

# **CS 816 - Software Production Engineering**

**Major Project - College Event Management**

**Submitted By:**

**Shreyank Buddhadev (MT2021130)**  
**Meet Patel (MT2021076)**

**Index:**

i.	Abstract	2
1	Introduction	3
2	What and Why DevOps	4
3	System Configuration	5
4	Software Development Life Cycle	6
5	Result & Discussion	20
6	Scope of Future work	25
7	Conclusion	25

## **Abstract:**

Any large organization has many events scheduled, for a student / management it becomes really hard to keep track of all ongoing events. Sometimes it happens that a student is interested in some specific domain events but he/she might never know regarding that event.

## **Users of the app:**

### **Users:**

1. Students
2. Event Organizer
3. Admin / Management committee

### **Links:**

Github: [codebase](#)

Dockerhub: [server](#), [client](#)

API Docs: [Docs](#)

Functional Req: [Sheet](#)

## **Introduction:**

### **Users and Features of the app:**

#### **Users:**

1. Students
2. Event Organizer
3. Admin / Management committee

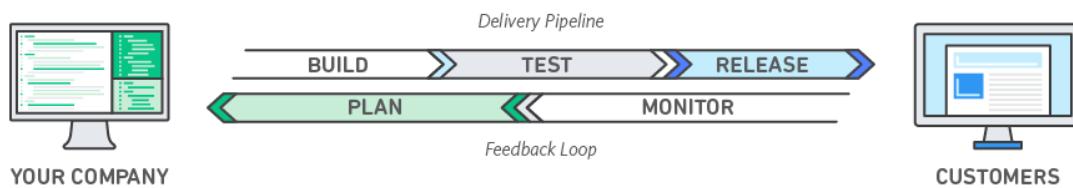
#### **Features:**

- **For Students:**
  - A student can log in and see all the ongoing / upcoming events, In which he/she can participate.
  - Students can filter events using tags such as *research talk, debate, seminar, technical event, cultural event etc.*
  - Students can directly register from the website itself for the event.
  - *Competition result*
  - Can download certificates
- **For Event Organizers:**
  - An Event organizer can raise a request to **Admin**, to organize event[s]. {A request can contain venue, cost, expected participation, students allowed to participate}. After successful registration event will be shown on the dashboard of students which are allowed to participate.
- **Admin / Management committee:**
  - Admin can view all the ongoing event[s] details on its dashboard.
  - Admin can approve / disapprove any event request from organizers.

## What and Why DevOps?

### **What is DevOps?**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team's velocity.

### **Why DevOps?**

- Speed
- Rapid Delivery
- Reliability
- Scale
- Improved Collaboration
- Security

## **System Configuration:**

- **Operating System**
  - Ubuntu 20.04 / Windows 10/11 / MacOS 12.2.1
- **CPU and RAM**
  - 4 core + Min 2GB Required
- **Language**
  - Frontend: ReactJS, materialUI
  - Backend: NodeJS(version 16.0.0), express, mongoDB
- **Database**
  - mongoDB cloud
  - MongoDB Compass (To access the cloud DB locally in the system)
- **Building tools**
  - npm to build react project
- **DevOps Tools**
  - SCM(Source Control Management) : git + github
  - Testing: Jest + Supertest
  - Build: npm
  - Continuous Integration - Jenkins
  - Containerization - Docker
  - Continuous deployment - Ansible
  - Monitoring - ELK Stack (Elastic Search, Logstash, Kibana)

## Software Development Life Cycle:

- **Installation:**

- **ReactJS Framework**

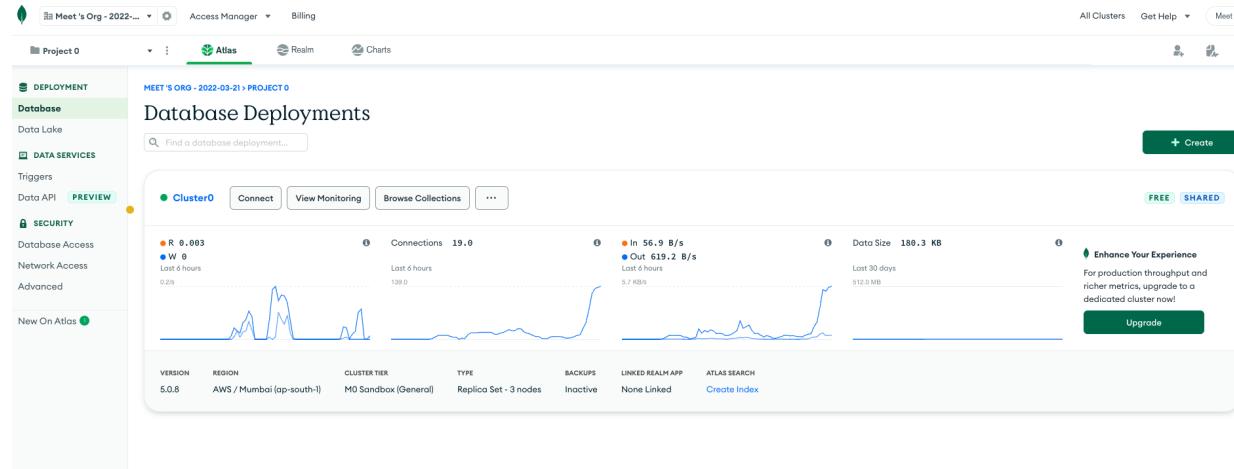
- React is one of the most popular JavaScript libraries in the web development field today. First, we need NPM to run the react app.
    - We can download the node from its official website <https://nodejs.org/en/> which also includes the package NPM. [Note: Recommended nodeJS version 16]
    - Frequently used commands:
      - `npm start`
      - `npm i <package-name>`
      - `npm run build`
      - `npm run test`

- **Node js Framework**

- As explained in above step install NodeJS.

