

International Institute of Information Technology, Bangalore

Software Production Engineering Mini Project

Outgoing Management Portal

TA : Vaibhav Aggarwal

Guide : Prof. B. Thangaraju



Archit Semwal
MT2019026

Mohit Bansal
MT2019065

Shashank Agarwal
MT2019100

CONTENTS

	Page No.
Abstract	3
1. Introduction	4
2. Workflow Diagram	6
2.1 Code Flow	7
3. System Configuration	8
4. Software Development Life Cycle	9
4.1 Source Code Management	9
4.2 Build	11
4.3 Testing	12
4.4 Docker Artifact	14
4.5 Continous Integration	16
4.6 Continous Deployment	24
4.7 Continous Monitoring	27
4.8 Webhook	31
5. Experimental Setup	32
6. Results and Discussion	33
7. Future work	39
8. Conclusion	39
9. References	40

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 hostellers, 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.

1. 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 automatically 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 amalgamation of the development and operation teams in terms of practices, philosophies as well as tools with a target to deliver products at a faster pace using proper software development methods and infrastructure.

The DevOps model enables organizations to serve their customers more efficiently as well as compete in the market in a much more organized and systematic manner. We used DevOps because it provides a various set of tools which have capabilities to provide continuous delivery, continuous integration along with continuous monitoring.

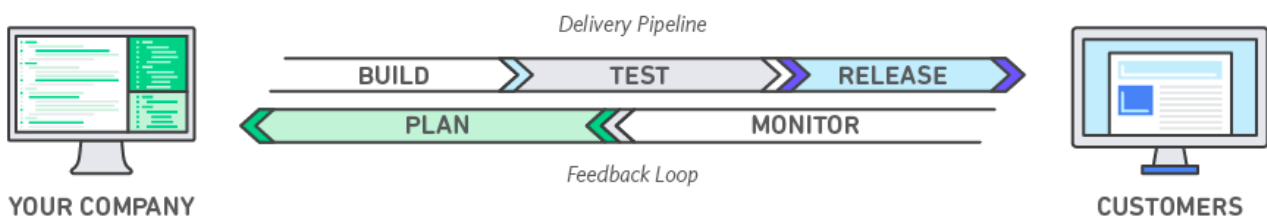


Fig 1: Basic devops model

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

Speed - Working at a higher velocity enables one to innovate faster thereby adjusting to the ever changing market demands and producing better business results.

Rapid Delivery - Faster the bug detection, quicker the improvement for rapid delivery of the product. In a similar fashion new features can be added and delivered at a faster pace.

Reliability - For the end user, the model guaranties quality application updates and an ever-growing infrastructure to create a positive user experience.

Scale - DevOps aims to provide scaling at the infrastructural and development levels. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

Improved Collaboration – Better collaborations between different teams working under a DevOps cultural model is made possible, concentrating on virtues like ownership and accountability of work. This reduces inefficiencies and saves time. There's a close collaboration between development and operation teams, sharing many responsibilities and bringing together their work.

2. Workflow Diagram

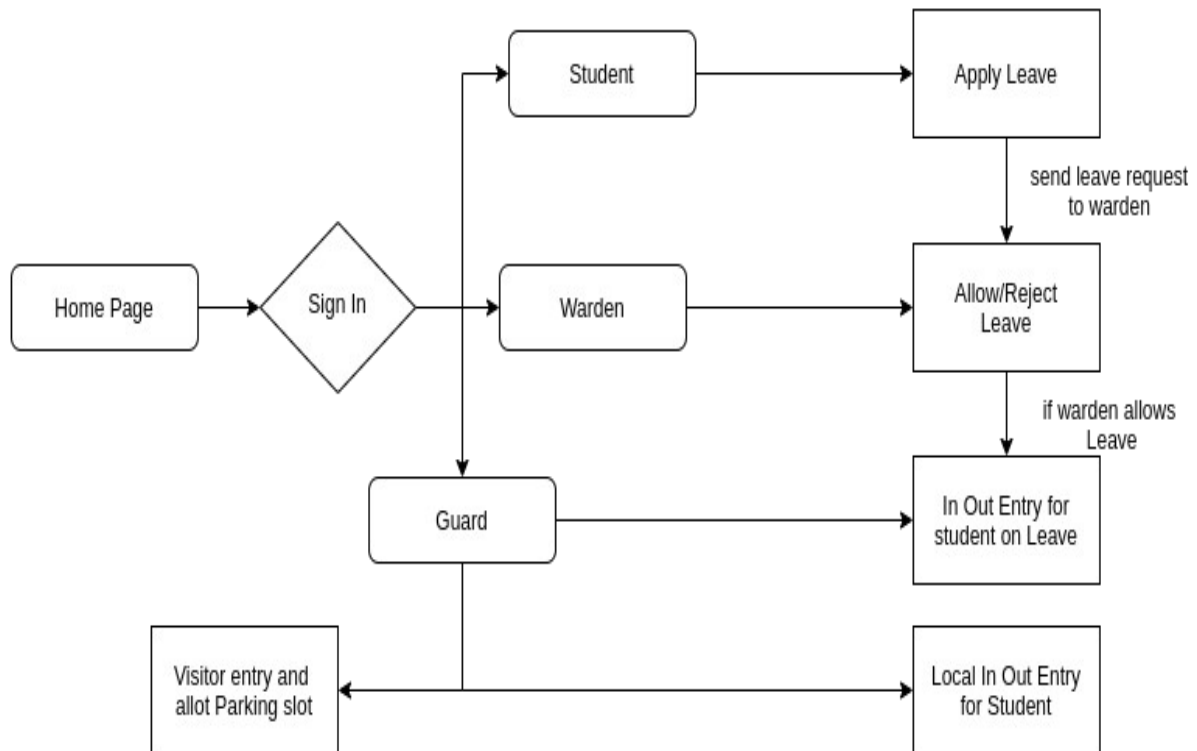


Fig 2: Workflow diagram of Portal.

The above diagram represents the flow of requests in the project, altogether there are three types of entities:

- **Student** - Can apply for leave. The leave request are immediately forwarded to the warden.
- **Warden** - Can either accept or reject student's leave application.
- **Guard** - Record student entries as in/out from the college hostel whenever a student goes for local outing. Monitor leave request status of students and allow passage accordingly. Apart from this the guard is also responsible for visitor's entry and allotment of parking space for vehicles.

2.1 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.

3. 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)

4. Software development Life Cycle

4.1 Source Code Management

GitHub is commonly used for Source Code Management. Being a no-cost open source distributed version control system, it helps to create projects scaling from larger to smaller ones easily.

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:

- Whenever multiple teams work on a common project it becomes quite difficult to keep a record of the updates and the updaters. In such scenario, github automates this job of handling all the changes made to the repositories.
- 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:

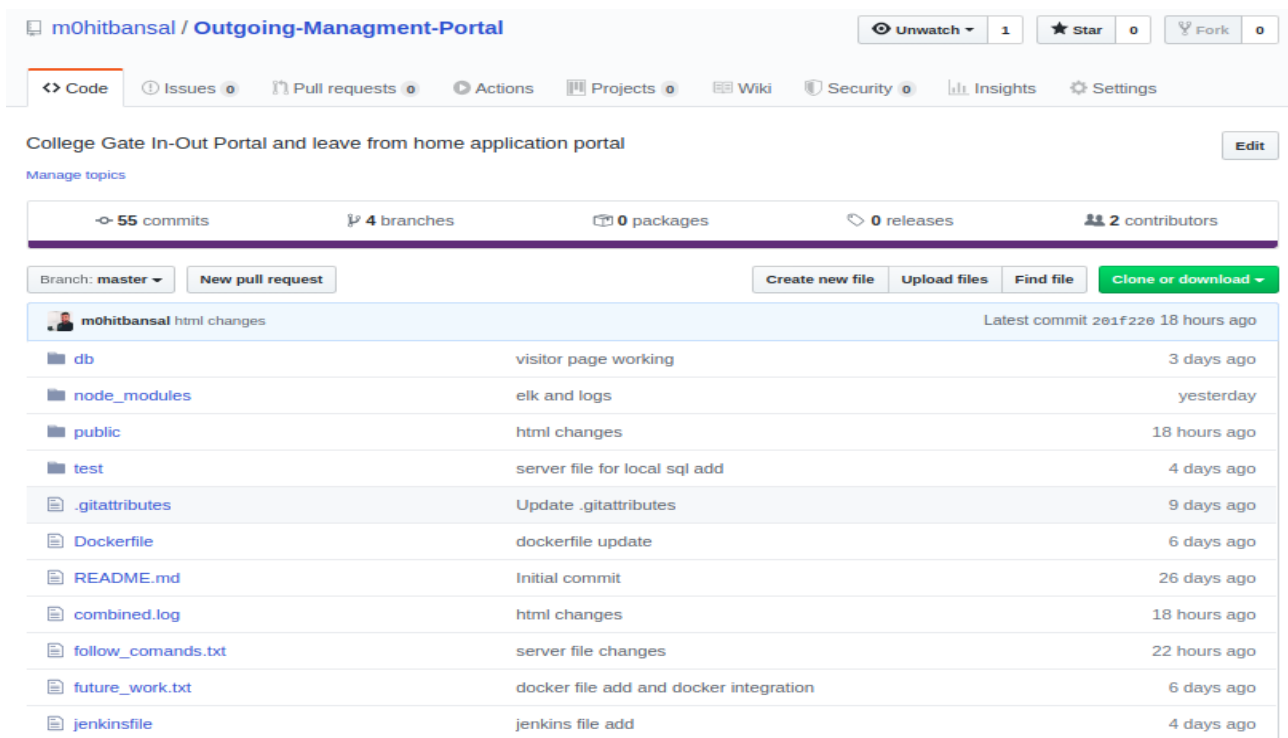
- i) `git init ./`
- ii) `git remote add origin https://github.com/m0hitbansal/Outgoing-Managment-Portal.git`
- iii) `git push -f origin <branch name>`

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

\$ git clone <https://github.com/m0hitbansal/Outgoing-Managment-Portal.git> : clone github repository in sustem

To push the code on to repository follow following commands:

- i) git add . : Add latest code in local git space
- ii) git commit -m “Commit message name”
- iii) git push origin <branch name> : push latest code on Github



The screenshot displays the GitHub repository page for `m0hitbansal / Outgoing-Managment-Portal`. The repository has 1 watch, 0 stars, and 0 forks. The main content area shows the repository name and a description: "College Gate In-Out Portal and leave from home application portal". Below this, there are statistics: 55 commits, 4 branches, 0 packages, 0 releases, and 2 contributors. A table lists the files and their commit history:

File	Commit Message	Time Ago
db	visitor page working	3 days ago
node_modules	elk and logs	yesterday
public	html changes	18 hours ago
test	server file for local sql add	4 days ago
.gitattributes	Update .gitattributes	9 days ago
Dockerfile	dockerfile update	6 days ago
README.md	Initial commit	26 days ago
combined.log	html changes	18 hours ago
follow_comands.txt	server file changes	22 hours ago
future_work.txt	docker file add and docker integration	6 days ago
jenkinsfile	jenkins file add	4 days ago

