Configuration Manual

MSc Research Project
Cloud Computing

Niranjan Karunanithi
Student ID: X18177727

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Niranjan Karunanithi
Student ID:	X18177727
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	1085
Page Count:	15

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.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	17th August 2020

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

Niranjan Karunanithi
X18177727

1 Environment for Mobile Application Development

To develop android application, Android Studio 3.6.3, an Integrated Development Environment(IDE) is used.

Figure 1: Android Studio

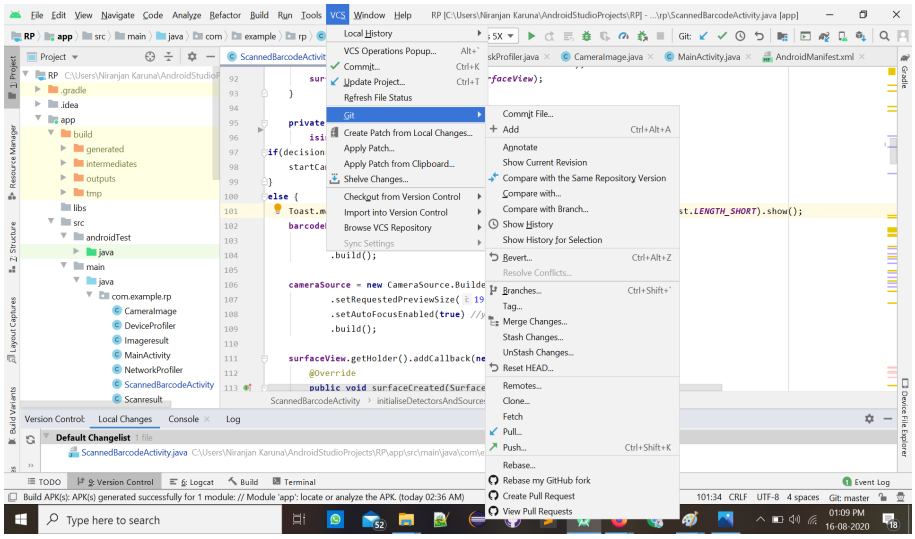

Figure 2: Android Studio and GitHub Integration

For version control, GitHub is integrated with Android studio. After creating new project in Android Studio, new repository can be created in Git through Android studio

As the project involves QR code scanner using google library, default libraries are implemented in Application level build.gradle. To use google vision library, play-services-vision:17.0.2 is implemented.

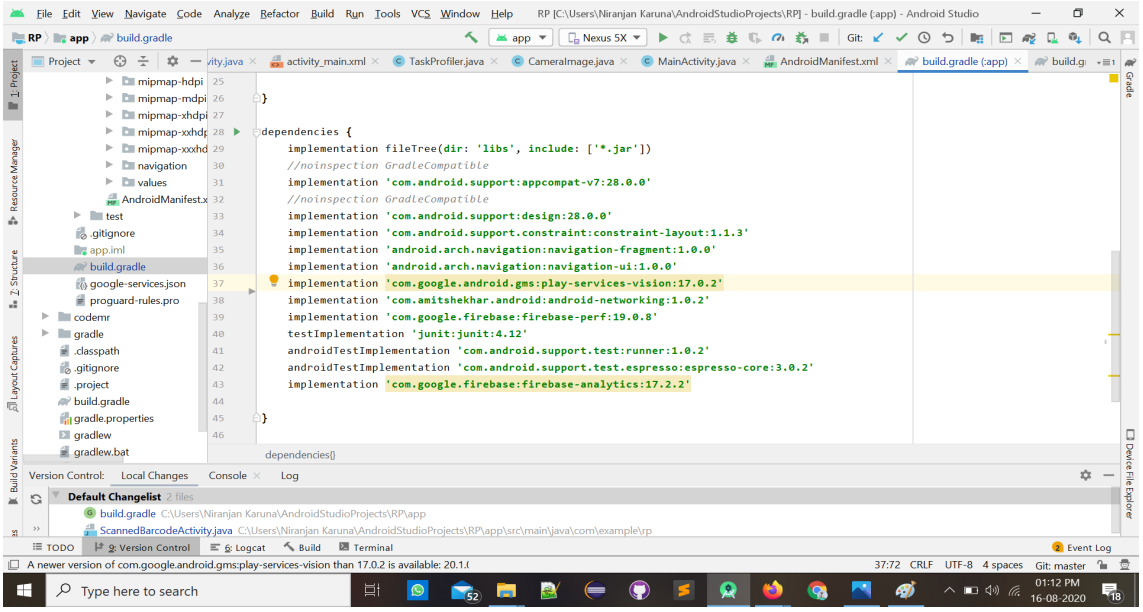

Figure 3: Android application - Build gradle (app level)

In Project level build.gradle google.services plugin and Firebase performance plugin in integrated.

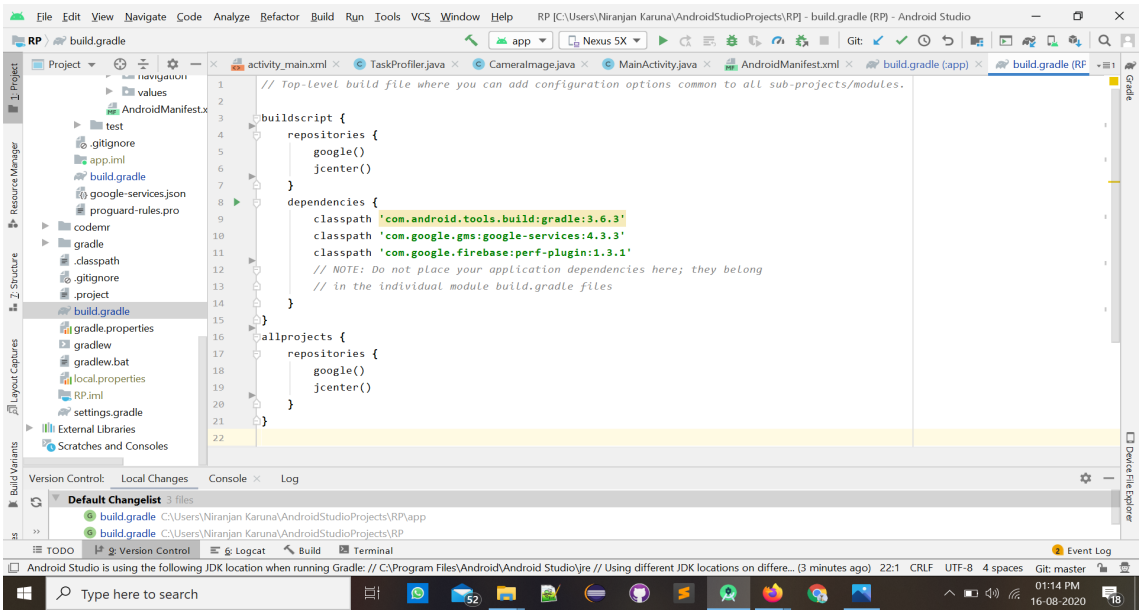

Figure 4: Android application - Build gradle (project level)

After setting up library, required Java class has to be created as Activity in Android application project structure.

