

Movie Magic-Smart Movie Ticket Booking System

Project Description:

With the increasing demand for a seamless and modern movie-watching experience, traditional ticket booking methods often fall short due to long queues, limited availability, and inconsistent service. To address this, the development team introduced Movie Magic—a smart, cloud-based movie ticket booking system. Built using Flask for backend development, hosted on AWS EC2, and integrated with DynamoDB for dynamic data management, the platform allows users to register, log in, and book movie tickets online with ease. Users can search for movies and events based on location, view real-time seat availability, and complete their bookings in just a few clicks. Upon booking, AWS SNS sends instant email notifications confirming ticket details, enhancing user engagement and trust. This cloud-native solution streamlines the entire movie ticketing process, ensuring fast, scalable, and user-friendly access to entertainment for all.

Scenarios :

Scenario 1: Efficient Ticket Booking System for Users

In the Movie Magic System, AWS EC2 provides a reliable infrastructure capable of handling multiple users accessing the platform simultaneously. For example, a user can log in, navigate to the movie selection page, and seamlessly browse available shows and events in their city. They can then select a showtime, pick their preferred seats using an interactive layout, and confirm the booking—all in real-time. Flask manages backend processes, ensuring smooth data flow and quick response times even during high-traffic periods such as weekends or blockbuster releases.

Scenario 2: Seamless Booking Confirmation Notifications

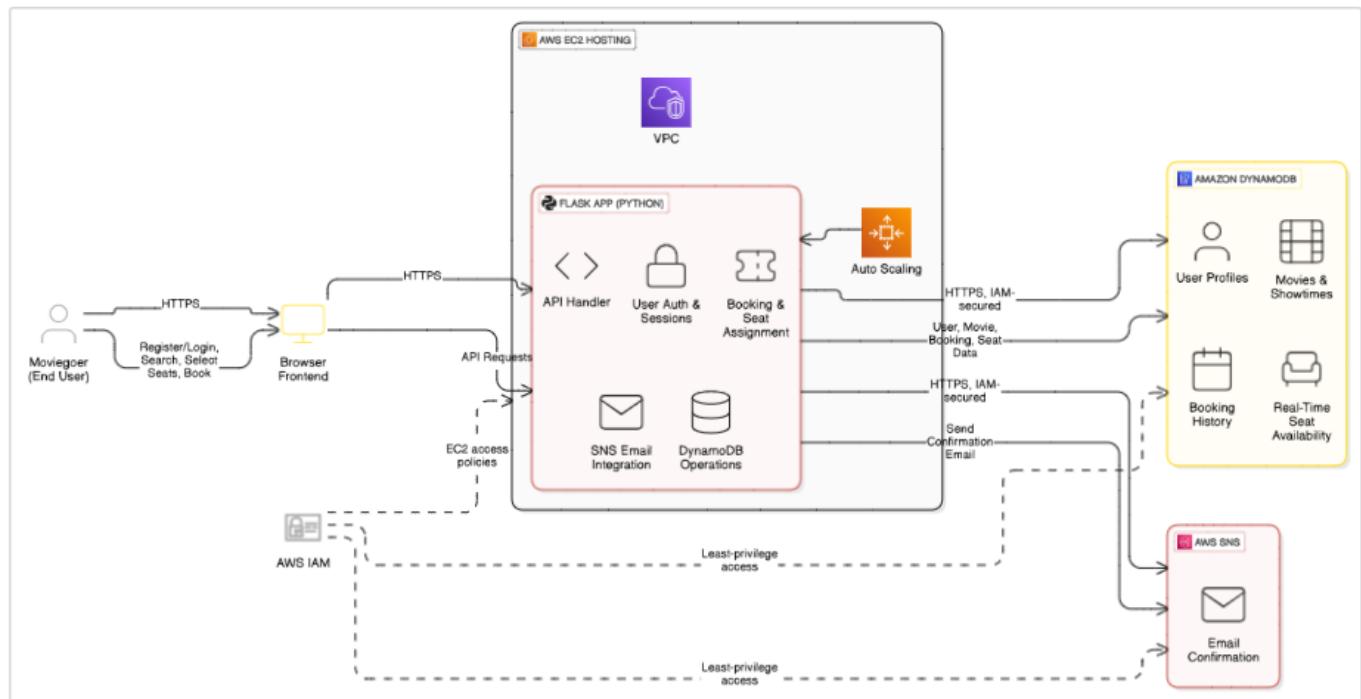
When a user completes a ticket booking, the Movie Magic System leverages AWS SNS to send instant email notifications to confirm the booking. For instance, once the booking is submitted, Flask processes the transaction, and SNS sends a customized email to the user with all ticket details, including movie name, date, time, and seat numbers. This real-time notification system enhances the customer experience and reduces uncertainty, while DynamoDB securely stores the booking records for both users and admins to manage and track.

Scenario 3: Easy Access to Movies and Events

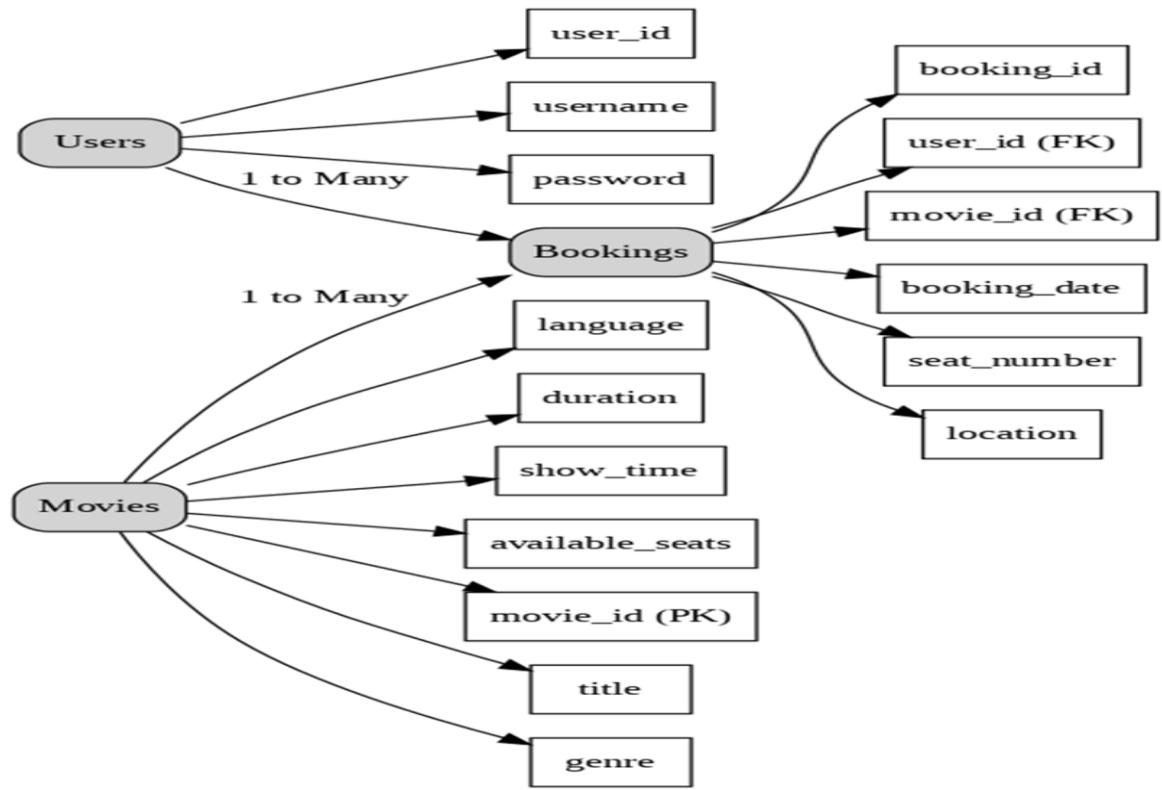
The Movie Magic platform offers users a seamless interface to explore currently running movies and upcoming live events. After logging in, a user can search by location or genre and instantly view listings with showtimes, ratings, and available seats. Flask dynamically fetches this data from DynamoDB, ensuring real-time updates on seat availability and event information. Meanwhile, the EC2-hosted application remains stable and responsive, even during traffic spikes, providing users with an uninterrupted and enjoyable booking experience.

AWS ARCHITECTURE

This AWS-based architecture powers a scalable and secure web application using Amazon EC2 for hosting the backend, with a lightweight framework like Flask handling core logic. Application data is stored in Amazon DynamoDB, ensuring fast, reliable access, while user access is managed through AWS IAM for secure authentication and control. Real-time alerts and system notifications are enabled via Amazon SNS, enhancing communication and user engagement.



Entity Relationship (ER)Diagram:



Pre-requisites

- AWS Account Setup :
<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
- AWS IAM (Identity and Access Management) :
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>
- AWS EC2 (Elastic Compute Cloud) :
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
- AWS DynamoDB :
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>
- Amazon SNS :
<https://docs.aws.amazon.com/sns/latest/dg/welcome.html>
- Git Documentation :
<https://git-scm.com/doc>
- VS Code Installation : (download the VS Code using the below link or you can get that in Microsoft store)
<https://code.visualstudio.com/download>

Project Flow

Milestone 1. Backend Development and Application Setup

- Develop the Backend Using Flask.
- Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

- Set up an AWS account if not already done.
- Log in to the AWS Management Console

Milestone 3. DynamoDB Database Creation and Setup

- Create a DynamoDB Table.
- Configure Attributes for User Data and Book Requests.

Milestone 4. SNS Notification Setup

- Create SNS topics for book request notifications.
- Subscribe users and library staff to SNS email notifications.

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

- Launch an EC2 instance to host the Flask application.
- Configure security groups for HTTP, and SSH access.

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

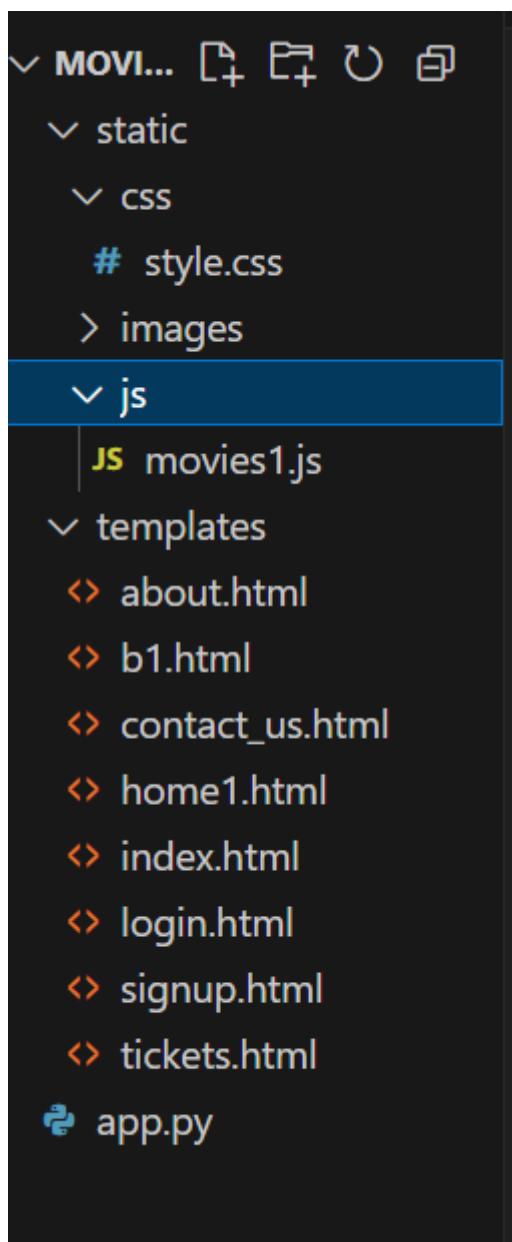
- Conduct functional testing to verify user registration, login, book requests, and notifications.
-

Milestone 1 : Web Application Development And Setup

Backend Development and Application Setup focuses on establishing the core structure of the application. This includes configuring the backend framework, setting up routing, and integrating database connectivity. It lays the groundwork for handling user interactions, data management, and secure access.

LOCAL DEPLOYMENT

- File Explorer Structure



Description of the code :

- **Flask App Initialization**

Imports and Configuration:

```
1  from flask import Flask, render_template, request, redirect, url_for, session, flash
2  from werkzeug.security import generate_password_hash, check_password_hash
3  from datetime import datetime
4  import boto3
5  import uuid
6  import json
7  import os
8  from botocore.exceptions import ClientError
```

Description: This project uses Flask for routing, session management, and user authentication with secure password hashing. It integrates AWS services via Boto3 for handling data storage, notifications, and unique user operations.

```
app = Flask(__name__)
```

Description: A new Flask application instance is initialized, and a secret key is set to securely manage user sessions and protect against cookie tampering.

- **Dynamodb and SNS Setup:**

```
11 # Use a static secret key
12 app.secret_key = 'your_static_secret_key_here' # Replace with your own secret string
13
14 # AWS Configuration - read from environment variables for security
15 AWS_REGION = os.environ.get('AWS_REGION', 'ap-south-1')
16
17 # Fix the SNS_TOPIC_ARN assignment - this was the main issue
18 # Instead of using os.environ.get with the ARN as the key, set it directly
19 SNS_TOPIC_ARN = 'arn:aws:sns:ap-south-1:605134430972:MovieTicketNotifications'
20
21 # Initialize AWS services with proper credentials handling
22 # On EC2, this will use the instance profile/role automatically
23 dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
24 sns = boto3.client('sns', region_name=AWS_REGION)
25
26 # DynamoDB tables
27 USERS_TABLE_NAME = os.environ.get('USERS_TABLE_NAME', 'MovieMagic_Users')
28 BOOKINGS_TABLE_NAME = os.environ.get('BOOKINGS_TABLE_NAME', 'MovieMagic_Bookings')
29
30 users_table = dynamodb.Table(USERS_TABLE_NAME)
31 bookings_table = dynamodb.Table(BOOKINGS_TABLE_NAME)
32
```

Description: Use **boto3** to connect to **DynamoDB** for handling user registration, movie bookings database operations and also mention region_name where Dynamodb tables are created.

- **SNS Connection**

- **Description:** Configure **SNS** to send notifications when a movie ticket is booked. Paste your stored ARN link in the sns_topic_arn space, along with the region_name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER_EMAIL section. Create an 'App password' for the email ID and store it in the SENDER_PASSWORD section.

