

Team 7

Team members

Ashrit Deebadi

Riti Gupta

Vedashree Bhandare

## END TO END CLOUD BASED E-COMMERCE WEBSITE

### 1. INTRODUCTION

The objective of this project is to develop a general-purpose e-commerce store where product like laptops, mobile phones can be bought from the comfort of home through the Internet. eCommerce websites are online portals that facilitate online transaction of goods and services through means of Internet. We have leveraged existing AWS services to handle scalability, availability and other design considerations. Our application is a SaaS model following multi tenancy where multiple users can simultaneously access the application. Any revisions to the application would be available to all the users of the application instantaneously.

#### 1.1 Previous works

To understand the work done in this area, we read few research papers and developed the project to make the services available to the customers. Lot of work has been done to serve consistency based on user needs. Also, the focus has been to minimize IT costs and resources required for the development. The customer should face minimum latency and the services should be readily available meeting all service agreement clauses.

#### References :

[1] F.Shaikh and D.Patil, "Multi-tenant ecommerce based on SaaS model to minimize IT cost", in *IEEE Conf. on Adv. in Eng. and Tech. Research*, 2014.

[2] A.G.Recuerdo, S.Esteves and L.Veiga, "Quality-of-data for consistency levels in geo-replicated cloud datastores", in *IEEE 5th Conf. on Cloud Computing Tech. and Science*, 2013.

#### 1.2 Intended users or service consumers

General public who is a registered user can use the website and place order for the available products.

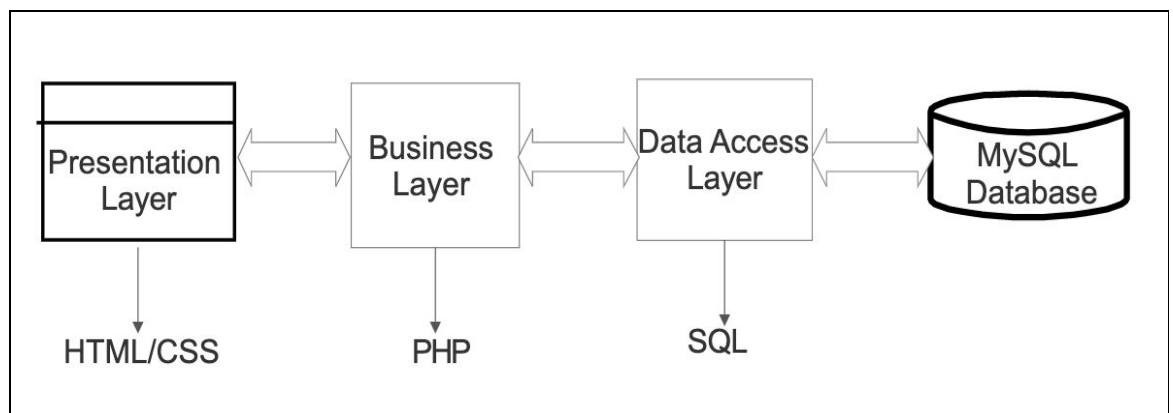
### 1.3 List of functionalities/operations

- ❖ Any member can sign up and create a new account.
- ❖ Any registered user can login/logout.
- ❖ User with existing account can reset password.
- ❖ Registered users can view available products.
- ❖ Only registered member can see the details about the products.
- ❖ Users are provided with hashed password for additional security.
- ❖ Search through catalog.
- ❖ Send marketing emails.
- ❖ Checkout.

### 1.4 Final major areas/components/tasks

Our application is based on client server-based web application and has three major components. We initially worked on deciding the components and services to be used. We designed a architecture diagram to understand the flow of data. Ashrit started working on setting up policy for Auto Scaling and creating VPC and security groups. Riti and Vedashree worked on setting up EC2 instances, S3 buckets and Database related part. We further divided the front end development among ourselves. Most of the parts of project were completed by April 30. We kept around one week for testing. Everyone in the team was involved in testing.

This three-tier architecture and can be broken down into three logical tiers: the presentation layer, the business logic layer and the data storage layer. Below is the simplified diagram for the same



## 1.5 Project URL

[www.ashritdeebadi.com](http://www.ashritdeebadi.com)

## 2. OVERALL DESCRIPTION

User can view Online Shopping portal and available products, but every user must login by his/her Username and password in order to do the same. Unregistered members can register by navigating to registration page. Once user registers site, his default role will be 'User'. User can look at the different product list and their details by logging into the system.

## 3. DESIGN

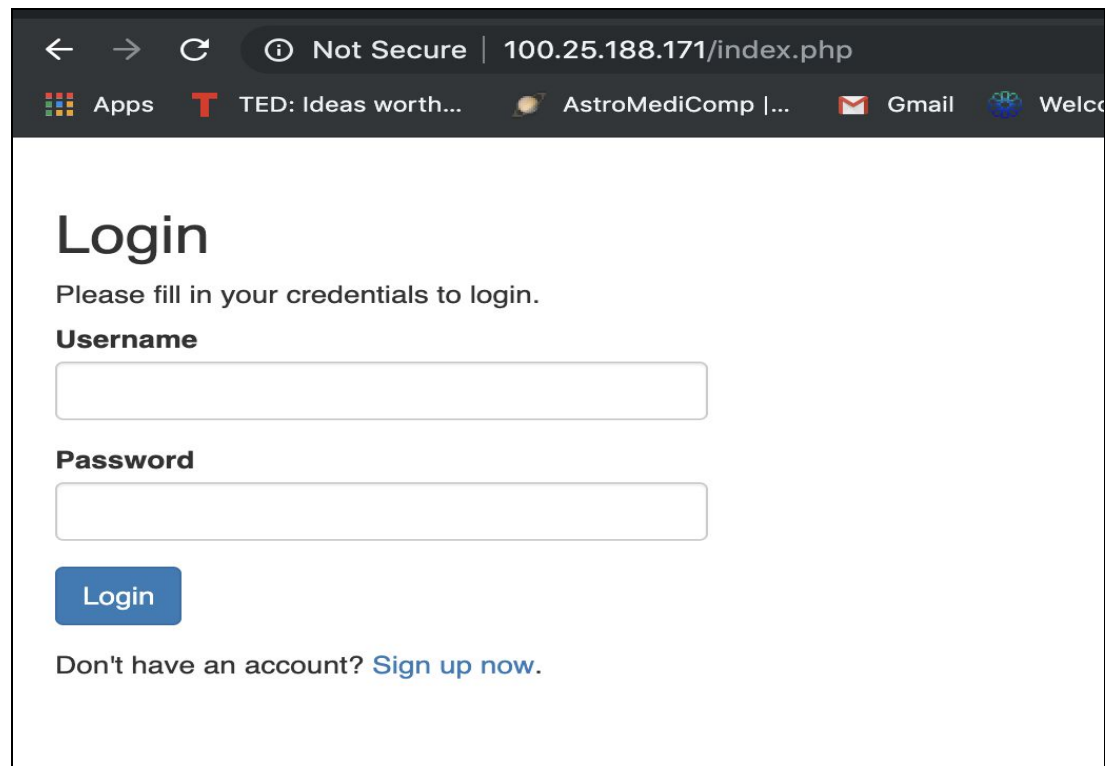
In order to maintain an e-commerce website with a large product catalog and a global consumer base can be a challenge. So, for our project we have leveraged existing cloud services offered by Amazon.

Following are the components of our three-layered architecture

### 3.1. Components

**3.1.1. Frontend and Business Logic:** We created web pages using HTML/CSS and integrated them using business logic implemented in php. Following different pages are created as part of our e-commerce site which basically contains catalog of electronic products.

**Login Page** : User can login using the username and the password that he/she provided during registration.



← → ↻ ⓘ Not Secure | 100.25.188.171/index.php

Apps TED: Ideas worth... AstroMediComp |... Gmail Welco...

# Login

Please fill in your credentials to login.

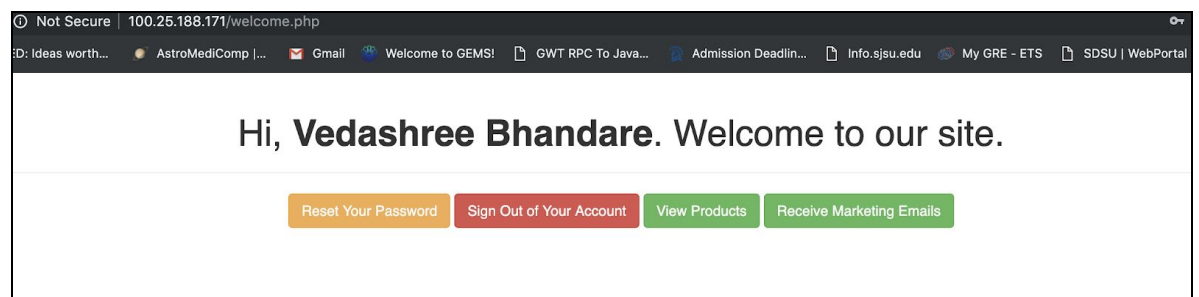
**Username**

**Password**

Login

Don't have an account? [Sign up now.](#)