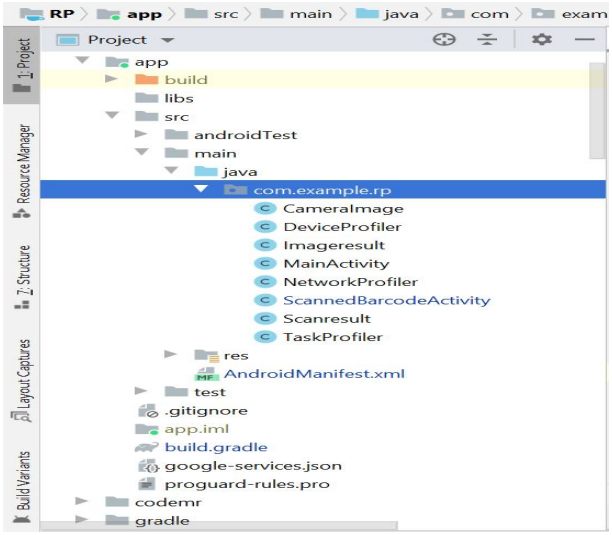

Figure 5: Android Application structure

In Android Manifest file, user permissions should be defined with all activity with respective Java classes.

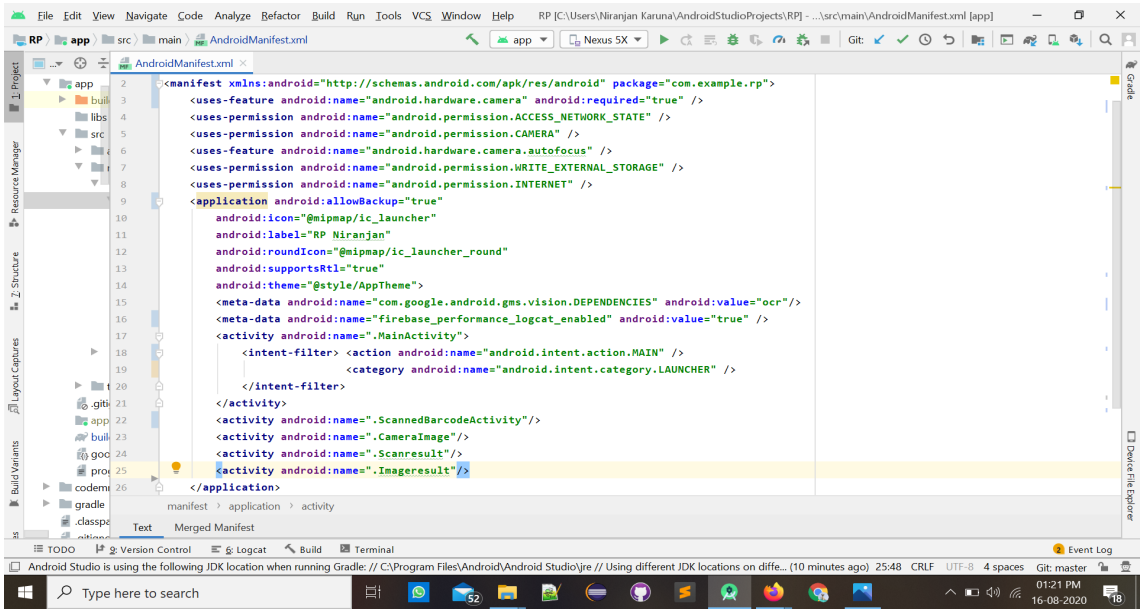

Figure 6: Android application - Manifest.xml

The complexity of the method is calculated with online tool Lizard using Java Cyclometeric method. By pasting code in respective place and selecting language as Java, it will calculate the complexity of all methods of class.

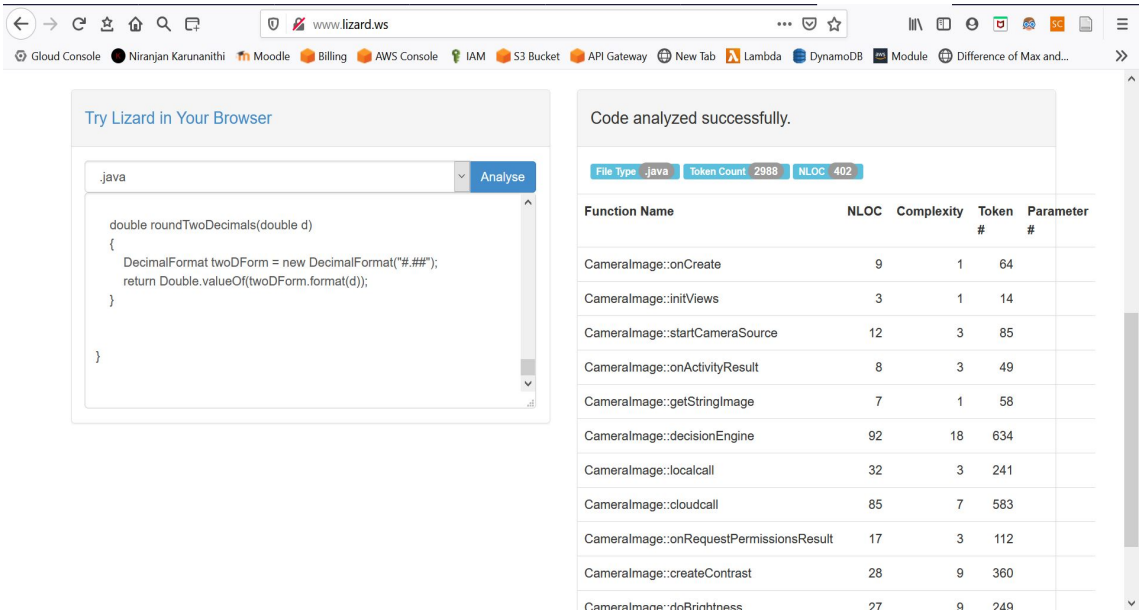

Figure 7: Complexity calculation - Lizard console

For Firebase integration, Google project should be created in Firebase console in Spark Plan(Free tier with limited accessibility)

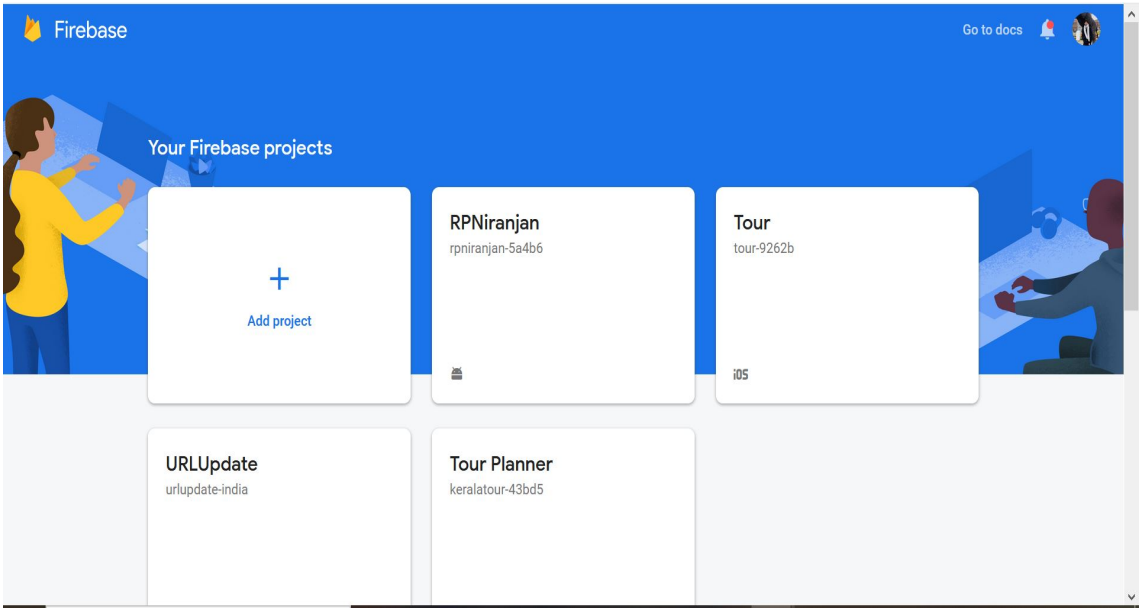

Figure 8: Firebase console - Project creation

For evaluation, Firebase performance plugin is integrated and it can be analysed in firebase console performance tab under Quality menu.

