

PROJECT REPORT

EXPRESSNEST

(Microblogging Application)

ENTERPRISE SOFTWARE PLATFORMS (CMPE-272)

Course Instructor: Prateek Sharma

Team expressNesters:

Harshith Akkapelli

Neeharika Singh

Sanjay Bhargav Kudupudi

GitHub: <https://github.com/ExpressNesters/ExpressNest.git>

Introduction:

ExpressNest is a microblogging application. This is platform for the users to share their feelings, convey their ideas and news to other audiences. The name “expressNest” signifies that users can express their ideas in a warm and comfortable environment.

Key Features:

Users can post a message, edit/delete a post.

Users can follow other users.

Users can view the posts of the followee on their timeline.

Technologies stack:

Authentication microservice:

Flask python, Firebase

Post service:

Java (Spring boot)

Database: Postgres

Feed service:

Java (Spring boot)

Database: MongoDB

Follow service:

Golang

Database: MongoDB

User profile management Service:

Golang

Database: Postgres

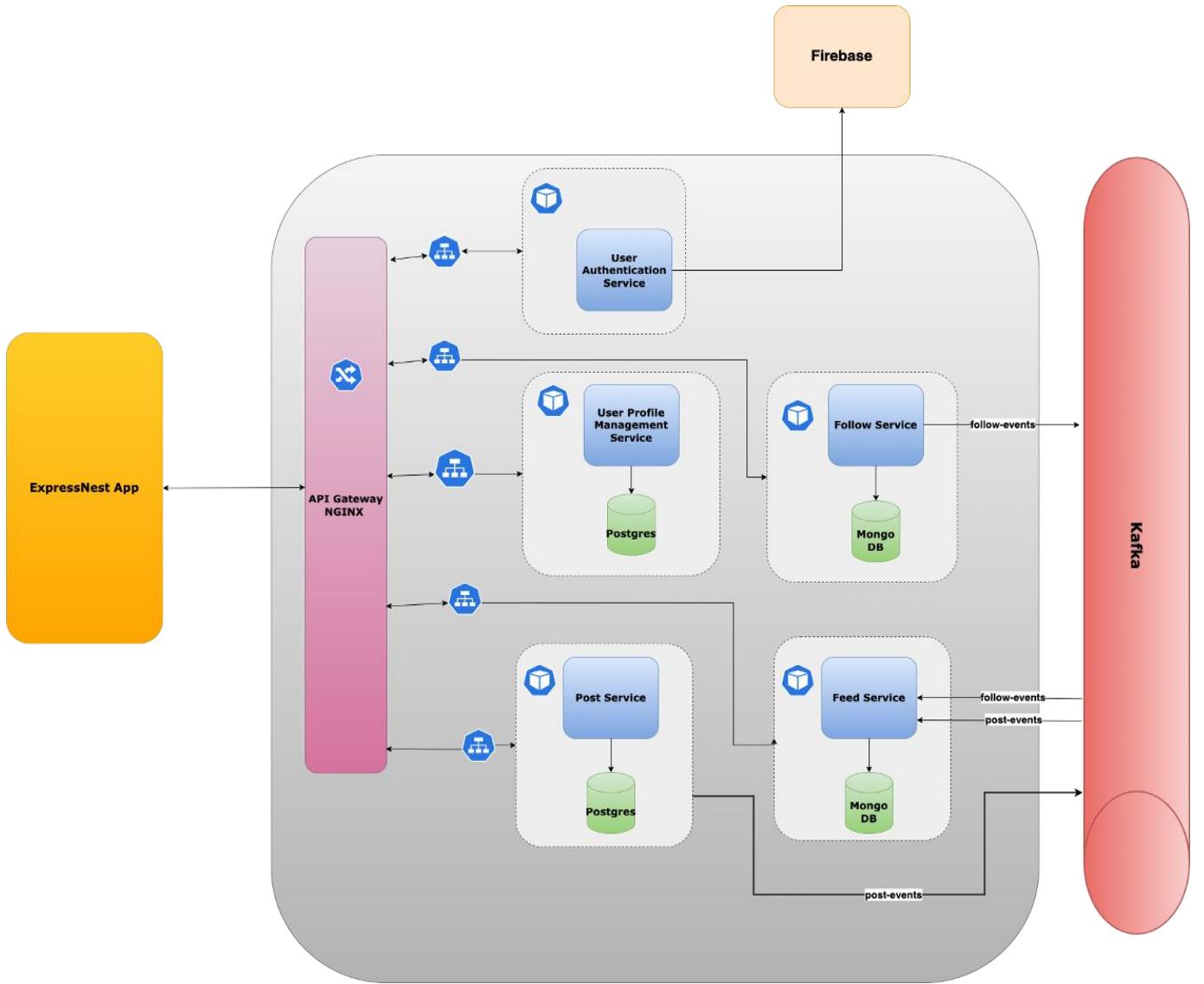
Frontend:

React Js,Html, Css

Caching:

Redis

Architecture:



Microservices:

Authentication Service:

User Authentication Service

Login

Input credentials
Validate credentials

Sign Up/Register

Input User Details
Verify User Details

FIREBASE

Authentication

Password Recovery

Recovery using User
ID

SSO Login

Eternal Identity Providers(IdP)
(Federated Sign IN) like Google ,Github.
Redirect to External IdP.
Authenticate with IdP
Return to application with authentication
confirmation

2Factor Authentication

Generate a Code
Validate Code

Post Service:

Post Service

Post Service

createPost()
getPostByPostId(long postId)
getPostsByUserId(long userId)
getPostsByPostIds(List<Long> postIds)
updatePostByPostId(long postId)
deletePostByPostId(long postId)

User can create a post

User can edit a post (post edit history will be created)

User can delete a post

Post can have attachments like images and videos.

If a new post is made by using the share post button then the shared post will be saved as an attachment in the DB and the user can add their own additional text on top of that.

Followers can like/unlike the post

Followers can comment on the post

Followers can share the post (retweet)

Database

Postgres database

Caching - Redis



Follow Service:

FOLLOW SERVICE

Features:

Follow User: A user can follow another user.

Unfollow User: A user can unfollow another user they're currently following.

Get All Followers By User ID: Retrieve a list of all users that are following a particular user.

User:

user_id: Unique identifier for a user.

username: Unique name for user identification.

email: User's email address.

password: Password encrypted for security.

Follow:

Follower_id: ID of the user who is following.

following id: ID of the user who is being followed.

timestamp: Time when the follow action took place.

Unfollow:

Follower_id: ID of the user who is unfollowing.

Following_id: ID of the user who is being unfollowed.

timestamp: Time when the follow action took place.

**MONGO
DB**

Feed Service:

CMPE-272 Enterprise. Software Platforms

- Users will have their **own timeline** which shows all the posts created by them
- Users will have **main feed** which will show the posts made by the followees (people whom the current user follows) in chronological order)
- If a user unfollows a person then that particular feed should not appear in the feeds.
- If a user starts following a person then the posts of that user should start showing up in the user's feed. (from the time the user started following.) (can decide from what time)
- If a user has blocked a person then the either of them cannot see each other's post

There will be more users who will be viewing their feeds than the users creating posts.