Fig 3: Github Repository of Outgoing Managment Portal

4.2 Build

NPM with Node.js

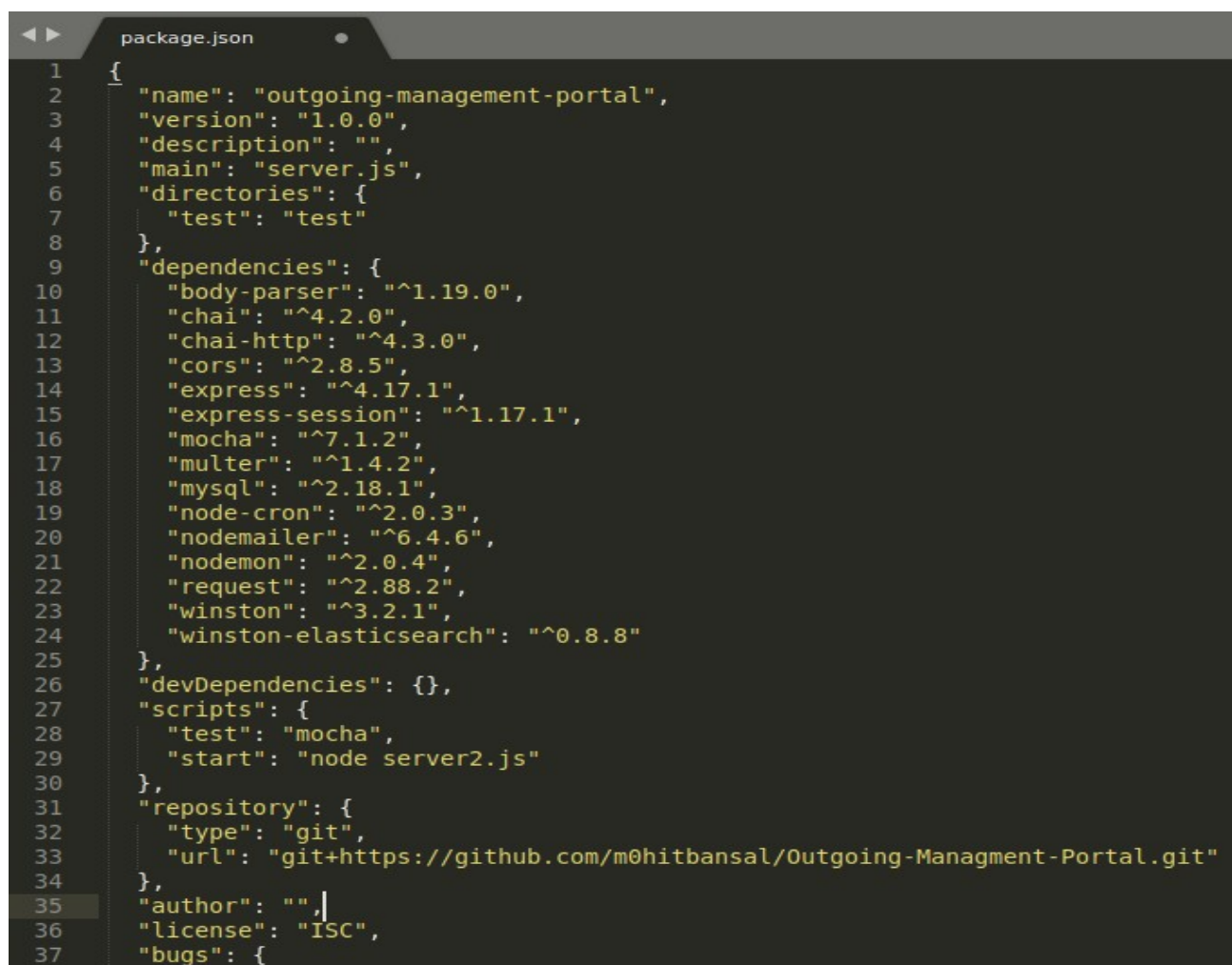
NPM is the default dependency management tool for node applications and is often used as a simple build tool as well. It works with all the node modules and with a single command, can fetch all the dependencies

\$ npm install : To install latest dependencies in system

and then build the project using

\$ npm build : Build project to install only required dependencies from package json

Again, it is extremely flexible and can easily be setup with 'package.json' file. It can be easily configured to work with express.



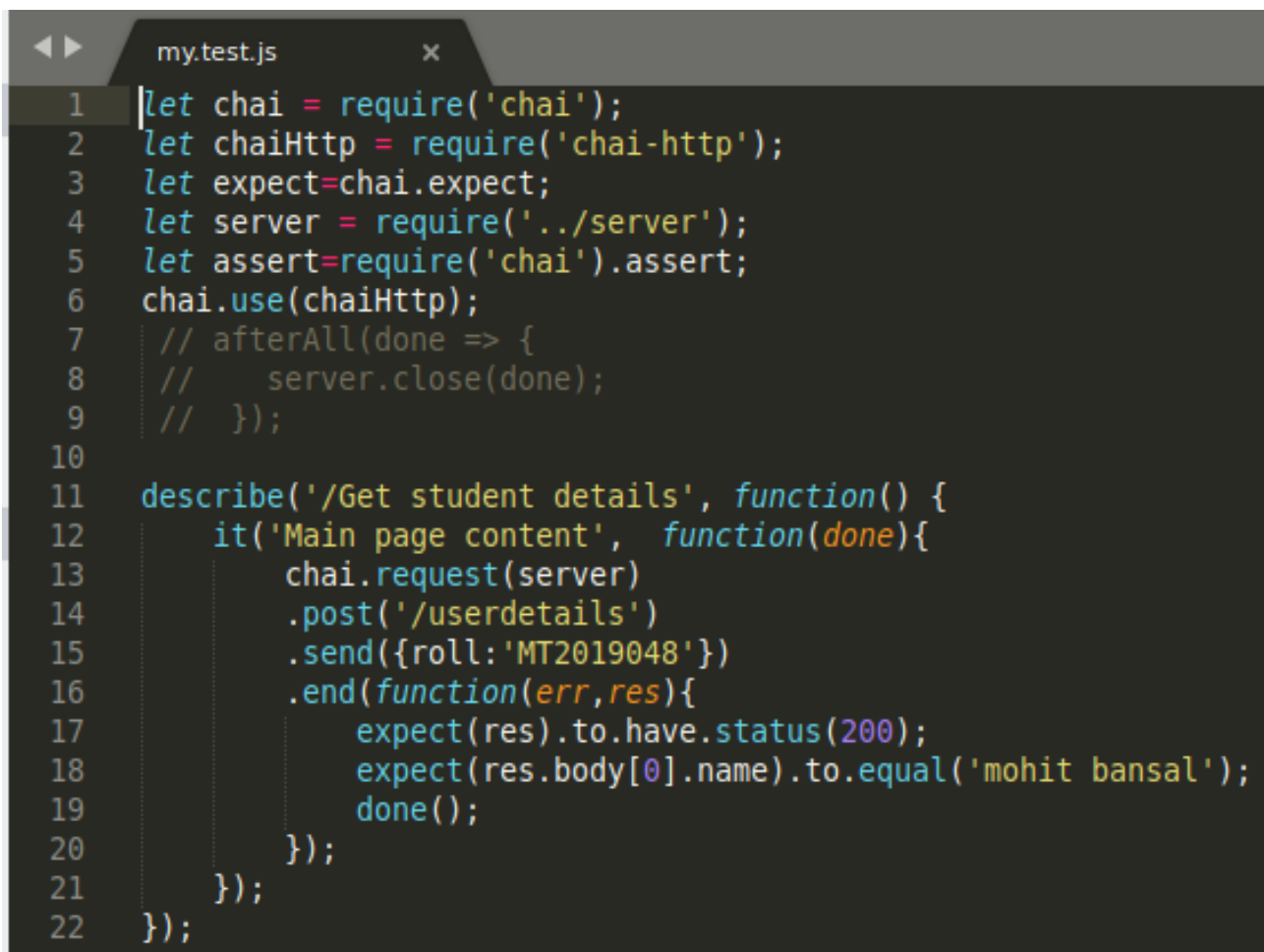
```
1 {
2   "name": "outgoing-management-portal",
3   "version": "1.0.0",
4   "description": "",
5   "main": "server.js",
6   "directories": {
7     "test": "test"
8   },
9   "dependencies": {
10    "body-parser": "^1.19.0",
11    "chai": "^4.2.0",
12    "chai-http": "^4.3.0",
13    "cors": "^2.8.5",
14    "express": "^4.17.1",
15    "express-session": "^1.17.1",
16    "mocha": "^7.1.2",
17    "multer": "^1.4.2",
18    "mysql": "^2.18.1",
19    "node-cron": "^2.0.3",
20    "nodemailer": "^6.4.6",
21    "nodemon": "^2.0.4",
22    "request": "^2.88.2",
23    "winston": "^3.2.1",
24    "winston-elasticsearch": "^0.8.8"
25  },
26  "devDependencies": {},
27  "scripts": {
28    "test": "mocha",
29    "start": "node server2.js"
30  },
31  "repository": {
32    "type": "git",
33    "url": "git+https://github.com/m0hitbansal/Outgoing-Managment-Portal.git"
34  },
35  "author": "",
36  "license": "ISC",
37  "bugs": {
```

Fig 4: package.json file

4.3 Testing

Mocha

Mocha is essentially a testing framework based on javascript for you to write unit tests to help ensure your functions are working as intended in your nodejs application. It can map uncaught exceptions to correct test cases. You can use asserts to check different things, like calculations being correct and so on, and by just specifying these tests in mocha specified file names you can run them seamlessly with just one command [5, 7].

A screenshot of a code editor window titled 'my.test.js'. The code is written in JavaScript and uses Mocha for testing. It starts with requiring 'chai' and 'chai-http', then 'expect' from 'chai', and 'server' from '../server'. It then uses 'chai.use(chaiHttp)'. There are comments for 'afterAll' and 'server.close'. The main test is a 'describe' block for '/Get student details' containing an 'it' block for 'Main page content'. Inside the 'it' block, a 'chai.request' is made to 'server' with a 'post' to '/userdetails' and a 'send' of an object with 'roll: 'MT2019048''. The 'end' block contains two 'expect' assertions: one for 'status(200)' and another for 'res.body[0].name' to equal 'mohit bansal'. The test ends with 'done()' and closing braces for the 'it' and 'describe' blocks.

```
1 let chai = require('chai');
2 let chaiHttp = require('chai-http');
3 let expect=chai.expect;
4 let server = require('../server');
5 let assert=require('chai').assert;
6 chai.use(chaiHttp);
7 // afterAll(done => {
8 //   server.close(done);
9 // });
10
11 describe('/Get student details', function() {
12   it('Main page content', function(done){
13     chai.request(server)
14       .post('/userdetails')
15       .send({roll:'MT2019048'})
16       .end(function(err,res){
17         expect(res).to.have.status(200);
18         expect(res.body[0].name).to.equal('mohit bansal');
19         done();
20       });
21   });
22 });
```

