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

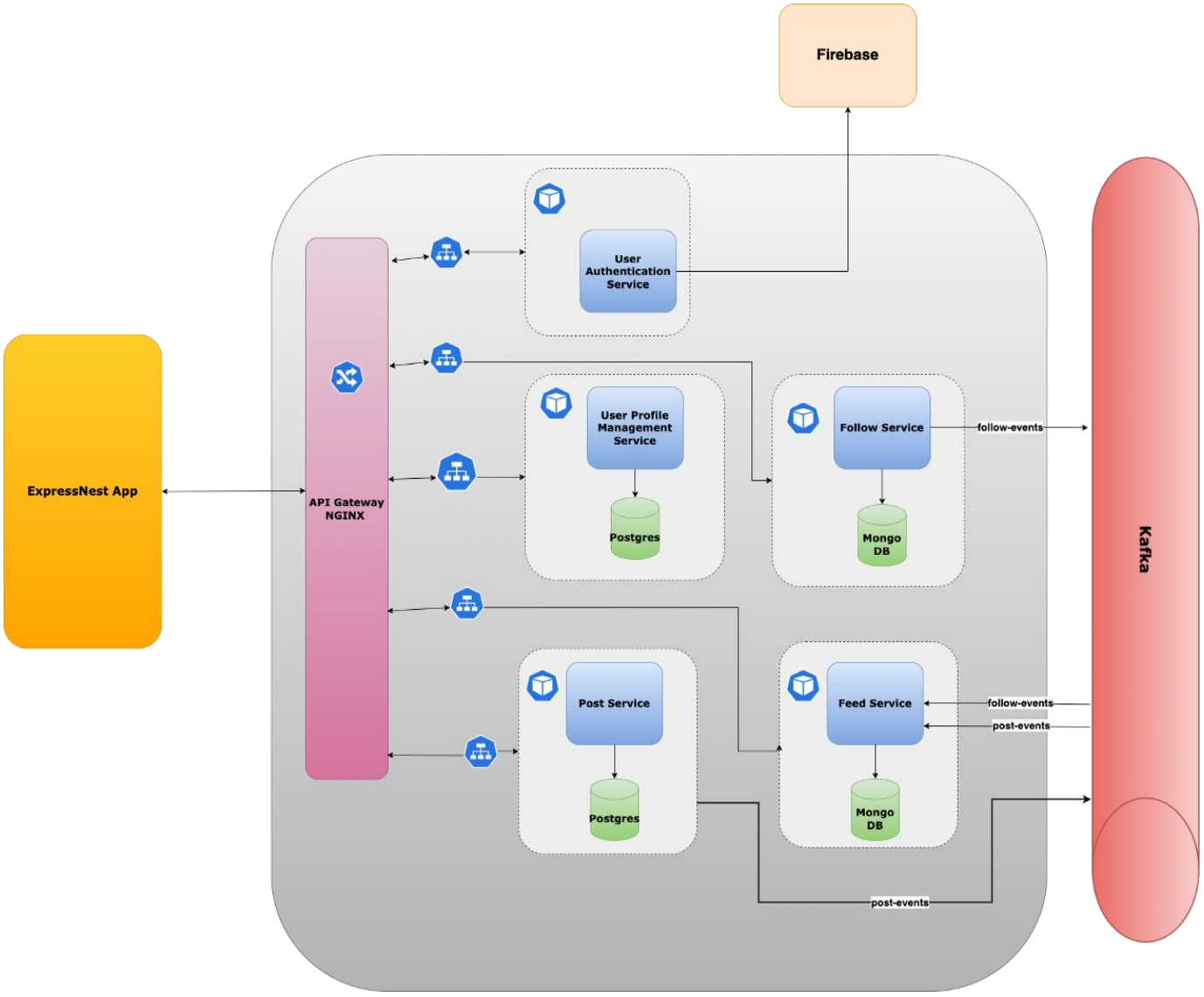
Frontend:

React Js,Html, Css

Caching:

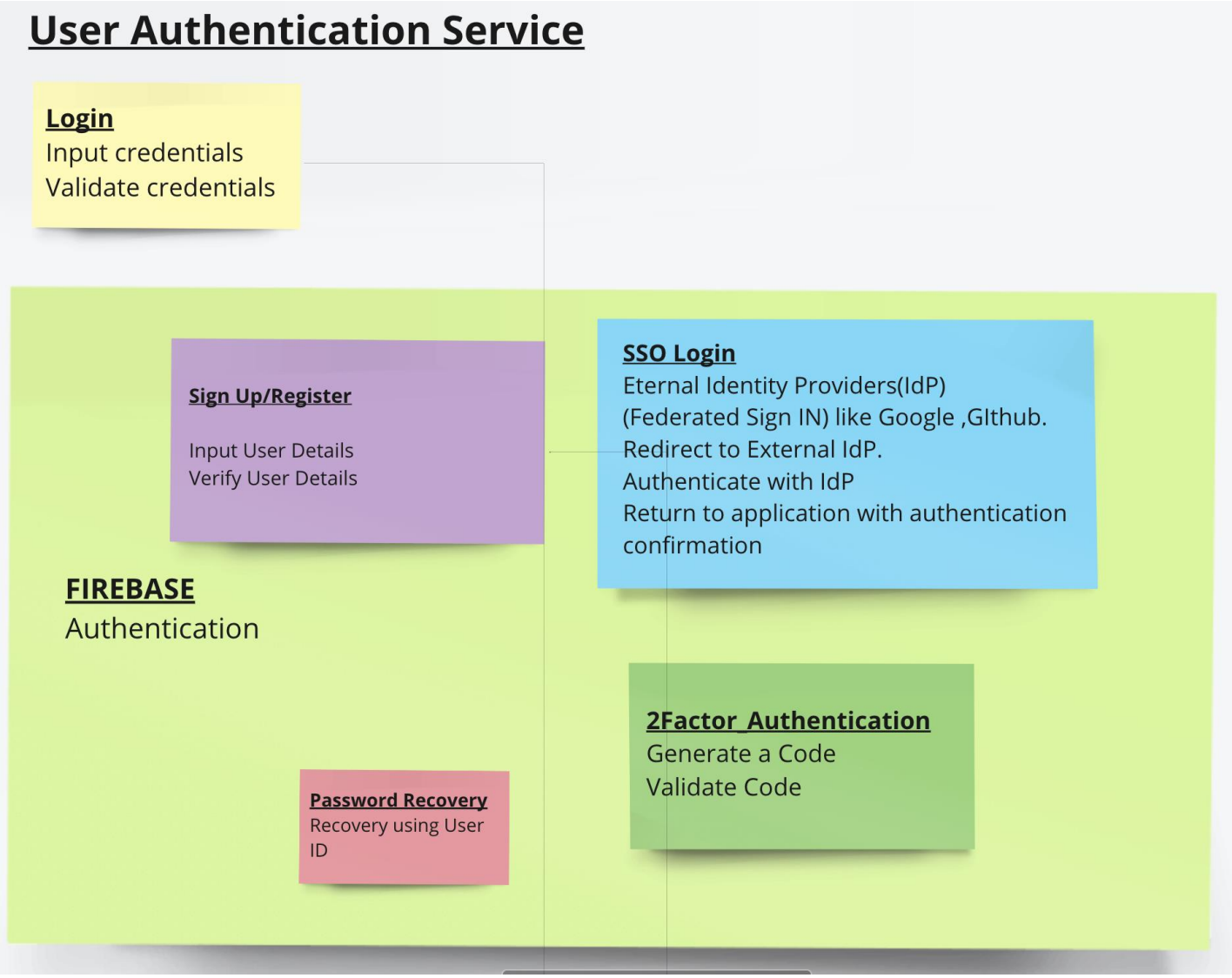
Redis

Architecture:



Microservices:

Authentication Service:



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

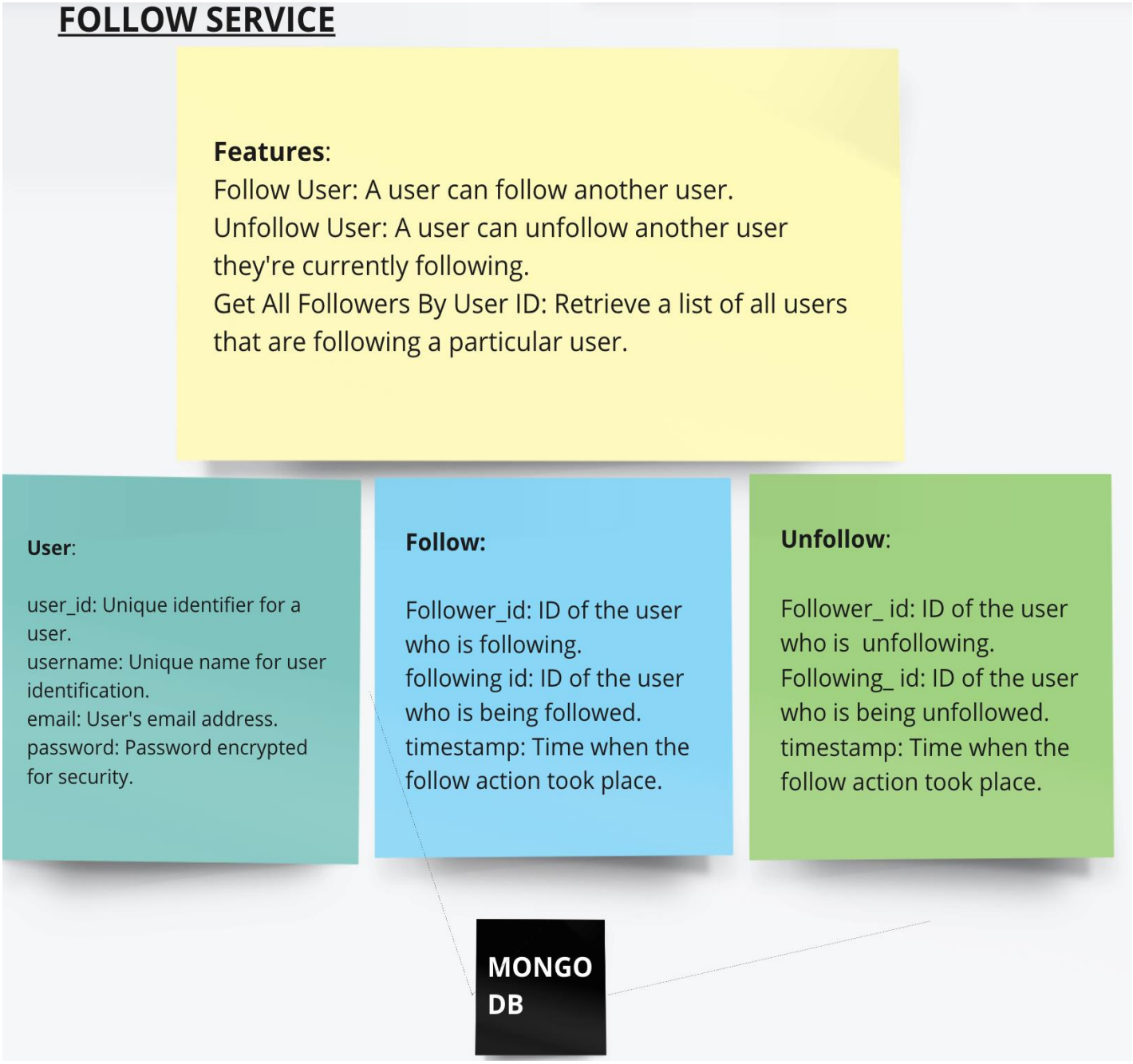
Post can have attachments like images and videos.
If a new post is made by using the share post button then the shared post will be saved as an attachment in the DB and the user can add their own additional text on top of that.

Followers can like/unlike the post
Followers can comment on the post
Followers can share the post (retweet)

Database
Postgres database

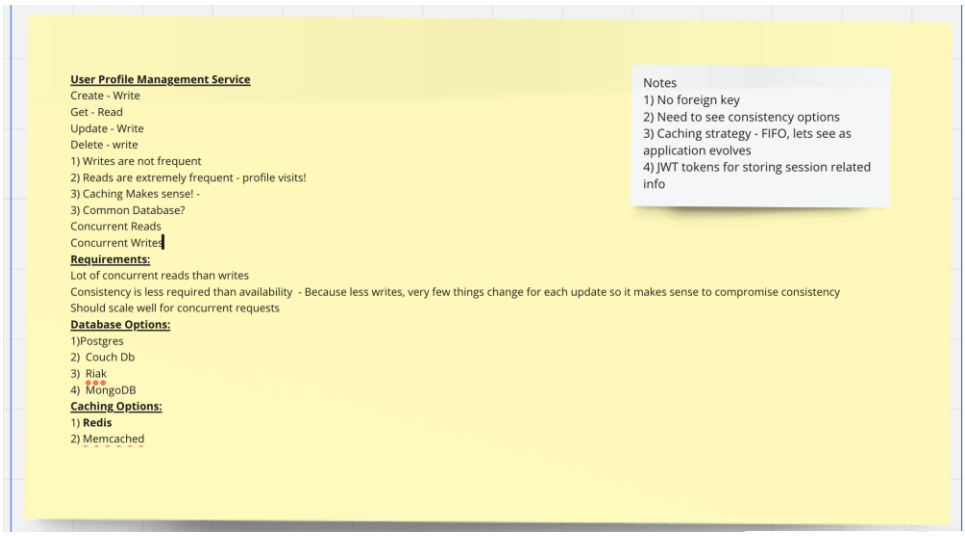
Caching - Redis

Follow Service:



Feed Service:

User profile Management Service:



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",} 5208736.0
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 2.5810546875
# 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 18809.0
# 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
```

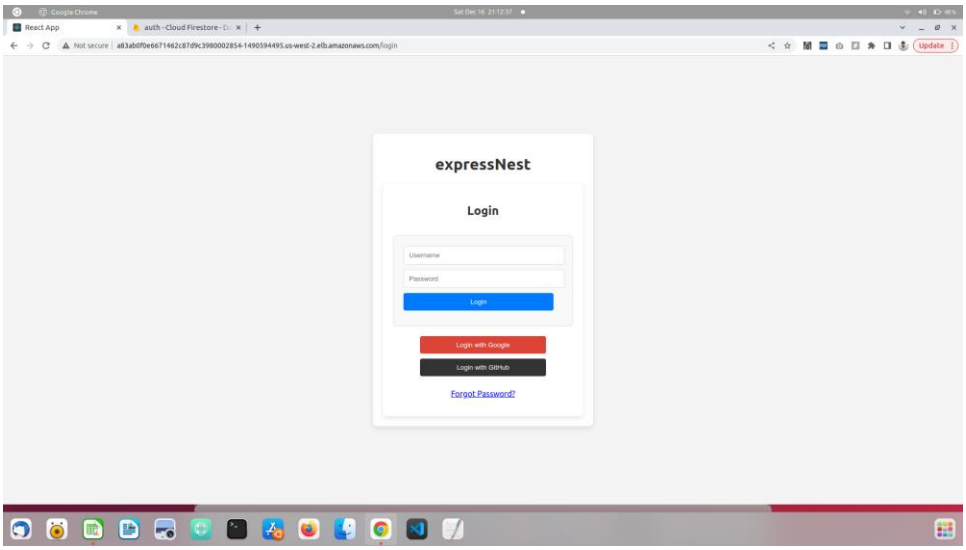
```
# TYPE jvm_threads_live_threads gauge
jvm_threads_live_threads 25.0
# HELP process_start_time_seconds Start time of the process since unix epoch.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.702785300311E9
# HELP jvm_gc_memory_allocated_bytes_total Incremented for an increase 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="G1 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_idle 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{area="nonheap",id="CodeHeap 'profiled nmethods'",} 1.22908672E8
```