More Reads

Comparatively less writes

There will be users which will have more followers, hence we would need to show their posts to more number of people

There will be users who follow large number of people. In this case their feeds will have more updates. (we can limit the number people a user can follow)
Need to rank the feeds which will be shown. (how? - show a max of 5 feeds from a particular followee)

Feeds need to be cached

Redis caching

Event based feed generation

AsyncFeedService

Whenever a user creates a post, then there will be a **AsyncFeedService** which will maintain feeds for all the users.

Caching will also be implemented here

DB:

MongoDB

feed_events Table

- user_id - the current user whose feed is being maintained
- post_id
- posted_by - user id of the followee

On demand feed generation

SyncFeedService

`getFeedForUser(long userID)`

Here we will first get the postIDs and then rank and determine which are the posts to be shown. After which we will fetch the other details from the DB

Caching at user level

key - user id

Value - posts to be shown to the user (20 posts will be cached)

`set ttl for cache`

User profile Management Service:

CMPE-272 Enterprise. Software Platforms

User Profile Management Service

Create - Write
Get - Read
Update - Write
Delete - write

- 1) Writes are not frequent
- 2) Reads are extremely frequent - profile visits!
- 3) Caching Makes sensel -
- 4) Common Database?
- 5) Concurrent Reads
- 6) Concurrent Write!

Requirements:
Lots of concurrent reads than writes
Consistency is less required than availability - Because less writes, very few things change for each update so it makes sense to compromise consistency
Should scale well for concurrent requests

Database Options:
1) Postgres
2) Couch Db
3) Riak
4) MongoDB

Caching Options:
1) Redis
2) Memcached

Notes

- 1) No foreign key
- 2) Need to see consistency options
- 3) Caching strategy - FIFO, lets see as application evolves
- 4) JWT tokens for storing session related info

Logs and metrics:(logs screenshot is available in the EKS section)

We have used Prometheus for our metrics

```
# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'"} 1.3726336E7
jvm_memory_used_bytes{area="heap",id="G1 Survivor Space"} 5288736.0
jvm_memory_used_bytes{area="heap",id="G1 Old Gen"} 5.1106304E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace"} 9.3209208E7
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-nmethods'"} 1430656.0
jvm_memory_used_bytes{area="heap",id="G1 Eden Space"} 2097152.0
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space"} 1.2825296E7
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-profiled nmethods'"} 4153216.0
# HELP system_load_average_1m The sum of the number of runnable entities queued to available processors and the number of runnable entities running on the available processors averaged over a period of time
# TYPE system_load_average_1m gauge
system_load_average_1m 2.5810546975
# HELP jvm_classes_loaded_classes The number of classes that are currently loaded in the Java virtual machine
# TYPE jvm_classes_loaded_classes gauge
jvm_classes_loaded_classes 18899.0
# HELP system_cpu_usage The "recent cpu usage" of the system the application is running in
# TYPE system_cpu_usage gauge
system_cpu_usage 0.0
# HELP process_files_open_files The open file descriptor count
# TYPE process_files_open_files gauge
process_files_open_files 152.0
# HELP hikaricp_connections_acquire_seconds Connection acquire time
# TYPE hikaricp_connections_acquire_seconds summary
hikaricp_connections_acquire_seconds_count{pool="HikariPool-1"} 0.0
hikaricp_connections_acquire_seconds_sum{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_acquire_seconds_max Connection acquire time
# TYPE hikaricp_connections_acquire_seconds_max gauge
hikaricp_connections_acquire_seconds_max{pool="HikariPool-1"} 0.0
# HELP tomcat_sessions_expired_sessions_total
# TYPE tomcat_sessions_expired_sessions_total counter
tomcat_sessions_expired_sessions_total 0.0
# HELP process_uptime_seconds The uptime of the Java virtual machine
# TYPE process_uptime_seconds gauge
process_uptime_seconds 22.482
# HELP jvm_gc_overhead_percent An approximation of the percent of CPU time used by GC activities over the last lookback period or since monitoring began, whichever is shorter, in the range [0..1]
# TYPE jvm_gc_overhead_percent gauge
jvm_gc_overhead_percent 0.001611492744201697
# HELP jvm_buffer_memory_used_bytes An estimate of the memory that the Java virtual machine is using for this buffer pool
# TYPE jvm_buffer_memory_used_bytes gauge
jvm_buffer_memory_used_bytes{id="mapped - 'non-volatile memory'"} 0.0
jvm_buffer_memory_used_bytes{id="mapped"} 0.0
jvm_buffer_memory_used_bytes{id="direct"} 8192.0
# HELP tomcat_sessions_created_sessions_total
# TYPE tomcat_sessions_created_sessions_total counter
tomcat_sessions_created_sessions_total 0.0
# HELP hikaricp_connections_max Max connections
# TYPE hikaricp_connections_max gauge
```