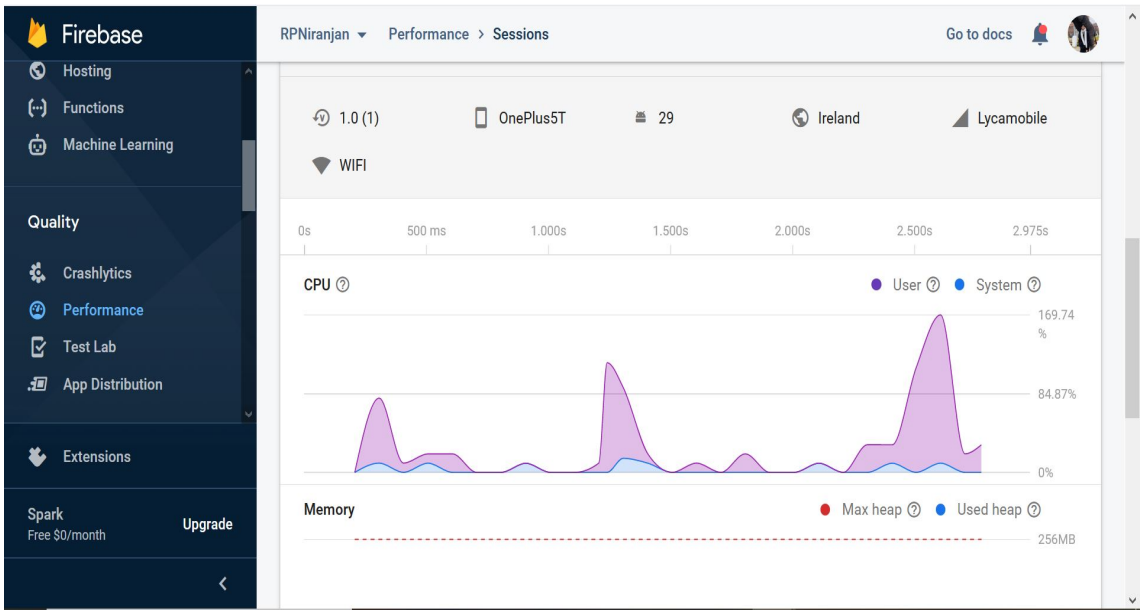

Figure 9: Firebase performance analytics

Developed Android application is tested for crashing, vulnerabilities and violations of policies in Testlab available in Firebase console.

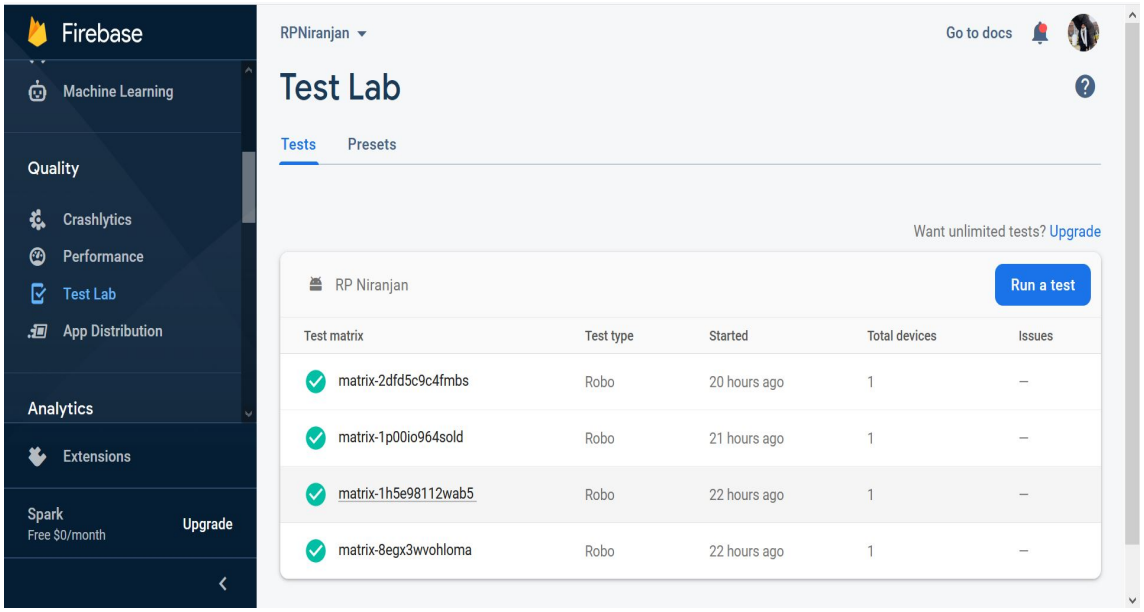

Figure 10: Firebase - Testlab

Since, Firebase is not explored fully due to time restrictions. And it can be used for analytical purpose of the developed application in future works.

2 Cloud Environment setup

As the proposed model uses cloud for offloading, AWS Cloud platform is choosed for offloading. After creating account in AWS cloud, choose AWS lambda function service.

Create function in AWS lambda function console by giving runtime environment as NodeJS 12.x

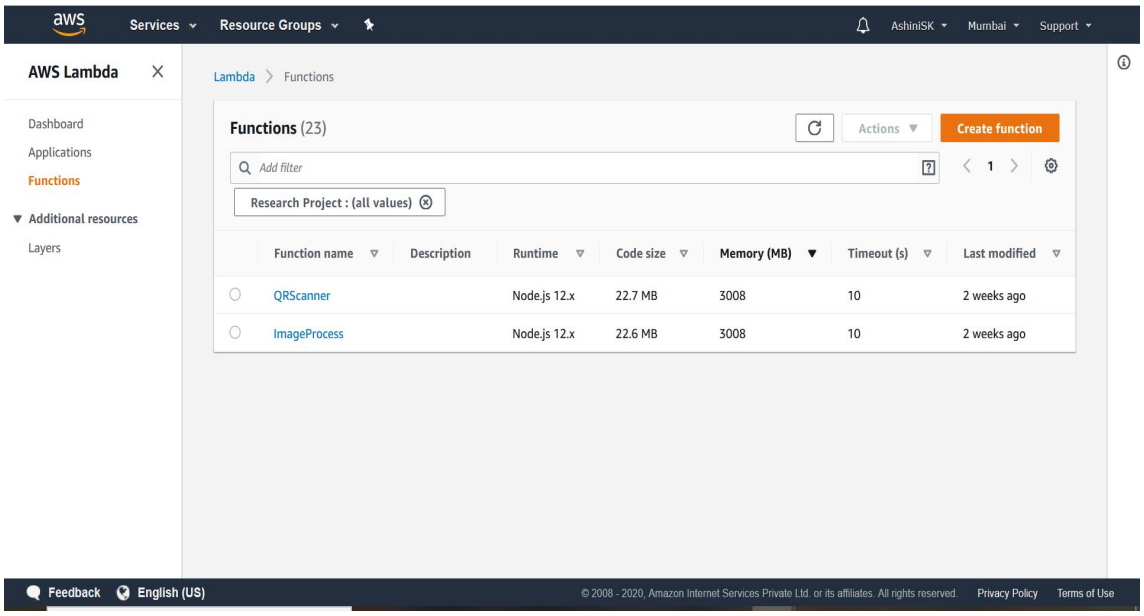

Figure 11: AWS Lambda Function console

Configuration can be changed in the console. In the proposed model, RAM memory is set to 3008 MB(maximum limit) and Time out set to 10 seconds. As the monitoring tool, cloud watch service is enabled.

