International Institute of Information Technology, Bangalore

Software Production Engineering Mini Project

Outgoing Management Portal

TA : Vaibhav Aggarwal

Guide : Prof. B. Thangaraju

Archit Semwal	Mohit Bansal	Shashank Agarwal
MT2019026	MT2019065	MT2019100

CONTENTS

1. ABSTRACT

Most institutes in our country become home to a large number of scholars every year. These widespread universities often have several entry and exit points which become the gates of influx to a variety of crowd including hostelers, day-scholars, faculty, staff and visitors. In such scenario, it becomes utterly important to have a record about the passage of people within the college premises. Outgoing Management Portal is a robust webapp which aims to bring the college authorities, students and their guardians under the same roof about the whereabouts of their ward. The portal is primarily developed using DevOps toolchain to store the incoming and outgoing activities of the students and provide timely notification to its respective users.

2. INTRODUCTION

About the web application

‘Outgoing Management Portal’ is a webapp which majorly capitalizes on bringing the college authorities, students and their guardians together to some extent of information. The webapp provides a common ground for students to apply for a leave or an outpass while it lays out an effective interface for the college/hostel incharge to look through the leave requests and permit accordingly. The web app also serves as a platform for the college security staff to record the incoming/outgoing of the students as well as visitors through it. The application has been programmed to implement an automatically triggered e-mail notifier too.

Scope of the project and features:

- The project efficiently solves the problem statement of managing and storing student outdoor visits along with time details.
- Allows a student to apply for leave and get real time updates on it. As soon as the student files for a leave from his account, a request card is generated at the warden’s dashboard for approval. The response is autimatically sent via e-mail to the student in real-time.
- Automates a warning e-mail notification whenever a student exceeds the local out pass time limits.
- The portal successfully replaces all kinds of manual entry by the security staff at any gate by providing a common platform to register leaves, record local outings and maintain visitor entries.

Why DevOps?

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

This speed enables organizations to better serve their customers and compete more effectively in the market. Under a DevOps model, development and operations teams are no longer “siloed.”

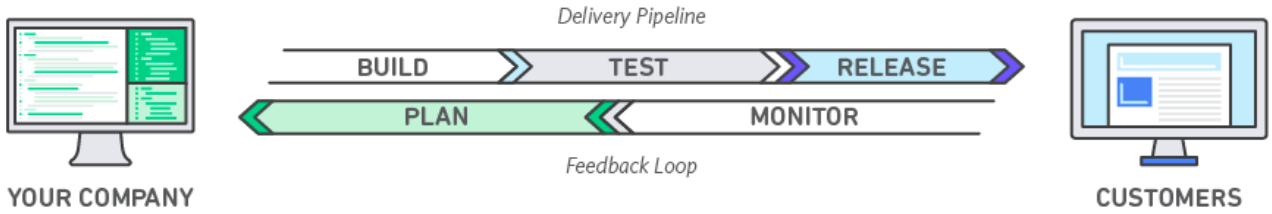

Fig 1: Basic devops model

We used DevOps toolchain in our model due to its various advantages as elaborated ahead.

Speed - Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results.

Rapid Delivery - Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage.

Reliability - Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users.

Scale - Operate and manage your infrastructure and development processes at scale. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

Improved Collaboration - Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time.

3. Workflow Diagram

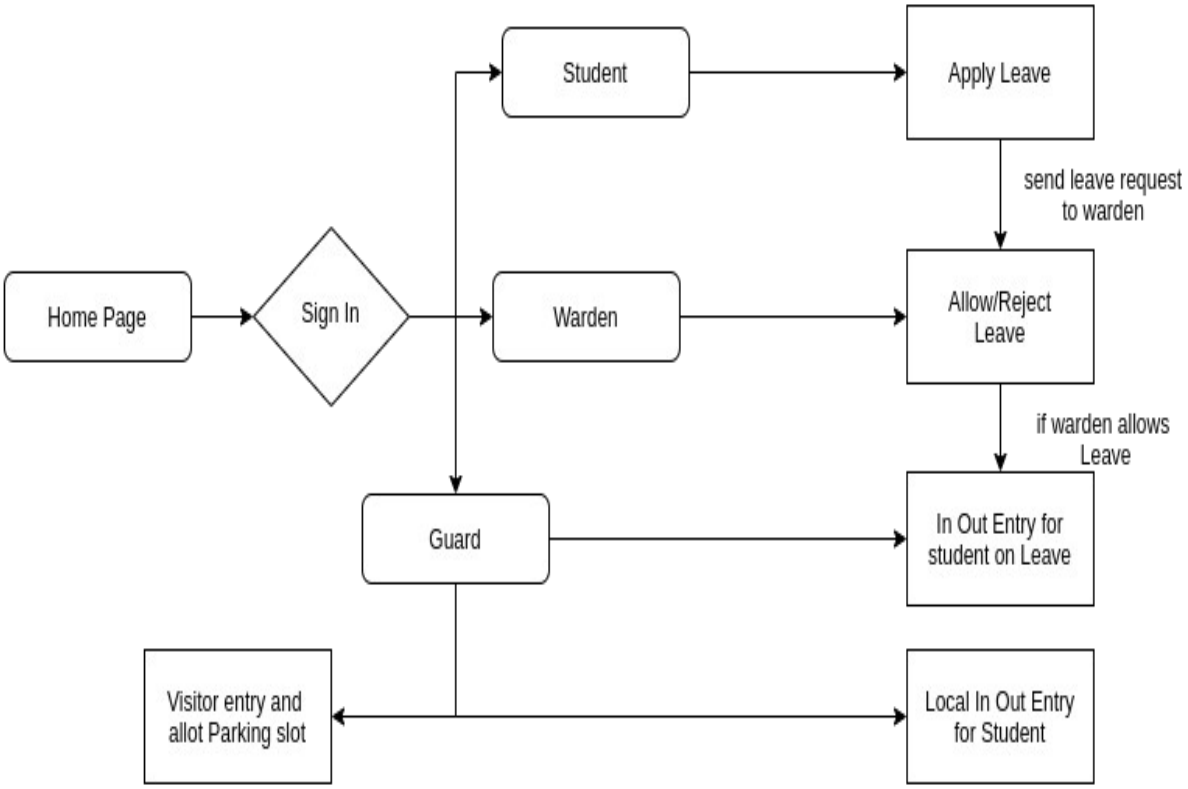

Fig 2: Workflow diagram of Portal

CODE FLOW

The root folder of the project contains the following files and directories :

- **server.js** : It is the main backend control file, coded in nodeJS and responsible for managing the control flow of the web application on the local system.
- **server2.js** : It is the docker backend control file, coded in nodeJS and responsible for managing the control flow of the web application when hosted using the docker container.
- **public directory** : This directory contains the heirarchical structure of all the resources used in developing the webapp including- Html pages, css stylesheets, js scripting files, bootstrap/JQuery resources along with the images used.
- **package.js** : This file actually contains all the nodeJS dependencies that have been employed for the web portal. It automatically creates the node_modules directory and installs the required dependencies.
- **db directory** : It contains the mysql docker file along with another file which describes the database schema.
- **Dockerfile** : It is the actual docker image of the web application.
- **jenkinsfile** : This file contains the pipeline script defining various stages of the pipeline.
- **test** : This directory contains the my.test.js file which is out testing file containing the defined test cases for our webapp.

4.SYSTEM CONFIGURATION

System Architecture:

Ubuntu version – 16:04 LTS
CPU configuration – 4 cores
RAM – 4GB
HDD –120GB
kernel version –4.15.0-96-generic

Project Architecture:

Frontend Framework – HTML/CSS, Bootstrap
Backend Framework – Nodejs
Database – MySql
Server – Docker Container
Github Repo – https://github.com/m0hitbansal/Outgoing-Managment-Portal.git
DockerHub Repo – https://hub.docker.com/repository/docker/m0hitbansal/outgoing-webapp

DevOps Toolchain:

SCM – Git
Build – npm
Continuous Integration – Jenkins
Containerization Tool – Docker
Continuous Delivery – Rundeck
Continuous Testing – Chai, Mocha
Monitoring & Logging – ELK Stack (Winston ,Elasticsearch, Kibana)

5.Software development Life Cycle

5.1 Source Code Management

GitHub is used for SCM. It is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

GIT has various benefits as VCS(Version Control System) listed as follows :

1) Revert the code files back to their previous state.
2) Recall and revert the entire project back to its previous state.
3) Compare code changes over specific durations of time.
4) Find who last modified a piece of code that might be causing an issue or a problem. Who introduced a particular issue and when.

Advantages of using git:

• When multiple people collaborate on a project, it's hard to keep track revisions who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.
• Git branching model lets you have multiple local branches which are independent of each other. Having this also enables you to have friction-less context switching.

URL of the GIT repository:

GitHub: ***https://github.com/m0hitbansal/Outgoing-Managment-Portal.git***

Steps used to add code to my git repository:

i) git init ./
ii) git remote add origin https://github.com/m0hitbansal/Outgoing-Managment-Portal.git
iii) git push -f origin <branch name>

To create a new Project, we can do a git clone:

$ git clone https://github.com/m0hitbansal/Outgoing-Managment-Portal.git

To push the code on to repository follow following commands:

i) git add .

ii) git commit -m “Commit message name”

iii) git push origin <branch name>

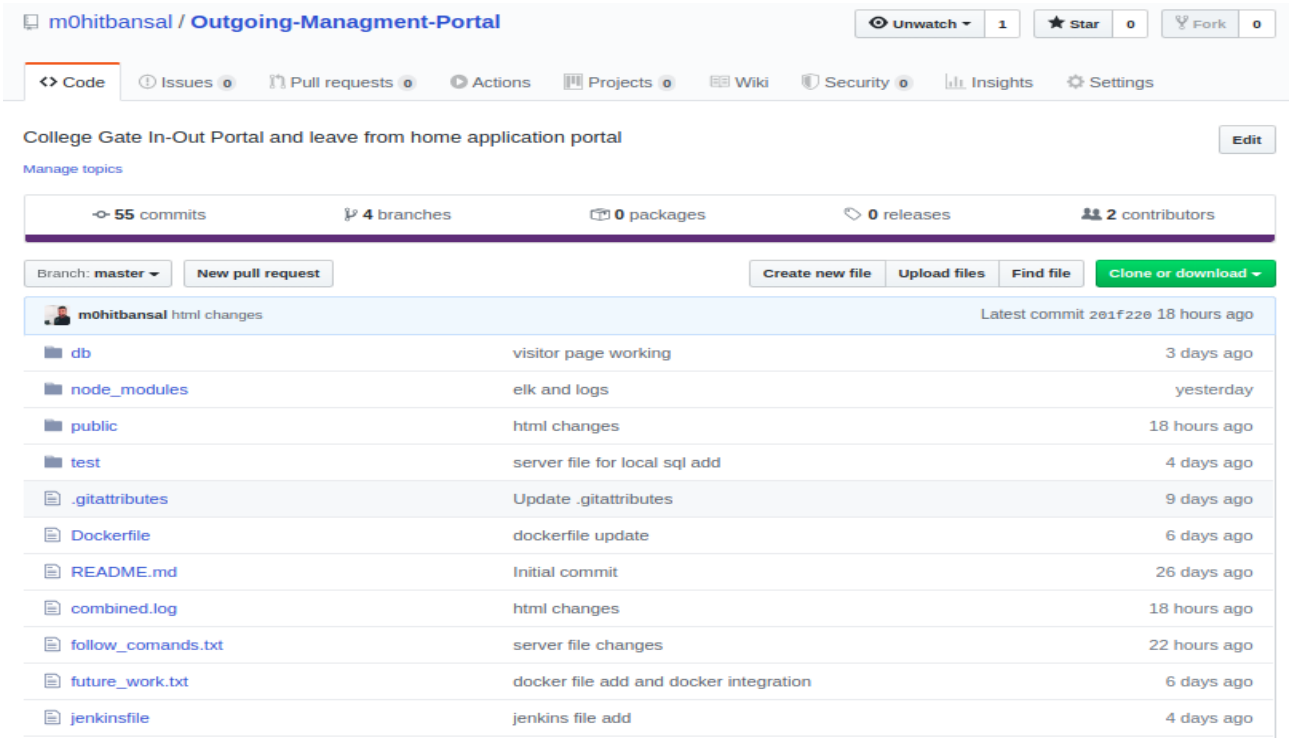

Fig 3: Github Repository

5.2 Build

NPM with Node.js
NPM is the default dependency management tool for node applications and is often used as a simple build tool as well. It works with all the node modules and with a single command, can fetch all the dependencies
$ npm install
and then build the project using
$ npm build
Again, it is extremely flexible and can easily be setup with ‘package.json’ file. It can be easily configured to work with express.

