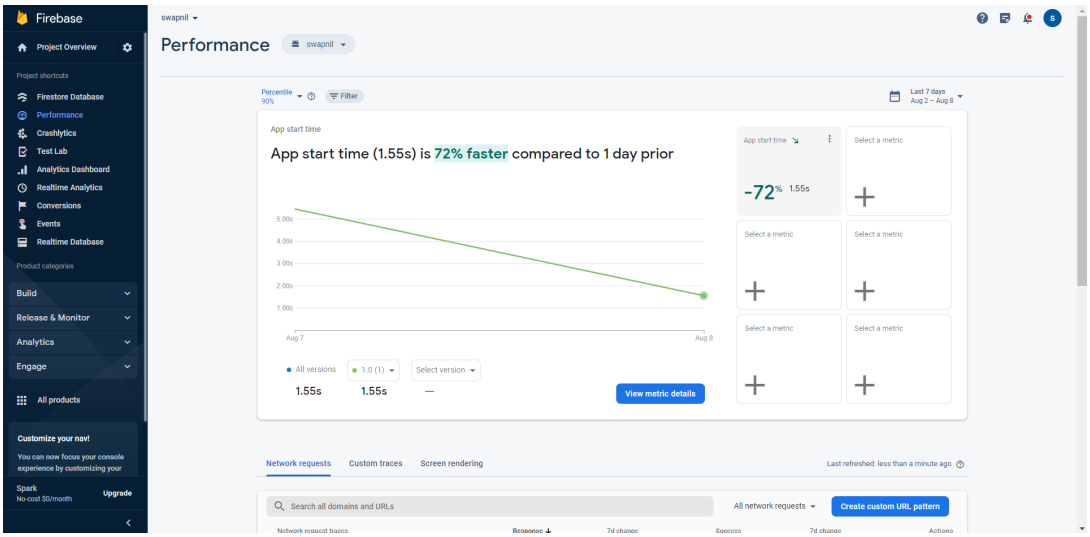

Figure 8: Firebase Console

To get access to this console requires signin to this service which can be done using the personal email id. Once an account is created, the console will popup showing all the insights of the application.

3 Cloud Environment Setup

For the purpose of development of the cloud environment for this research work, microservice architecture is been used where in AWS'S lambda function plays an important role for the execution of the computational task. The lambda function is been created by selecting the environment as NodeJs as the node js application will be deployed on it.

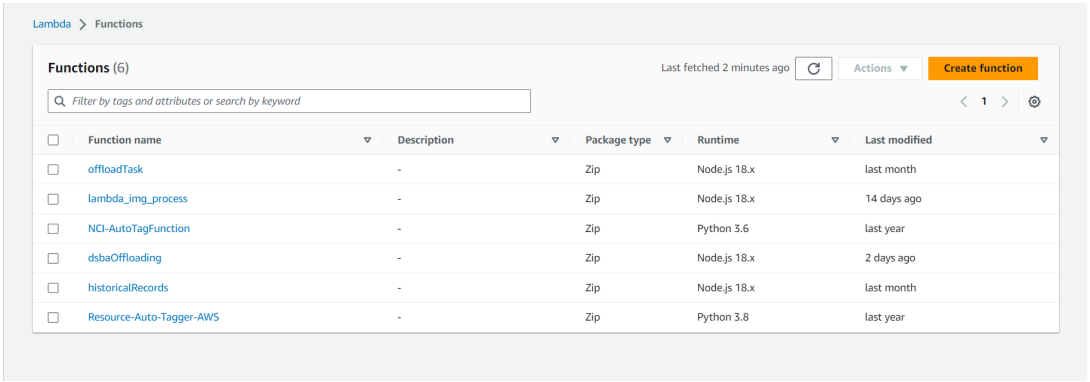

Figure 9: Lambda Function

As the node.js code to be uploaded is more than 20 mb, an s3 bucket is to be used where all the node.js file is uploaded and then lambda function is pointed to this bucket to fetch the code and finally deploy it. In order to trigger this lambda function, an API

gateway needs to be setup which will provide with an api end point which can be used to invoke this lambda function remotely using the application and to start the computation. For creating the api gateway, search for the api gateway on the search area of AWS and then click on create API. Select the rest API as it supports Lambda, HTTP, AWS Services.