```
# 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 4.294967296E9
# HELP process_files_max_files The maximum file descriptor count
# TYPE process_files_max_files gauge
process_files_max_files 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
```

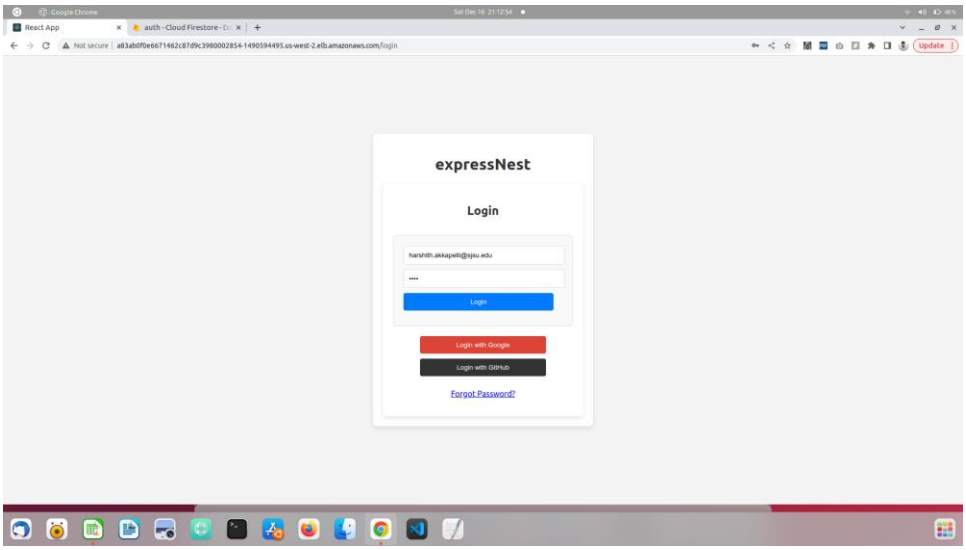
User Interface:

CMPE-272 Enterprise. Software Platforms

Login page

