MINOR PROJECT–1 REPORT

ON

CONTINUOUS DEPLOYMENT OF A MICROSERVICES-BASED WEB APPLICATION

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE

SUBMITTED BY:

RAJ ARYAN SHARMA 18103143

MAYANK YADAV 18103136

SHRIYANSH AGGARWAL 17103035

UNDER THE SUPERVISION OF: DR. SULABH TYAGI

DEPARTMENT OF COMPUTER SCIENCE/IT,

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA, UP, INDIA

CERTIFICATE

This is to certify that the minor project report entitled, “**CONTINUOUS DEPLOYMENT OF A MICROSERVICES-BASED WEB APPLICATION**” submitted by RAJ ARYAN SHARMA,MAYANK YADAV, SHRIYANSH AGGARWAL in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in CSE of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature of Supervisor:

Name of the Supervisor: Dr.Sulabh Tyagi

CSE/IT Department, JIIT,Noida

Dated:

DECLARATION

We hereby declare that this written submission represents our own ideas in our own words and where others' ideas or words have been included, have been adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

Dated: RAJ ARYAN SHARMA 18103143

MAYANK YADAV 18103136

SHRIYANSH AGGARWAL 17103035

ACKNOWLEDGEMENT

We wish to express our sincere appreciation to our supervisor, Dr. Sulabh Tyagi for his timely, informative feedback and support during this effort. It has been a great pleasure to learn from him during the process of our work. He has been a generous mentor. We extend our most sincere thanks to Dr.Ankit Vidyarthi (Coordinator of Minor Project – I) and our panelists for their cooperation during the project work.

OBJECTIVE

The main objective behind this project is to take advantage of the flexibility that a Microservices architecture provides in the development of an application, and to use Docker as a common tool in the development phase and the deployment phase of our application.

There are several reasons to choose a Microservices architecture for development of a Web-app and one of them is to increase our productivity. Using Microservices, we would be able to work better as a distributed team and have our individual tasks sorted out.

With Microservices, we would be able to use multiple languages and frameworks to build our application, with improved productivity and speed.

We aim to understand and learn to successfully implement Microservices architecture in an application and automatically deploy the application on a cloud environment.

RESEARCH PAPERS

Analysis and Summary of the Research Papers read:

Research Paper 1: Application deployment using Microservice and Docker containers:Framework and optimization, By Xili Wana,Xinjie Guana,Tianjing Wanga,Guangwei Baia,Baek-Young Choib.

In this paper we learnt how microservices and docker containers make an application's deployment easier and how these related tools and concepts can be used effectively in the development and deployment of the an application. Microservices architecture and using Docker containers can help in improving the scalability and elasticity of application deployment and operation in cloud environments.
As an alternative to Hypervisor based virtualisation,container based virtualisation (like Docker containers) has been proposed because it saves system resources and it also offers operation system level abstraction and isolation.
If we talk about microservice architecture then it helps us to make complicated applications into lightweight and loosely linked components.To deploy complex applications, developers can partition the apps into subtasks. For each subtask, a container based replica would be deployed towards a certain optimisation goal. Sharing an operation system as well as supporting libraries, container based virtualization offers a great opportunity for reducing application deployment cost and improving final users' experience.

SUMMARY- This research paper aimed to provide understanding of why Microservice Architecture and Docker Container concept and techniques should be used in application deployment and operation over traditional techniques.

Research Paper 2: Adopting DevOps in the Real World: A Theory, a Model, and a Case Study,
By Welder Pinheiro Luz, Gustavo Pinto, Rodrigo Bonifacio.

SUMMARY- This paper presented a theory and model on adopting DevOps and applying it in practice.

Research Paper 3: Automated deployment of a microservice-based monitoring infrastructure,
by Augusto Ciuffoletti.

SUMMARY- This Research paper discussed a container based approach to automatically deploy an application in an open cloud computing interface(OCCI).

Research Paper 4: A Qualitative Study of DevOps Usage in Practice, June 2017 Journal of Software: Evolution and Process, Project: DevOps
By Floris Erich:National Institute of Advanced Industrial Science and Technology Chintan Amrit:University of Amsterdam.

SUMMARY-This research paper presented the concept of DevOps, as a methodology which has become a field of great interest for everyone as it is used by many organisations to reduce the conflict between Developer team and Operation team while doing any application development and its deployment. The paper explains how DevOps works.

Research Paper 5: Benefits of AWS in Modern Cloud, March 2019
By Sourav Mukherjee University of the Cumberlands.

SUMMARY- This paper presented the overview of Amazon Web Services, a cloud computing platform. It talks about the concept of cloud computing especially AWS, its advantages and its use. It tells how this technology improves the quality of service and reduces cost of complete development of any application and other services. It also gives the reasons that differentiate AWS from other cloud computing platforms.

PREREQUISITES

Operating System Requirements-

- Windows 10 or Linux or MacOS

Software Requirements-

- NodeJS v14.15.1
- Node Modules
- Npm 6.14.8
- Git 2.29.2.windows.3
- React 16.5.1
- React Router Dom 5.2.0
- JS-Base64 3.6.0
- React scripts 1.1.5
- Express 4.17.1
- MongoDB 3.6.3
- IDE
  - Visual Studio Code or Xcode
- Browser
  - Chrome or Edge or Firefox
- Docker
- MongoDB Compass
- MongoDB Atlas
- Aws Account

Hardware Requirements-

- Machine: PC or Laptop with required accessories
- RAM: Should at least be 4GB
- ROM: At least 5GB should be free
- Processor: At least Intel Core i3 or its equivalent (x64-based processor)

Analysing the system requirement for our minor project:

We have used the Windows operating system for our project due its vast availability and it is easier to work in teams on Windows as it is the most widely used OS.

Memory Management:
Memory management is an important task to manage resources with the available computer memory. The main purpose of memory management is to dynamically allocate memory to various programs as per their needs or when requested. The memory once used is then freed and then reused for some other program. Windows OS calls a function alloca for dynamically allocating stack memory in a similar way then heap malloc.There is mostly no need to manually free the memory when used alloca in Windows it automatically frees the memory when the function returns but there is a risk of memory overflow so it is often freed manually for security reasons.

