School of Architecture Computing and Engineering

CN7026 – Cloud Computing Report

ON

TRAVEL BUS

PRESENTED TO

GAURAV MALIK

Module Leader

BY

SIVA KRISHNA MANDALAPU

(U2014107)

At

University of East London

Docklands Campus, University Way, London E162RD

May 2021

TABLE OF CONTENT

1. Introduction

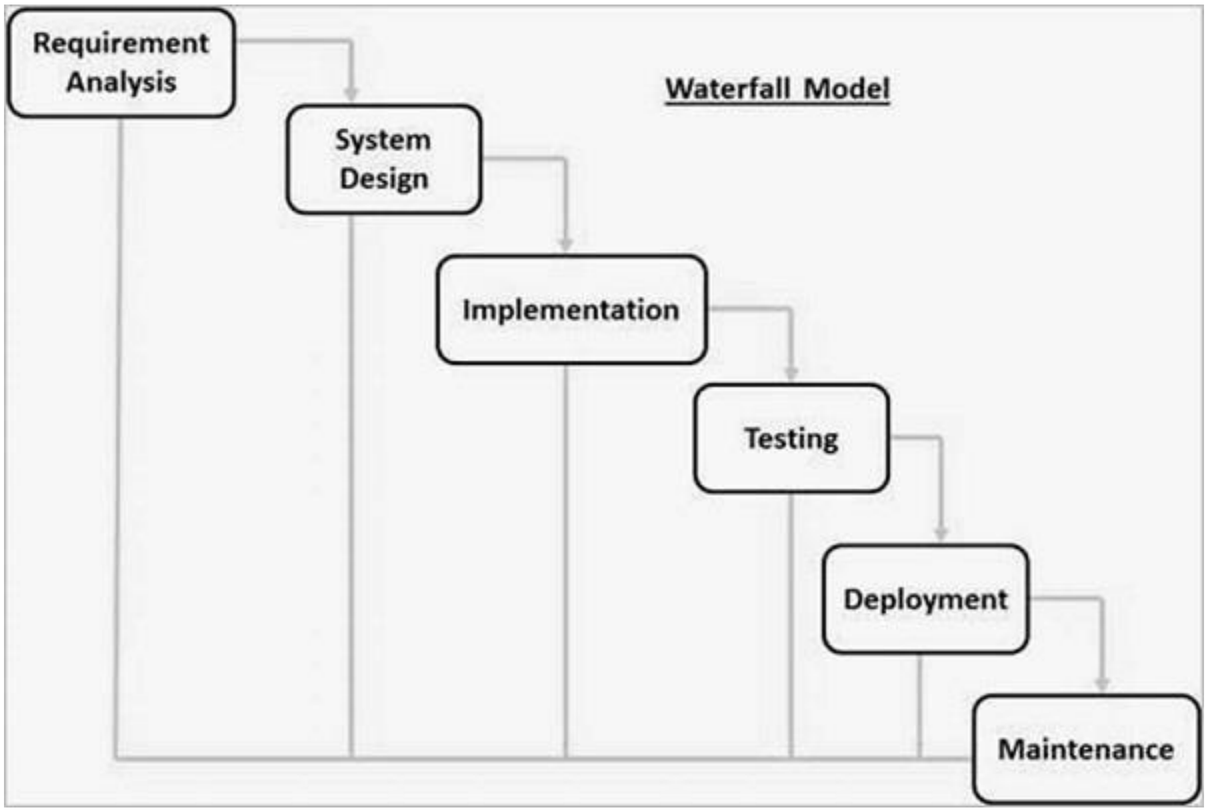
Travel Bus is a cloud-based web application that allows users to book bus tickets from different travelers to different destinations. Some of the main features of this application are comparing ticket fares from different travel agencies with help of this feature users can book tickets within their budget. A tracking system allows users to track bus route, arrival time. Users can select their seats at the time of booking the ticket, with this application users not only book the tickets they have a chance to cancel their tickets at any point of the time. The main aim of this application is to eliminate the manual ticket booking process.

The existing application runs on on-premises servers there are many problems with choosing the on-premises infrastructure. The first disadvantage is initial setup costs i.e Buying a stack of servers costs heavily. The second thing is we have to monitor servers to avoid downtime issues for that we need extra manpower. The third problem is disturbing requests among the servers. Due to this, we are unable to focus on our business.

In the proposed system all our infrastructure moves to the AWS cloud because of this we can overcome all the disadvantages of the existing system. The first thing is AWS uses pay as you go which means we have to pay how many hours we run servers that many hours only have to pay, with help of this feature we can overcome the first disadvantage. The second feature is auto-scaling the main feature is when ever our server facing a downtime issue it will add a new server and divert load to that server with help of this we can overcome downtime issues. Cloud Watch is another feature that takes care of monitoring all the resources deployed in our infrastructure.

2. Project plan

For this project, I am choosing the waterfall model as a development life cycle. The main reasons to choose this model are simple and easy to use and understand. Best suitable for small projects. Every stage is defined or explained clearly. Well understood milestones.



SNO	PROJECT PHASE	TIME ALLOCATED
1	Requirement analysis	Week1
2	System design	Week2
3	Implementation/coding	Week3
4	Testing Phase	Week4
5	Deployment	Week5
6	Maintenance	Week6

REQUIREMENT ANALYSIS: All the possible developed requirements are gathered in this phase.

SYSTEM DESIGN: The requirements gathered in the first phase are studied in this phase and the system is designed according to the requirements.

IMPLEMENTATION: With the inputs from the design phase the system is developed in this phase.

TESTING: The system developed in the implementation phase is tested in this phase.

DEPLOYMENT: Once all the functional and non-functional testing is done system is developed in the customer environment.

MAINTENANCE: Some issues come up in the client environment to fix those issues patches are released in this phase.

3. Requirement gathering

3.1 Functional requirements:

Functional requirements are the requirements that specify WHAT THE SYSTEM SHOULD DO in below I am listing functional requirements of my TRAVEL BUS application.