**Welcome Page** : This page is used to display different options to user after he/she has logged in.



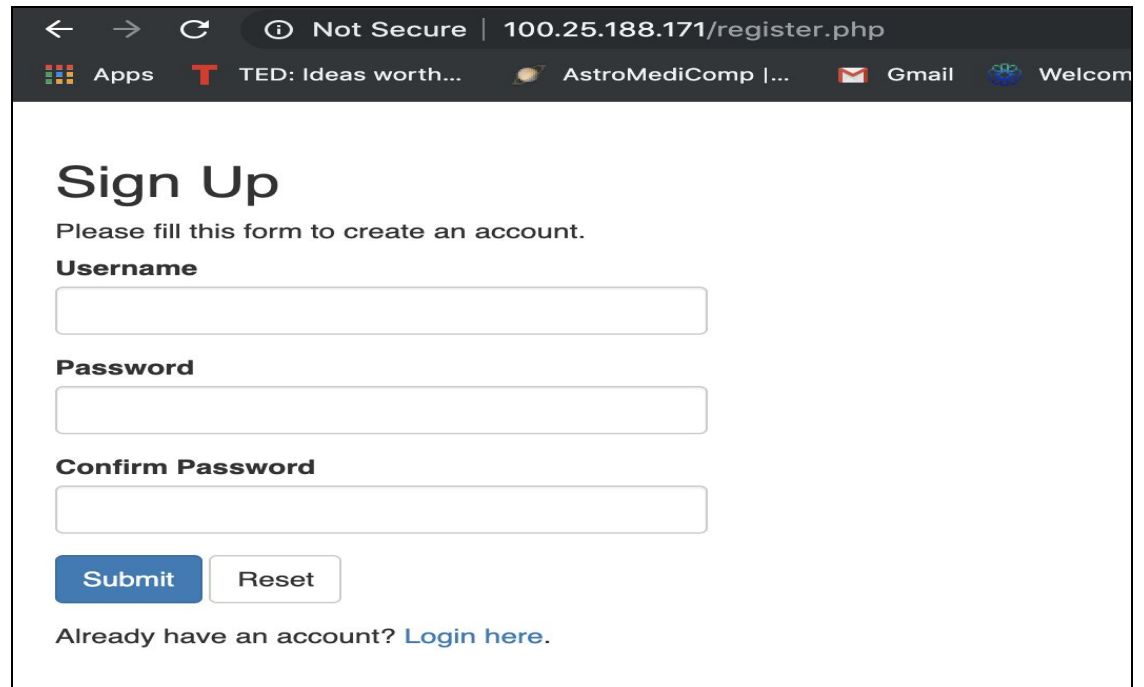
ⓘ Not Secure | 100.25.188.171/welcome.php

ID: Ideas worth... AstroMediComp |... Gmail Welcome to GEMS! GWT RPC To Java... Admission Deadlin... Info.sjsu.edu My GRE - ETS SDSU | WebPortal

## Hi, Vedashree Bhandare. Welcome to our site.

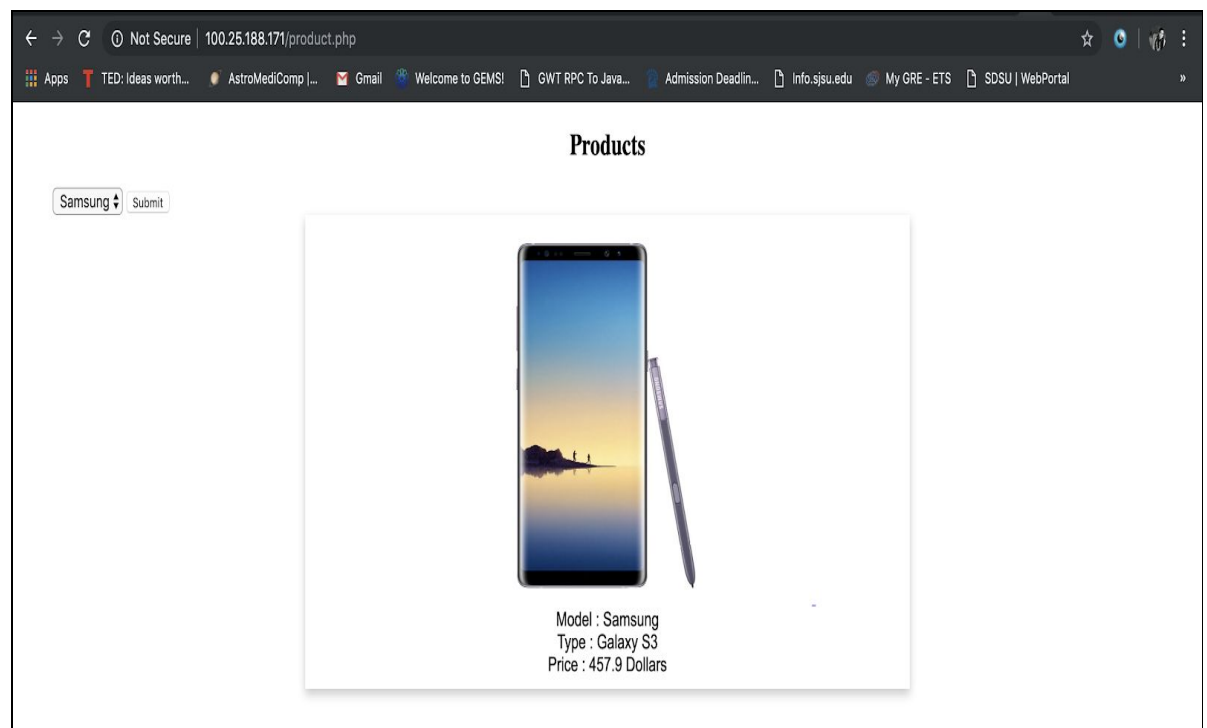
Reset Your Password Sign Out of Your Account View Products Receive Marketing Emails

**Registration Page** : This page is used by new user to create an account.

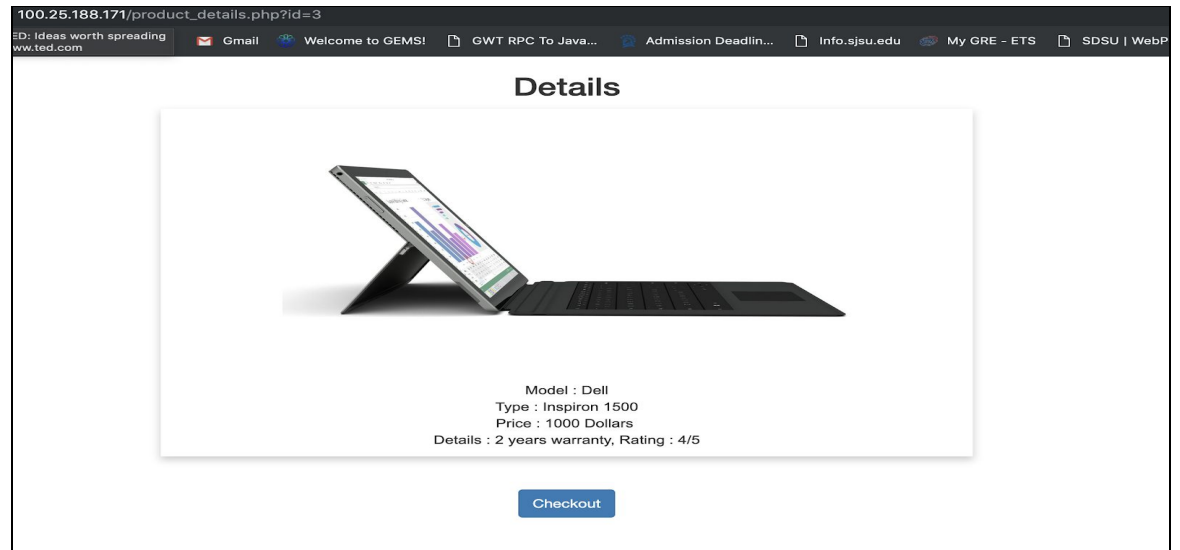


A screenshot of a web browser showing a registration page. The address bar indicates the URL is 100.25.188.171/register.php. The page has a title "Sign Up" and a subtitle "Please fill this form to create an account." Below this, there are three input fields labeled "Username", "Password", and "Confirm Password". At the bottom of the form are two buttons: "Submit" (blue) and "Reset" (white). Below the buttons, there is a link that says "Already have an account? [Login here.](#)"

**Product Page** : This page displays different products that are available for sale.



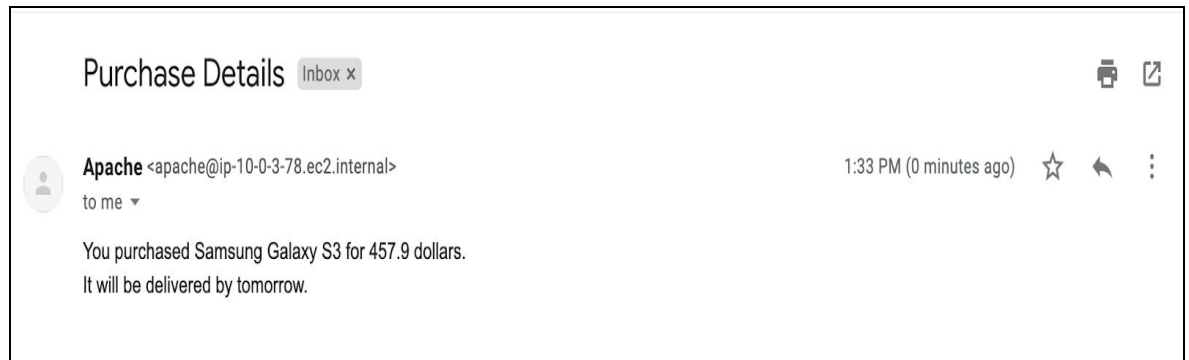
**Product Details Page** : This page displays the information about the product.



**Checkout Page**: After user selects a product and checks it out he will be asked for Credit card details and email address to send confirmation mail.

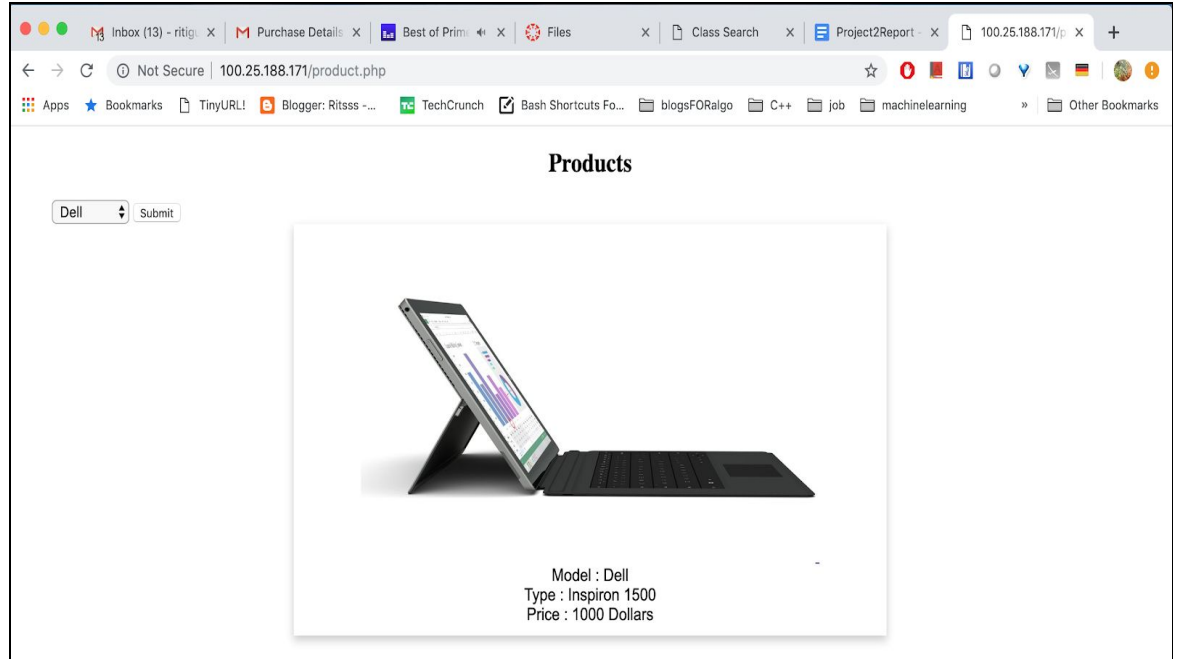
A screenshot of a web browser displaying a checkout page. The browser's address bar shows the URL "100.25.188.171/checkout.php" and indicates the connection is "Not Secure". The page has a dark header with navigation links: "Apps", "TED: Ideas worth...", "AstroMediComp |...", "Gmail", and "Welco". The main content area is titled "Credit Card" and features a large input field for the credit card number. Below this is an "Email" section with a large input field for the email address. At the bottom of the form is a blue "Send mail" button.