Scheduling:
Windows scheduled its tasks based on priority. High priority programs or functions are performed first i.e. high priority programs have higher processor affinity. Most OS tasks are given high priority; all other programs work on normal priority. It uses a preemptive scheduling system where the tasks are assigned with priorities. Preemption is important as sometimes it is important to run a task with higher priority before another lower priority task, even when the lower priority task is still running. So then the lower priority tasked is preempted and then the higher priority task executes first.

File management:

File management in windows is easier than most operating systems. It provides us with Windows Explorer and My Computer where all the storage drives can be seen and their remaining spaces and used spaces could be analysed easily. It keeps different drives for different functionalities like Local Drive C is mostly used by Windows to store its files needed for OS to operate and no one can easily keep other things in other drives. It displays a hierarchical list of files,folders and storage drives on the computer. It also lists network drives that are being used on the computer. Windows provides us with all basic operations like create, move, copy, delete and search for files and folders.

Thread Management:

A process may require to run on one or more threads to make sure proper functioning of the program. It basically creates an illusion to the processor that more than one programs are running and thus providing each thread with a limited time to execute. In Windows, the Windows kernel-mode process and thread manager handles the working of all the threads in a computer machine. It manages the working of threads on processors irrespective of the number of processors present in the machine.

Concurrency Control:

The main concurrency mechanism in Windows OS is the usage of threads. The threads are created using the API for threads and calling CreateThread function which generates threads for the various processes and managing the control of concurrency in the Windows Operating System. Creating threads is relatively easy and they are easy to use though the operating system allocates a significant amount of time and other resources to manage them. Concurrency control in Windows is better than many other Operating systems present.

b) Factors to choosing Windows:

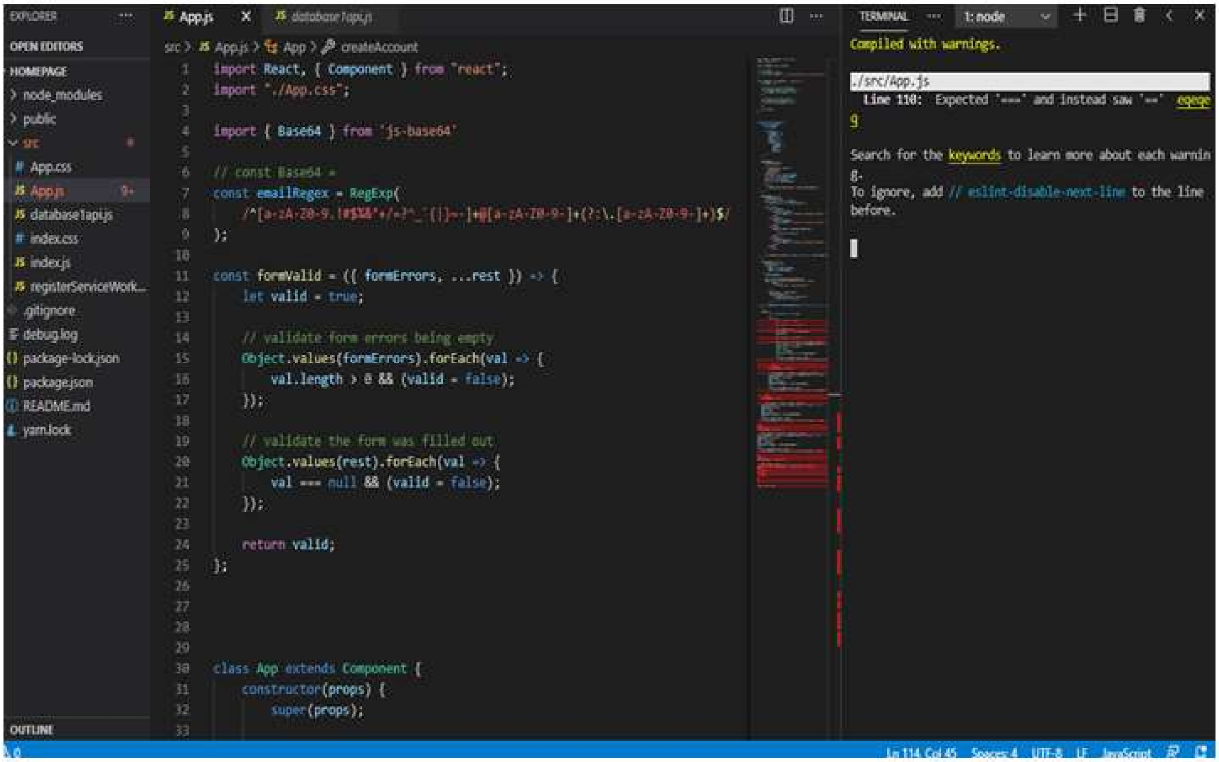
- Better UI
- Ease of Use
- Availability
- Good Memory management
- Good file directory system
- Features like Virtual Machines and WSL available