SNO	REQUIREMENTS	PRIORITY
1	Login	High
2	Enter the Date, Source, Destination details	High
3	Check the availability	High
4	Show available buses along with fares	High
5	Select bus	High
6	Select number of seats	High
7	Generates total cost	High
8	Review ticket	High
9	Make the payment	High
10	Displays the ticket	High

3.2 Non Functional Requirements:

Non Functional Requirements will describe how the system should behave and what the limits on its functionality are. Below I am listing some of the non-functional requirements

- Performance
- Security
- Usability
- Data integrity

4. Selection of cloud platform

For the implementation of the idea selected for the project, the AWS cloud service platform, which is an IAAS/PAAS-based service solution which is used here. The advantage of using in terms of security AWS utilizes an end-to-end approach to secure and harden our infrastructure, including physical, operational, and software measures. For Scalable and high performance, using AWS tools, Auto Scaling, and Elastic Load balancing we can make our application scale up and down based on demand. To make it cost-effective we can pay only for the compute power, storage, and other resources you use with no long-term contracts and up-front commitments. To make it flexible AWS enables you to select the operating system, programming language, web application platform, database, and other services you need.

5. CHOICE OF DATA CENTER AND STANDARDS

Data centers are simply a collection of servers to collect, store, processing, distributing, or allowing access to a large amount of data they provide important services like data storage, backup and recovery, data management, and networking.

For this project, we are choosing tire3 data centers standards why because tire3 data centers are located with redundant and dual powered servers, storage, network links, and other IT equipment. It is one of the most commonly used data centers where IT components are powered with multiple active and independent sources of power cooling resources. With a redundant and always active power supply there is minimal planned and unplanned downtime. It guarantees 99.82 percent of availability with a fractionally less than two hours of downtime per year.

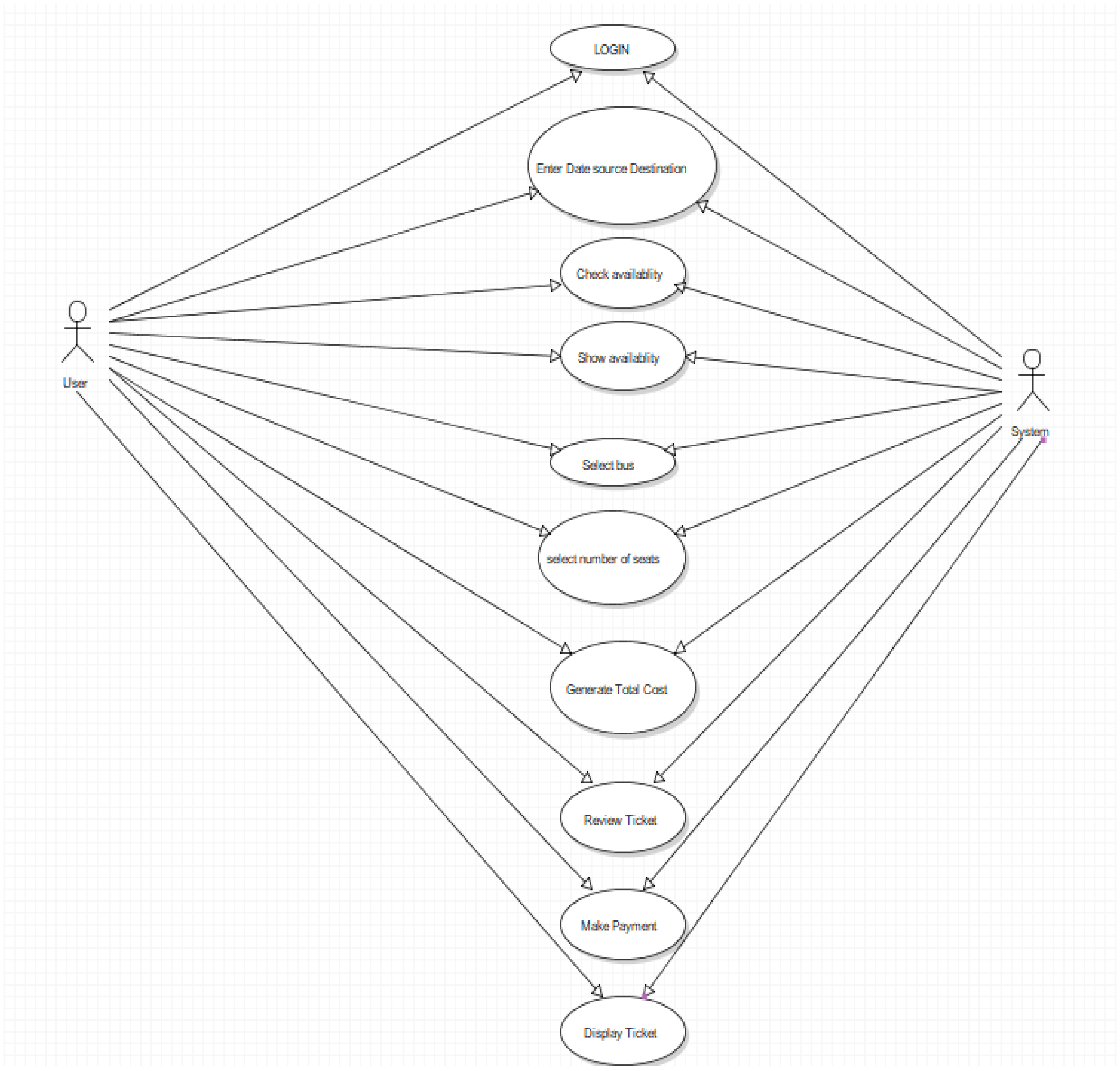
6. THE ARCHITECTURE OF THE CLOUD SYSTEM

6.1 BEST PRACTICES FOLLOWED:

- The Excellence Of Operation: The design of the solution provides a quick response which will be a business need for the customers who wish to book the tickets using this application.
- Security: Every resource used in the architecture provides the best security using SSL certifications and Authentication so we can even integrate third-party modules.
- Reliability: The design system is capable of recovery and horizontal scaling is also implemented.
- Performance Efficiency: The choice design is reviewed regularly to evaluate the nature of the AWS cloud through this the performance of the system can be enhanced.
- Cost Optimization: More managed services are used to reduce the cost of ownership. To save the cost usage of the appropriate resource must be ensured.