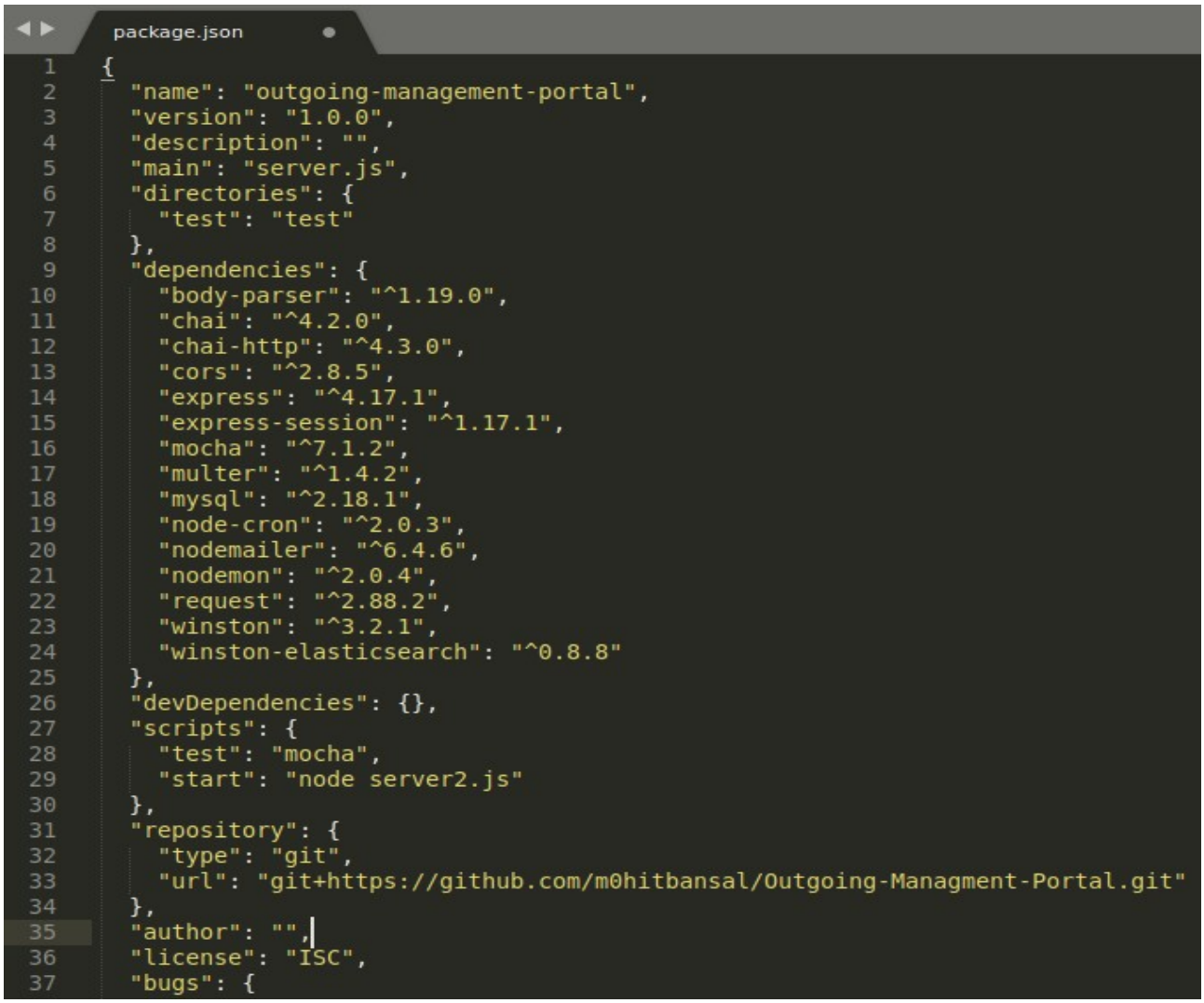

Fig 4: package.json file

5.3 Testing

Mocha

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. The main advantage with Mocha is that it can seamlessly integrate with your node application, and supports various assertion libraries.

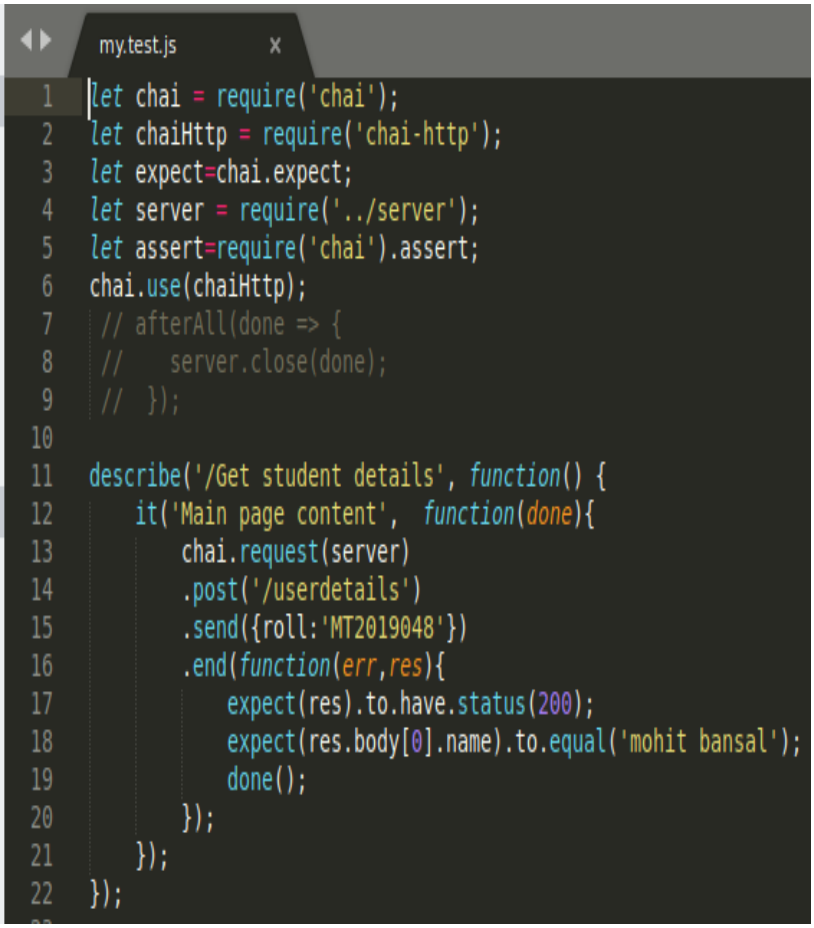

Fig 5: Test file

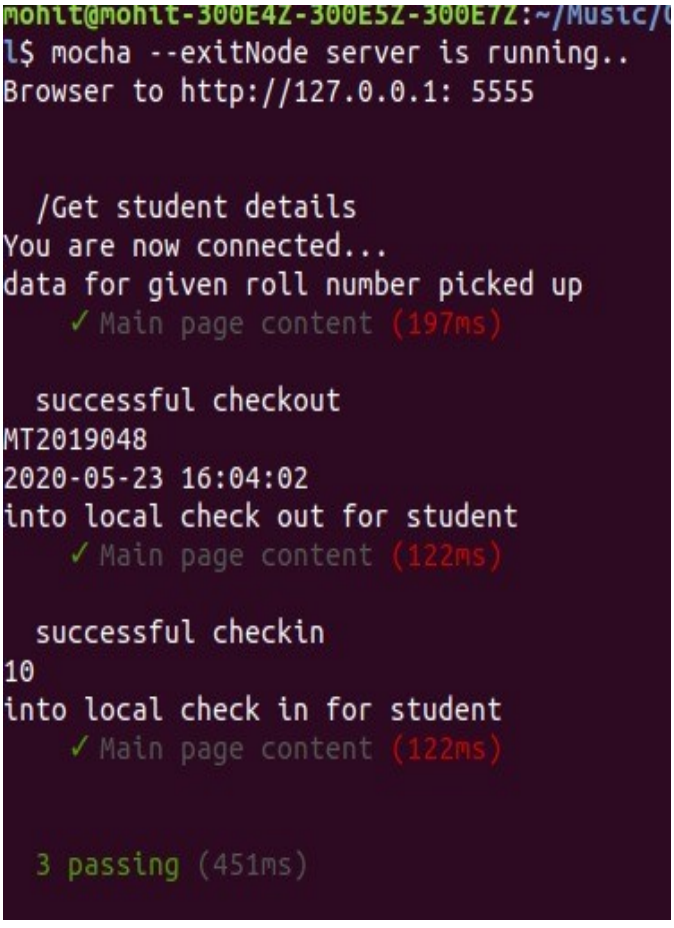

Fig 6: Test file execution

To run this test file we use command
$mocha –exit

Chai

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

Why chai?

Actually Mocha provide the environment for making our test, But we need to test our API and our API using http calls like GET, PUT, DELETE, POST etc. So we need a assertion library to fix this challenge. Chai helps us to determine the output of this test case.

5.4 Docker Artifact

Docker is officially defined as "*A set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers.*" A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

A Dockerfile is a text file written in an easy-to-understand syntax that includes the instructions to build a Docker *image.*

We used Docker for setting up Mysql, Frontend and Backend (NodeJS) of our application.

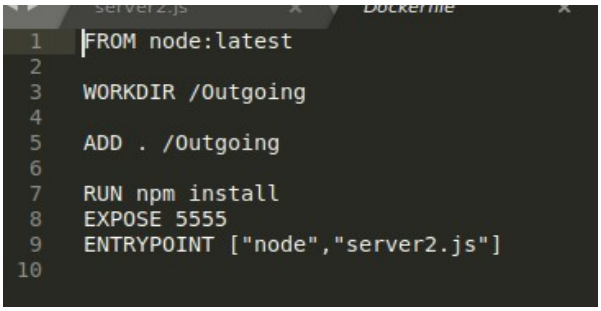

Fig 8: Dockerfile for webapp

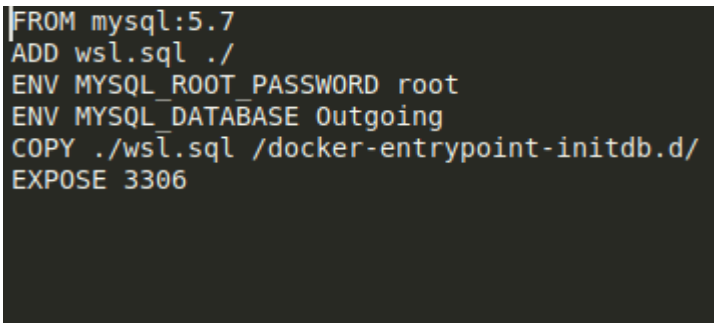

Fig 7: Docker file for mysql database

Create an account on docker hub and create a repository and the image's name will be <dockerhub-username>/<repository-name>for example, m0hitbansal/outgoing-webapp
The following commands are used to create a docker image using dockerfile. (keep the Dockerfile in the same directory)

```
$ su
$ docker build -t m0hitbansal/outgoing-webapp .
$ docker build -t m0hitbansal/outgoing-sql .
$ docker login
```

$ docker push m0hitbansal/outgoing-sql
$ docker push m0hitbansal/outgoing-webapp
$ docker run -it --name websql -d m0hitbansal/outgoing-sql
$ docker run --link websql:db -e DATABASE_HOST=db --name webapp -p 5555:5555 m0hitbansal/outgoing-webapp