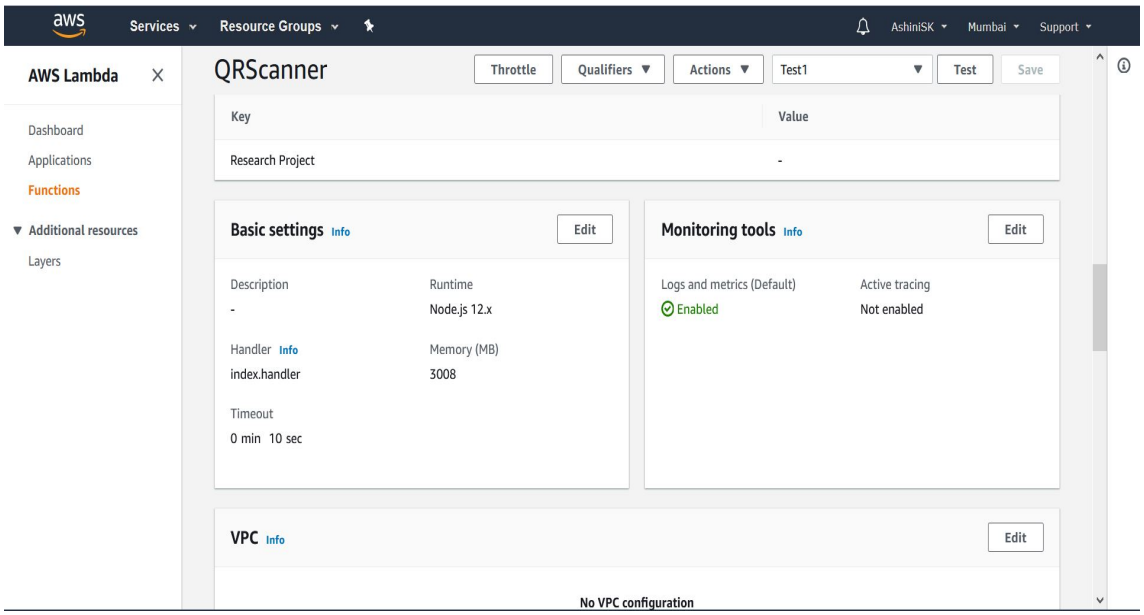

Figure 12: AWS Lambda Function configuration update

As the environment in the cloud has been ready for deployment, NodeJS application has to be zipped and uploaded to AWS lambda function. With the option Upload a .zip file and Upload a file from Amazon S3, the code can be deployed into lambda function.

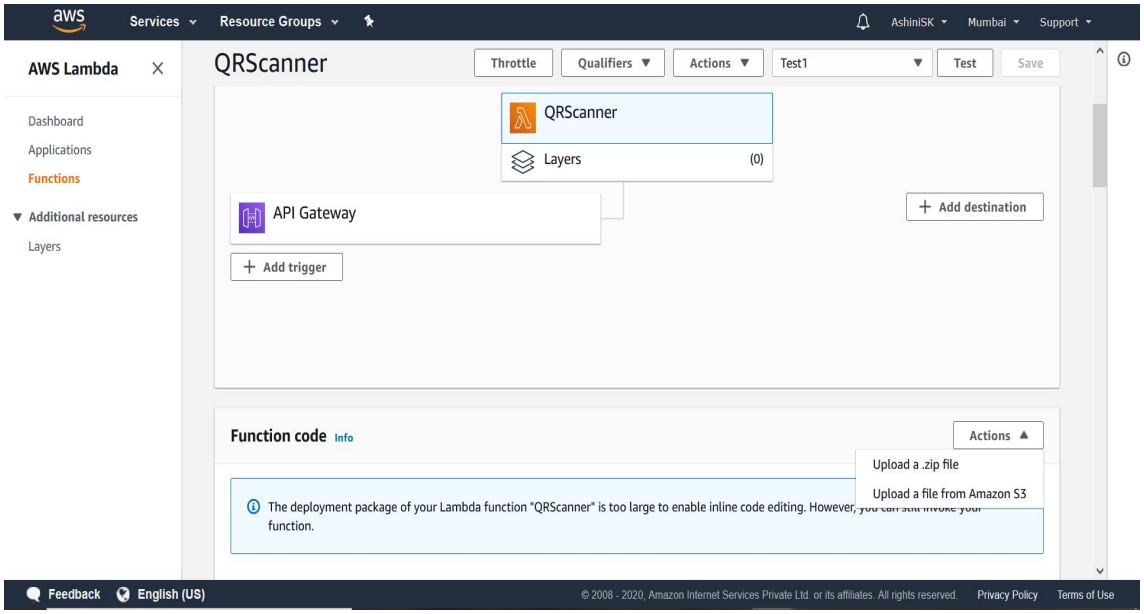

Figure 13: AWS Lambda Function - code deployment

To create NodeJS Application in local device, NodeJS and npm should be installed to the local machine. It can be downloaded from the official website nodejs.org. As the local machine has Windows OS, Windows installer is downloaded and installed.

