MAJOR-1 PROJECT

END TERM REPORT

For

CHIME: Let's connect

Submitted By

Specialization	SAP ID	Name
CCVT (H)	500086703	Shikhar Nag
CCVT(H)	500082951	Mani Paliwal
CCVT(H)	500083727	Bhoomika
CCVT(H)	500085000	Nupur Sharma

Department of Systemics

School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN- 248007. Uttarakhand

Dr. Alind
Project Guide

Dr. Hitesh Kumar Sharma
Cluster Head

End Term Report

1. Project Title

CHIME: Let’s connect

2. Abstract

This project explores the innovative world of cloud-based interview scheduling tools, which play a pivotal role in revolutionizing the recruitment process. Leveraging cloud computing capabilities, organizations can seamlessly coordinate interviews, optimize scheduling efficiency, and enhance the overall candidate experience. Cloud-based interview scheduling tools are revolutionizing the recruitment process. These tools use cloud computing to coordinate interviews seamlessly, optimize scheduling efficiency, and enhance the candidate experience. One of the biggest advantages of cloud-based interview scheduling tools is that they allow organizations to easily coordinate interviews across different time zones and locations. This is essential in today's globalized economy, where many companies hire remote workers. Cloud-based interview schedulers also make it easy to track the status of interviews and ensure that they are scheduled in a timely manner.

In addition to improving efficiency, cloud-based interview scheduling tools also enhance the candidate experience. Applicants can easily select interview slots that align with their schedules, which can create a positive impression of the hiring organization. These tools can also be used to send automated reminders to candidates about upcoming interviews. While cloud-based interview scheduling tools offer many advantages, there are also some challenges to consider. One challenge is that these tools can be expensive to implement and maintain. Another challenge is that they can be difficult to integrate with existing HR systems. Despite these challenges, cloud-based interview scheduling tools offer several benefits that make them a valuable tool for any organization looking to improve its recruitment process.

3. Introduction

The advent of cloud technology has revolutionized various aspects of modern business operations, including the realm of human resources. This project delves into the innovative landscape of cloud-based interview schedulers, a crucial tool in streamlining the recruitment process. By harnessing the benefits of the cloud, organizations can seamlessly coordinate interviews, optimize scheduling efficiency, and enhance candidate experiences. The evolution of cloud technology has fundamentally reshaped the landscape of modern business operations. Among its many applications, cloud-based interview schedulers have emerged as a transformative solution in the realm of human resources.

The system architecture for "CHIME: Let's Connect" encompasses a well-structured frontend built with EJS to provide user-friendly interfaces for recruiters and applicants. The frontend facilitates interview slot management, invitation sending, and response tracking for recruiters. On the backend, Node.js and Express.js power a RESTful API that manages HTTP requests,

handles core application logic, and interfaces with the MongoDB database. The database is organized into collections for various data types, ensuring efficient querying and scheduling algorithms. For robust hosting and scalability, the system leverages cloud infrastructure using AWS, Google Cloud and MongoDB Atlas for cloud-based database management.

In summary, "CHIME: Let's Connect" adopts a modern system architecture that combines EJS for the frontend, Node.js and Express.js for the backend, and MongoDB for data storage. Cloud infrastructure choices enhance scalability and reliability, making it a well-rounded solution for efficient interview scheduling and management.

In an era marked by increased global connectivity and remote work, the ability to efficiently coordinate interviews across time zones and locations is paramount. Cloud-based interview schedulers offer organizations a dynamic platform to manage and streamline this intricate process. By centralizing scheduling, they eliminate the complexities of coordinating between multiple stakeholders, optimizing the recruitment pipeline's efficiency.

Moreover, these solutions enhance the candidate experience, offering flexibility and ease of use. Applicants can conveniently select interview slots that align with their schedules, fostering a positive impression of the hiring organization.

However, the implementation of cloud-based interview schedulers is not without its challenges.

4. Literature Review

This [1] web-based scheduling system leverages participants' schedules to recommend suitable timings, streamlining the process. Initiators extend invitations to participants, who can then respond after reviewing the meeting agenda. This innovative solution confers substantial benefits upon organizers, offering a remedy to common scheduling obstacles.

Numerous enterprises arrange meetings, leading to avoidable costs such as printing minutes on paper. To address this, MPL Perera and DU Vidanagama [2] introduced a paper-free meeting management system for Kotelwala Defence University. This innovative system enables easy access to meeting information and provides navigation assistance to the meeting location.

Through deliberate selection of dates and venues for each meeting, the system [3] employs JavaScript, CSS, and MySQL for its functionality. The integration of the Meeting Management System within an organization holds the potential to enhance scheduling efficiency and resource allocation, leading to a more optimized timetable and resource allocation for meetings.

The scheduling of meetings in most meeting rooms or within a meeting room management system [4] follows a specific timetable. Nevertheless, occasional frustrations arise due to

unforeseen scheduling conflicts during meetings, as specific timings aren't always available. By integrating PIR sensor fusion devices and Ethernet connectivity, it becomes feasible to efficiently schedule meeting rooms, enhancing the utilization of these spaces and addressing such issues effectively.

5. Problem Statement

In today's fast-paced work environment, coordinating and managing daily meetings efficiently is a challenge that many individuals and teams face. The process of manually scheduling meetings, finding suitable time slots, and accommodating participants' availability can be time-consuming and error-prone. This often leads to missed meetings, overlapping schedules, and decreased productivity.

6. Objectives

- To distribute incoming tasks or jobs based on importance evenly across available resources to prevent resource bottleneck and also help users to get information according to the requirements.
- To generate instant notifications for real time communication with clientele.
- To deploy schedulers on the cloud to ensure high availability of applications and services by distributing tasks across multiple servers or instances.

7. Methodology

We will be creating a comprehensive system architecture for "CHIME: Let's Connect" will involve designing the overall structure of the application, including its components, databases, and the interaction between various elements. Additionally, determining the cloud infrastructure and technologies is essential for hosting and scaling the application.

System Architecture:

1. Frontend:

The user-friendly interfaces for both recruiters and applicants are a critical part of the system. Using EJS (Embedded JavaScript) for the front end, which is a good choice for generating dynamic HTML content. Here's how you can structure it:

Recruiter Interface: This interface is designed for recruiters to create, manage, and schedule interviews. It should include features like creating interview slots, sending invitations using mail, tracking candidate responses, and integrating Google Calendar for saving the dates.

2. Backend:

Using Node.js and Express.js for the backend. This is a robust choice for building a RESTful API to handle requests and manage the application's logic.

API Layer: This layer is responsible for handling HTTP requests from the frontend, processing data, and interacting with the database. It should include endpoints for creating, updating, and deleting interview slots, sending invitations, and handling applicant responses.

Business Logic: Implement the core business logic for scheduling interviews, avoiding time clashes, and sending notifications or reminders to both recruiters and applicants.

3. Database:

MongoDB Cloud is the database for storing user calendar data and other relevant information. Organize our data into collections for recruiters, applicants, interviews, and time slots. This ensures that data is structured efficiently to support querying and scheduling algorithms.

4. Cloud Infrastructure and Technologies:

Cloud Platform: We'll use a cloud platform like Amazon Web Services (AWS) and Google Cloud.

Server Hosting: We are going to Deploy our Node.js application on cloud platforms like Render and then create a development instance of AWS EC2 and use Docker and Jenkins(Continuous building tool) for checking if our application has errors or not .

Google Calendar API: This is used to mark the slots for the booked interview on the Google Calendar so that a person can keep track of their aligned interviews.

Database Hosting MongoDB Atlas is a cloud-based MongoDB service that integrates seamlessly with cloud platforms. It offers managed database clusters, automated backups, and high availability.

5. Devops

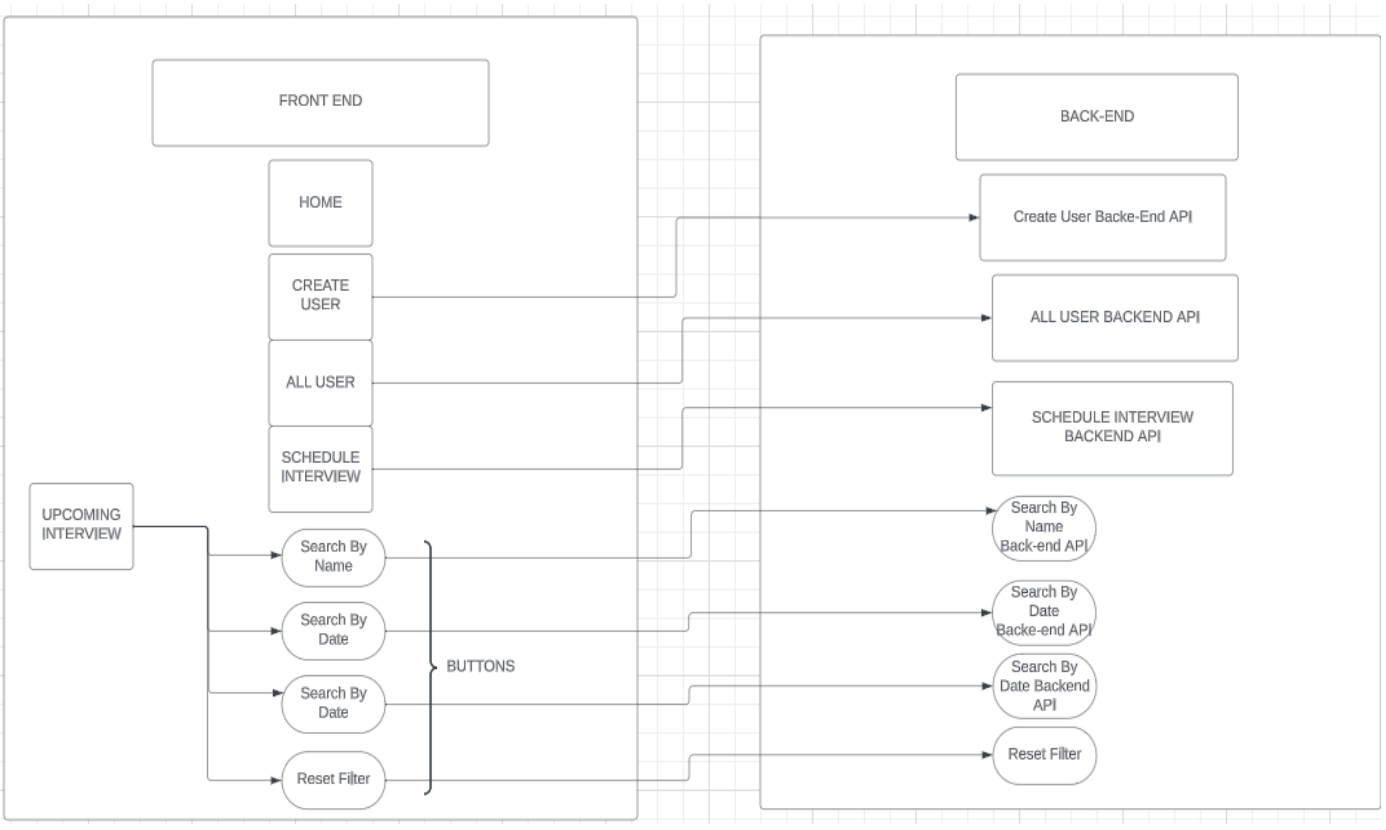
DevOps practices play a crucial role in ensuring the smooth integration, deployment, and continuous improvement of the CHIME application. Docker and Jenkins are fundamental tools in this process.