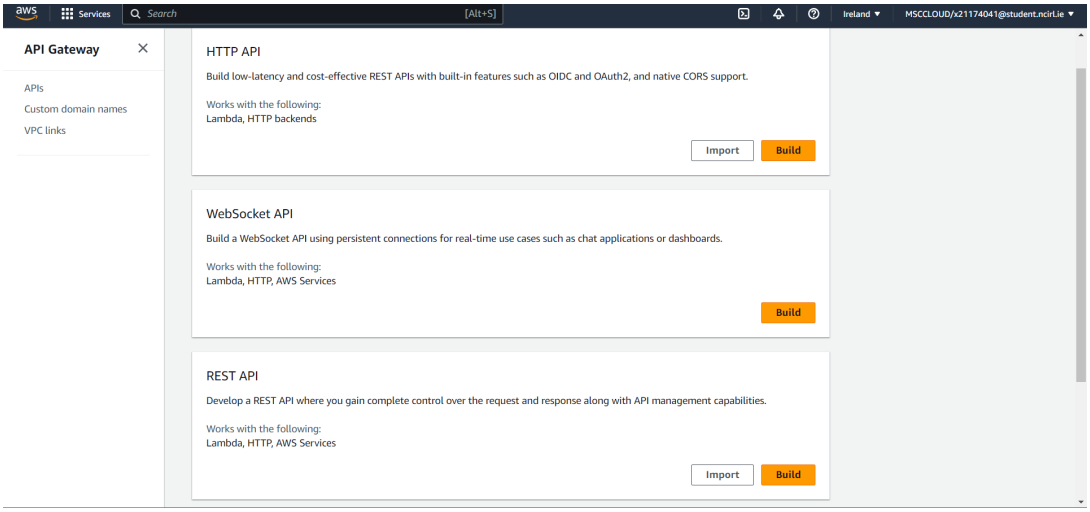

Figure 10: API Gateway

Give an appropriate name to the api created and then select the lambda function to be invoked. Then click on create function.

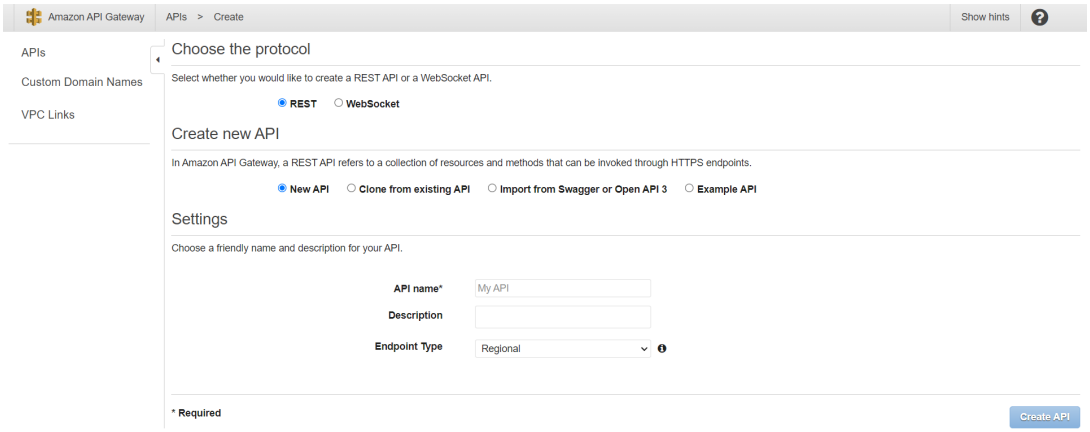

Figure 11: API Gateway creation

Once the api gateway is setup, click on actions and create a resource. Give the appropriate name for it as it will be the endpoint of the api. Then create the resource.