CMPE-272 Enterprise. Software Platforms

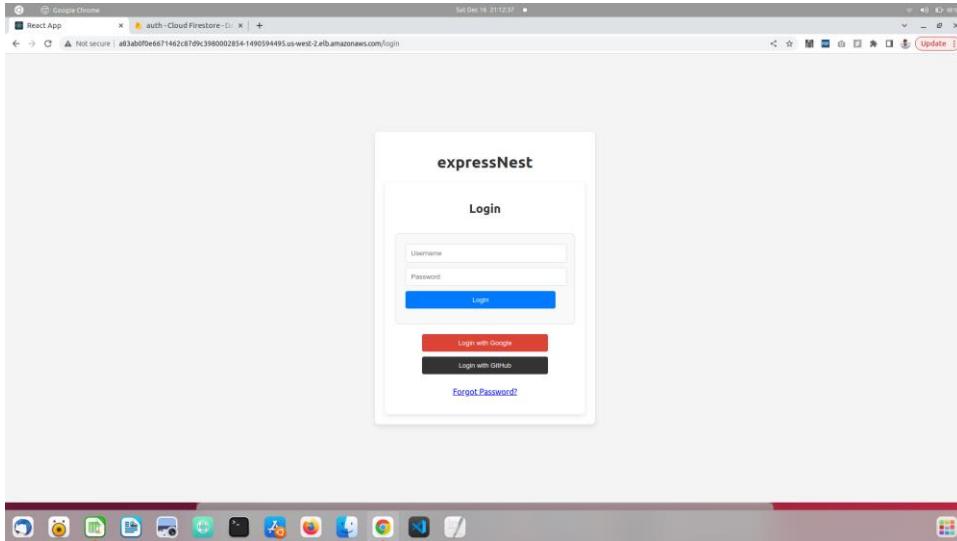
```
# TYPE jvm_threads_live_threads gauge
jvm_threads_live_threads 25.0
# HELP process_start_time_seconds Start time of the process since unix epoch.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.702785300311E9
# TYPE jvm_gc_memory_allocated_bytes_total Incremented for an increase in the size of the (young) heap memory pool after one GC to before the next
# TYPE jvm_gc_memory_allocated_bytes_total counter
jvm_gc_memory_allocated_bytes_total 4.4040192E7
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'",} 1.376256E7
jvm_memory_committed_bytes{area="heap",id="G1 Survivor Space",} 6291456.0
jvm_memory_committed_bytes{area="heap",id="G1 Old Gen",} 7.340032E7
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 9.3978624E7
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'non-nmethods'",} 2555904.0
jvm_memory_committed_bytes{area="heap",id="Eden Space",} 4.6137344E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 1.3172736E7
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'non-profiled nmethods'",} 4194304.0
# HELP hikaricp_connections_timeout_total Connection timeout total count
# TYPE hikaricp_connections_timeout_total counter
hikaricp_connections_timeout_total{pool="HikariPool-1",} 0.0
# HELP hikaricp_connections_pending Pending threads
# TYPE hikaricp_connections_pending gauge
hikaricp_connections_pending{pool="HikariPool-1",} 0.0
# HELP application_ready_time_seconds Time taken for the application to be ready to service requests
# TYPE application_ready_time_seconds gauge
application_ready_time_seconds{main_application_class="edu.sjsu.expressnest.postservice.PostServiceApplication",} 11.651
# HELP hikaricp_connections_active Active connections
# TYPE hikaricp_connections_active gauge
hikaricp_connections_active{pool="HikariPool-1",} 0.0
# HELP jdbc_connections Number of established but idle connections.
# TYPE jdbc_connections_idle gauge
jdbc_connections_idle{name="dataSource",} 10.0
# HELP spring_kafka_template_seconds_max KafkaTemplate Timer
# TYPE spring_kafka_template_seconds_max gauge
spring_kafka_template_seconds_max{exception="none",name="kafkaTemplate",result="success",} 0.0
# HELP spring_kafka_template_seconds KafkaTemplate Timer
# TYPE spring_kafka_template_seconds summary
spring_kafka_template_seconds_count{exception="none",name="kafkaTemplate",result="success",} 0.0
spring_kafka_template_seconds_sum{exception="none",name="kafkaTemplate",result="success",} 0.0
# HELP tomcat_sessions_rejected_sessions_total
# TYPE tomcat_sessions_rejected_sessions_total counter
tomcat_sessions_rejected_sessions_total 0.0
# HELP jvm_memory_usage_after_gc_percent The percentage of long-lived heap pool used after the last GC event, in the range [0..1]
# TYPE jvm_memory_usage_after_gc_percent gauge
jvm_memory_usage_after_gc_percent{area="heap",pool="long-lived",} 0.011899113655090332
# HELP jvm_memory_max_bytes The maximum amount of memory in bytes that can be used for memory management
# TYPE jvm_memory_max_bytes gauge
jvm_memory_max_bytes{id="CodeHeap 'profiled nmethods'",} 1.279088677FR

# HELP process_cpu_usage The "recent cpu usage" for the Java Virtual Machine process
# TYPE process_cpu_usage gauge
process_cpu_usage 0.0
# HELP hikaricp_connections_usage_seconds Connection usage time
# TYPE hikaricp_connections_usage_seconds summary
hikaricp_connections_usage_seconds_count{pool="HikariPool-1",} 0.0
hikaricp_connections_usage_seconds_sum{pool="HikariPool-1",} 0.0
# HELP hikaricp_connections_usage_seconds_max Connection usage time
# TYPE hikaricp_connections_usage_seconds_max gauge
hikaricp_connections_usage_seconds_max{pool="HikariPool-1",} 0.0
# HELP hikaricp_connections Total connections
# TYPE hikaricp_connections gauge
hikaricp_connections{pool="HikariPool-1",} 10.0
# HELP tomcat_sessions_active_current_sessions
# TYPE tomcat_sessions_active_current_sessions gauge
tomcat_sessions_active_current_sessions 0.0
# HELP jvm_gc_live_data_size_bytes Size of long-lived heap memory pool after reclamation
# TYPE jvm_gc_live_data_size_bytes gauge
jvm_gc_live_data_size_bytes 0.0
# HELP jvm_threads_peak_threads The peak live thread count since the Java virtual machine started or peak was reset
# TYPE jvm_threads_peak_threads gauge
jvm_threads_peak_threads 25.0
# HELP hikaricp_connections_idle Idle connections
# TYPE hikaricp_connections_idle gauge
hikaricp_connections_idle{pool="HikariPool-1",} 10.0
# HELP jvm_gc_max_data_size_bytes Max size of long-lived heap memory pool
# TYPE jvm_gc_max_data_size_bytes gauge
jvm_gc_max_data_size_bytes 294967296F
# HELP process_files_max_file The maximum file descriptor count
# TYPE process_files_max_file gauge
process_files_max_file 10240.0
# HELP jdbc_connections_min Minimum number of idle connections in the pool.
# TYPE jdbc_connections_min gauge
jdbc_connections_min{name="dataSource",} 10.0
# HELP system_cpu_count The number of processors available to the Java virtual machine
# TYPE system_cpu_count gauge
system_cpu_count 8.0
# HELP jdbc_connections_max Maximum number of active connections that can be allocated at the same time.
# TYPE jdbc_connections_max gauge
jdbc_connections_max{name="dataSource",} 10.0
# HELP executor_pool_max_threads The maximum allowed number of threads in the pool
# TYPE executor_pool_max_threads gauge
executor_pool_max_threads{name="applicationTaskExecutor",} 2.147483647E9
# HELP jvm_buffer_count_buffers An estimate of the number of buffers in the pool
# TYPE jvm_buffer_count_buffers gauge
jvm_buffer_count_buffers{id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_count_buffers{id="mapped",} 0.0
jvm_buffer_count_buffers{id="direct",} 1.0
```

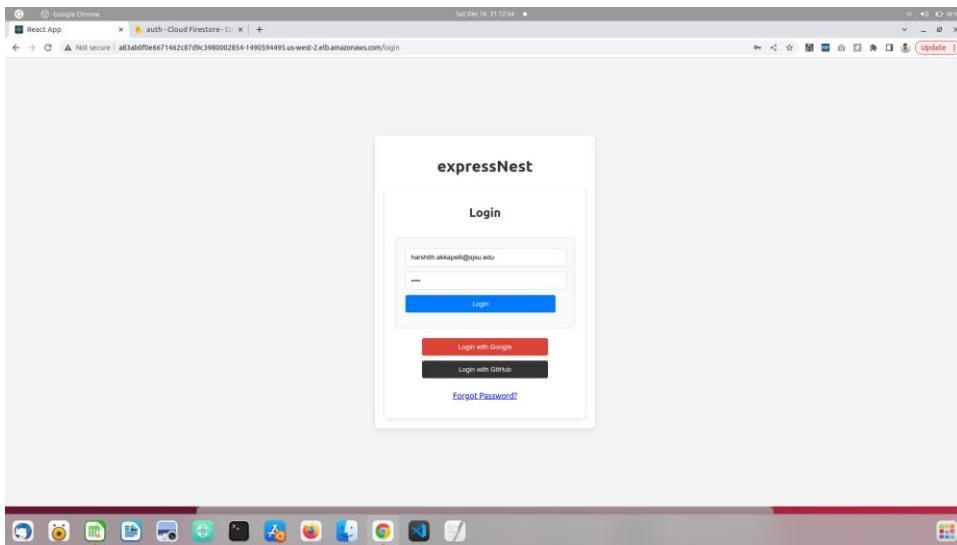
User Interface:

CMPE-272 Enterprise. Software Platforms

Login page

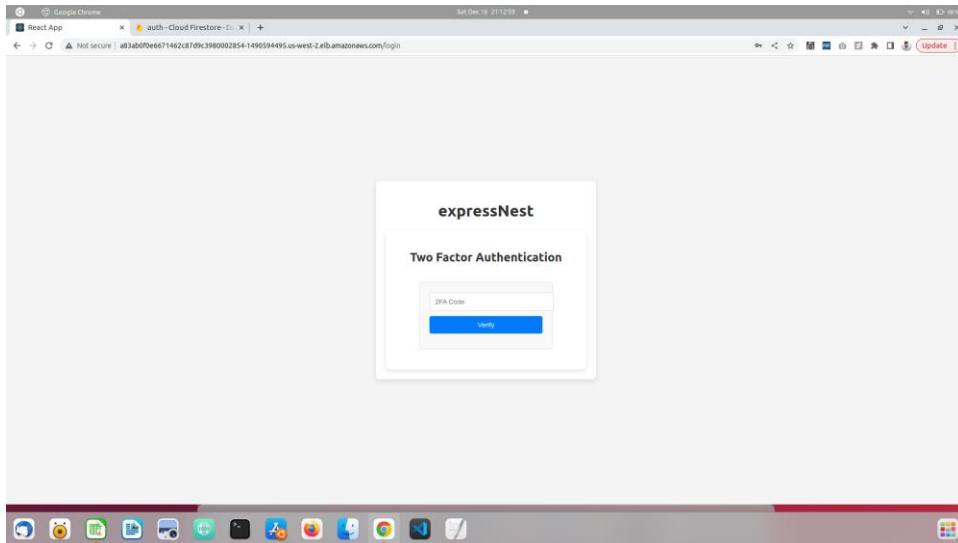


User login

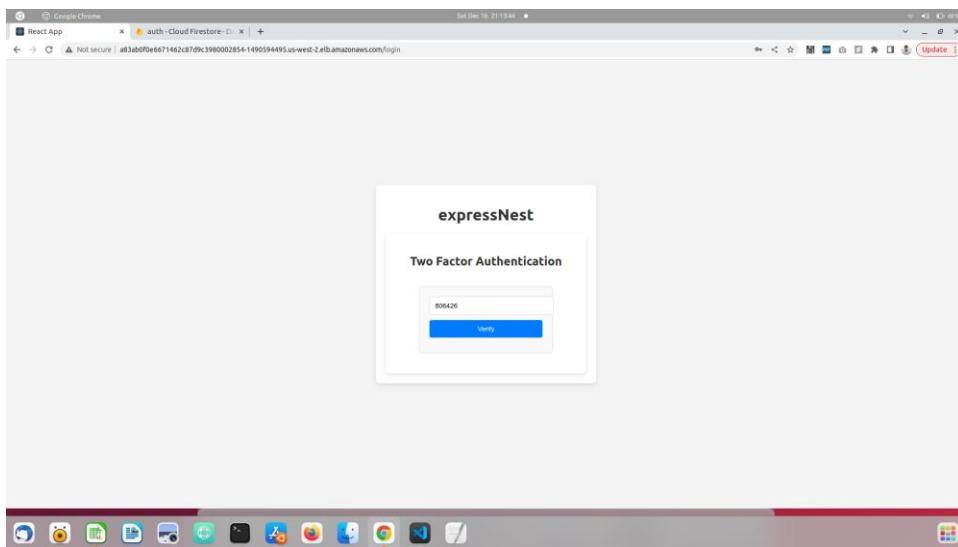


2FA

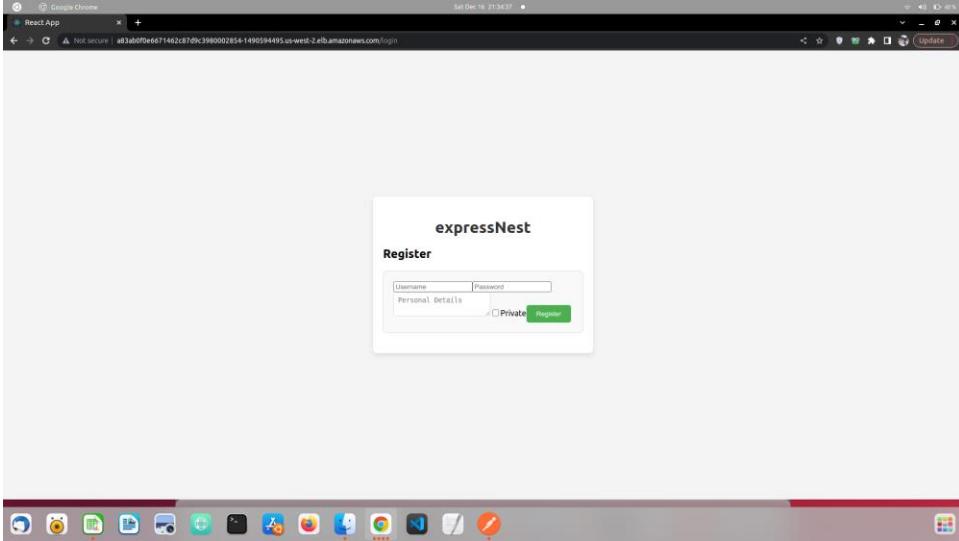
CMPE-272 Enterprise. Software Platforms



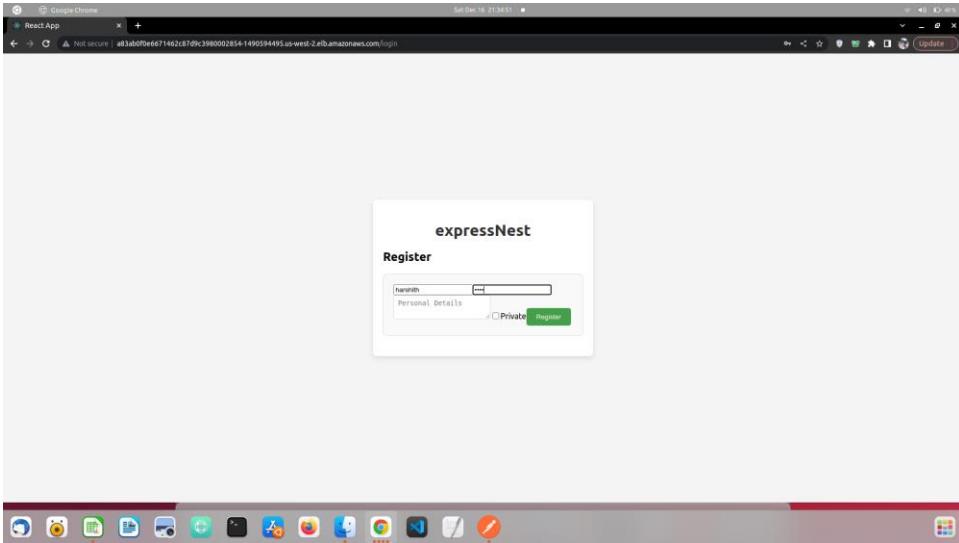
Entering 2FA



Registration Page

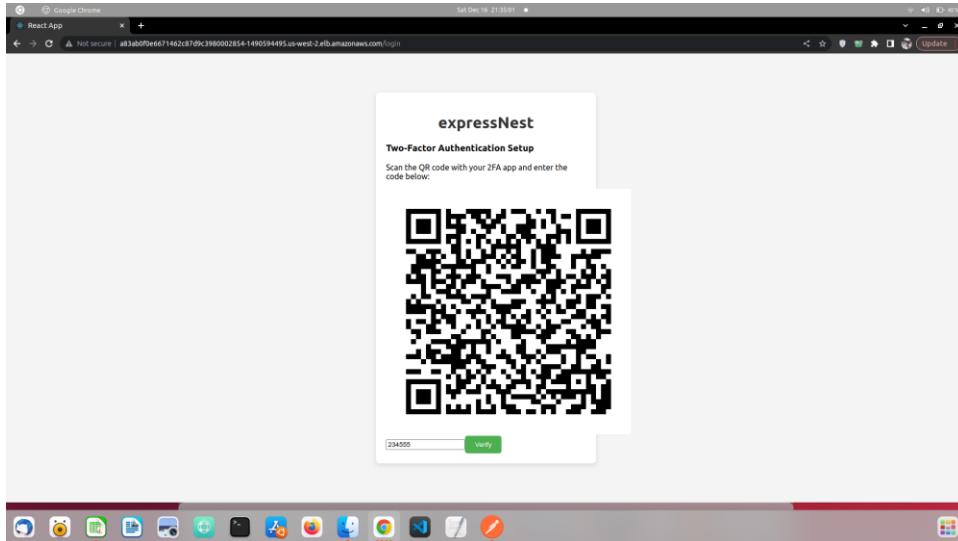


User Registration

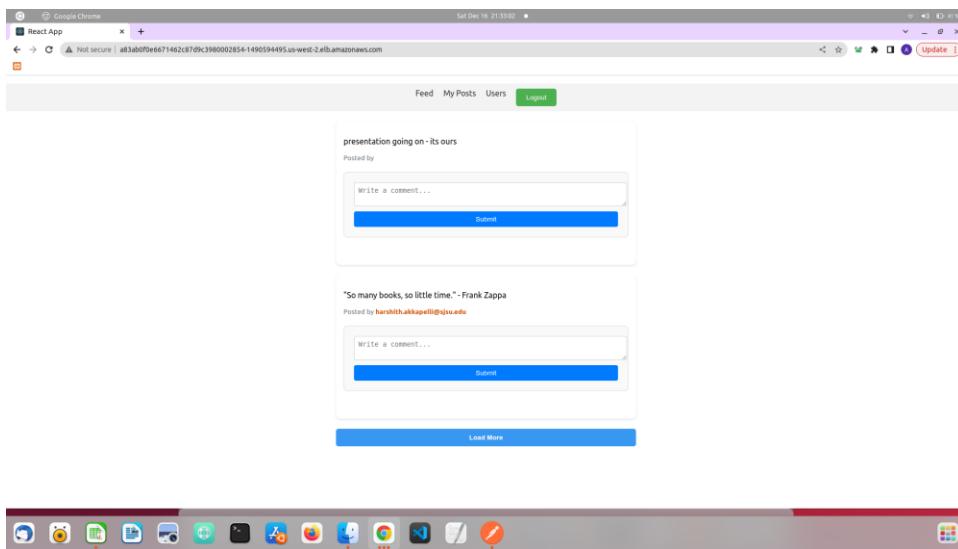


QR Code

CMPE-272 Enterprise. Software Platforms