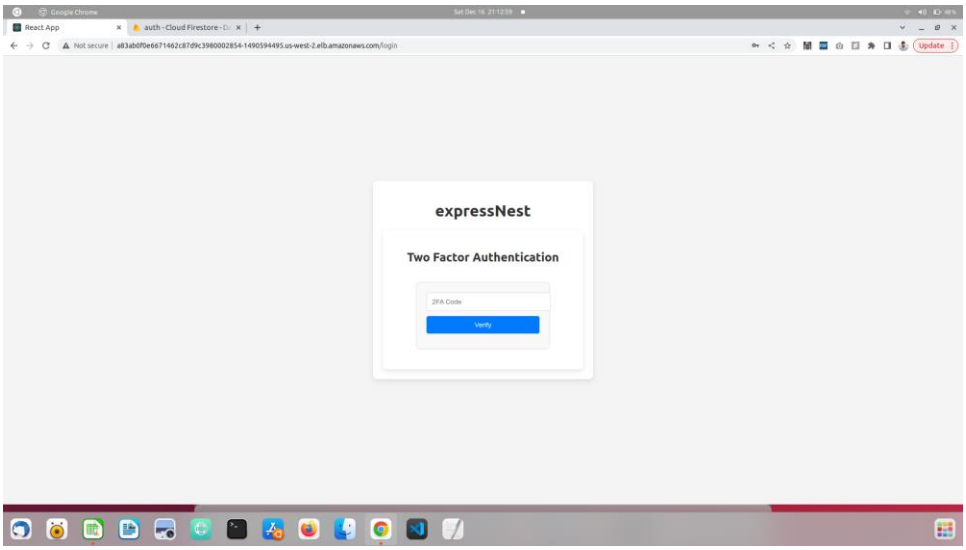


User login

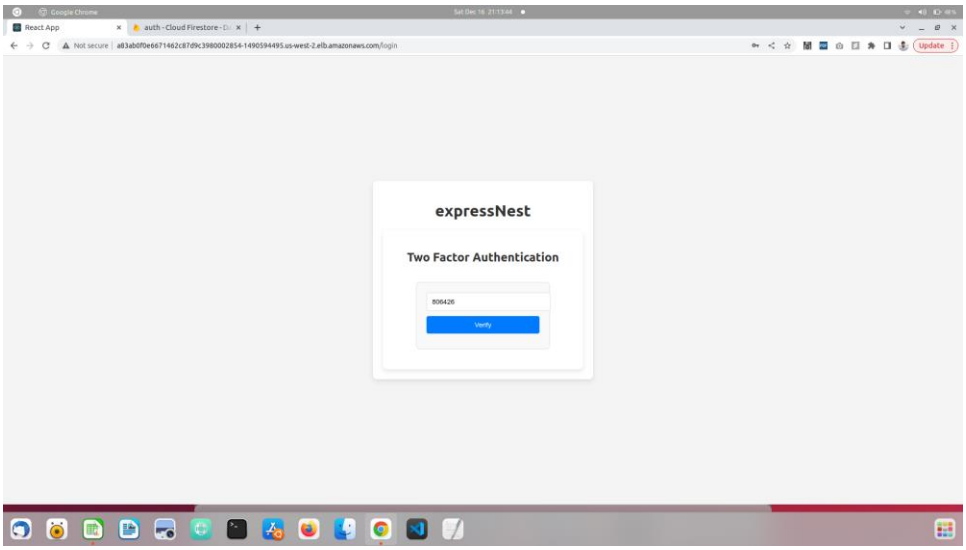


2FA

CMPE-272 Enterprise. Software Platforms

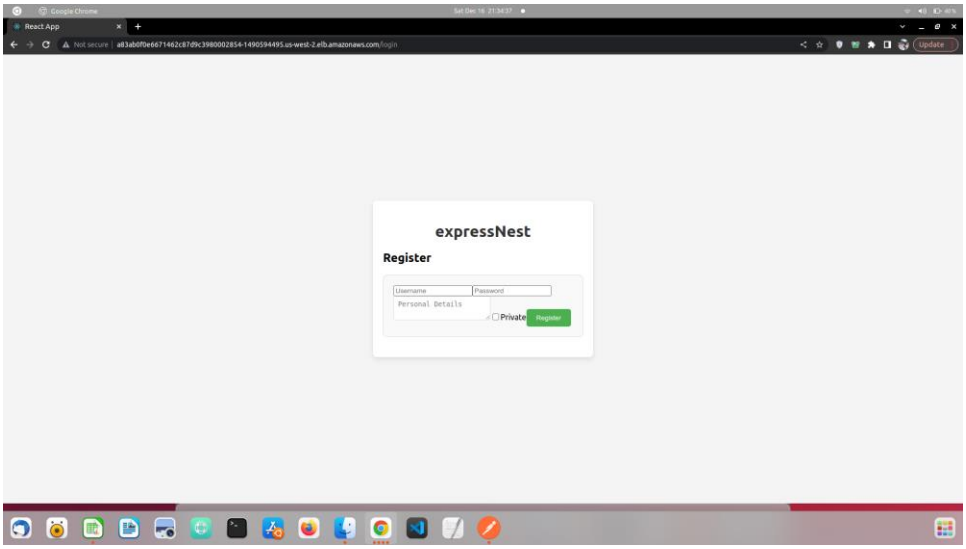


Entering 2FA

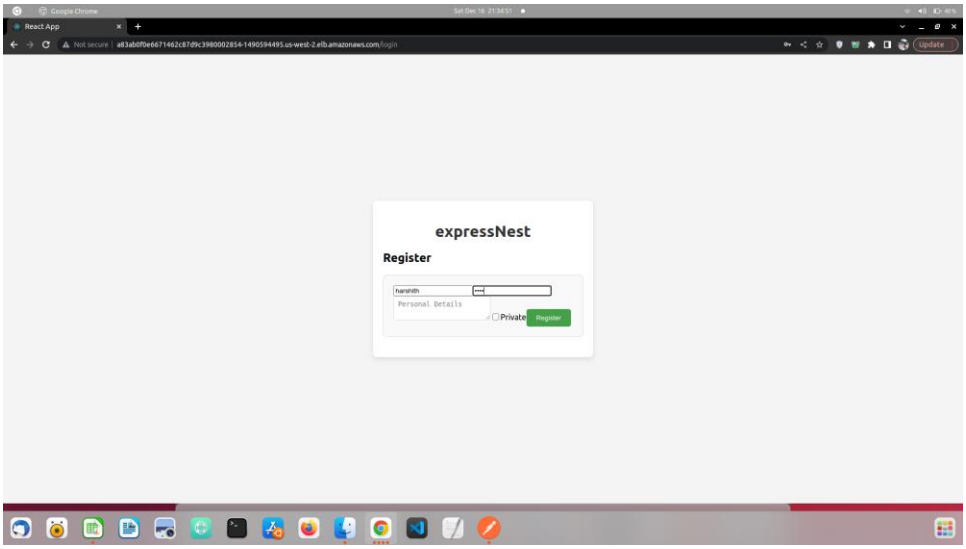


CMPE-272 Enterprise. Software Platforms

Registration Page

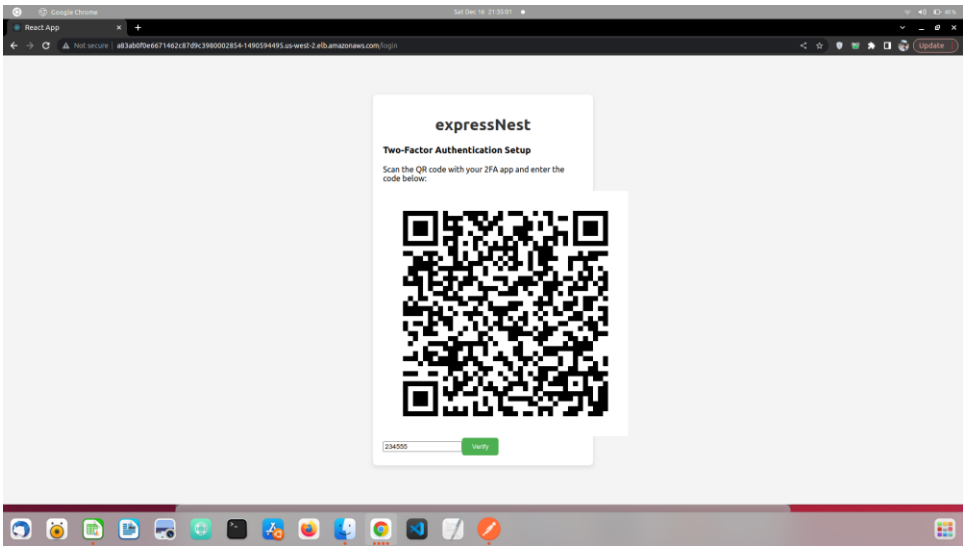


User Registration