Docker: (Containerization and Orchestration)

Docker is employed to containerize the Node.js application and its dependencies. Containers provide consistency across development, testing, and production environments. Each component of the application, including the Node.js server, can run in a lightweight, isolated container, ensuring easy deployment and scalability. Docker allows for efficient resource utilization and helps mitigate the classic "it works on my machine" problem.

Jenkins Pipeline: (Continuous Integration and Continuous Deployment (CI/CD))

Jenkins plays a pivotal role in the Continuous Integration and Continuous Deployment (CI/CD) pipeline for the CHIME: Let's Connect application. By connecting to the version control system, such as GitHub, Jenkins automatically triggers the pipeline upon code changes. The process involves building Docker images and running tests to validate the codebase's integrity. Subsequently, the built Docker images are pushed to a container registry, enabling accessibility for deployment. Jenkins seamlessly orchestrates the deployment of the application to cloud platforms like AWS or Google Cloud, fostering automation and reducing the likelihood of errors. The CI/CD pipeline provides rapid feedback on code changes, expediting issue identification and resolution, while the automated deployment process facilitates easy scalability based on demand, ensuring a consistent and reliable deployment lifecycle.

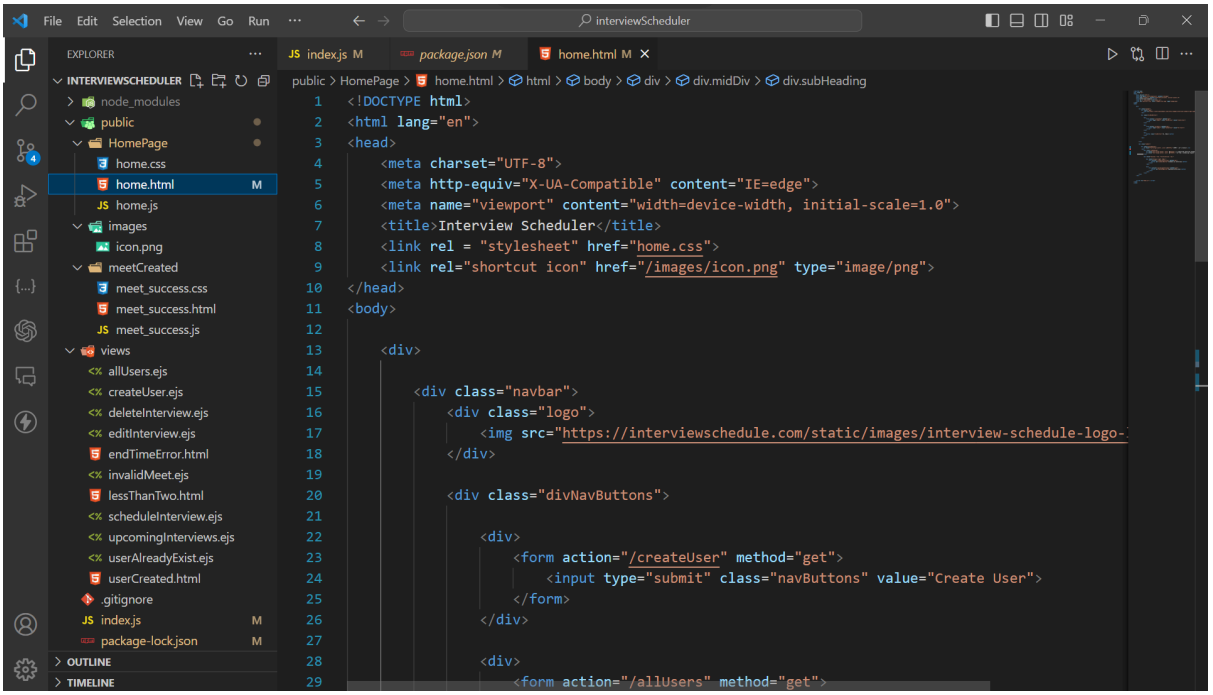


8. Implementation

Github Link: https://github.com/Bhoomika121002/Interview_Scheduler

Website Link: https://interviewscheduler-2szw.onrender.com/

Code:



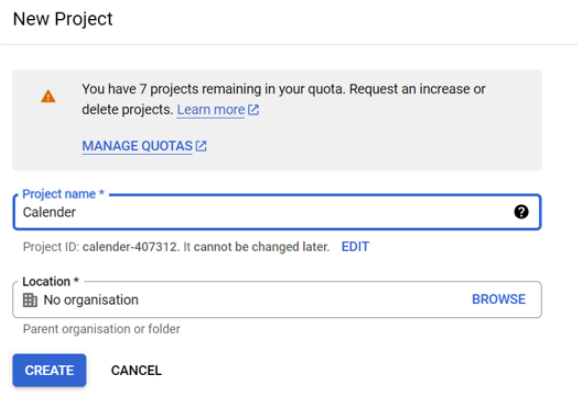
```
})

let sendMail = (async (data, sub, message) => {
    participants = data.participants
    interview_date = data.interview_date
    startTime = data.startTime
    endTime = data.endTime

    const nodemailer = require('nodemailer')

    app.use(express.json());
app.use(express.urlencoded({ extended: false }));
const transporter = nodemailer.createTransport({
  service: "gmail",
  auth: {
    user: "nagshikharcareer@gmail.com",
    pass: "ruenjpyhwukvwpjm"//get this password from two step authentication of gmail.
  }
});

    var mailList = []

    for (const participant of participants) {

        var result = await db.collection('busySlots').findOne({ name: participant })

        console.log(result)
```

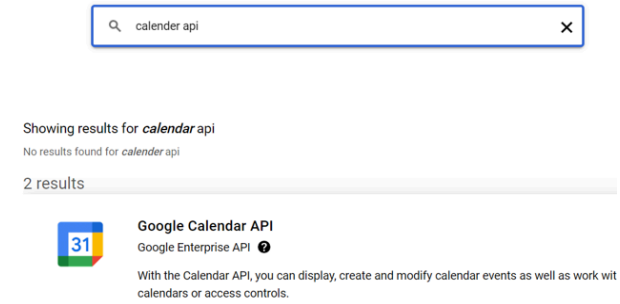
```
service: "gmail",
auth: {
  user: "nagshikharcareer@gmail.com",
  pass: "ruenjpyhwukvwpjm"//get this password from two step authentication of gmail.
}
});

  var mailList = []

  for (const participant of participants) {

      var result = await db.collection('busySlots').findOne({ name: participant })

      console.log(result)

      mailList.push(result.email)
  }

  console.log(mailList)

  const options = {
      from: "nagshikharcareer@gmail.com",
      to: mailList,
      subject: sub,
      text: 'Click the following link to join our Google Meet: ' + 'https://meet.google.com/tdr-raxa-mha',
  };

  transporter.sendMail(options, function (err, info) {
      if (err) {
```

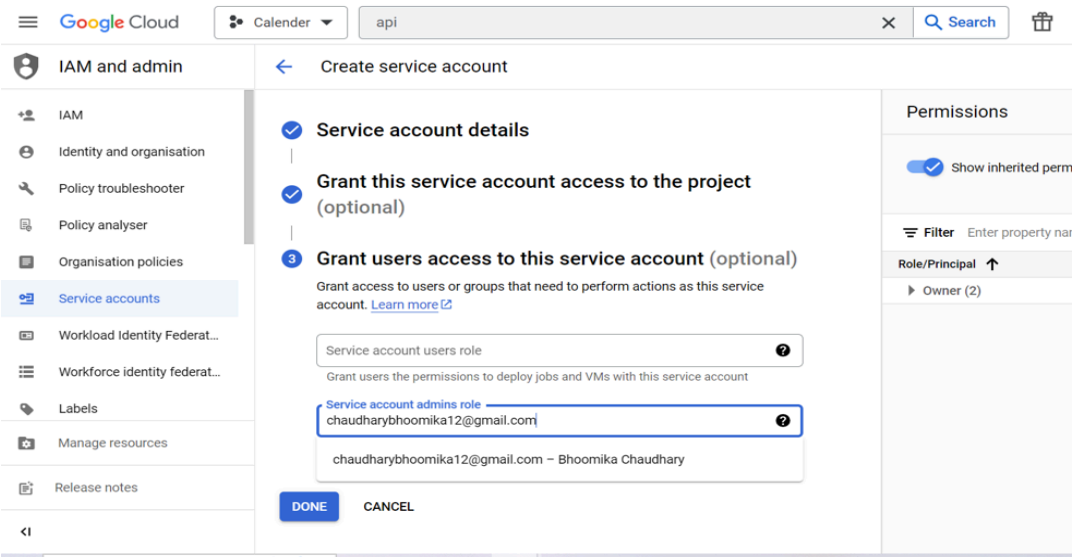
```
const options = {
    from: "nagshikharcareer@gmail.com",
    to: mailList,
    subject: sub,
    text: 'Click the following link to join our Google Meet: ' + 'https://meet.google.com/tdr-raxa-mha',
};

transporter.sendMail(options, function (err, info) {
    if (err) {
        console.log(err)
        return;
    }
    console.log("Sent : " + info.response);
})
```

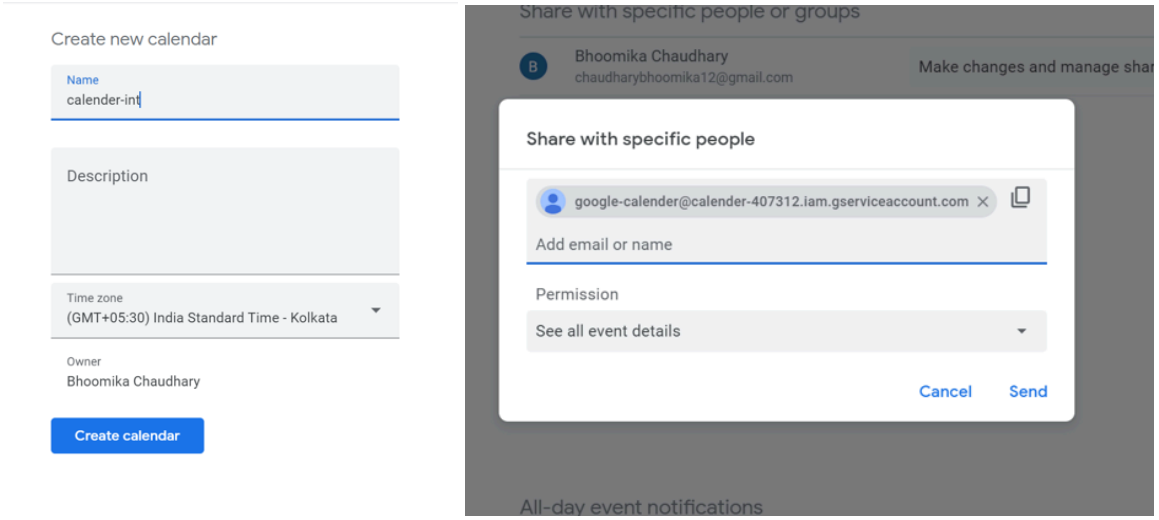
```
require('dotenv').config();
var express = require("express");
var bodyParser = require("body-parser");

const mongoose = require('mongoose');
const mongodbUri = "mongodb://0.0.0.0:27017/interviewScheduler";// Adjust with your link
mongoose.set('strictQuery', true);

mongoose.connect(mongodbUri, { useNewUrlParser: true }, (err) => {

    if (!err) {
        console.log("MongoDB connected successfully.");
    }
    else {
        console.log("Error in DB connection : ", err);
    }
})
var db = mongoose.connection;
db.on('error', console.log.bind(console, "connection error"));
db.once('open', function (callback) {
    console.log("connection succeeded");
})
```

Google Calendar Integration:

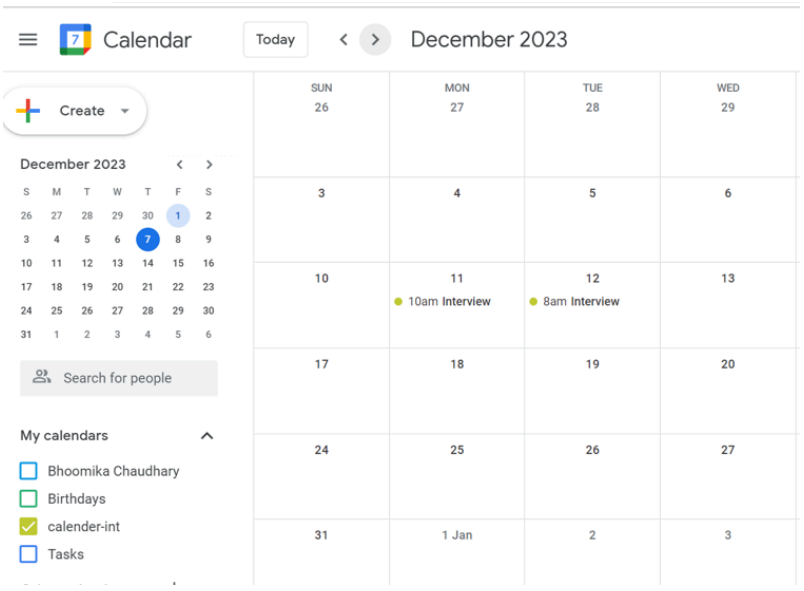









```
PS C:\Users\hp\Downloads\interviewScheduler-main> npm install googleapis

added 26 packages, and audited 285 packages in 11s

32 packages are looking for funding

12 vulnerabilities (7 moderate, 2 high, 3 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS C:\Users\hp\Downloads\interviewScheduler-main> set GOOGLE_APPLICATION_CREDENTIALS=C:\Users\hp\Downloads\calender-4073
12-229586166dc1.json
PS C:\Users\hp\Downloads\interviewScheduler-main> nodemon app.js
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Server is running on port 3001
Event created: {
  kind: 'calendar#event',
  etag: '"3403789251878000"',
  id: 'rj26ql7mteqg42rq6h0vl16igk',
  status: 'confirmed',
  htmlLink: 'https://www.google.com/calendar/event?eid=cmoyNnFsN210ZXFnNDJycTZoMHZsMTZpZ2sgNWYzMWM3MzMxMmM0Y2JkYmI0ZWY5Z
jg0YzQ3MDYyZWQzMDA1Mjk0OTdjOGZjNzgyMTZhMmY5Nzk3OTc0YTMzYkBn',
  created: '2023-12-06T20:30:25.000Z',
  updated: '2023-12-06T20:30:25.939Z',
```

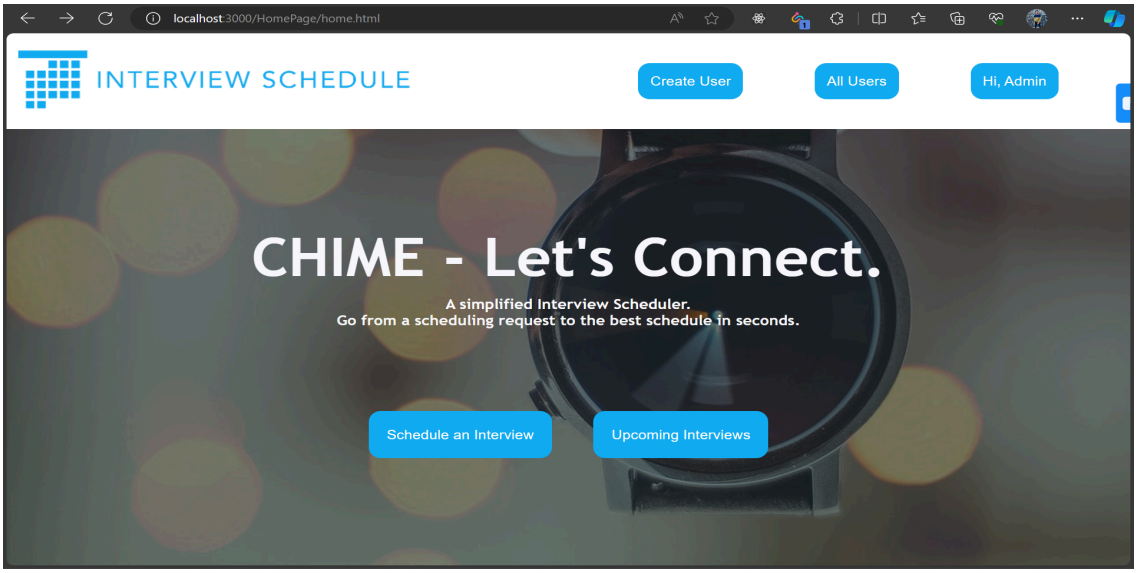


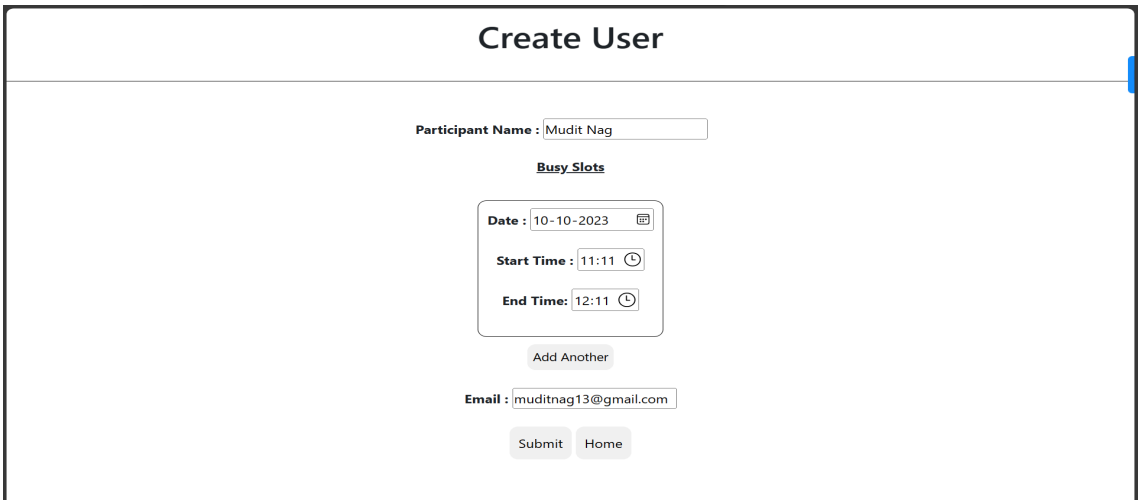
```
const express = require('express');
const { google } = require('googleapis');

const app = express();
const port = 3001;


const SCOPES = 'https://www.googleapis.com/auth/calendar.readonly';
const GOOGLE_PRIVATE_KEY = "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCyZdgyE
const GOOGLE_CLIENT_EMAIL = "google-calender@calender-407312.iam.gserviceaccount.com"
const GOOGLE_PROJECT_NUMBER = "23792116215"
const GOOGLE_CALENDAR_ID = "5f31c73312c4cbdbb4ef9f84c47062ed300529497c8fc78216a2f9797974a33b@group.calendar.google

const jwtClient = new google.auth.JWT(
  GOOGLE_CLIENT_EMAIL,
  null,
  GOOGLE_PRIVATE_KEY,
  SCOPES
);

const calendar = google.calendar({
  version: 'v3',
  project: GOOGLE_PROJECT_NUMBER,
  auth: jwtClient
});
```

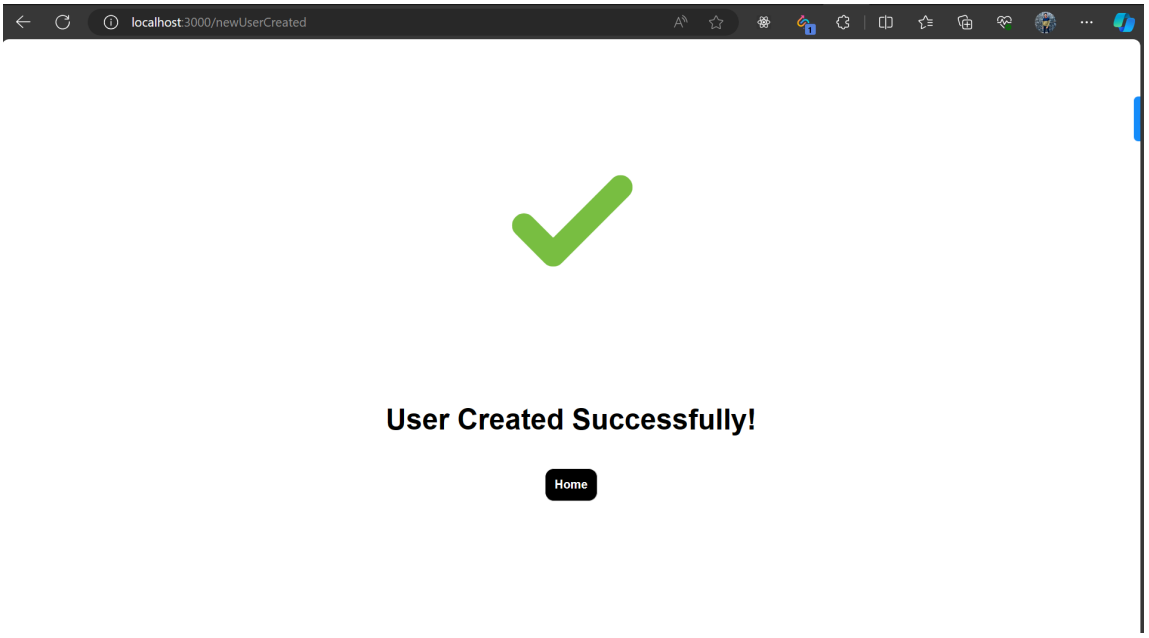
Dockerfile:

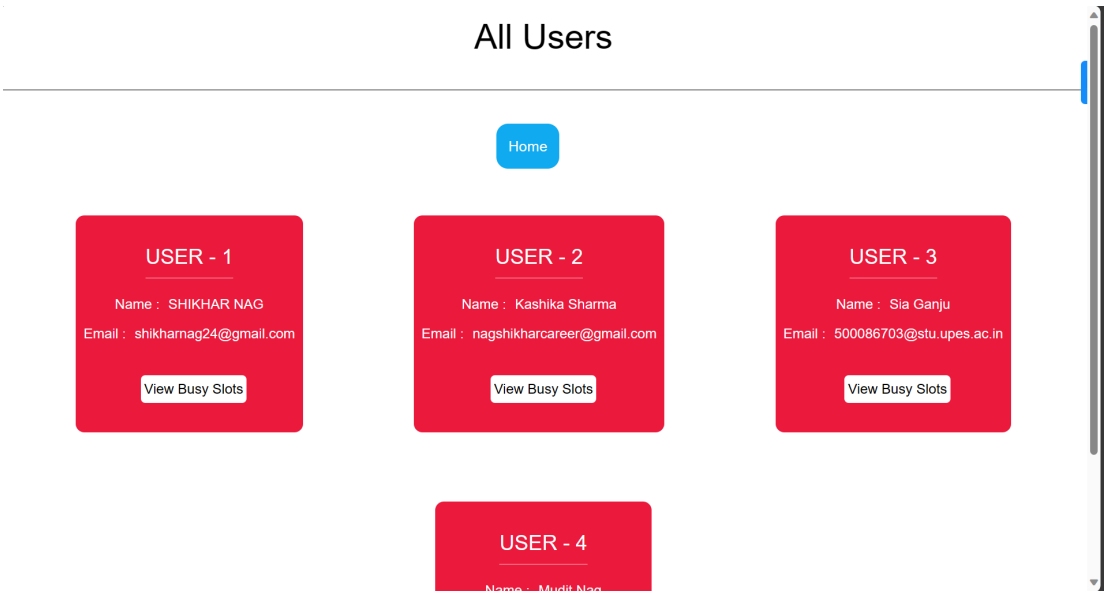
```
You, 13 hours ago | 1 author (You)
# Use an official Node.js runtime as a parent image
FROM node:14

# Set the working directory in the container
WORKDIR /home/index

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install application dependencies
RUN npm install

# Bundle app source
COPY . .

# Expose the port the app runs on       You, 22 hours ago • commit
EXPOSE 3000

# Define the command to run your application
CMD ["node", "index.js"]
```