Fig 5: Node js test cases in my.test.js file

To run this test file we use command

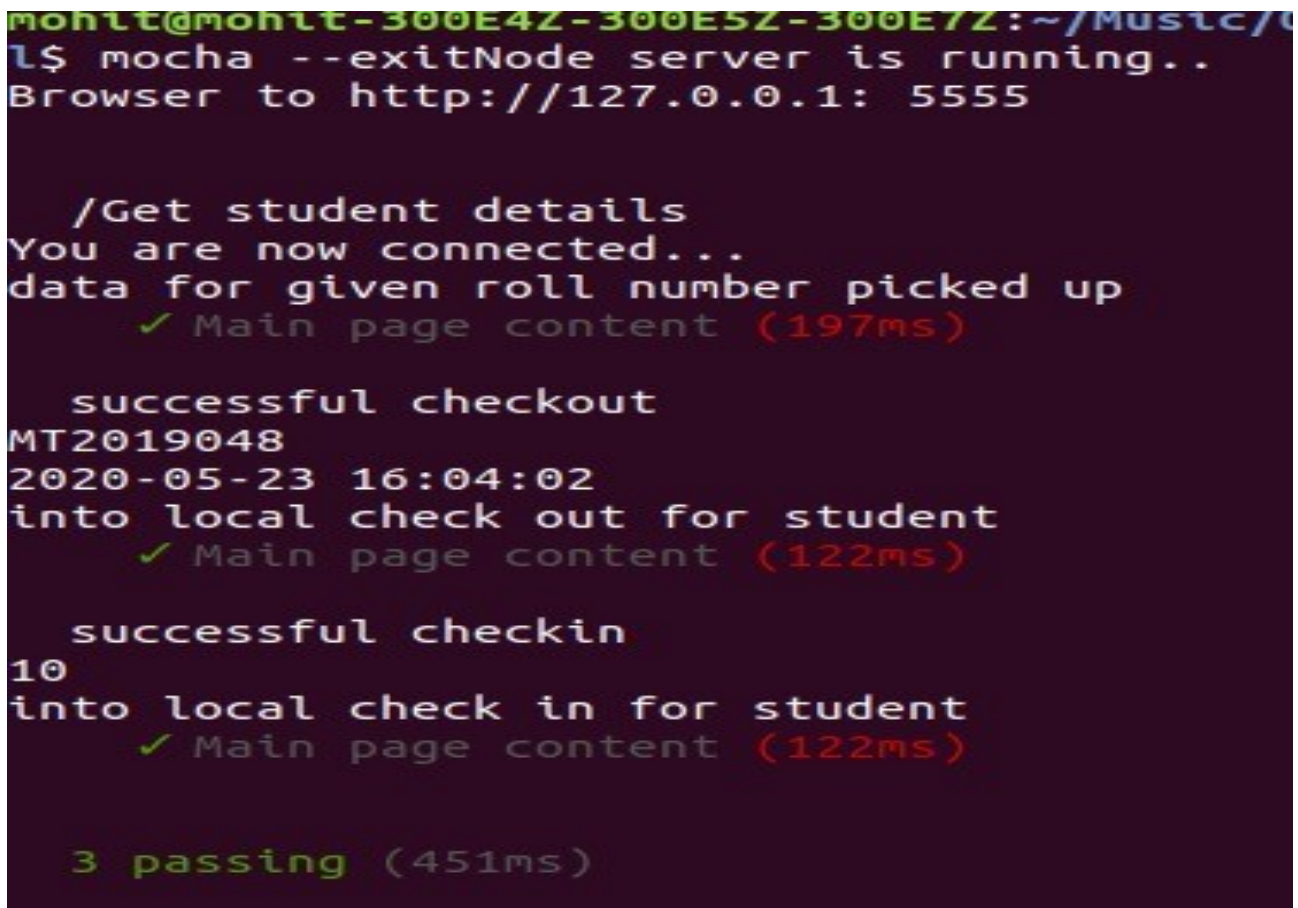
`$mocha -exit`

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 [6].

A terminal window with a dark purple background and light green text. The prompt is 'moht@moht-300E42-300E52-300E72:~/Music/'. The command 'mocha --exitNode server is running..' is entered. The output shows a browser connection to 'http://127.0.0.1: 5555'. The test suite is '/Get student details'. The first test 'You are now connected...' passes with 'data for given roll number picked up' and 'Main page content (197ms)'. The second test 'successful checkout' passes with 'MT2019048', '2020-05-23 16:04:02', and 'into local check out for student', with 'Main page content (122ms)'. The third test 'successful checkin' passes with '10' and 'into local check in for student', with 'Main page content (122ms)'. The final output is '3 passing (451ms)' in green.

```
moht@moht-300E42-300E52-300E72:~/Music/
$ mocha --exitNode server is running..
Browser to http://127.0.0.1: 5555

  /Get student details
  You are now connected...
  data for given roll number picked up
    ✓ Main page content (197ms)

  successful checkout
  MT2019048
  2020-05-23 16:04:02
  into local check out for student
    ✓ Main page content (122ms)

  successful checkin
  10
  into local check in for student
    ✓ Main page content (122ms)

  3 passing (451ms)
```

Fig 6: Test file execution using mocha command

4.4 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 [10].

A Dockerfile is a text file written in an easy-to-understand syntax that includes the instructions to build a Docker *image*. [15]

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

<https://github.com/m0hitbansal/Outgoing-Managment-Portal/blob/master/db/Dockerfile>

<https://github.com/m0hitbansal/Outgoing-Managment-Portal/blob/master/Dockerfile>

```
1 FROM node:latest
2
3 WORKDIR /Outgoing
4
5 ADD . /Outgoing
6
7 RUN npm install
8 EXPOSE 5555
9 ENTRYPOINT ["node","server2.js"]
10
```

Fig 8: Dockerfile for webapp

```
FROM mysql:5.7
ADD wsl.sql ./
ENV MYSQL_ROOT_PASSWORD root
ENV MYSQL_DATABASE Outgoing
COPY ./wsl.sql /docker-entrypoint-initdb.d/
EXPOSE 3306
```

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 . : Build web-app docker image
```

```
$ docker build -t m0hitbansal/outgoing-sql . : Build web-database docker image
```