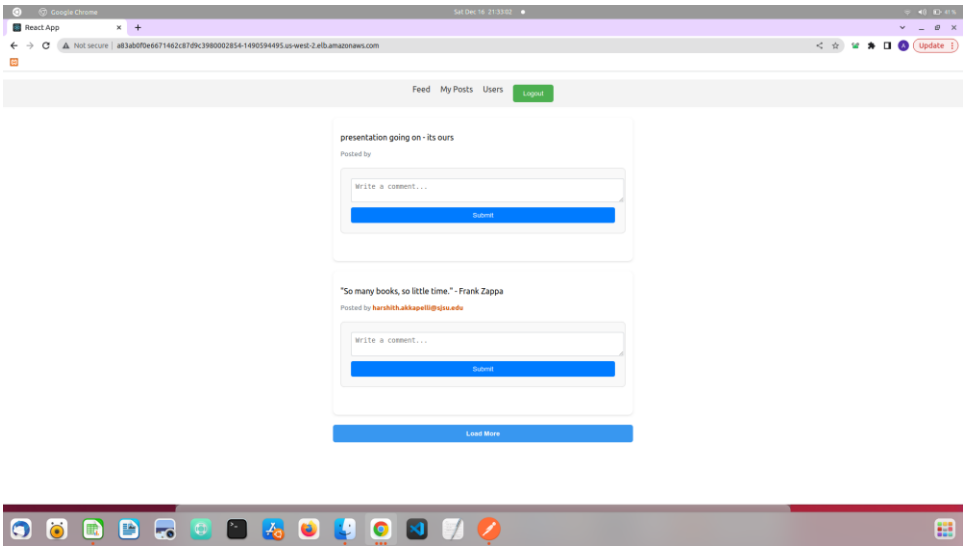


QR Code

CMPE-272 Enterprise. Software Platforms

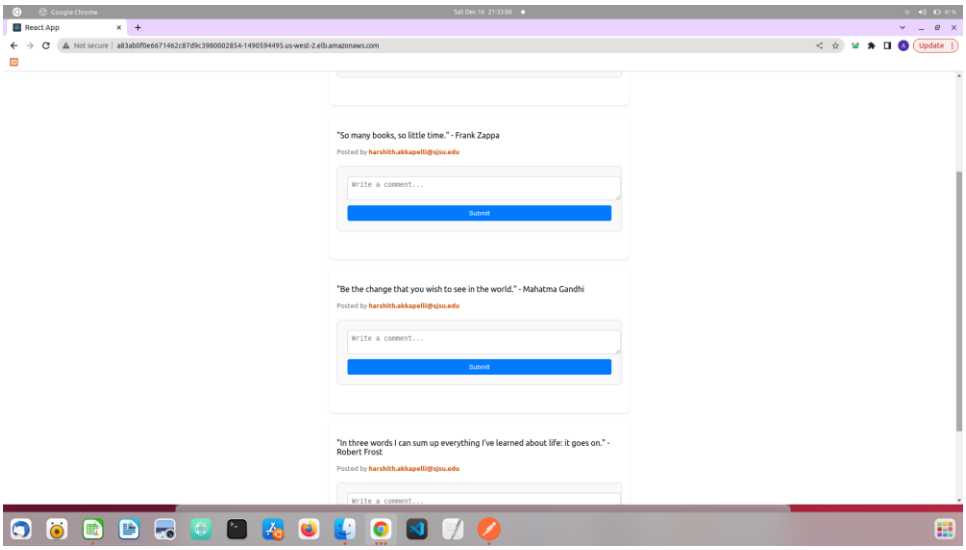


Feed

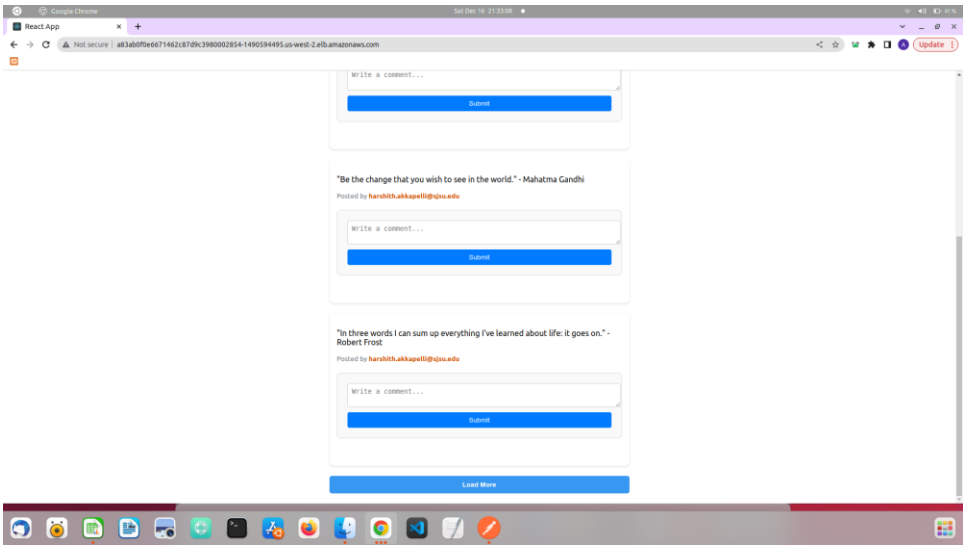


CMPE-272 Enterprise. Software Platforms

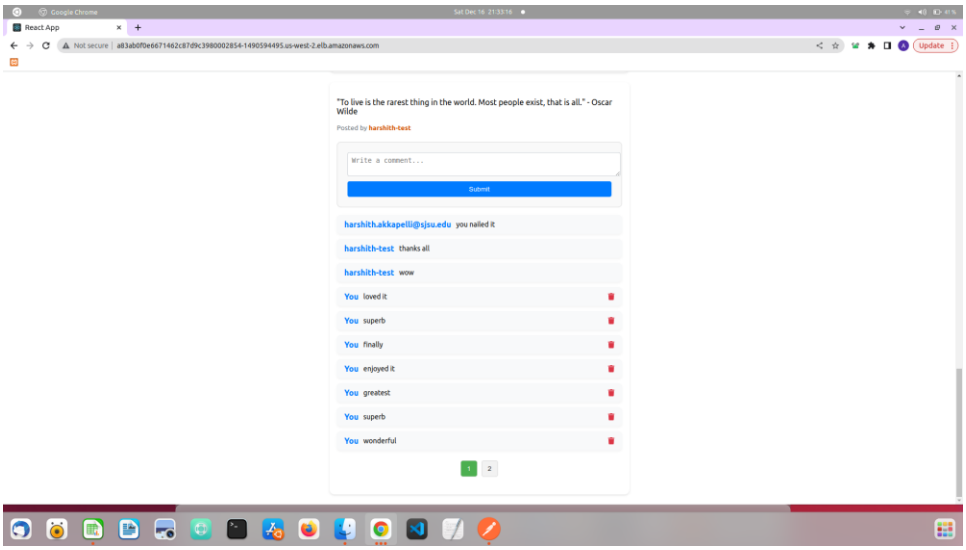
Post



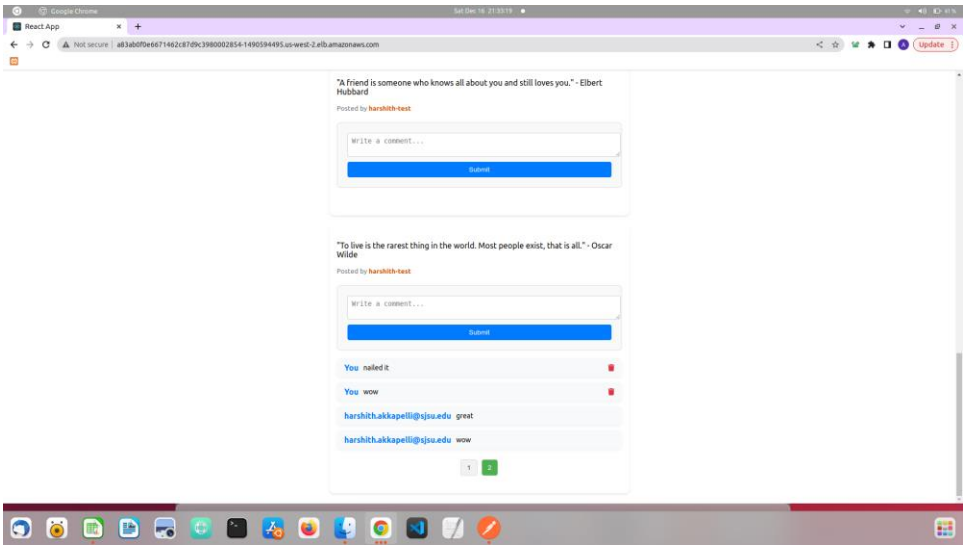
Feed



Comments

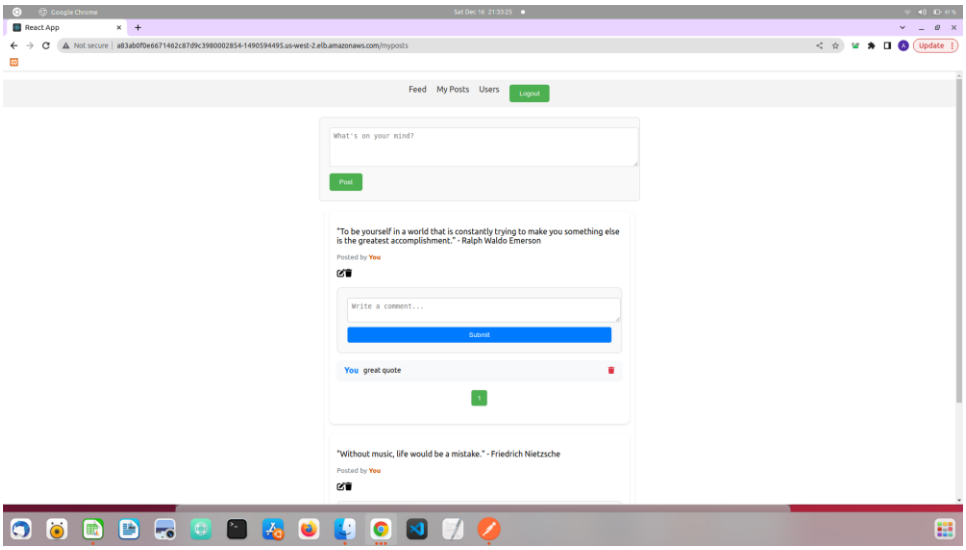


Page 2