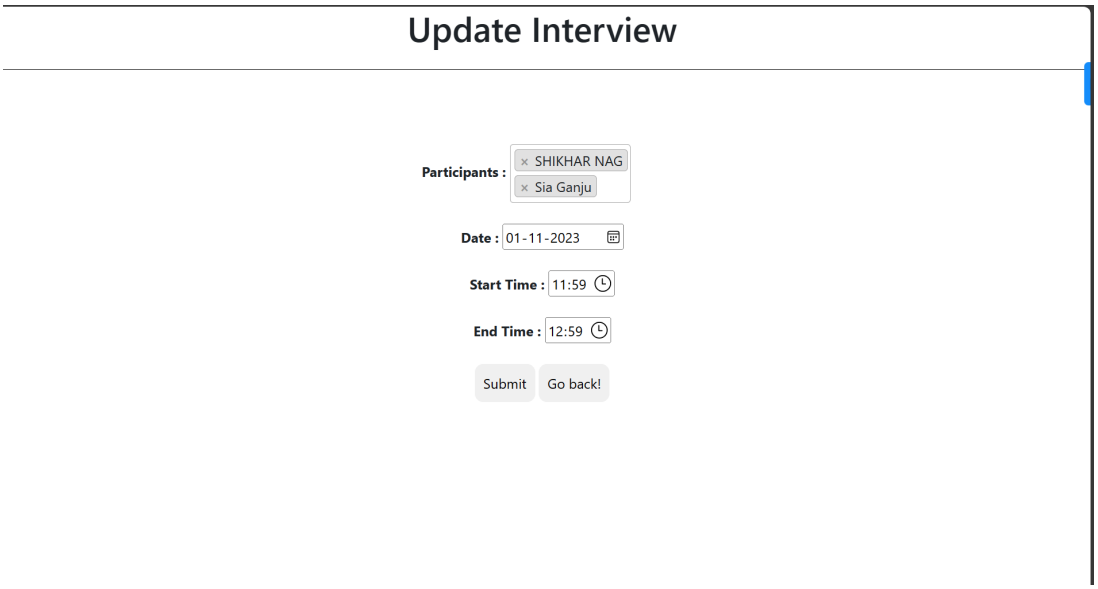
Output Screenshots:

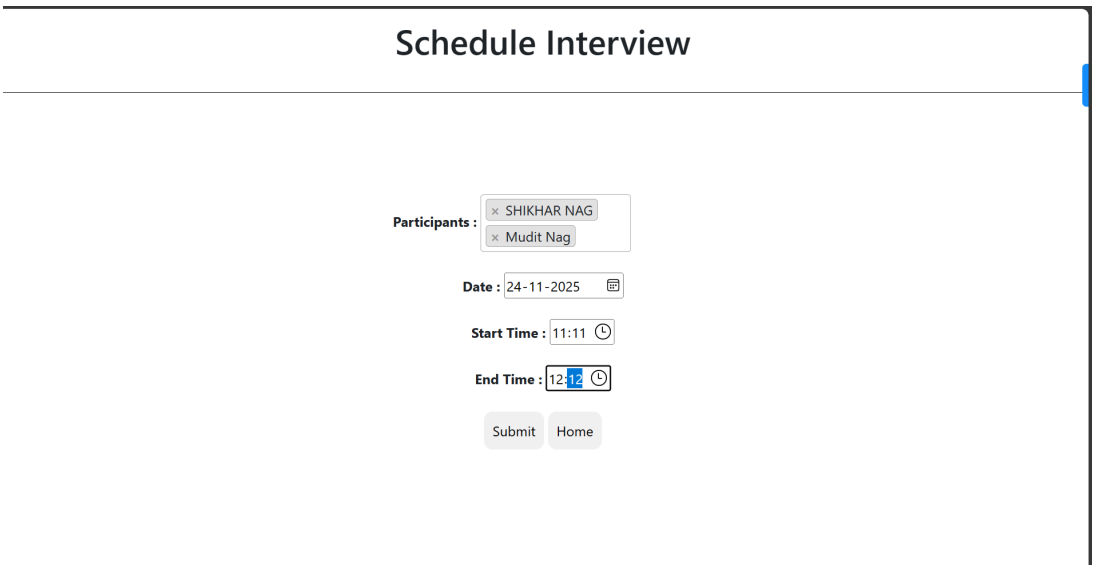


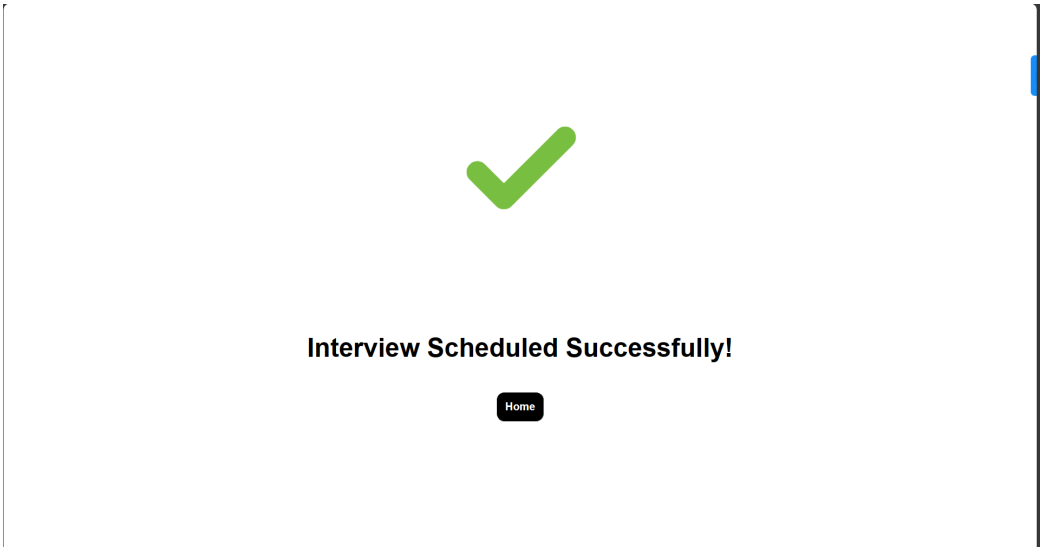


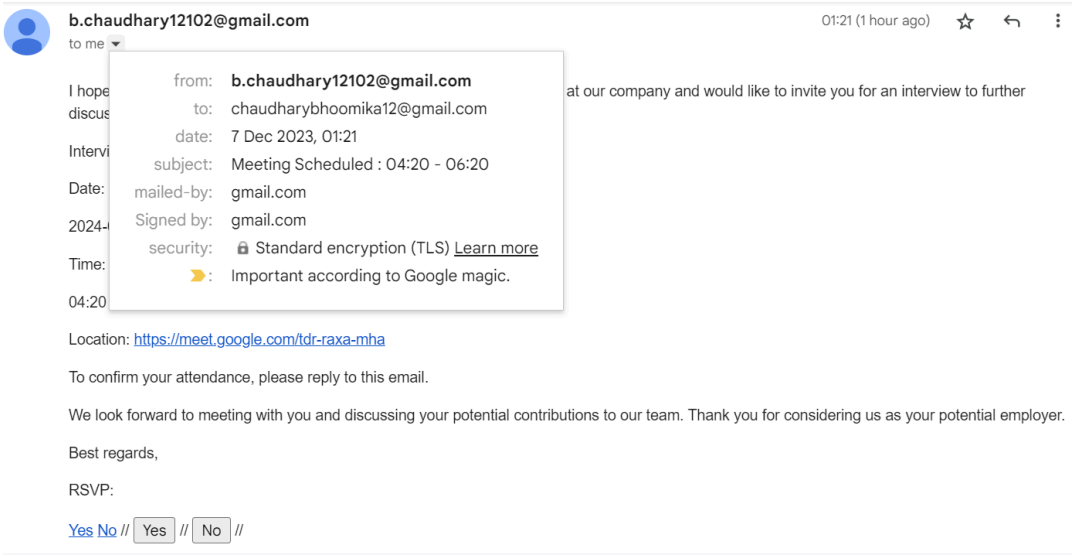



All Users
Home
USER - 1
Name : SHIKHAR NAG
Email : shikharnag24@gmail.com
View Busy Slots
USER - 2
Name : Kashika Sharma
Email : nagshikharcareer@gmail.com
View Busy Slots
USER - 3
Name : Sia Ganju
Email : 500086703@stu.upes.ac.in
View Busy Slots
USER - 4

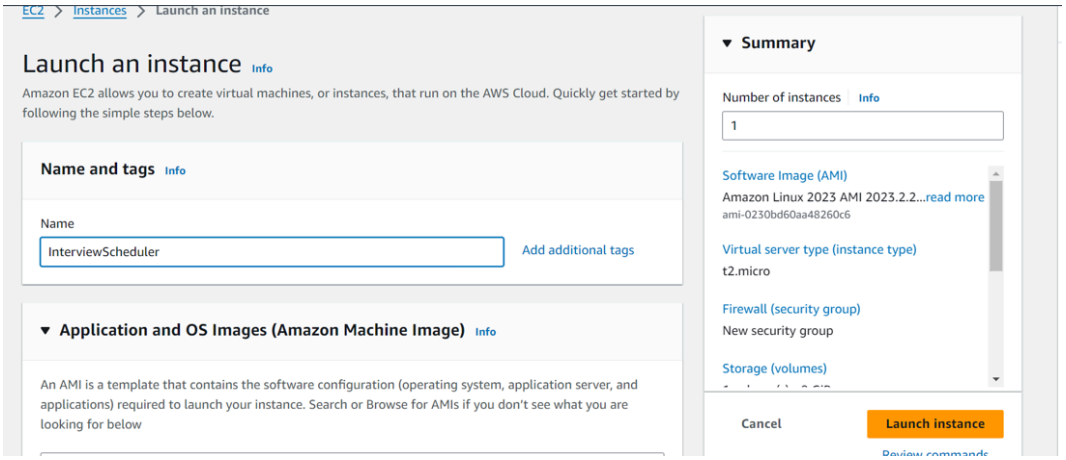

Update Interview
Participants :
× SHIKHAR NAG
× Sia Ganju
Date : 01-11-2023
Start Time : 11:59
End Time : 12:59
Submit
Go back!