- **Function to send the Notifications:**

Description: This function sends a booking confirmation email using AWS SNS. It formats the booking details into a message and publishes it to a specified SNS topic, notifying the user via email about their successful movie ticket booking.

```
def send_booking_confirmation(booking):
    """Send booking confirmation email using SNS"""
    # Check if SNS_TOPIC_ARN is set
    if not SNS_TOPIC_ARN:
        print("SNS_TOPIC_ARN is not set. Unable to send notification.")
        return False

    try:
        # Format email content
        email_subject = f"MovieMagic Booking Confirmation - {booking['booking_id']}"

        email_message = f"""
Hello {booking['user_name']},

Your movie ticket booking is confirmed!

Booking Details:
-----
Booking ID: {booking['booking_id']}
Movie: {booking['movie_name']}
Date: {booking['date']}
Time: {booking['time']}
Theater: {booking['theater']}
Location: {booking['address']}
Seats: {booking['seats']}
Amount Paid: ₹{booking['amount_paid']}

Please show this confirmation at the theater to collect your tickets.
    
```

```

Thank you for choosing MovieMagic!
"""

# User email
user_email = booking['booked_by']

print(f"Attempting to send notification to {user_email} via SNS topic {SNS_TOPIC_ARN}")

# Send directly to the email using SNS
response = sns.publish(
    TopicArn=SNS_TOPIC_ARN,
    Subject=email_subject,
    Message=email_message,
    MessageAttributes={
        'email': {
            'DataType': 'String',
            'StringValue': user_email
        }
    }
)

print(f"SNS publish response: {response}")
print(f"Booking confirmation sent to {user_email}")
return True

except Exception as e:
    print(f"Error sending booking confirmation: {str(e)}")
    return False

```

- **Routes for Web Pages**
- **Register User:** Collecting registration data, hashes the password, and stores user details in the database.

```

1 @app.route('/signup', methods=['GET', 'POST'])
2 def signup():
3     if request.method == 'POST':
4         name = request.form['name']
5         email = request.form['email']
6         password = generate_password_hash(request.form['password'])
7
8         try:
9             # Check if user already exists
10            response = users_table.get_item(Key={'email': email})
11            if 'Item' in response:
12                flash('Email already registered!', 'danger')
13                return redirect(url_for('signup'))
14
15            # Create new user in DynamoDB
16            user_id = str(uuid.uuid4())
17            users_table.put_item(
18                Item={
19                    'id': user_id,
20                    'name': name,
21                    'email': email,
22                    'password': password,
23                    'created_at': datetime.now().isoformat()
24                }
25            )
26
27            flash('Registration successful! Please login.', 'success')
28            return redirect(url_for('login'))
29
30        except ClientError as e:
31            print(f"Error accessing DynamoDB: {e.response['Error']['Message']}")
32            flash('An error occurred during registration. Please try again.', 'danger')
33
34        return render_template('signup.html')
35
36
37
38
39
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
249
249
250
251
252
253
254
255
255
256
257
257
258
259
259
260
261
262
263
264
264
265
266
266
267
268
268
269
269
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
141
```

```

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

    try:
        # Get user from DynamoDB
        response = users_table.get_item(Key={'email': email})

        if 'Item' in response:
            user = response['Item']
            if check_password_hash(user['password'], password):
                session['user'] = {
                    'id': user['id'],
                    'name': user['name'],
                    'email': user['email']
                }
                return redirect(url_for('home1'))

            flash('Invalid email or password', 'danger')
        except ClientError as e:
            print(f"Error accessing DynamoDB: {e.response['Error']['Message']} ")
            flash('An error occurred. Please try again later.', 'danger')

    return render_template('login.html')

```

- These Flask routes handle key navigation in the app: /logout logs out the user by clearing the session and showing a flash message; /home1 is a protected route accessible only to logged-in users; /about and /contact_us render static pages with information about the app and ways to get in touch.

```
0  @app.route('/logout')
1  def logout():
2      session.pop('user', None)
3      flash('You have been logged out!', 'info')
4      return redirect(url_for('index'))
5
6  # Application Routes
7  @app.route('/home1')
8  def home1():
9      if 'user' not in session:
10          return redirect(url_for('login'))
11      return render_template('home1.html')
12
13  @app.route('/about')
14  def about():
15      return render_template('about.html')
16
17  @app.route('/contact_us')
18  def contact():
19      return render_template('contact_us.html')
20
```

- **Booking Page Route:**

Description: This route displays the booking page (b1.html) with movie, theater, address, and price details passed as query parameters. It ensures only logged-in users can access the page.

```
# Booking page route
@app.route('/b1', methods=['GET'], endpoint='b1') # Add explicit endpoint
def booking_page():
    if 'user' not in session:
        return redirect(url_for('login'))

    return render_template('b1.html',
        movie=request.args.get('movie'),
        theater=request.args.get('theater'),
        address=request.args.get('address'),
        price=request.args.get('price')
    )
```

- **Tickets Page Route:**

Description: This route processes movie ticket bookings by collecting form data, generating a unique booking ID, storing details in DynamoDB, and sending a confirmation email via AWS SNS. It then displays the booking details on the tickets page.

```
@app.route('/tickets', methods=['POST'])
def tickets():
    if 'user' not in session:
        return redirect(url_for('login'))

    try:
        # Extract booking details from form
        movie_name = request.form.get('movie')
        booking_date = request.form.get('date')
        show_time = request.form.get('time')
        theater_name = request.form.get('theater')
        theater_address = request.form.get('address')
        selected_seats = request.form.get('seats') # Changed from selected_seats
        amount_paid = request.form.get('amount') # Changed from total_price

        # Generate a unique booking ID
        booking_id = f'MVM-{datetime.now().strftime("%Y%m%d")}-{str(uuid.uuid4())[:8]}'

        # Store booking in DynamoDB
        booking_item = {
            'booking_id': booking_id,
            'movie_name': movie_name,
            'date': booking_date,
            'time': show_time,
            'theater': theater_name,
            'address': theater_address,
            'booked_by': session['user']['email'],
            'user_name': session['user']['name'],
            'seats': selected_seats,
```

```

        'user_name': session['user']['name'],
        'seats': selected_seats,
        'amount_paid': amount_paid,
        'booking_time': datetime.now().isoformat()
    }

bookings_table.put_item(Item=booking_item)

# Send email notification via SNS
notification_sent = send_booking_confirmation(booking_item)
if notification_sent:
    flash('Booking confirmation has been sent to your email!', 'success')

# Pass the booking details to the tickets template
return render_template('tickets.html', booking=booking_item)

except Exception as e:
    print(f"Error processing booking: {str(e)}")
    flash('Error processing booking', 'danger')
    return redirect(url_for('home1'))

```

Application Entry point:

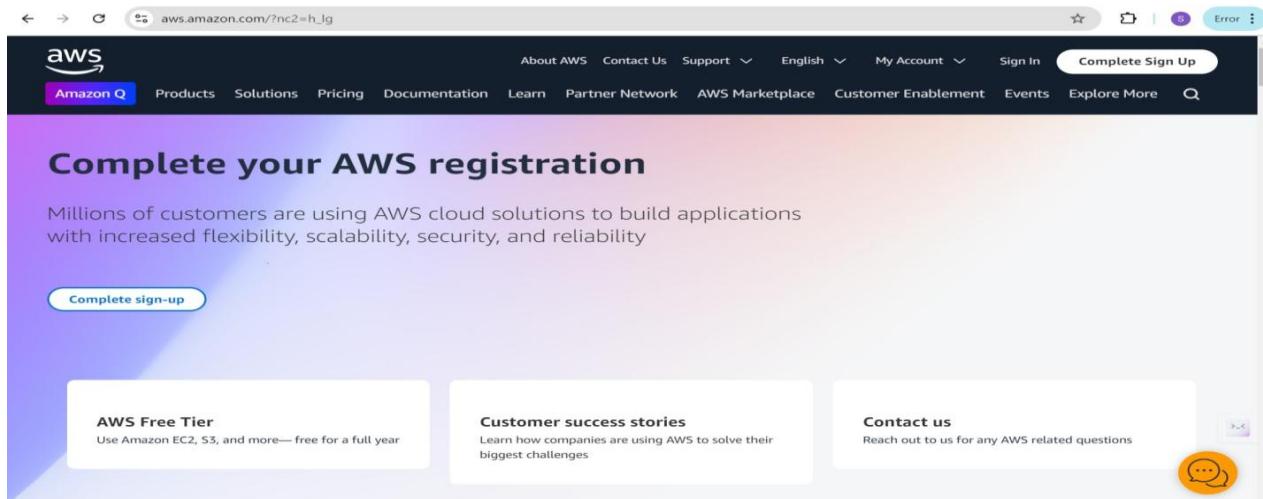
```

if __name__ == '__main__':
    # Using Flask's built-in server as requested
    port = int(os.environ.get('PORT', 5000))
    # You can set debug=False in production
    app.run(host='0.0.0.0', port=port, debug=True)

```

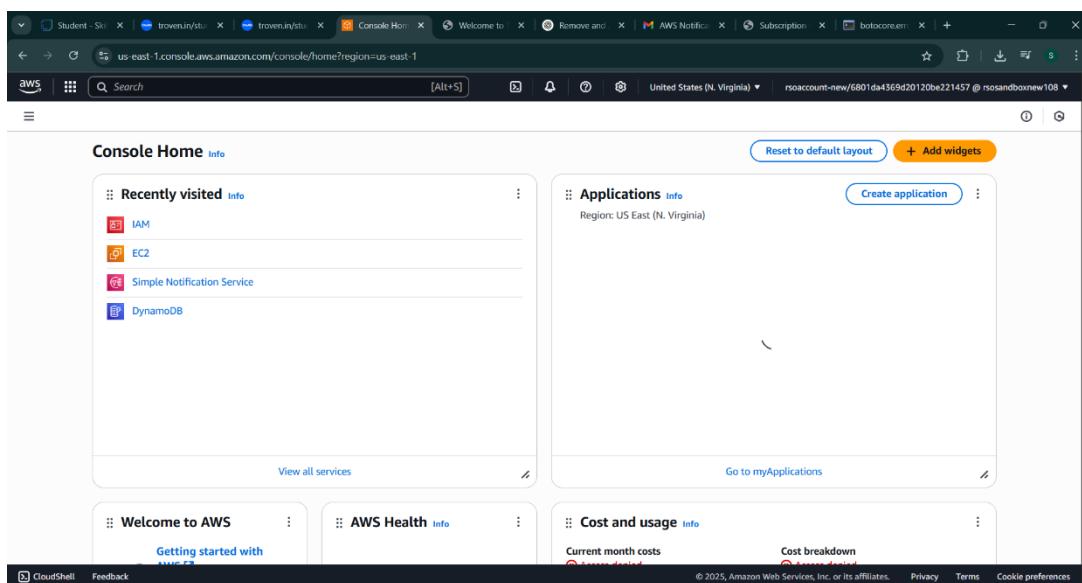
- **Description:** This block starts the Flask application using the built-in development server, setting the host, port, and enabling debug mode for easier development and testing.

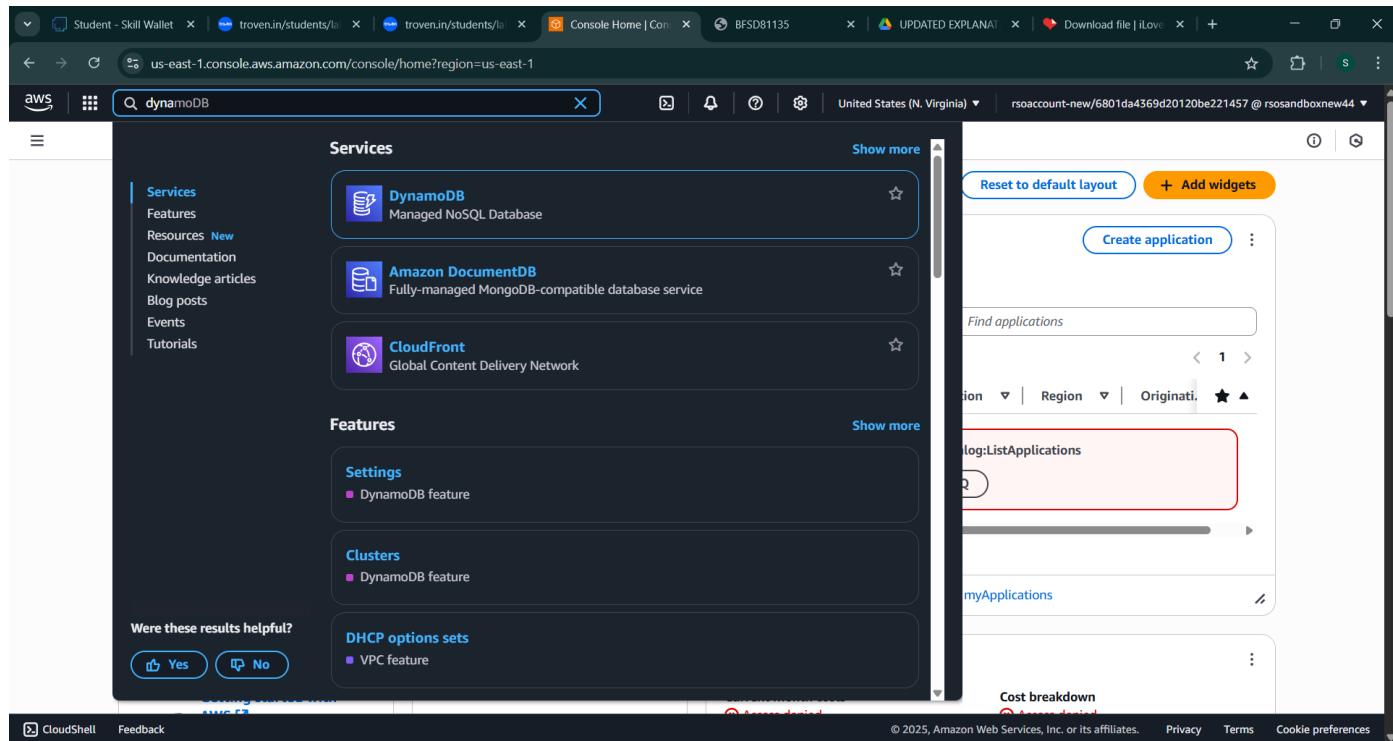
Milestone 2 : AWS Account Setup



- **Activity 2.2: Log in to the AWS Management Console**

- After setting up your account, log in to the [AWS Management Console](#).