METHODOLOGY

PRODUCTION STAGE:-

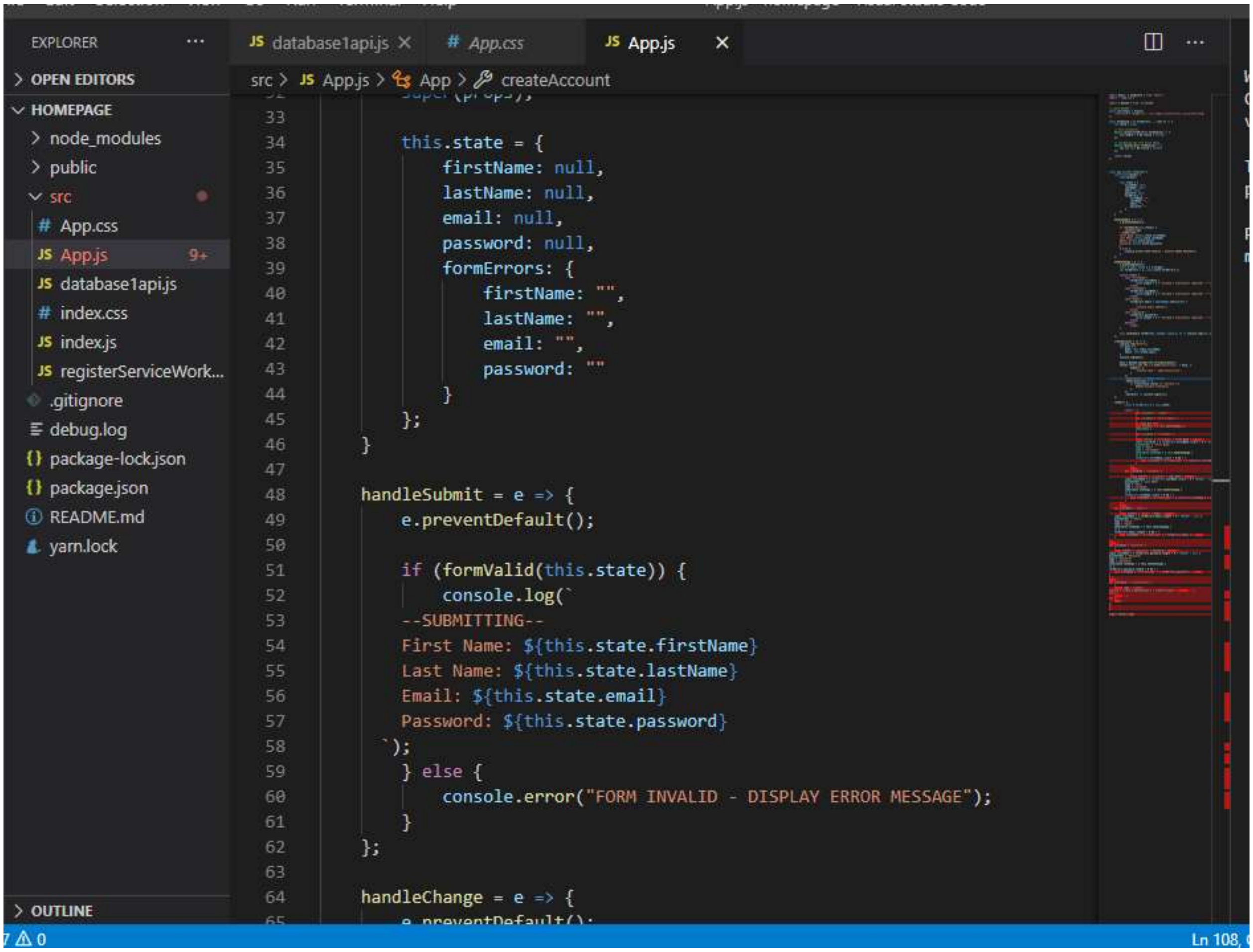
FRONTEND:

Service1- Sign Up form

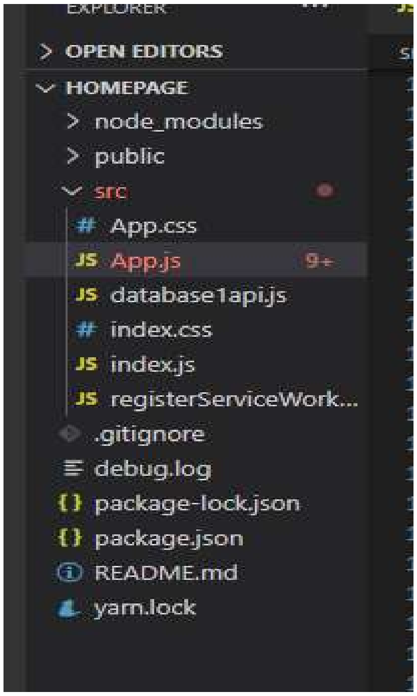


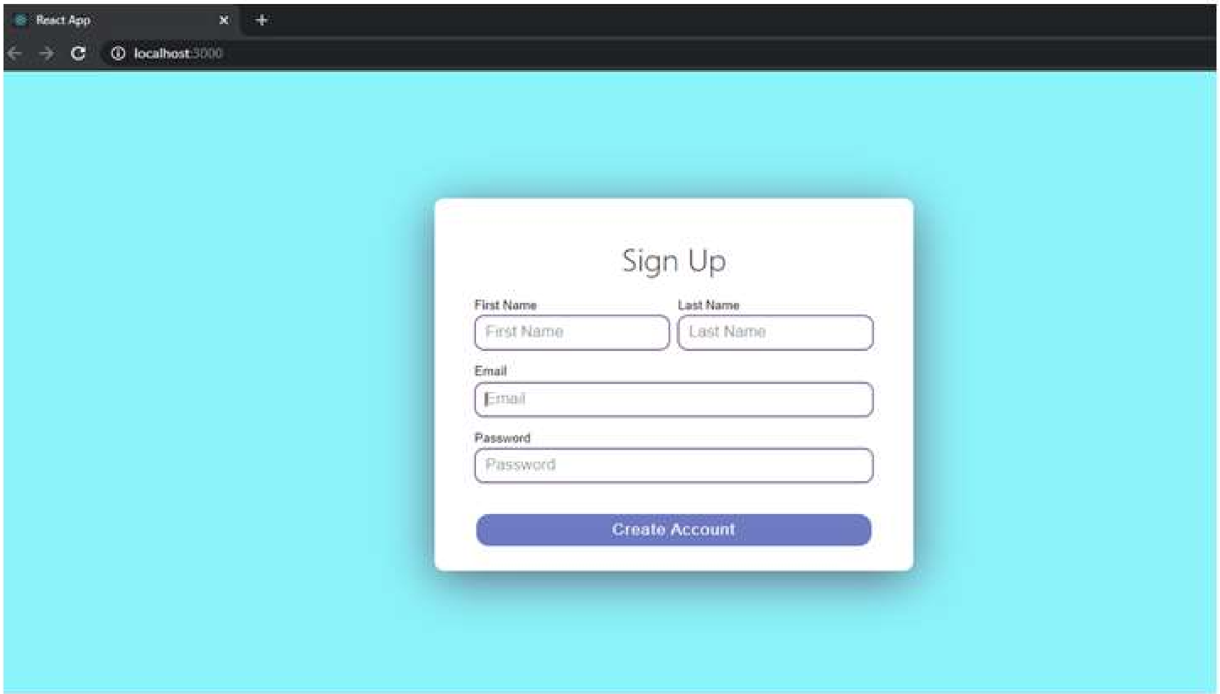
```
import React, { Component } from "react";
import "./App.css";

import { Base64 } from 'js-base64'

// const Base64 =
const emailRegex = RegExp(
    /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)$/
);

const formValid = ({ formErrors, ...rest }) => {
    let valid = true;

    // validate form errors being empty
    Object.values(formErrors).forEach(val => {
        val.length > 0 && (valid = false);
    });

    // validate the form was filled out
    Object.values(rest).forEach(val => {
        val === null && (valid = false);
    });

    return valid;
};




class App extends Component {
    constructor(props) {
        super(props);
```