- **MongoDB :**

- To begin the sign-up process, navigate to the [MongoDB Atlas sign-up page](#).



- Above snapshot shows the MongoDB dashboard of the database.

The screenshot shows the MongoDB Compass interface. At the top, it displays "Cluster0" and "VERSION 5.0.8 REGION AWS Mumbai (ap-south-1)". Below the header, there are tabs for Overview, Real Time, Metrics, Collections (which is selected), Search, Profiler, Performance Advisor, Online Archive, and Cmd Line Tools. A button to "VISUALIZE YOUR DATA" is also present.

In the main area, it shows "DATABASES: 1 COLLECTIONS: 4". The "project" collection is expanded, showing "event-participation", "events-approved", "events-requested", and "user-data". The "user-data" collection has 5 documents and a total size of 72KB. The document details are as follows:

```

_id: ObjectId("5c6d5f1056a6a39a6fab5e")
name: "Shreyank Student"
email: "shreyank@stu.com"
password: "$2a$10$WPMVjJ50L0K0aN2G4juFOA61dn40fP8phaCvxotY4duUfKAGTku"
phono: "9898959884"
role: "Student"
course: "M.Tech"
> eventsp: Array
__v: 0

```

Another document is partially visible below it:

```

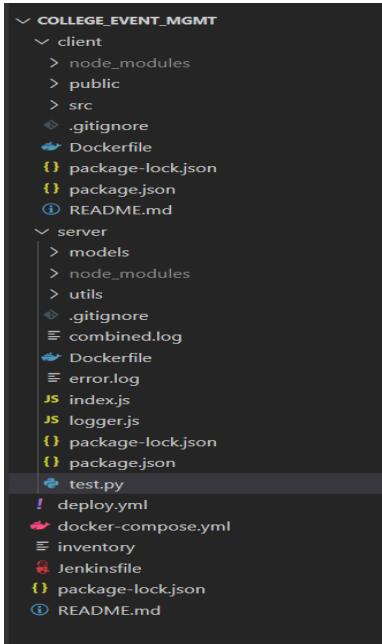
_id: ObjectId("626d6f2df956a6a39a6fab60")
name: "Shreyank Event"
email: "Shreyank@stu.com"
password: "1234567890"
phono: "9898989808"
role: "EventManager"
course: "M.Tech"
> eventsp: Array
__v: 0

```

- Above snapshot shows the different tables present in the database and collection's in it.

## ● Code Structure:

- Below, picture depicts code structure of the project
- *Client* folder contains front end part of the project. *Server* folder contains backend part of the project.



- **Pipeline Setup:**

- Jenkins is an open-source Continuous Integration server for running a chain of actions to achieve Continuous Integration in an automated fashion. Jenkins supports the complete SDLC from building, testing, deploying, and other stages of the SDLC.
- To start Jenkins from the terminal we run the following command -
  - `sudo systemctl start jenkins`
- and then we can run it on `http://localhost:8080/`

The screenshot shows the Jenkins dashboard at `http://192.168.31.128:8080`. On the left sidebar, there are links for New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, and New View. Below this is a 'Build Queue' section which is currently empty. The main area displays a table of build jobs:

All	Name	Last Success	Last Failure	Last Duration
<span style="color: green;">✓</span>	CPU Info	2 mo 19 days - #12	2 mo 21 days - #7	0.1 sec
<span style="color: green;">✓</span>	DevOps-Calci	6 days 4 hr - #27	19 days - #19	1 min 21 sec
<span style="color: green;">✓</span>	DevOps-Calculator	19 days - #26	19 days - #24	1 min 21 sec
<span style="color: green;">✓</span>	Event-Management	1 hr 24 min - #27	1 hr 33 min - #26	9 min 54 sec
<span style="color: red;">✗</span>	git-pollscm	N/A	N/A	N/A
<span style="color: red;">✗</span>	GitJenkins	N/A	1 mo 7 days - #4	0.49 sec
<span style="color: green;">✓</span>	My First Build	2 mo 22 days - #1	N/A	0.12 sec
<span style="color: blue;">...</span>	pipeline test	N/A	N/A	N/A

- Now we add plugins by going to `Manage Jenkins → Manage Plugins`  
Then adding Docker, Git, and Ansible.

The screenshot shows the 'Manage Jenkins' page at `http://192.168.31.128:8080/manage`. The left sidebar includes links for New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins (which is selected), My Views, Lockable Resources, and New View. The main content area displays a message about a new Jenkins version (2.332.2) available for download. It also lists several warning messages for installed components:

- Pipeline: Multibranch 706.vd43c6dec013:**
  - OS command execution vulnerabilities in Pipeline-related plugins
  - Vulnerabilities in multiple Pipeline-related plugins allow reading arbitrary files on the controller
- Pipeline: Groovy 2648.va9433432b33c:**
  - OS command execution vulnerabilities in Pipeline-related plugins
  - Vulnerabilities in multiple Pipeline-related plugins allow reading arbitrary files on the controller
  - Sensitive information disclosure
- Credentials Plugin 1074.v60e6c29b\_b\_44b\_:**
  - Stored XSS vulnerability
- Pipeline: Shared Groovy Libraries 552.vd9cc05b8a2ef:**
  - OS command execution vulnerabilities in Pipeline-related plugins
  - Untrusted users can modify some Pipeline libraries
  - Sandbox bypass vulnerability