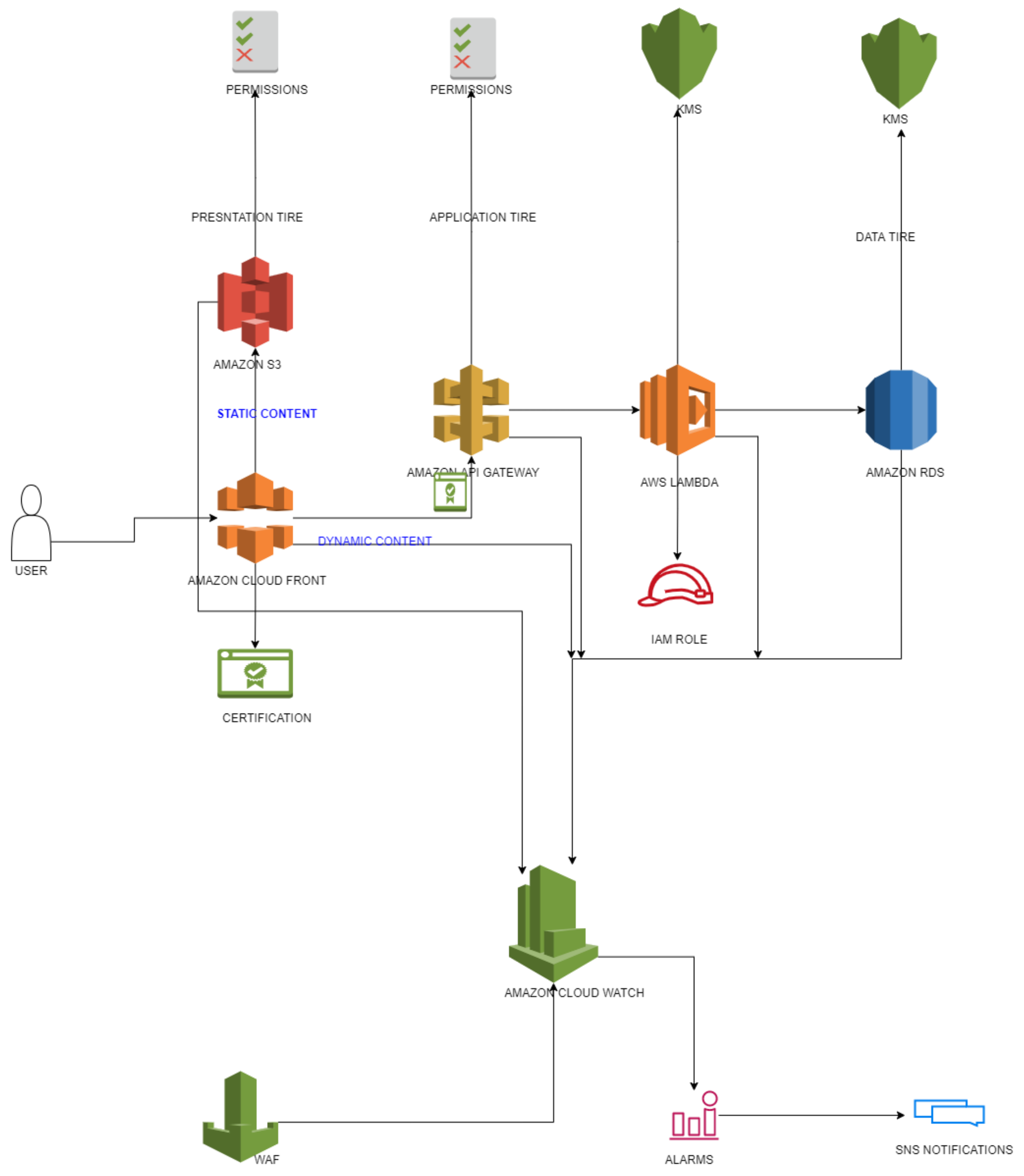
6.2 SYSTEM DESIGN:

The below UML diagram explains how the system will works.



6.3 SYSTEM ARCHITECTURE:

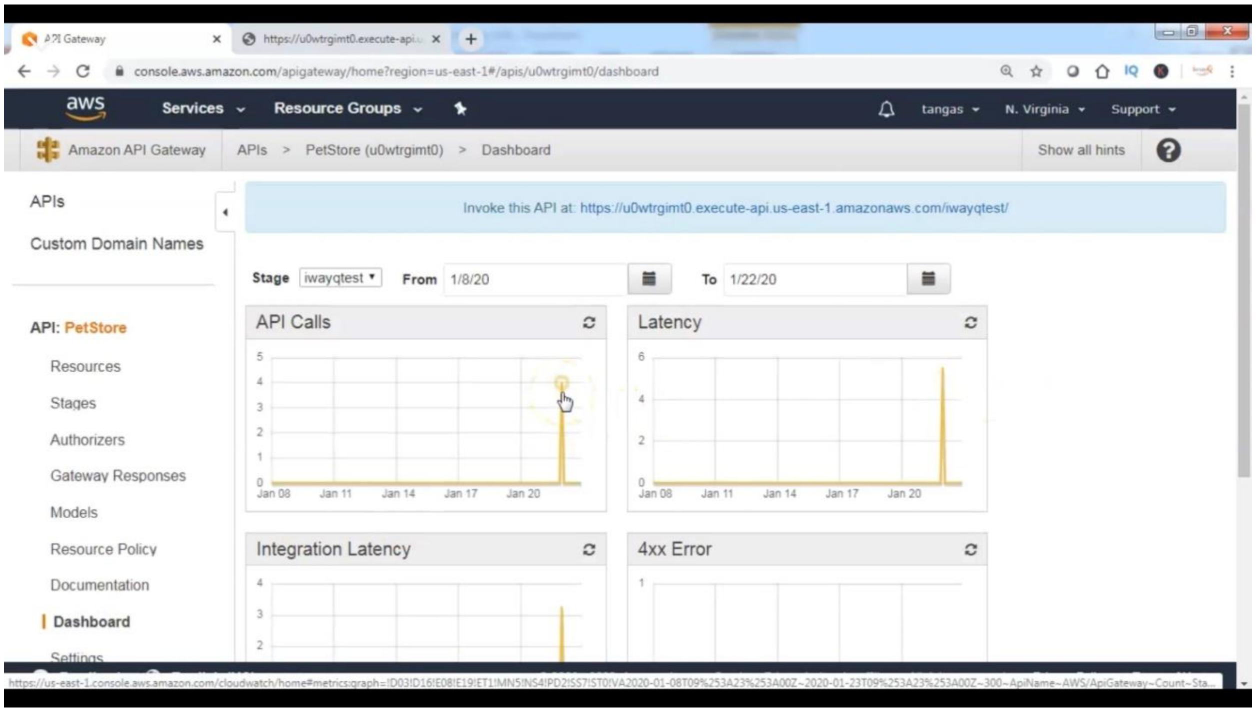
The system design architecture is given as below:



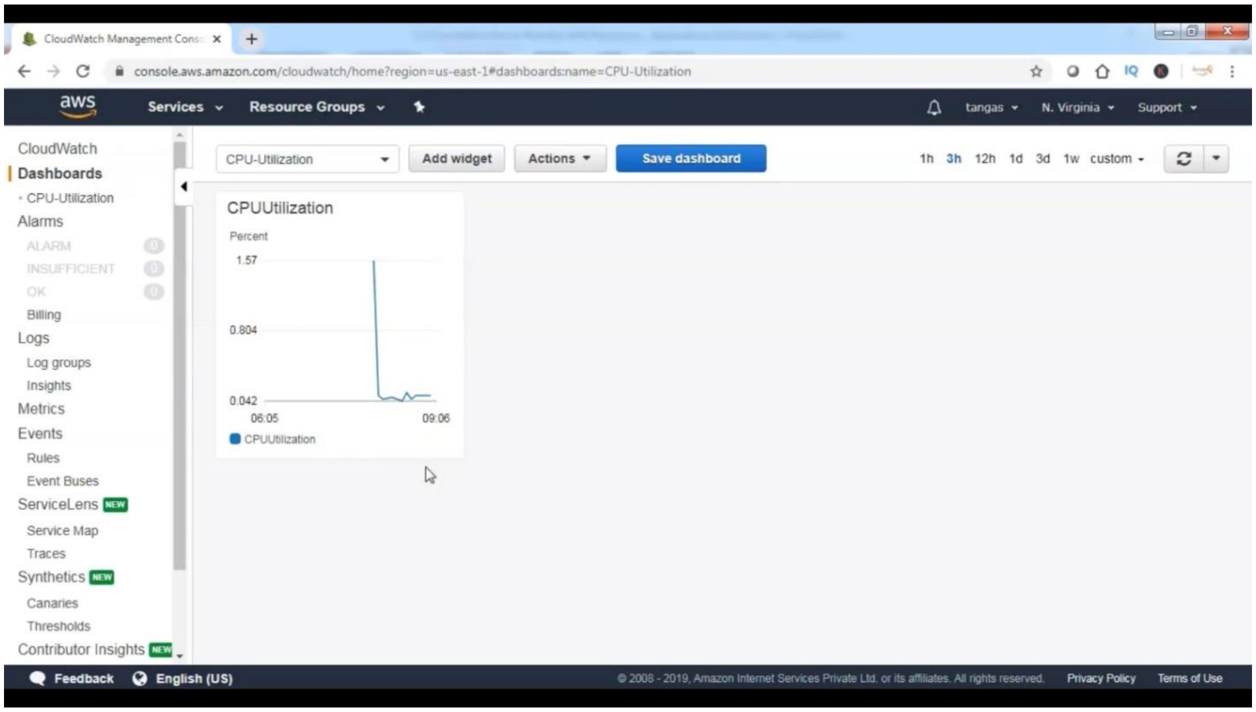
EXPLANATION OF ARCHITECTURE:

The entire application has divided into three tires Presentation tire where the end-user goes and interacts with the application. The second Application tire will collect the inputs from the users and process those requests stores into databases. Third Database tire stores all the business and users' information. To provide security SSL certificates, permissions, KMS, IAM roles are assigned to every layer based on their requirement. Cloud watch is the monitoring service that sets alarms if any server facing issues like downtime, storage then sent emails to the appropriate team with help of Simple Notification services.