```
$ docker login : login docker in system
```

```
$ docker push m0hitbansal/outgoing-sql : Push docker image on docker hub
$ docker push m0hitbansal/outgoing-webapp : Push docker image on docker hub
$ docker run -it --name webserv -d m0hitbansal/outgoing-sql : Run outgoing-sql image on system
$ docker run --link webserv:db -e DATABASE_HOST=db --name webapp -p 5555:5555 m0hitbansal/outgoing-webapp : Run Outgoing-webapp image and link with outgoing-sql image
```

Reference Images and links below:

<https://hub.docker.com/repository/docker/m0hitbansal/outgoing-webapp>

<https://hub.docker.com/repository/docker/m0hitbansal/outgoing-sql>

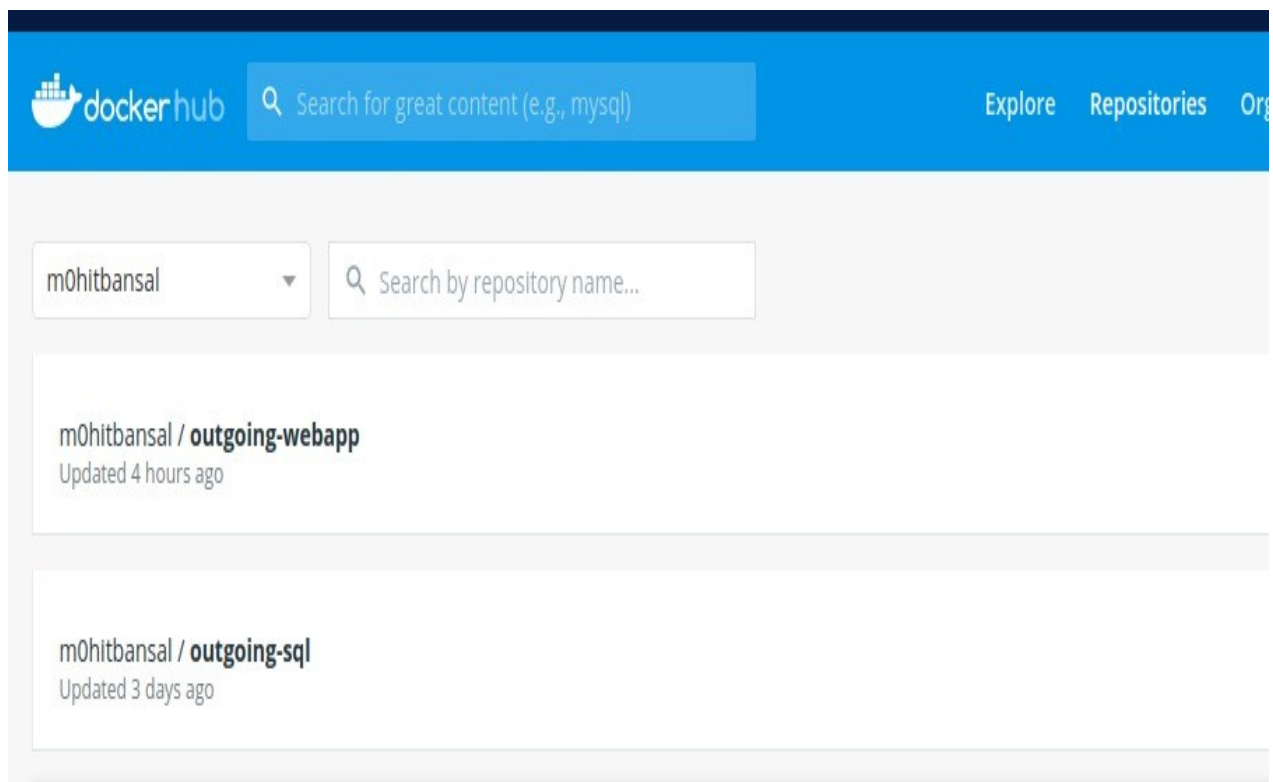


Fig 9: Dockerhub-Repository of Outgoing management portal

4.5 Continuous Integration

Continuous integration is the practice of constantly merging development work with a Master/Trunk/Mainline branch so that you can test changes and test that those changes work with other changes. The idea here is to test your code as often as possible so you can catch issues early on. In the continuous integration process, most of the work is done by an automated tests technique which requires a unit test framework.

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 [8].

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 [13].

`$ sudo usermod -aG docker Jenkins` – This command adds jenkins to docker group you can verify it with

`$ sudo grep jenkins /etc/gshadow` - This command verifies if jenkins has been added in group or not

Rundeck Configuration in Jenkins

Go to Manage Jenkins → configure system

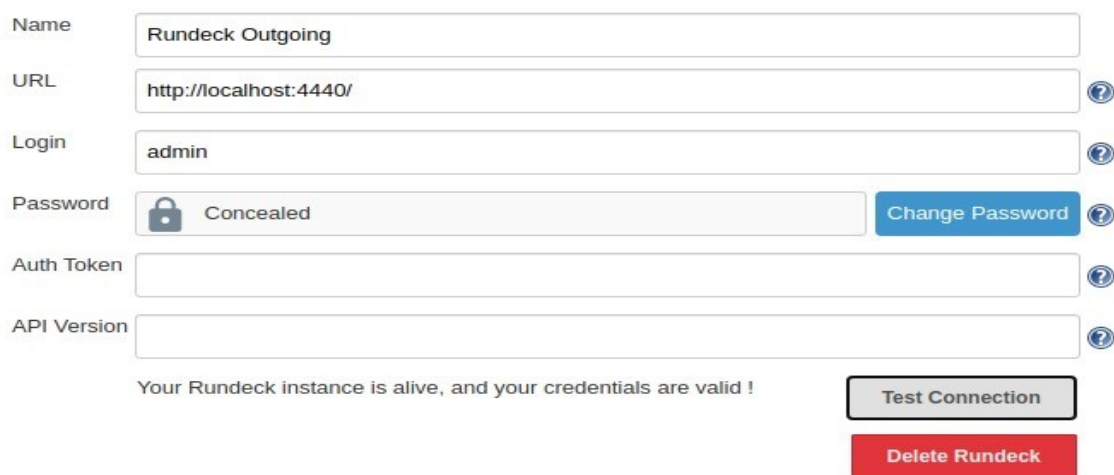
Under Rundeck enter the following configuration:

Name: Rundeck Outgoing

Url: http://localhost:4440

Login: admin

Password: admin



Name: Rundeck Outgoing

URL: http://localhost:4440/

Login: admin

Password: Concealed Change Password

Auth Token

API Version

Your Rundeck instance is alive, and your credentials are valid !

Test Connection

Delete Rundeck

Fig 10: Rundeck configuration in jenkins

Docker Configuration

Go to Jenkins → credentials → System → Global Credentials

Under Global Credentials:

Username: m0hitbansal

Password: *****

ID: docker-hub-credentials

The screenshot shows the Jenkins 'Global credentials (unrestricted)' configuration page for the user 'm0hitbansal/*****'. The left sidebar contains links: 'Back to Global credentials (unrestricted)', 'Update', 'Delete', and 'Move'. The main form includes the following fields:

- Scope:** A dropdown menu set to 'Global (Jenkins, nodes, items, all child items, etc)'.
- Username:** A text input field containing 'm0hitbansal'.
- Password:** A text input field with a lock icon and the word 'Concealed', followed by a blue 'Change Password' button.
- ID:** A text input field containing 'docker-hub-credentials'.
- Description:** An empty text input field.

A blue 'Save' button is located at the bottom left of the form.

Fig 11: Docker configuration in jenkins

Github webhook Configuration in Jenkins

Go to Manage Jenkins → configure system

Under Github Server:

Name: webhook

API Url: <https://api.github.com>

The screenshot shows the 'GitHub Servers' configuration page in Jenkins. The main configuration for the 'GitHub Server' is as follows:

- Name:** A text input field containing 'webhook'.
- API URL:** A text input field containing 'https://api.github.com'.
- Credentials:** A dropdown menu showing 'GitHub (https://api.github.com) auto generated token credentials for m', with an 'Add' button next to it.
- Status:** A message states 'Credentials verified for user m0hitbansal, rate limit: 4998'.
- Manage hooks:** A checkbox that is checked.
- Buttons:** 'Test connection' (grey), 'Advanced...' (grey), and 'Delete' (red).

At the bottom, there is an 'Add GitHub Server' button with a dropdown arrow.

Fig 12: Github webhook Configuration in Jenkins

Setting up Jenkins Pipeline

Click on new item

Enter project Name: Outgoing-management-pipeline

Select pipeline project

Click Ok

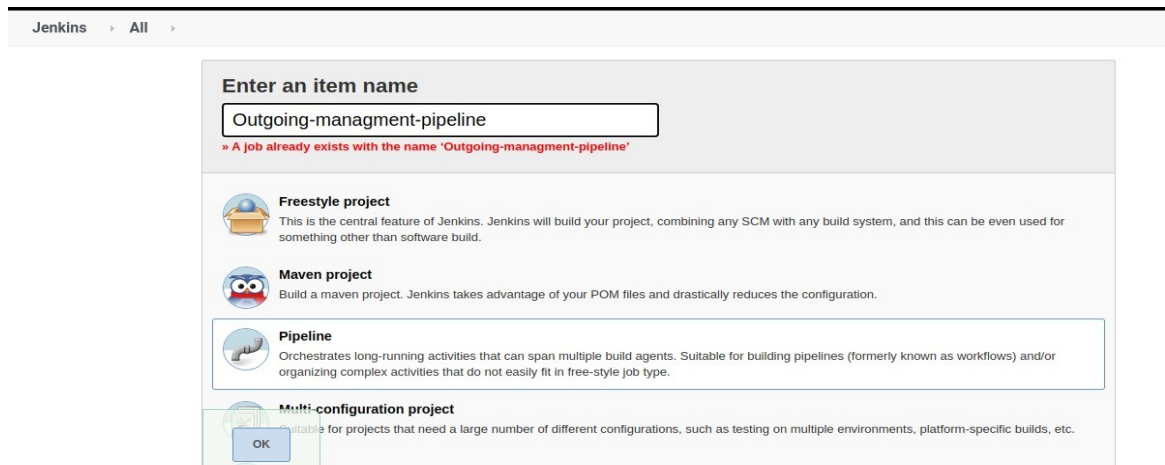


Fig 13: Create New Pipeline project

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

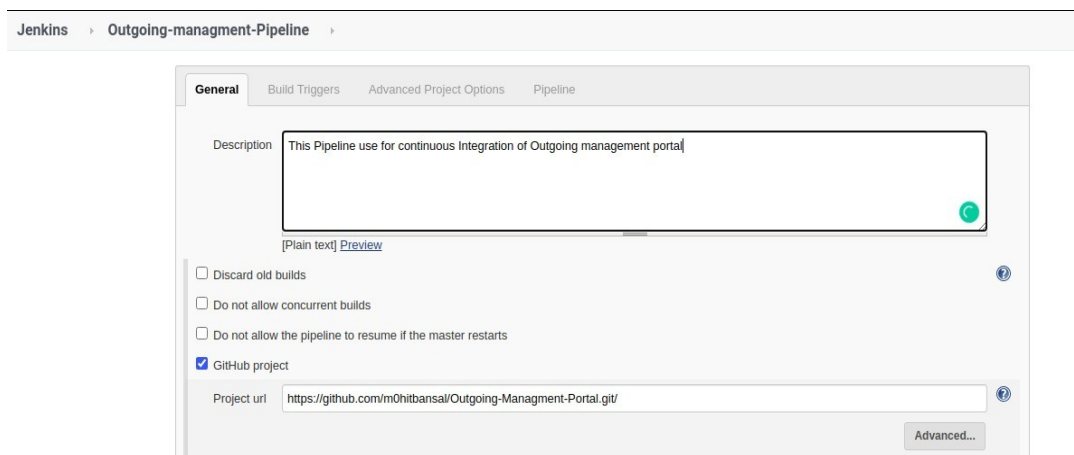


Fig 14: Add Github project

In Build Triggers

Check github hook trigger GITScm Polling

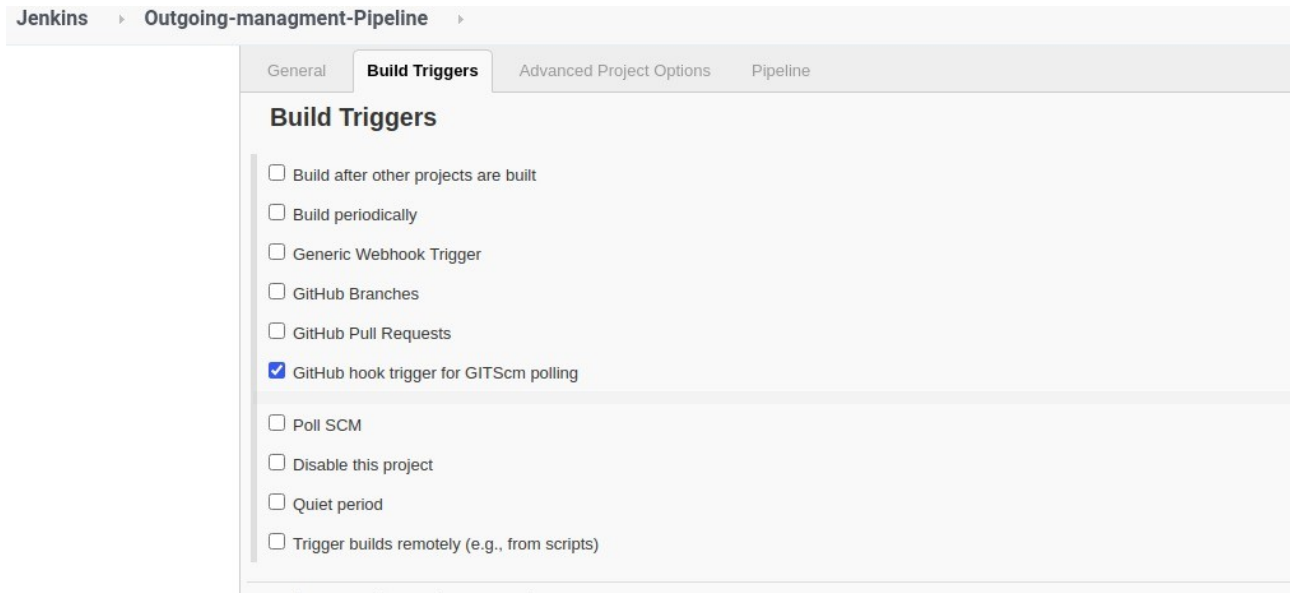


Fig 15: Set Webhook as Build trigger

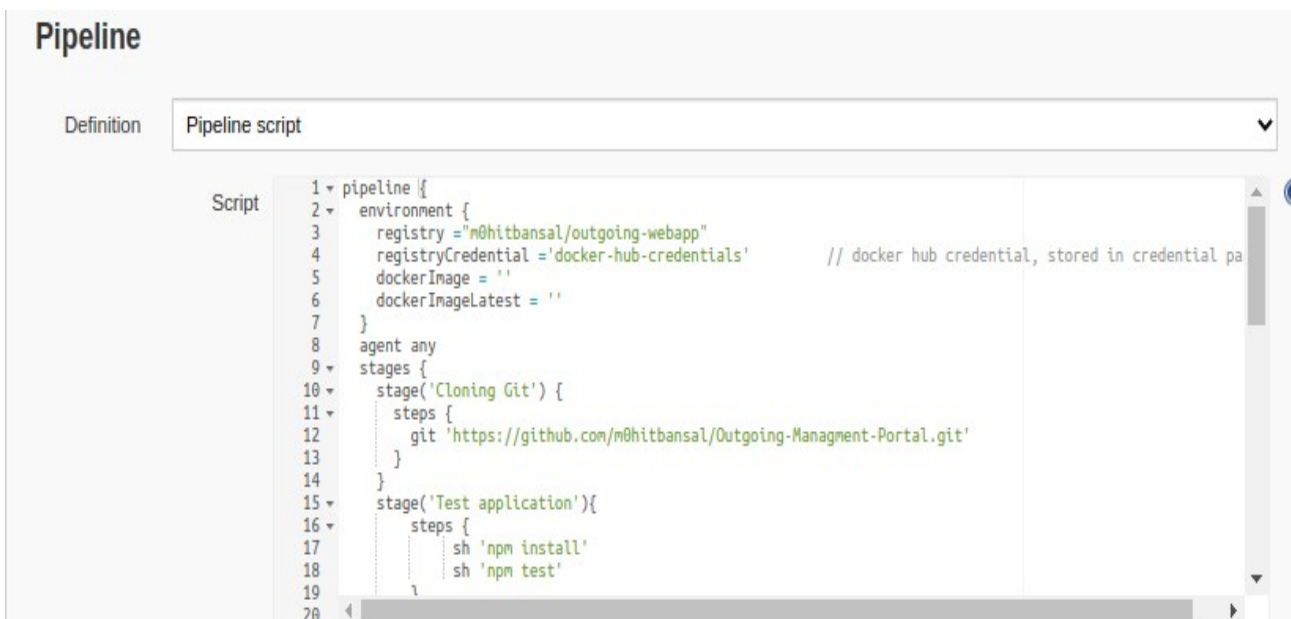


Fig 16: Add Pipeline script

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

Reference link and image shown below:

<https://github.com/m0hitbansal/Outgoing-Management-Portal/blob/master/jenkinsfile>

```
1 pipeline {
2   environment {
3     registry = "m0hitbansal/outgoing-webapp"
4     registryCredential = 'docker-hub-credentials' // docker hub cred
5     dockerImage = ''
6     dockerImageLatest = ''
7   }
8   agent any
9   stages {
10    stage('Cloning Git') {
11      steps {
12        git 'https://github.com/m0hitbansal/Outgoing-Management-Portal.git'
13      }
14    }
15    stage('Test application'){
16      steps {
17        sh 'npm install'
18        sh 'npm test'
19      }
20    }
21    stage('Building image') {
22      steps{
23        script {
24          dockerImageLatest = docker.build registry + ":latest"
25        }
26      }
27    }
28  }
```

Fig 17: Pipeline script-1

```
28   stage('Deploy Image') {
29     steps{
30       script {
31         docker.withRegistry( '', registryCredential ) {
32           dockerImageLatest.push()
33         }
34       }
35     }
36   }
37   stage('Remove Unused docker image') {
38     steps{
39       sh "docker rmi $registry:latest"
40     }
41   }
42   stage('Execute Rundeck job') {
43     steps {
44       script {
45         step([$class: "RundeckNotifier",
46             includeRundeckLogs: true,
47             jobId: "8eb64841-d14e-4c86-9a7e-a3ea08dfd0cd",
48             rundeckInstance: "Rundeck Outgoing",
49             shouldFailTheBuild: true,
50             shouldWaitForRundeckJob: true,
51             tailLog: true])
52       }
53     }
54   }
55 }
56 }
```

Fig 18: 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.