- Now we link our Github and Docker account by going to Manage Jenkins → Manage Credentials → global → Add Credential  
Fill username and password for git and docker separately and then save it

The screenshot shows the Jenkins 'Credentials' page. On the left sidebar, there are various links like 'New Item', 'People', 'Build History', etc. The main area is titled 'Credentials' and contains a table with the following data:

T	P	Store	Domain	ID	Name
		Jenkins	(global)	ShreyankB	ShreyankB
		Jenkins	(global)	Shreyank	Shreyank
		Jenkins	(global)	Github	ShreyankB/*****
		Jenkins	(global)	DockerHub	shreyankb/*****

Below the table, there's a section titled 'Stores scoped to Jenkins' with a single entry:

P	Store	Domains
	Jenkins	\$(global)

- We can now create a pipeline by going on New Item → Enter name of Project → Pipeline Project → Save it

The screenshot shows the Jenkins 'Enter an item name' dialog. The input field contains 'Colle\_Event\_m'. Below the input field, there are several project types listed:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Configuration project**: Projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

At the bottom right of the dialog, there is an 'OK' button.

- For various initializations in Jenkins, CI Pipeline is used to describe it respectively.

### **Source Control Management:**

- SCM is the process of managing and tracking the changes in code. As software projects grow in lines of code and headcount of contributors, the costs of communication and management also grow. SCM is an important tool to reduce growing development costs.
- Following are some useful commands in Git -
  - **git clone <<https://name-of-the-repository-link>>**
    - - This will copy the entire data from the git url
  - **git branch <branch-name>**
    - - This command will make a new branch in the code repository.
  - **git status**
    - - This command displays the state of the working directory and the staging area.
  - **git add <file>**
    - This will add changes in the working directory to the staging area.
  - **git commit -m "commit message"**
    - - This command is used to save your changes to the local repository with -m used to give a small description that can help your teammates understand what happened.
  - **git push <remote> <branch-name>**
    - - This command pushes all the latest code to the repository.
  - **git pull <remote>**
    - - This command is used to update the local version of a repository from a remote.

```
stages {
    stage('Git clone') {
        steps {
            git url: 'https://github.com/meetcric/College_Event_MGMT.git', branch: 'master'
        }
    }
}
```

- Installing Dependency:

- **NPM:**
  - Build App is a full-stack JS App build system. Inspired from create-react-app by Facebook and other client build systems it is going one step further by providing similar facilities for full-stack Java Script development.

```
stage('Install dependency') {  
    steps {  
        sh 'cd ./client/ && npm i'  
        sh 'cd ./server/ && npm i'  
    }  
}
```

- Build:

- 

```
stage('Build') {  
    steps {  
        sh 'cd ./client/ && npm run build'  
    }  
}
```

- 

- Testing :

We have used **Jest** and **SuperTest** for the testing of our project.

**Jest** is a JavaScript testing framework with a focus on simplicity. It can be installed with npm.

It fits into a category of utilities known as test runners.

To install Jest we run **npm install --save-dev jest** and then we have to change the test field in the package.json to **jest --forceExit**.

**SuperTest** provide a high-level abstraction for testing HTTP, while still allowing you to drop down to the lower-level API provided by superagent.

To install supertest we run **npm install --save-dev supertest**.

For Testing in my project, I have created a app.tests.js file which looks like this:

```

const supertest = require("supertest");
const { response } = require("./app");
const app = require("./app");

describe("GET /allUserList", () => {
  describe("Get all UserList", () => {
    it("responds with json", function (done) {
      supertest(app)
        .get("/api/allUserList")
        .set("Accept", "application/json")
        .expect("Content-Type", /json/)
        .expect(200, done);
    });
  });
});

describe("GET /allEventList", () => {
  describe("Get all approved EventList", () => {
    it("responds with json", function (done) {
      supertest(app)
        .get("/api/allEvents")
        .set("Accept", "application/json")
        .expect("Content-Type", /json/)
        .expect(200, done);
    });
  });
});

```

Now to do testing we have to run npm run test or npm test in the terminal.  
Here is the testing and result of our Project -

```

PASS  ./app.test.js
  GET /allUserList
    Get all UserList
      ✓ responds with json (930 ms)
  GET /allEventList
    Get all approved EventList
      ✓ responds with json (43 ms)
  POST /login
    ✓ Succesfull login should return 200 (141 ms)
    ✓ Invalid login should return 404 (109 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:  0 total
Time:        1.629 s, estimated 2 s
Ran all test suites.

```

- **Docker Artifact:**

- Docker is a tool that makes it easy to create, deploy, and run applications by using containers. Containers allow you to package up an application with every part it needs, like libraries and other dependencies, and deploy it as one package.
- Docker hub is the world's largest repository of container images with an array of content sources including container community developers, open-source projects, and independent software vendor building and distributing their code in containers.
- Some useful commands in Docker are as follows -
  - docker images
  - sudo docker run -it name container ID
  - sudo docker service ls
  - sudo docker swarm it

```

stage('Docker Build') {
    steps {
        // delete if image already exists
        // sh 'docker rm college_event_management_server college_event_management_server'
        sh 'docker-compose build'
        sh 'docker tag event-management_server:latest shreyankb/event_management_server:latest'
        sh 'docker tag event-management_client:latest shreyankb/event_management_client:latest'
    }
}
stage('Docker Push') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'DockerHub', passwordVariable: 'dockerHubPassword', usernameVariable: 'dockerHubUser')])
        sh "docker login -u ${env.dockerHubUser} -p ${env.dockerHubPassword}"
        sh 'docker push shreyankb/event_management_server:latest'
        sh 'docker push shreyankb/event_management_client:latest'
    }
}
stage('Clean Docker Images') {
    steps {
        sh 'docker rmi -f shreyankb/event_management_server'
        sh 'docker rmi -f shreyankb/event_management_client'
    }
}