After the user enters the Email address, the details sent to the user.



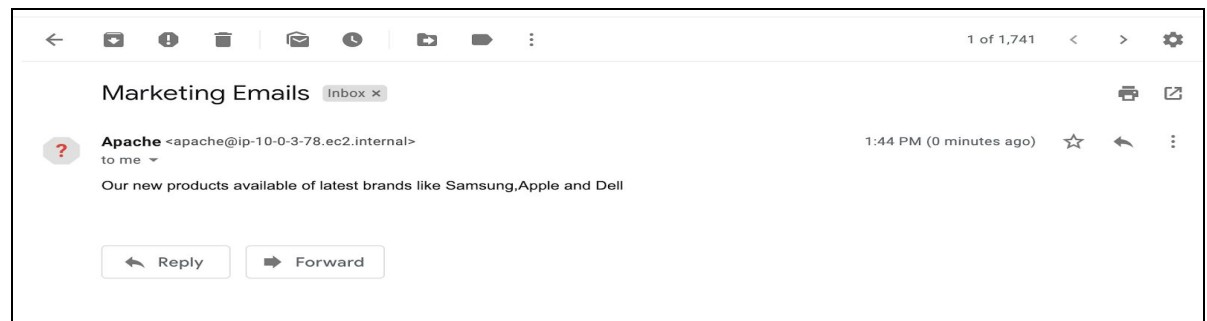
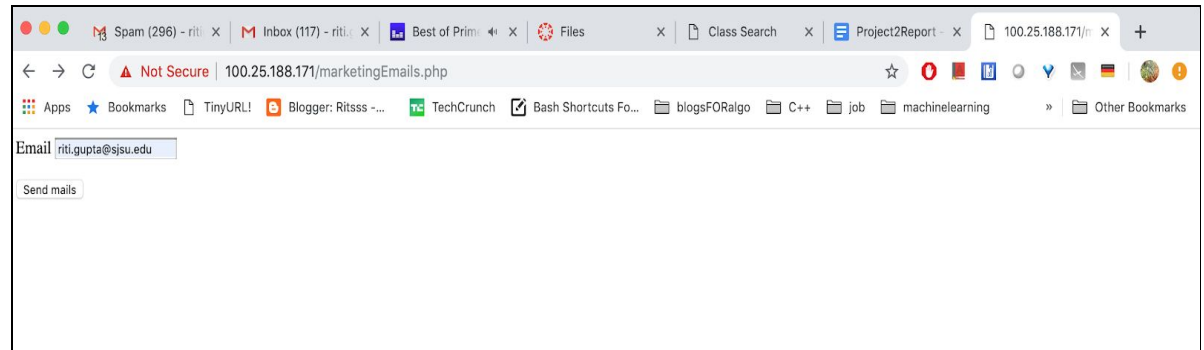
## Product Search Page :

The user also has the option to search the products based on certain search criterias. We have added support for searching based on brands. This can be extended to other criterias as well. In the below example, user has selected Dell.



## Receive Marketing Emails Page:

The user has the option to receive marketing emails. For this project, the marketing mails are sent just once. This can be extended to receive periodically as well.



### 3.1.2. Data Management

We have used MySQL relation database from Amazon. Schema of database is as follow

- ❖ User - Users table consist of user details. Like user id, password and username.
- ❖ Products - Product consist details about the products available on the catlog.



## Data Objects

➤ **Products** - id, model, type, price, details, image.

```
MySQL [ecommerce]> describe products;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO		NULL	
model	varchar(200)	NO		NULL	
type	varchar(60)	NO		NULL	
price	double	NO		NULL	
details	text	NO		NULL	
image	varchar(200)	NO		NULL	

6 rows in set (0.01 sec)

➤ **Users** - id, username, password.