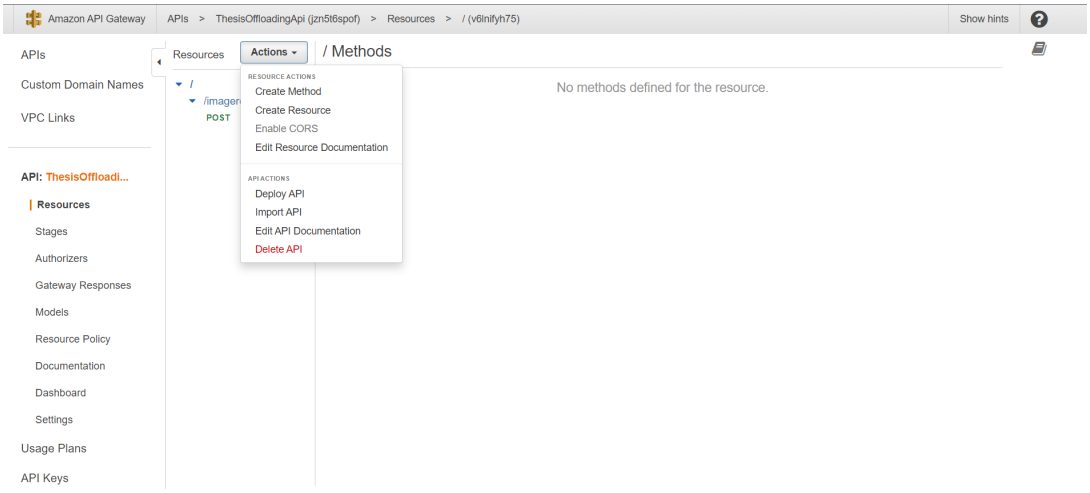

Figure 12: API Gateway creation

After the resource is created, click on create method and select POST method as the application will be sending the image to the lambda function. Select integration type as the lambda function and then finally select the lambda function to be invoked and then save. Then deploy the api by selecting deploy tab. Give the appropriate deployment name and click on deploy.

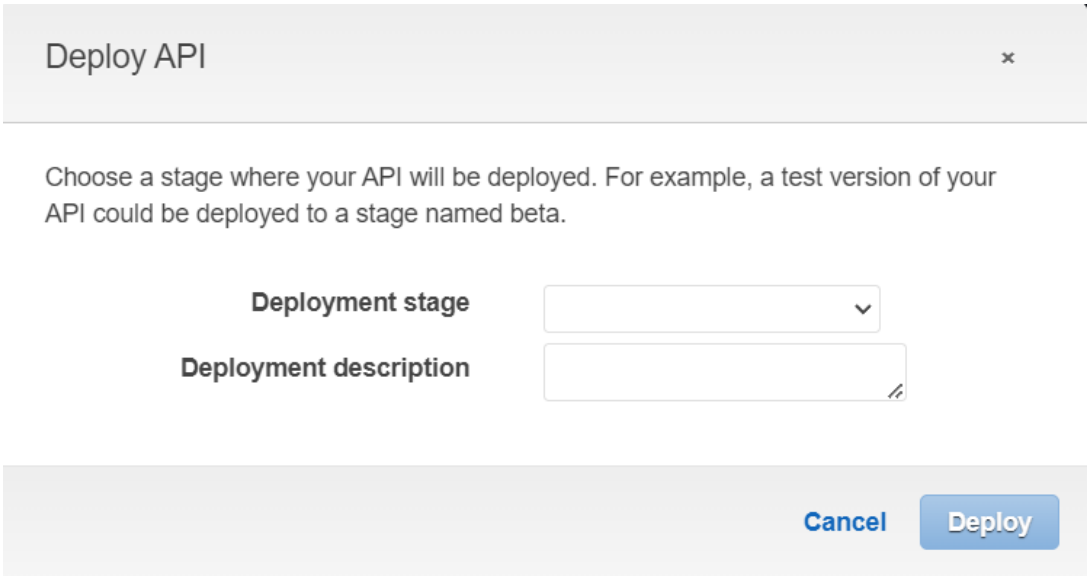

Figure 13: Deploy API

The api gateway will be completely configured and the flow of the api call will be as follows.