```

- **Docker-compose:**

- Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.
- Docker Compose commands :
  - *docker-compose up* : Start all services(it will run docker-compose.yml file)
  - *docker-compose build* : This command will build the images with the help of docker-compose.yml file.
  - *docker-compose down* : Stop all the services.

The screenshot shows a code editor with two tabs: 'Dockerfile' and 'docker-compose.yml'. The 'docker-compose.yml' tab is active, displaying the following configuration:

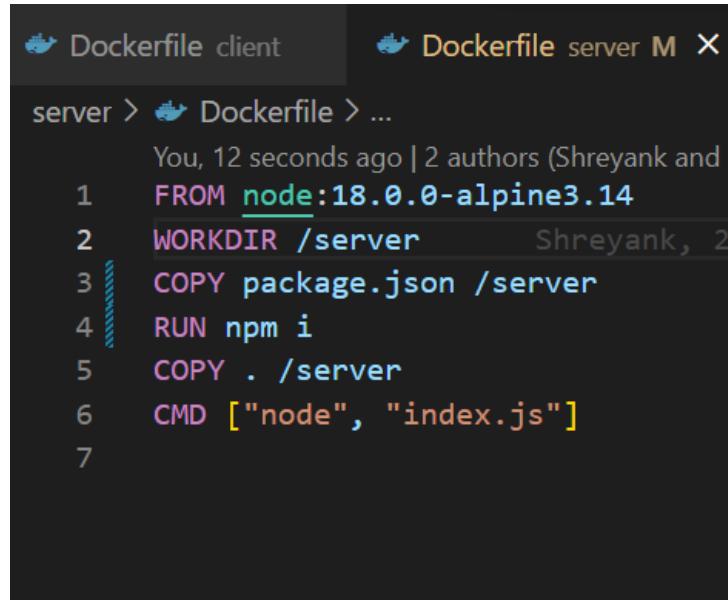
```
version: '3.0'
services:
  server:
    container_name: server
    stdin_open: true
    restart: always
    build: ./server/
    ports:
      - "8000:8000"
  client:
    container_name: client
    stdin_open: true
    restart: always
    build: ./client
    ports:
      - "3000:3000"
    depends_on:
      - server
```

- **Docker file for frontend:**

The screenshot shows a code editor with a single Dockerfile tab. The Dockerfile contains the following code:

```
FROM nginx:alpine
WORKDIR /usr/share/nginx/html
RUN rm -rf /*
COPY ./build .
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

- **Docker file for backend:**



The screenshot shows a Dockerfile editor interface with two tabs: "Dockerfile client" and "Dockerfile server". The "Dockerfile server" tab is active, displaying the following Dockerfile content:

```
FROM node:18.0.0-alpine3.14
WORKDIR /server
COPY package.json /server
RUN npm i
COPY . /server
CMD ["node", "index.js"]
```

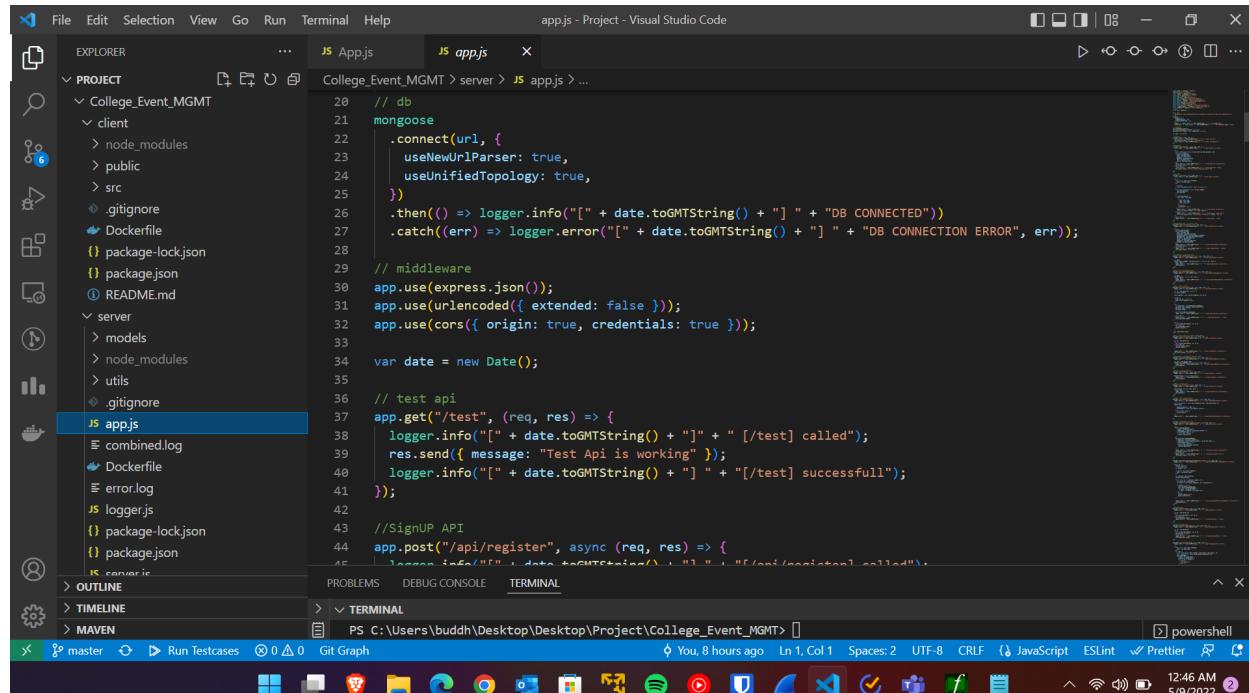
- Deploy

- **Ansible:**
- Ansible is an open source IT Configuration Management, Deployment & Orchestration tool. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments. It runs on many Unix-like systems and can configure both Unix-like systems and Windows.

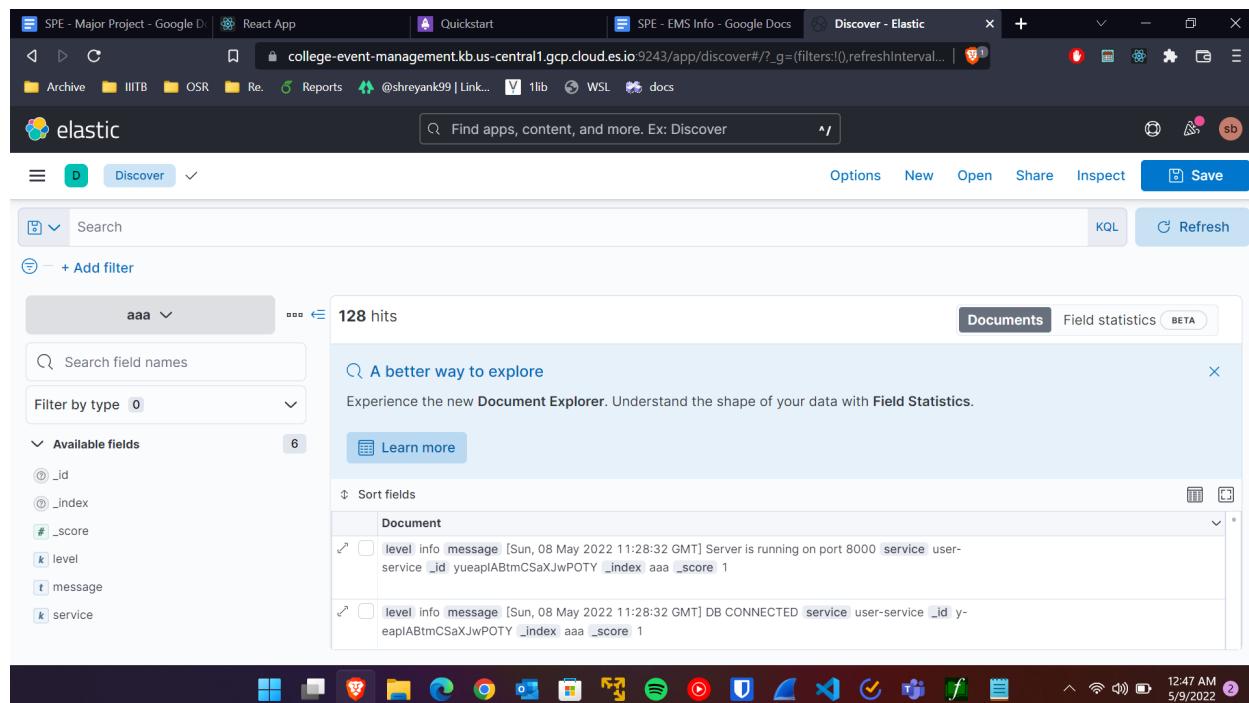
```
stage('Ansible Deploy') {
    steps {
        ansiblePlaybook colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inventory', playbook: 'deploy.yml'
    }
}
```

- **Monitor**

- **Monitoring for backend**
- To monitor the backend logs ELK Stack is used. **Winston logger** is used to generate the backend logs.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure for 'College\_Event\_MGMT' with files like 'client', 'server', and 'app.js'. The 'TERMINAL' tab at the bottom shows a command-line session with the prompt 'PS C:\Users\buddh\Desktop\Desktop\Project\College\_Event\_MGMT>'. The main editor area shows the 'app.js' file with code related tomongoose connection and middleware setup.

The screenshot shows the Elastic Stack interface, specifically the 'Discover' view. It displays 128 hits from a search query. The results show log entries such as 'level info message [Sun, 08 May 2022 11:28:32 GMT] Server is running on port 8000 service user-service \_id yueaplABtmCSaXJwPOTY \_index aaa \_score: 1' and 'level info message [Sun, 08 May 2022 11:28:32 GMT] DB CONNECTED service user-service \_id yaplABtmCSaXJwPOTY \_index aaa \_score: 1'. The interface includes a search bar, filter options, and a 'Field statistics' section.

- **Monitoring for frontend**
- To monitor front end logs **logrocket** is used. It records all front end activities and generates a video for it as well.

The screenshot shows a Windows desktop environment with several open windows. At the top is a Visual Studio Code window displaying the file `app.js` from a project named `College_Event_MGMT`. The code includes imports for React, routes, and components, and a call to `LogRocket.init`.

Below the code editor is a terminal window showing the command `PS C:\Users\buddh\Desktop\Project\College_Event_MGMT>`. The system tray indicates the date as May 9, 2022, and the time as 12:48 AM.

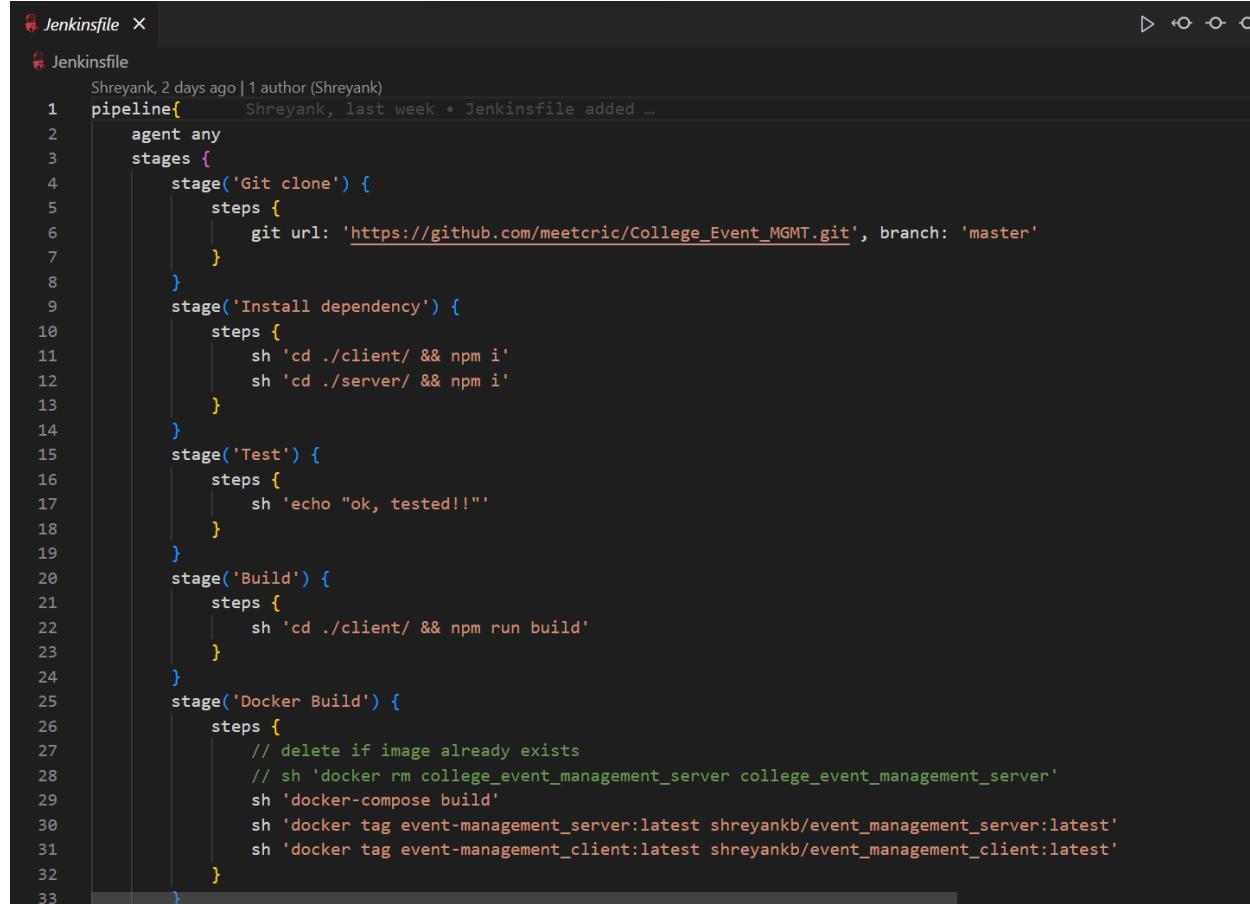