```
MySQL [ecommerce]> describe users;
```

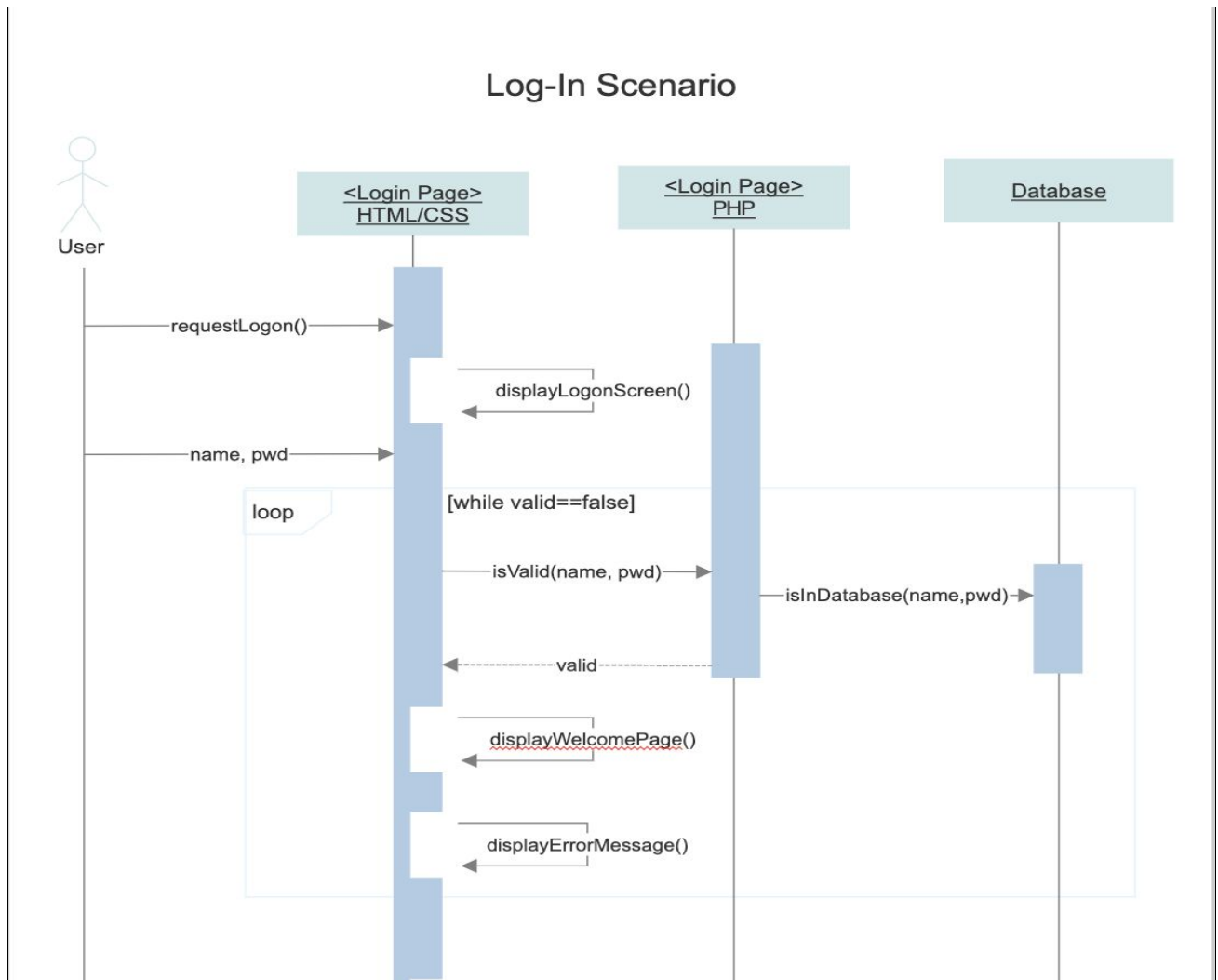
Field	Type	Null	Key	Default	Extra
id	int(11)	NO		NULL	
username	varchar(200)	NO		NULL	
password	varchar(60)	NO		NULL	

3 rows in set (0.00 sec)

### 3.2. Sequence Diagram and Dataflow.

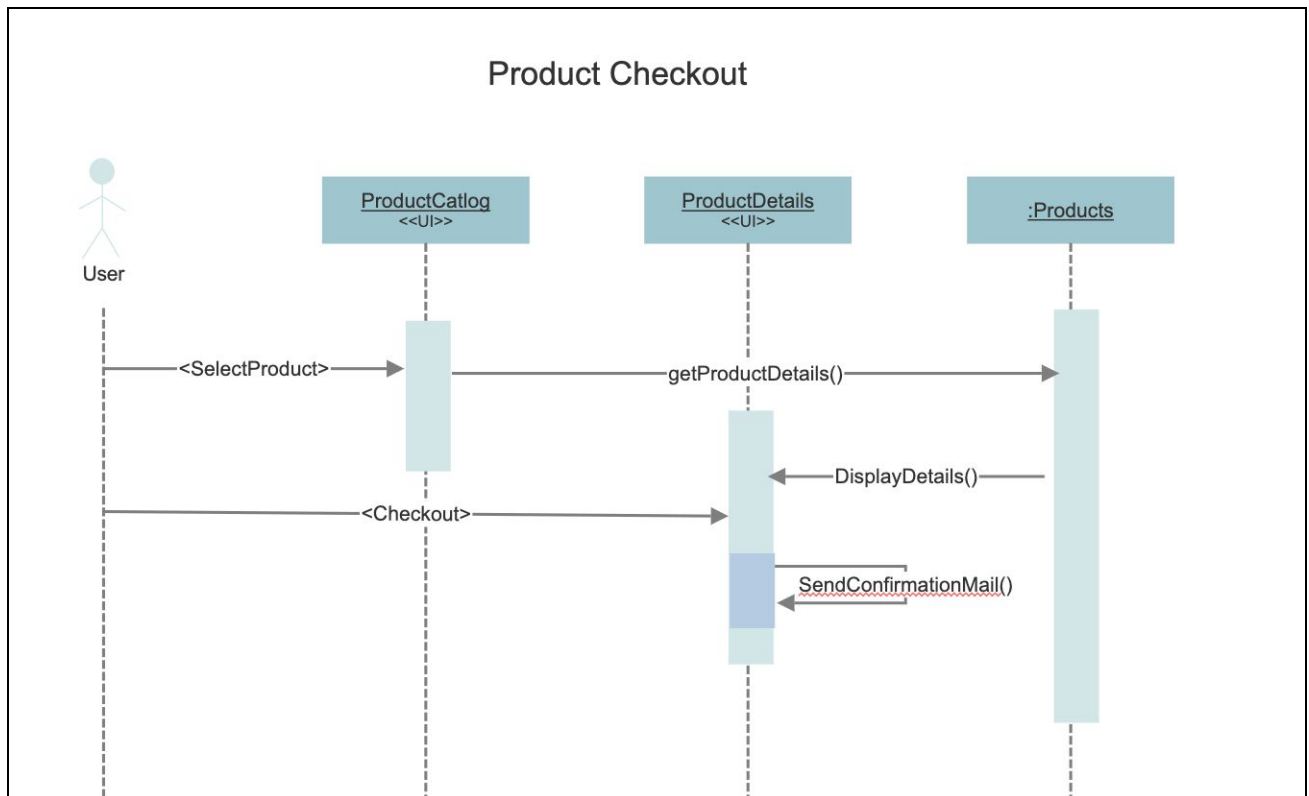
#### 1. Log-in Event

The above diagram explains the flow of the data. Whenever user sends login request using login button, it invokes the business logic to check the validity of the credentials inputted. If they are valid Welcome page is displayed else a error message is displayed on the screen. This includes frontend page displayed to user and a controller for business logic.



## 2. Product Checkout

Product catalog will contain list for all the products available and when user selects a product he/she will be directed to a Product details page and will be able to checkout. the email id and credit card details provided, an confirmation email will be sent to the user.



### 3.3. Load balancing, high availability, scalability:

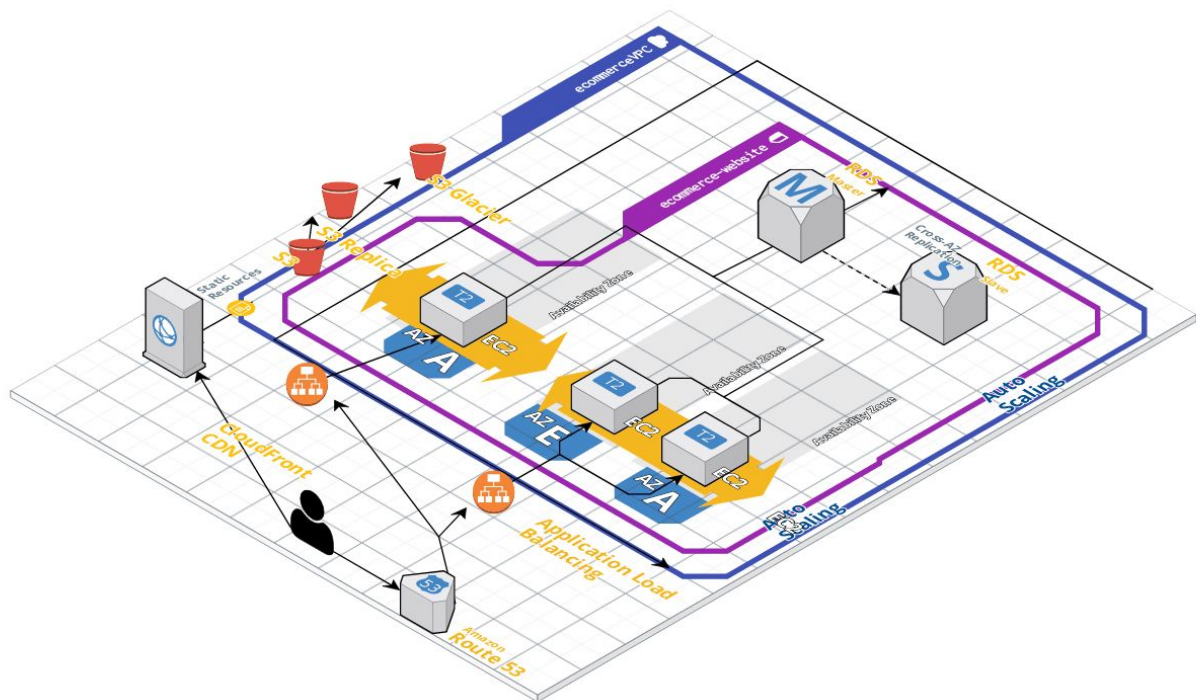
**Load Balancing:** Application Load Balancers are used in order to avoid hotspot to one specific instance. To improve throughput, when a user sends a request from the browser it is triggered to Aws Route53. Using 50% load distribution policy, the request is either sent to USA or India. The request is sent to the respective Application Load Balancer in the region. Here using equal Load balancing policy the request is further forwarded to one EC2 instance

**High Available:** It is very important to ensure that if an instance is down or there is high load Then the website shouldn't breakdown. This is ensured by Auto Scaling service by Aws where The AutoScaling runs health checks on the Ec2 instances and if there is no response in two Consecutive requests in 10 seconds then the instance is assumed to be down and a new

Instance will be triggered using the AMI. If the RDS database goes down then the replica is Served as a backup. To support disaster recovery, Multi AZ RDS of AWS is used. The static images are stored in S3 bucket , a back of this bucket is designed to support failure.

**Scalability :** The website runs behind AutoScaling, when the overall cpu utilization goes over 60% then scale out happens and once the load decreases then scale in happens

### 3.4. Architecture



### 3.5. Design Tradeoff

We have leveraged existing cloud services provided by Amazon which include Relational database service. In a distributed system like the one that we have designed where the resources are distributed in different Availability Zones, Amazon inherently supports availability over consistency. In our design this tradeoff is justifiable. Consider a case that we displayed available quantity of a product on website as 50 but the actual value in the database is 49 this won't cause any major service downtime. But service unavailability will cause loss of customer base and business loss which cannot be traded off.

## 4. IMPLEMENTATION

### 4.1 OS, languages, platform, technologies, and frameworks:

OS: Amazon Linux 2 AMI (Linux kernel 4.14)

Language: PHP, HTML, CSS

Frameworks : MySql

Platform : AWS

### 4.2 AWS Services leveraged

1. **Virtual Private Cloud(VPC):** VPC is virtual network where you create and manage your AWS resource in a more secure and scalable manner. We have created 2 subnets in the private cloud which is a way for us to group our resources within the VPC with their IP range. Following are the details of 2 subnets

#### Subnet 1

name: 10.0.1.0-us-east-1a

VPC: vpc-0186cc52b45224bbf

Availability zone: us-east-1a

IPv4 CIDR block: 10.0.1.0/24

#### Subnet 2:

name: 10.0.2.0-us-east-1e

VPC: vpc-0186cc52b45224bbf

Availability zone: us-east-1e

IPv4 CIDR block: 10.0.3.0/24

2. **Internet Gateway:** Internet gateway is used for communication between the EC2 instances in the VPC and the internet. Following steps were followed for adding an Internet gateway

1. Got to Internet gateways option

- Create internet gateway

- Name: ecommerceIGW

- Action : Attach to VPC (VPC: vpc-0186cc52b45224bbf)

3. **Application Load Balancer (ALB) :** It has come up with the main aim to spread the traffic to web servers to provide high performance. It provides elastic load balancing service where a high amount of traffic is distributed to various multiple zones dynamically. Two Application Load Balancers are used one in USA and one in India

4. **Auto Scaling:** With auto scaling our application our application will scale to accommodate traffic or shrink when there is no traffic to save cost. Automatically adjust the size of the EC2 instances serving the application based on need. We have designed our system such that if overall cpu utilization of ec2 instances is Over 60% we perform autoscaling.
5. **Cloud Front:** CloudFront is used for the faster and low latency delivery of the content through a worldwide network of data centers called edge locations. When a user requests product image that is served with CloudFront, the user is directed to the edge location that provides the lowest delivery time ,so that image is delivered with the best possible performance. If the images are already in the edge location with the lowest latency, CloudFront delivers it immediately. If the content is not in that edge location, CloudFront retrieves it from an origin that has been defined—such as an Amazon S3 bucket. We used following steps to create cloudfront