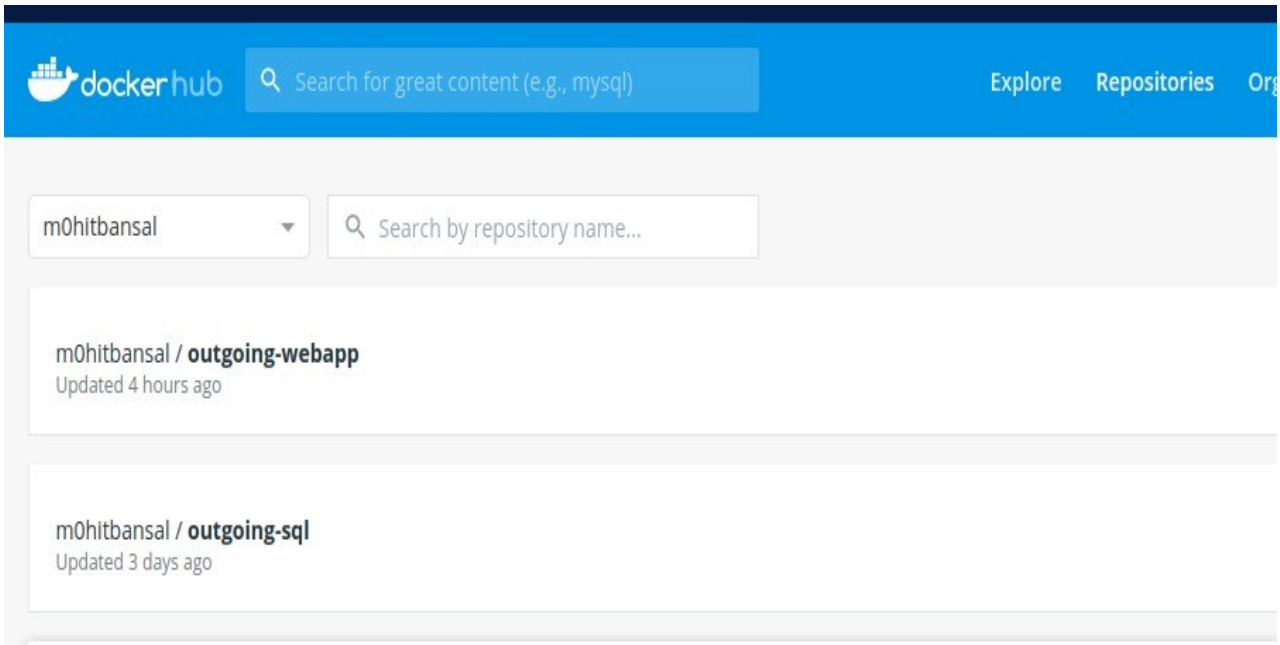

Fig 9: Dockerhub-Repository

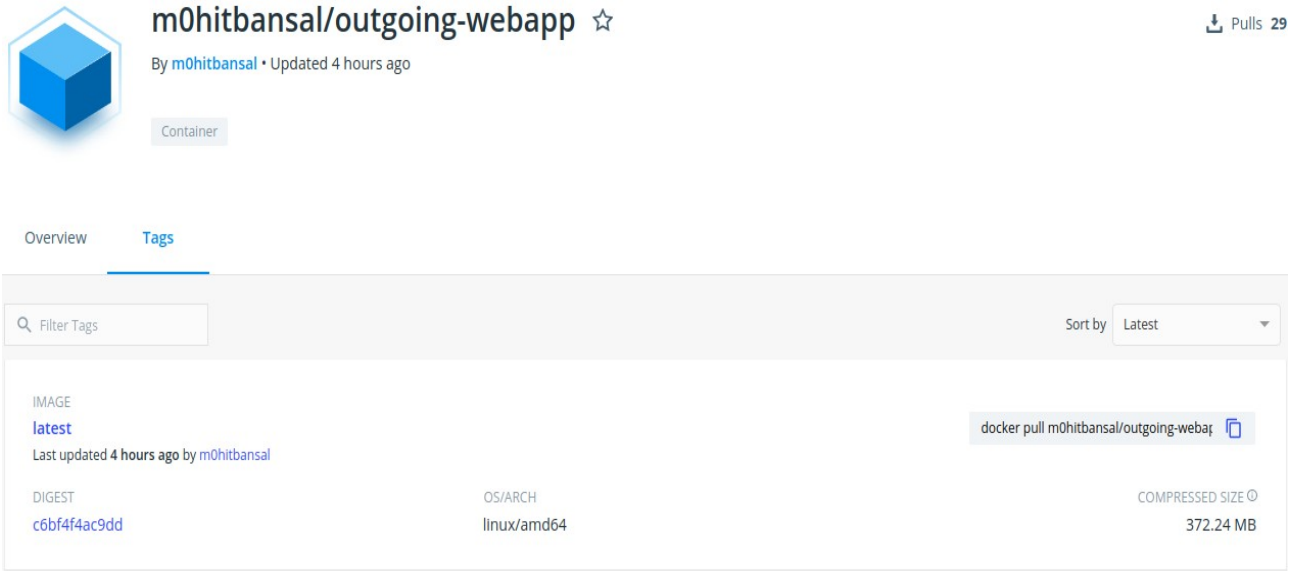

Fig 10: Image on dockerhub Repository

5.5 Continous Integration

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily.

JENKINS

Jenkins is an open source CI tool written in Java, platform independent and easy to use. The user interface is simple, intuitive and visually appealing. It has a very low learning curve. It is extremely flexible and hundreds of open source plugins are available with more coming out every week.These plugins cover everything from version control systems, build tools, code quality metrics, build notifiers, integration with external systems, UI customizations, and much more.

Advantages of using Jenkins

- Jenkins is being managed by the community which is very open. Every month, they hold public meetings and take inputs from the public for the development of Jenkins project.
- As technology grows, so does Jenkins. So far Jenkins has around 320 plugins published in its plugins database.
- Jenkins also supports cloud-based architecture so that you can deploy Jenkins in cloud-based platforms.

Environment setup - Jenkins

Now we setup Jenkins and other plugins of it.
Make sure to add Jenkins to docker group, so that Jenkins can use docker for build procedure.
$ sudo usermod –aG docker Jenkins, you can verify it with
$ sudo grep jenkins/etc/gshadow

Rundeck Configuration in Jenkins

Go to Manage Jenkins → configure system
Under Rundeck enter the following comfiguration:
Name: Rundeck Outgoing
Url: http://localhost:4440
Login: admin
Password: admin

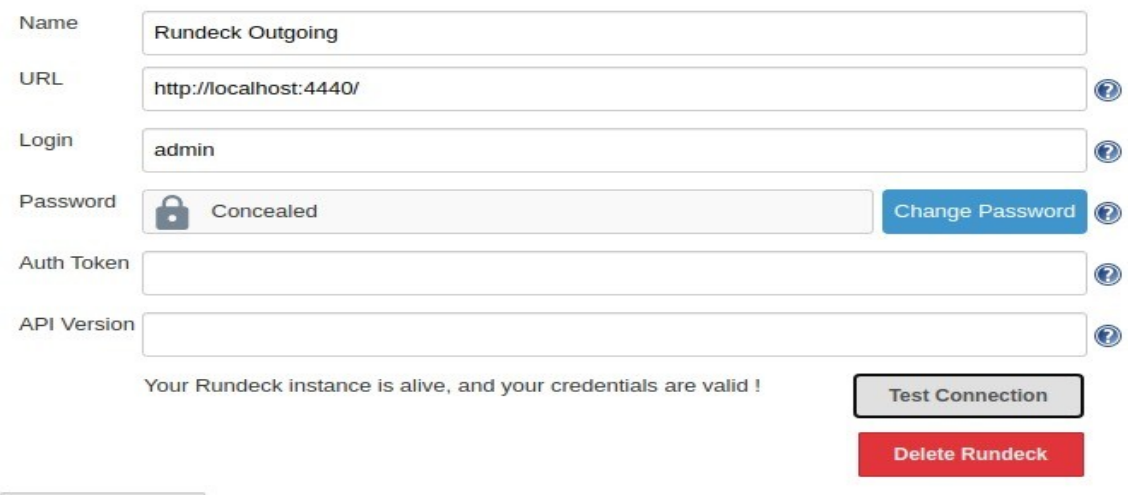

Fig 11: Rundeck configuration in jenkins

Docker Configuration

Go to Jenkins → credentials→System→Global Credentials

Under Global Credentials:

Username: m0hitbansal

Password: *******

ID: docker-hub-credentials

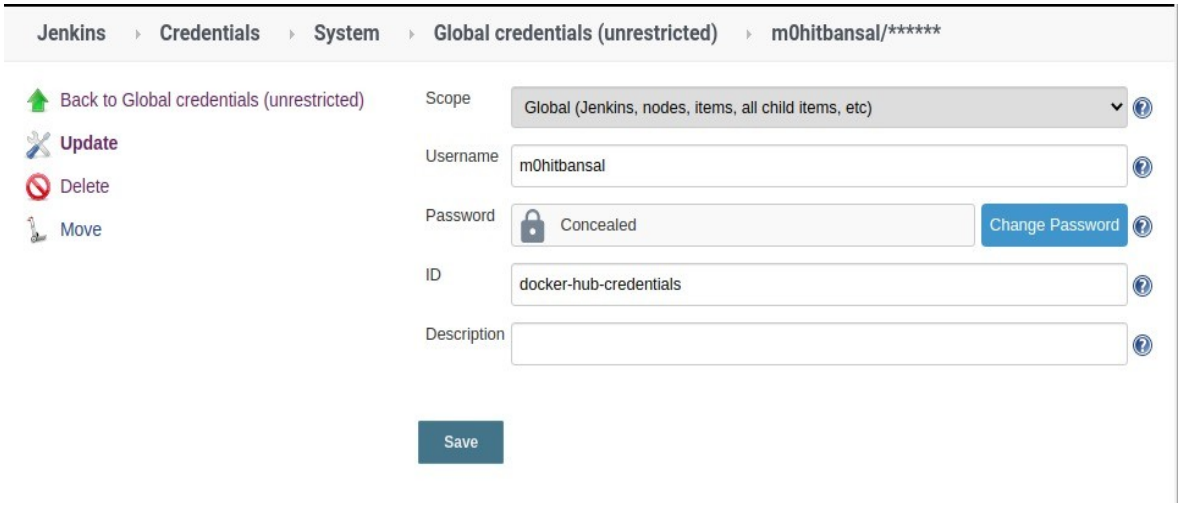

Fig 12: Docker configuration in jenkins

Github webhook Configuration in Jenkins

Go to Manage Jenkins → configure system

Under Github Server:

Name: webhook

API Url: http://api.github.com

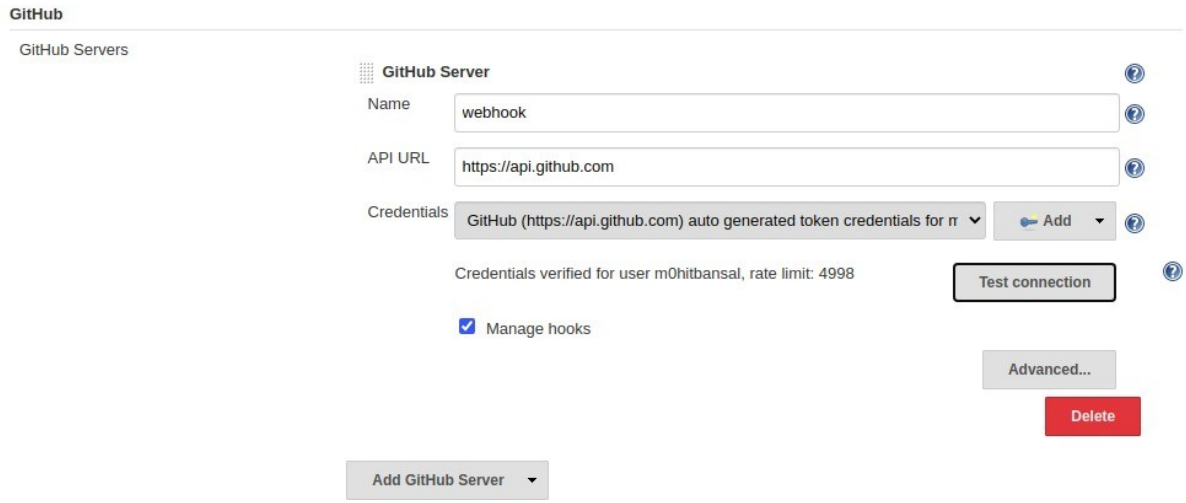

Fig 13: Github webhook Configuration in Jenkins

Setting up Jenkins Pipeline

Click on new item

Enter project Name: Outgoing-management-pipeline

Select pipeline project

Click Ok

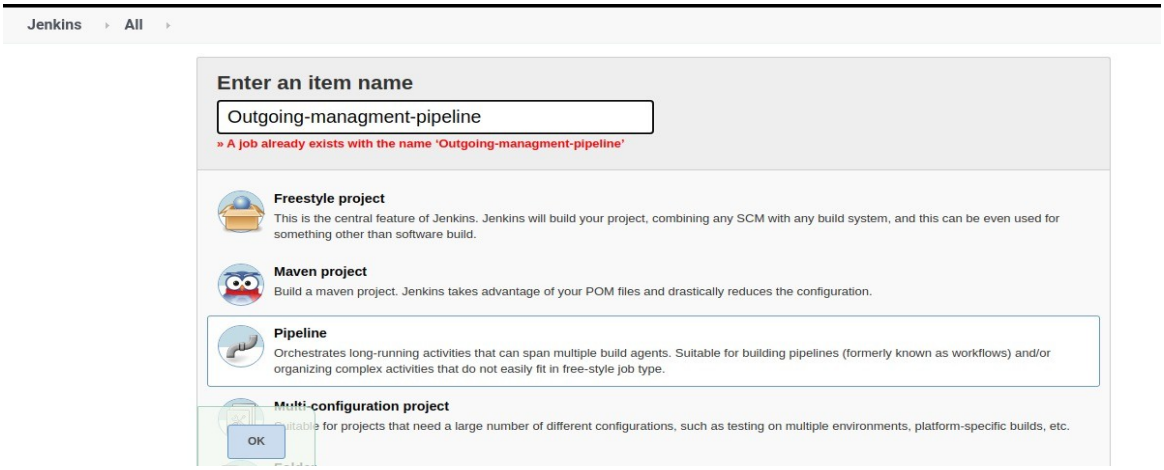

Fig 14: Create New Pipeline project

In project configuration we check github project to pull latest code from github as shown in the figure below.