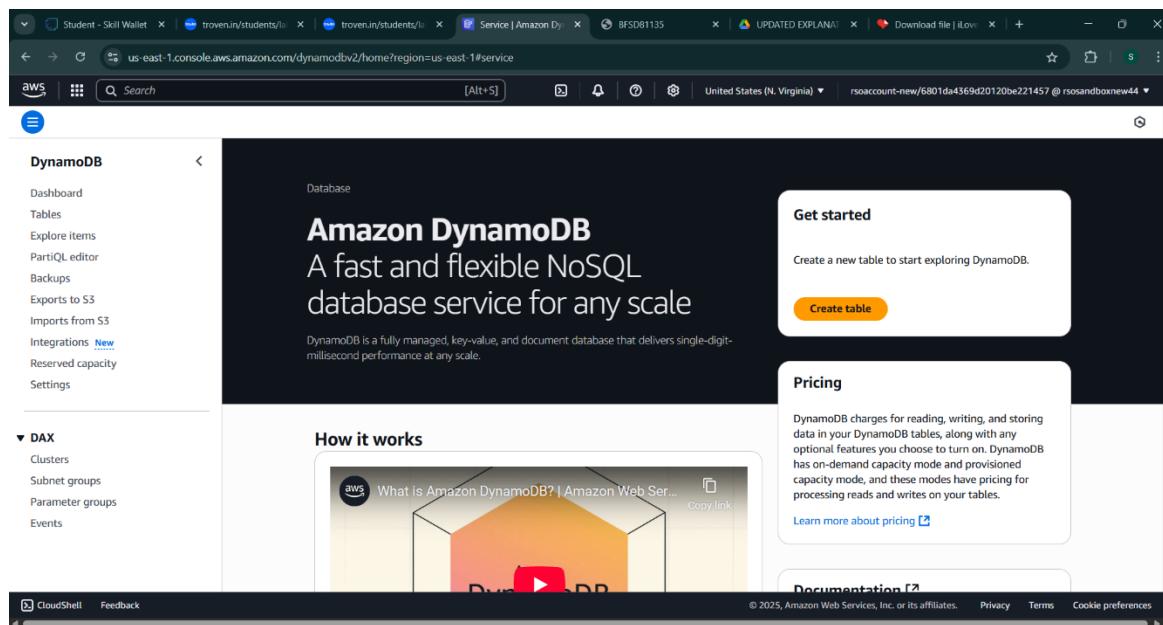


The screenshot shows the AWS search results for 'dynamoDB'. The left sidebar has links for Services, Features, Resources, Documentation, Knowledge articles, Blog posts, Events, and Tutorials. The main search results section has two tabs: 'Services' and 'Features'. Under 'Services', there are cards for DynamoDB (Managed NoSQL Database), Amazon DocumentDB (Fully-managed MongoDB-compatible database service), and CloudFront (Global Content Delivery Network). Under 'Features', there are cards for Settings (DynamoDB feature), Clusters (DynamoDB feature), and DHCP options sets (VPC feature). At the bottom, there are 'Were these results helpful?' buttons ('Yes' and 'No'). On the right, there is a sidebar with 'Create application' and 'Find applications' sections, and a 'Cost breakdown' section.

Milestone 3: DynamoDB Database Creation and Setup

- **Activity 3.1: Navigate to the DynamoDB**

- In the AWS Console, navigate to DynamoDB and click on create tables.



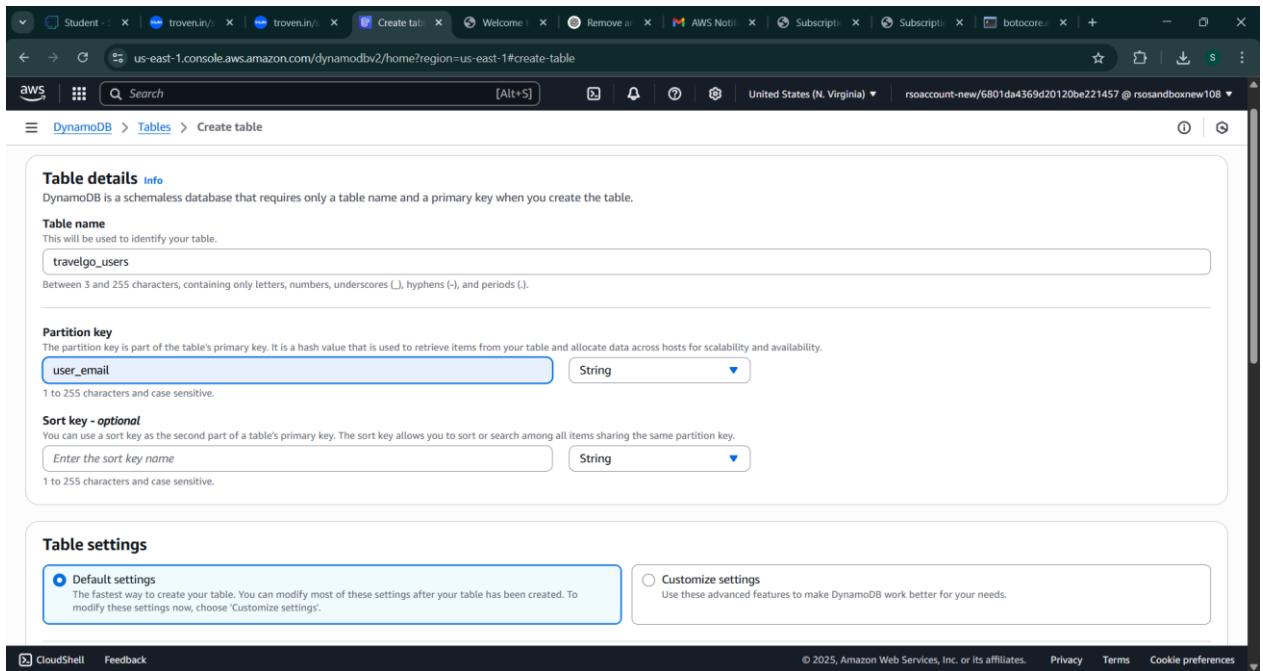
The screenshot shows the Amazon DynamoDB service page. The left sidebar has links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and settings for DAX. The main content area has a 'Database' header and a large section titled 'Amazon DynamoDB' with the subtext 'A fast and flexible NoSQL database service for any scale'. It explains that DynamoDB is a fully managed, key-value, and document database that delivers single-digit millisecond performance at any scale. There are 'Get started' and 'Create table' buttons. Below this, there is a 'How it works' section with a video thumbnail and a 'Pricing' section explaining charges for reading, writing, and storing data. At the bottom, there are 'Documentation' and 'Feedback' links.

2.
3.
○

- **Activity 3.2:Create a DynamoDB table for storing registration details and book requests.**

- Create Users table with partition key “user_email” with type String and click on create tables.

4.



The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. In the 'Table details' section, the table name is set to 'travelgo_users'. The 'Partition key' is defined as 'user_email' of type 'String'. In the 'Table settings' section, the 'Default settings' radio button is selected, indicating it's the fastest way to create the table. The 'Customize settings' option is also available for advanced features.



Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)

[Create table](#)

- Follow the same steps to create a requests table with `user_email` as the primary key for book requests data.

us-east-1.console.aws.amazon.com/dynamodbv2/home#create-table

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
 This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
 The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 String
1 to 255 characters and case sensitive.

Sort key - optional
 You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 String
1 to 255 characters and case sensitive.

Table settings

Default settings
 The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

Customize settings
 Use these advanced features to make DynamoDB work better for your needs.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

us-east-1.console.aws.amazon.com/dynamodbv2/home#create-table

Create table

Table details Info
 DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
 This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
 The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 String
1 to 255 characters and case sensitive.

Sort key - optional
 You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 String
1 to 255 characters and case sensitive.

Table settings

Default settings
 The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

Customize settings
 Use these advanced features to make DynamoDB work better for your needs.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS DynamoDB 'Create table' wizard.

Table Configuration:

Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	AWS owned key	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags:
 Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.
 No tags are associated with the resource.
[Add new tag](#)
 You can add 50 more tags.

Note: This table will be created with auto scaling deactivated. You do not have permissions to turn on auto scaling.

[Cancel](#) [Create table](#)

Milestone 4: SNS Notification Setup

● Activity 4.1: Create SNS topics for sending email notifications to users

Screenshot of the AWS Simple Notification Service (SNS) console.

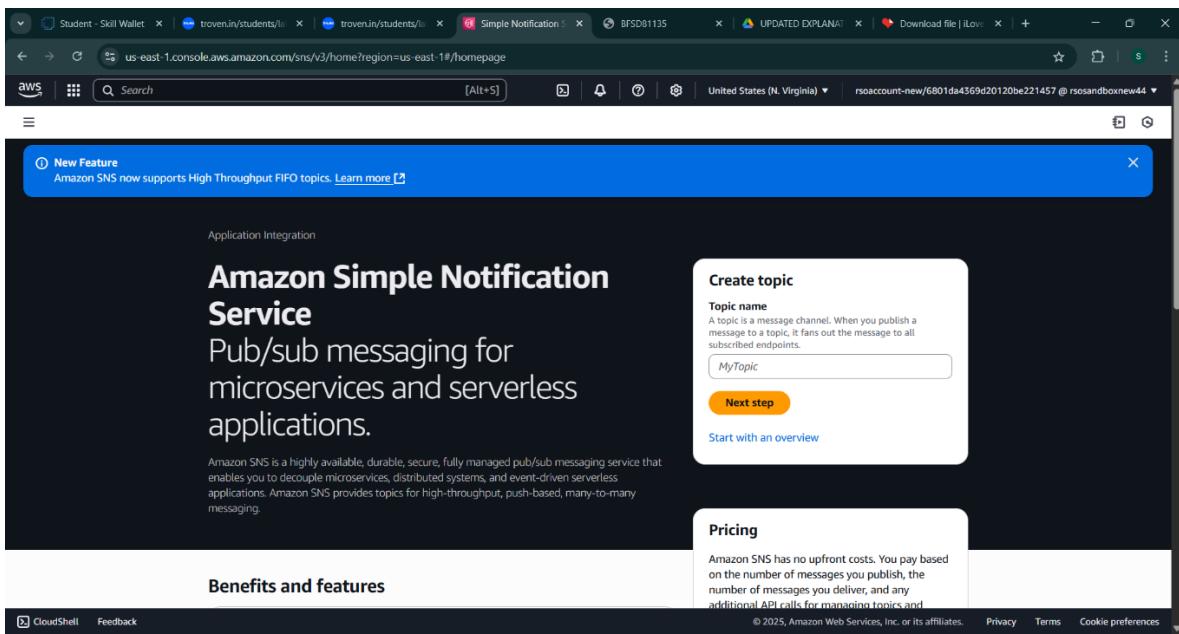
The left sidebar shows the navigation menu for SNS, including Services, Features, Resources, Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials.

The main content area displays the following information:

- Services:**
 - Simple Notification Service**: SNS managed message topics for Pub/Sub.
 - Route 53 Resolver**: Resolve DNS queries in your Amazon VPC and on-premises network.
 - Route 53**: Scalable DNS and Domain Name Registration.
- Features:**
 - Events**: ElastiCache feature.
 - SMS**: AWS End User Messaging feature.
 - Hosted zones**: Route 53 feature.

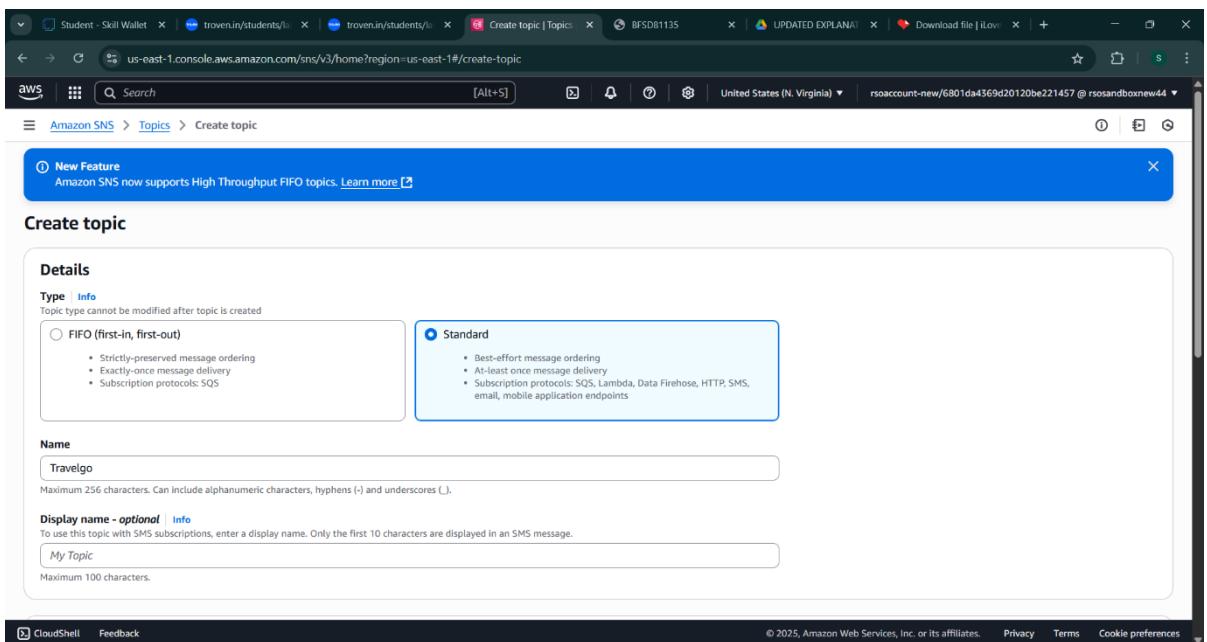
A modal window titled 'Create table' is open on the right side of the screen, showing fields for Region, Name, and Description. It also includes tabs for 'Actions', 'Delete', and 'Create table'.

- In the AWS Console, search for SNS and navigate to the SNS Dashboard.