The main focus is the LogRocket interface, which has three tabs: PROBLEMS, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab shows the URL `app.logrocket.com/mbixxx/college.event.mgmt`. The DEBUG CONSOLE tab shows a log entry: "meet Patel last month".

The LogRocket Overview page displays session data for an anonymous user from May 9, 2022, at 12:37 AM GMT+5:30. It shows 41 events over 0:01:13. The interface includes sections for Dashboards, Sessions, Issues, Metrics, Heatmaps, Path Analysis, and Project Setup (67% complete). A sidebar on the left shows support links for What's New and Support, and a message from shreyankbuddhadev99.

At the bottom of the LogRocket window is a Network tab showing a timeline of requests and a corresponding playback video player. The video player shows a recording of a browser session with a warning message: "This session was recorded locally, the video may not include stylesheets or images. Learn more..."

- **Building Pipeline and Running on Machine:**

- In the above sections we have integrated npm, ansible, docker and other necessary tools with jenkins. We were able to trigger ansible from Jenkins and also build docker images with Jenkins. Here is a figure of the entire pipeline and its final full-stage view after a successful run of Jenkins pipeline.



The screenshot shows a code editor window with a dark theme, displaying a Jenkins pipeline script named 'Jenkinsfile'. The file was last modified by Shreyank, 2 days ago, with 1 author (Shreyank). The pipeline definition is as follows:

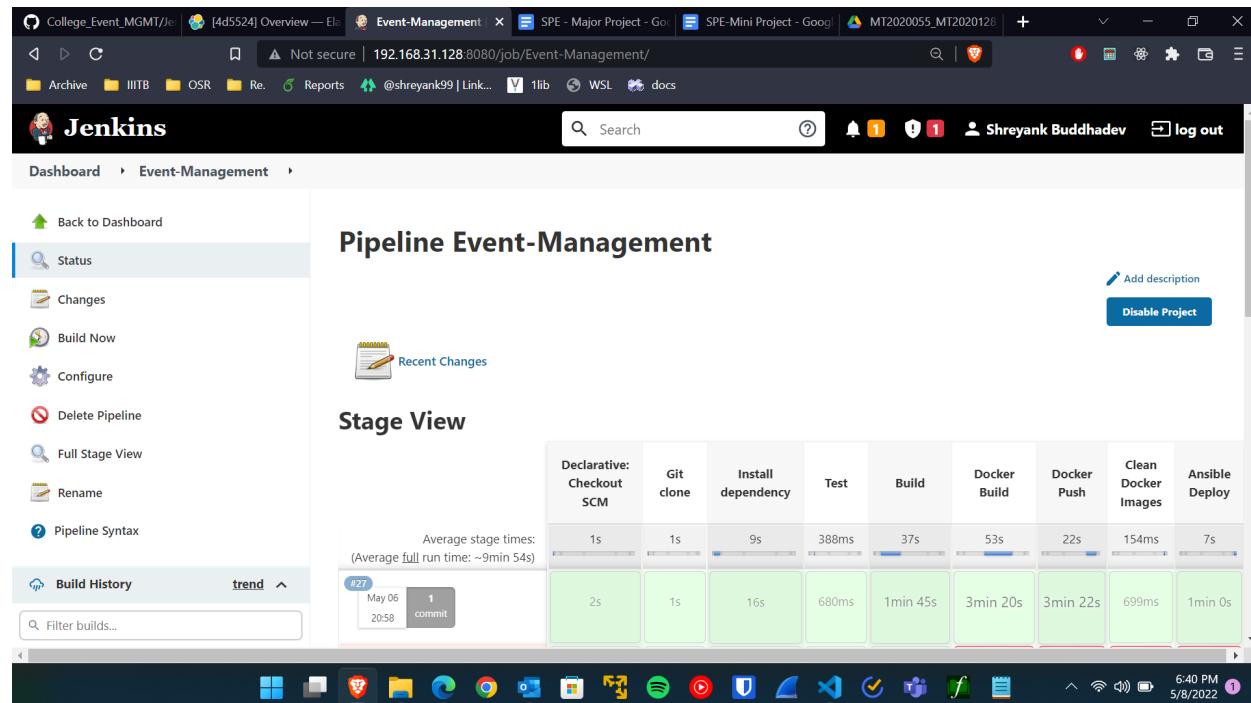
```
1 pipeline{           Shreyank, last week • Jenkinsfile added ...
2   agent any
3   stages {
4     stage('Git clone') {
5       steps {
6         git url: 'https://github.com/meetcric/College_Event_MGMT.git', branch: 'master'
7       }
8     }
9     stage('Install dependency') {
10    steps {
11      sh 'cd ./client/ && npm i'
12      sh 'cd ./server/ && npm i'
13    }
14  }
15  stage('Test') {
16    steps {
17      sh 'echo "ok, tested!!"'
18    }
19  }
20  stage('Build') {
21    steps {
22      sh 'cd ./client/ && npm run build'
23    }
24  }
25  stage('Docker Build') {
26    steps {
27      // delete if image already exists
28      // sh 'docker rm college_event_management_server college_event_management_server'
29      sh 'docker-compose build'
30      sh 'docker tag event-management_server:latest shreyankb/event_management_server:latest'
31      sh 'docker tag event-management_client:latest shreyankb/event_management_client:latest'
32    }
33  }
}
```