```
nt-Pipeline  , #20
> git --version # timeout=10
> git fetch --tags --progress -- https://github.com/m0hitbansal/Outgoing-M
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision d3af1ff2be352493c60f24a5ac23b0401d3c5037 (refs/remote
> git config core.sparsecheckout # timeout=10
> git checkout -f d3af1ff2be352493c60f24a5ac23b0401d3c5037 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D master # timeout=10
> git checkout -b master d3af1ff2be352493c60f24a5ac23b0401d3c5037 # timeout
Commit message: "Merge pull request #14 from m0hitbansal/archit"
> git rev-list --no-walk 16dfee49ale0b49668a1d921383a4bb550d6c0cf # timeout
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test application)
[Pipeline] sh
+ mocha --exit
Node server is running..
Browser to http://127.0.0.1: 5555

/Get student details
You are now connected...
data for given roll number picked up
  ✓ Main page content (268ms)

successful checkout
MT2019048
2020-05-23 12:28:47
into local check out for student
  ✓ Main page content (116ms)

successful checkin
9
into local check in for student
  ✓ Main page content (69ms)

3 passing (466ms)
```

Fig 20: Console Output-1

```
[12:37:09] [NORMAL] Error: No such image: m0hitbansal/outgo
[12:37:09] [NORMAL] Using default tag: latest
[12:37:13] [NORMAL] latest: Pulling from m0hitbansal/outgoi
[12:37:13] [NORMAL] Digest: sha256:3032612e13b51d9b0f88c573
[12:37:13] [NORMAL] Status: Downloaded newer image for m0hi
[12:37:13] [NORMAL] docker.io/m0hitbansal/outgoing-sql:late
[12:37:14] [NORMAL] Using default tag: latest
[12:37:17] [NORMAL] latest: Pulling from m0hitbansal/outgoi
[12:37:17] [NORMAL] 1c6172af85ee: Already exists
[12:37:17] [NORMAL] b194b0e3c928: Already exists
[12:37:17] [NORMAL] 1f5ec00f35d5: Already exists
[12:37:17] [NORMAL] 93b1353672b6: Already exists
[12:37:18] [NORMAL] 3d7f38db3cca: Already exists
[12:37:18] [NORMAL] 21e102f9fe89: Already exists
[12:37:18] [NORMAL] e63dac87cb8e: Already exists
[12:37:18] [NORMAL] 2e15f38664d9: Already exists
[12:37:18] [NORMAL] 7639f95d3d43: Already exists
[12:37:18] [NORMAL] ba53c21961fb: Pulling fs layer
[12:37:18] [NORMAL] 8aeb23a843f1: Pulling fs layer
[12:37:20] [NORMAL] ba53c21961fb: Verifying Checksum
[12:37:20] [NORMAL] ba53c21961fb: Download complete
[12:37:21] [NORMAL] ba53c21961fb: Pull complete
[12:37:42] [NORMAL] 8aeb23a843f1: Verifying Checksum
[12:37:42] [NORMAL] 8aeb23a843f1: Download complete
[12:37:47] [NORMAL] 8aeb23a843f1: Pull complete
[12:37:47] [NORMAL] Digest: sha256:c6bf4f4ac9dd4be879729578
[12:37:47] [NORMAL] Status: Downloaded newer image for m0hi
[12:37:47] [NORMAL] docker.io/m0hitbansal/outgoing-webapp:l
END RUNDECK TAILED LOG OUTPUT
Rundeck execution #110 finished in 40 seconds, with status
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Fig 19: Console Output-2

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.

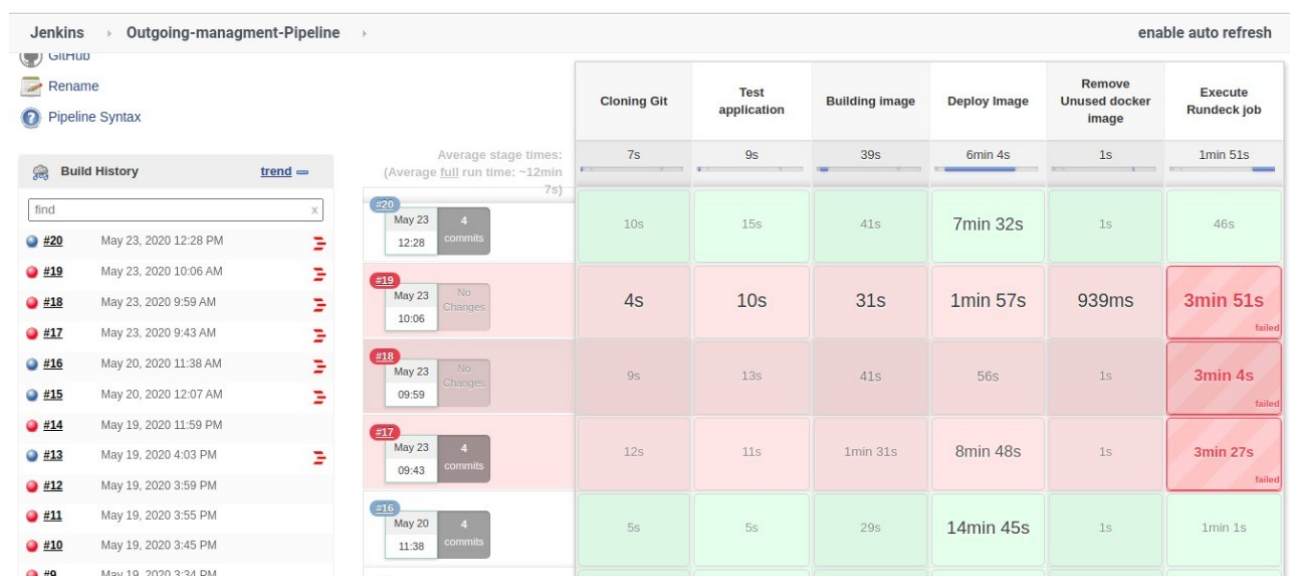


Fig 21: Pipeline Stage View

4.6 Continuous 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

In simple words 'Rundeck is runbook automation'. It gives anyone an individual-service access to the operations tasks that previously were only limited to experts. It provides us with self-functioning runbooks that work the way you want them to work. Rundeck comes as an open source automation service with its own web console, command line tools and a WebAPI. With the help of rundeck we can comfortably execute tasks on a set of nodes with automation. It is a cross-platform open source service that automates ad-hoc and routine procedures on cloud environment or data centers. Other features that make RunDeck easy to scale up your scripting efforts include: access control, building workflow, scheduling, logging, and integration with external sources for node and option data [9, 11].

Environment setup for Rundeck

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

`$ usermod -aG docker rundeck` : This command adds rundeck to the docker group

Restart Rundeck i.e.

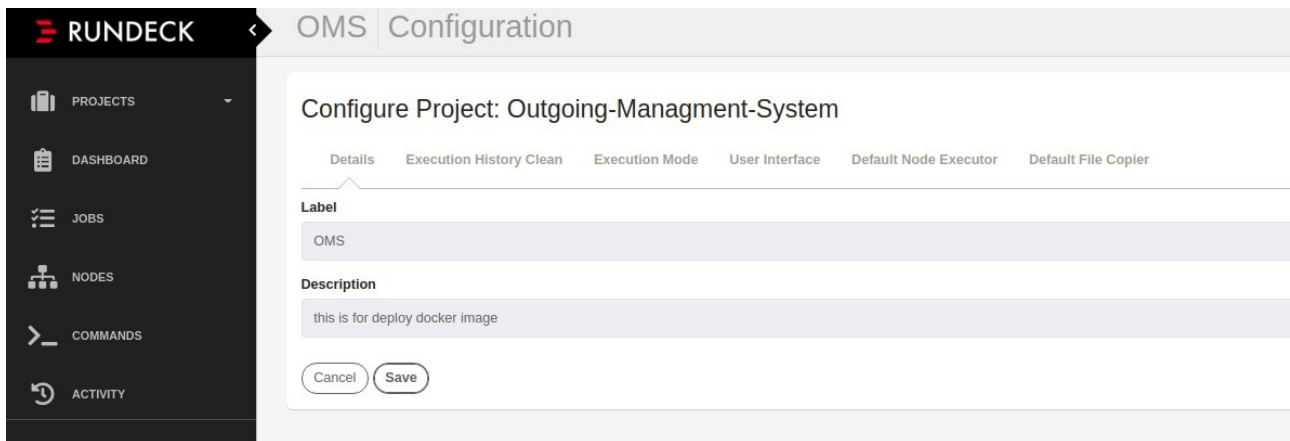
`$ systemctl restart rundeckd` : This command restart rundeck

`$ sudo grep docker /etc/gshadow` : This command verifies if rundeck added or not in docker group

This way, we can see how rundeck belongs to the same group and now we can run *root* commands in rundeck without using *sudo* [14].

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.

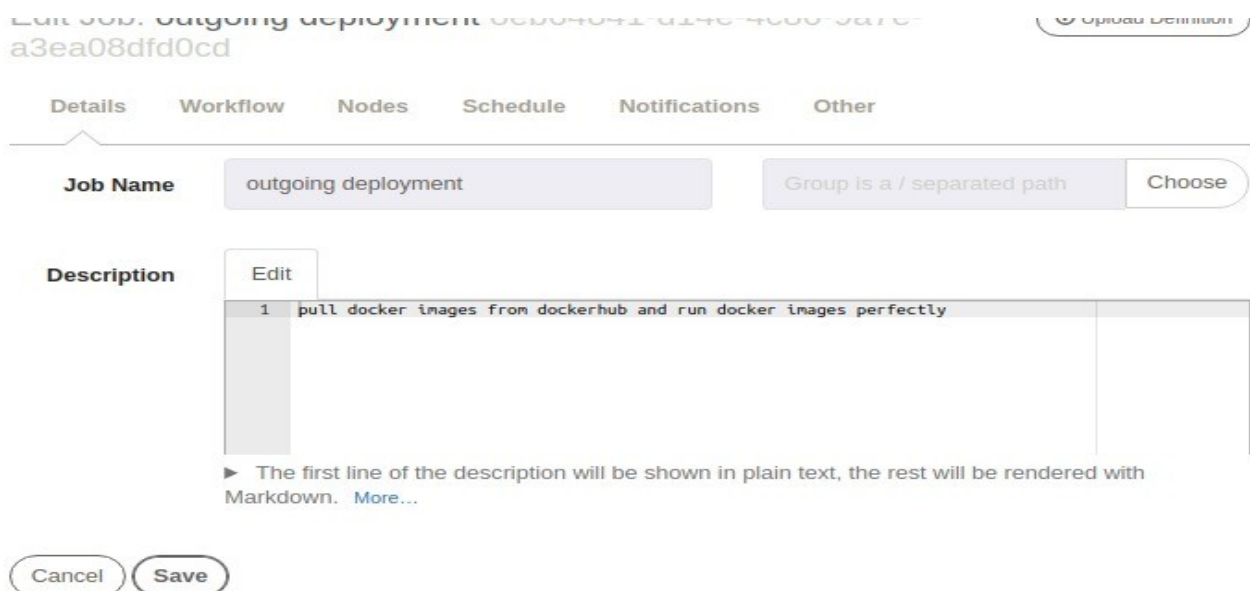


The screenshot shows the Rundeck web interface. On the left is a dark sidebar with navigation links: PROJECTS, DASHBOARD, JOBS, NODES, COMMANDS, and ACTIVITY. The main content area is titled 'Configure Project: Outgoing-Management-System'. It has several tabs: Details (selected), Execution History Clean, Execution Mode, User Interface, Default Node Executor, and Default File Copier. Under the 'Details' tab, there are two text input fields: 'Label' with the value 'OMS' and 'Description' with the value 'this is for deploy docker image'. At the bottom of the form are 'Cancel' and 'Save' buttons.

Fig 22: 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.

3) Create a new job, enter job name and description.



The screenshot shows the 'Edit Job' form in Rundeck. The job name is 'outgoing deployment'. The description is 'pull docker images from dockerhub and run docker images perfectly'. The form has tabs for Details, Workflow, Nodes, Schedule, Notifications, and Other. At the bottom are 'Cancel' and 'Save' buttons.

Job Name: outgoing deployment

Description: pull docker images from dockerhub and run docker images perfectly

23: Create New Job

Fig

4) Go to workflow and add the script to execute on the registered node.

Reference of job description in below link and image

https://github.com/m0hitbansal/Outgoing-Managment-Portal/blob/master/outgoing_deployment.yaml

Options

Undo Redo

No Options

+ Add an option

Workflow

If a step fails:

☒ Stop at the failed step. ☐ Run remaining steps before failing.

Strategy: Node First

Execute all steps on a node before proceeding to the next node.

Explain >

Global Log Filters + add

1. **Enter the entire script to execute**

```
1 /bin/bash
2 docker stop $(docker ps -aq)
3 docker rm $(docker ps -aq)
4 docker rmi -f m0hitbansal/outgoing-webapp
5 docker pull m0hitbansal/outgoing-sql
6 docker pull m0hitbansal/outgoing-webapp
7
```

Fig 24: Job definition

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
```

4.7 Continuous Monitoring

Elastic Search

Elastic search is a indexing querying over apache's lucene engine.