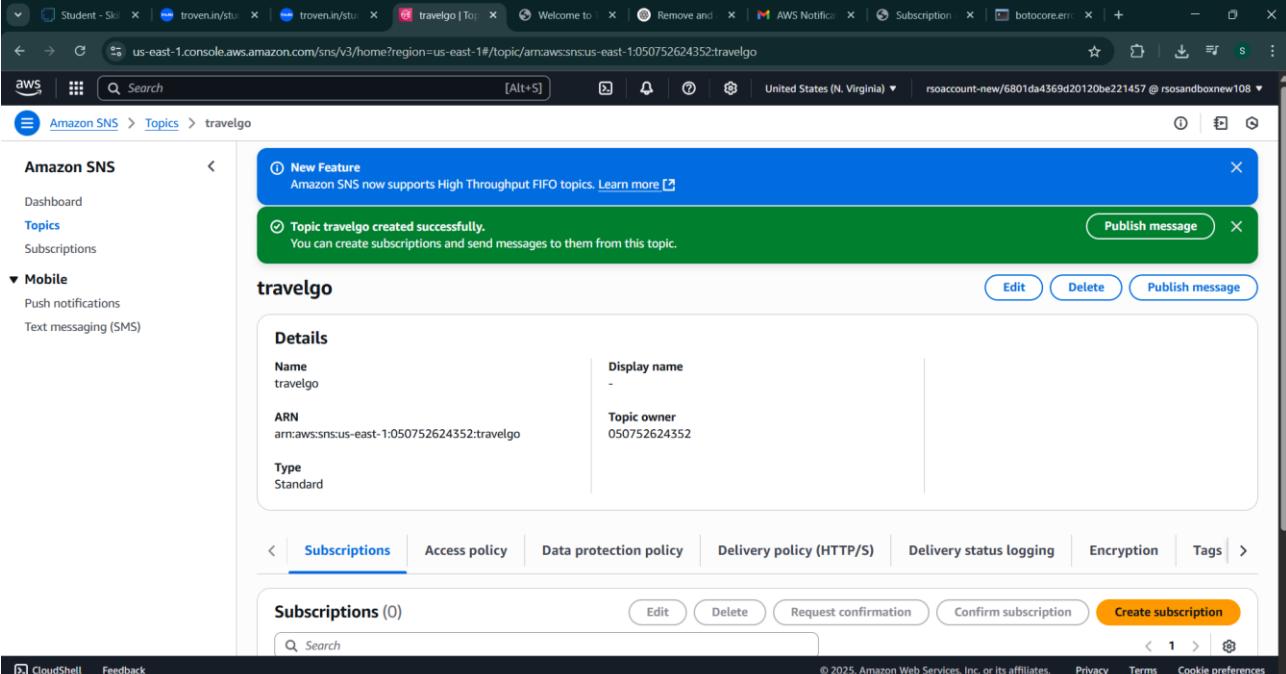
The screenshot shows the AWS SNS dashboard. At the top, there is a blue banner with the text "Amazon SNS now supports High Throughput FIFO topics. Learn more". Below the banner, the main heading is "Amazon Simple Notification Service" with the subtitle "Pub/sub messaging for microservices and serverless applications". To the right, there is a "Create topic" dialog box. Inside the dialog, there is a "Topic name" input field containing "MyTopic", a "Next step" button, and a "Start with an overview" link. Below the dialog, there is a "Pricing" section with a note about no upfront costs based on message volume. At the bottom of the page, there are links for "CloudShell", "Feedback", and copyright information.

- Click on **Create Topic** and choose a name for the topic.



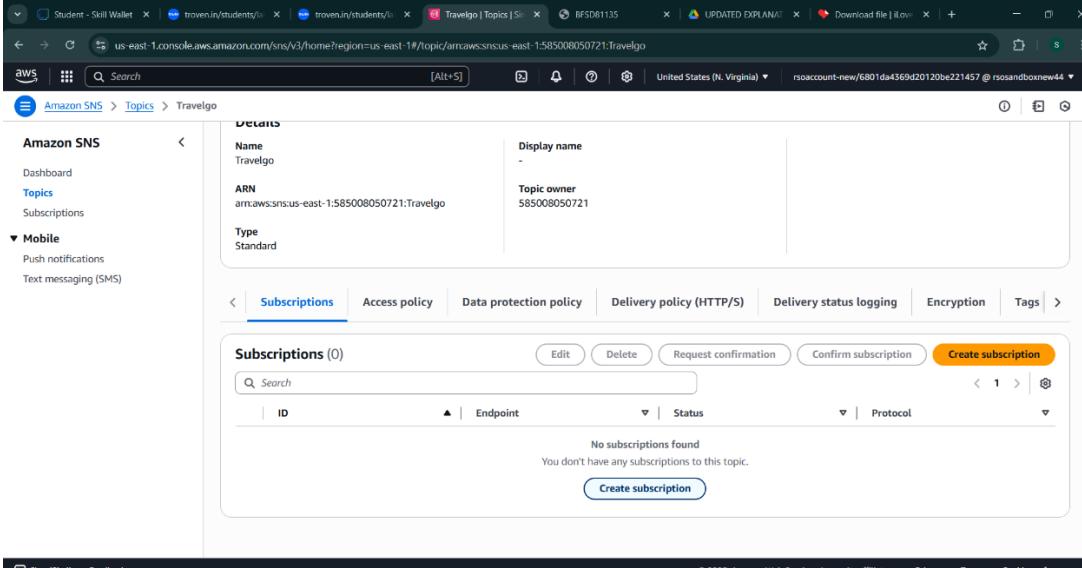
The screenshot shows the "Create topic" configuration page. It has two main sections: "Details" and "Standard". Under "Details", there is a "Type" dropdown set to "Info" and a note that the topic type cannot be modified after creation. There are two options: "FIFO (first-in, first-out)" and "Standard". The "Standard" option is selected and includes a bulleted list of features: "Best-effort message ordering", "At-least once message delivery", and "Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints". Below the "Details" section, there is a "Name" input field with "Travelgo" entered, a note about character limits, and a "Display name - optional" input field with "My Topic" entered, also with a character limit note. At the bottom, there are "CloudShell", "Feedback", and copyright information.

- Choose Standard type for general notification use cases and Click on Create Topic.



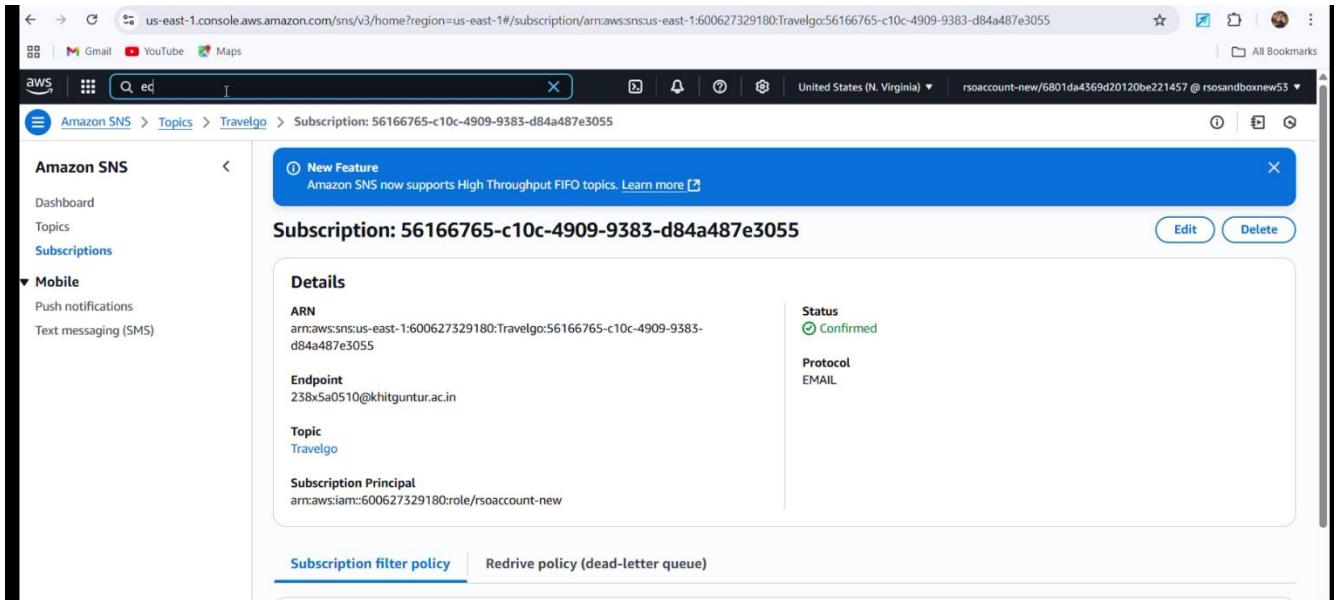
The screenshot shows the AWS SNS Topics page. A success message indicates that the topic 'travelgo' was created successfully. The ARN is listed as arn:aws:sns:us-east-1:050752624352:travelgo. The Subscriptions tab shows 0 subscriptions.

- Configure the SNS topic and note down the **Topic ARN**.
- **Activity 4.2: Subscribe users relevant SNS topics to receive real-time notifications when a book request is made.**



The screenshot shows the AWS SNS Topics page for the 'travelgo' topic. The ARN is listed as arn:aws:sns:us-east-1:585008050721:Travelgo. The Subscriptions tab shows 0 subscriptions.

- Subscribe users to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails.

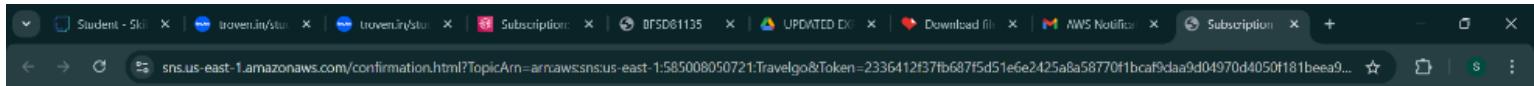


The screenshot shows the AWS SNS console with a subscription details page. The URL in the browser is `us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/subscription/arm:aws:sns:us-east-1:600627329180:Travelgo:56166765-c10c-4909-9383-d84a487e3055`. The page title is "Subscription: 56166765-c10c-4909-9383-d84a487e3055". The left sidebar shows "Amazon SNS" with "Subscriptions" selected. The main content area displays the following subscription details:

Details	Status
ARN arn:aws:sns:us-east-1:600627329180:Travelgo:56166765-c10c-4909-9383-d84a487e3055	Confirmed
Endpoint 238x5a0510@khitguntur.ac.in	Protocol EMAIL
Topic Travelgo	
Subscription Principal arn:aws:iam::600627329180:role/rsoaccount-new	

At the bottom, there are tabs for "Subscription filter policy" (which is selected) and "Redrive policy (dead-letter queue)".

After subscription request for the mail confirmation



Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
arn:aws:sns:us-east-1:585008050721:Travelgo:fcbac1ec-8c7a-401f-8fff-4457a1855207

If it was not your intention to subscribe, [click here to unsubscribe](#).

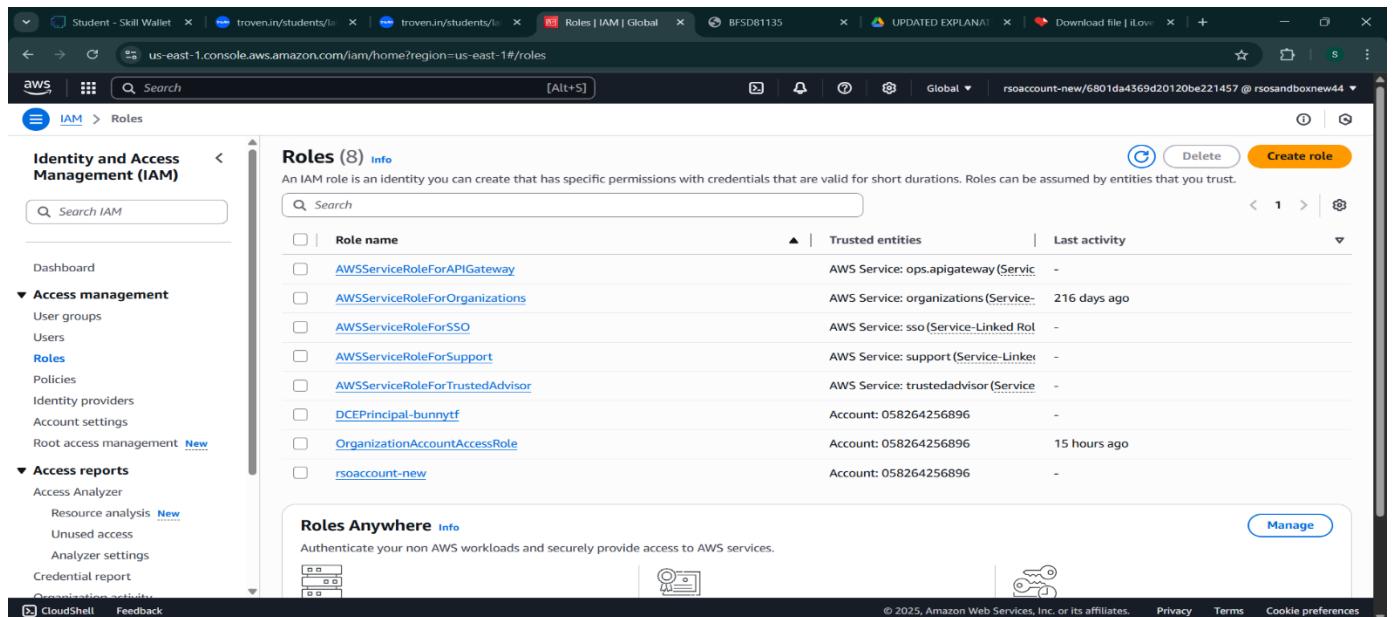
Successfully done with the SNS mail subscription and setup, now store the ARN link

Milestone 5: IAM Role Setup

- **Activity 5.1: Create IAM Role.**

- In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.