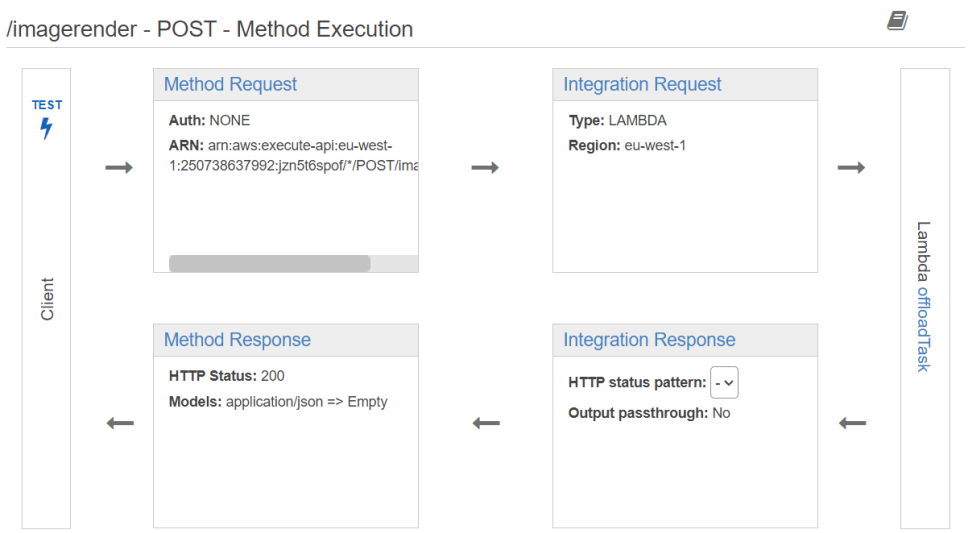

Figure 14: API Gateway connection to lambda function

Once the API gateway is connected to the lambda function, the lmbda console will show the connectivity to the API which is created and is linked to.

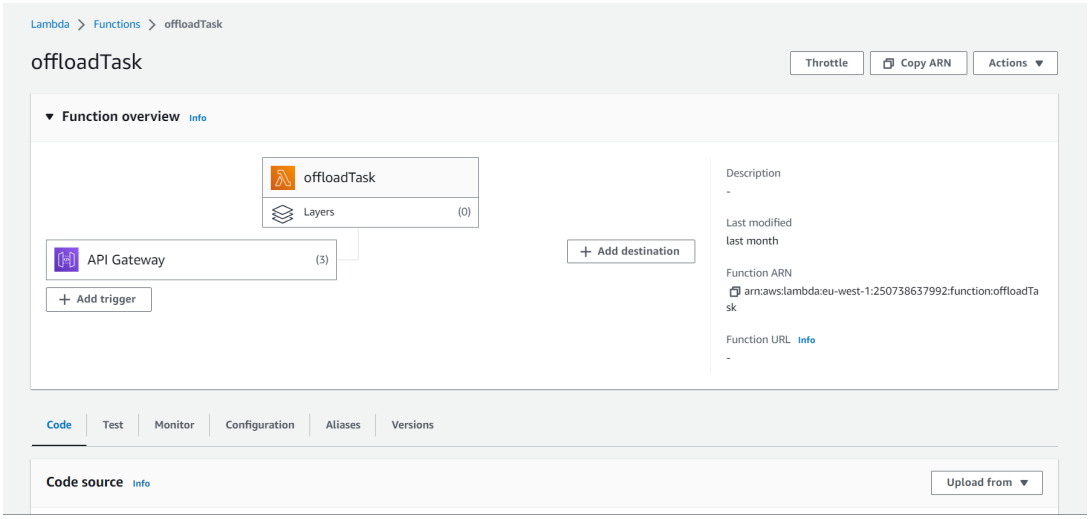

Figure 15: Lambda console

As node js is used for cloud environment development of the image rendering algorithm, all the node modules needs to be installed for it to work properly. But before that, the system must be installed with node.js post which further steps can be followed.