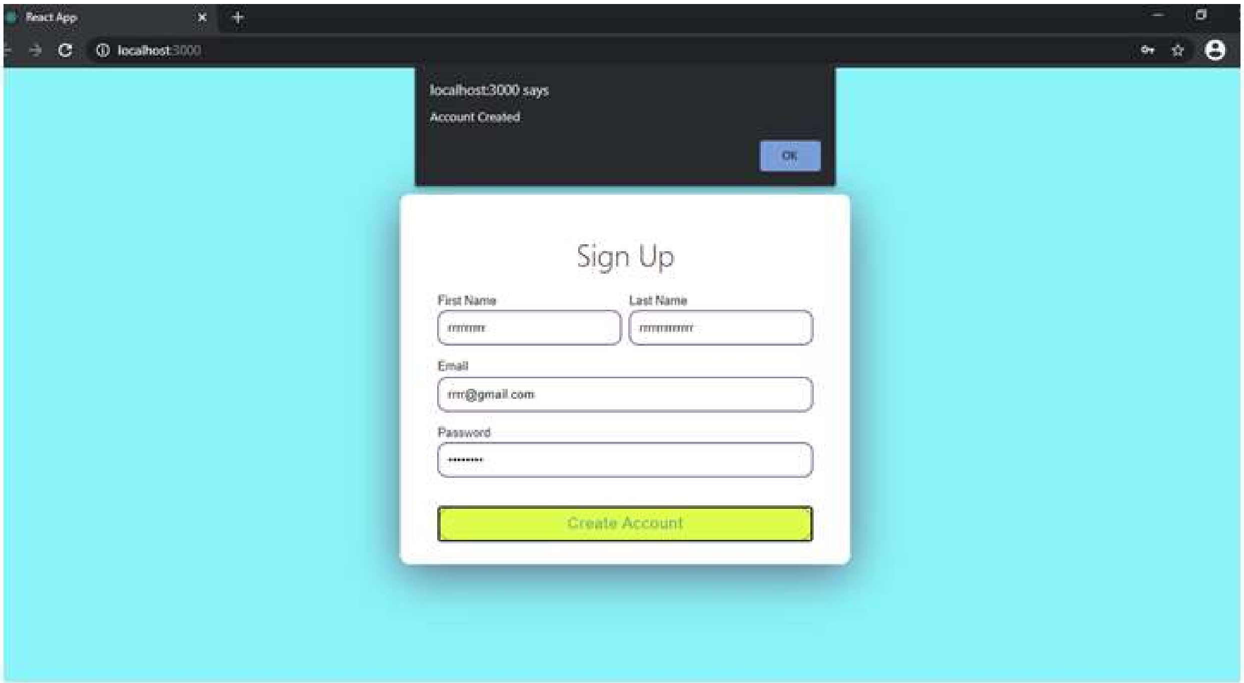
```
Compiled with warnings.

./src/App.js
  Line 110:  Expected '===' and instead saw '=='  eqeqe
q

Search for the keywords to learn more about each warnin
g.
To ignore, add // eslint-disable-next-line to the line
before.
```



```
    super(props);

    this.state = {
      firstName: null,
      lastName: null,
      email: null,
      password: null,
      formErrors: {
        firstName: "",
        lastName: "",
        email: "",
        password: ""
      }
    };
  }

  handleSubmit = e => {
    e.preventDefault();

    if (formValid(this.state)) {
      console.log(`
    --SUBMITTING--
    First Name: ${this.state.firstName}
    Last Name: ${this.state.lastName}
    Email: ${this.state.email}
    Password: ${this.state.password}
  `);
    } else {
      console.error("FORM INVALID - DISPLAY ERROR MESSAGE");
    }
  };

  handleChange = e => {
    e.preventDefault();
```