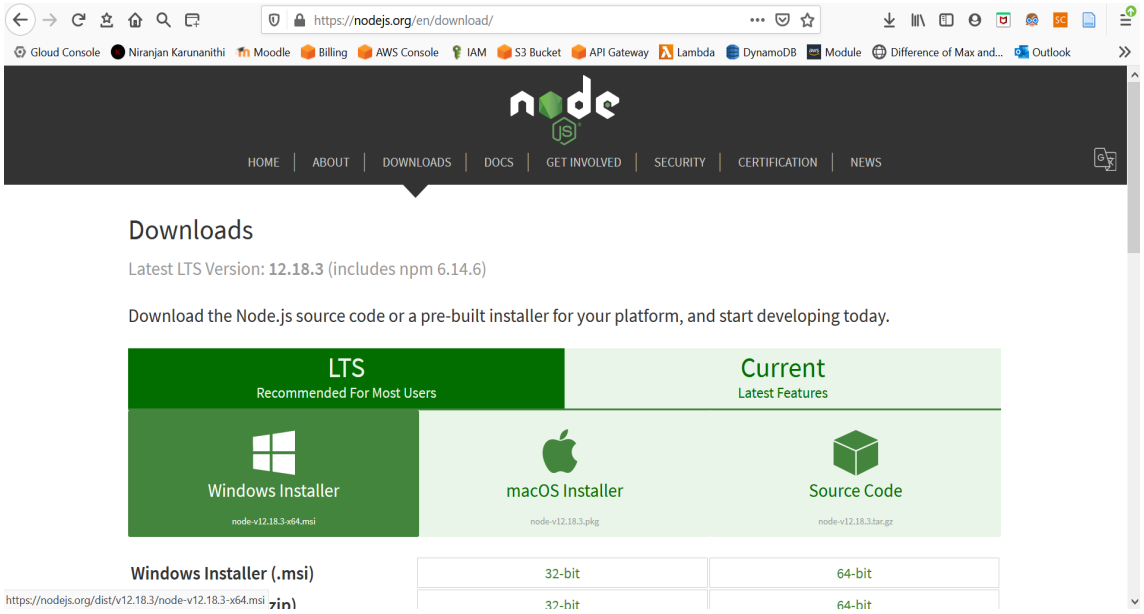

Figure 14: NodeJS and Npm installation

Installation verification can be done in command prompt with following command,

1. node -v
2. npm -v

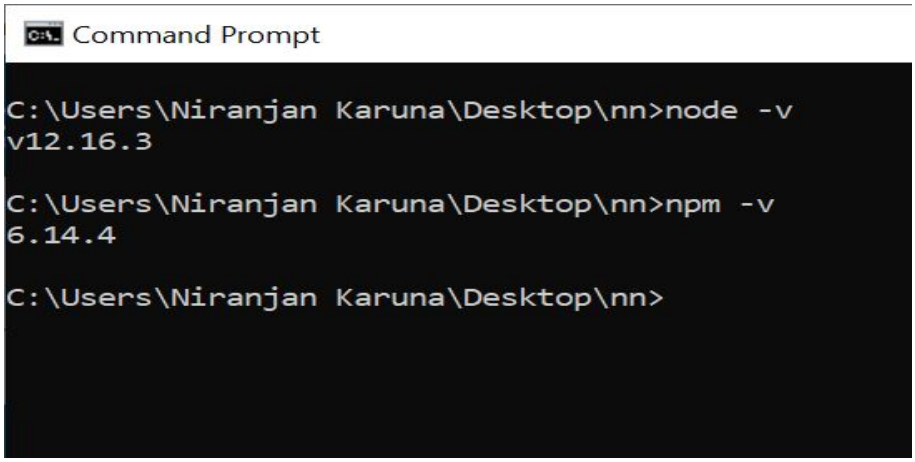

Figure 15: NodeJS and Npm installation verification

To create the NodeJS application, create folder with index.js and package.json file

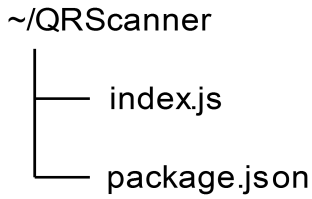

Figure 16: Folder structure for NodeJS application

Navigate to the folder in command prompt, and install required libraries for QRScanner. Jimp and qr-codereader libraries are used for QRScanner cloud execution. Libaries can be installed with the following command.

1. npm install jimp
2. npm install qrcode-reader

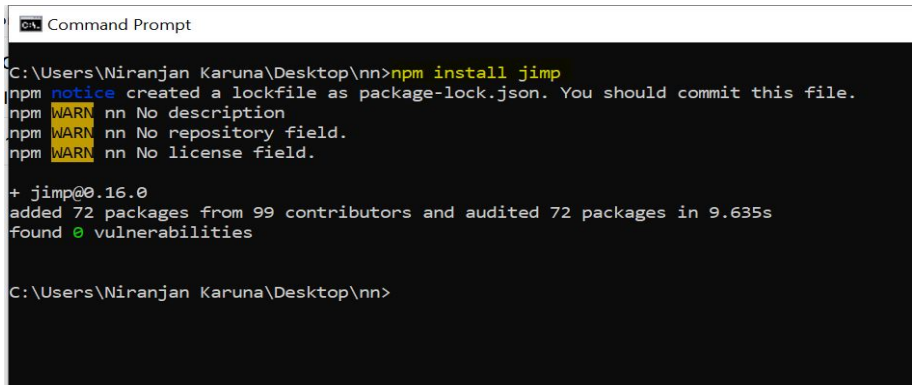

Figure 17: NodeJS Jimp libary installation

Now the project structure will get changed with node_modules library. The project folder should be zipped with NodeJS application structure and this steps are explained by Hendrix (n.d.) in AWS documentation.

Same steps should be followed for Image editing NodeJS application but Image editing only uses Jimp library.

```
~/QRScanner
├── index.js
├── node_modules
├── package.json
└── package-lock.json
```

Figure 18: NodeJS application structure

```
~/QRScanner.zip
├── index.js
├── node_modules
├── package.json
└── package-lock.json
```

Figure 19: NodeJS application Zip file structure

Zipped NodeJS application size exceeds 10MB, So it is recommended to use Amazon S3 bucket for code deployment. Amazon S3 bucket should be created in Amazon S3 console.

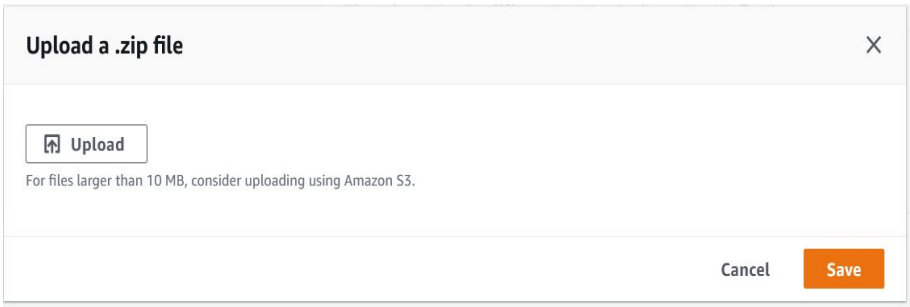

Figure 20: Code deployment suggestion from AWS

Zipped NodeJS project will be uploaded to Amazon S3 bucket with the Web console.