```
$ cd elasticsearch
```

```
$ ./bin/elasticsearch : start Elastic search in terminal
```

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 [1].

Kibana

```
$ cd kibana
```

```
$ ./bin/kibana : start kibana in terminal
```

You can access kibana GUI at <http://localhost:5601>

Logstash helps normalizing data from various data sources such as apache, log events, sql and other data sources.

Also, logstash enhances data us various filters such as geoip, grok, matcher and many more. Open another terminal and proceed for logstash [1].

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 25: Winston logs generate code

Winston send logs directly to Elastic-search so we don't need to configure Logstash
Below Link show how logs generate and stored in log file

<https://github.com/m0hitbansal/Outgoing-Managment-Portal/blob/master/combined.log>

Below image show how many logs generated and showing in kibana dashboard



Fig 26: Kibana -1 Logs Dashboard

Below image show how many times the various requests on Outgoing portal

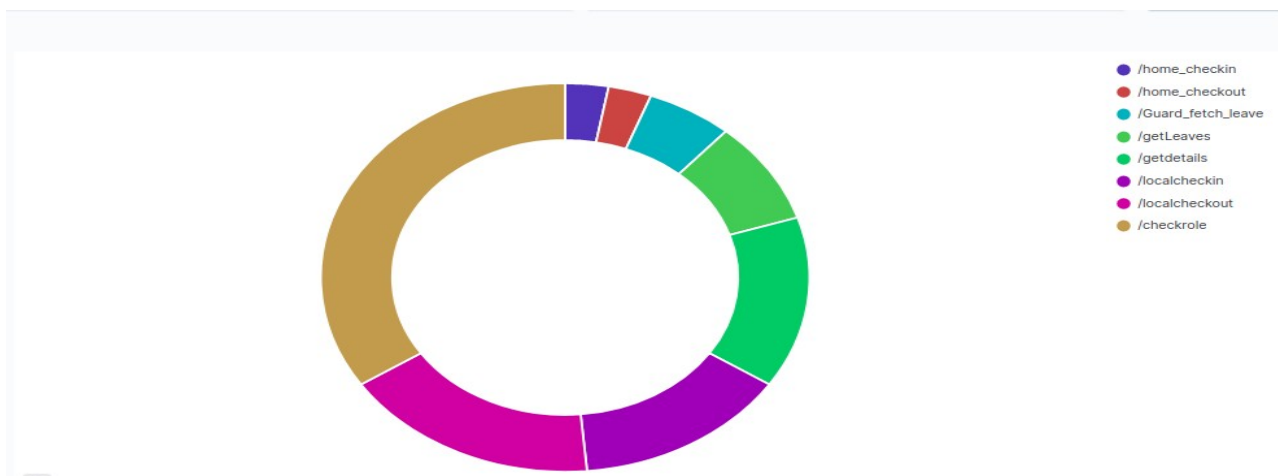


Fig 27: Count methods call

Below image shows that how many post and get request come to server

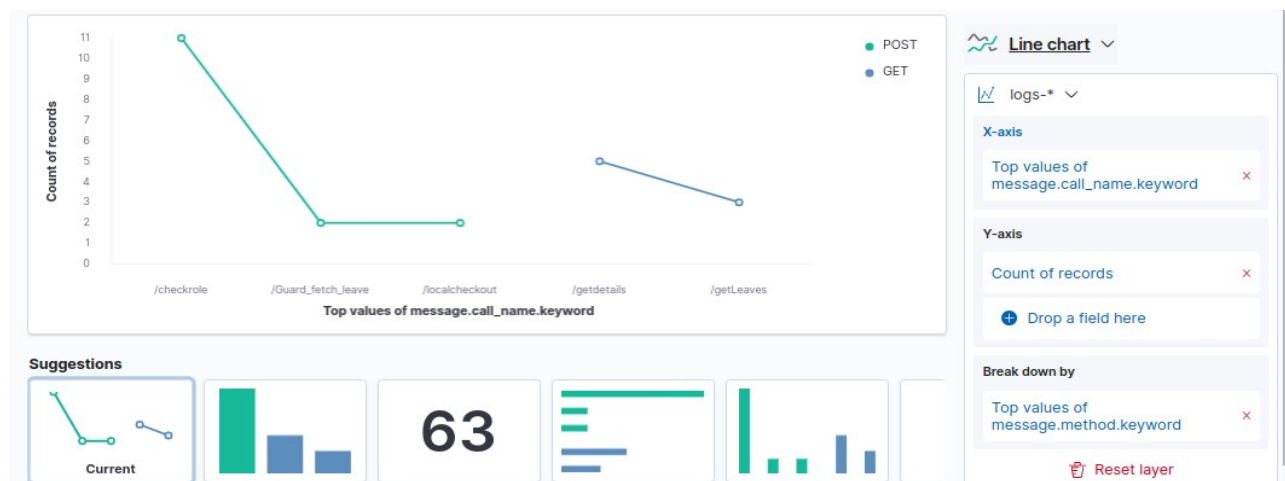


Fig 28: Kibana-3 Get/Post With timestamp

Below image shows that how many post and get request come to server with respect to index id.

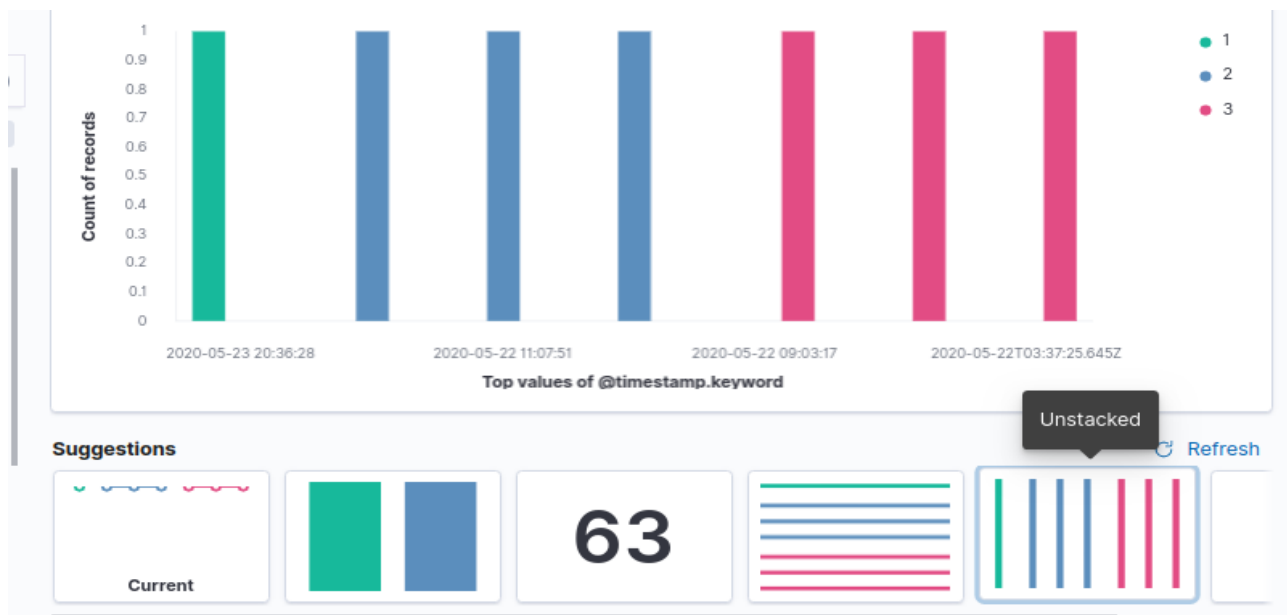


Fig 29: Kibana-4 Index id with time stamp

4.8 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 build.
- 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` : This command uses for give public ip to jenkins

```
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Account             Mohit Bansal (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://ffe5aa71.ngrok.io -> http://localhost:8080
Forwarding           https://ffe5aa71.ngrok.io -> http://localhost:8080