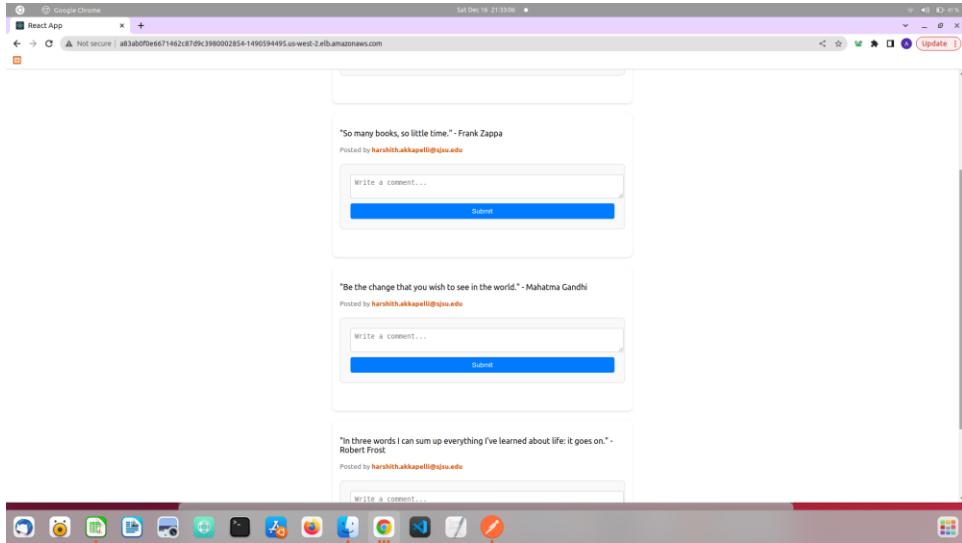


Feed

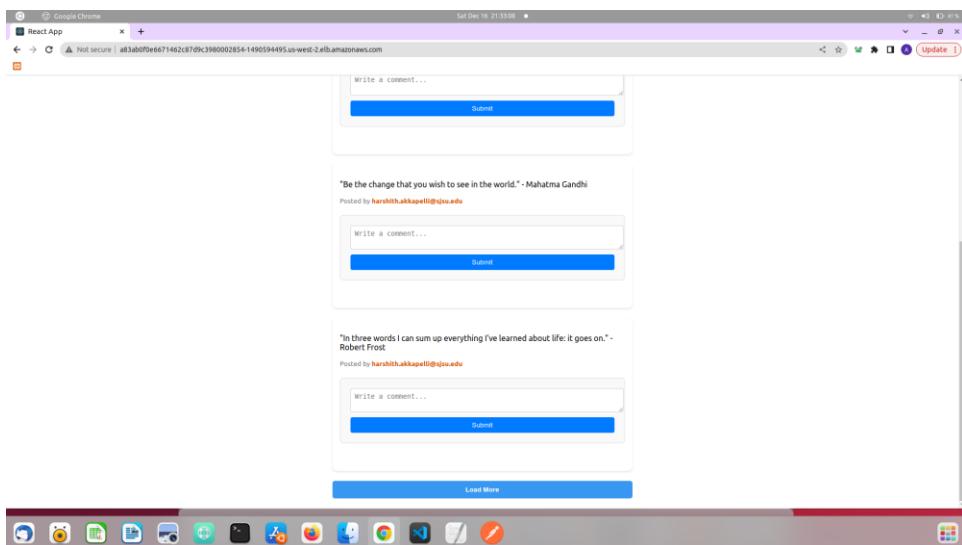


CMPE-272 Enterprise. Software Platforms

Post

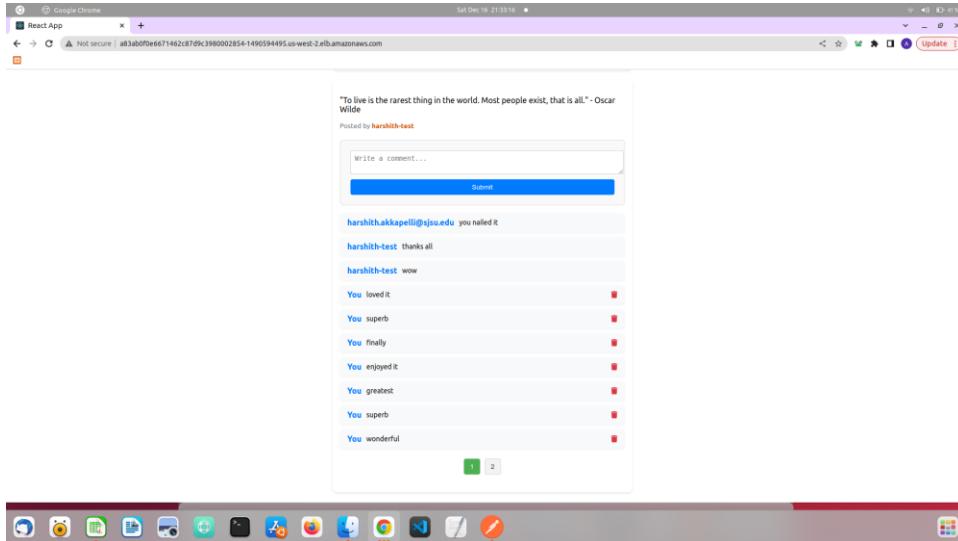


Feed

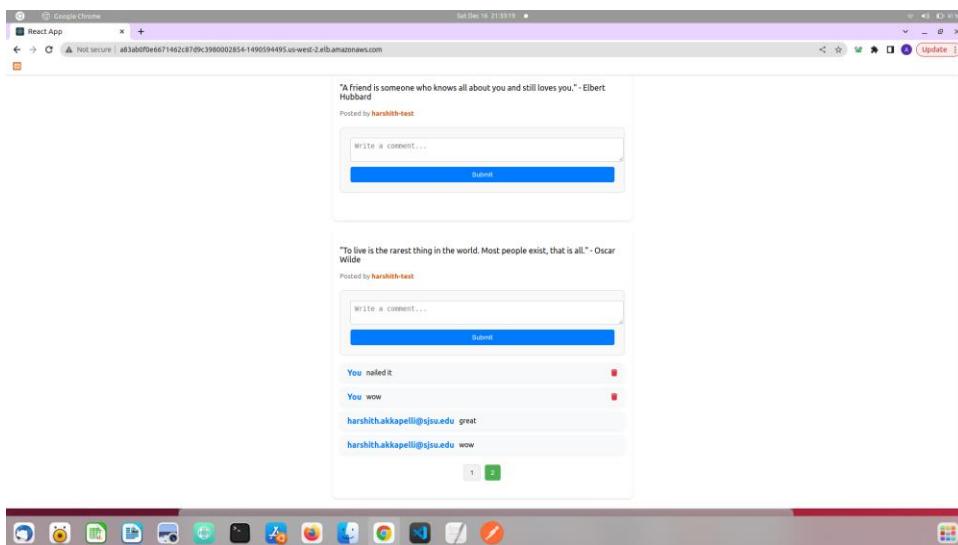


Comments

CMPE-272 Enterprise. Software Platforms

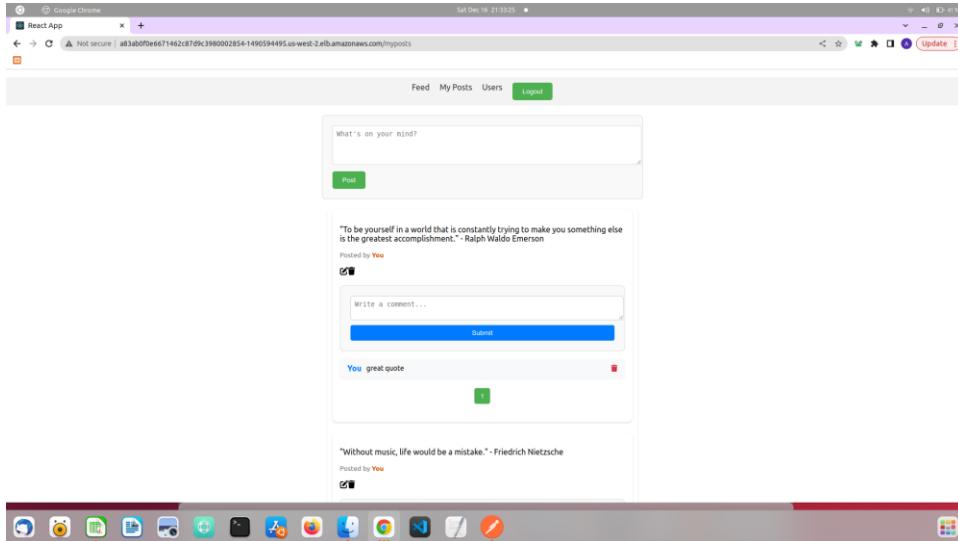


Page 2

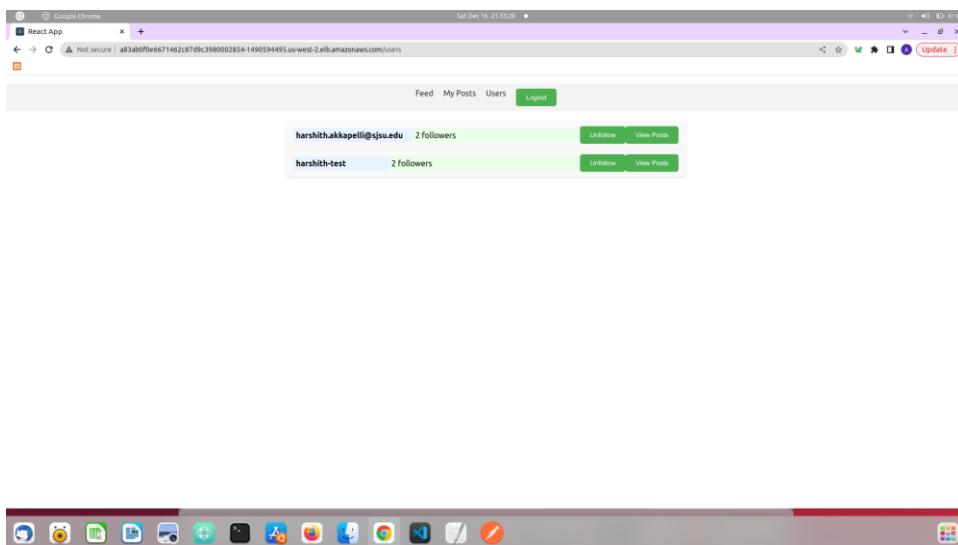


Post creation

CMPE-272 Enterprise. Software Platforms



Users



Deployments:

We have deployed our application in Aws

CMPE-272 Enterprise. Software Platforms

ExpressNestEksCluster-w2

New AMI release versions are available for 1 node group. [Learn more](#)

Cluster info

Status Active	Kubernetes version Info 1.28	Support type Standard support until November 2024	Provider EKS
------------------	---	--	-----------------

Overview | **Resources** | **Compute** | **Networking** | **Add-ons** | **Access** | **Observability** | **Update history** | **Tags**