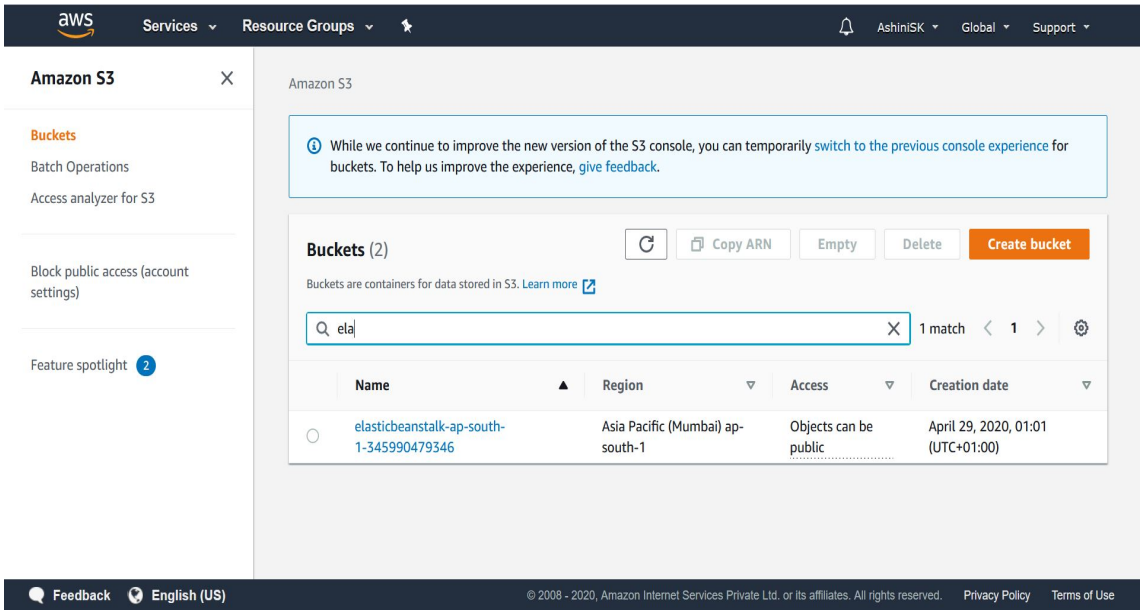

Figure 21: AWS S3 bucket creation

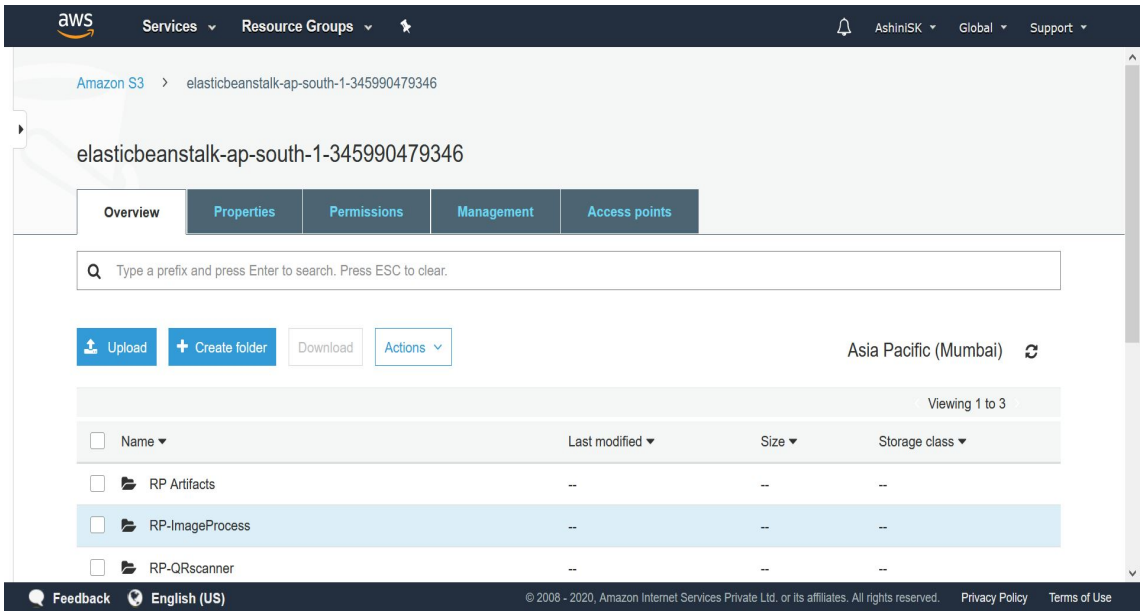

Figure 22: Uploading Zip files to AWS S3 bucket

3 API Integration

As the proposed model using REST API, AWS API Gateway service is used for RESTful API calls. REST API is selected in API Gateway service while creating API.

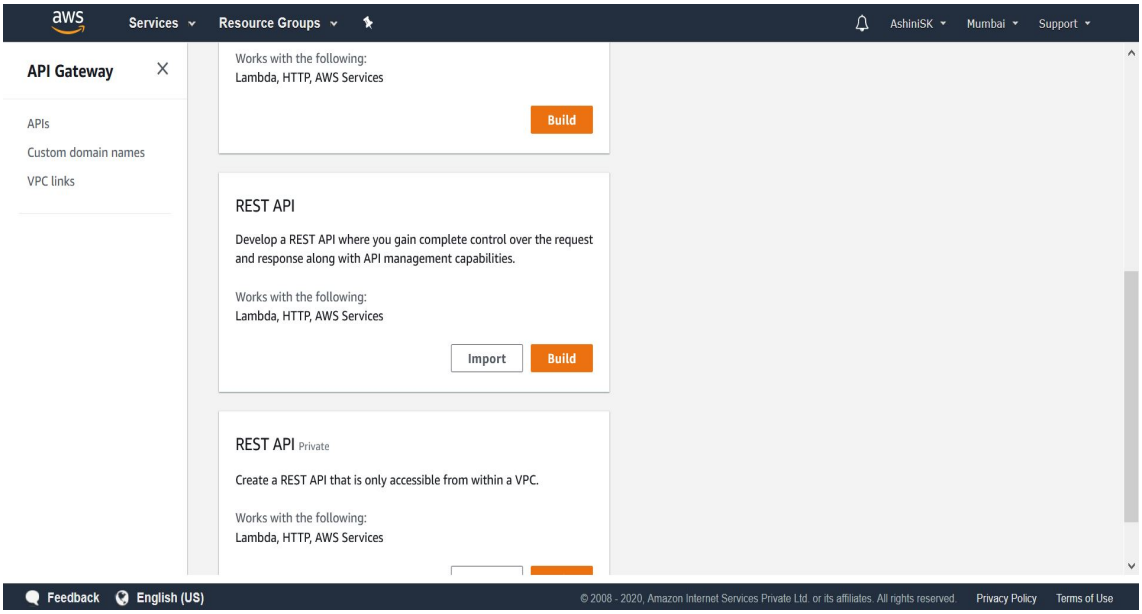

Figure 23: AWS API Gateway - API creation

At first, Resource should be created with the Action button in AWS API Gateway console.

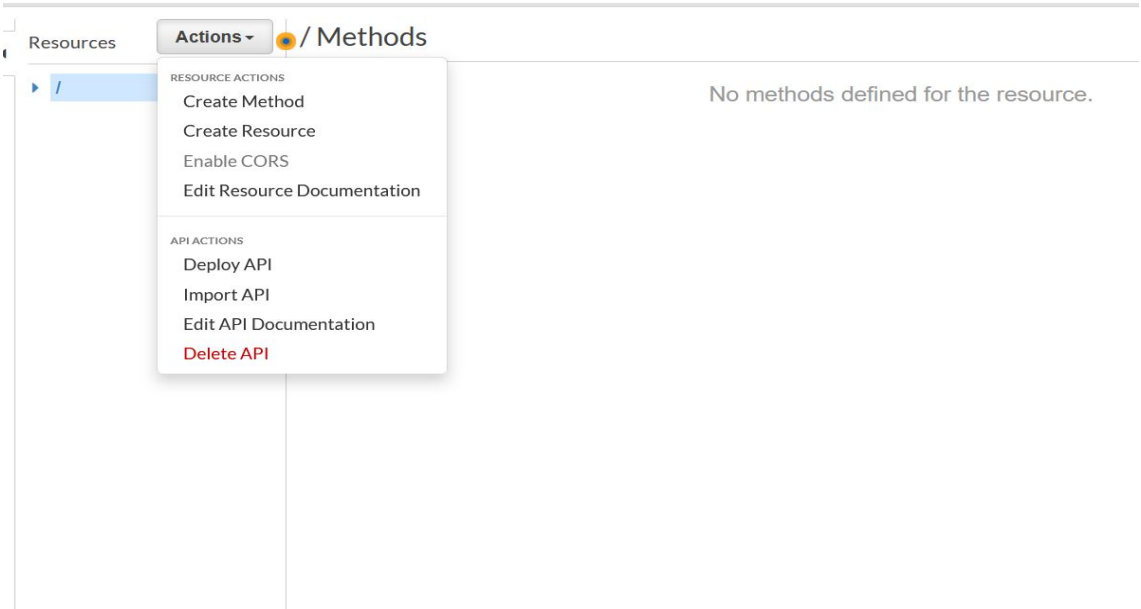

Figure 24: AWS API Gateway - Resource creation

By giving name for resource, resource path will get change as same as resource name. For security perspective, CORS policy should be enabled.