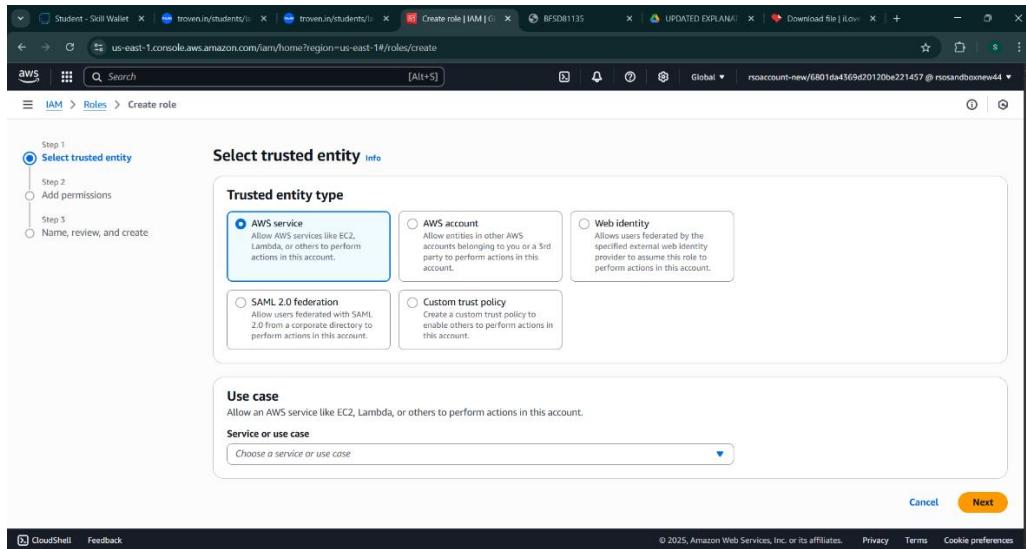
-



The screenshot shows the AWS IAM Roles page with 8 roles listed:

Role name	Trusted entities	Last activity
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service)	-
AWSServiceRoleForOrganizations	AWS Service: organizations (Service)	216 days ago
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
DCEPrincipal-bunnytf	Account: 058264256896	-
OrganizationAccountAccessRole	Account: 058264256896	15 hours ago
rsoaccount-new	Account: 058264256896	-

Below the table, there is a section titled "Roles Anywhere" with a "Manage" button.



Step 1
 Select trusted entity Info

Step 2
 Add permissions

Step 3
 Name, review, and create

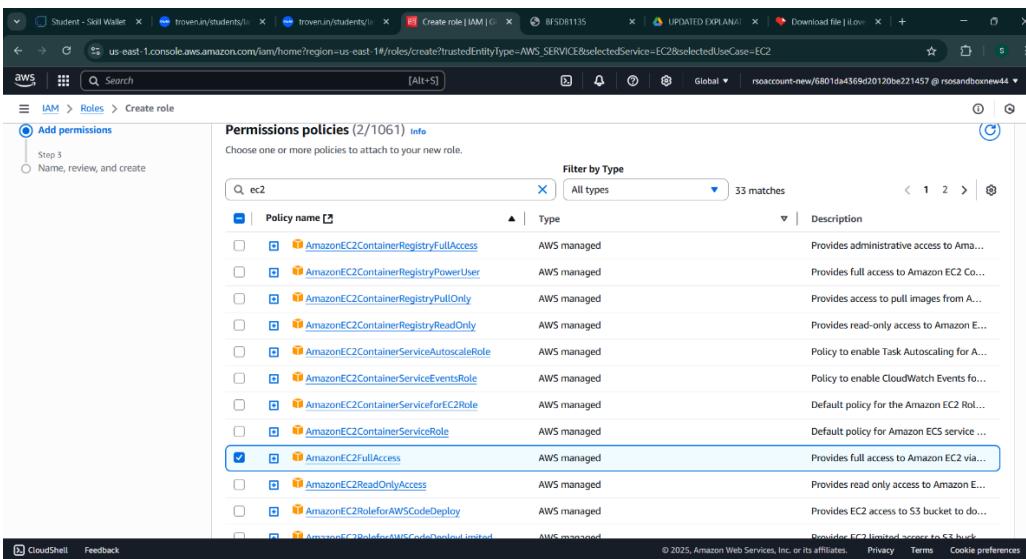
Select trusted entity Info

Trusted entity type

- AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**
Allow users authenticated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case



Step 1
 Select trusted entity Info

Step 2
 Add permissions

Step 3
 Name, review, and create

Permissions policies (2/1061) Info

Choose one or more policies to attach to your new role.

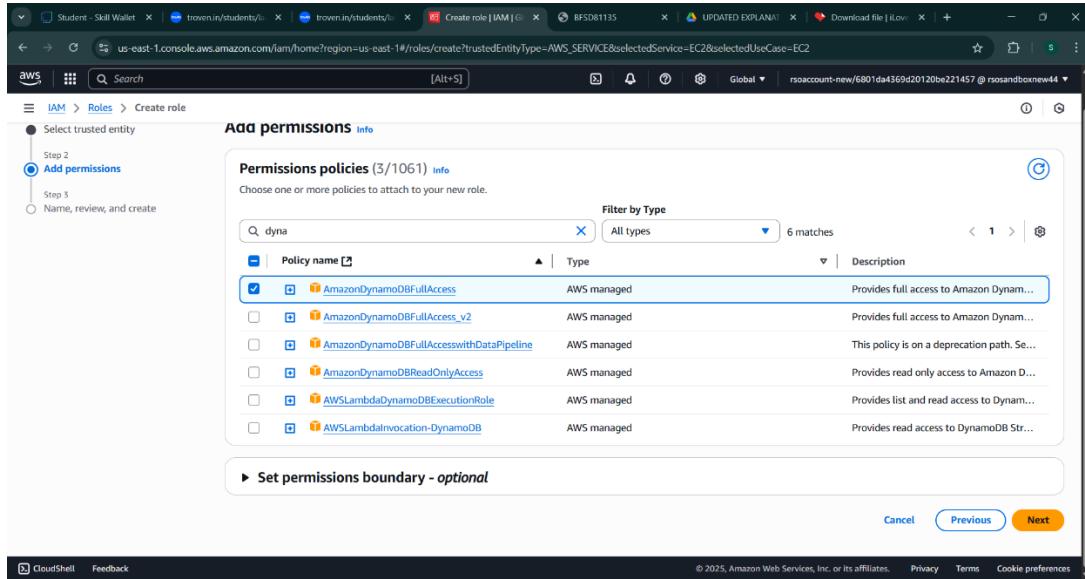
Filter by Type

Policy name	Type	Description
AmazonEC2ContainerRegistryFullAccess	AWS managed	Provides administrative access to Amazon E...
AmazonEC2ContainerRegistryPowerUser	AWS managed	Provides full access to Amazon EC2 Co...
AmazonEC2ContainerRegistryPullOnly	AWS managed	Provides access to pull images from A...
AmazonEC2ContainerRegistryReadOnly	AWS managed	Provides read-only access to Amazon E...
AmazonEC2ContainerServiceAutoscaleRole	AWS managed	Policy to enable Task AutoScaling for A...
AmazonEC2ContainerServiceEventsRole	AWS managed	Policy to enable CloudWatch Events fo...
AmazonEC2ContainerServiceforEC2Role	AWS managed	Default policy for the Amazon EC2 Rol...
AmazonEC2ContainerServiceRole	AWS managed	Default policy for Amazon ECS service ...
AmazonEC2FullAccess	AWS managed	Provides full access to Amazon EC2 via...
AmazonEC2ReadonlyAccess	AWS managed	Provides read only access to Amazon E...
AmazonEC2RoleforAWSCodeDeploy	AWS managed	Provides EC2 access to S3 bucket to do...
AmazonEC2RoleforAWSCodeDeployWithLambda	AWS managed	Provides EC2 limited access to S3 buck...

● Activity 5.2: Attach Policies.

Attach the following policies to the role:

- **AmazonDynamoDBFullAccess:** Allows EC2 to perform read/write operations on DynamoDB.
- **AmazonSNSFullAccess:** Grants EC2 the ability to send notifications via SNS.



aws | Search [Alt+S]

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 **Add permissions**

Step 3 Name, review, and create

Add permissions Info

Permissions policies (3/1061) Info

Choose one or more policies to attach to your new role.

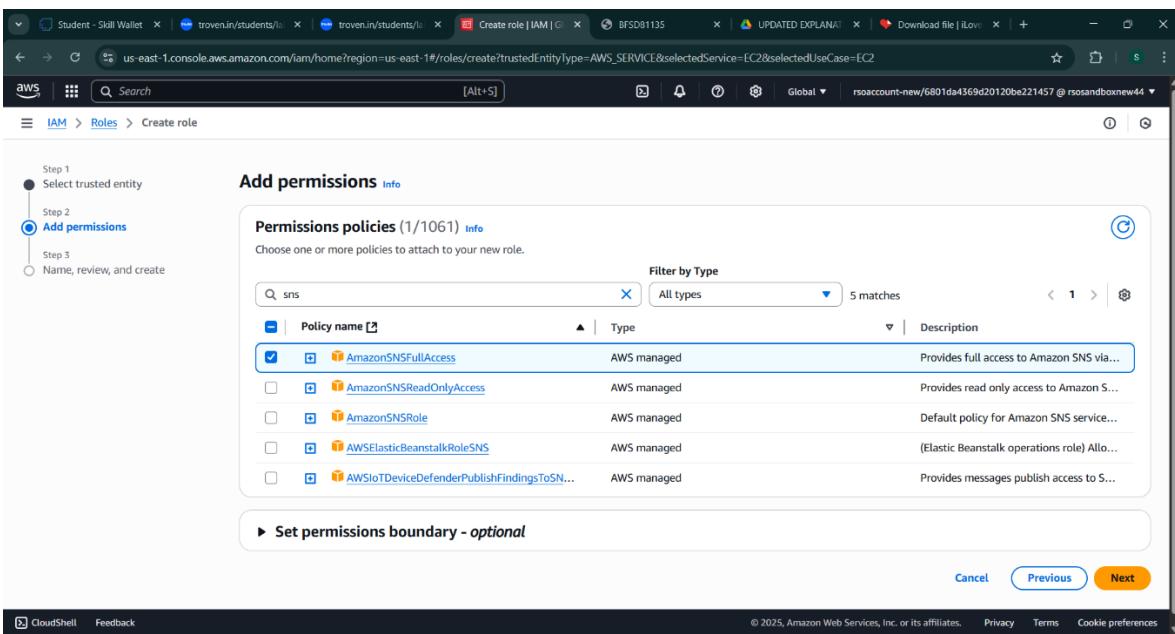
Filter by Type All types 6 matches

Policy name	Type	Description
<input checked="" type="checkbox"/>  AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB.
<input type="checkbox"/>  AmazonDynamoDBFullAccess_v2	AWS managed	Provides full access to Amazon DynamoDB.
<input type="checkbox"/>  AmazonDynamoDBFullAccesswithDataPipeline	AWS managed	This policy is on a deprecation path. See AmazonDynamoDBFullAccess .
<input type="checkbox"/>  AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon DynamoDB.
<input type="checkbox"/>  AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to DynamoDB.
<input type="checkbox"/>  AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Stream.

▶ Set permissions boundary - optional

Cancel Previous Next

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



aws | Search [Alt+S]

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 **Add permissions**

Step 3 Name, review, and create

Add permissions Info

Permissions policies (1/1061) Info

Choose one or more policies to attach to your new role.

Filter by Type All types 5 matches

Policy name	Type	Description
<input checked="" type="checkbox"/>  AmazonSNSFullAccess	AWS managed	Provides full access to Amazon SNS via the AWS Lambda permission model.
<input type="checkbox"/>  AmazonSNSReadOnlyAccess	AWS managed	Provides read only access to Amazon SNS.
<input type="checkbox"/>  AmazonSNSRole	AWS managed	Default policy for Amazon SNS service role.
<input type="checkbox"/>  AWSElasticBeanstalkRoleSNS	AWS managed	(Elastic Beanstalk operations role) Allows the Elastic Beanstalk service to publish messages to SNS.
<input type="checkbox"/>  AWSIoTDeviceDefenderPublishFindingsToSN...	AWS managed	Provides messages publish access to SNS.

▶ Set permissions boundary - optional

Cancel Previous Next

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Milestone 6: EC2 Instance Setup

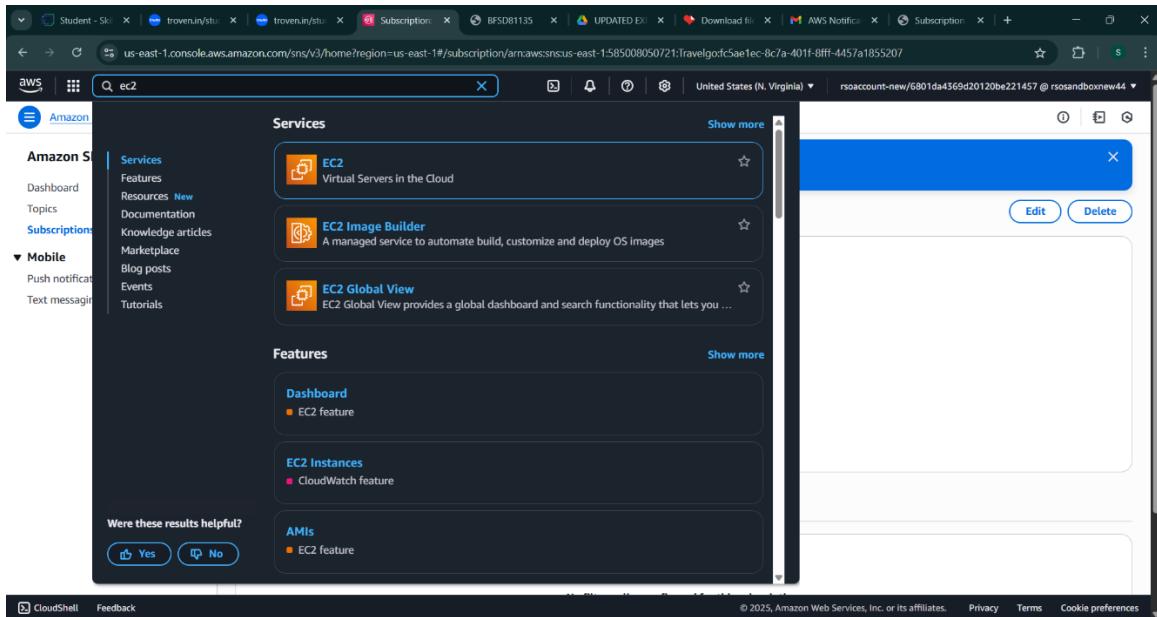
- Note: Load your Flask app and Html files into GitHub repository.

 static	Added full project files
 templates	app.py changed
 venv	Added full project files
 venv_new	Added full project files
 README.md	first commit
 app.py	Update app.py

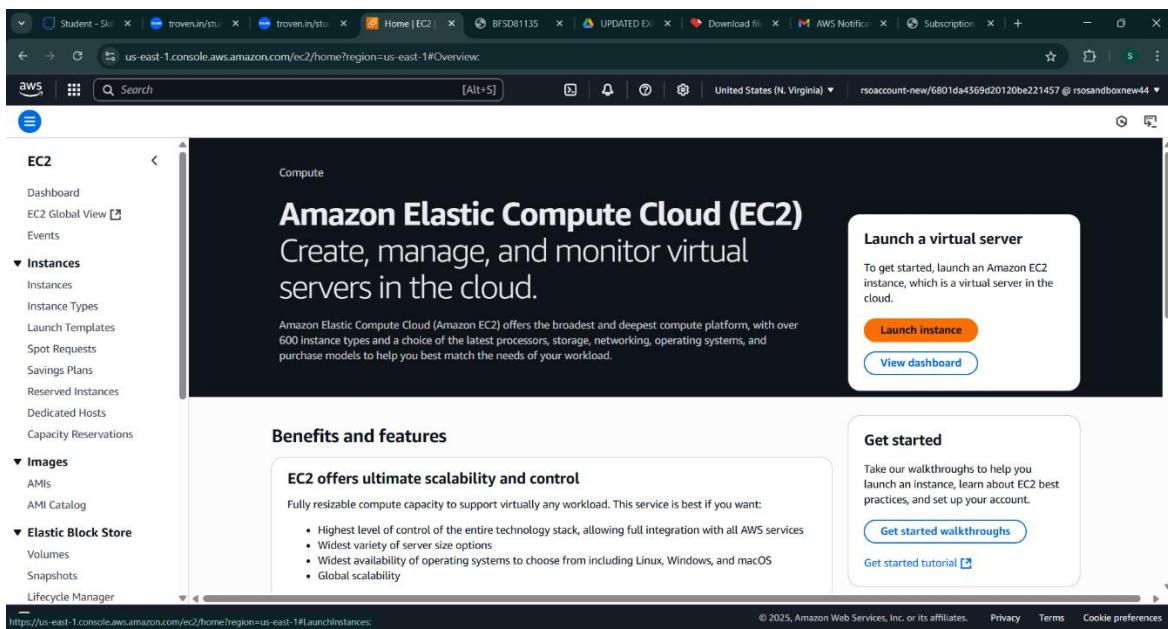
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **Launch EC2 Instance**

- In the AWS Console, navigate to EC2 and launch a new instance.