```

34     }
35     }
36     stage('Docker Push') {
37         steps {
38             withCredentials([usernamePassword(credentialsId: 'DockerHub', passwordVariable: 'dockerHubPassword',
39                             usernameVariable: 'dockerHubUser')])
40             sh "docker login -u ${env.dockerHubUser} -p ${env.dockerHubPassword}"
41             sh 'docker push shreyankb/event_management_server:latest'
42             sh 'docker push shreyankb/event_management_client:latest'
43         }
44     }
45     stage('Clean Docker Images') {
46         steps {
47             sh 'docker rmi -f shreyankb/event_management_server'
48             sh 'docker rmi -f shreyankb/event_management_client'
49     }
50     stage('Ansible Deploy') {
51         steps {
52             ansiblePlaybook colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory:
53         }
54     }
55 }
56

```

- Here is the successful build of our Jenkins Pipeline



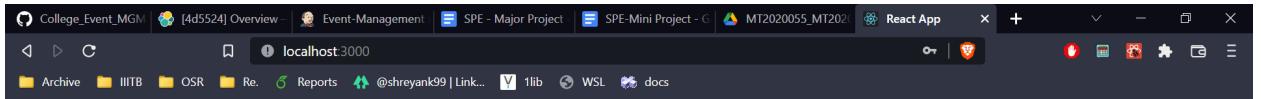
- The docker image which is being pushed to the local host of our system is shown in the below figure
- This can be checked using the command : docker images