Connections         ttl      opn      rt1      rt5      p50      p90
                   0        0        0.00     0.00     0.00     0.00
```

Fig 30: start Ngrok on port 8080

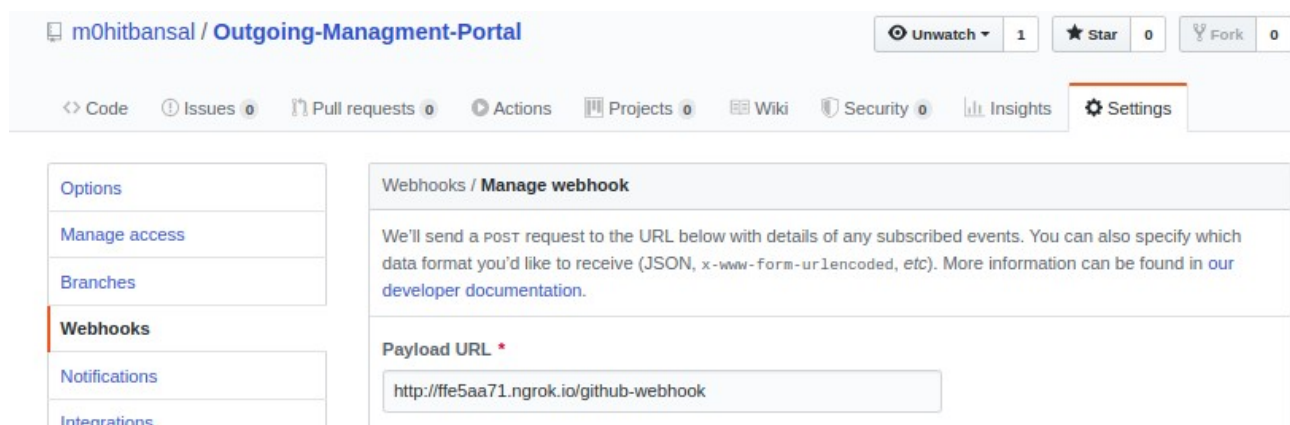


Fig 31: Github webhook setup

5. 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.

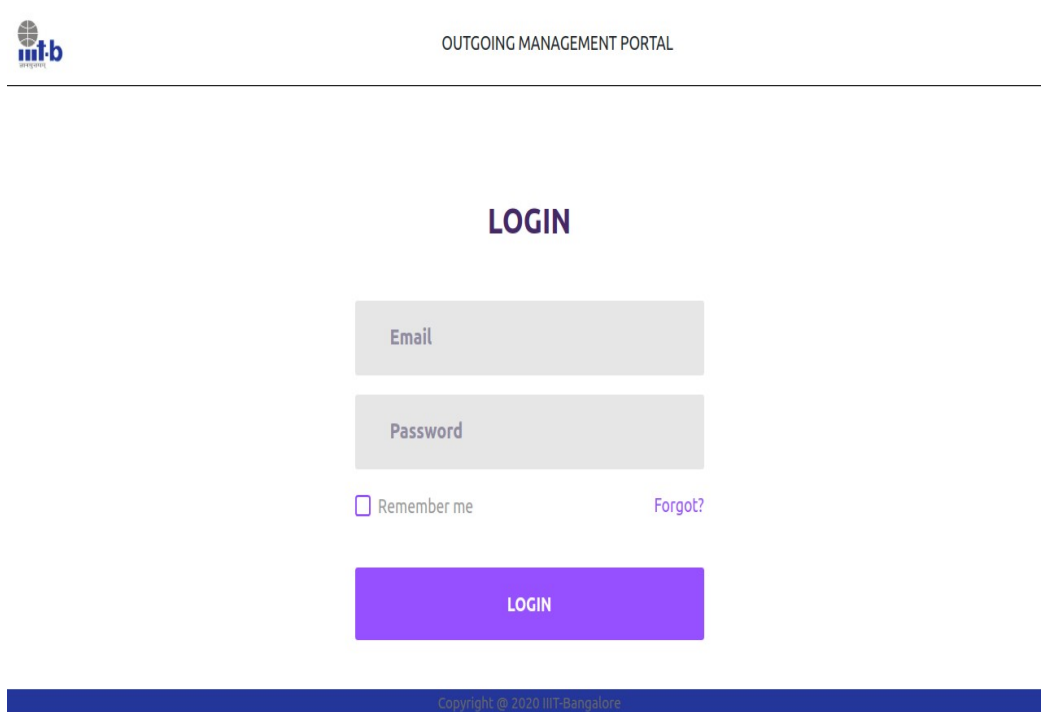
```
$ dpkg -i rundeck_2.10.8-1-GA_all.deb
```

Install ELK Stack:

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

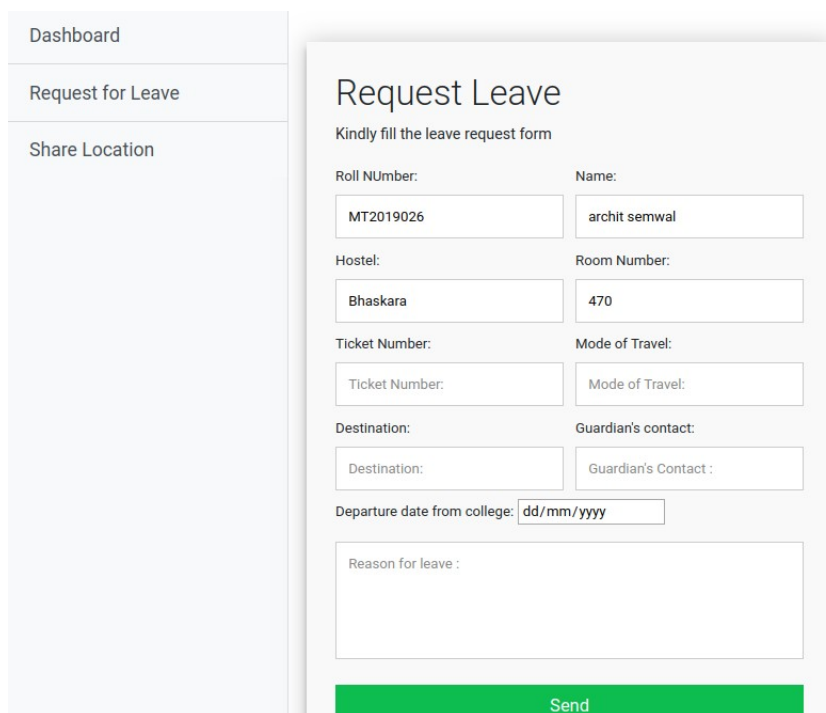
6. Result and Discussion

1. Login page for warden , Guard, and students.



The image shows a login page for the 'OUTGOING MANAGEMENT PORTAL'. At the top left is the 'mtb' logo. The title 'OUTGOING MANAGEMENT PORTAL' is centered at the top. Below the title, the word 'LOGIN' is displayed in a large, bold, purple font. Underneath, there are two input fields: 'Email' and 'Password', both with placeholder text. Below these fields is a checkbox labeled 'Remember me' and a link labeled 'Forgot?'. A large purple button labeled 'LOGIN' is positioned below the input fields. At the bottom of the page, a dark blue footer contains the text 'Copyright @ 2020 IIT-Bangalore'.

Fig 32: Login Page



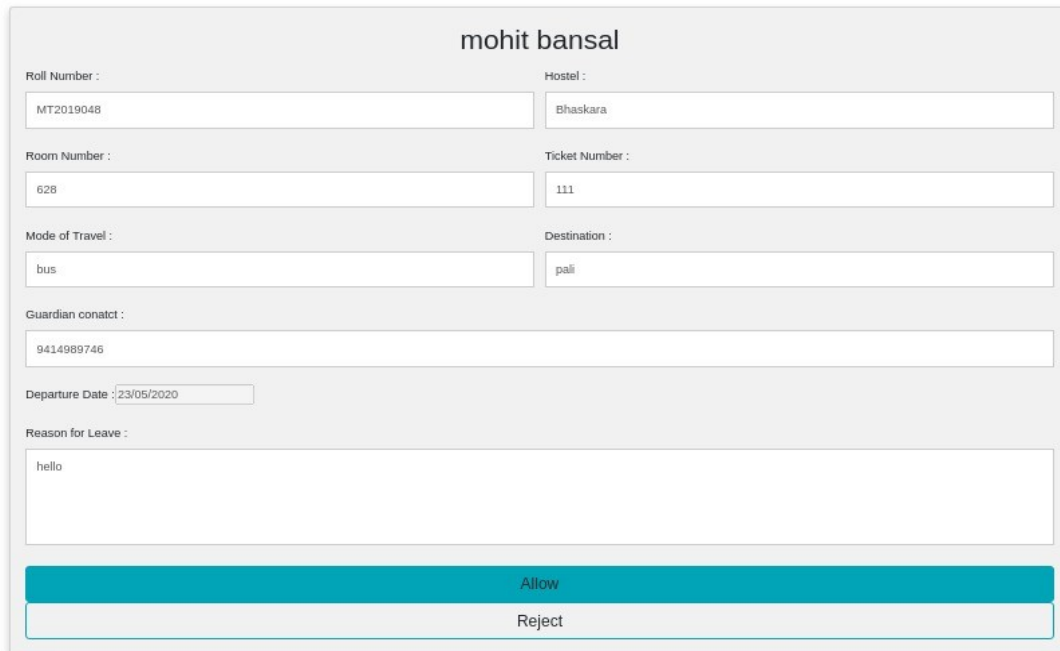
The image displays a 'Request Leave' form. On the left is a sidebar with three menu items: 'Dashboard', 'Request for Leave', and 'Share Location'. The main content area is titled 'Request Leave' and includes the instruction 'Kindly fill the leave request form'. The form contains several input fields: 'Roll Number' (with value 'MT2019026'), 'Name' (with value 'archit semwal'), 'Hostel' (with value 'Bhaskara'), 'Room Number' (with value '470'), 'Ticket Number' (with placeholder 'Ticket Number:'), 'Mode of Travel' (with placeholder 'Mode of Travel:'), 'Destination' (with placeholder 'Destination:'), 'Guardian's contact' (with placeholder 'Guardian's Contact :'), and 'Departure date from college' (with placeholder 'dd/mm/yyyy'). A large text area for 'Reason for leave :' is located below the date field. A green button labeled 'Send' is at the bottom of the form.

Fig 33: Student Leave Form Page

2. Warden permission of leave approval page

Leave Request Column Cards

Click on Accept/Reject...



The form is titled "mohit bansal" and contains the following fields:

Roll Number :	Hostel :
MT2019048	Bhaskara
Room Number :	Ticket Number :
628	111
Mode of Travel :	Destination :
bus	pali
Guardian contact :	
9414989746	
Departure Date : 23/05/2020	
Reason for Leave :	
hello	

At the bottom, there are two buttons: "Allow" (teal) and "Reject" (light blue).

Fig 34: Warden's Approval Page

3. After Reject or Approve permission of leave Mail has been sent to student .

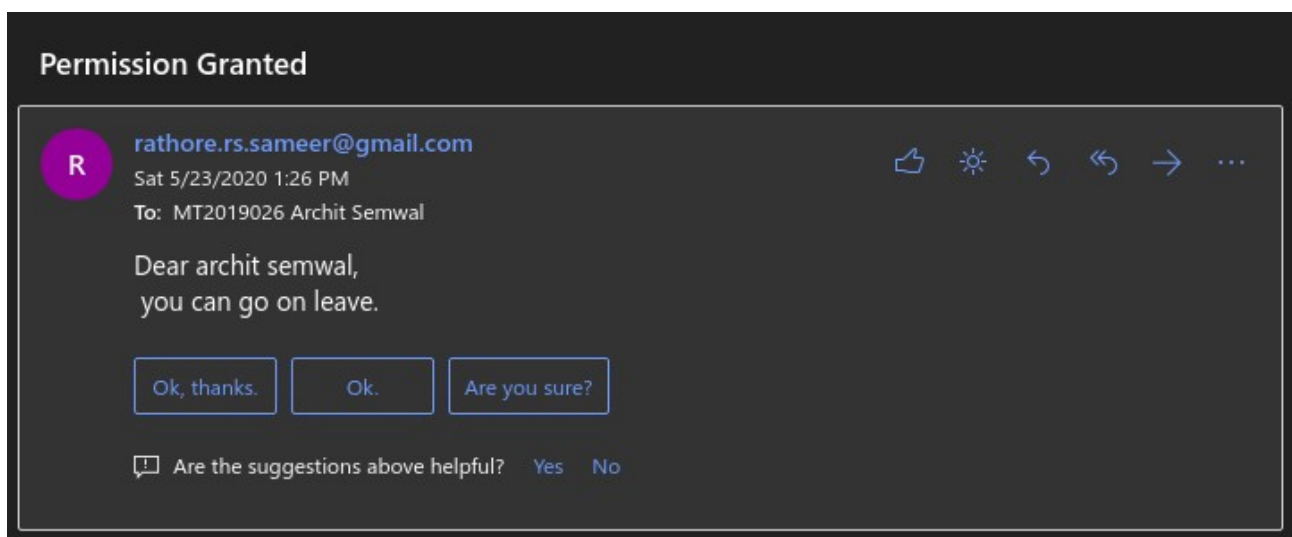
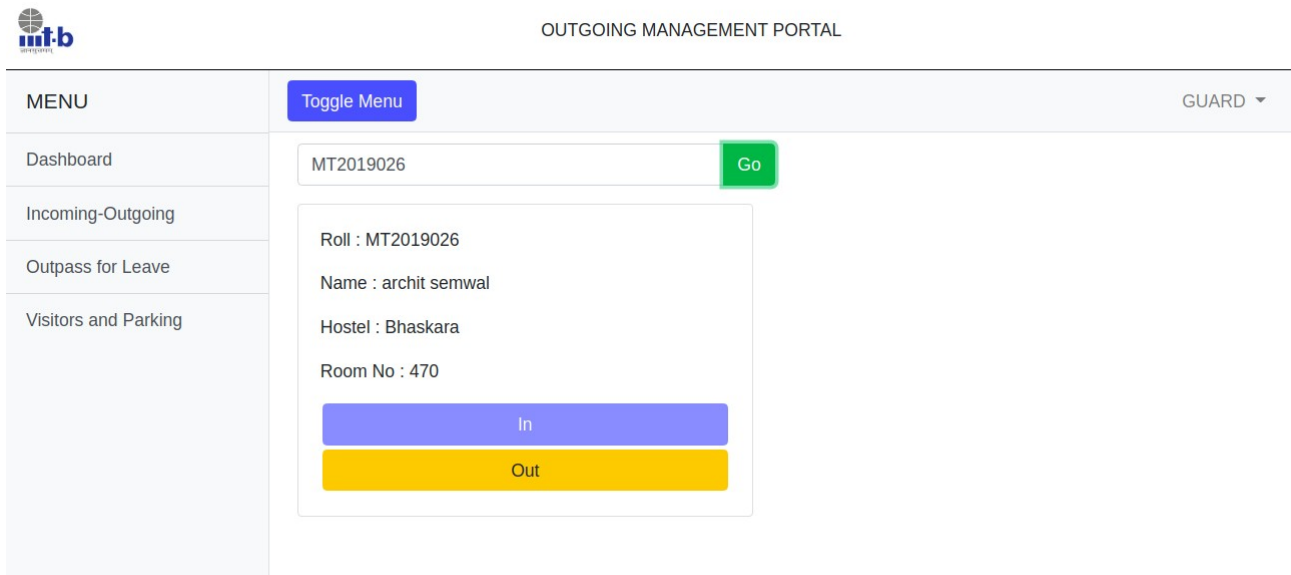


Fig 35: Autogenerated email for permission granted

4. Student local In or Out entry page on Guard side



OUTGOING MANAGEMENT PORTAL

MENU Toggle Menu GUARD ▾

Dashboard Go

Incoming-Outgoing

Outpass for Leave

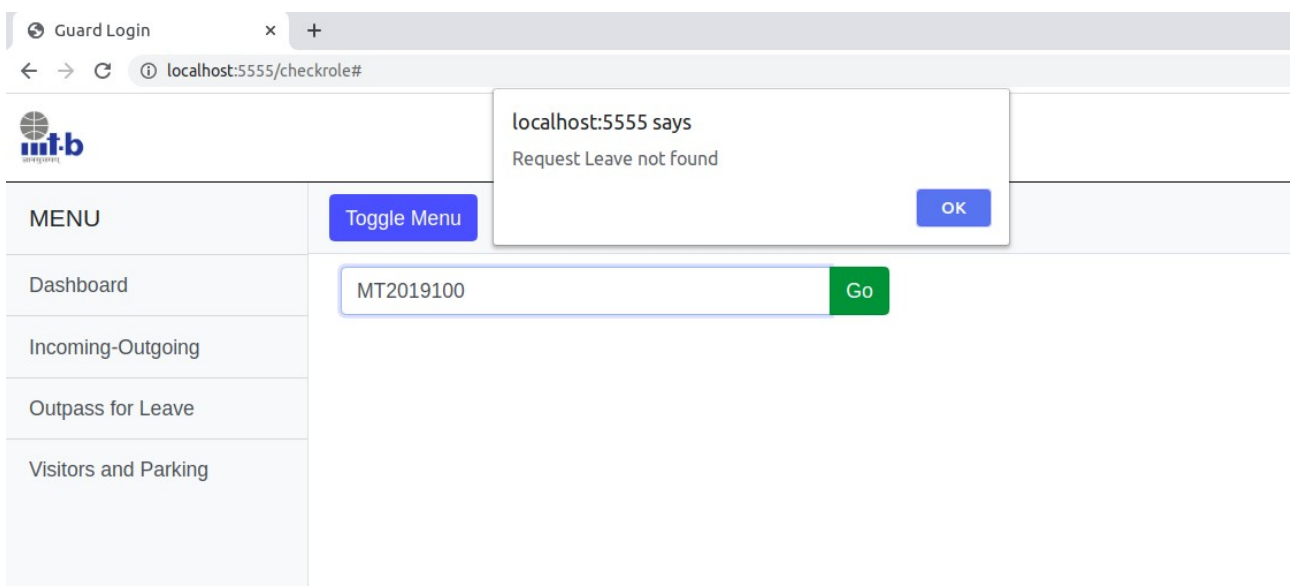
Visitors and Parking

Roll : MT2019026
Name : archit semwal
Hostel : Bhaskara
Room No : 470

In
Out

Fig 36: Local Outpass

5. If Student doesn't request for leave or leave is not same day then pop show that request leave not found.



Guard Login x +

localhost:5555/checkrole#

localhost:5555 says
Request Leave not found OK

MENU Toggle Menu

Dashboard Go

Incoming-Outgoing

Outpass for Leave

Visitors and Parking

Fig 37: Leave not found alert