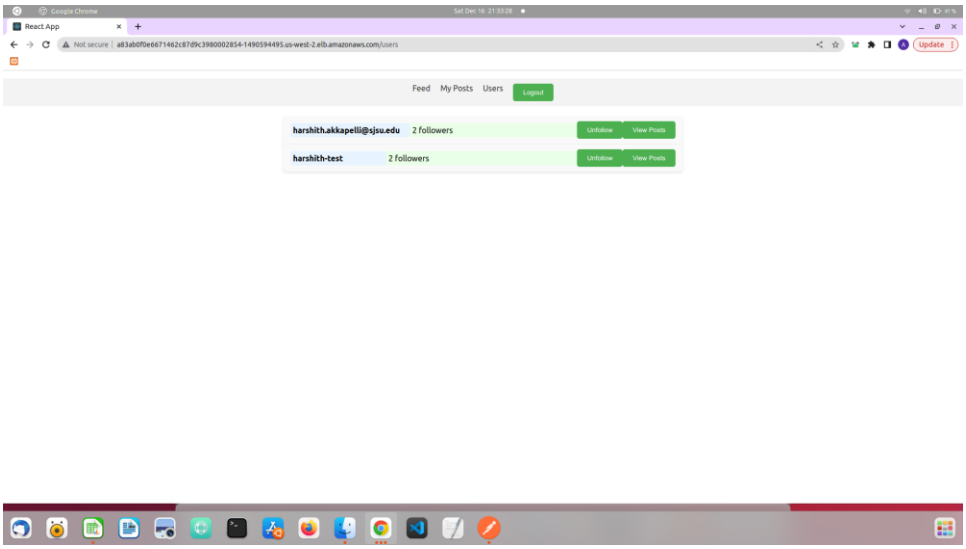


Post creation

CMPE-272 Enterprise. Software Platforms

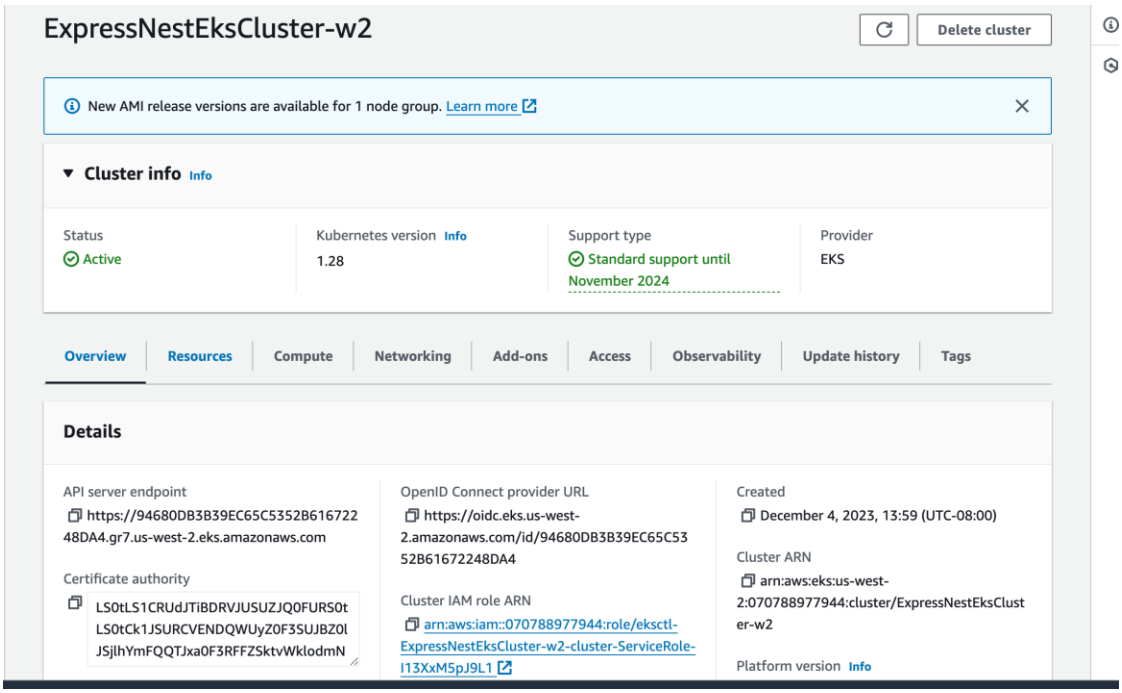


Users

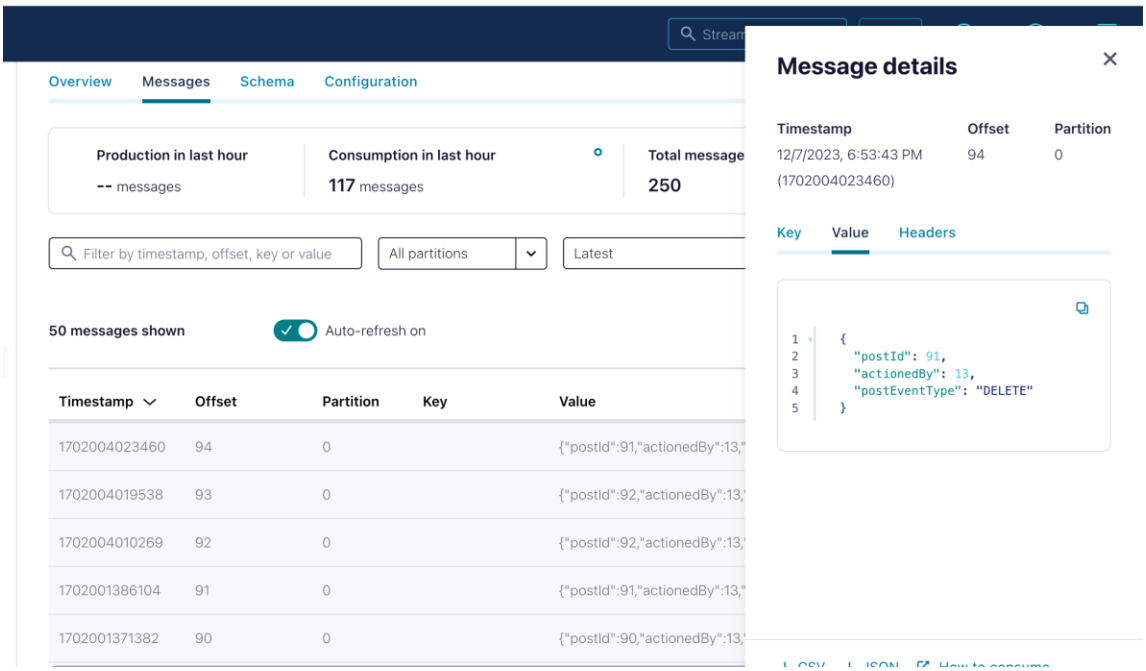


Deployments:

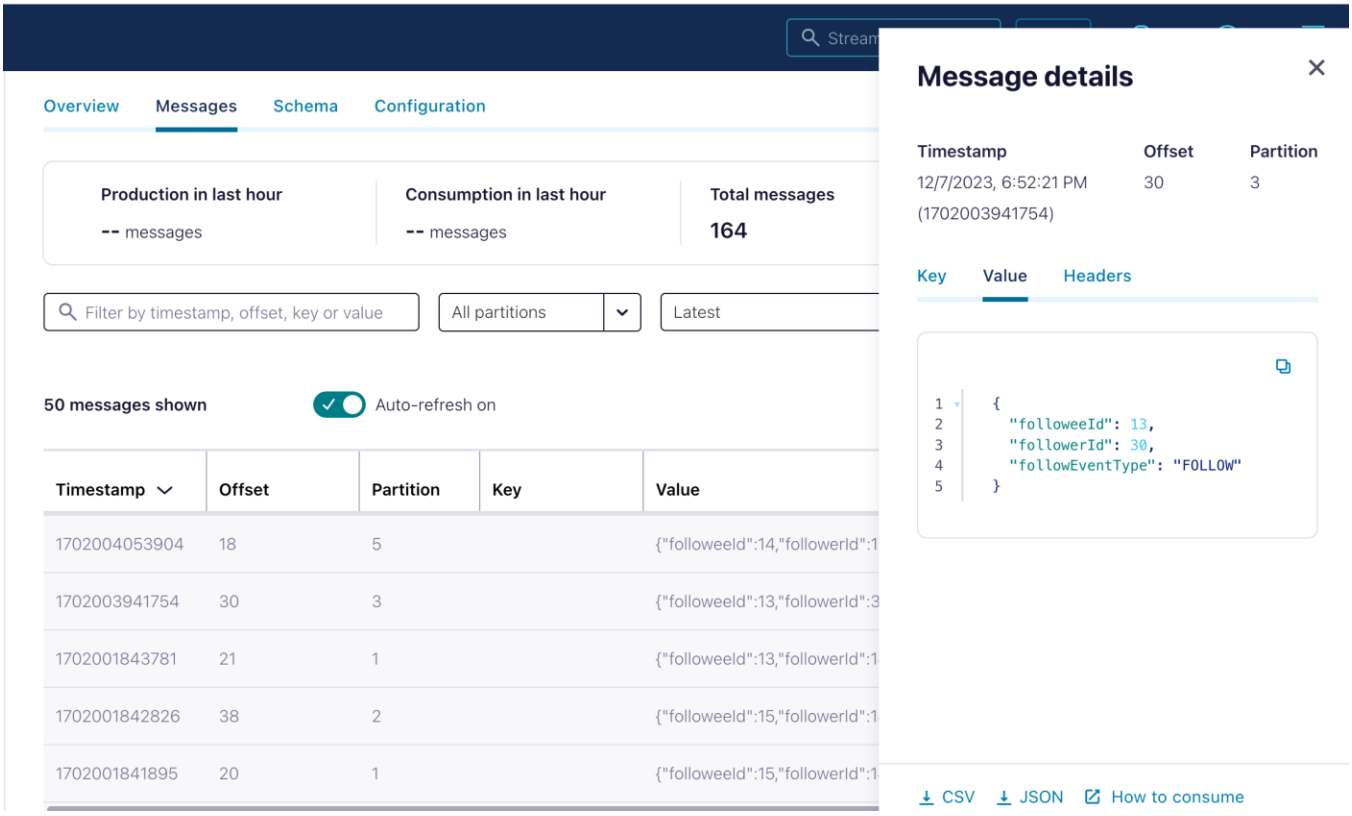
We have deployed our application in Aws



Kafka post events



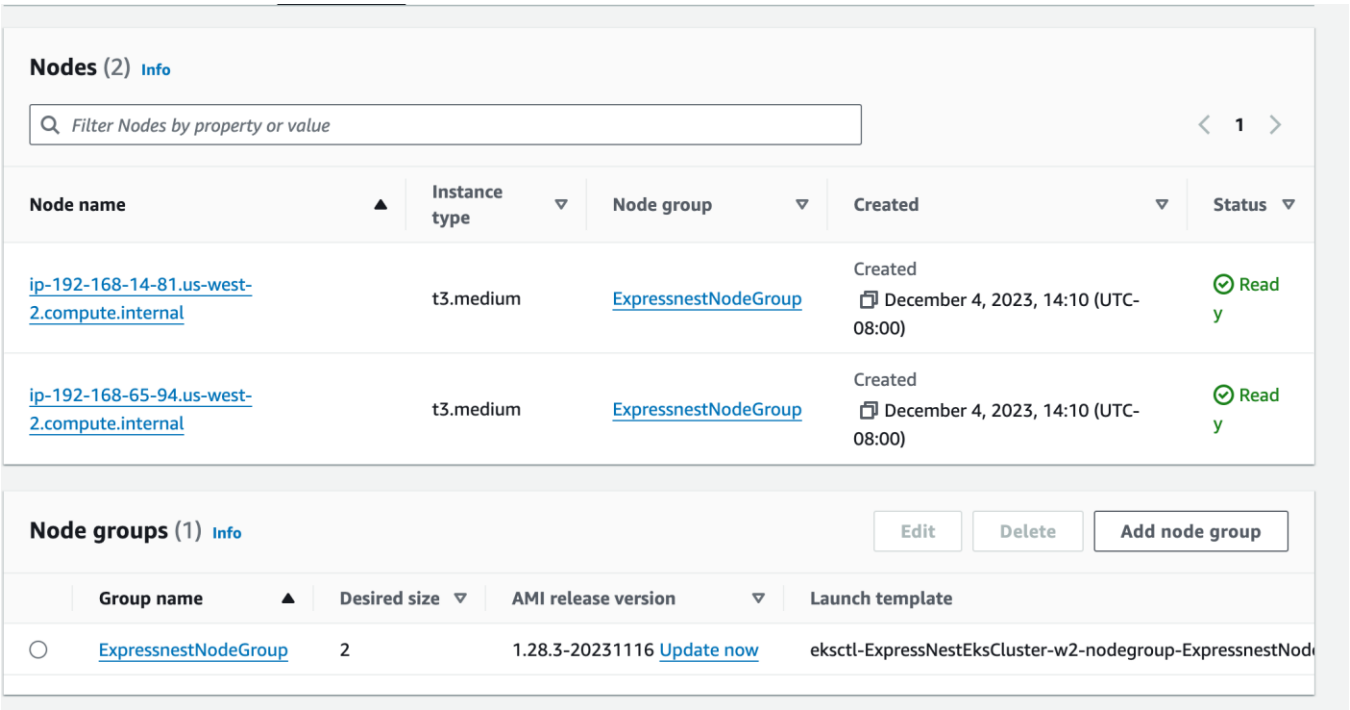
Kafka follow events



EKS Services