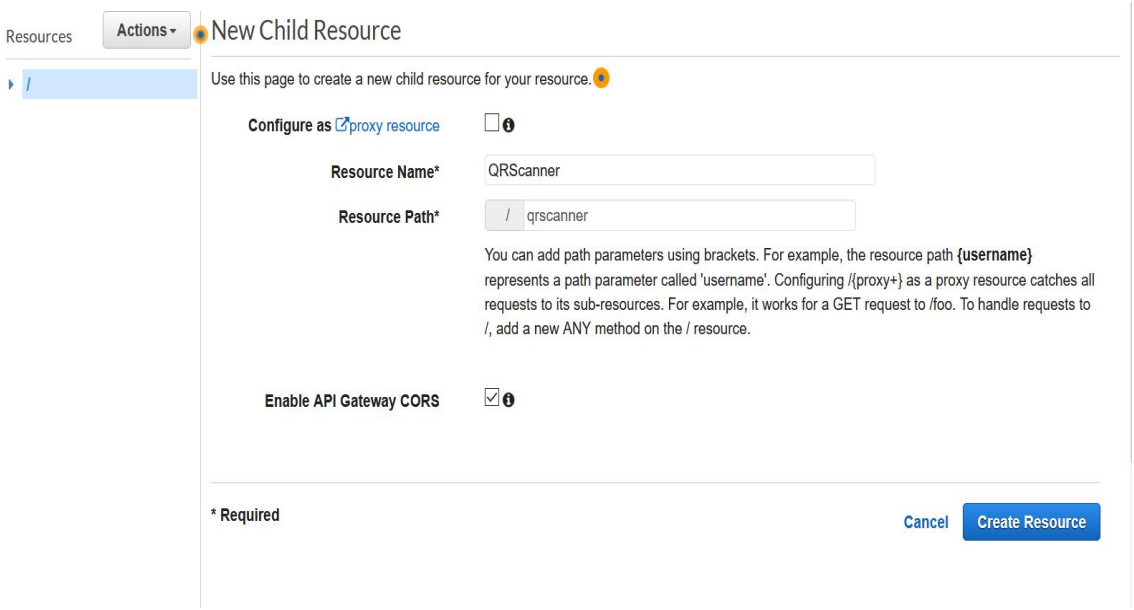

Figure 25: AWS API Gateway - Resource configuration

After creating resource, corresponding method should be created. In the proposed model, both Image editing and QR scanner will send image to AWS lambda function. SO POST method is created for both lambda function. While creating method, lambda functions also integrated with the API Gateway.

Lambda Proxy integration should be checked, so there is no need to rephrase the response from lambda functions. Default timeout for API Gatway is 29000 milliseconds which is 29 seconds. The Deployment region has been selected to ap-south(Asia/Pacific-south) which is Mumbai India.

Respective lambda functions should be selected in Lambda function text box.

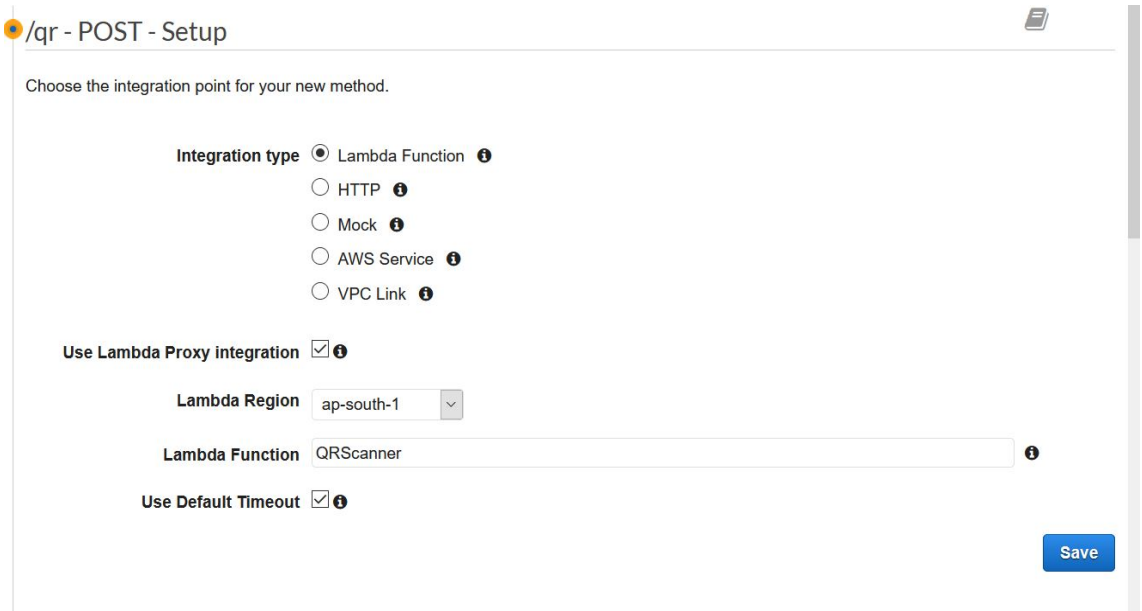

Figure 26: AWS API Gateway - Lambda function integration

By using AWS Cognito service, Authorization for request can be given, as developed project has no Cognito users as there is no Registration or Login functionality in Mobile

application. In future, it can be set in the POST Method setting by creating cognito pool users in AWS Cognito Service.