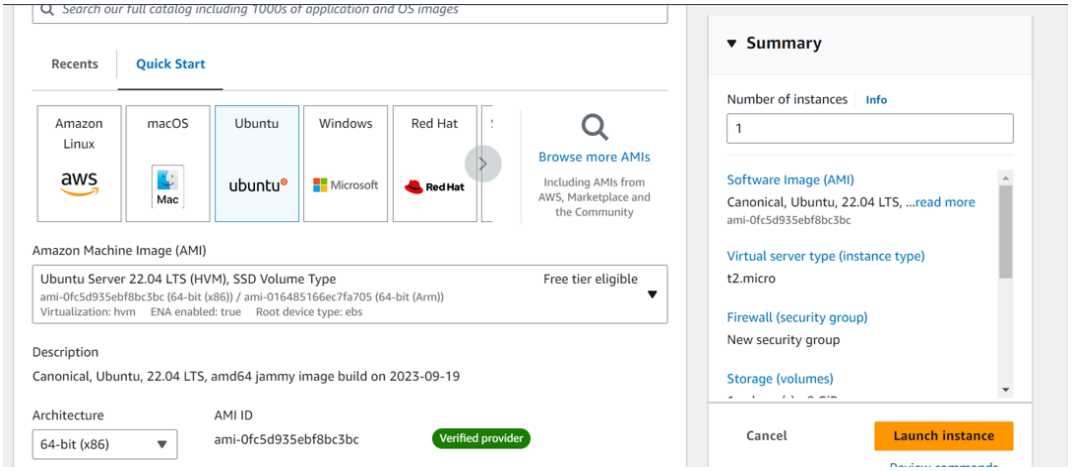

Schedule Interview
Participants :
× SHIKHAR NAG
× Mudit Nag
Date : 24-11-2025
Start Time : 11:11
End Time : 12:12
Submit
Home

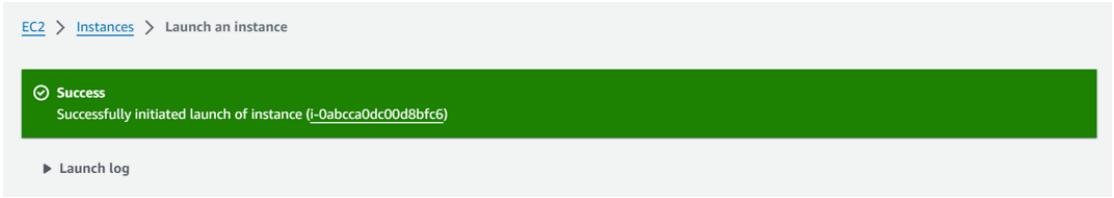

Interview Scheduled Successfully!
Home

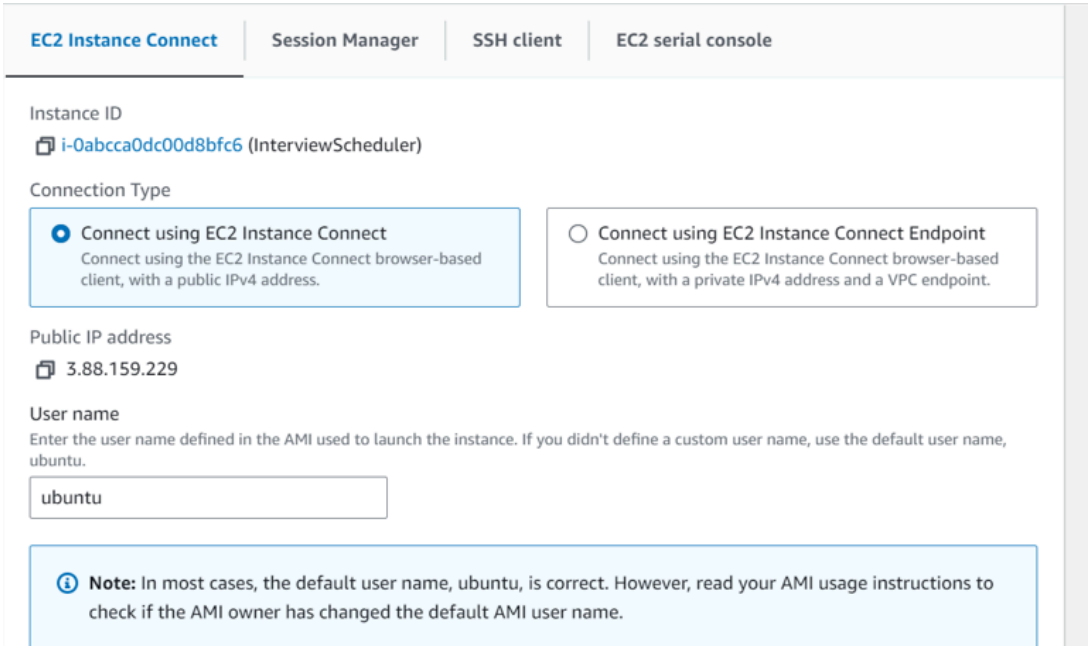

b.chaudhary12102@gmail.com
01:21 (1 hour ago)
to me
from: b.chaudhary12102@gmail.com
to: chaudharybhoomika12@gmail.com
date: 7 Dec 2023, 01:21
subject: Meeting Scheduled : 04:20 - 06:20
mailed-by: gmail.com
Signed by: gmail.com
security: Standard encryption (TLS) Learn more
Important according to Google magic.
at our company and would like to invite you for an interview to further
Location: https://meet.google.com/tdr-raxa-mha
To confirm your attendance, please reply to this email.
We look forward to meeting with you and discussing your potential contributions to our team. Thank you for considering us as your potential employer.
Best regards,
RSVP:
Yes No // Yes // No //

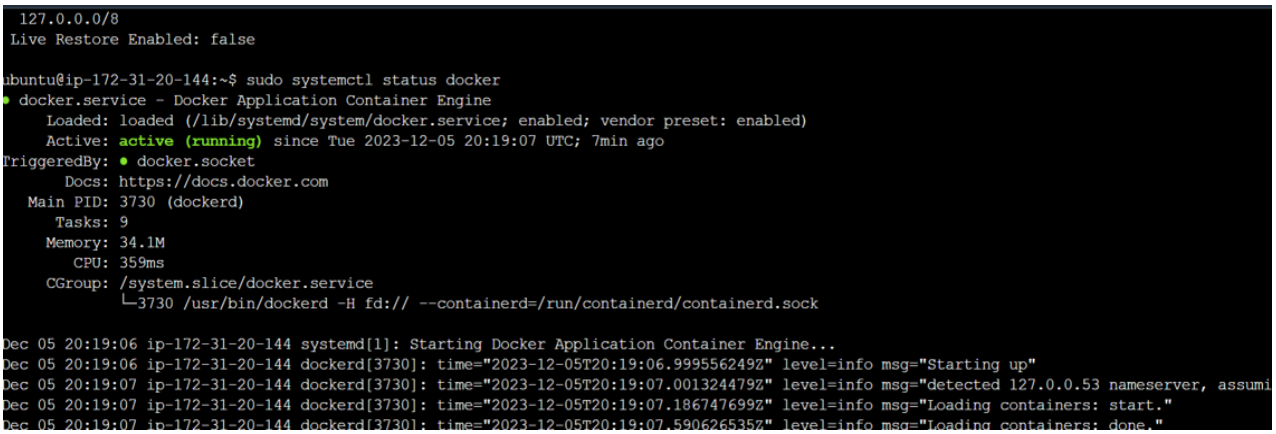
Deployment


EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
InterviewScheduler
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.2.2...read more
ami-0230bd60aa48260c6
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
Cancel
Launch instance


Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
Free tier eligible
ami-0fc5d935ebf8bc3bc (64-bit (x86)) / ami-016485166ec7fa705 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-09-19
Architecture
64-bit (x86)
AMI ID
ami-0fc5d935ebf8bc3bc
Verified provider
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-0fc5d935ebf8bc3bc
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
Cancel
Launch instance

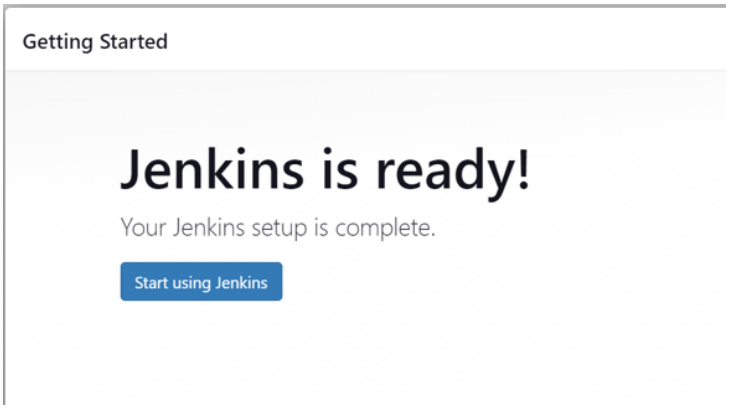

EC2 > Instances > Launch an instance
Success
Successfully initiated launch of instance (i-0abcca0dc00d8bfc6)
Launch log

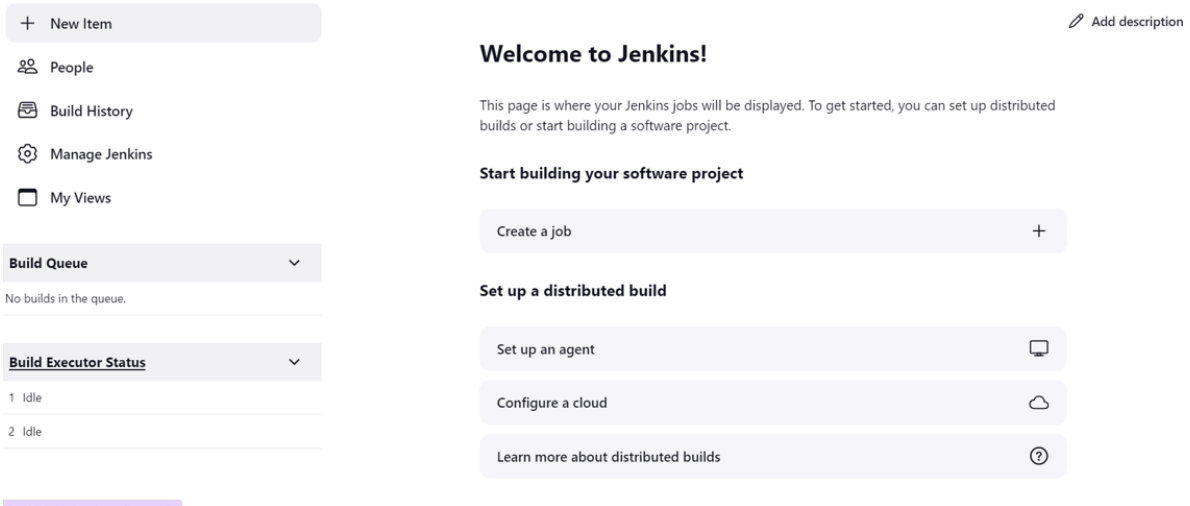

EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0abcca0dc00d8bfc6 (InterviewScheduler)
Connection Type
Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.
Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
Public IP address
3.88.159.229
User name
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ubuntu.
ubuntu
Note: In most cases, the default user name, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

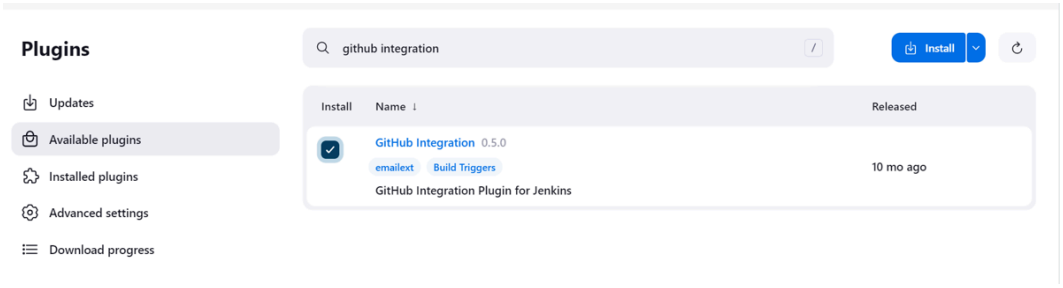

127.0.0.0/8
Live Restore Enabled: false
ubuntu@ip-172-31-20-144:~$ sudo systemctl status docker
docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
Active: active (running) since Tue 2023-12-05 20:19:07 UTC; 7min ago
TriggeredBy: docker.socket
Docs: https://docs.docker.com
Main PID: 3730 (dockerd)
Tasks: 9
Memory: 34.1M
CPU: 359ms
CGroup: /system.slice/docker.service
└─3730 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
Dec 05 20:19:06 ip-172-31-20-144 systemd[1]: Starting Docker Application Container Engine...
Dec 05 20:19:06 ip-172-31-20-144 dockerd[3730]: time="2023-12-05T20:19:06.999556249Z" level=info msg="Starting up"
Dec 05 20:19:07 ip-172-31-20-144 dockerd[3730]: time="2023-12-05T20:19:07.001324479Z" level=info msg="detected 127.0.0.53 nameserver, assumi
Dec 05 20:19:07 ip-172-31-20-144 dockerd[3730]: time="2023-12-05T20:19:07.186747699Z" level=info msg="Loading containers: start."
Dec 05 20:19:07 ip-172-31-20-144 dockerd[3730]: time="2023-12-05T20:19:07.590626535Z" level=info msg="Loading containers: done."