The screenshot shows the 'OUTGOING MANAGEMENT PORTAL' interface. A 'Leave Outpass' modal is open, containing the following fields:

MT2019026	archit semwal
Bhaskara	470
INDIGO-QR45Zn2X	flight
Dehradun	Approved By Warden
23/05/2020	

At the bottom of the modal are two buttons: 'Departure' (blue) and 'Return' (yellow).

Fig 38: Outstation outpass Page and entry in database

The notification is titled 'Your Ward is leaving for home'. It includes the following details:

- Sender:** rathore.rs.sameer@gmail.com
- Date/Time:** Sat 5/23/2020 1:27 PM
- To:** MT2019026 Archit Semwal
- Message:** Dear Parent, Your ward archit semwal, has left for home. Please be informed.
- Response Options:**
 - Thank you for letting us know.
 - Thanks for the heads up.
 - Thank you for informing me.
- Feedback:** Are the suggestions above helpful? Yes No

Fig 39: Guardian's Notification when student go to home

The notification is titled 'Your Ward has arrived college'. It includes the following details:

- Sender:** rathore.rs.sameer@gmail.com
- Date/Time:** Sat 5/23/2020 1:27 PM
- To:** MT2019026 Archit Semwal
- Message:** Dear Parent, Your ward archit semwal, has arrived in college. Please be informed.
- Response Options:**
 - Thank you for letting us know.
 - Thank you!
 - Thank you for the update.
- Feedback:** Are the suggestions above helpful? Yes No

Fig 40: Guardian's Notification when student come from home

Outpass for Leave

Visitors and Parking

Visitor Details

Kindly fill the visitor form

Name :

Email Id. :

Meeting with Prof. G. N. Prasanna

Vehicle Entry : ☒

Vehicle Number :

Parking Slot :

Enter

Fig 41: This page uses for Visitor entry with vehicle

Dashboard

Incoming-Outgoing

Outpass for Leave

Visitors and Parking

Visitor Exit

Exit

Visitor Details

Kindly fill the visitor form

Name :

Email Id. :

Meeting with Prof. G. N. Prasanna

Vehicle Entry : ☐

Enter

Fig 42: This page uses for Visitor entry without vehicle

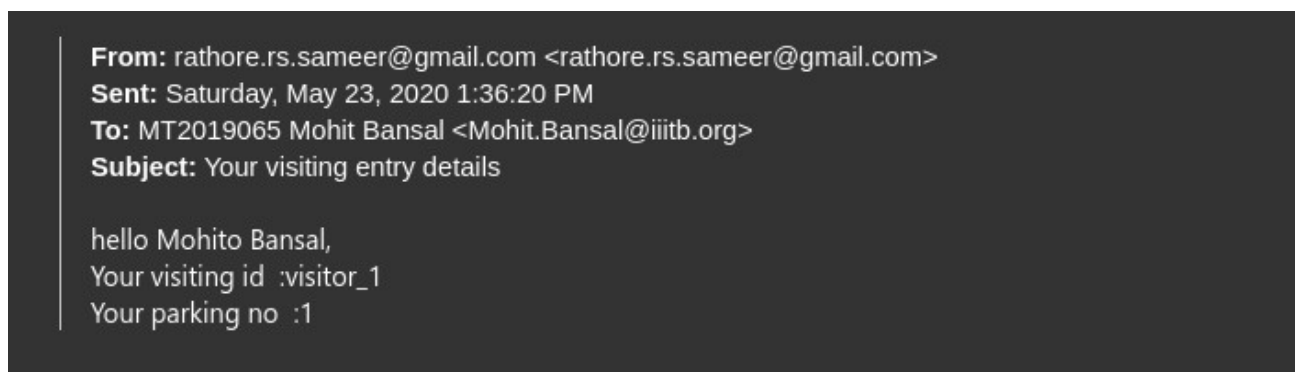


Fig 43: Allot unique id and parking slot to Visitor and send mail to visitor mail id

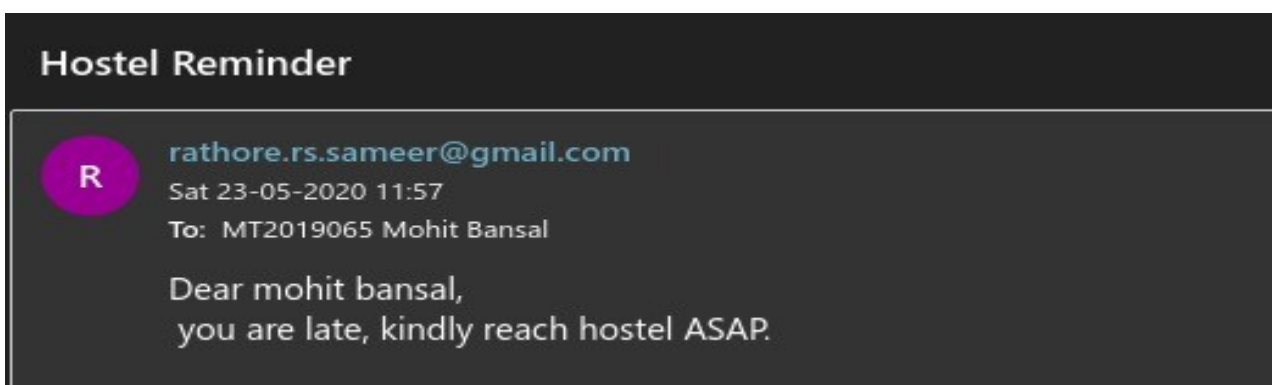


Fig 44: Student's late reminder email

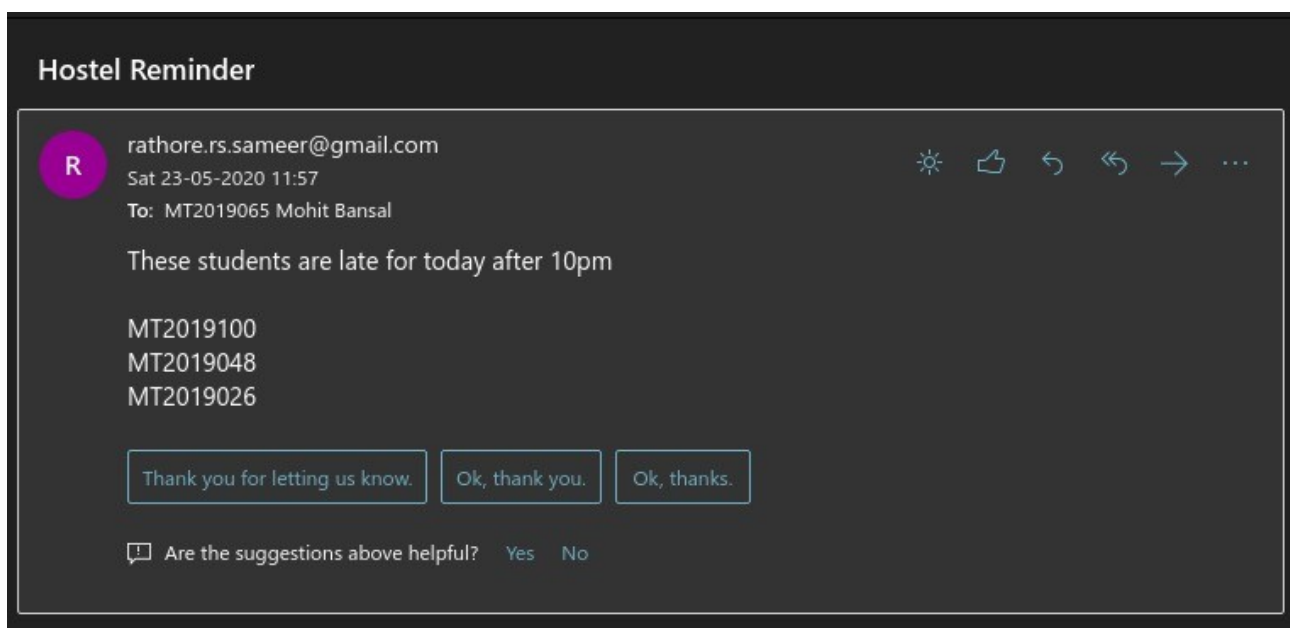


Fig 45: Send mail of list of all late student to warden

7. 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.

8. 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. References

- [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>
- [5] <https://mochajs.org/>
- [6] <https://www.chaijs.com/>
- [7] A. M. Fard and A. Mesbah, "JavaScript: The (Un)Covered Parts," 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), Tokyo, 2017, pp. 230-240, doi: 10.1109/ICST.2017.28.
- [8] <https://www.jenkins.io/doc/>
- [9] <https://docs.rundeck.com/docs/>
- [10] <https://docs.docker.com/>
- [11] <https://foxutech.com/what-is-rundeck/>
- [12] C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, "DevOps," in IEEE Software, vol. 33, no. 3, pp. 94-100, May-June 2016, doi: 10.1109/MS.2016.68.
- [13] V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery," 2015 IEEE/ACM 3rd International Workshop on Release Engineering, Florence, 2015, pp. 24-27, doi: 10.1109/RELENG.2015.19.
- [14] D. Spinellis, "Service Orchestration with Rundeck," in IEEE Software, vol. 31, no. 4, pp. 16-18, July-Aug. 2014, doi: 10.1109/MS.2014.92.
- [15] H. Kang, M. Le and S. Tao, "Container and Microservice Driven Design for Cloud Infrastructure DevOps," 2016 IEEE International Conference on Cloud Engineering (IC2E), Berlin, 2016, pp. 202-211, doi: 10.1109/IC2E.2016.26.