Details

API server endpoint https://94680DB3B39EC65C5352B61672248DA4.gr7.us-west-2.eks.amazonaws.com	OpenID Connect provider URL https://oidc.eks.us-west-2.amazonaws.com/id/94680DB3B39EC65C5352B61672248DA4	Created December 4, 2023, 13:59 (UTC-08:00)
Certificate authority LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCK1JSURCVENDQWUyZ0F3SUJBZ0lJSjhYmFQQTJxa0F3RFFZSktvWklodmN	Cluster IAM role ARN arn:aws:iam::070788977944:role/eksctl-ExpressNestEksCluster-w2-cluster-ServiceRole-I13XxM5pJ9L1	Cluster ARN arn:aws:eks:us-west-2:070788977944:cluster/ExpressNestEksCluster-w2
		Platform version Info 1.13.10-x86_64

Kafka post events

Message details

Timestamp	Offset	Partition
12/7/2023, 6:53:43 PM	94	0
(1702004023460)		

Key **Value** **Headers**

```

1  {
2   "postId": 91,
3   "actionedBy": 13,
4   "postEventType": "DELETE"
5 }
```

Overview | **Messages** **Schema** | **Configuration**

Production in last hour | **Consumption in last hour** | **Total message**

-- messages | 117 messages | 250

50 messages shown | **Auto-refresh on**

Timestamp	Offset	Partition	Key	Value
1702004023460	94	0		{"postId": 91, "actionedBy": 13, "postEventType": "DELETE"}
1702004019538	93	0		{"postId": 92, "actionedBy": 13, "postEventType": "DELETE"}
1702004010269	92	0		{"postId": 92, "actionedBy": 13, "postEventType": "DELETE"}
1702001386104	91	0		{"postId": 91, "actionedBy": 13, "postEventType": "DELETE"}
1702001371382	90	0		{"postId": 90, "actionedBy": 13, "postEventType": "DELETE"}

CSV | **JSON** | **Raw text**

Kafka follow events

CMPE-272 Enterprise. Software Platforms

The screenshot shows a Kafka message viewer interface. At the top, there are tabs for Overview, Messages (selected), Schema, and Configuration. Below the tabs, a summary box shows Production in last hour (164 messages) and Consumption in last hour (164 messages). The total messages count is 164. A search bar at the top right says "Stream". On the left, there's a filter bar with "Filter by timestamp, offset, key or value", dropdowns for "All partitions" (set to 30) and "Latest". The main area shows 50 messages listed in a table with columns: Timestamp, Offset, Partition, Key, and Value. The Value column contains JSON objects. A modal window titled "Message details" is open, showing a specific message: Timestamp: 12/7/2023, 6:52:21 PM, Offset: 30, Partition: 3. The Value field shows the following JSON:

```

1 {
2   "followeeId": 13,
3   "followerId": 30,
4   "followEventType": "FOLLOW"
5 }

```

At the bottom of the modal, there are download buttons for CSV and JSON, and a link to "How to consume".

EKS Services

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/auth-app-service	ClusterIP	10.100.230.152	<none>	8098/TCP	2d15
h app=auth-app					
service/feeds-app-service	ClusterIP	10.100.80.220	<none>	8081/TCP	4d19
h app=feeds-app					
service/follow-app-service	ClusterIP	10.100.132.82	<none>	8089/TCP	4d15
h app=follow-app					
service/frontend-app-service	ClusterIP	10.100.29.88	<none>	80/TCP	45h
h app=frontend-app					
service/kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	4d21
h <none>					
service/nginx-api-gateway	LoadBalancer	10.100.13.187	a83ab0f0e6671462c87d9c3980002854-1490594495.us-west-2.elb.amazonaws.com	80:31817/TCP	4d20
h app=nginx-api-gateway					
service/post-app-service	ClusterIP	10.100.10.144	<none>	8080/TCP	4d20
h app=post-app					
service/upm-app-service	ClusterIP	10.100.161.58	<none>	8088/TCP	4d13
h app=upm-app					
service/ups-app-service	ClusterIP	10.100.170.132	<none>	8082/TCP	21h
h app=ups-app					
service/wss-app-service	ClusterIP	10.100.142.4	<none>	8090/TCP	20h
h app=wss-app					

Eks pods

The screenshot shows the AWS EKS console interface. At the top, there's a search bar labeled "Filter Nodes by property or value" and a page indicator "1". Below it is a table titled "Nodes (2) Info" with columns: Node name, Instance type, Node group, Created, and Status. Two nodes are listed:

Node name	Instance type	Node group	Created	Status
ip-192-168-14-81.us-west-2.compute.internal	t3.medium	ExpressnestNodeGroup	Created December 4, 2023, 14:10 (UTC-08:00)	Read only
ip-192-168-65-94.us-west-2.compute.internal	t3.medium	ExpressnestNodeGroup	Created December 4, 2023, 14:10 (UTC-08:00)	Read only

Below the nodes is another section titled "Node groups (1) Info" with a "Add node group" button. It has a table with columns: Group name, Desired size, AMI release version, and Launch template. One node group is listed:

Group name	Desired size	AMI release version	Launch template
ExpressnestNodeGroup	2	1.28.3-20231116 Update now	eksctl-ExpressNestEksCluster-w2-nodegroup-ExpressnestNodeGroup

Eks deployment

CMPE-272 Enterprise. Software
Platforms

app=wss-app							
NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
deployment.apps/auth-app-deployment	1/1	1	1	2d15h	auth-app-container	neeharikasingh/auth-app:0.2	app=auth-app
deployment.apps/feeds-app-deployment	1/1	1	1	4d19h	feeds-app-container	neeharikasingh/feeds-app:0.8	app=feeds-app
deployment.apps/follow-app-deployment	1/1	1	1	4d15h	follow-app-container	neeharikasingh/follow-app:0.5	app=follow-app
deployment.apps/frontend-app-deployment	1/1	1	1	45h	frontend-app-container	neeharikasingh/frontend-app:0.5	app=frontend-a
pp							
deployment.apps/nginx-api-gateway	1/1	1	1	4d20h	nginx	neeharikasingh/nginx-app:1.0	app=nginx-api-gateway
deployment.apps/post-app-deployment	1/1	1	1	3d23h	post-app-container	neeharikasingh/post-app:0.6	app=post-app
deployment.apps/test-deployment	1/1	1	1	4d20h	post-app-container	neeharikasingh/post-app:0.3	app=test-app
deployment.apps/upm-app-deployment	1/1	1	1	4d13h	upm-app-container	neeharikasingh/upm-app:0.4	app=upm-app
deployment.apps/ups-app-deployment	1/1	1	1	20h	ups-app-container	neeharikasingh/ups-app:0.4	app=ups-app
deployment.apps/wss-app-deployment	1/1	1	1	20h	wss-app-container	neeharikasingh/wss-app:0.3	app=wss-app