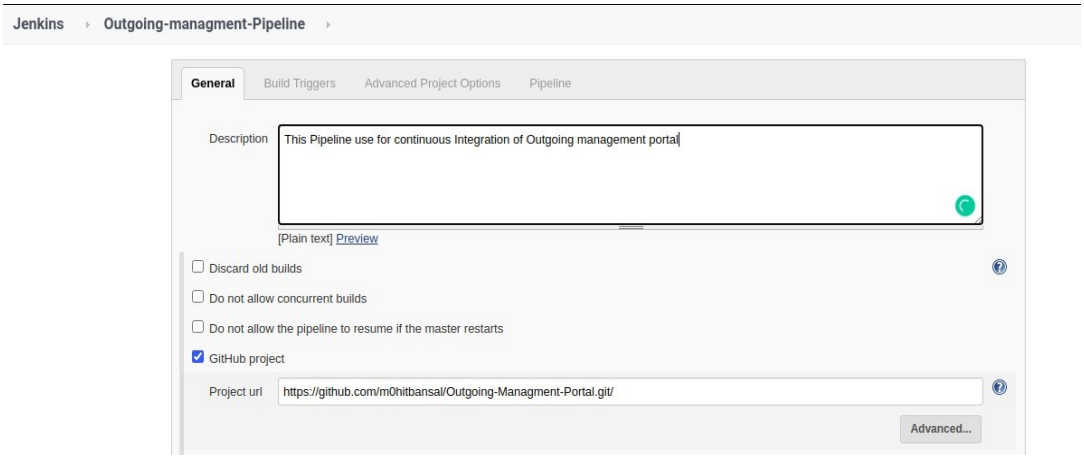

Fig 15: Add Github project

In Build Triggers
Check github hook trigger GITScm Polling

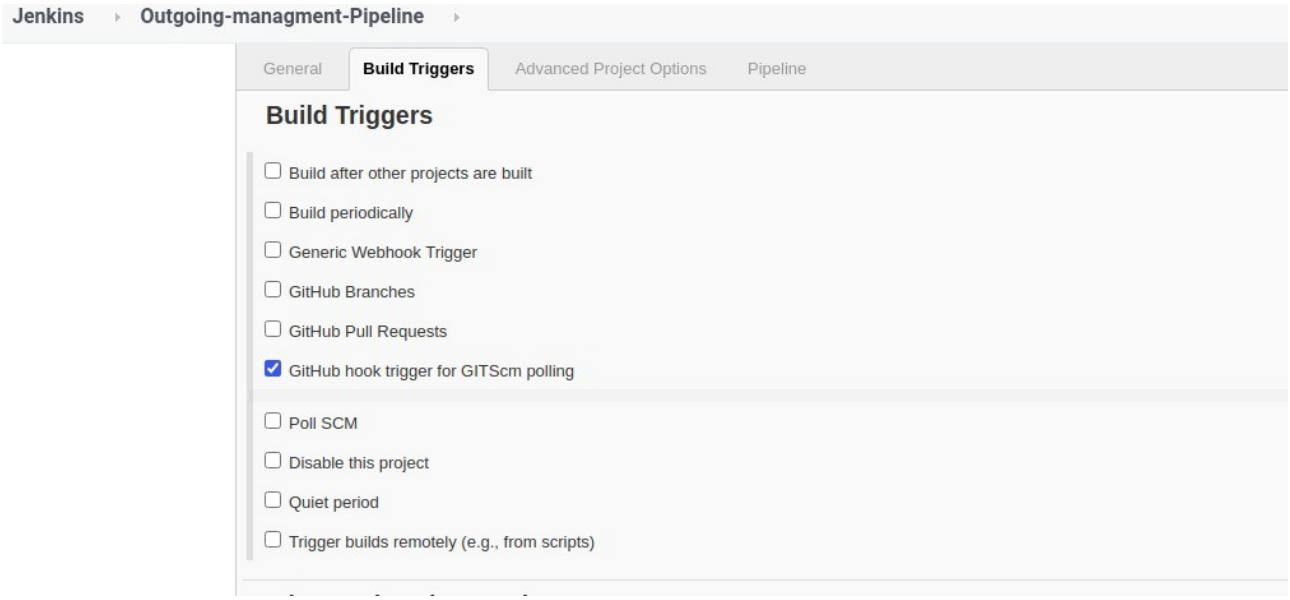

Fig 16: Set Build trigger

In Pipeline stage we write pipeline script as shown in the figure.

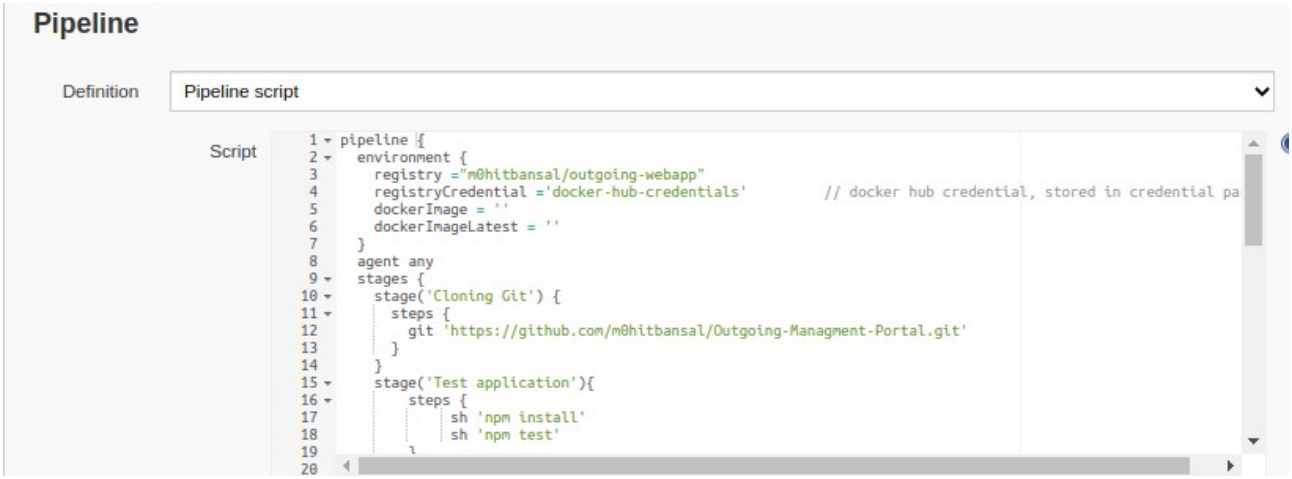

Fig 17: Add Pipeline script

```
pipeline {
  environment {
    registry ="m0hitbansal/outgoing-webapp"
    registryCredential ='docker-hub-credentials'         // docker hub crede
    dockerImage = ''
    dockerImageLatest = ''
  }
  agent any
  stages {
    stage('Cloning Git') {
      steps {
        git 'https://github.com/m0hitbansal/Outgoing-Managment-Portal.git'
      }
    }
    stage('Test application'){
        steps {
             sh 'npm install'
             sh 'npm test'
        }
    }
    stage('Building image') {
      steps{
        script {
          dockerImageLatest = docker.build registry + ":latest"
        }
      }
    }
```

Fig 18: Pipeline script-1

```
    stage('Deploy Image') {
      steps{
        script {
          docker.withRegistry( '', registryCredential ) {
            dockerImageLatest.push()
          }
        }
      }
    }
    stage('Remove Unused docker image') {
      steps{
        sh "docker rmi $registry:latest"
      }
    }
    stage('Execute Rundeck job') {
        steps {
          script {
            step([$class: "RundeckNotifier",
                  includeRundeckLogs: true,
                  jobId: "8eb64841-d14e-4c86-9a7e-a3ea08dfd0cd",
                  rundeckInstance: "Rundeck Outgoing",
                  shouldFailTheBuild: true,
                  shouldWaitForRundeckJob: true,
                  tailLog: true])
          }
        }
    }
  }
}
```

Fig 19: Pipeline script-2

Explanation of the pipeline script

1) We initially declare the environment variable which contains dockerhub credentials and dockerhub repository name.

2) In the first stage we add the github project url to clone the project.

3) In the second stage we build project and run the test file using mocha.

4) In the third stage the docker image is build. The name of the image can be generic or declared in environment variables.

5) In the fourth stage the image is deployed on DockerHub. Here, we use the *registryCredential* we configured in the previous slide and then push the image to the repository <dockerHub Username>/<repository name>.

6) In the fifth stage we remove the same unused image from the docker.

7) Jenkins can handle the builds for the continuous integration cycle of development and triggering of Rundeck is required to control distributed orchestration across the deployment. In the sixth stage we called rundeck job using job-id as shown in the pipeline script.

8) After writing the script click on save and apply button.

9) Click on build now.

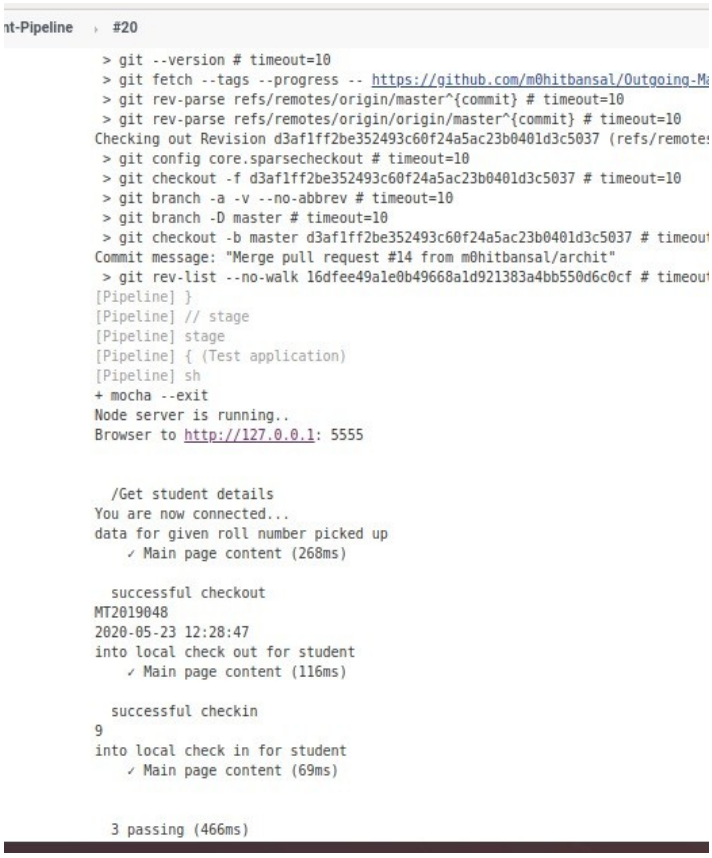

Fig 21: Console Output-1

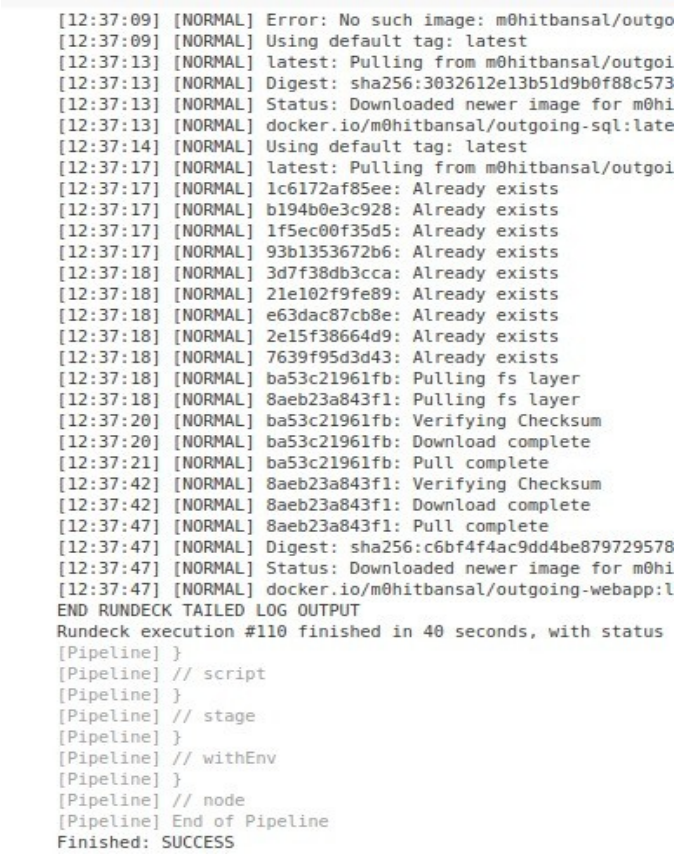

Fig 20: Console Output-2

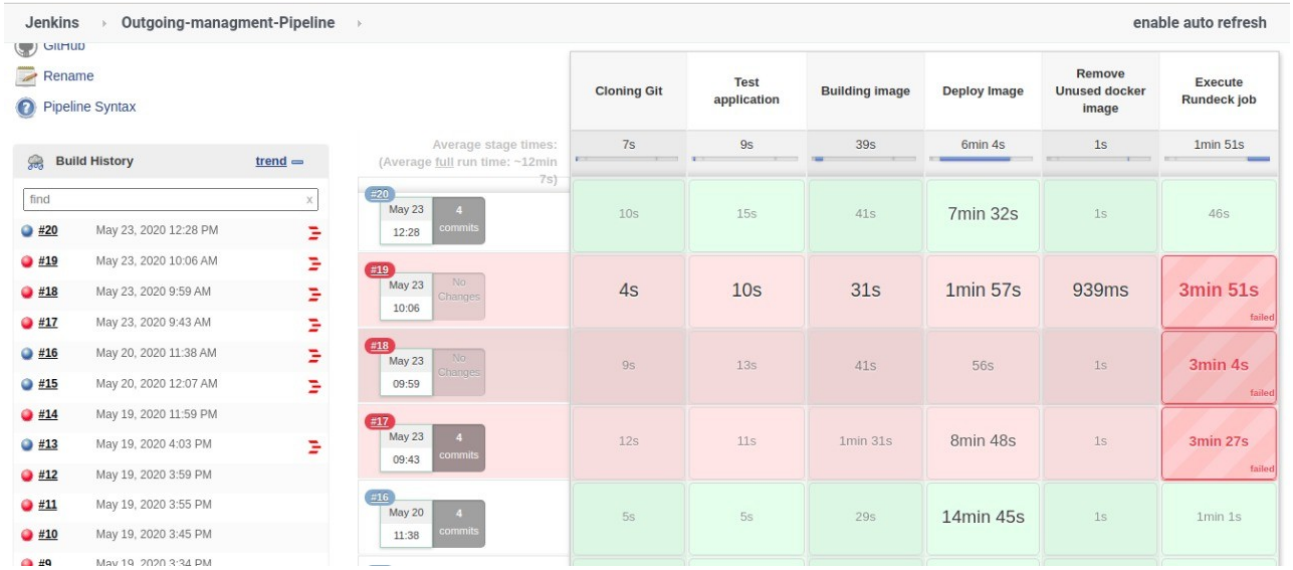

Fig 22: Pipeline Stage View

5.6 Continous Deployment

Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users. Continuous deployment eliminates the human safeguards against unproven code in live software.