```
NAME                          TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)        AGE
    SELECTOR
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
    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
    app=ups-app
service/wss-app-service       ClusterIP      10.100.142.4     <none>                                                                   8090/TCP       20h
    app=wss-app
```

Eks pods



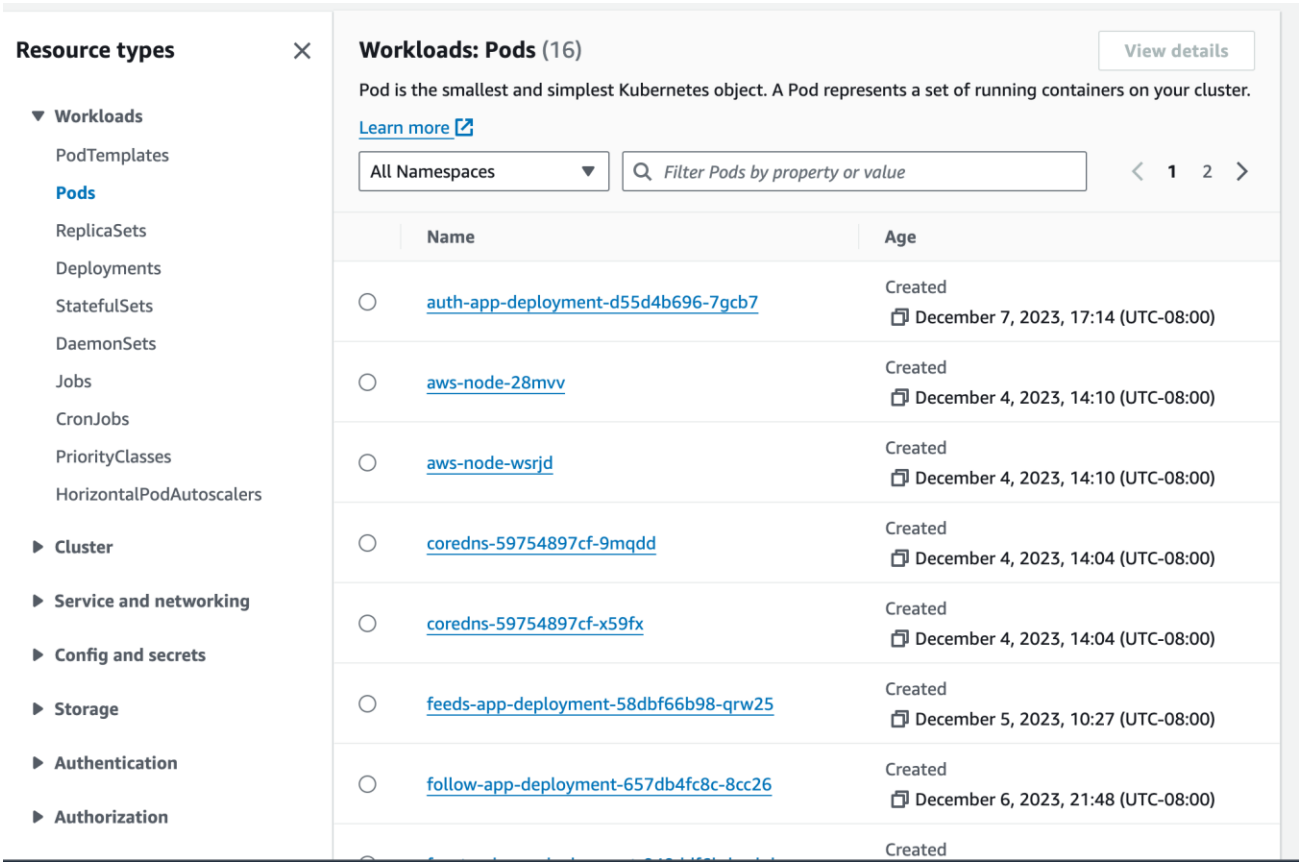
Eks deployment

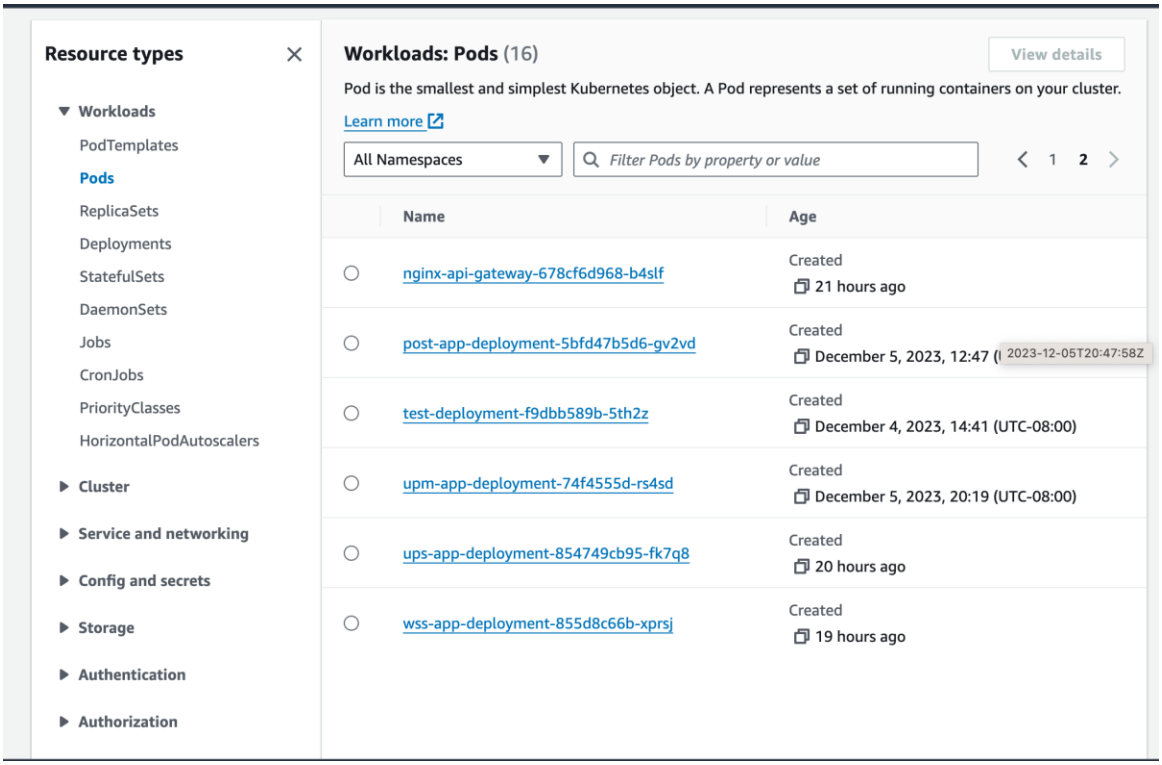
```
    app=wss-app

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=CRE
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=CRE
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=CRE
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=CRE
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-even
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=CRE
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=CRE
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-even
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=CRE
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=UPD
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=DEL
ETE sent: PostId=91, Offset=94
```





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.