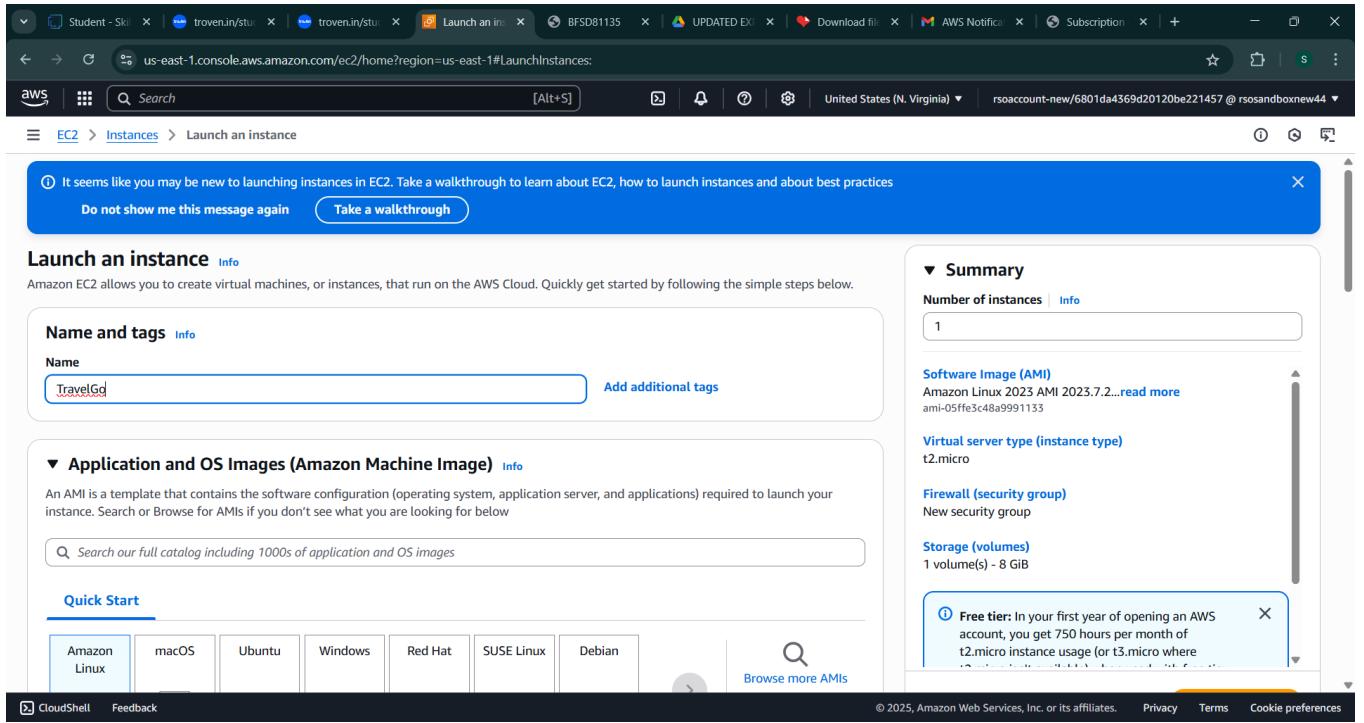
The screenshot shows the AWS search results for the query "ec2". The results are displayed in two main sections: Services and Features. The Services section includes cards for EC2, EC2 Image Builder, and EC2 Global View. The Features section includes cards for Dashboard, EC2 Instances, and AMIs. A modal window titled "EC2" is open on the right side of the screen, showing options to "Edit" or "Delete". The URL in the browser is <https://us-east-1.console.aws.amazon.com/sns/v3/home/?region=us-east-1#/subscription/armaws:ssncus-east-1:585008050721:travelgo:fSae1ec-8c7a-40f1-8ff-4457a1855207>.



The screenshot shows the AWS EC2 Overview page. The left sidebar contains navigation links for EC2, Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. The main content area features a large heading "Amazon Elastic Compute Cloud (EC2)" with the subtext "Create, manage, and monitor virtual servers in the cloud." Below this, there is a "Benefits and features" section with a box titled "EC2 offers ultimate scalability and control". This box lists the following benefits: "Fully resizable compute capacity to support virtually any workload. This service is best if you want:" followed by a bulleted list: "Highest level of control of the entire technology stack, allowing full integration with all AWS services", "Widest variety of server size options", "Widest availability of operating systems to choose from including Linux, Windows, and macOS", and "Global scalability". To the right of this section is a "Get started" box containing "Launch a virtual server" and "Launch instance" and "View dashboard" buttons. The URL in the browser is <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances>.

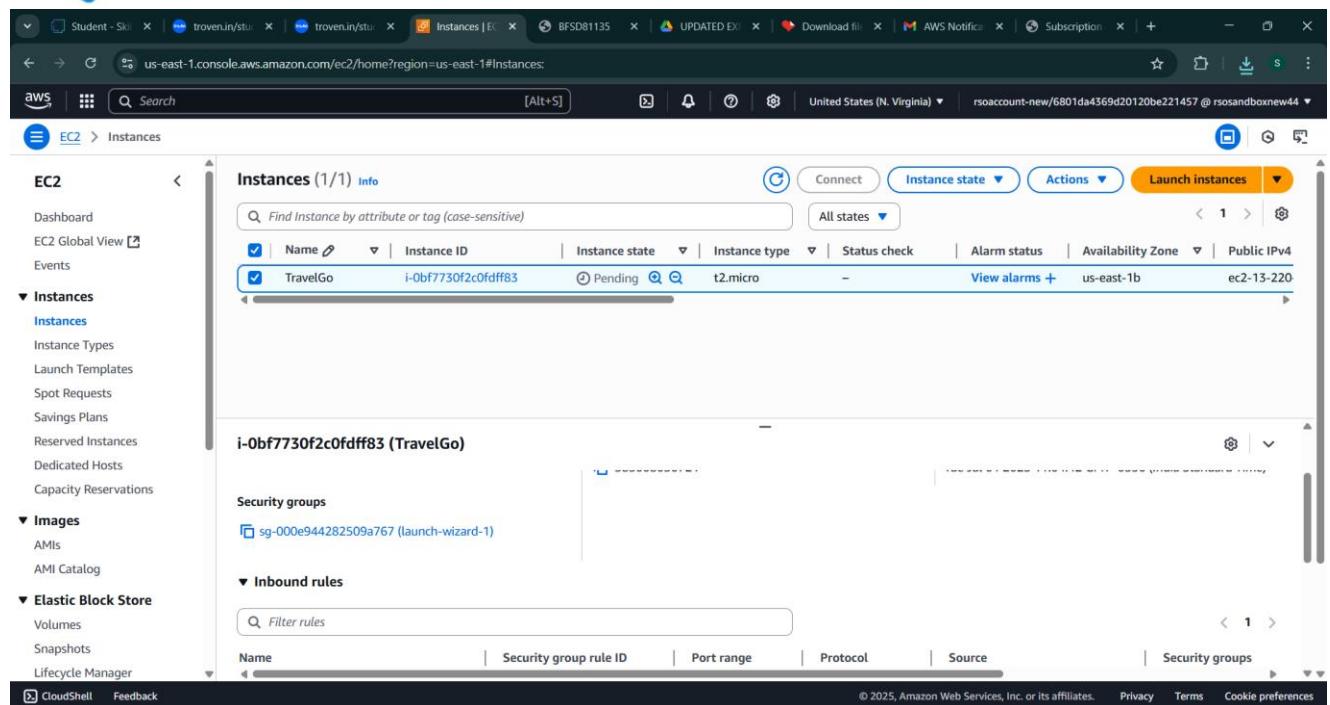
- Click on Launch instance to launch EC2 instance

- Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).



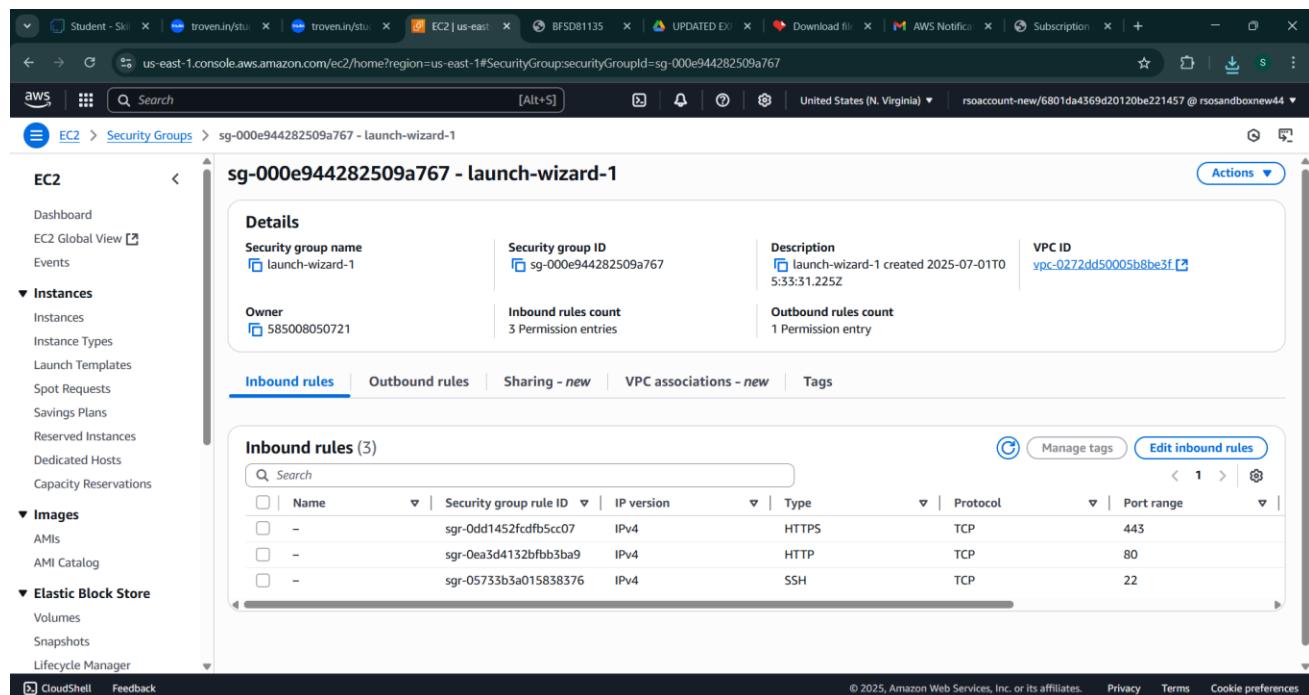
The screenshot shows the AWS EC2 'Launch an instance' wizard. The first step, 'Name and tags', has 'TravelGd' entered in the 'Name' field. The second step, 'Application and OS Images (Amazon Machine Image)', shows 'Amazon Linux' selected from a list of operating systems. The third step, 'Summary', shows the configuration: 1 instance, Amazon Linux 2023 AMI 2023.7.2..., t2.micro instance type, New security group, and 1 volume(s) - 8 GiB storage. A note about the free tier is visible on the right.

- Create and download the key pair for Server access.



The screenshot shows the AWS EC2 Instances page. On the left, there is a navigation sidebar for EC2, including sections for Dashboard, EC2 Global View, Events, Instances (with sub-options like Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area displays the 'Instances (1/1) Info' section. A table lists one instance: TravelGo (i-0bf7730f2c0fdff83). The instance is in a Pending state, t2.micro type, and is associated with the sg-000e944282509a767 security group. Below the table, the specific configuration for this security group is shown, including its inbound rules.