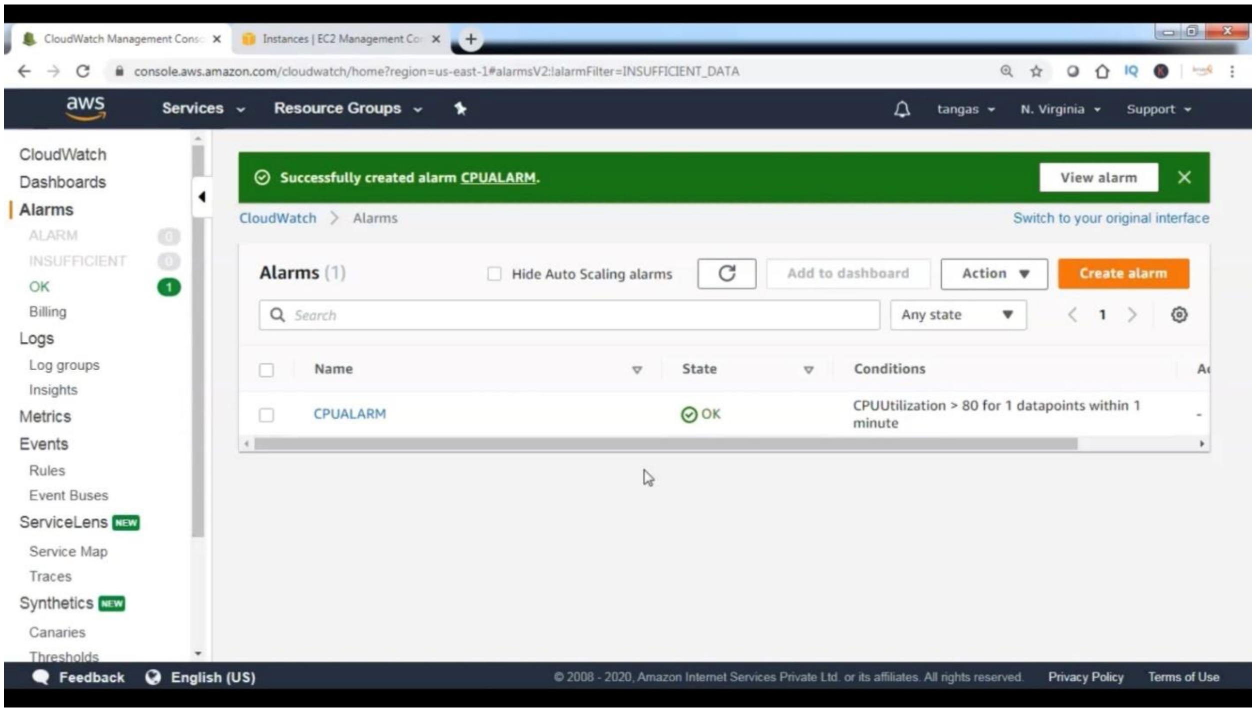
7. IMPLEMENTATION OF ARCHITECTURE IN AWS CLOUD



AMAZON API Showing number of times API called and latency, integrity, errors.



AMAZON CLOUD WATCH monitoring CPU utilization of EC2 instance



Successfully created alarm CPUALARM.

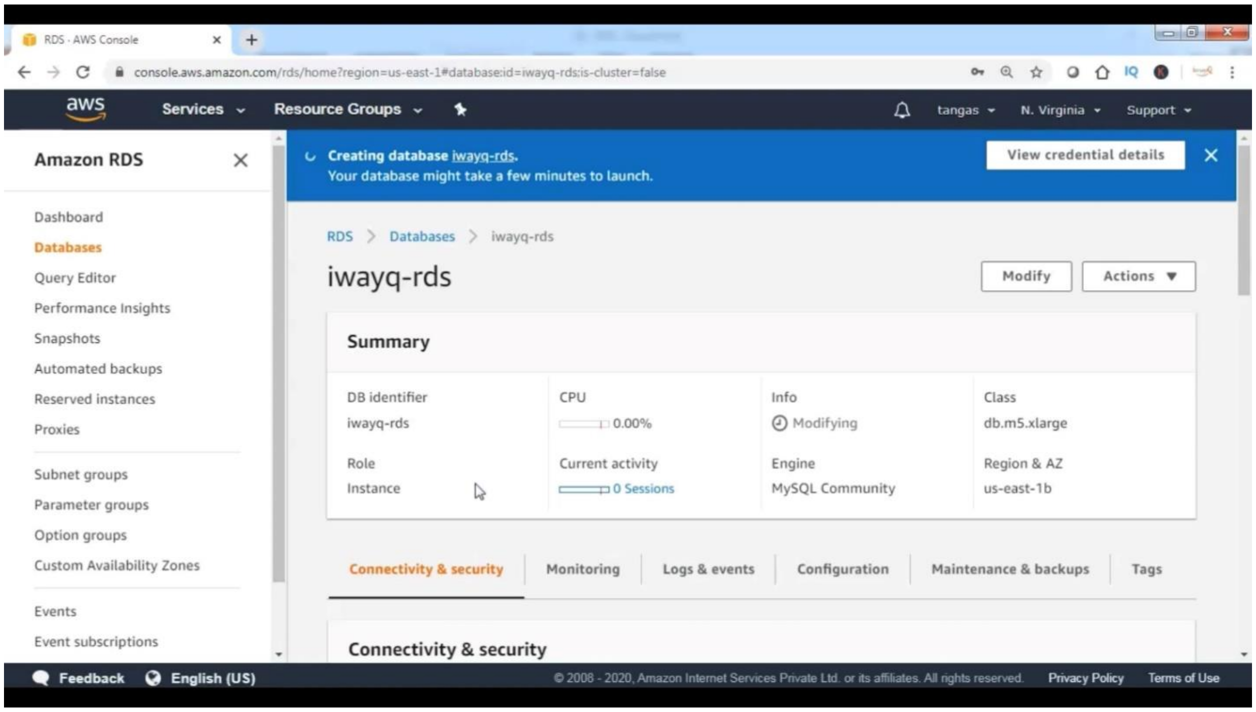
CloudWatch > Alarms

Alarms (1)

Name	State	Conditions
CPUALARM	OK	CPUUtilization > 80 for 1 datapoints within 1 minute

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

ALARMS created for CPU utilization goes beyond the threshold limit alarm starts



Creating database iwayq-rds.
Your database might take a few minutes to launch.