Rundeck

Rundeck is an open source automation service with a web console, command line tools and a WebAPI. It lets you easily run automation tasks across a set of nodes.RunDeck is cross-platform open source software that helps you automate ad-hoc and routine procedures in data center or cloud environments. RunDeck allows you to run tasks on any number of nodes from a web-based or command-line interface. RunDeck also includes other features that make it easy to scale up your scripting efforts including: access control, workflow building, scheduling, logging, and integration with external sources for node and option data.

Environment setup for Rundeck

To run docker on Rundeck, we need to add Rundeck to the root group.

```
$ usermod -aG docker rundeck
```

Restart Rundeck i.e.

```
$ systemctl restart rundeckd
```

This way, we can see how rundeck belongs to the same group and now we can run *root* commands in rundeck without using *sudo.*

Creating new Project and job in Rundeck:

1) go to url and create a project a new project name it Outgoing-management-system and save it.

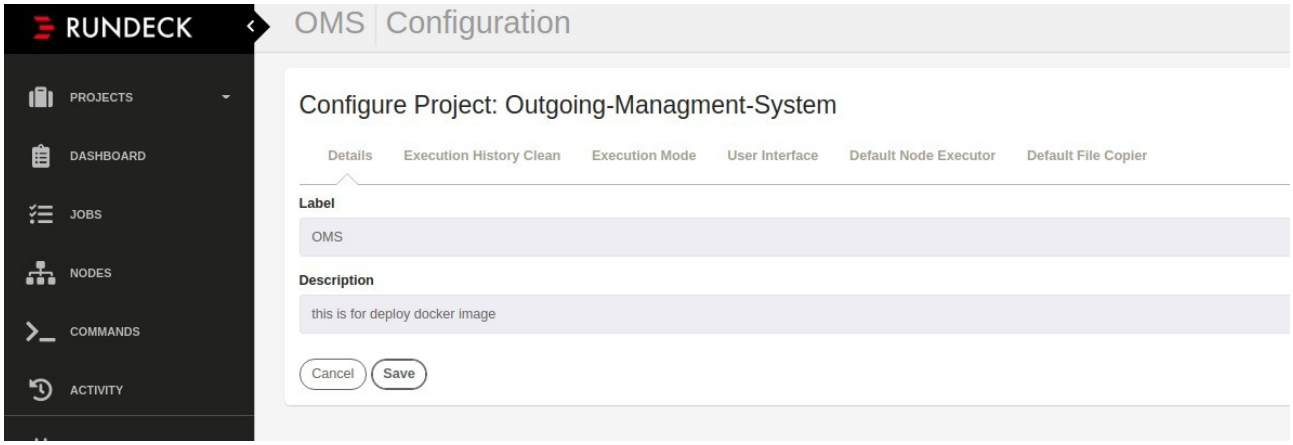

Fig 23: Create New Rundeck Project

2) Click on add new node source,select file and then select resource.xml as the format of the file. Add file path as */var/*lib/rundeck/outgoing.xml and check generate, include servernode and writable. Click on save button.

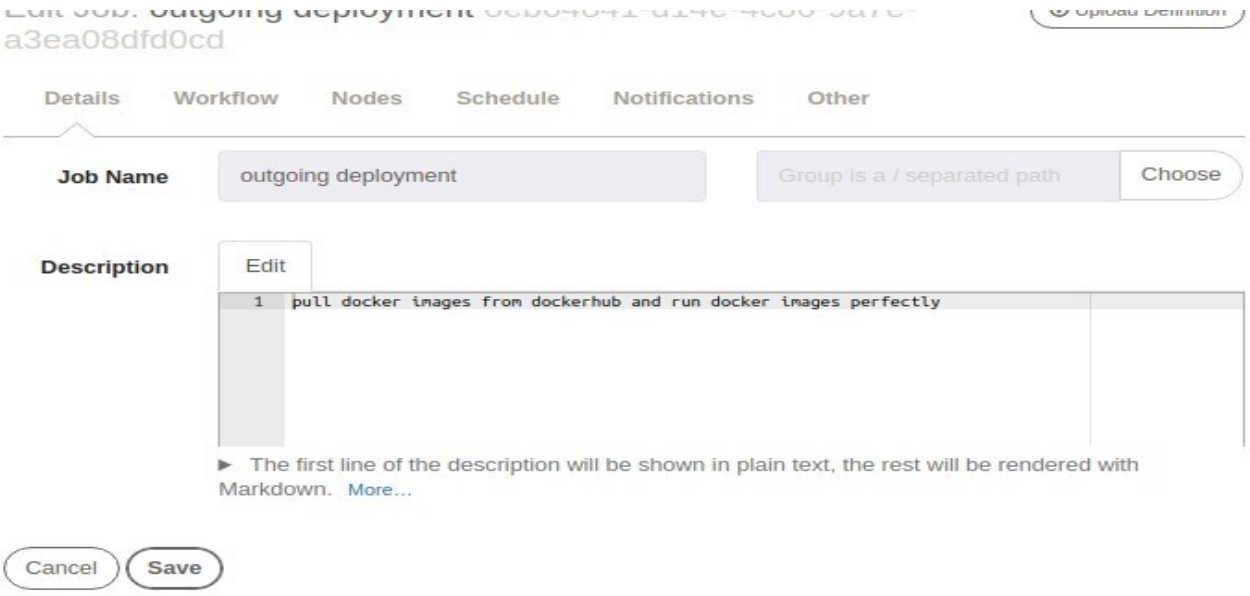

Fig 24: Create New Job

3) Create a new job, enter job name and description.
4) Go to workflow and add the script to execute on the registered node.

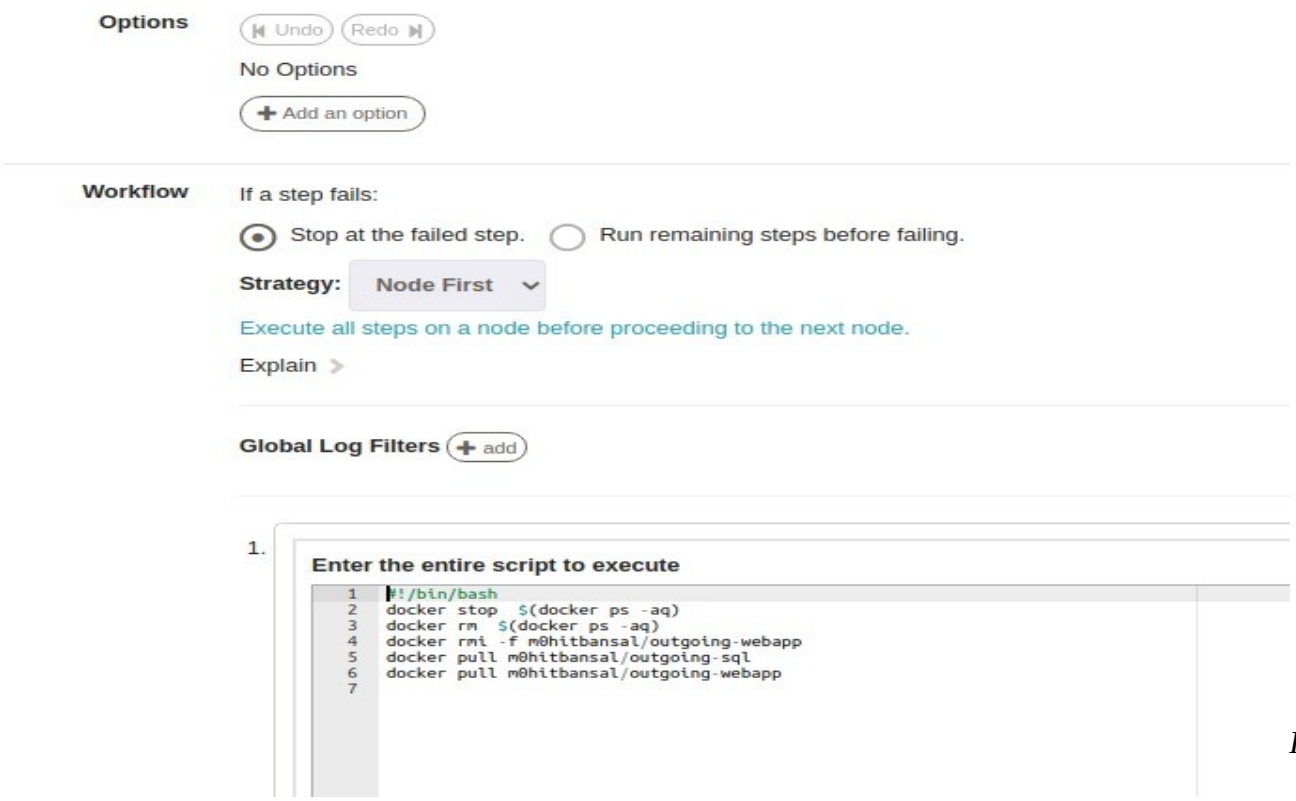

Fig 25: Job defination

4) Under Nodes select execute locally because all these commands will be execute on local system. We can specify the specific nodes on which we want commands to execute using dispatch to nodes in node option. Make note of UUID for future reference and save.

After that Run following command

$docker run -it --name websql -d m0hitbansal/outgoing-sql

$docker run --link websql:db -e DATABASE_HOST=db --name webapp -p 5555:5555 m0hitbansal/outgoing-webapp

5.7 Continous Monitoring

Elastic Search

Elastic search is a indexing querying over apache's lucene engine.
$ cd elasticsearch
$ ./bin/elasticsearch
you can check if elasticsearch is up and running at http://localhost:9200
open another terminal and proceed for kibana.
Kibana helps visualizing data using after querying using elastic search.

Kibana

$ cd kibana
$ ./bin/kibana
You can access kibana GUI at http://localhost:5601
Logstash helps normazling data from various data soruces such as apache, log events, sql and other data sources.
Also, logstash enhances data us varius filters such as geoip, grok, matcher and many more. Open another terminal and proceed for logstash.

Winston

Winston is designed to be a simple and universal logging library with support for multiple transports. A transport is essentially a storage device for your logs.

Advantages of Winston

1) Distributed platforms are fantastic for solving a lot of troubles, such as scaling, high availability, even maintainability of a big code base. But for all the great benefits they provide, they also come with some added baggage you need to take into account when working on one.

2) A scalable logging strategy is exactly what the name implies: you're able to log as much as you need. Just like you can (and should) scale your processing power or your bandwidth when your platform is experiencing a spike on traffic, your logging capabilities should have a similar elasticity.

```
const winston = require('winston');
const Elasticsearch = require('winston-elasticsearch');
var esTransportOpts = {
  level: 'info'
};
const logger = winston.createLogger({
  level: 'info',
  format: winston.format.combine(
            winston.format.timestamp({
      format: 'YYYY-MM-DD HH:mm:ss'
    }),
            winston.format.json()
    ),

  transports: [
    //
    // - Write all logs with level `error` and below to `error.log`
    // - Write all logs with level `info` and below to `combined.log`
    new winston.transports.File({ filename: 'combined.log' }),
    new Elasticsearch(esTransportOpts)
  ]
});

function logstore(caller,method,text){
    id = id +1;
    logger.info({"index":{"index":"Outgoing", "_id":id}, "level":'info', 'message':""});
    logger.info({"type":'api-call', "call_name":caller, "method":method, "text_entry":text});

}
module.exports = app; // for testing
```