localhost:3000 says

Account Created

rrrrrrrrr

rrrrrrrrrrr

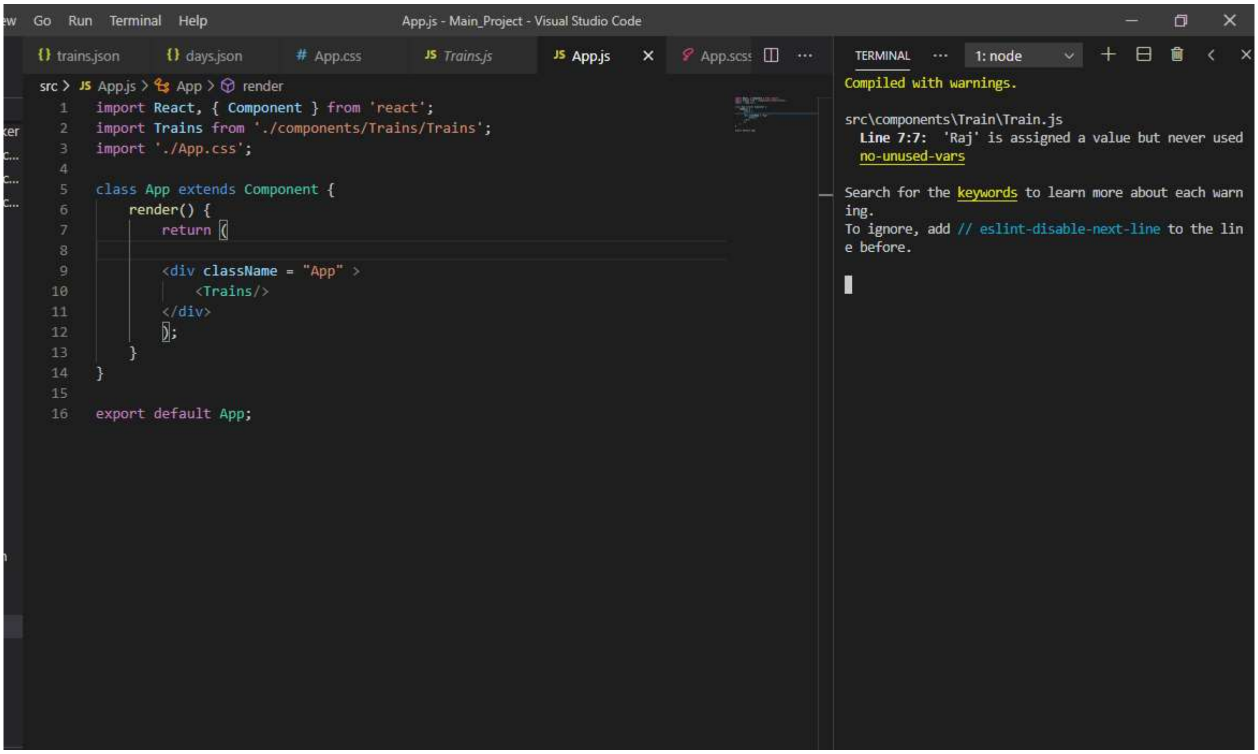
rrrr@gmail.com

•••••••••

Only when the API is running, the account would be successfully created.

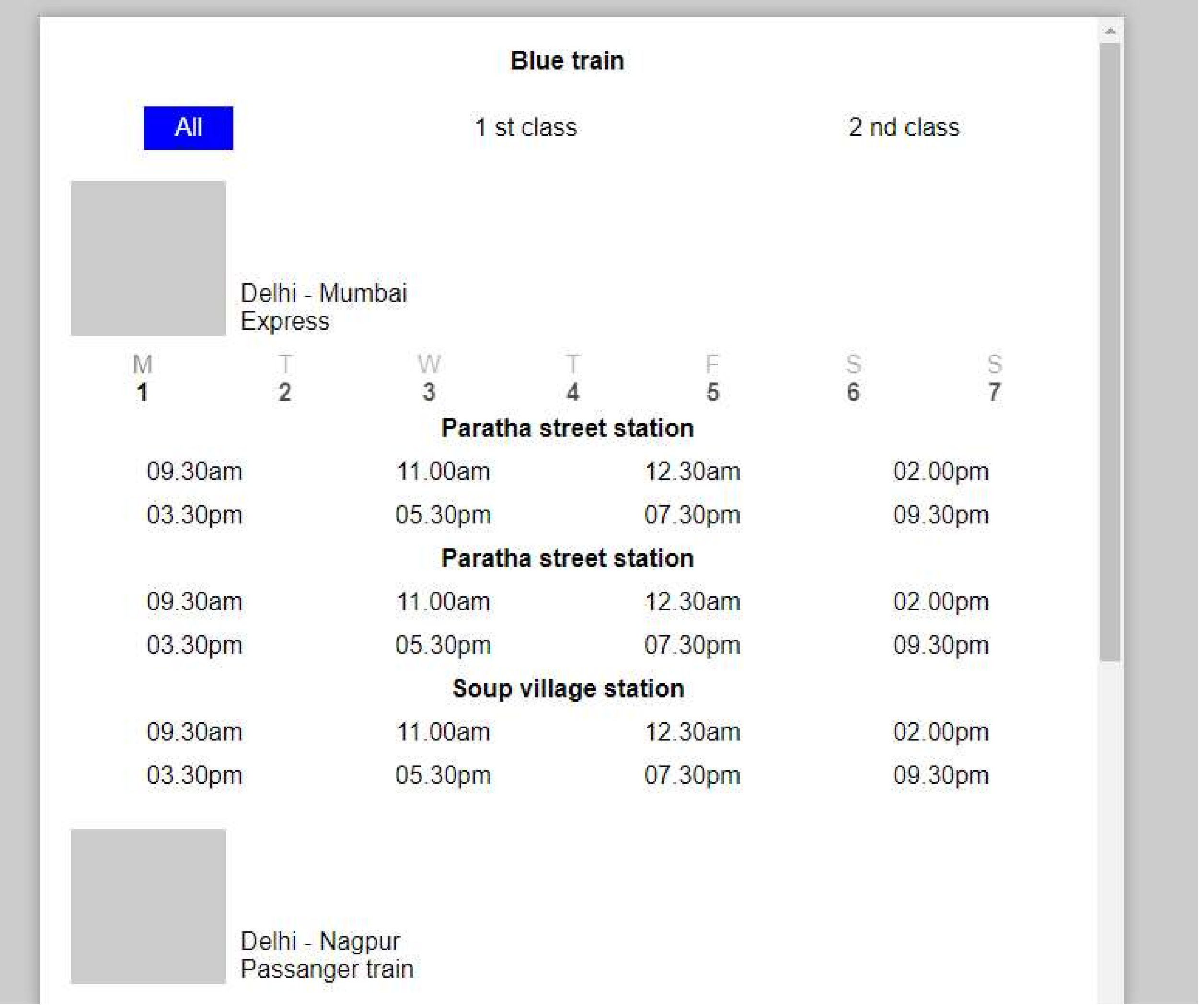
- Using ReactJS and CSS, a registration form was created to register new accounts and to get the required information to book a train ticket.
- The form includes form validation using Regex.
- The form is linked with a backend service through an API and after successful execution, an account gets created.
- The service has JSON code as well that encodes the form data and sends it to the API.