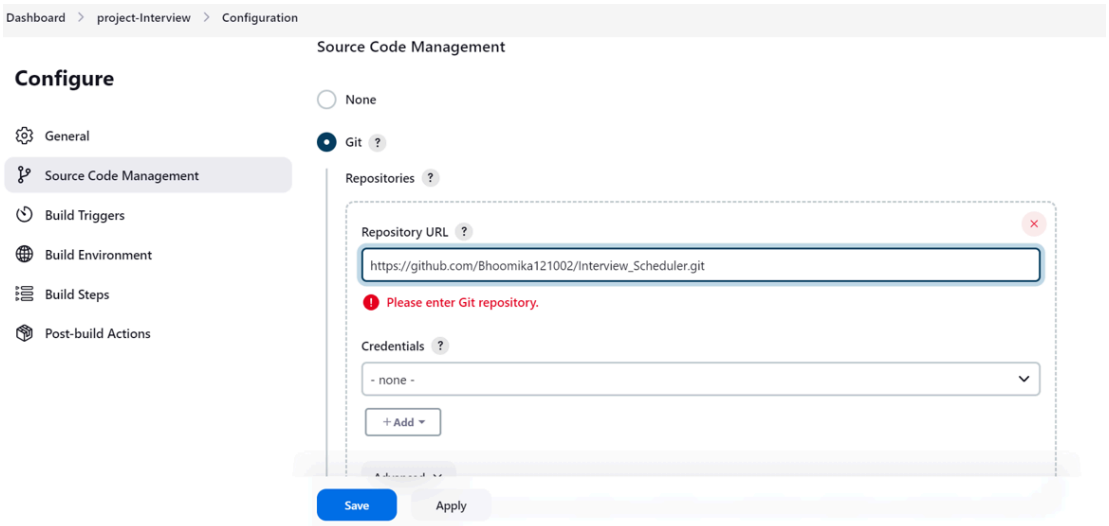
```
ubuntu@ip-172-31-20-144:~$ java -version
openjdk version "11.0.21" 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-20-144:~$ nano jenkins.sh
ubuntu@ip-172-31-20-144:~$ sh jenkins.sh
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
```

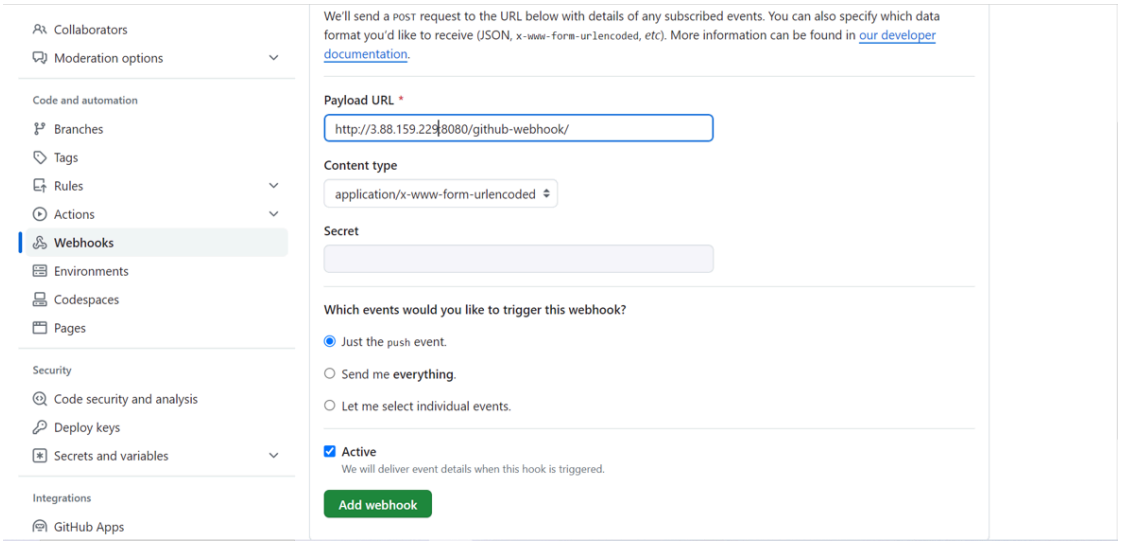
```
jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-12-05 20:40:08 UTC; 4min 59s ago
   Main PID: 21213 (java)
      Tasks: 39 (limit: 1121)
     Memory: 306.5M
        CPU: 39.484s
     CGroup: /system.slice/jenkins.service
             └─21213 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

ec 05 20:39:39 ip-172-31-20-144 jenkins[21213]: b777336b402d4fcf99f37a6a28f1f509
ec 05 20:39:39 ip-172-31-20-144 jenkins[21213]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
ec 05 20:39:39 ip-172-31-20-144 jenkins[21213]: ************************************************************
ec 05 20:39:39 ip-172-31-20-144 jenkins[21213]: ************************************************************
ec 05 20:39:39 ip-172-31-20-144 jenkins[21213]: ************************************************************
ec 05 20:40:08 ip-172-31-20-144 jenkins[21213]: 2023-12-05 20:40:08.505+0000 [id=32]        INFO        jenkins.InitReactorRunner$1#onAttained
ec 05 20:40:08 ip-172-31-20-144 jenkins[21213]: 2023-12-05 20:40:08.532+0000 [id=24]        INFO        hudson.lifecycle.Lifecycle#onReady: Je
ec 05 20:40:08 ip-172-31-20-144 systemd[1]: Started Jenkins Continuous Integration Server.
ec 05 20:40:08 ip-172-31-20-144 jenkins[21213]: 2023-12-05 20:40:08.686+0000 [id=47]        INFO        h.m.DownloadService$Downloadable#load:
ec 05 20:40:08 ip-172-31-20-144 jenkins[21213]: 2023-12-05 20:40:08.687+0000 [id=47]        INFO        hudson.util.Retrier#start: Performed t
```

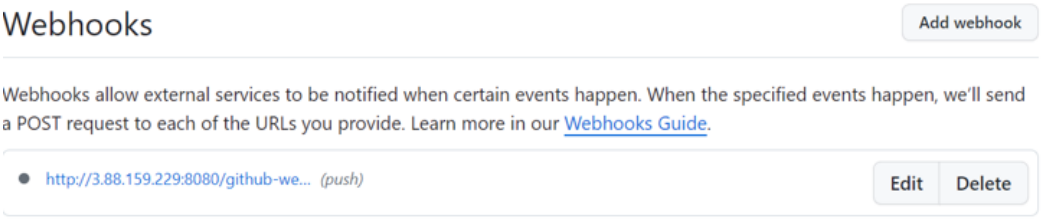





Plugins
github integration
Install
Updates
Available plugins
Installed plugins
Advanced settings
Download progress
Install
Name
Released
GitHub Integration 0.5.0
emailext
Build Triggers
10 mo ago
GitHub Integration Plugin for Jenkins


Dashboard
project-Interview
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/Bhoomika121002/Interview_Scheduler.git
Please enter Git repository.
Credentials
- none -
+ Add
Save
Apply


Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security and analysis
Deploy keys
Secrets and variables
Integrations
GitHub Apps
We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.
Payload URL *
http://3.88.159.229:8080/github-webhook/
Content type
application/x-www-form-urlencoded
Secret
Which events would you like to trigger this webhook?
Just the push event.
Send me everything.
Let me select individual events.
Active
We will deliver event details when this hook is triggered.
Add webhook


Webhooks
Add webhook
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
http://3.88.159.229:8080/github-we... (push)
Edit
Delete

```
ubuntu@ip-172-31-20-144:~$ cd ~/.ssh
ubuntu@ip-172-31-20-144:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-20-144:~/.ssh$ ssh-keygen -o
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:LTFOLQ82wc3Cs3HL/NSoQ75zdcc0b0dMO8etkcd45g8 ubuntu@ip-172-31-20-144
The key's randomart image is:
+---[RSA 3072]----+
|        o.o      |
|         *o+     .|
|         OO.. o Bo|
|        +.O= o =+%|
|         Soo+   @=|
|           .+ . E O|
```

```
ubuntu@ip-172-31-20-144:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCgUnEBYr+04+eBiZhtOnUHojyGEaTKCuTvCFIc64KEbRXHM/aWculINUyVQuT/dIrlbsex2/NDG7/pctu9CRB7Kk+OrcAl2oO
BUPwBEDxM4ISEe9cWfL5ddtEbT00VbajV9/LkXr4Ca7B4zunZktUFHn3zulbUHnlC+QvwuW4Yb+QwxnlvOBg221QyHRZMHM19BQzkhi+8Ucx73Vbp98bqc3iZ6AyJ2FQ2VFeORp
xJzcMQ96jE6wUf9OMjWZ6r9nEKViHGr9wYzuazIlp9reQyWVaMAumCcHgbTC2PSsfZ53Jzhzjh5TnWyh9UGvS9l3u6FPn04q/PcXNh7vQ3+5hjgfakfL9Ly2heQlGL0q6VxVKVWh
Iqh8LYahPL861CE9ZbljA2cmK62JO4n0c/rh5EotMefZ+HwMsCX8HAQ/WqJtn7qraLdZG0guN3r7ubfE48V+3NqzTIE2erThOLVcGXoJUnDCiQRgj5tuK4lE01RA+mi0= ubuntu
72-31-20-144
```

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

projectshkey
SHA256:ydT0/59FwK8PZkK/oA6lLLG04V7iWBDtBuxyHwFXkcs
SSH
Added on Jun 16, 2023
Never used — Read/write
Delete

projectIntersshkey
SHA256:LTFOLQ82wc3Cs3HL/NSoQ75zdcc0b0dMO8etkcd45g8
SSH
Added on Dec 6, 2023
Never used — Read/write
Delete

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind
Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