Fig 26: Winston logs generate code

Fig 27: Kibana -1 Logs Dashboard

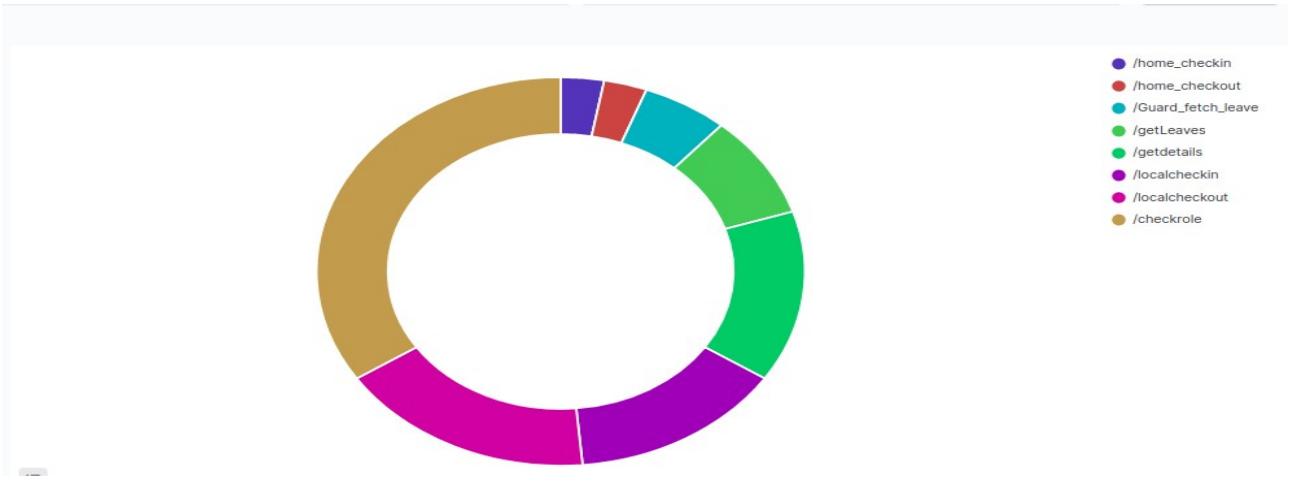

Fig 28: Count methods call

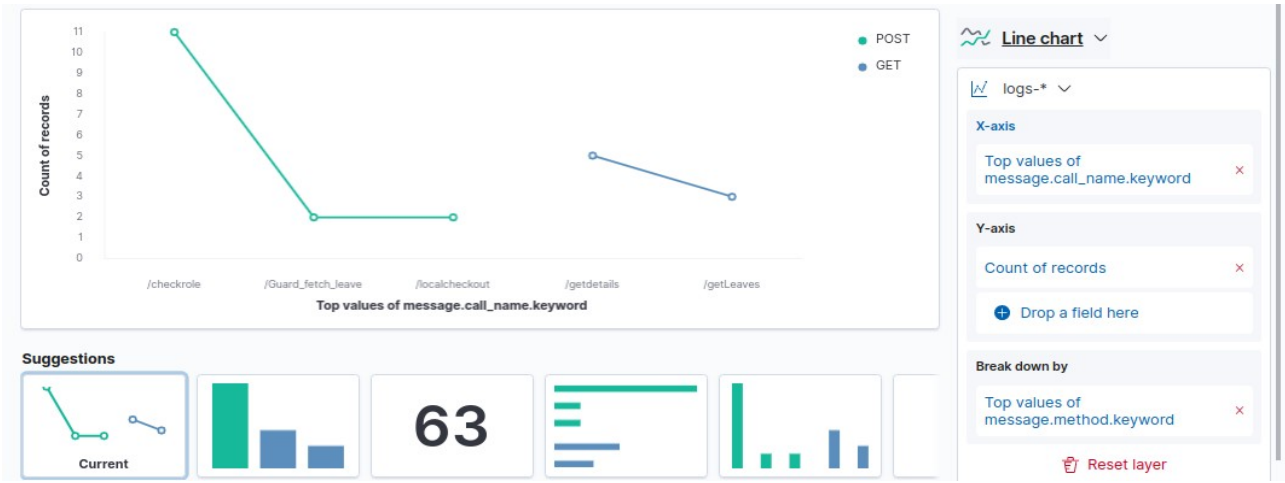

Fig 29: Kibana-3 Get/Post With timestamp

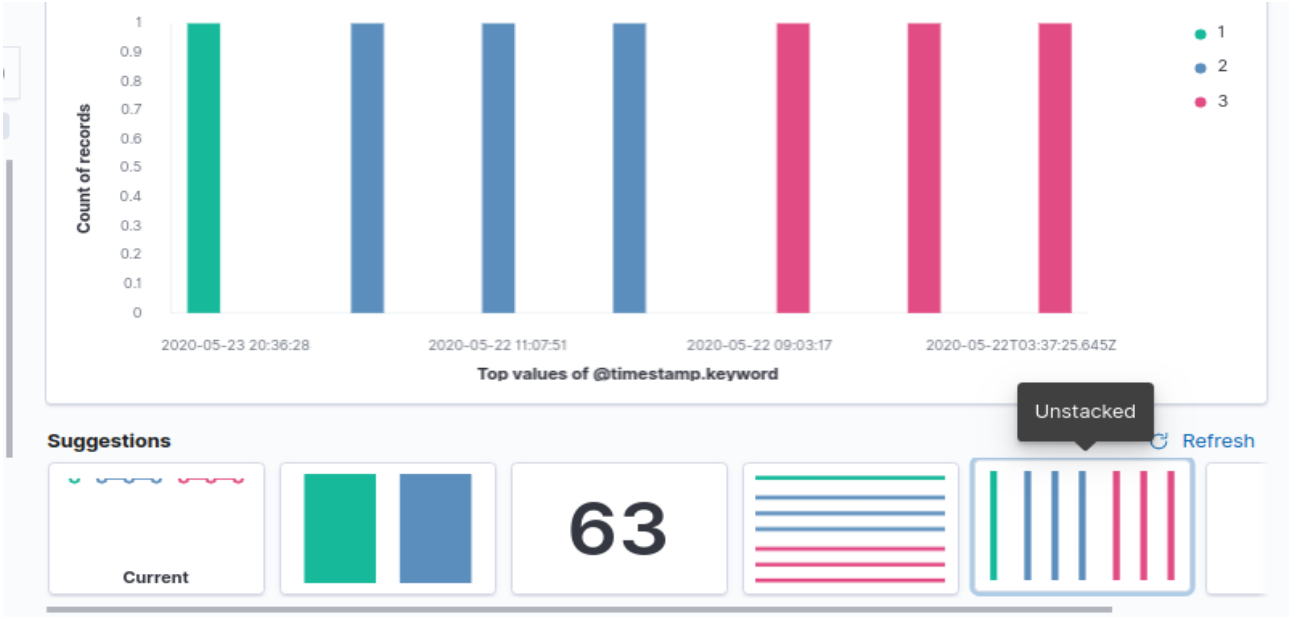

Fig 30: Kibana-4 Index id with time stamp

5.8 Webhooks

- The next problem in the pipeline was that each time, we manually needed to build the pipeline in Jenkins. We can automate this using webhooks which is a feature provided by GitHub.
- Thus, webhooks is a feature, where after each new commit/change in the repository, it calls the Jenkins for a built.
- Jenkins runs on localhost. To make it available, we need a tunneling application to make local ports publically available and we use **ngrok** for that purpose. Install it using
  $ sudo snap install ngrok
- Run ngrok on the same port on which your Jenkins is running i.e.
- $ngrok http 8080

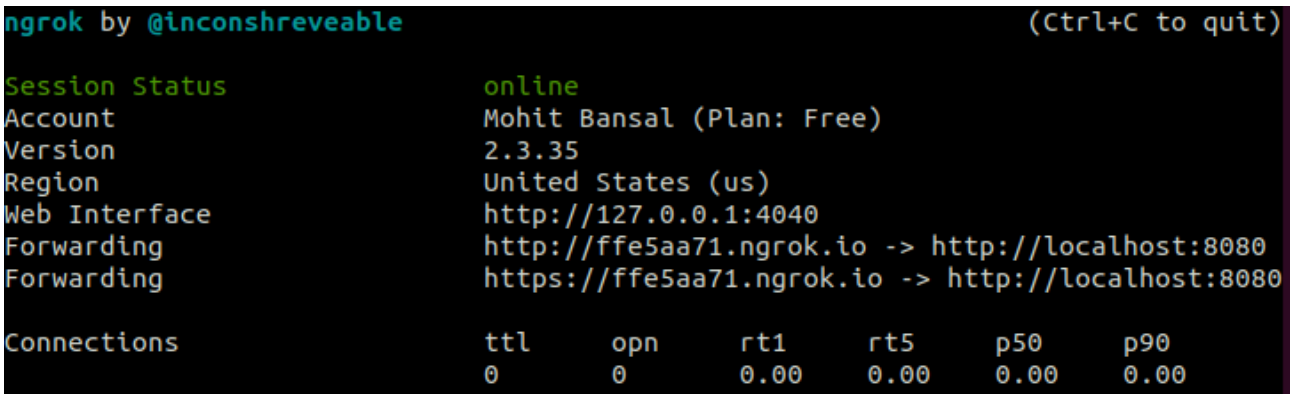

Fig 31: Ngrok start

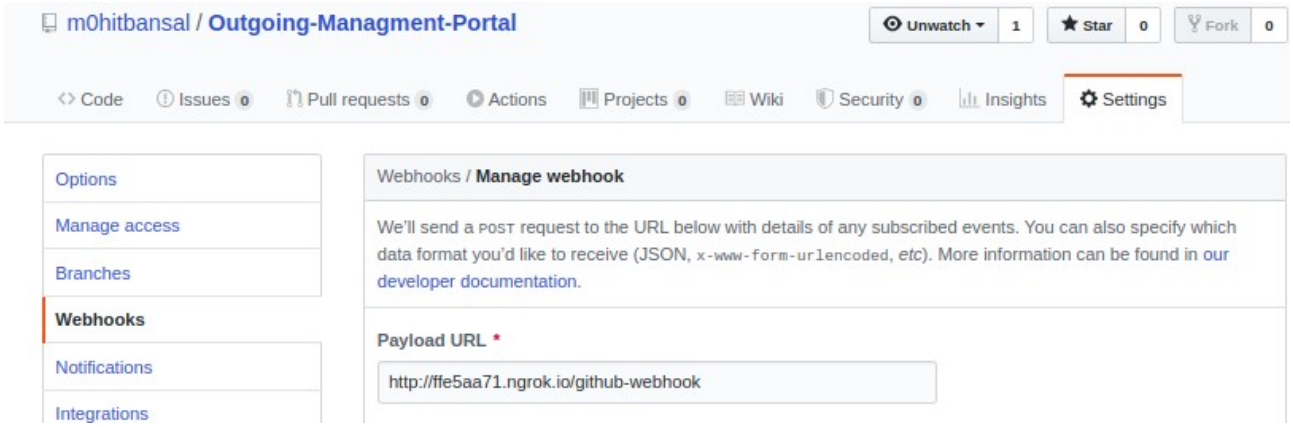

Fig 32: Github webhook setup

6.Experimental Setup

Installation

Install Nodejs and its libraries.

$ npm install nodejs

* Following Packages are also required:

express, express-session, mysql, body-parser, formidable, cors, nodemon, multer, chai, mocha, chai-http, node-cron, nodemailer, winston, winston-elasticsearch, nodemon

Git:

Install git and configure it using username and password.

$ sudo apt-get install git

$ git config --global user.name “m0hitbansal”

$ git config --global user.email “mohit.bansal@iiitb.org”

Install Docker:

Find resources at https://docs.docker.com/engine/install/ubuntu/

Docker Hub: Create a new account and create a new repository of the required name. It can be pushed/pulled using

$ docker pull/push <DockerHub Username>/<DockerHub Repository Name>

Install Jenkins:

Jenkins can be installed from the official documentation. In manage-jenkins, find the plugins below, download and install them. Plugins – maven, pipeline, rundeck, Nodejs, GitHub Pull Request Builder, Build Pipeline, Dashboard & Email extension.

$ java -jar jenkins.war

Install Rundeck :

Install Rundeck using the official documentation. The default port for web-interface is 4440 and default username and password is admin.

$ dpkg -i rundeck_2.10.8-1-GA_all.deb

Install ELK Stack:

ELK Stack can be downloaded from https://www.elastic.co/downloads/

7.Result and Discussion

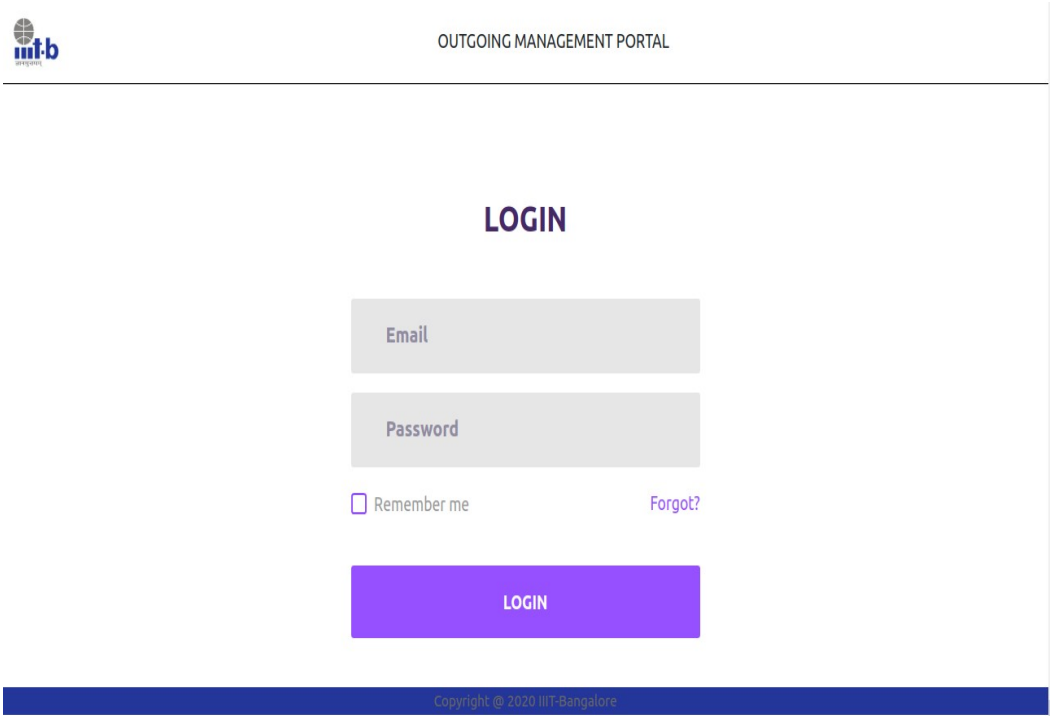

Fig 33: Login Page

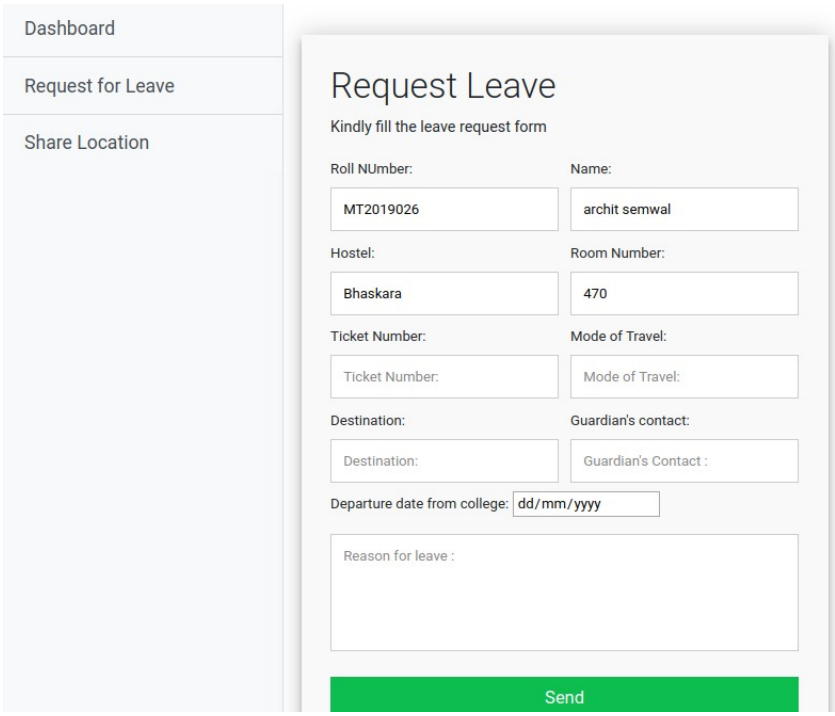

Fig 34: Student Leave Form Page

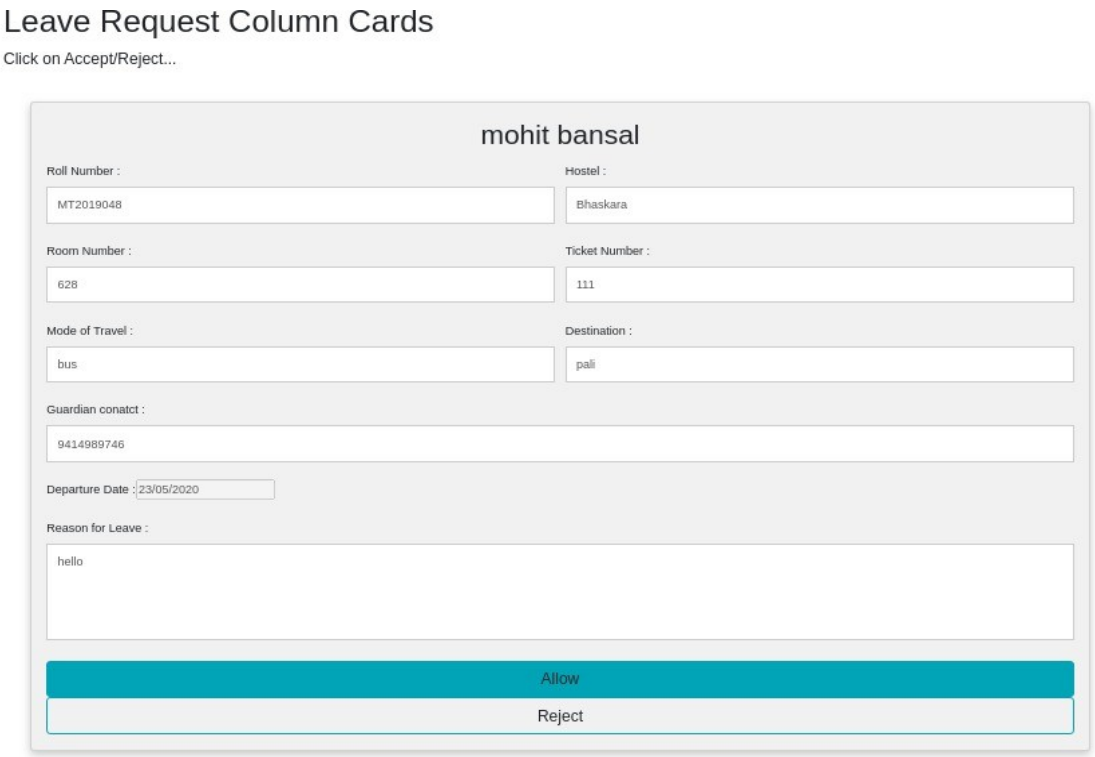

Fig 35: Warden's Approval Page

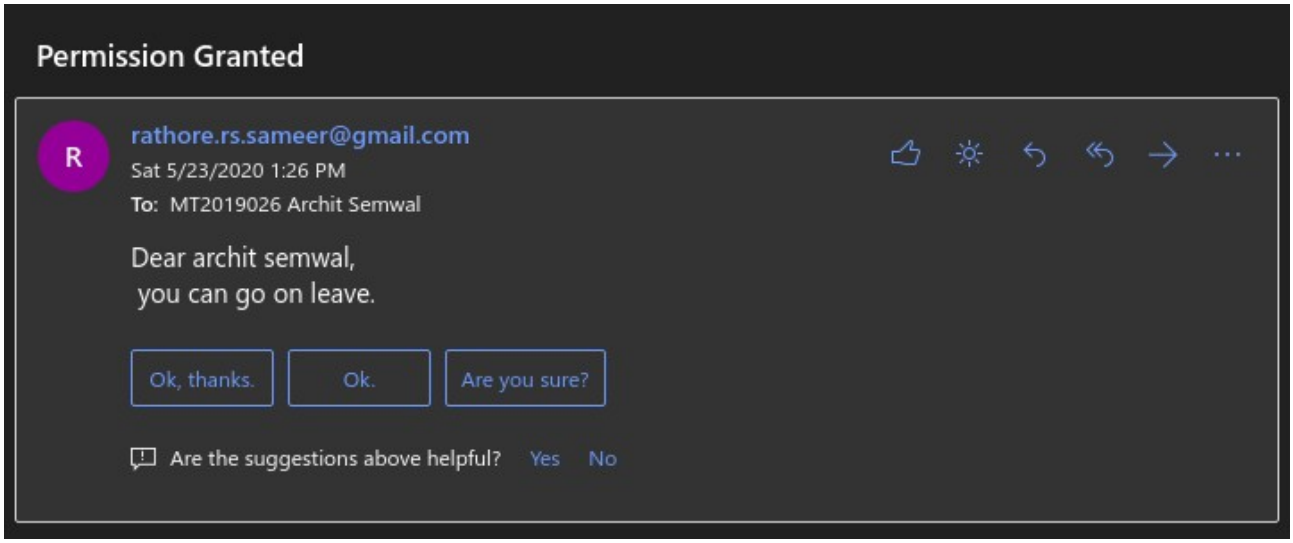

Fig 36: Autogenerated email for permission granted

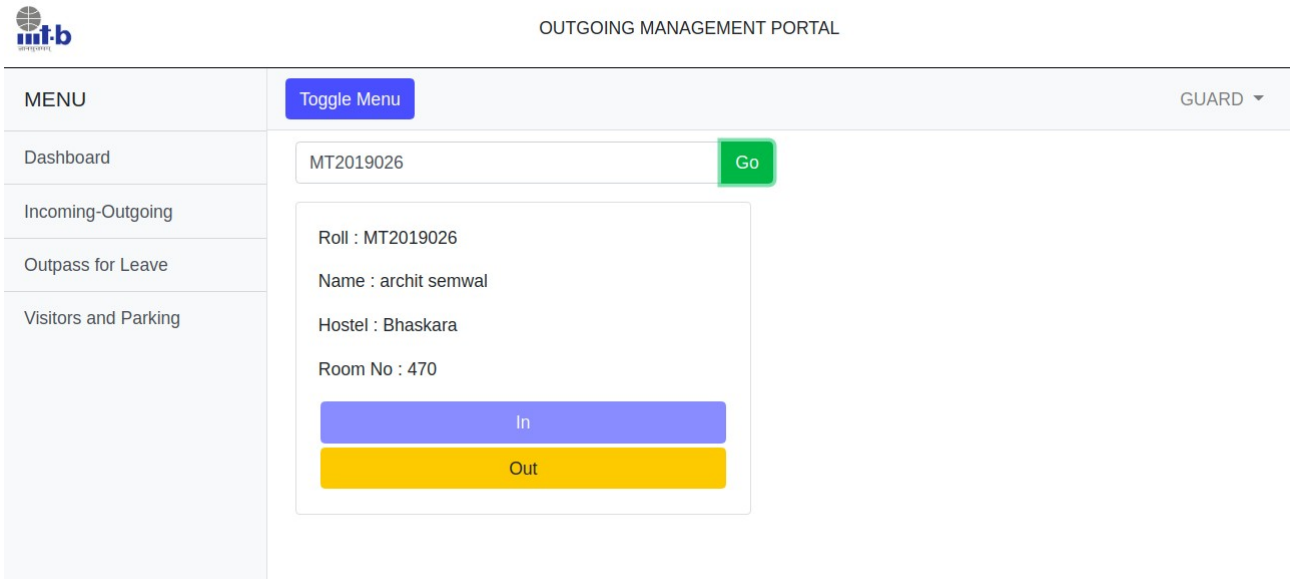

Fig 37: Local Outpass

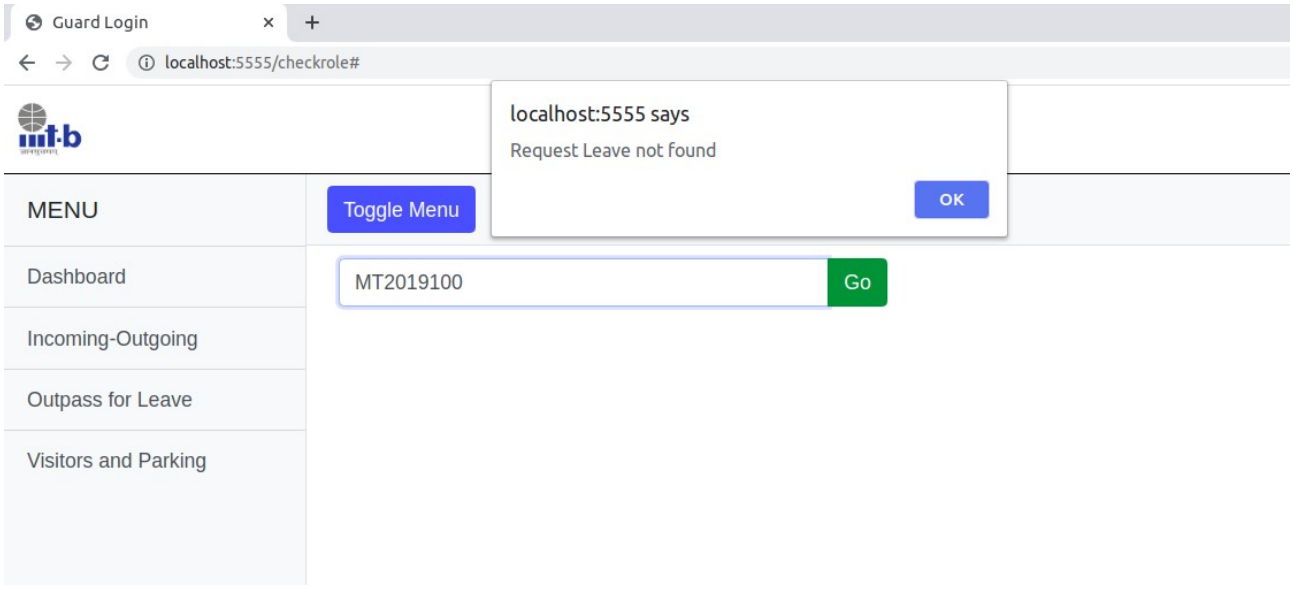

Fig 38: Leave not found alert

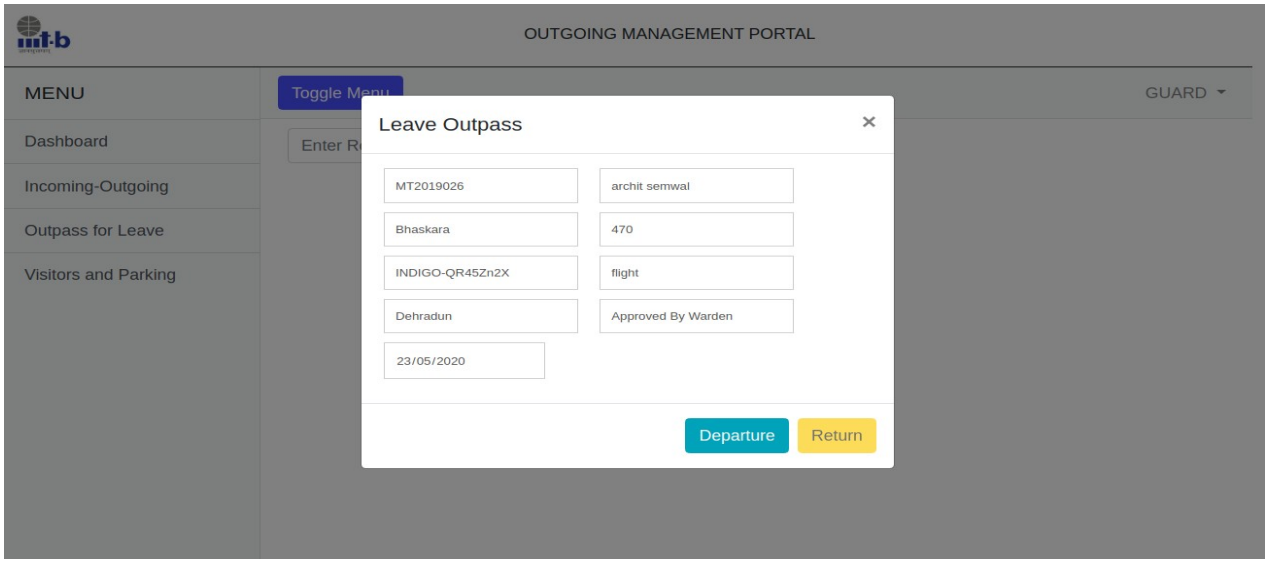

Fig 39: Outstation outpass Page

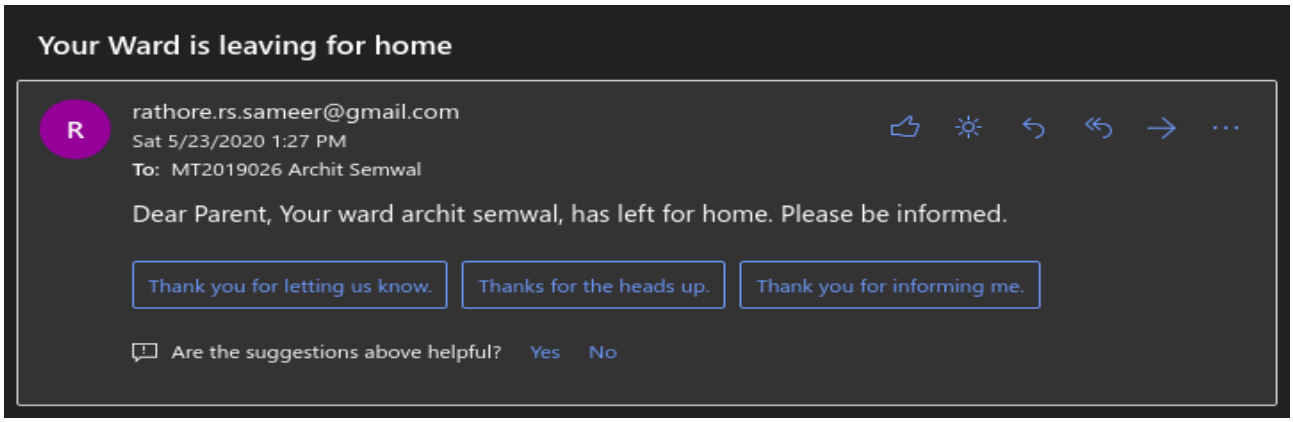

Fig 40: Guardian's Notification

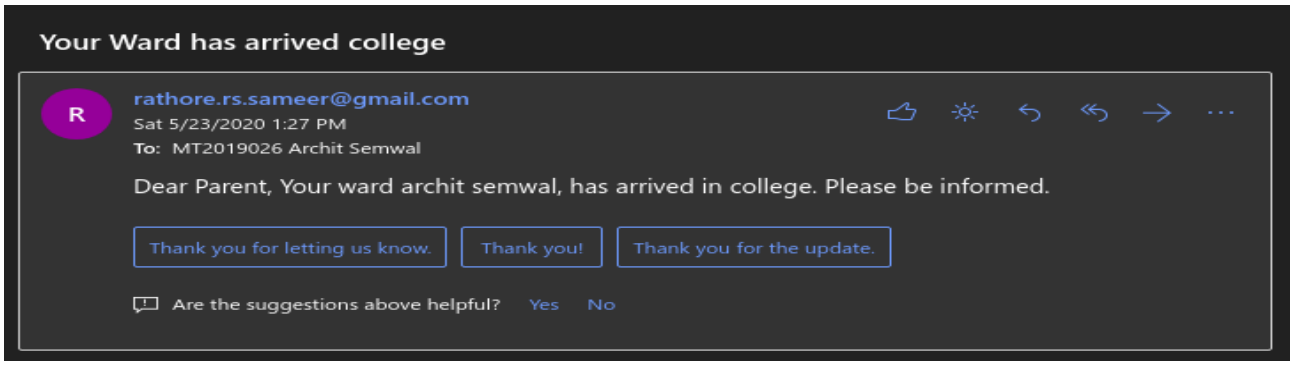

Fig 41: Guardian's Notification

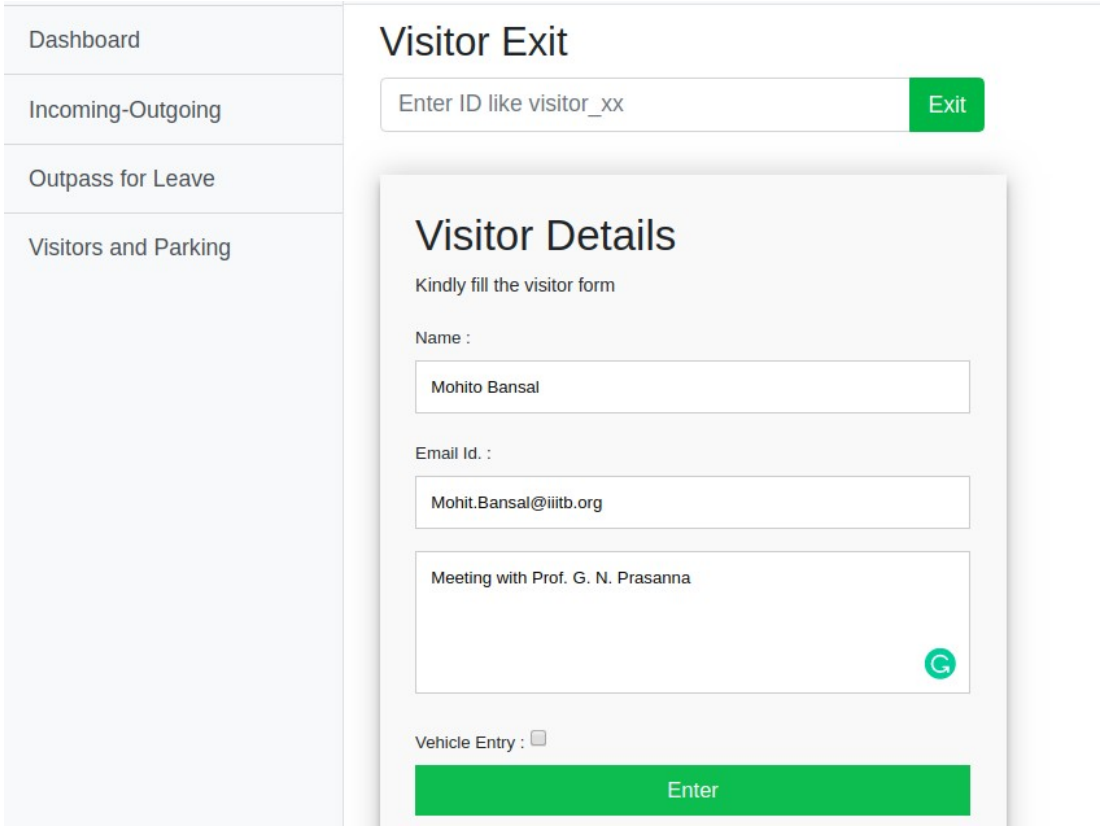

Fig 42: Visitor entry without vehicle

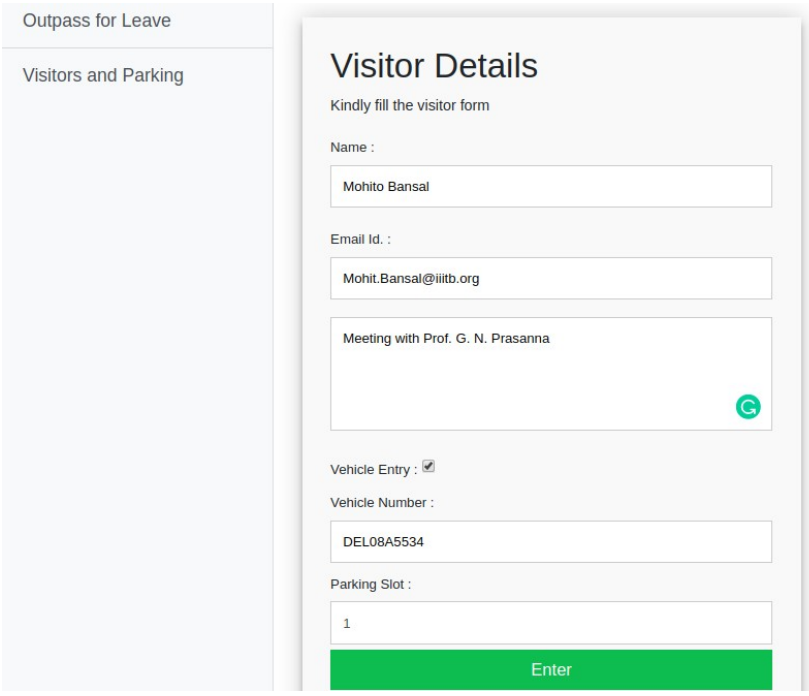

Fig 43: Visitor entry with vehicle

Fig 44: Allot unique id and parking slot to Visitor

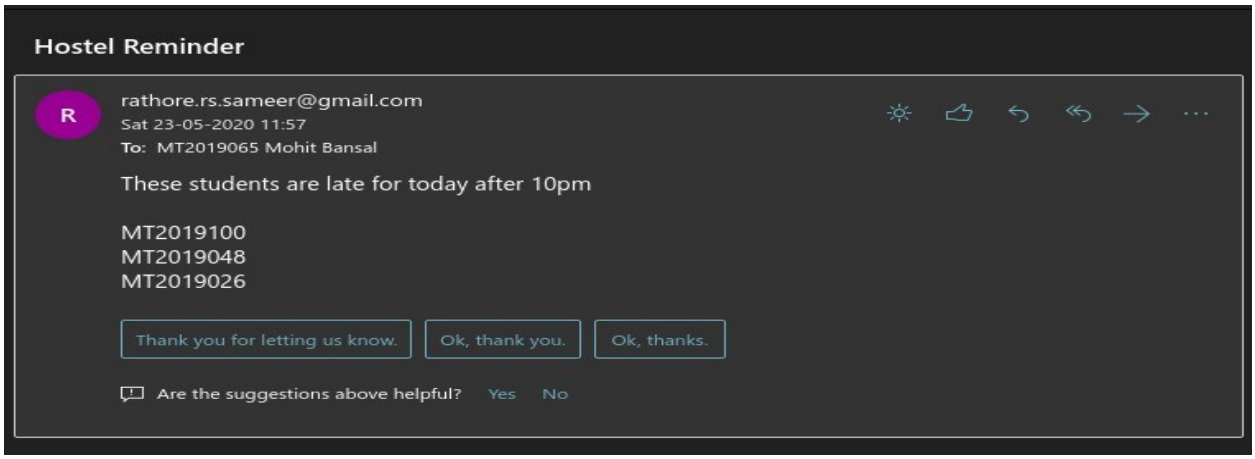

Fig 45: Autogenerated email to warden about late student's

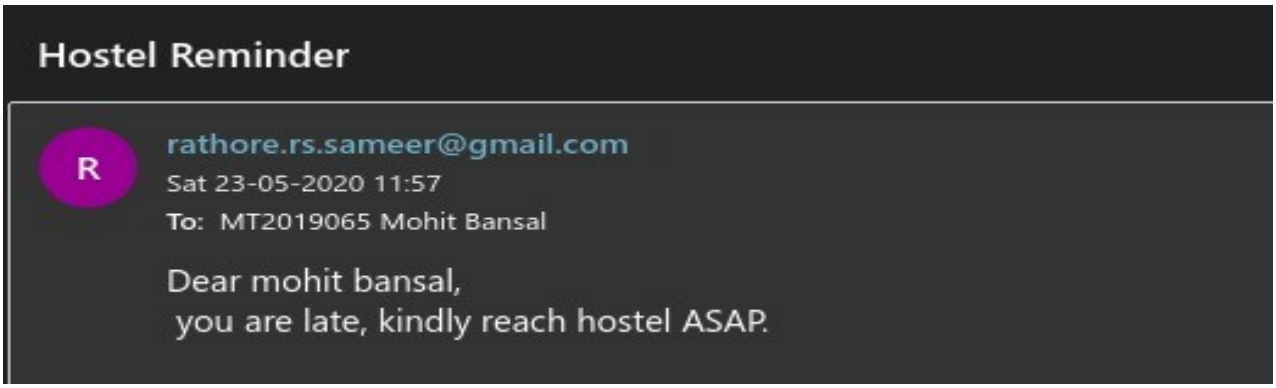

Fig 46: Student's late reminder email

8.Future Work

1. Warden can find list of students on leave for a specific date. warden when login can see students on leave on that day he logged in.

2. Student can see past records of his leave.

3. Student can share his locational co-ordinates when he is late on the web-portal.

4. We can append student id card scanner with the portal.

9.Conclusion

Deployment time: Avg 7.2 min.

We were successfully able to achieve the required results, and were able to integrate all the tools. However one issue is the time taken to deploy the app. The npm command to fetch the dependencies takes a lot of time due to the high number of dependencies being used.

Overall, it was great experience to learn and implement the DevOps tools as it will give us a headstart in the corporate world where these tools are used on a daily basis. DevOps tools are making the task quite easier manual interaction is getting reduced. Once all the things are set up, these tools will run automatically and building, integration, deployment, monitoring, testing will take place in a continuous manner.

10.Refrences

[1] https://github.com/Alakazam03

[2] https://dev.to/deleteman123/logging-at-scale-done-right-3m0e

[3] https://semaphoreci.com/community/tutorials/getting-started-with-node-js-and-mocha

[4]https://medium.com/@sece.cosmin/docker-logs-with-elastic-stack-elk-filebeat-50e2b20a27c6