```
Last login: Fri May  6 21:07:39 2022 from 127.0.0.1
doshreyank@ubuntu:~$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
event-management_client    latest   ed4d85f18e2b  46 hours ago  28MB
shreyankb/event_management_client    latest   ed4d85f18e2b  46 hours ago  28MB
event-management_server    latest   dd22e0908cae  46 hours ago  189MB
shreyankb/event_management_server    latest   dd22e0908cae  46 hours ago  189MB
```

- We don't have to run docker images on client machine to run our code because docker-compose did it for us. So now our server is running on <http://localhost:8000> url and Frontend running on <http://localhost:3000> url.

## Result and Discussion

- Sign In page:

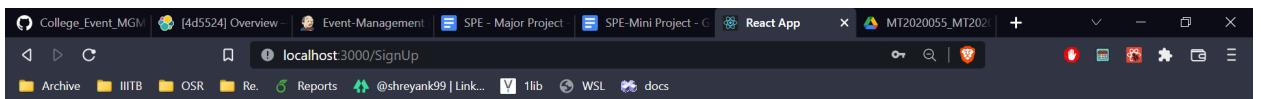


 Sign in

[Forgot password?](#)      [Don't have an account? Sign Up](#)



- Register Page:



 Sign up

User Role

Course

Already have an account? [Sign in](#)



- **Admin Dashboard**

- **User tab**

Email	Phone	Role	Course	Action
meet@gmail.com	9726467772	Admin	M.Tech	
shreyank@student.com	8980598884	Student	M.Tech	
Shreyank@Em.com	8980898088	EventManager	M.Tech	

- **Pending Events**

I.	Event Name	Event Type	Added By	Max...	Allowed U...	Date & Time
6277c17...	Meet	research-talk	Shreyank EM	100	IM.tech	2022-05-13T18:41
6277c17...	Meet	research-talk	Shreyank EM	100	IM.tech	2022-05-13T18:41

- **All Events**

The screenshot shows a web browser window titled "localhost:3000/AdminDashboard/AllEvents". The page has a header "IITB EM" and a sidebar with "Admin Dashboard" and links for "Users", "Pending Events", and "All Events". The main content is a table with columns: Event Name, Event Type, Added By, Max..., Allowed U..., and Date & Time. One row is visible: "ID 7c17... Meet research-talk Shreyank EM 100 IM.tech 2022-05-13T18:41".

- **Event Manager Dashboard**

- **Apply for Events**

The screenshot shows a web browser window titled "localhost:3000/EventDashboard". The page has a header "IITB EM" and a sidebar with a "Quick Menu" containing "Apply for Events", "Pending Events", and "Approved Events". The main content is a "New Event" form with fields: Name (Dumb Charades), Event Type (Research talk), Maximum Participation (100), Allowed User Groups (Select Option), and Date and Time (mm/dd/yyyy --:-- --).

- **Pending Events**

IITB EM

I..	Event Name	Event Type	Max...	Allowed U...	Date & Time	Venue
6277c17...	Meet	research-talk	100	IM.tech	2022-05-13T18:41	AUDI

## o Approved Events

IITB EM

I..	Event Name	Event Type	Added By	Max...	Allowed U...	Date & Time
6277c17...	Meet	research-talk	Shreyank EM	100	IM.tech	2022-05-13T18:41

- **User Dashboard**

- **All Events**

The screenshot shows a web browser window titled "localhost:3000/StudentDashboard/allEvents". The page has a header "IITB EM" and a sidebar with a "Quick Menu" containing "All Events" (which is highlighted) and "Events participated". The main content is a table with the following columns: I.., Event Name, Event Type, Added By, Max..., Allowed U..., and Date & Time. There are two rows of data:

I..	Event Name	Event Type	Added By	Max...	Allowed U...	Date & Time
6277ca1...	a	research-talk	Shreyank EM	100	M.Tech	2022-05-20T22:21
6277ca1...	a1	research-talk	Shreyank EM	100	M.Tech	2022-05-20T22:21

The status bar at the bottom shows the URL "localhost:3000/StudentDashboard/allEvents" and the date "5/8/2022".

- **Events Participated**

The screenshot shows a web browser window titled "localhost:3000/StudentDashboard/products". The page has a header "IITB EM" and a sidebar with a "Quick Menu" containing "All Events" and "Events participated" (which is highlighted). The main content is a table with the same columns as the previous screenshot. There is one row of data:

I..	Event Name	Event Type	Added By	Max...	Allowed U...	Date & Time
6277ca1...	a	research-talk	Shreyank EM	100	M.Tech	2022-05-20T22:21

The status bar at the bottom shows the URL "localhost:3000/StudentDashboard/products" and the date "5/8/2022".

## **Scope for future works**

### **Cloud deployment**

We can deploy our project using amazon AWS and other cloud services which makes our project platform-independent so that it can be used from anywhere on the earth.

### **Generalization**

We can also generalise this project and increase its scope to accommodate all the requirement of a college event management portal, So that next year INFIN8 team can use this code instead of using google form to manage events.

### **Other Feature Enhancement**

We can build more modules like live Dashboard for admin, Event manager. Such modules will help in showing live statics of event.

We can enhance the Student experience by including modules like result of each event and other required features. (Aim to provide the experience same as HackerEarth).

We can emphasize more on the security of the both frontend, backend part.

## **Conclusion**

We successfully built an College event management system using DevOps tools. The tools we used are: GitHub, Jenkins, Docker, Ansible. These tools are integrated using Jenkins. The entire pipeline has been automated. For deployment of the entire project, it's taking a minimum of 15 minutes.

The Devops methodology and lifecycle tools prove to be better than the Agile methodology in terms of technical, cultural and business benefits. By minimizing friction between independent teams, DevOps enables a collaborative approach for enterprise software development and delivery that reflects the needs of the entire application lifecycle for today's modern enterprises.

Thus, we can develop, test, deploy and monitor the application easily.