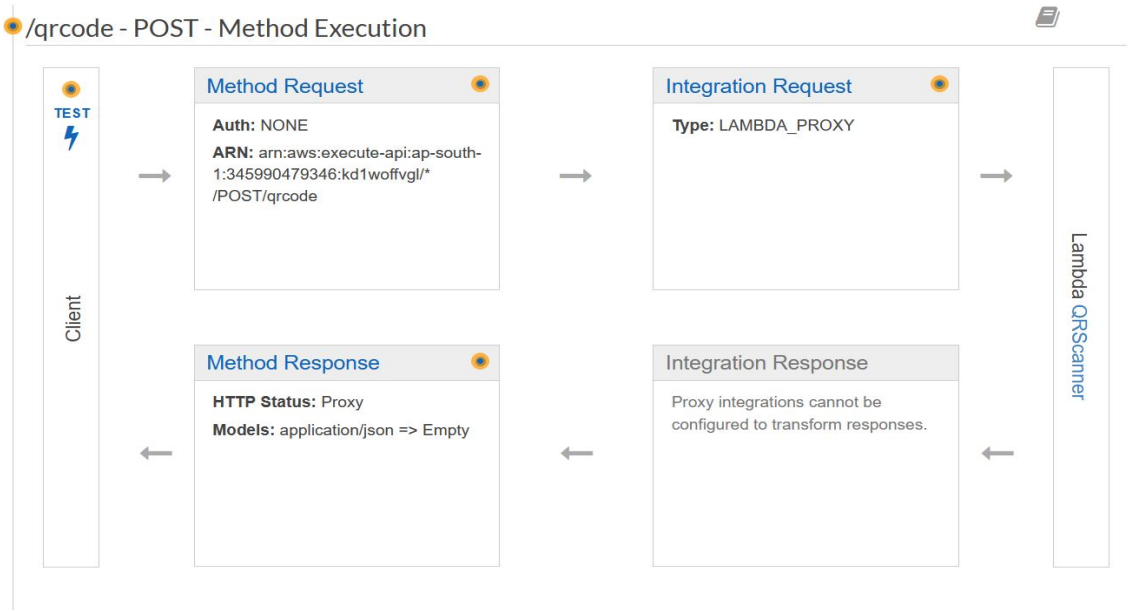

Figure 27: AWS API Gateway - POST method execution flow

After integrating Lambda functions to API Gateway, It should be deployed. From the Action button at the top, API can be deployed.

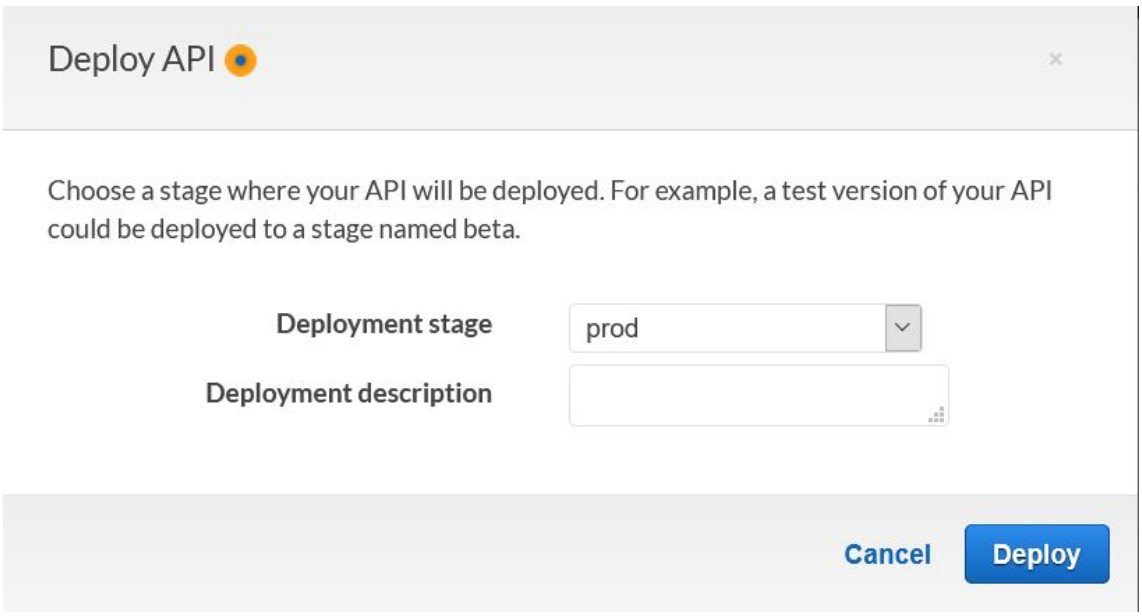

Figure 28: AWS API Gateway - API Deployment

Url will be generated from API Gateway and it should be used for API calls with required parameters.

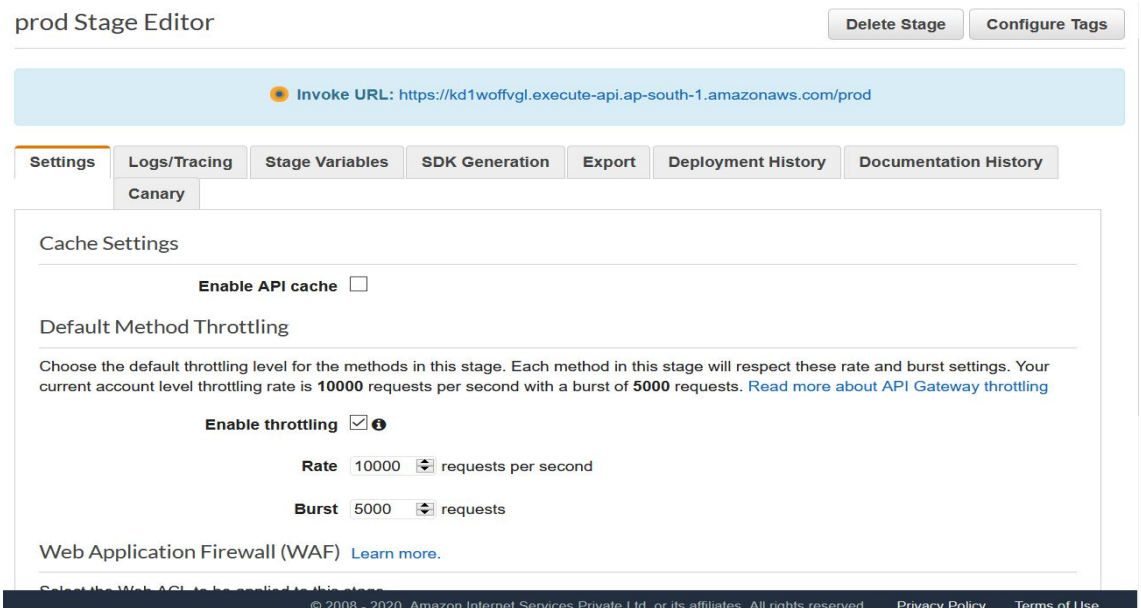

Figure 29: AWS API Gateway - Deployed API

4 Repositories

Android Application Git Repository link :
https://github.com/niranjankaruna/Android-Application

QRScanner Lambda function Git Repository link :
https://github.com/niranjankaruna/QRScanner

Image Editing Lambda function Git Repository link :
https://github.com/niranjankaruna/Image-Editing

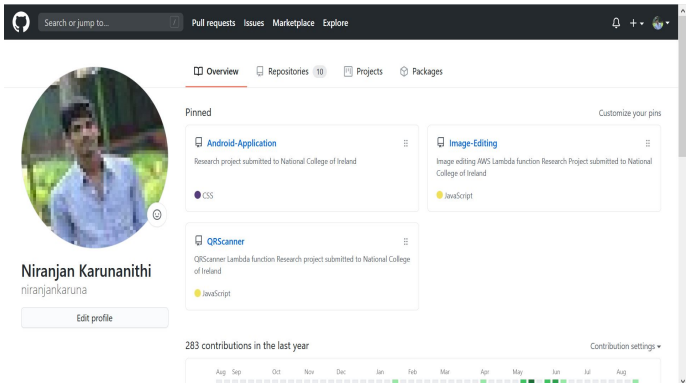

Figure 30: Repositories for Android application and Lambda functions

References

Hendrix, R. W. (n.d.). Lambda.
URL: *https://docs.aws.amazon.com/lambda/latest/dg/nodejs-package.html*