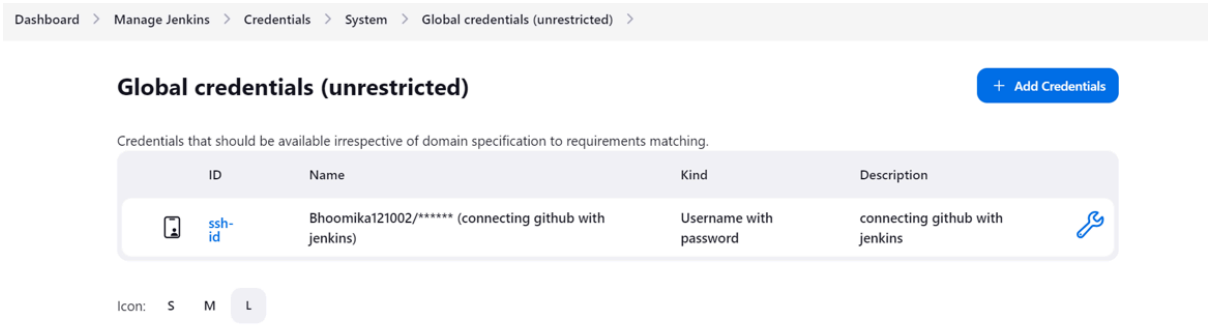
Username ?
Bhoomika121002

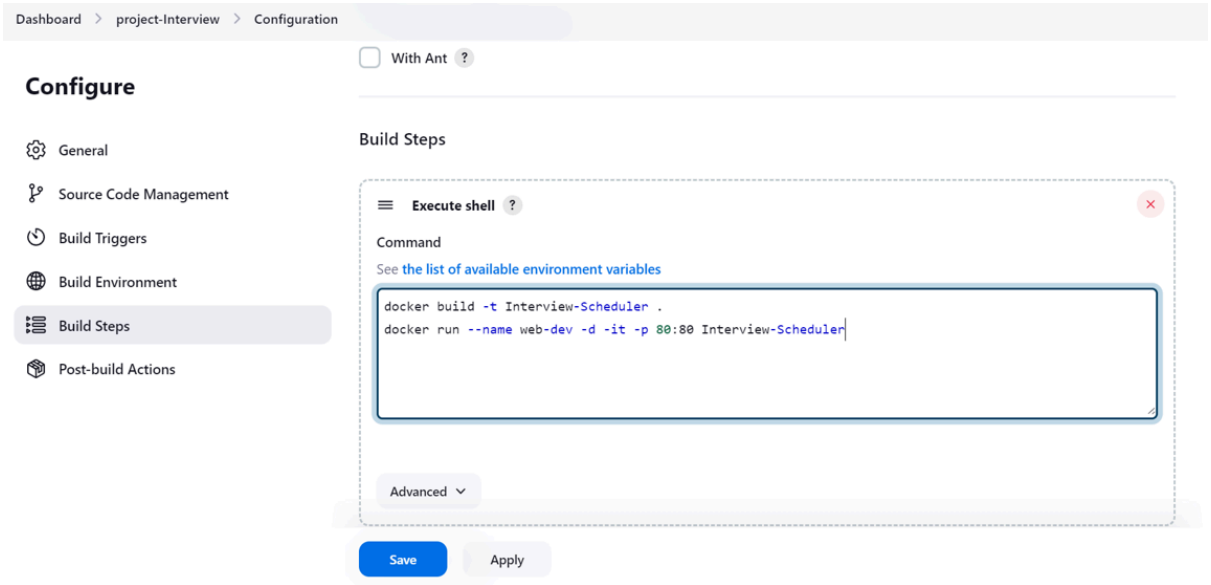
Treat username as secret ?

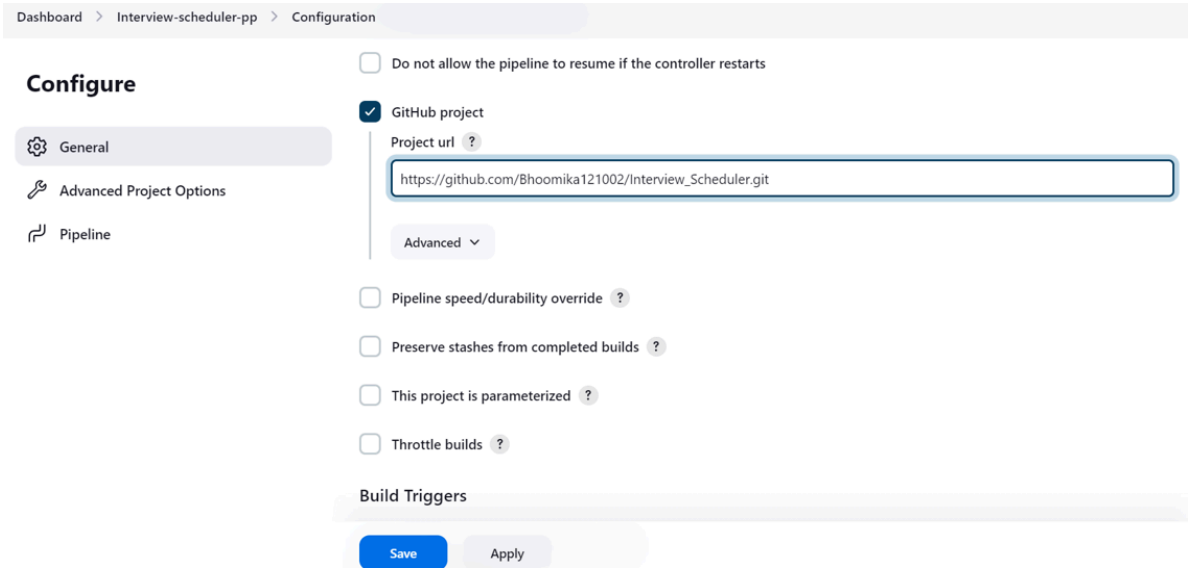
Password ?
••••••••••

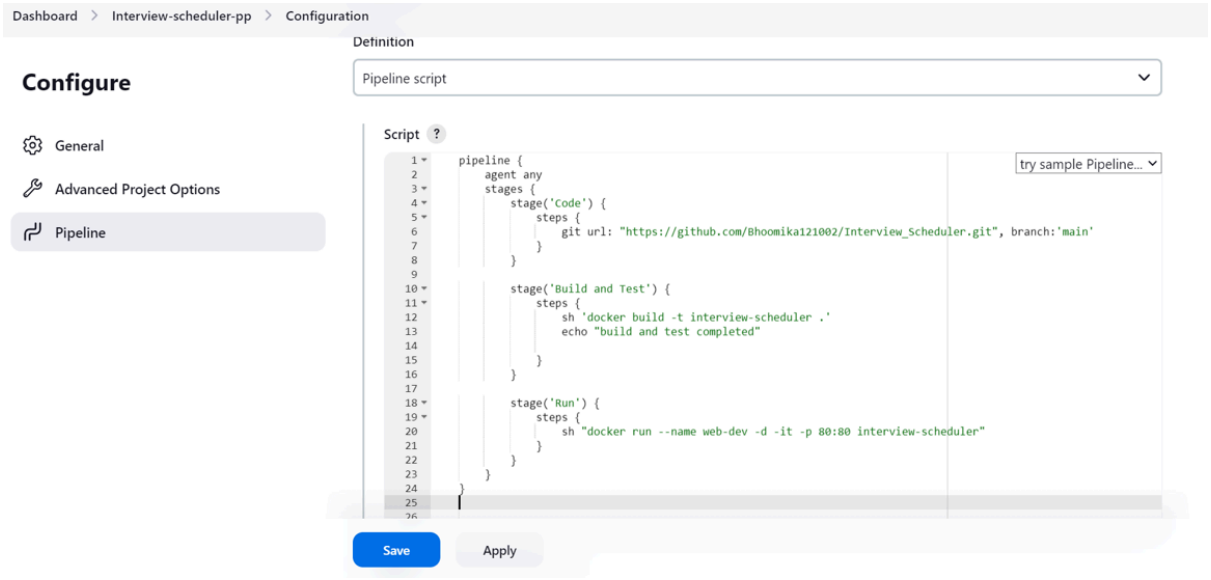
ID ?

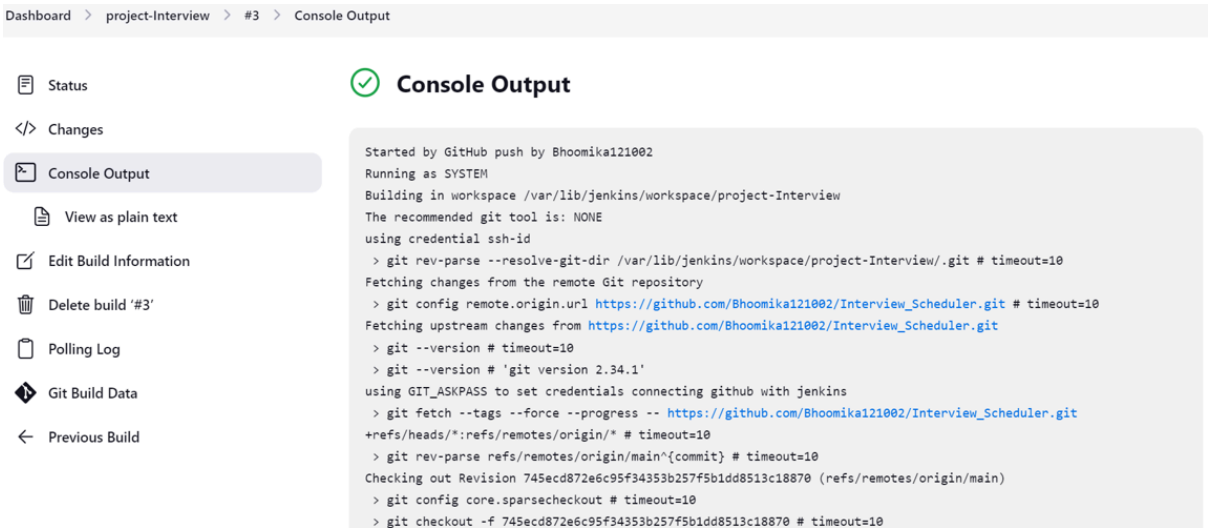
Create

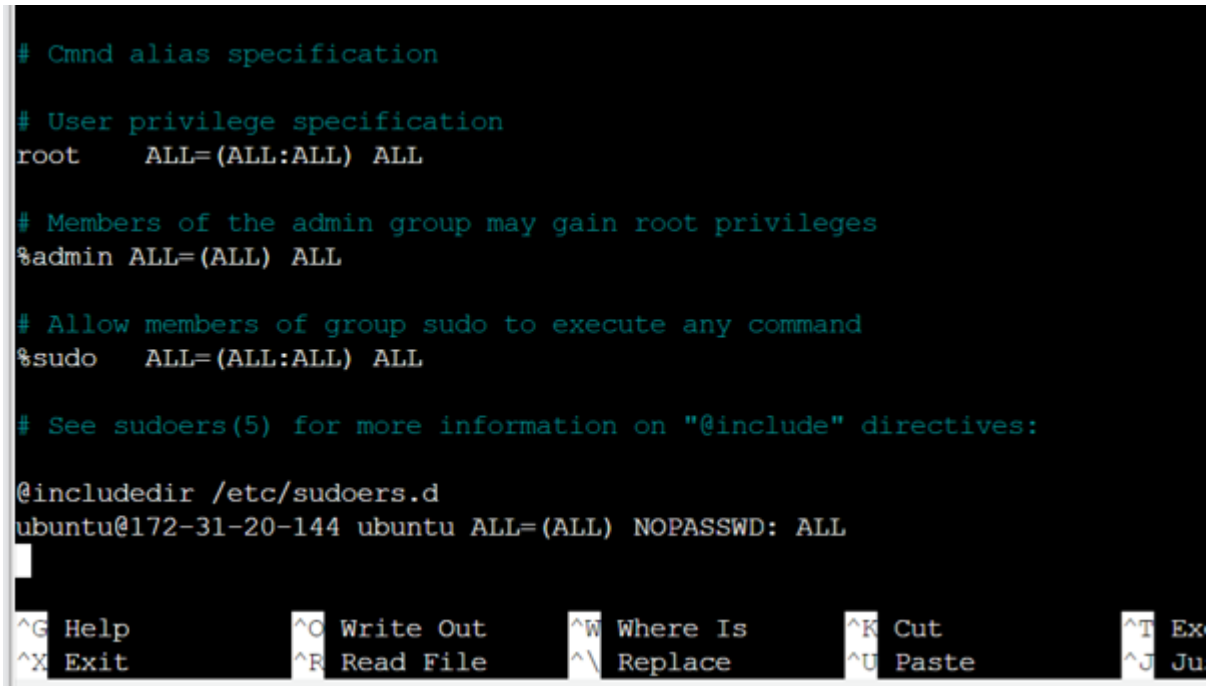

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
ssh-id
Bhoomika121002/****** (connecting github with jenkins)
Username with password
connecting github with jenkins
Icon: S M L