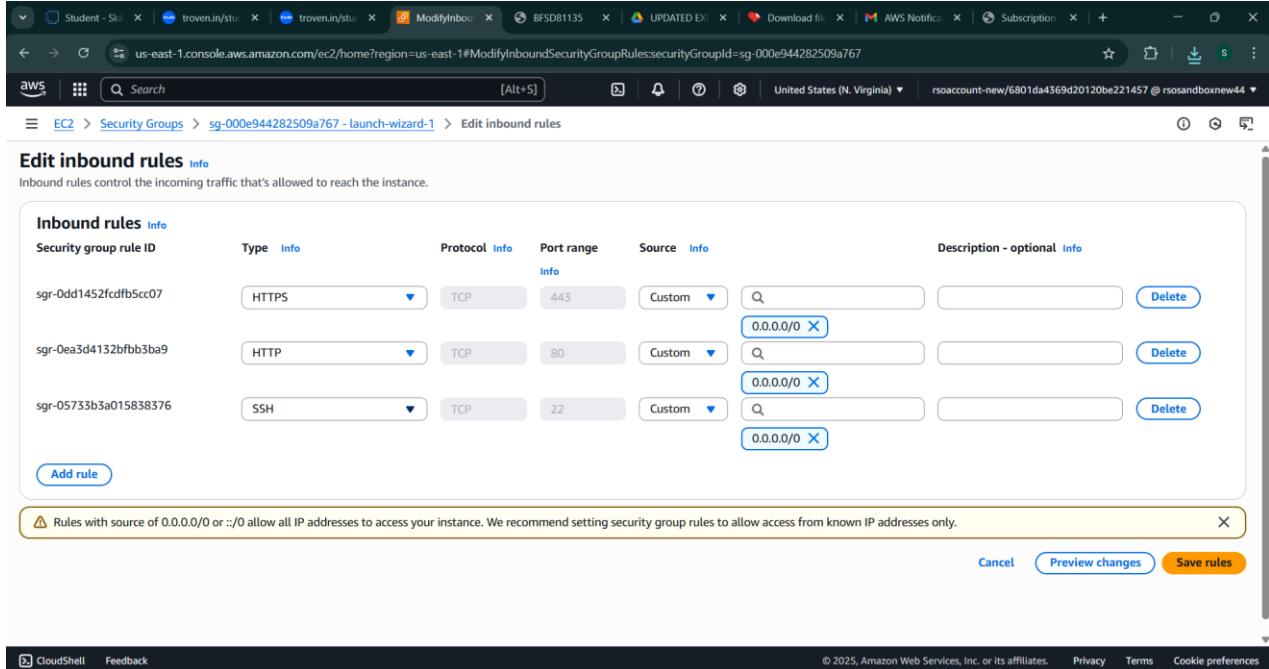
- **Activity 6.2:Configure security groups for HTTP, and SSH access.**



The screenshot shows the AWS Security Groups page. The security group selected is sg-000e944282509a767 - launch-wizard-1. The 'Details' section provides information about the security group, including its name, ID, description, owner, and VPC ID. The 'Inbound rules' tab is selected, showing three rules:

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0dd1452fcfb5cc07	IPv4	HTTPS	TCP	443
-	sgr-0ea3d4132bfbb3ba9	IPv4	HTTP	TCP	80
-	sgr-05733b3a015838376	IPv4	SSH	TCP	22

- **Activity 6.2:Configure security groups for HTTP, and SSH access.**



Edit inbound rules Info

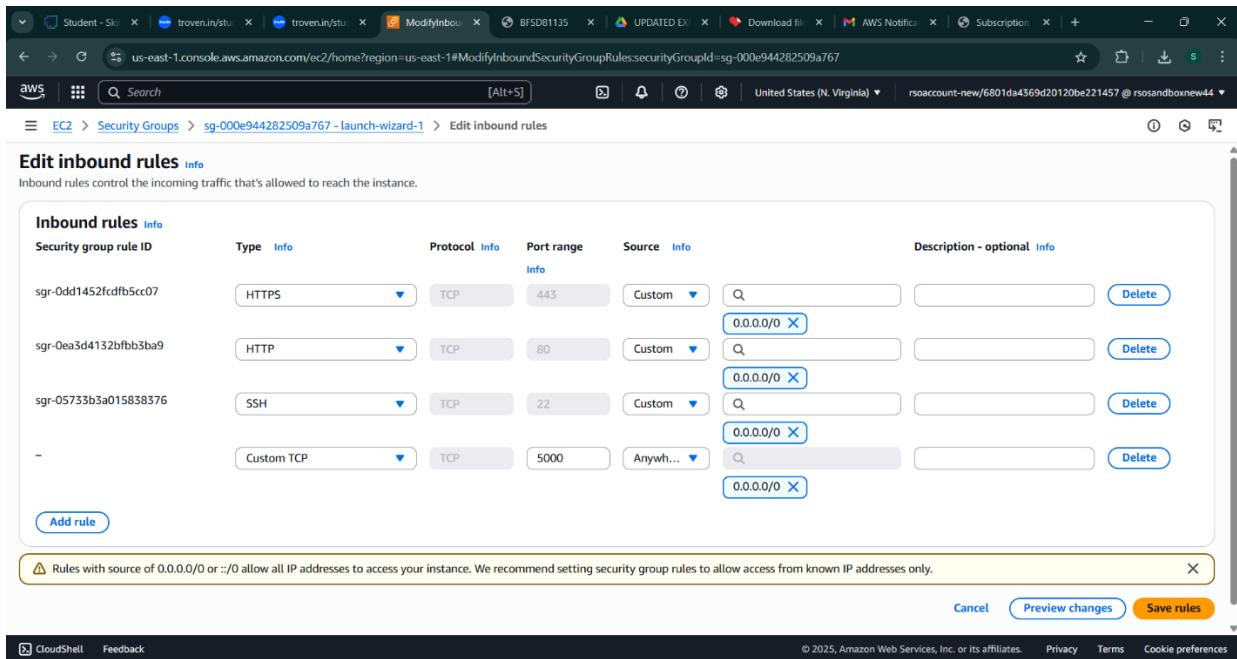
Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0dd1452fcdfb5cc07	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0ea3d4132bfb3ba9	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-05733b3a015838376	SSH	TCP	22	Custom	0.0.0.0/0

Add rule

⚠ Rules with source of 0.0.0.0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes **Save rules**



Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

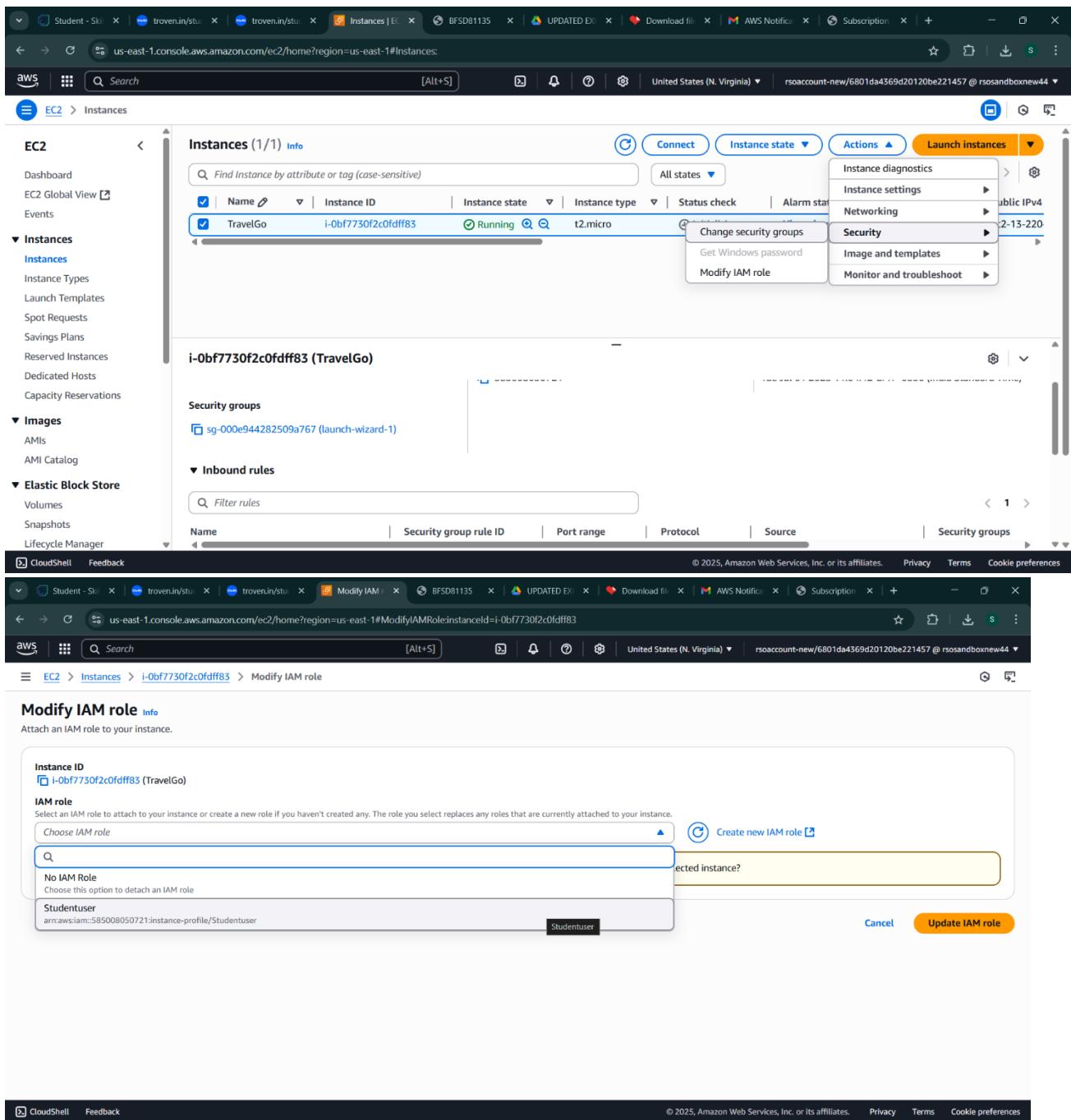
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0dd1452fcdfb5cc07	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0ea3d4132bfb3ba9	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-05733b3a015838376	SSH	TCP	22	Custom	0.0.0.0/0
-	Custom TCP	TCP	5000	Anywhere	0.0.0.0/0

Add rule

⚠ Rules with source of 0.0.0.0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

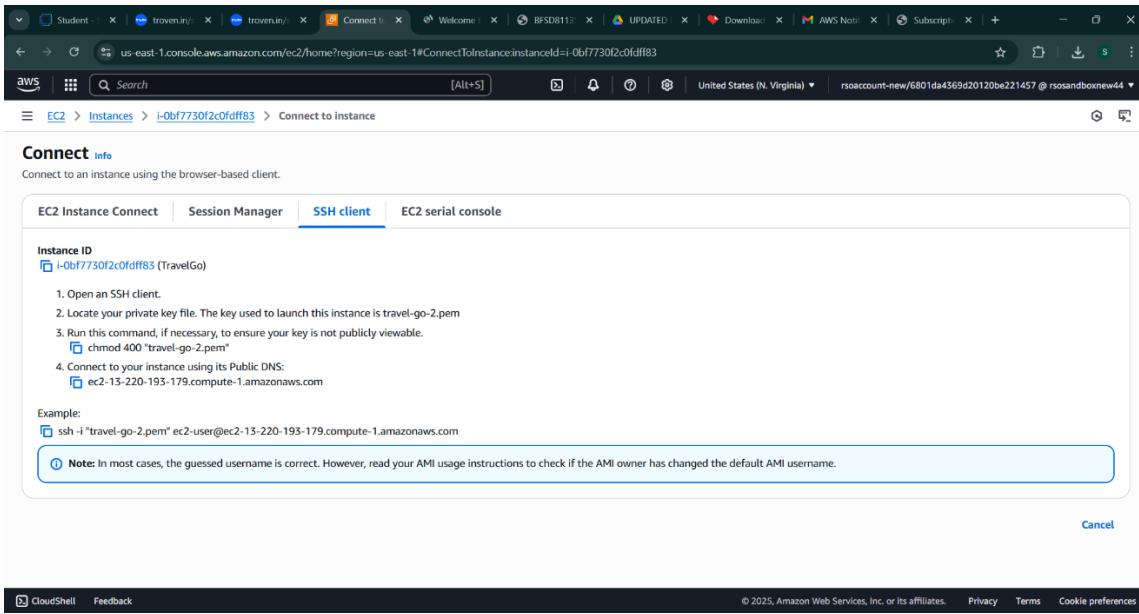
Cancel Preview changes **Save rules**

- To connect to EC2 using **EC2 Instance Connect**, start by ensuring that an **IAM role** is attached to your EC2 instance. You can do this by selecting your instance, clicking on **Actions**, then navigating to **Security** and selecting **Modify IAM Role** to attach the appropriate role. After the IAM role is connected, navigate to the **EC2** section in the **AWS Management Console**. Select the **EC2 instance** you wish to connect to. At the top of the **EC2 Dashboard**, click the **Connect** button. From the connection methods presented, choose **EC2 Instance Connect**. Finally, click **Connect** again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.

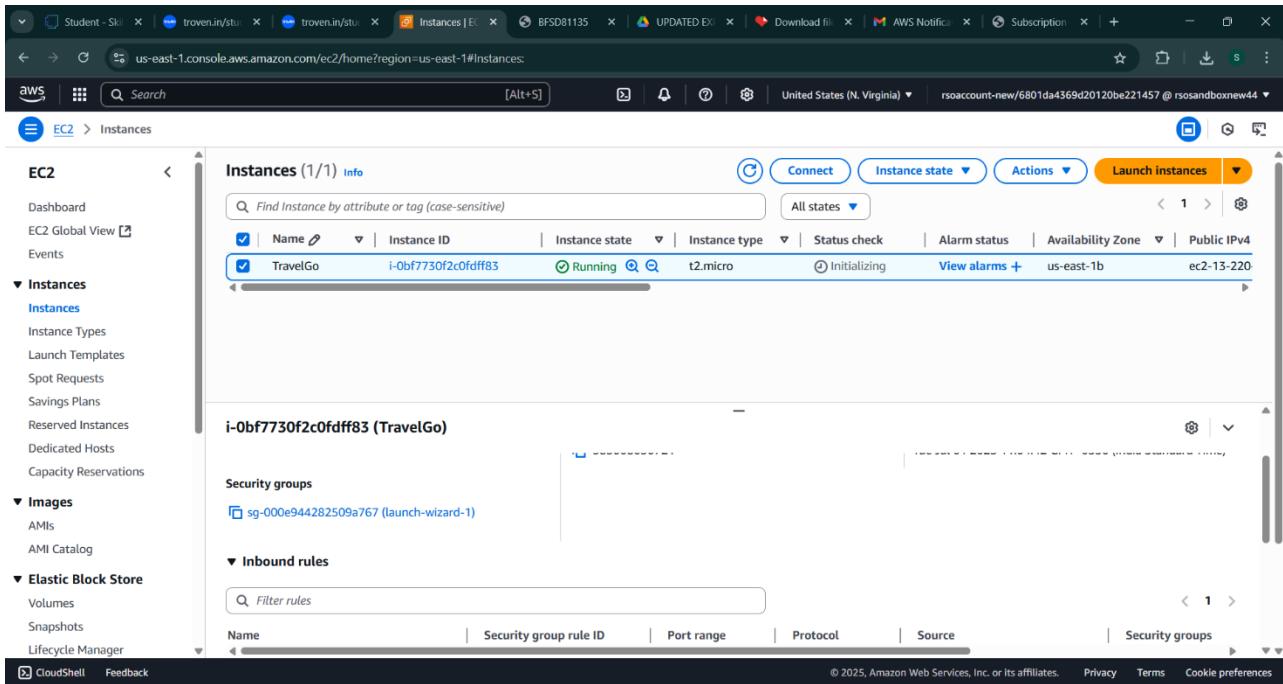


The screenshot shows two overlapping browser windows. The top window displays the AWS Management Console's EC2 Instances page. On the left, a sidebar menu includes options like Dashboard, EC2 Global View, Events, Instances (selected), Images, and Elastic Block Store. The main content area shows one instance named "TravelGo" (i-0bf7730f2c0fdff83) which is "Running". A context menu is open over this instance, with the "Security" option highlighted. The bottom window shows the "Modify IAM role" dialog for the same instance. It has fields for "Instance ID" (i-0bf7730f2c0fdff83 (TravelGo)), "IAM role" (with a dropdown for "Choose IAM role" and a "Create new IAM role" button), and "Selected instance?" (with a dropdown for "Studentuser"). Buttons at the bottom include "Cancel" and "Update IAM role".