RDS > Databases > iwayq-rds

iwayq-rds

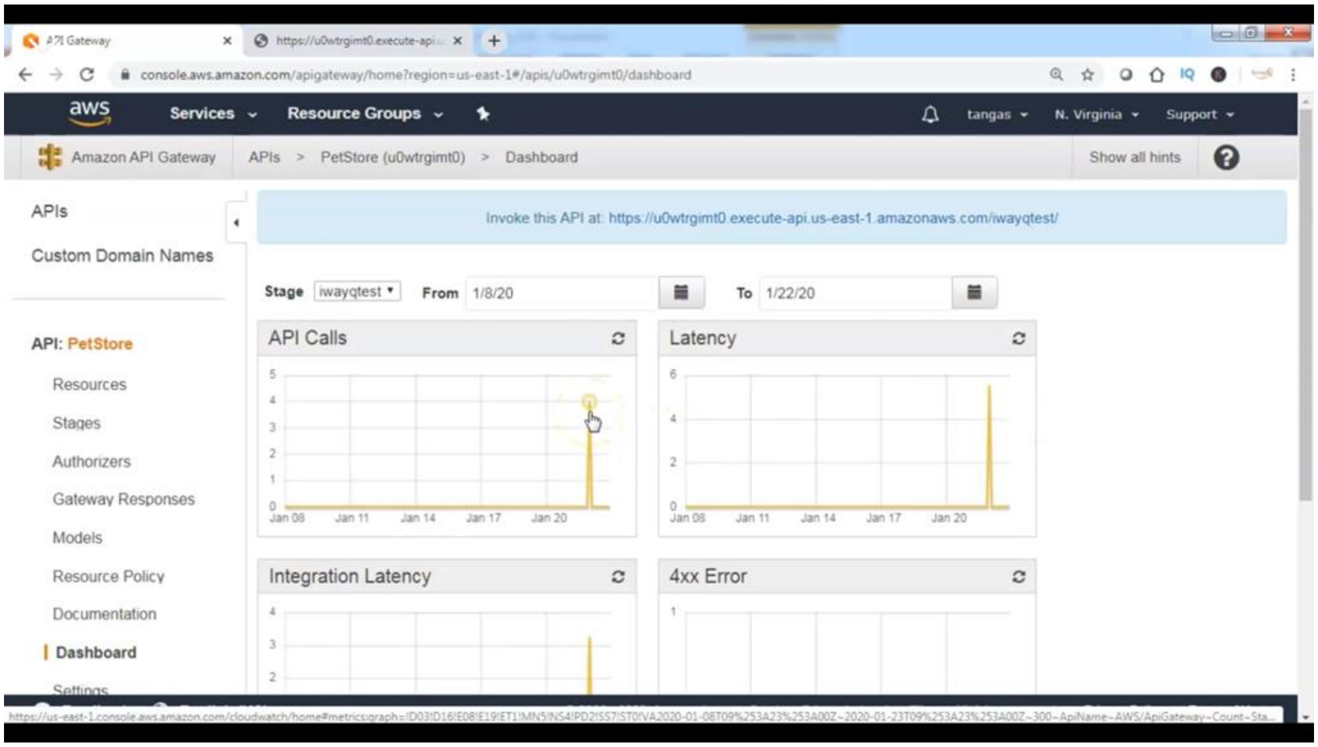
Summary

DB identifier	CPU	Info	Class
iwayq-rds	0.00%	Modifying	db.m5.xlarge
Role	**Current activity**	**Engine**	**Region & AZ**
Instance	0 Sessions	MySQL Community	us-east-1b

Connectivity & security

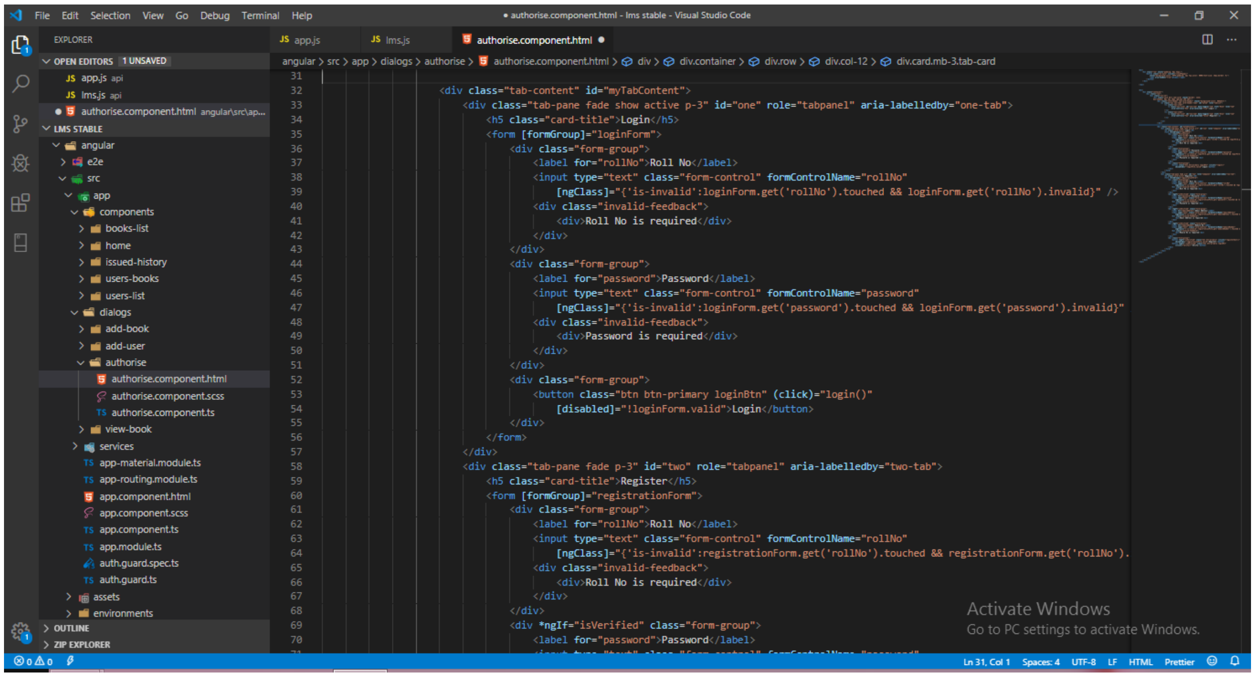
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

AMAZON RDS is created for storing and processing our business data.