```
create a distribution->bucket->ecommerce-datastorage-cs218
name: d2h6qiygz71y88.cloudfront.net
```

6. **Amazon Elastic Compute Cloud:** We have deployed our server side application on Apache web server with PHP. EC2 instances will be scaled up or down based on the amount of load on each of the web server. All the business logic will be executed at These EC2 instances running in different regions.  
We have set the policy such that if the workload on any of the ec2 exceeds 60% then we create another instance of EC2 instance.
7. **Security Groups:** Security groups is similar to an inbound networks firewall. Each EC2 is simply assigned to certain security groups which are configured with IP addresses.
8. **S3 Buckets:** We have used S3 buckets to store images of the product to be displayed on the website. S3 provides us with secure, durable, highly scalable object storage. It involved following steps :
  1. Configuring Bucket
    - a. Choose Region.
    - b. Choose bucket name complaint with domain Name.

## 2. Uploading the content

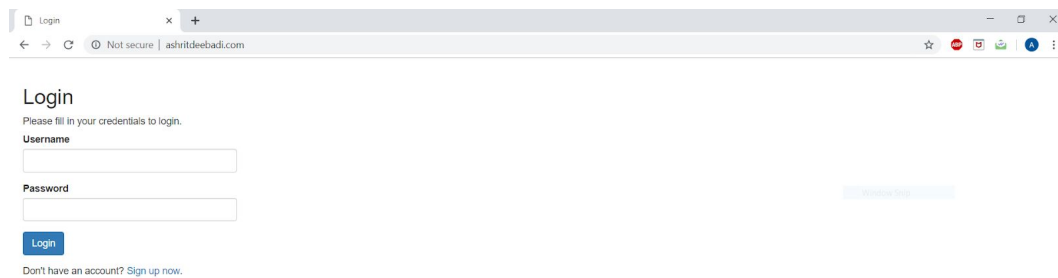
Content to be uploaded can be of any form upload the HTML, images, JavaScript files, CSS files, and other static assets.

## 3. Making Content Publicly Accessible

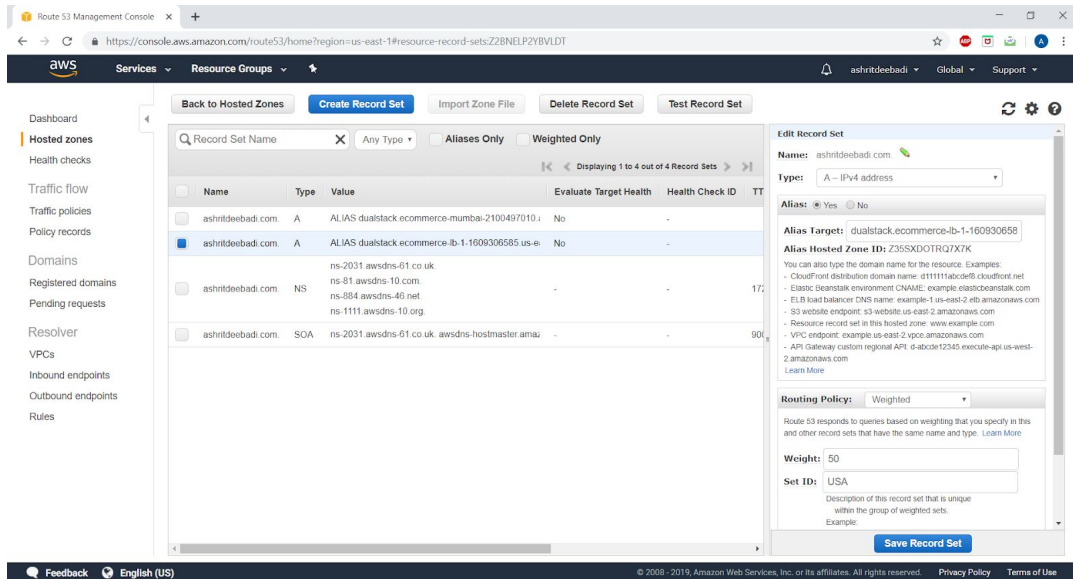
For visitor to be able to access the image it should be made public.

9. **Relational Database Service** : We chose MySQL relational database service provided by amazon for our project. Configuring database included setting Different details like storage type, dbname and password. For failover resistance we selected the Multi AZ deployment option. It creates Replica in Different Zone to have Amazon RDS maintain a synchronous standby replica in a different Availability Zone than the DB instance.

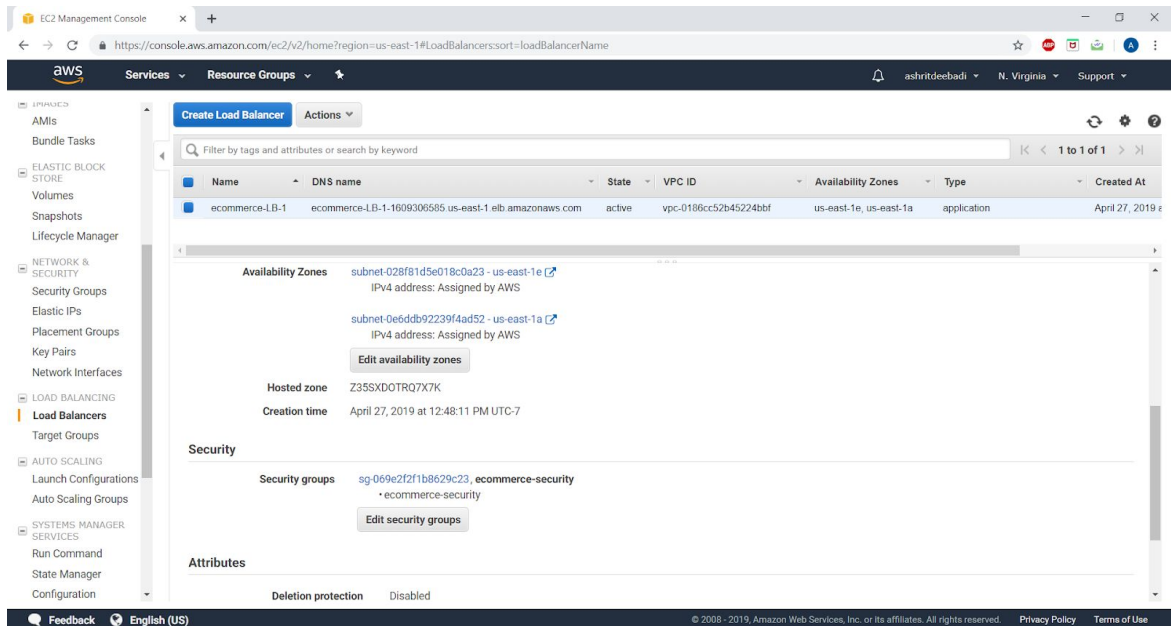
## 4.3 Screenshots of functionalities



Above image : Webpage being displayed from using route53 domain:ashritdeebadi.com



Above image : Route53 routing policies



Elastic loadbalancer1 , availabilityzones, security group shown above



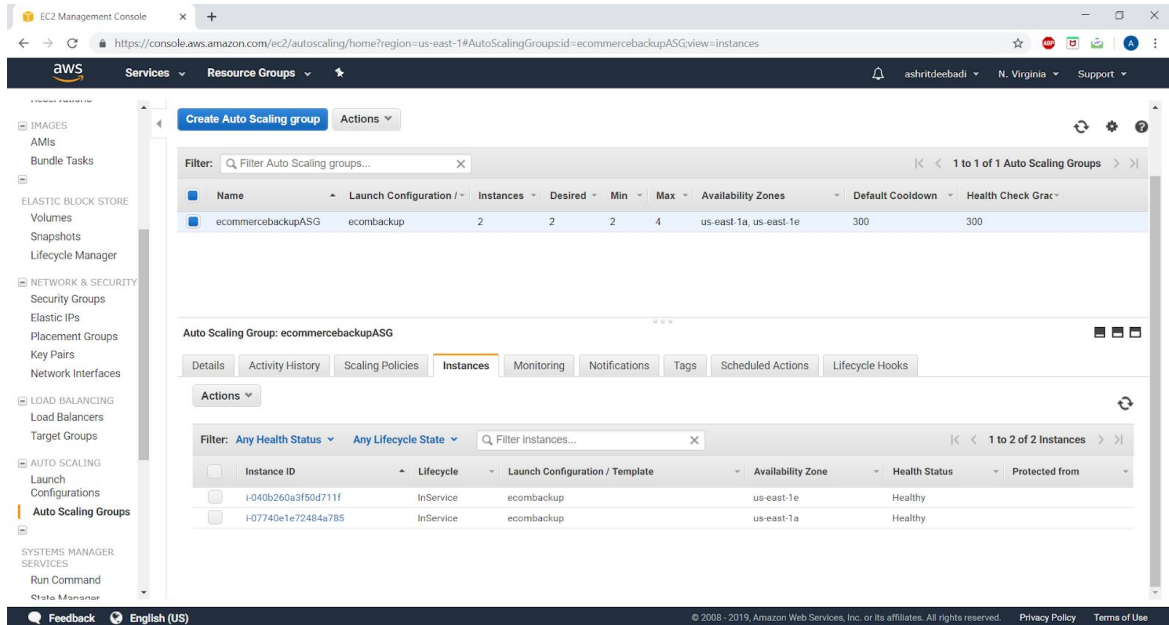
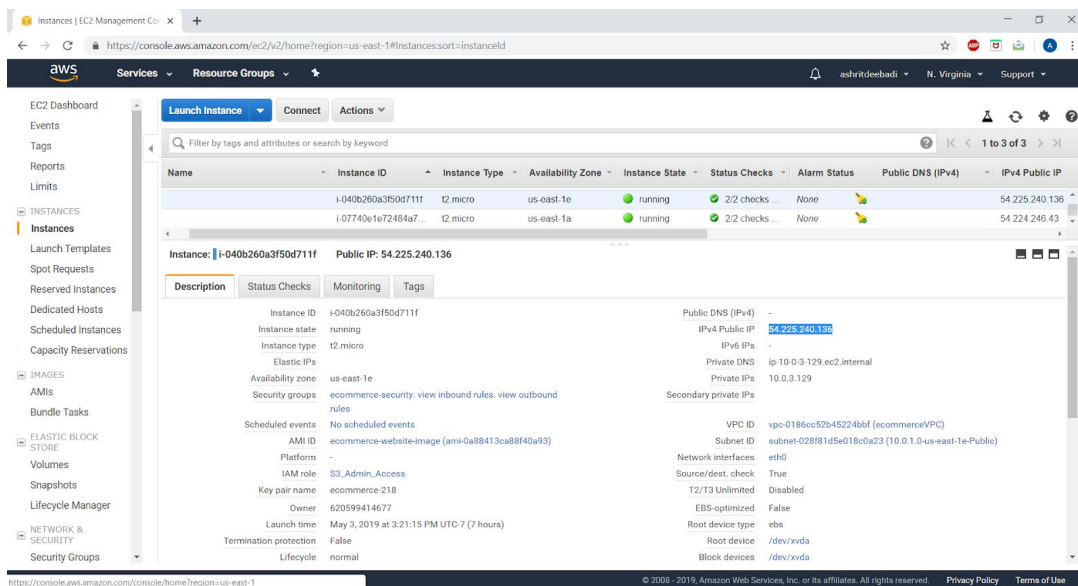
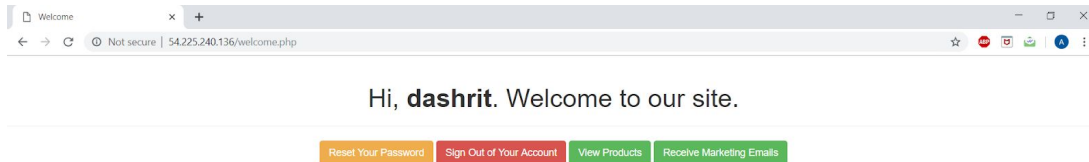


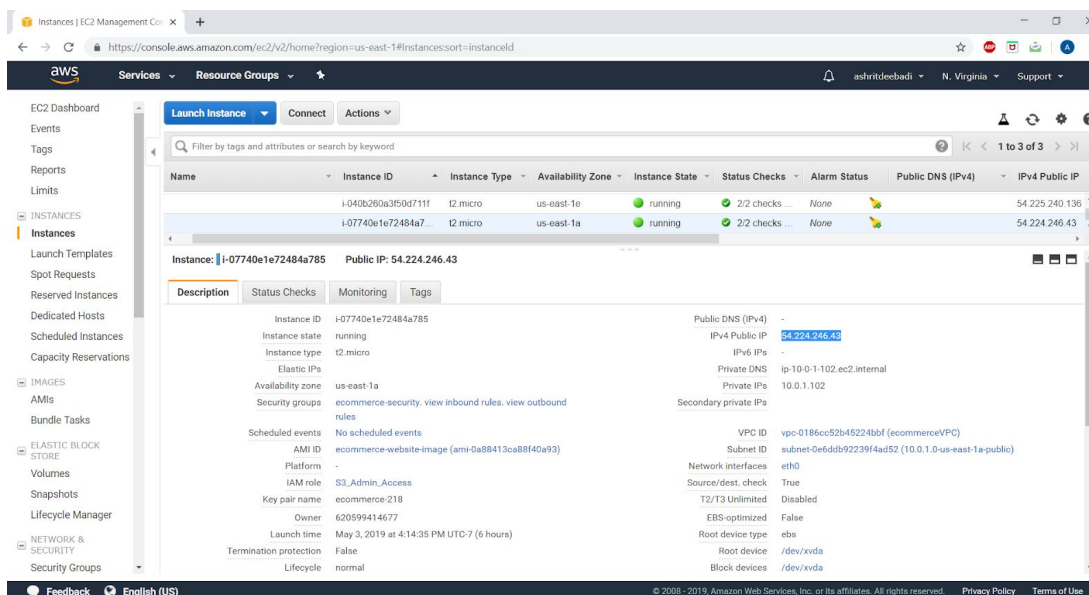
Image above : Autoscaling policies and instances registered under them



Instance-1 of our website : above details



Above image: Instance 1 running at client's end



Above image has details : Instance-2 of our website

Login

Please fill in your credentials to login.

Username

Password

Login

Don't have an account? [Sign up now.](#)

Above image : instance-2 running at client's end

Instances | EC2 Management | ashriteebadi.com/product... | WhatsApp+Image+2019-05... | Welcome | MySQL Database Log Files | Welcome

https://ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#instances:sort=instanceId

Launch Instance | Connect | Actions

Filter by tags and attributes or search by keyword

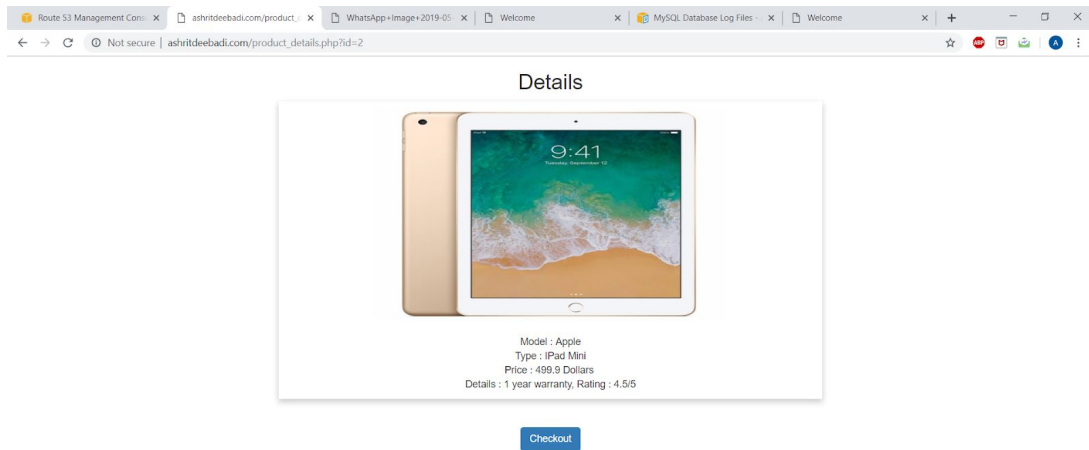
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
test	i-0294d048cc04cb760	t2.micro	ap-south-1a	running	2/2 checks ...		ec2-13-234-217-44.ap-...	13.234.217.44	-
	i-08bd0e9755fa16c4	t2.micro	ap-south-1a	terminated			-	-	-