Service 2- Ticket display window



```
import React, { Component } from 'react';
import Trains from './components/Trains/Trains';
import './App.css';

class App extends Component {
    render() {
        return (

        <div className = "App" >
            <Trains/>
        </div>
        );
    }
}

export default App;
```

```
Compiled with warnings.

src\components\Train\Train.js
  Line 7:7:  'Raj' is assigned a value but never used
  no-unused-vars

Search for the keywords to learn more about each warn
ing.
To ignore, add // eslint-disable-next-line to the lin
e before.
```



Blue train

Delhi - Mumbai
Express

M	T	W	T	F	S	S
1	2	3	4	5	6	7

Paratha street station

09.30am	11.00am	12.30am	02.00pm
03.30pm	05.30pm	07.30pm	09.30pm

Paratha street station

09.30am	11.00am	12.30am	02.00pm
03.30pm	05.30pm	07.30pm	09.30pm

Soup village station

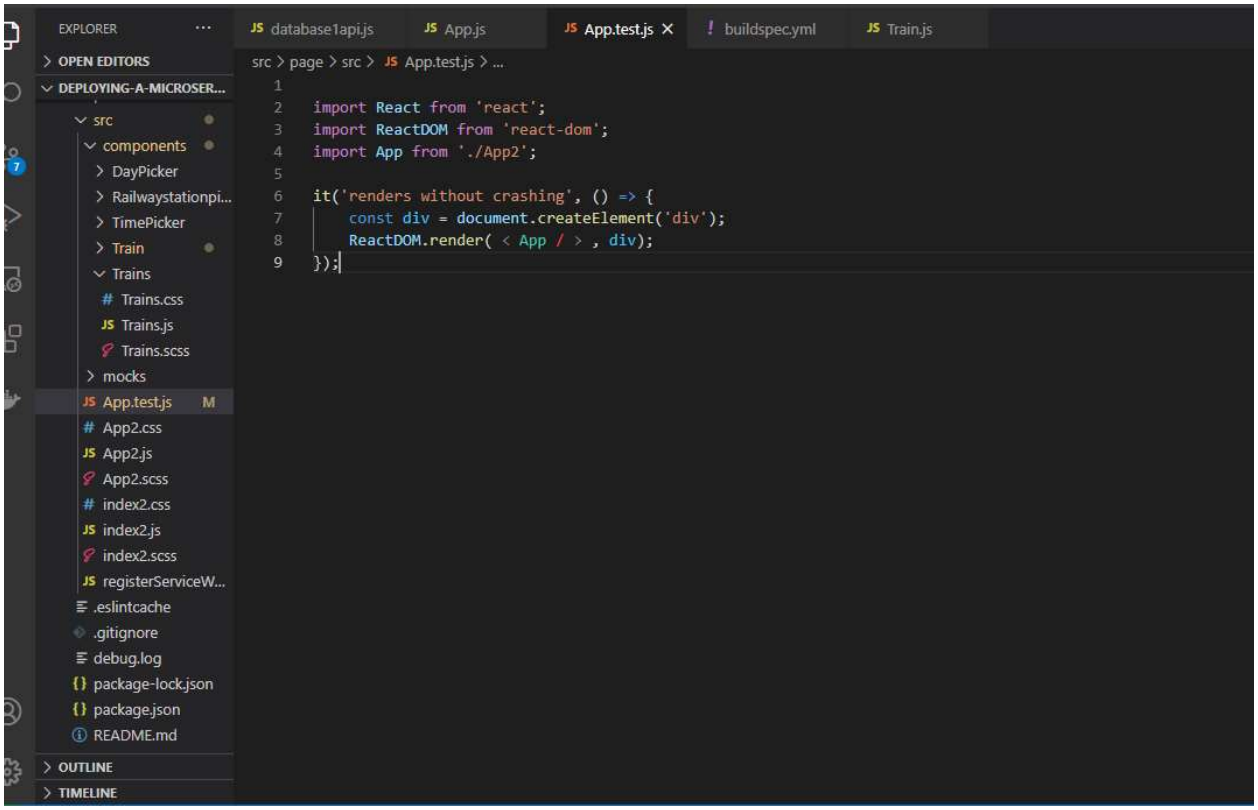
09.30am	11.00am	12.30am	02.00pm
03.30pm	05.30pm	07.30pm	09.30pm

Delhi - Nagpur
Passanger train

- Using ReactJS, a website was created in the form of a window displaying some random trains available for booking along with some of the relevant information such as:-

1. Name of the station
2. Name of the location: from -> to
3. Time
4. Running date and day
5. Class

The window includes a method for shorting out the station from which the train would be running on a particular day .



```

import React from 'react';
import ReactDOM from 'react-dom';
import App from './App2';

it('renders without crashing', () => {
    const div = document.createElement('div');
    ReactDOM.render( < App / > , div);
});
```

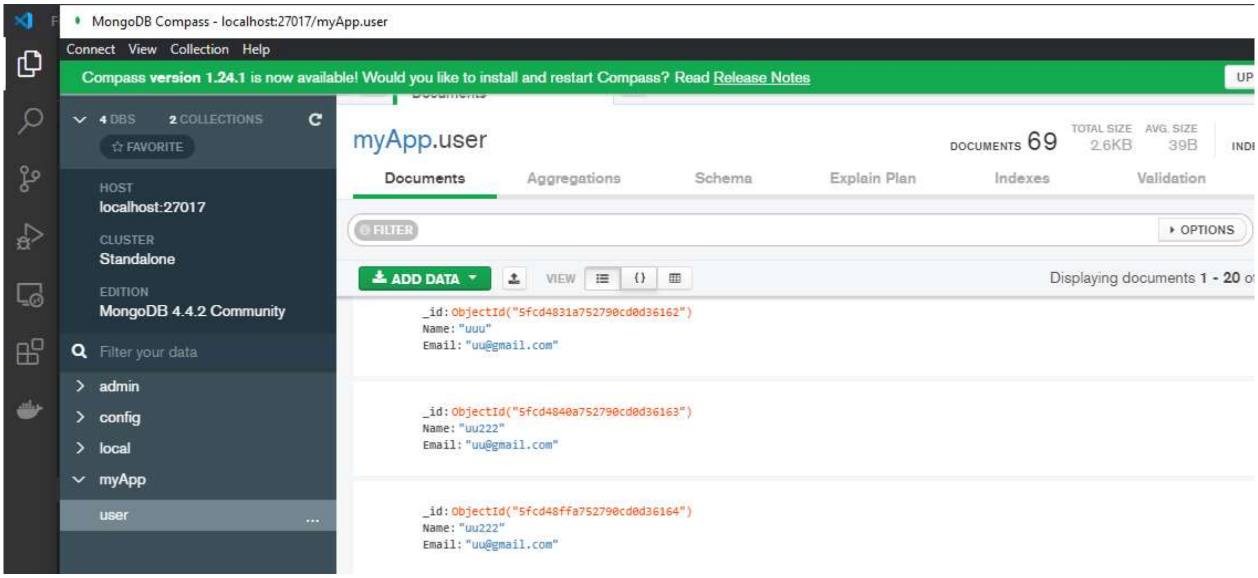
Working:

- The app is runned through app.js file
- App.js file calls Trains.js file which is in component folder
- Trains.js file has all the ticket window elements and how the overall feel of the window will be ,calls Train.js
- Train.js display how an individual train would be seen and interacted with,call’s Daypicker.js .
- Daypicker.js displays on which railway station the train would be available on that particular day,call’s Railwaystationpicker.js
- Railwaystationpicker.js shorts and displays the information for the station from which the Train is available on the particular day and on which time,call’s Timepicker.js
- Timepicker.js displays the time at which the train is available on which station on that particular day.