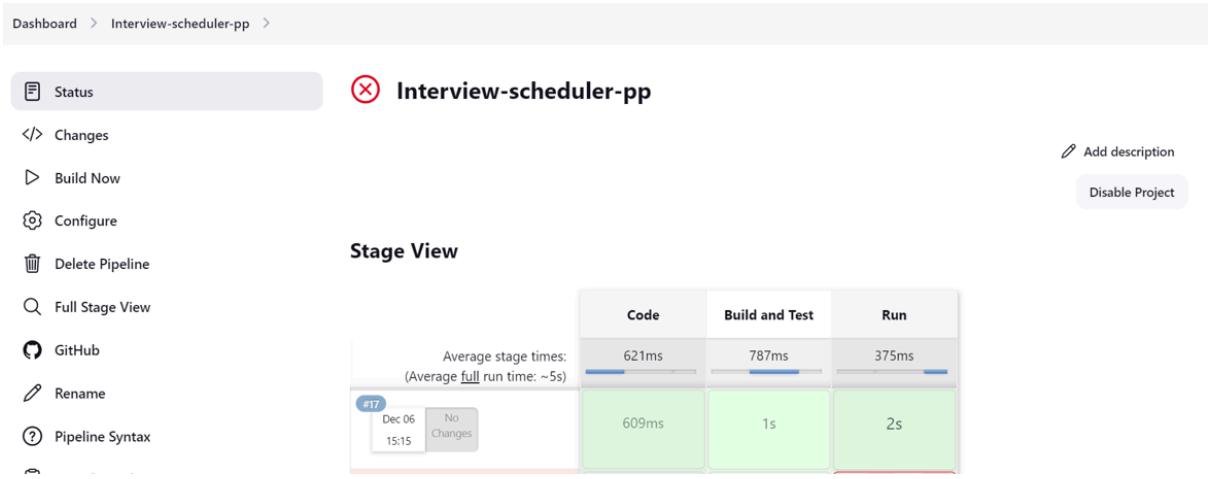

Dashboard > project-Interview > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
docker build -t Interview-Scheduler .
docker run --name web-dev -d -it -p 80:80 Interview-Scheduler
Advanced
Save
Apply

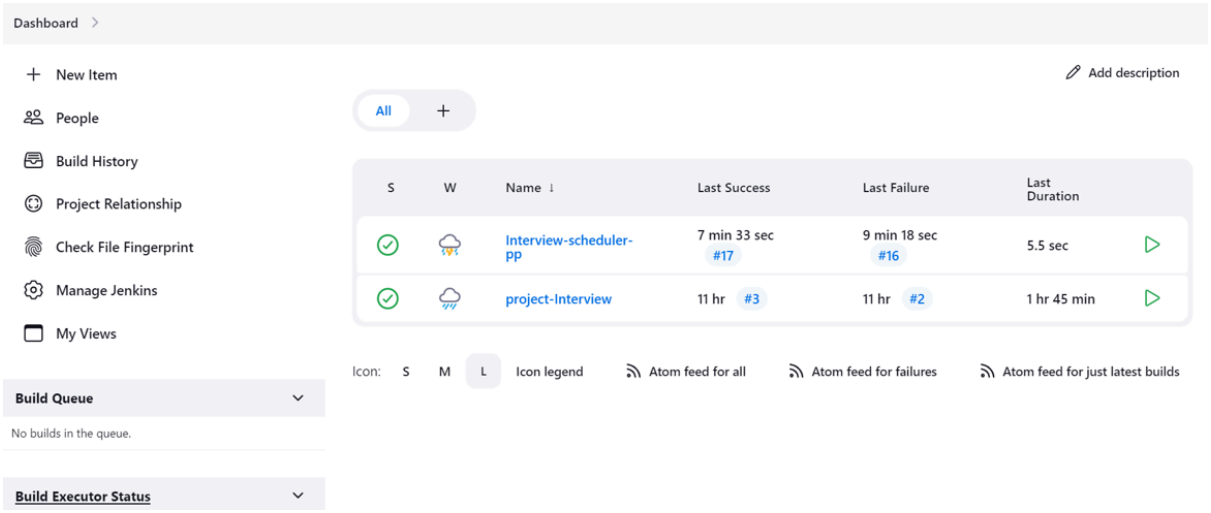

Dashboard > Interview-scheduler-pp > Configuration
Configure
General
Advanced Project Options
Pipeline
Do not allow the pipeline to resume if the controller restarts
GitHub project
Project url
https://github.com/Bhoomika121002/Interview_Scheduler.git
Advanced
Pipeline speed/durability override
Preserve stashes from completed builds
This project is parameterized
Throttle builds
Build Triggers
Save
Apply


Dashboard > Interview-scheduler-pp > Configuration
Configure
General
Advanced Project Options
Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
pipeline {
agent any
stages {
stage('Code') {
steps {
git url: "https://github.com/Bhoomika121002/Interview_Scheduler.git", branch:'main'
}
}
stage('Build and Test') {
steps {
sh 'docker build -t interview-scheduler .'
echo "build and test completed"
}
}
stage('Run') {
steps {
sh "docker run --name web-dev -d -it -p 80:80 interview-scheduler"
}
}
}
}
Save
Apply

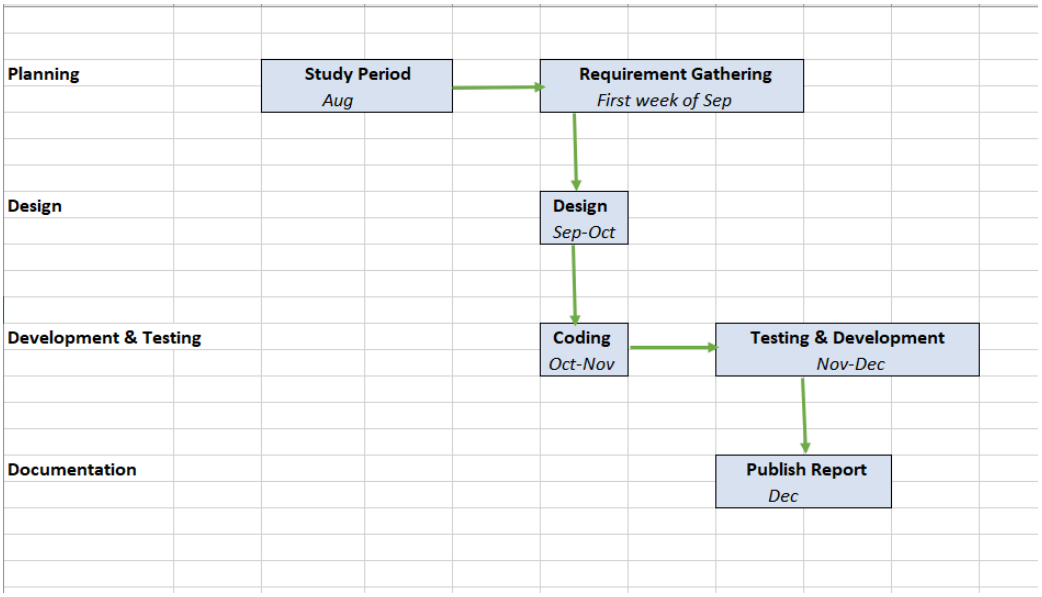

Dashboard > project-Interview > #3 > Console Output
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#3'
Polling Log
Git Build Data
Previous Build
Console Output
Started by GitHub push by Bhoomika121002
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/project-Interview
The recommended git tool is: NONE
using credential ssh-id
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/project-Interview/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Bhoomika121002/Interview_Scheduler.git # timeout=10
Fetching upstream changes from https://github.com/Bhoomika121002/Interview_Scheduler.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_ASKPASS to set credentials connecting github with jenkins
> git fetch --tags --force --progress -- https://github.com/Bhoomika121002/Interview_Scheduler.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 745ecd872e6c95f34353b257f5b1dd8513c18870 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 745ecd872e6c95f34353b257f5b1dd8513c18870 # timeout=10


Cmnd alias specification
User privilege specification
root ALL=(ALL:ALL) ALL
Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d
ubuntu@172-31-20-144 ubuntu ALL=(ALL) NOPASSWD: ALL
^G Help ^O Write Out ^W Where Is ^K Cut ^T Ex
^X Exit ^R Read File ^\ Replace ^U Paste ^J Ju


Dashboard
Interview-scheduler-pp
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
GitHub
Rename
Pipeline Syntax
Interview-scheduler-pp
Add description
Disable Project
Stage View
Code
Build and Test
Run
Average stage times:
(Average full run time: ~5s)
621ms
787ms
375ms
#17
Dec 06
15:15
No Changes
609ms
1s
2s


Dashboard
New Item
People
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
Interview-scheduler-pp
7 min 33 sec #17
9 min 18 sec #16
5.5 sec
project-Interview
11 hr #3
11 hr #2
1 hr 45 min
Icon: S M L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds
Build Queue
No builds in the queue.
Build Executor Status

9. Pert Chart


Planning
Study Period
Aug
Requirement Gathering
First week of Sep
Design
Design
Sep-Oct
Development & Testing
Coding
Oct-Nov
Testing & Development
Nov-Dec
Documentation
Publish Report
Dec

10. Conclusion

- In this project, we will design and develop an application for interview scheduling.
- Users will have access to this application in order to view their assigned slots for the interview.
- It will be simpler for people to obtain the required details concerning their interview.
- It will save a lot of time, effort, and money as well as a large amount of paperwork for the interviewer as well as the learner.

Technological advancements have caught up with most industries and the smaller tasks they conduct. The scheduling and management of interviews is no exception.

Proven online scheduling software is now readily available to all-sized organizations and for all scheduling needs, regardless of the scope of the job, the number of staff members, and their budgets. This technology can transform the daunting process of scheduling and enable them to run more efficiently and profitably.

11. References

1. Thalawattha, Sathsara & Vidanagama, Dushyanthi. (2021). A Survey on Web-based Meeting Scheduling Application.
2. Perera, Poornima & Vidanagama, Dushyanthi. (2020). A WEB-BASED PAPERLESS MEETING MANAGEMENT SYSTEM.
3. Erik Timmerman, C., Shik Choi, C., 2017. Meeting Technologies. https://doi.org/10.1002/9781118955567.wbieoc132
4. Tran, Linh & Stojcevski, Alex & Pham, Thanh & Souza-Daw, Tony & Nguyen, Nhan & Nguyen, Vinh & Nguyen, Chau. (2016). A smart meeting room scheduling and management system with utilization control and ad-hoc support based on real-time occupancy detection. 186-191. 10.1109/CCE.2016.7562634.