	i-0d267f8ee95330b	t2.micro	ap-south-1a	terminated			-	-	-

Instance: i-0294d048cc04cb760 (test) Public DNS: ec2-13-234-217-44.ap-south-1.compute.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID	i-0294d048cc04cb760	Public DNS (IPv4)	ec2-13-234-217-44.ap-south-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	13.234.217.44
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-21-185.ap-south-1.compute.internal
Availability zone	ap-south-1a	Private IP	172.31.21.185
Security groups	default - view inbound rules - view outbound rules	Secondary private IP	
Scheduled events	No scheduled events	VPC ID	vpc-1b281d73
AMI ID	ecom (ami-0dc878c9d0db8bbfb)	Subnet ID	subnet-972270ff
Platform	-	Network interfaces	eth0
IAM role	S3_Admin_Access	Source/dest. check	True
Key pair name	ecommerce-mumbai	T2/T3 Unlimited	Disabled

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

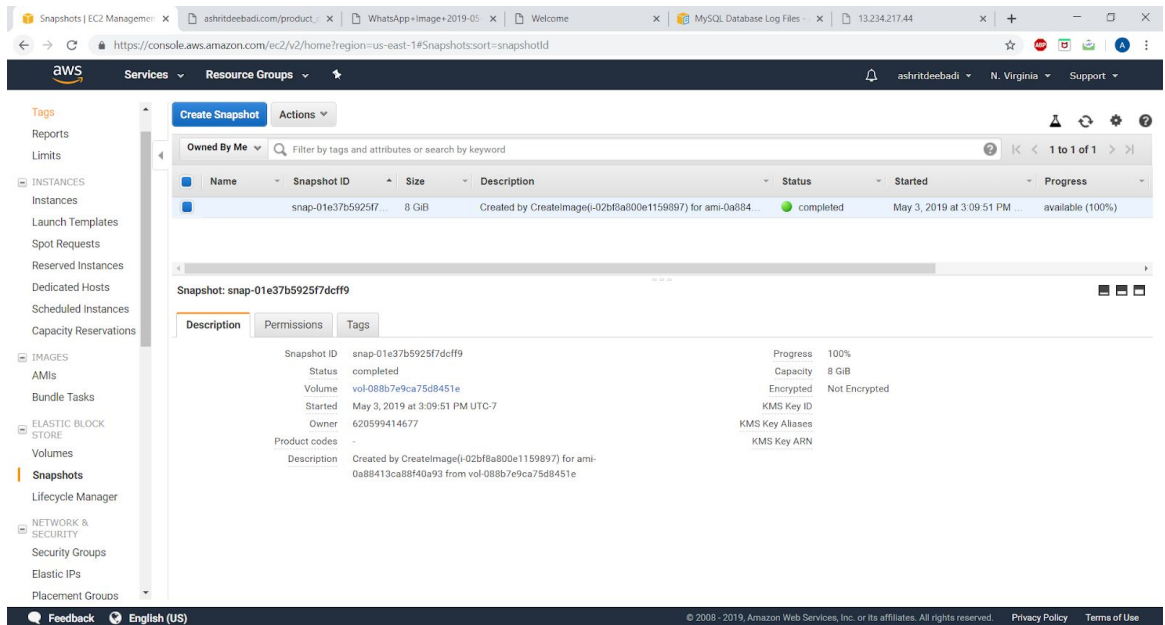
Above image : Instance in India details



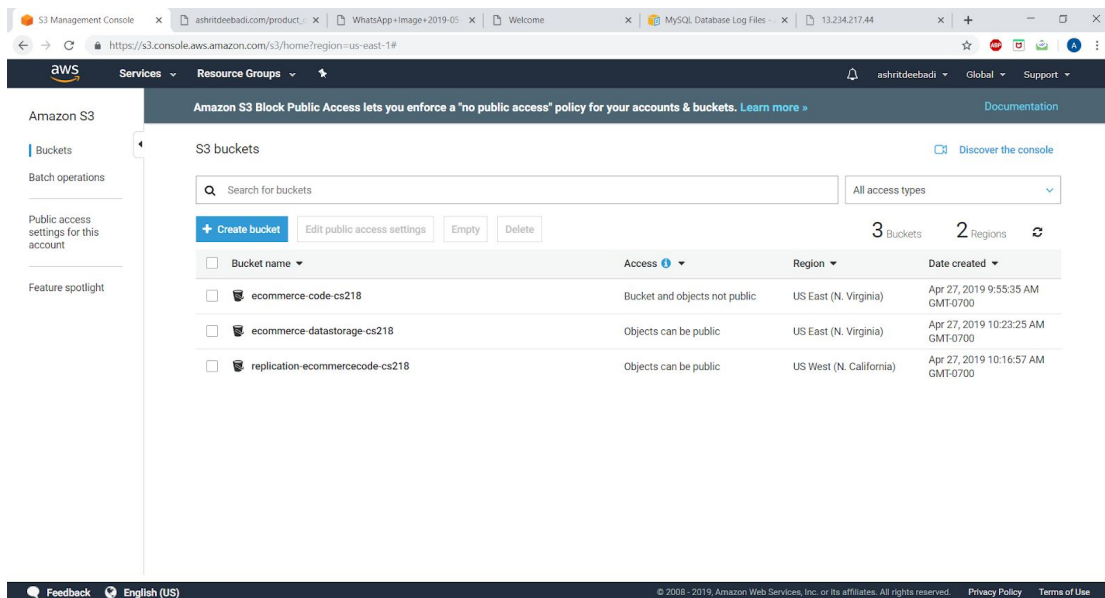
Above image : accessed by client in India



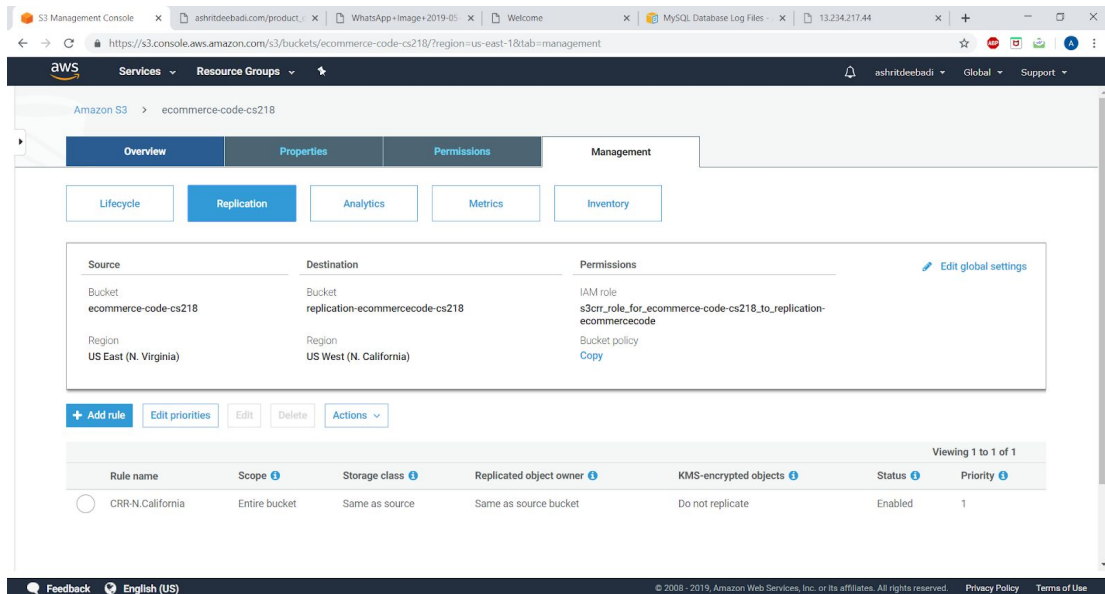
Images are accessed using cloudfront : Above image has cloudfront URL



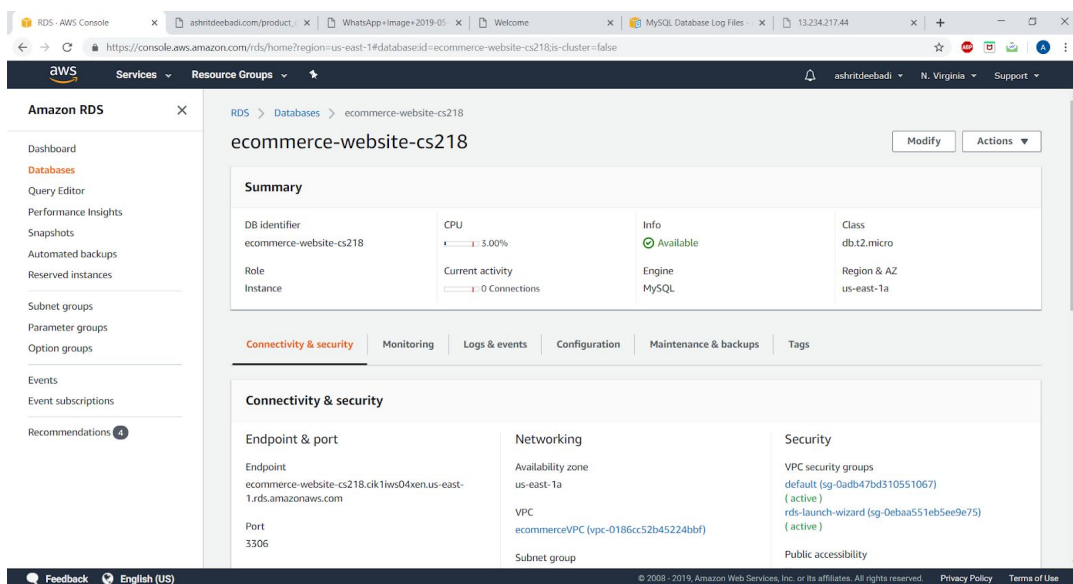
Snapshot of instance : Above image



S3 buckets : Above image



Replication policies of S3 : Above image



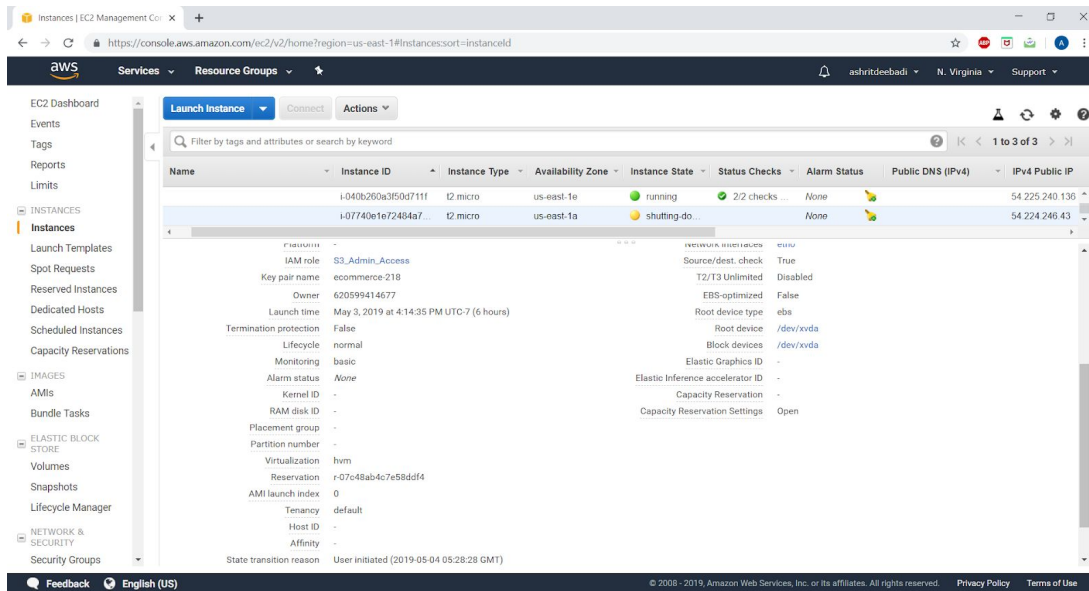
Rds : above image

4.4 Database script : We have generated backup.sql(Submitted with the project) script for our database. This can be used to create a copy of the database. This includes all the sample data that has been inserted into the tables of 'ecommerce' database. (Filename : backup.sql)

4.5 Sample log files submitted: cloudwatchLogs.jpeg, mysqlUpgradelogs, mysql-error-running.log.4

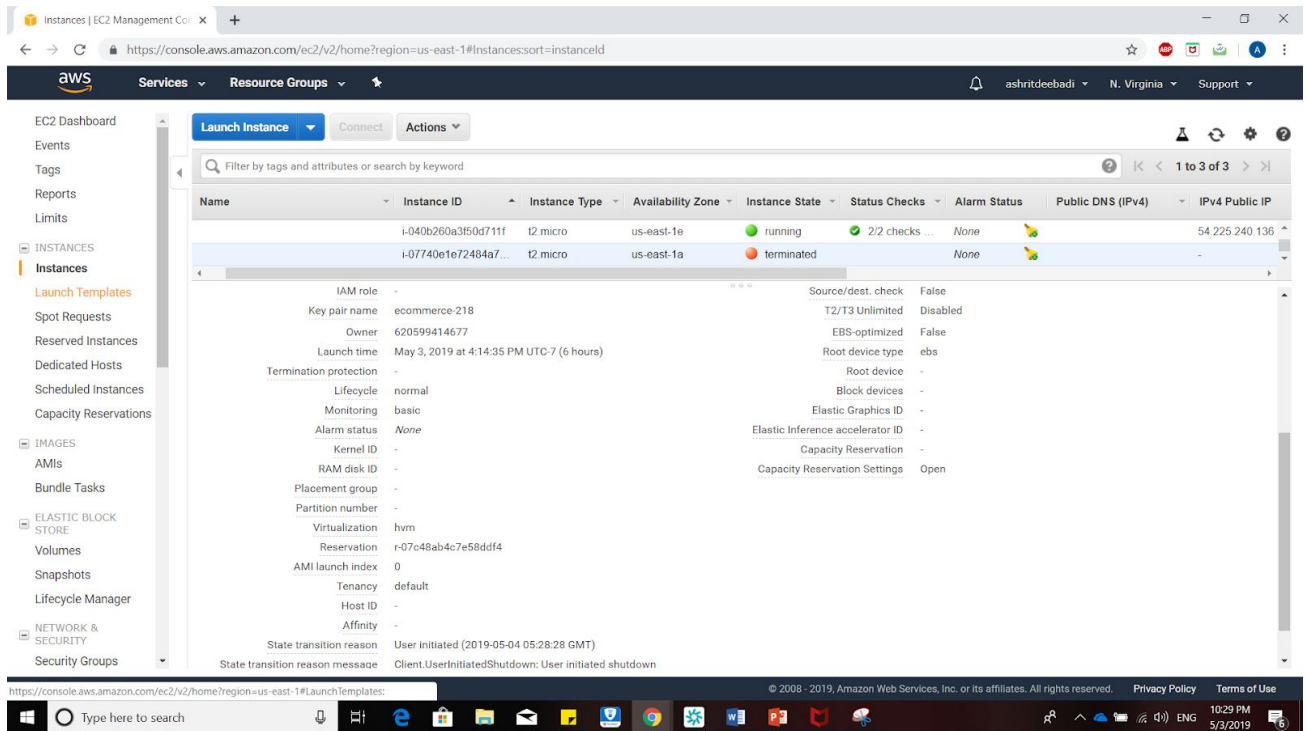
## 5. TEST CASES AND TEST PLAN EXECUTION

1. Testing High Availability by turning down an instance:
  - a) Open the instance in the AWS Console and shut down the instance by right clicking on it. The Instance state turns to shutting down

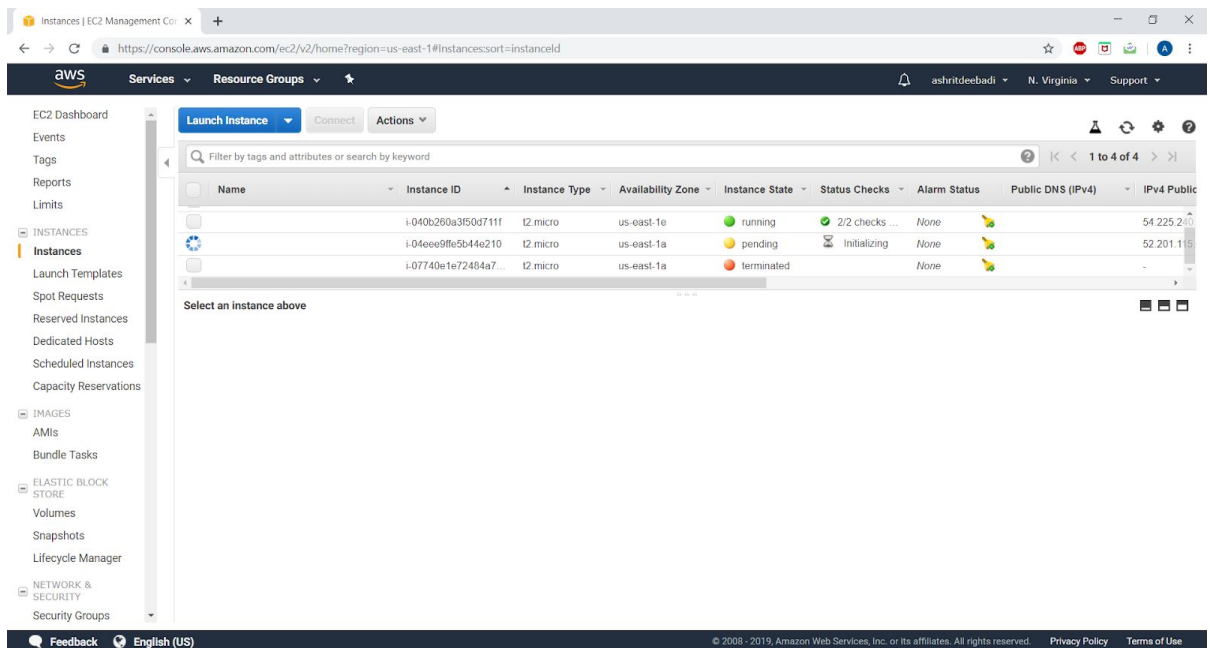


- b) After one minute the Instance state turns to Terminated, the instance is now unavailable





c) The AutoScaling policies are triggered and an instance is created automatically using the AMI. A new instance is displayed in AWS console with Instance state as pending



d) After a minute the instance state is turned to running



Instances | EC2 Management Console

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances:sort=instanceid

Services Resource Groups

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public
	i-040b260a3f50d711f	t2.micro	us-east-1e	running	2/2 checks ...	None		54.225.210
	i-04eee9ffe5b44e210	t2.micro	us-east-1a	running	Initializing	None		52.201.115.68
	i-07740e1e72484a7...	t2.micro	us-east-1a	terminated		None		

Instance: i-04eee9ffe5b44e210 Public IP: 52.201.115.68

Description Status Checks Monitoring Tags

Instance ID: i-04eee9ffe5b44e210  
 Instance state: running  
 Instance type: t2.micro  
 Elastic IPs: -  
 Availability zone: us-east-1a  
 Security groups: ecommerce-security, view inbound rules, view outbound rules  
 Scheduled events: No scheduled events  
 AMI ID: ecommerce-website-image (ami-0a88413ca88f40a93)  
 Platform: -  