BACKEND:

Service 1-

- Our first back-end service is made in MongoDB and NodeJS.
- This service is connected to the Frontend form through an ExpressJS API.
- The API has JSON code as well that decodes the encoded form data.
- The NodeJS service verifies whether JSON decoding was successful or not and then sends an alert on the frontend about successful account creation.
- MongoDB stores the form data in form of data nodes.



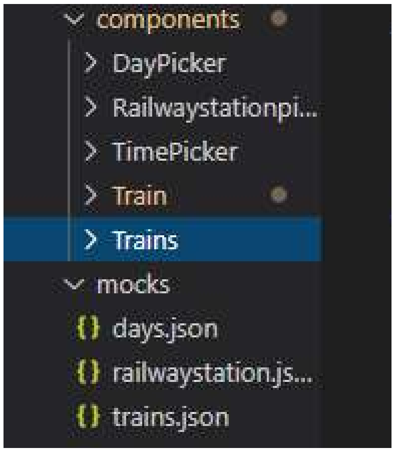
```
_id: ObjectId("5fcd4831a752790cd0d36162")
Name: "uuu"
Email: "uu@gmail.com"

_id: ObjectId("5fcd4840a752790cd0d36163")
Name: "uu222"
Email: "uu@gmail.com"

_id: ObjectId("5fcd48ffa752790cd0d36164")
Name: "uu222"
Email: "uu@gmail.com"
```

- We used Mongo Atlas for the online server of the database needed for the application. We exported our database and collection from MongoDB Compass to MongoDB Atlas.

Service 2-

Explaining the code present in '../src/page/src':-

- The files Daypicker.js calls information from file days.json containing information about the days to be displayed.
- Railwaystationpicker.js calls info from file railstations.json containing information about railway stations to be displayed and sorted from.
- Timepicker.js calls info from file time.json containing information about time the train would be available to be displayed and sorted from.
- The displayed information on the second page is coming through JSON files.



BUILD STAGE:-

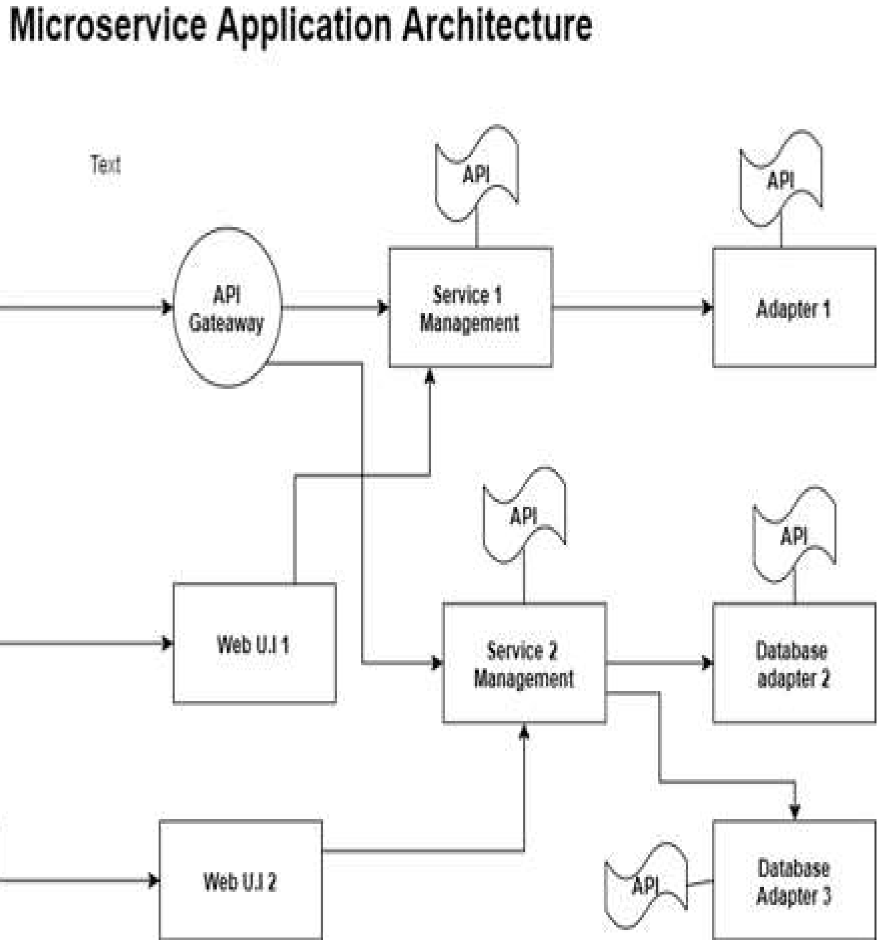
- Code build service of AWS (AWSCodeBuild) manages the build stage of the project.
- To implement the build stage, we first created a docker image of the source code of the web application.

- A docker container was created.
- The container was then pushed to and hosted on the Docker Hub.
- After the Sourcing is done, the code is sent to the Build stage.
- While building, the buildspec.yml file and the ..test.js files are read.
- After all the tests run fine, the building is complete and the code is sent for deployment.