To check the version of node installed use command **node -v**

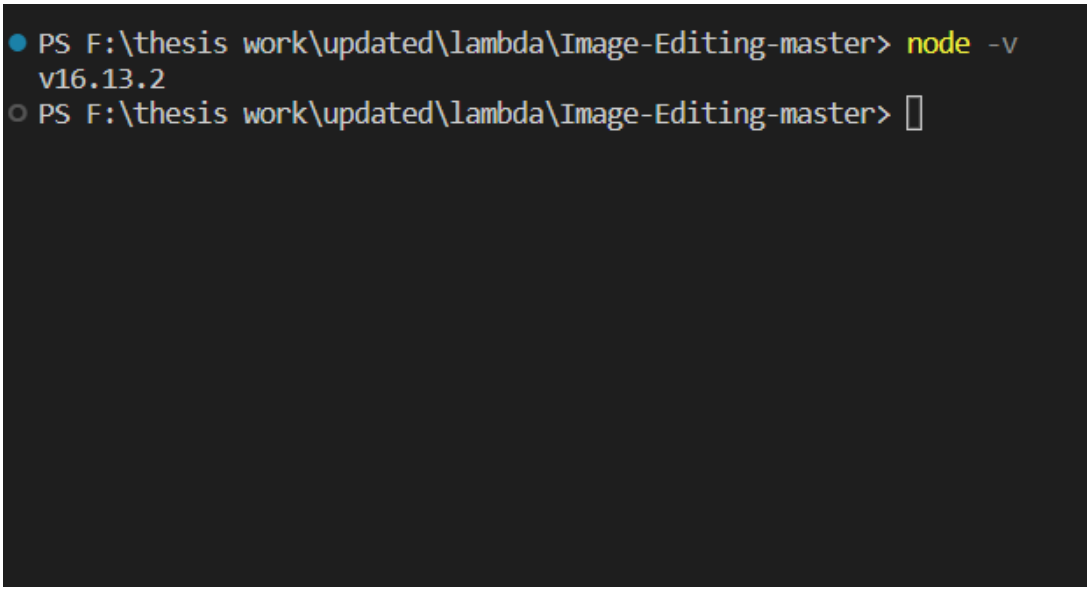

Figure 16: Node version

After the above step, install aws-sdk and jimp library using **npm install aws-sdk** and **npm install jimp** command. After installing these packages, there will be a node modules folder created with all the dependencies and the file structure will be similar to the image below.

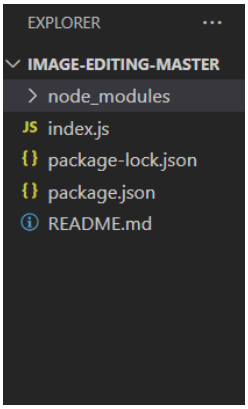

Figure 17: Project Structure

Now this project can be zipped and be uploaded on S3 bucket.

To create the S3 bucket for storing the deployment file for the image processing algorithm, search for S3 bucket in search bar in AWS .

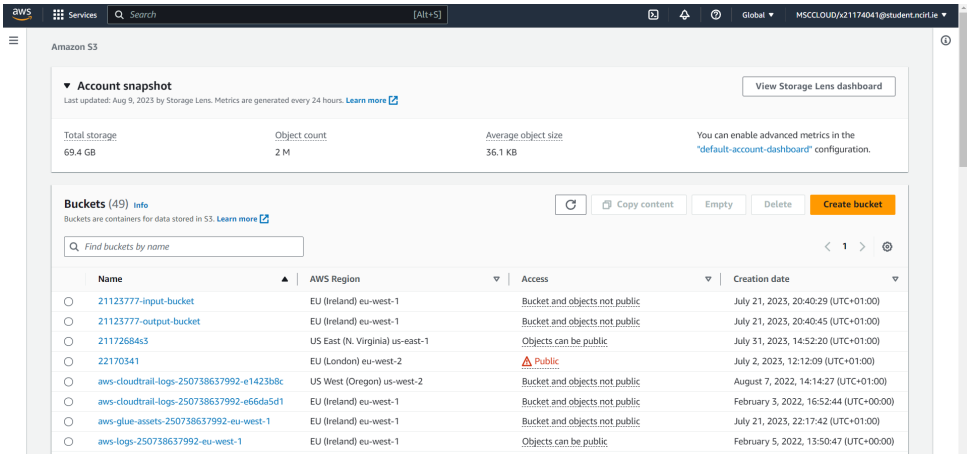

Figure 18: S3 bucket creation

Click on create bucket button and give appropriate name to the bucket. Leave all other settings as it is. After the bucket is created, the zip file can now be uploaded on this bucket for lambda to fetch and deploy it.