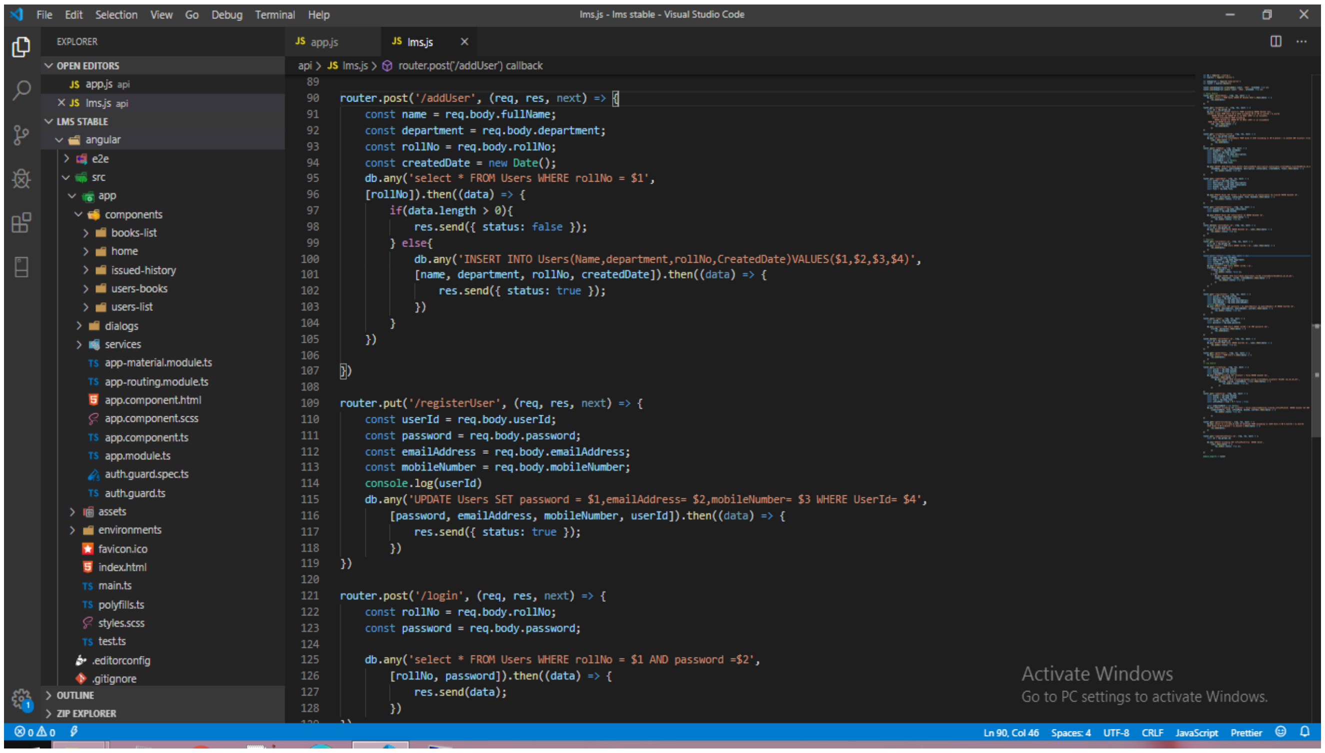
CODING SCREENSHOTS:

Below you can observe some of the coding used to build the application.



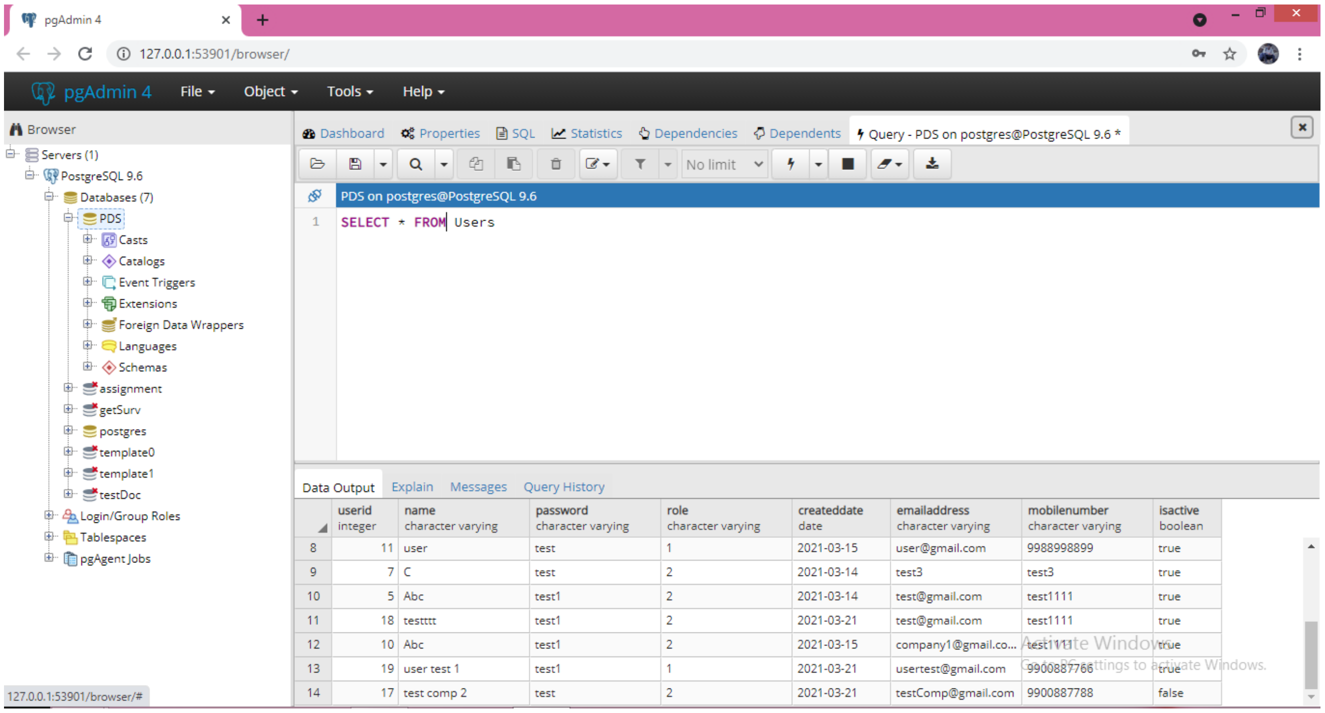
```html

                    <div class="tab-content" id="myTabContent">
                        <div class="tab-pane fade show active p-3" id="one" role="tabpanel" aria-labelledby="one-tab">
                            <h5 class="card-title">Login</h5>
                            <form [formGroup]="loginForm">
                                <div class="form-group">
                                    <label for="rollNo">Roll No</label>
                                    <input type="text" class="form-control" formControlName="rollNo"
                                        [ngClass]="{'is-invalid':loginForm.get('rollNo').touched && loginForm.get('rollNo').invalid}" />
                                    <div class="invalid-feedback">
                                        <div>Roll No is required</div>
                                    </div>
                                </div>
                                <div class="form-group">
                                    <label for="password">Password</label>
                                    <input type="text" class="form-control" formControlName="password"
                                        [ngClass]="{'is-invalid':loginForm.get('password').touched && loginForm.get('password').invalid}"
                                    <div class="invalid-feedback">
                                        <div>Password is required</div>
                                    </div>
                                </div>
                                <div class="form-group">
                                    <button class="btn btn-primary loginBtn" (click)="login()"
                                        [disabled]="!loginForm.valid">Login</button>
                                </div>
                            </form>
                        </div>
                        <div class="tab-pane fade p-3" id="two" role="tabpanel" aria-labelledby="two-tab">
                            <h5 class="card-title">Register</h5>
                            <form [formGroup]="registrationForm">
                                <div class="form-group">
                                    <label for="rollNo">Roll No</label>
                                    <input type="text" class="form-control" formControlName="rollNo"
                                        [ngClass]="{'is-invalid':registrationForm.get('rollNo').touched && registrationForm.get('rollNo').
                                    <div class="invalid-feedback">
                                        <div>Roll No is required</div>
                                    </div>
                                </div>
                                <div *ngIf="isVerified" class="form-group">
                                    <label for="password">Password</label>
```