 IAM role: S3\_Admin\_Access  
 Key pair name: ecommerce-218  
 Owner: 620599414677  
 Launch time: May 3, 2019 at 10:31:00 PM UTC-7 (less than one hour)

Public DNS (IPv4): -  
 IPv4 Public IP: 52.201.115.68  
 IPv6 IPs: -  
 Private DNS: ip-10-0-1-25.ec2.internal  
 Private IPs: 10.0.1.25  
 Secondary private IPs: -  
 VPC ID: vpc-0186cc52b45224bbf (ecommerceVPC)  
 Subnet ID: subnet-0e6ddb92239f4ad52 (10.0.1.0-us-east-1a-public)  
 Network interfaces: eth0  
 Source/dest. check: True  
 T2/T3 Unlimited: Disabled  
 EBS-optimized: False  
 Root device type: ebs

https://console.aws.amazon.com/console/home?region=us-east-1

## e) Verify the logs in AutoScaling

EC2 Management Console

https://console.aws.amazon.com/ec2/autoscaling/home?region=us-east-1#AutoScalingGroups?id=ecommercebackupASG&view=history

Services Resource Groups

Create Auto Scaling group Actions

Filter: Filter Auto Scaling groups...

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	DefaultCooldown	HealthCheckGracePeriod
ecommerceba...	ecombackup	3	3	2	4	us-east-1a, us-east-1e	300	300

Successful Launching a new EC2 instance: i-04eee9ffe5b44e210 2019 May 3 22:31:01 UTC-7 2019 May 3 22:31:37 UTC-7

Description: Launching a new EC2 instance: i-04eee9ffe5b44e210

Cause: At 2019-05-04T05:30:58Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.

Successful Terminating EC2 instance: i-07740e1e72484a785 2019 May 3 22:30:28 UTC-7 2019 May 3 22:35:43 UTC-7

Description: Terminating EC2 instance: i-07740e1e72484a785

Cause: At 2019-05-04T05:30:28Z an instance was taken out of service in response to a EC2 health check indicating it has been terminated or stopped.

https://console.aws.amazon.com/console/home?region=us-east-1

f) Check if the website can be accessed by using the public ip if this instance

Instances | EC2 Management Console | Login

← → ↻ Not secure | 52.201.115.68

### Login

Please fill in your credentials to login.

**Username**

**Password**

Login

Don't have an account? [Sign up now.](#)

2. Testing if new instance is created when the overall CPU utilization is above 60%

a) Create cpu utilization in all 3 instances by running

i=0

While i!=1:

print(testing)

b) Open the AWS console and verify if a new instance is generated with instance state Running and status check initializing

Instances | EC2 Management Console | Login | test an application load balancer

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances:sort=instanceid

aws Services Resource Groups

ashritdeebadi N. Virginia Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES

Instances

Launch Templates Spot Requests Reserved Instances Dedicated Hosts Scheduled Instances Capacity Reservations

IMAGES

AMIs Bundle Tasks

ELASTIC BLOCK STORE

Volumes Snapshots Lifecycle Manager

NETWORK & SECURITY

Security Groups

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public
<input type="checkbox"/>		i-021c22201632c8ab2	t2.micro	us-east-1e	running	Initializing	None		54.172.55.1
<input type="checkbox"/>	e-commerce website-N Virg-2	i-02b8a900e1159897	t2.micro	us-east-1e	running	2/2 checks passed	None		100.25.188.
<input type="checkbox"/>		i-040b260a3f50d711f	t2.micro	us-east-1e	running	2/2 checks passed	None		54.225.240.