EKS pods logs

```
s: PostID=86: ActionBy=13
2023-12-08 02:09:16 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=86, Offset=86
2023-12-08 02:09:19 [http-nio-8080-exec-3] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=CREATE to topic=post-event
s: PostID=87: ActionBy=13
2023-12-08 02:09:19 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=87, Offset=87
2023-12-08 02:09:22 [http-nio-8080-exec-2] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=CREATE to topic=post-event
s: PostID=88: ActionBy=13
2023-12-08 02:09:22 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=88, Offset=88
2023-12-08 02:09:28 [http-nio-8080-exec-6] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=CREATE to topic=post-event
s: PostID=89: ActionBy=13
2023-12-08 02:09:28 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=89, Offset=89
2023-12-08 02:09:31 [http-nio-8080-exec-10] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=CREATE to topic=post-event
ts: PostID=90: ActionBy=13
2023-12-08 02:09:31 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=90, Offset=90
2023-12-08 02:09:46 [http-nio-8080-exec-6] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=CREATE to topic=post-event
s: PostID=91: ActionBy=13
2023-12-08 02:09:46 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=91, Offset=91
2023-12-08 02:53:30 [http-nio-8080-exec-10] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=CREATE to topic=post-event
ts: PostID=92: ActionBy=13
2023-12-08 02:53:30 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=CREATE
ATE sent: PostId=92, Offset=92
2023-12-08 02:53:39 [http-nio-8080-exec-9] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=UPDATE to topic=post-event
s: PostID=92: ActionBy=13
2023-12-08 02:53:39 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=UPDATE
ATE sent: PostId=92, Offset=93
2023-12-08 02:53:43 [http-nio-8080-exec-6] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - Sending PostEvent=DELETE to topic=post-event
s: PostID=91: ActionBy=13
2023-12-08 02:53:43 [kafka-producer-network-thread | producer-1] INFO edu.sjsu.expressnest.postservice.messaging.PostEventProducer - SUCCESS: PostEvent=DELETE
ATE sent: PostId=91, Offset=94
```

CMPE-272 Enterprise. Software Platforms

Resource types

- ▼ Workloads
 - PodTemplates
 - Pods**
 - ReplicaSets
 - Deployments
 - StatefulSets
 - DaemonSets
 - Jobs
 - CronJobs
 - PriorityClasses
 - HorizontalPodAutoscalers
- ▶ Cluster
- ▶ Service and networking
- ▶ Config and secrets
- ▶ Storage
- ▶ Authentication
- ▶ Authorization

Workloads: Pods (16)

Pod is the smallest and simplest Kubernetes object. A Pod represents a set of running containers on your cluster.

[Learn more ↗](#)

Name	Age
auth-app-deployment-d55d4b696-7gcb7	Created Dec 7, 2023, 17:14 (UTC-08:00)
aws-node-28mvv	Created Dec 4, 2023, 14:10 (UTC-08:00)
aws-node-wsrjd	Created Dec 4, 2023, 14:10 (UTC-08:00)
coredns-59754897cf-9mqdd	Created Dec 4, 2023, 14:04 (UTC-08:00)
coredns-59754897cf-x59fx	Created Dec 4, 2023, 14:04 (UTC-08:00)
feeds-app-deployment-58dbf66b98-qrw25	Created Dec 5, 2023, 10:27 (UTC-08:00)
follow-app-deployment-657db4fc8c-8cc26	Created Dec 6, 2023, 21:48 (UTC-08:00)
...	Created

Resource types

- ▼ Workloads
 - PodTemplates
 - Pods**
 - ReplicaSets
 - Deployments
 - StatefulSets
 - DaemonSets
 - Jobs
 - CronJobs
 - PriorityClasses
 - HorizontalPodAutoscalers
- ▶ Cluster
- ▶ Service and networking
- ▶ Config and secrets
- ▶ Storage
- ▶ Authentication
- ▶ Authorization

Workloads: Pods (16)

Pod is the smallest and simplest Kubernetes object. A Pod represents a set of running containers on your cluster.

[Learn more ↗](#)

Name	Age
nginx-api-gateway-678cf6d968-b4slf	Created 21 hours ago
post-app-deployment-5bfd47b5d6-gv2vd	Created Dec 5, 2023, 12:47 (2023-12-05T20:47:58Z)
test-deployment-f9dbb589b-5th2z	Created Dec 4, 2023, 14:41 (UTC-08:00)
upm-app-deployment-74f4555d-rs4sd	Created Dec 5, 2023, 20:19 (UTC-08:00)
ups-app-deployment-854749cb95-fk7q8	Created 20 hours ago
wss-app-deployment-855d8c66b-xprsj	Created 19 hours ago

Scaling Strategies:

Frontend Performance

1. Moving the static files to CDN (Amazon CloudFront)
2. GraphQL for UI facing queries - GraphQL backend (another microservice) which will aggregate all data from services and send it back to the UI. For example, currently for getting the feeds, UI first calls the feed-service to get the all the postIds, then it calls the post-service to get the content of the posts for the various postIds. It also calls the user-profile management service to get the user details. In this case we can have a separate microservice which acts as an aggregator for all these operations.

Infrastructure Problems

3. For Postgres SQL database:
 - When the database is not able to handle amount of read queries received, we will add • Caching for all the entities
 - We will analyze what are the entities for which we have higher cache misses and check whether the entities can be easily divided into rarely and frequently modified fields and look into moving frequently modified fields into its own table e.g. (columns like totalNoOfreactions totalNoOfComments which are currently present in the post table)
 - Increase the number of read replicas
 - When the database is not able to handle the amount of write queries then we will:
 - Add sharding and increase the number of shards. We can shard it based on the user.
 - We will analyze the fields that are frequently modified and plan on moving those to a different database which can support higher write QPS
4. Kafka Event Processing:
 - We will increase the number of partitions in Kafka.
 - We will increase the number of consumers.
 - Each consumer can do batch processing of multiple events at a time.
5. Currently we have a single EKS cluster hosted. We will explore solutions for multi-cluster options in the same region. Then we will add more number of clusters.

Special Cases

6. **Celebrity Problem:** Currently the user feed is generated asynchronously(feed-service). Whenever user A creates a post. A post event is sent to the post-events topic in Kafka. Feed Service listens to that topic and it updates the user feeds of the users that are following User A. In case the User A is a celebrity who has millions of followers, then the number of feeds that would be generated by feed-service would increase and the system would be overwhelmed in that case. In order to mitigate this problem, we will identify these special celebrity users in the system (using the number of followers). Once a celebrity, always be a celebrity, even if the number of followers fall below the threshold. Whenever there are posts made by these users, the feed service will ignore these events. Now whenever a user's feed is generated, it will query all followed celebrities' posts (post service) synchronously based on the current page. This will be merged with the result from the feeds table.
7. User following many people: System will identify such users; we will also put a threshold on the number of historical feeds for such users. We might also delete historical feeds.

Challenges:

- Microservices interaction methods.
- Initially for authentication we started using Keycloak. But we faced issues with it due to no proper documentation.
- We have faced CORS issues during integrating frontend with backend.
- Application deployments.

Learnings and Experiences:

- Gained a lot of hands-on experience on various technologies used in the project.
- Got to know how to build, integrate, deploy and establish communication between various Services.
- Overcoming the bottlenecks and hurdles during the process.

Lessons learnt:

- Should have regular standups and improved communication.
- We should have started with building major basic functionalities, integrating and deploying them and then we could have added more functionalities.
- We know now what NOT to do!!!! (biggest lesson learnt).

Thing that could have been improved:

- More functionality.
- The User Interface could have been even more interactive and appealing to the user.