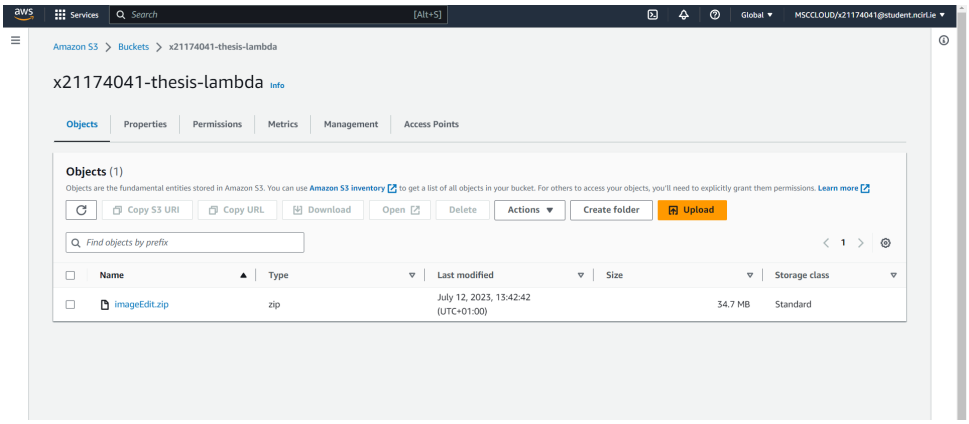

Figure 19: S3 bucket creation

Once the zip file is uploaded, go to lambda and select the lambda function which was created. Scroll down and select upload from, and click on amazon S3 location. Give the S3 link and click on save. The code will then be imported on the lambda function.

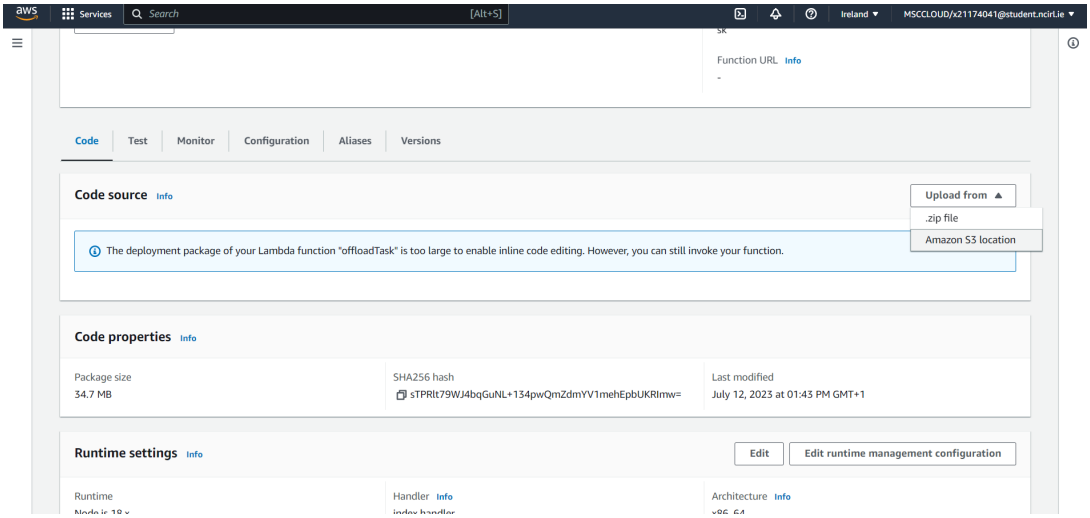

Figure 20: Upload Zip

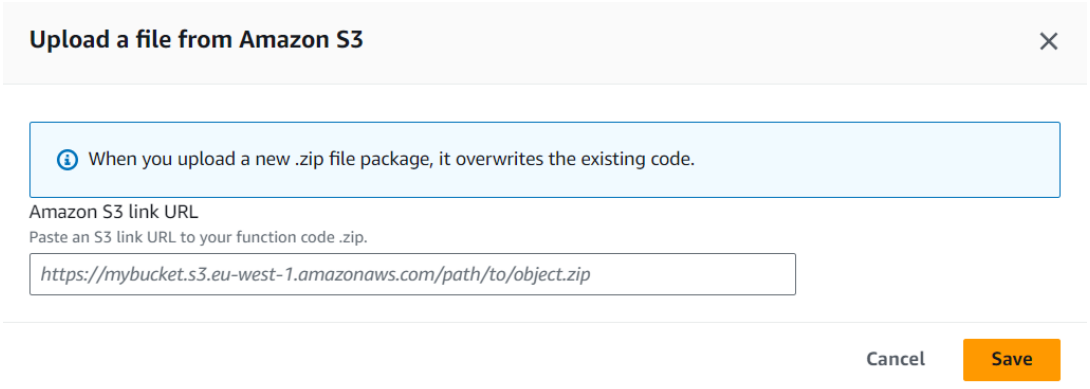

Figure 21: Giving S3 link

After the code is deployed, the system is configured and the task can now be completely offloaded on the cloud.

References

[1] android, "android-studio-flamingo-patch-2," in *Android*. [Online]. Available: https://androidstudio.googleblog.com/2023/05/android-studio-flamingo-patch-2-now.html?hl=cs