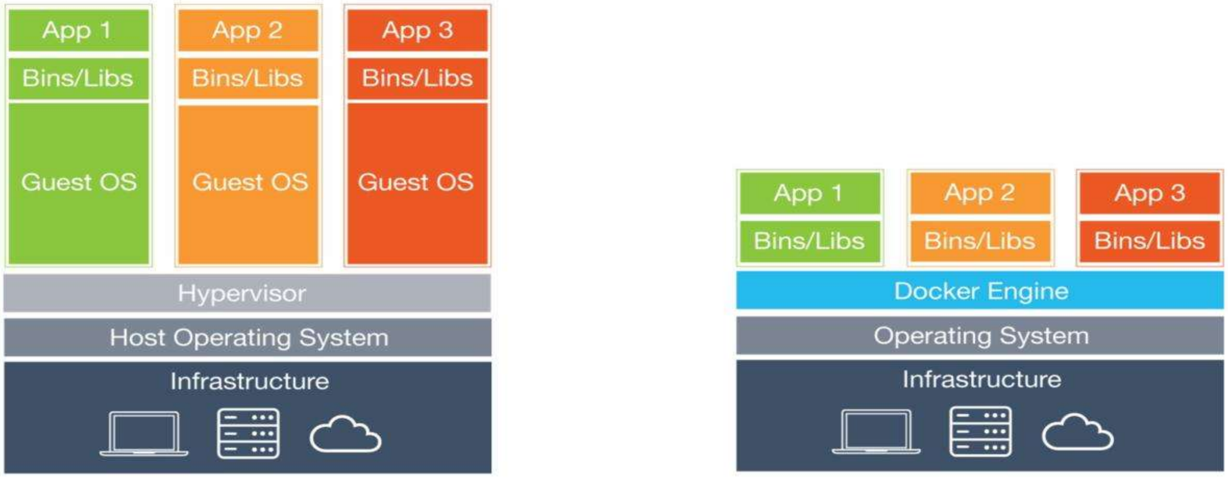
DEPLOYMENT STAGE:-

1. After successful building, the code is finally deployed on AWS using AWS S3.
2. The code is connected to the Github Repository, so it is a continuous integration, continuous deployment pipeline (CI/CD)

ARCHITECTURE AND DESIGN

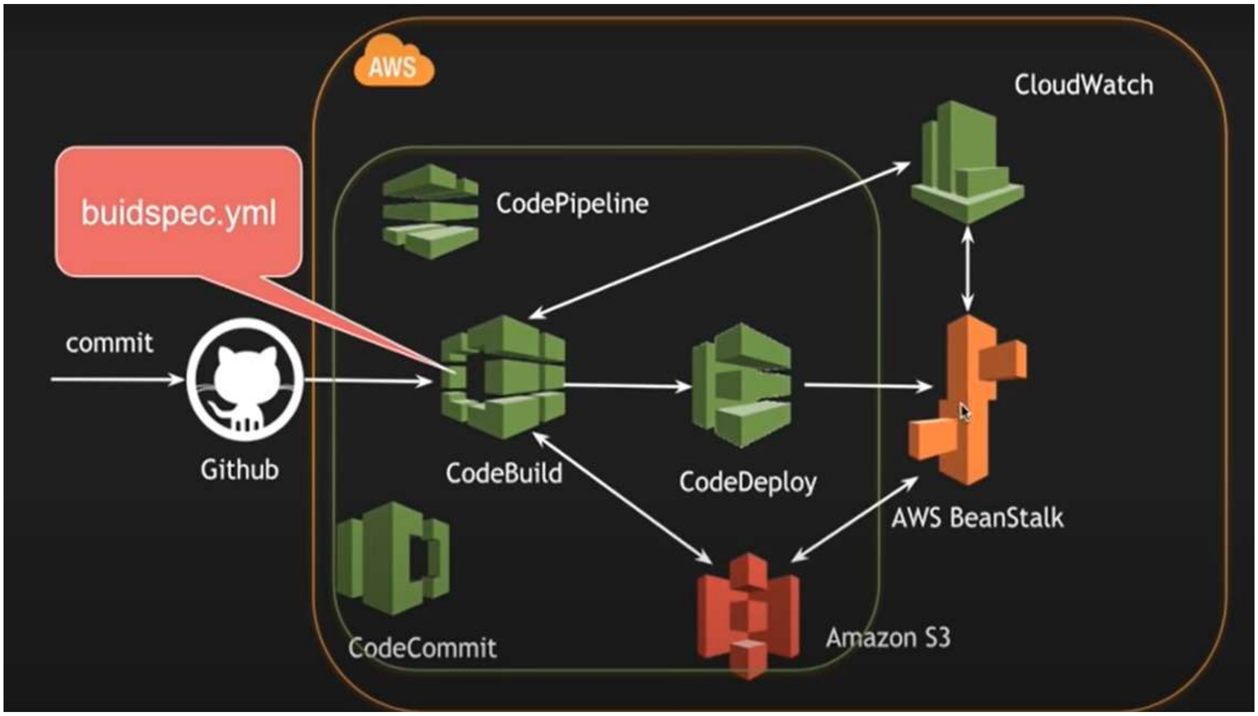


<u>Some examples of Microservices based architecture companies are Netflix, Uber, Swiggy etc.</u>



Hypervisor and Docker Containers

- ★ While learning to make the CI/CD pipeline, we got to know that there are several ways of doing so and that Amazon itself provides multiple services for doing so in AWS. What type of service to choose depends upon our application. Whether it is a purely React-based application or a Node application etc.



CI/CD using Elastic Beanstalk along with AWS S3.

OUTCOME

After setting up all the softwares required for our project, we built the code through various stages. We started with discussing the idea, then production and then towards making our CI/CD pipeline using AWS. We managed the project day-to-day tasks and assignments using an automated Github Project template. Deployment goes through various stages from building the image to running an application.The result of this project is that we have now created a Microservices web application using MERN (Mongodb, Express, React and Node) Stack. We learnt the use of APIs by making a Rest API for connecting two different services. For example, the frontend to the backend service. We learnt several new technologies and concepts like Docker and DevOps respectively. In this project, we created a basic Railway ticket booking service using all concepts which we have mentioned above.

We achieved the aim to successfully perform Continuous Integration and Deployment of a Microservices-based Web Application using various web technologies.

REFERENCES

1. React Router: Declarative Routing for React.js
2. www.medium.com
3. Build and run your image | Docker Documentation
4. https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html
5. https://mherman.org/blog/dockerizing-a-react-app/
6. Create Cluster | Atlas: MongoDB Atlas
7. Orientation and setup | Docker Documentation
8. Getting Started – React (reactjs.org)
9. Amazon Simple Storage Service Documentation
10. Documentation | Node.js