Code for login and register for new users and existing users.



```
router.post('/addUser', (req, res, next) => {
    const name = req.body.fullName;
    const department = req.body.department;
    const rollNo = req.body.rollNo;
    const createdDate = new Date();
    db.any('select * FROM Users WHERE rollNo = $1',
    [rollNo]).then((data) => {
        if(data.length > 0){
            res.send({ status: false });
        } else{
            db.any('INSERT INTO Users(Name,department,rollNo,CreatedDate)VALUES($1,$2,$3,$4)',
            [name, department, rollNo, createdDate]).then((data) => {
                res.send({ status: true });
            })
        }
    })

})

router.put('/registerUser', (req, res, next) => {
    const userId = req.body.userId;
    const password = req.body.password;
    const emailAddress = req.body.emailAddress;
    const mobileNumber = req.body.mobileNumber;
    console.log(userId)
    db.any('UPDATE Users SET password = $1,emailAddress= $2,mobileNumber= $3 WHERE UserId= $4',
        [password, emailAddress, mobileNumber, userId]).then((data) => {
            res.send({ status: true });
        })
})

router.post('/login', (req, res, next) => {
    const rollNo = req.body.rollNo;
    const password = req.body.password;

    db.any('select * FROM Users WHERE rollNo = $1 AND password =$2',
        [rollNo, password]).then((data) => {
            res.send(data);
        })
```

REST API code for establishing a connection between frontend and backend databases.



```
SELECT * FROM Users
```

	userid integer	name character varying	password character varying	role character varying	createddate date	emailaddress character varying	mobilenumber character varying	isactive boolean
8	11	user	test	1	2021-03-15	user@gmail.com	9988998899	true
9	7	C	test	2	2021-03-14	test3	test3	true
10	5	Abc	test1	2	2021-03-14	test@gmail.com	test1111	true
11	18	testttt	test1	2	2021-03-21	test@gmail.com	test1111	true
12	10	Abc	test1	2	2021-03-15	company1@gmail.co...	test1111	true
13	19	user test 1	test1	1	2021-03-21	usertest@gmail.com	9900887766	true
14	17	test comp 2	test	2	2021-03-21	testComp@gmail.com	9900887788	false

Database query to see the registered users in our application.

8. ANALYSIS AND REFLECTION

This project is completed by 70% and the remaining 30% is scheduled for the next week. Through effective implementation following things are achieved.

The booking process of the customers is online. An Amazon RDS is used for storing the data of customers and buses information. To make responses much faster we are using the Amazon API gateway to reduce the downtime and making architecture serverless we are using AWS LAMBDA and auto-scaling groups. To make the application more secure we are using SSL certifications, IAM roles, KMS guards, permissions are used. For monitoring the entire application infrastructure CLOUD WATCH is used and alarms are set if anything went wrong SNS notifications service sends mail to respective teams.

Infrastructure is easy to operate and an attractive system is designed customers can get an interactive interface mostly automated to get proper assistance and submit their feedback.

9. REFERENCES

Capello, E., Dentis, M., Guglieri, G., Mascarello, L.N. and Cuomo, L.S., 2017. An Innovative Cloud-based Supervision System for the Integration of RPAS in Urban Environments. Transportation Research Procedia, 28, pp.191-200.

The APP solutions. (2020). FUNCTIONAL VS NON-FUNCTIONAL REQUIREMENTS: MAIN DIFFERENCES & EXAMPLES.
https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/

Online article
adolfogracia.com, 2020, Fintech: Choosing a Cloud Services Provider, available at: http://www.adolfogracia.com/documents/PhillipsExecSum.pdf [Accessed on 26.5.2020]

datacenterknowledge.com, 2020, Data Center Design: Which Standards to Follow? available

at: https://www.datacenterknowledge.com/archives/2016/01/06/data-center-design-which-standards-to-follow.