<input type="checkbox"/>		i-04ee90ffe5b44e210	t2.micro	us-east-1a	running	2/2 checks passed	None		52.201.115.
<input type="checkbox"/>		i-07740e1e72484a7...	t2.micro	us-east-1a	terminated		None		-

Select an instance above

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

### c) Verify the logs in Auto Scaling Group

The screenshot displays the AWS Management Console interface for an Auto Scaling Group. The left sidebar shows the navigation menu with categories like INSTANCES, IMAGES, ELASTIC BLOCK STORE, and NETWORK & SECURITY. The main content area shows the 'ecommercebackupASG' Auto Scaling Group. Below the group details, the 'Activity History' tab is selected, showing a list of scaling activities. The first activity is 'Waiting for instance warmup' with a description: 'Launching a new EC2 instance: i-021c22201632c8ab2'. The second activity is 'Successful' with a description: 'Launching a new EC2 instance: i-04eee9ffe5b44e210'. The third activity is 'Failed' with a description: 'Terminating EC2 instance: i-0734a513b664c39f'. The bottom of the console shows the footer with 'Feedback', 'English (US)', and copyright information.

Status	Description	Start Time	End Time
Waiting for instance warmup	Launching a new EC2 instance: i-021c22201632c8ab2	2019 May 3 22:57:30 UTC-7	
Successful	Launching a new EC2 instance: i-04eee9ffe5b44e210	2019 May 3 22:31:01 UTC-7	2019 May 3 22:31:37 UTC-7
Failed	Terminating EC2 instance: i-0734a513b664c39f	2019 May 3 22:31:01 UTC-7	2019 May 3 22:31:37 UTC-7

### Any major modifications from proposal and why

There are no major modifications from the proposal. The minor modifications from the project are as follows:

- We have used PHP instead of Java to build the website
- Support for recommendations based on previous searches by the user has not been added.

### Any uniqueness(design, implementation, etc.) that you are proud of:

We have supported hashed passwords which gives higher security. Also, we have used strong consistency for usernames and passwords as these details should be the latest ones and security for these cannot be compromised at any cost. We have leveraged many AWS cloud services which made it easy to develop the application. We developed the application in a way which can easily be extended based on the requirements.

**Post mortem:**

**Issues uncovered :** The eventual consistency for different users can be setup based on their requirements as discussed in the baseball application research paper in class.

**Implement something differently:** To send the marketing mails, Amazon Mail Services could have been used which can be configured as per the roles of user.

**Potential improvements:** This application was developed for understanding the usage of cloud services that are available and lot of improvements can be added to this application. Some of them are as below:

- Incorporate Artificial Intelligence based recommendation system
- Add chatting service where users can enquire about the products.

**References**

<https://aws.amazon.com/blogs/aws/three-new-aws-reference-architectures-for-e-commerce/>  
<https://d1.awsstatic.com/whitepapers/Building%20Static%20Websites%20on%20AWS.pdf>  
<https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html>