- Now connect the EC2 with the files



The screenshot shows the AWS EC2 Connect interface. At the top, it displays the URL `us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance$instanceId=i-0bf7730f2c0fdff83`. Below this, the breadcrumb navigation shows `EC2 > Instances > i-0bf7730f2c0fdff83 > Connect to instance`. The main content area is titled "Connect Info" and contains instructions for connecting via an SSH client. It includes steps like opening an SSH client, locating the private key file, running commands to ensure the key is not publicly viewable, and connecting using the Public DNS. A note at the bottom states: "Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username." A "Cancel" button is located at the bottom right.



The screenshot shows the AWS EC2 Instances page. The left sidebar is expanded to show categories like Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. Under the Instances category, "Instances" is selected. The main content area displays a table titled "Instances (1/1) Info". The table lists one instance: "TravelGo" (Instance ID: i-0bf7730f2c0fdff83), which is "Running" and has an "t2.micro" instance type. The "Actions" column for this instance shows "View alarms" and "Launch instances". Below the table, the specific instance details for "i-0bf7730f2c0fdff83 (TravelGo)" are shown, including its security group ("sg-000e944282509a767 (launch-wizard-1)") and inbound rules. A note at the bottom of the page states: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3, Flask, and Git:

On Amazon Linux 2:

```
sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
```

Verify Installations:

```
flask --version
git --version
```

Activity 7.2: Clone Your Flask Project from GitHub

Clone your project repository from GitHub into the EC2 instance using Git.

Run: 'git clone <https://github.com/your-github-username/your-repository-name.git>'

Note: change your-github-username and your-repository-name with your credentials

here: 'git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git'

- This will download your project to the EC2 instance.

To navigate to the project directory, run the following command:

```
cd InstantLibrary
```

Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

Run the Flask Application

```
sudo flask run --host=0.0.0.0 --port=80
```

Verify the Flask app is running:

<http://your-ec2-public-ip>

- Run the Flask app on the EC2 instance

```

  Downloading zipp-3.23.0-py3-none-any.whl (10 kB)
Installing collected packages: zipp, markupsafe, werkzeug, jinja2, itsdangerous, importlib-metadata, click, blinker, flask
Successfully installed blinker-1.9.0 click-8.1.8 flask-3.1.1 importlib-metadata-8.7.0 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.2 werkzeug-3.1.3 zipp-3.23.0
[ec2-user@ip-172-31-26-55 Travel-go]$ pip install boto3
Defaulting to user installation because normal site-packages is not writeable
Collecting boto3
  Downloading boto3-1.39.3-py3-none-any.whl (139 kB)
    |████████| 139 kB 16.3 MB/s
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
    |████████| 85 kB 7.6 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting botocore<1.40.0,>=1.39.3
  Downloading botocore-1.39.3-py3-none-any.whl (13.8 MB)
    |████████| 13.8 MB 39.1 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.3->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.39.3 botocore-1.39.3 s3transfer-0.13.0
[ec2-user@ip-172-31-26-55 Travel-go]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.26.55:5000
Press CTRL+C to quit
 * Restarting with stat

```

Milestone 8: Testing and Deployment

- **Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.**

Welcome Page:

Register Page:

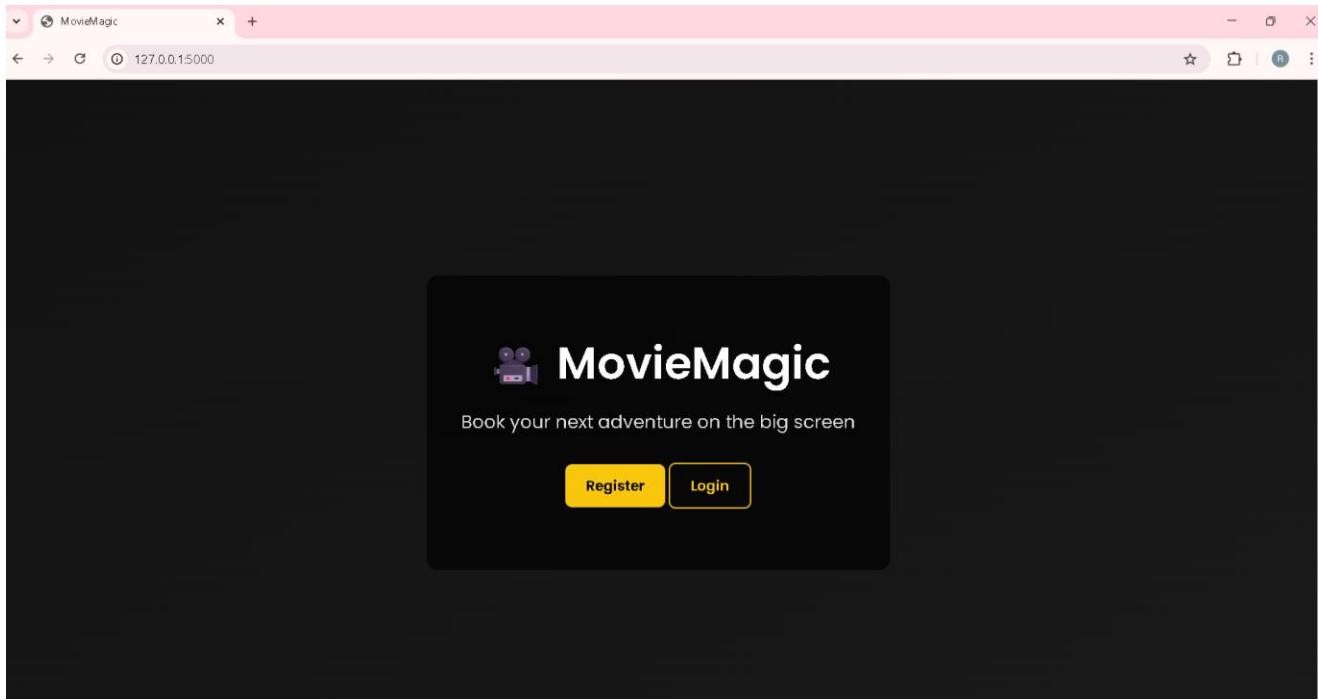
Login Page:

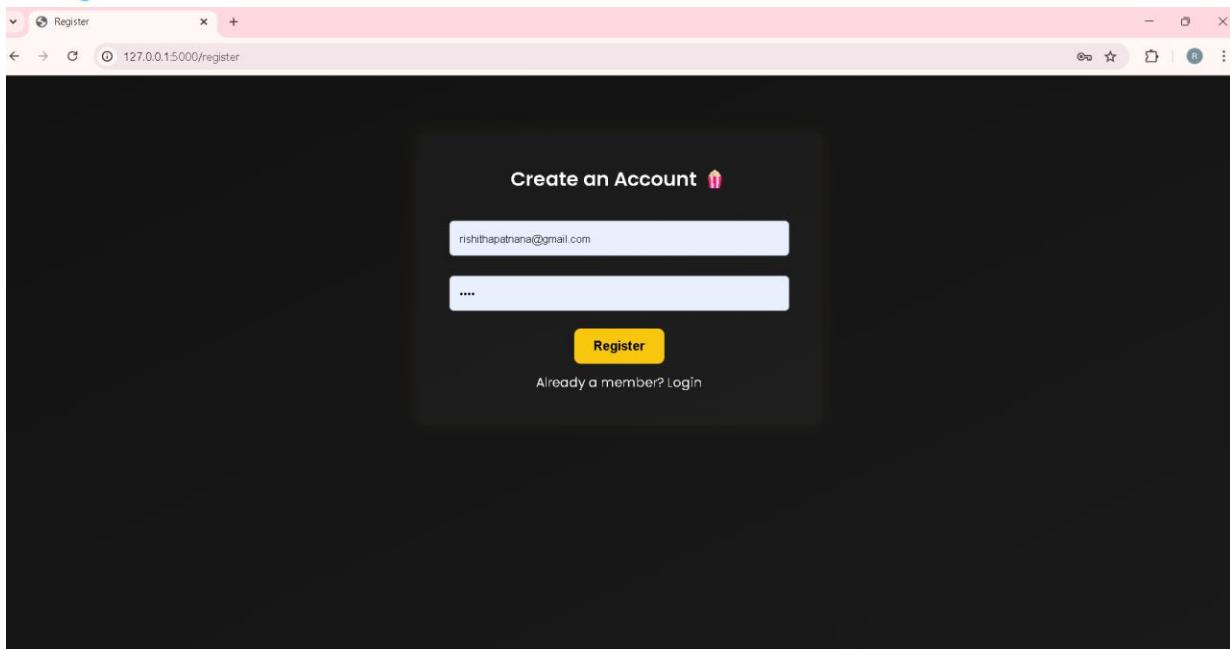
Home page:

Booking page:

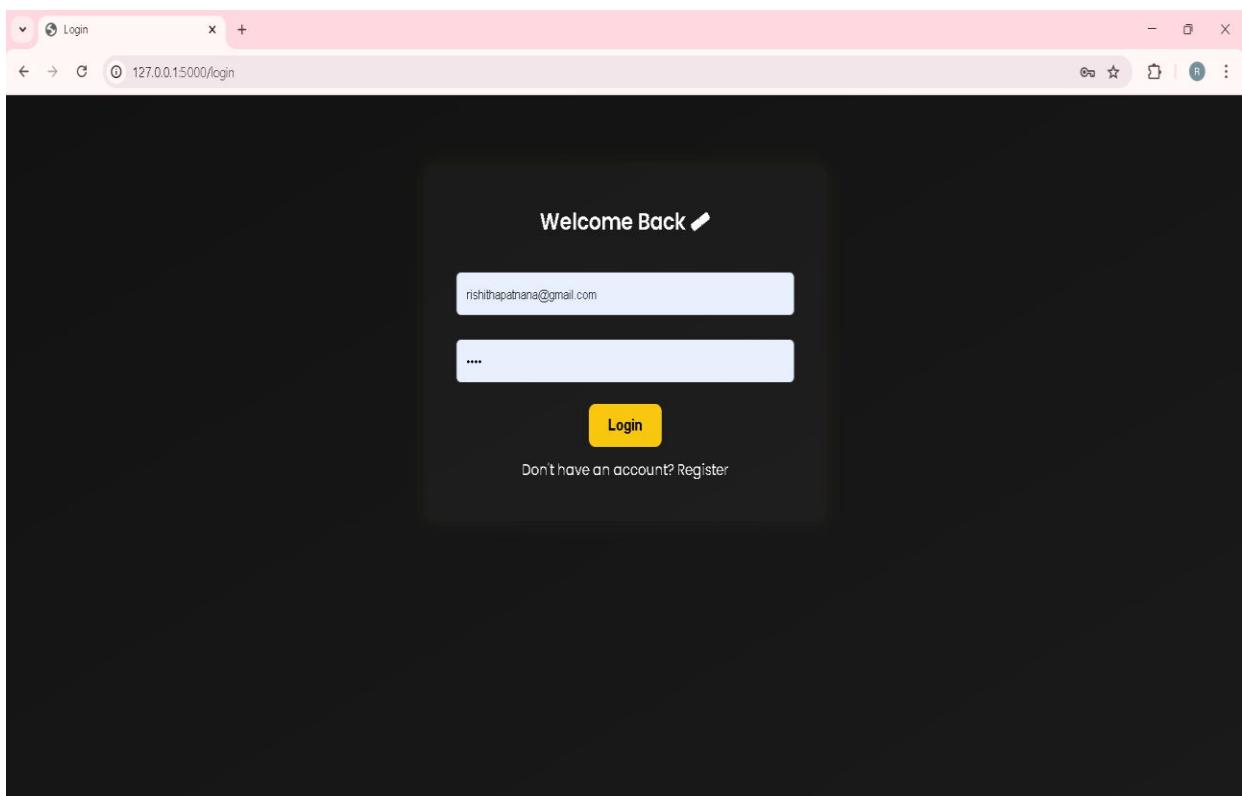
Payment page:

Confirm page:





The screenshot shows a registration form titled "Create an Account". It features two input fields: one for email (rishithapathana@gmail.com) and one for password (laptop123). A yellow "Register" button is centered below the fields. Below the button, a link says "Already a member? Login". The background is black, and the overall design is clean and modern.



The screenshot shows a login form titled "Welcome Back". It has two input fields for email (rishithapathana@gmail.com) and password (laptop123). A yellow "Login" button is positioned at the bottom. Below the button, a link says "Don't have an account? Register". The layout is identical to the registration page, maintaining a consistent design across both pages.

MovieMagic | Home

127.0.0.1:5000/home

Welcome to MovieMagic

Discover, book, and experience the magic of cinema

[Book Tickets](#)

All Action Comedy Drama Horror Romance

Now Showing

The Grand Premiere	Lahgt Riot	Edge of Tomorrow	Haunted Nights
 The Grand Premiere	 Laugh Riot	 Edge of Tomorrow	 Haunted Nights
Drama	Comedy	Action	Horror
Book Now	Book Now	Book Now	Book Now

Book Tickets

127.0.0.1:5000/booking

Book for Example Movie

Date

12-07-2025

Time

09:30

Seat

 Example Movie

C5

A1	A2	A3	A4	A5
B1	B2	B3	B4	B5
C1	C2	C3	C4	C5

[Proceed to Payment](#)

Payment

127.0.0.1:5000/payment

Payment Details

Cardholder Name: Rishitha

Card Number: 3994u030i

Expiry (MM/YY): 03/04

CVV: 304

Pay & Confirm

Confirmation

127.0.0.1:5000/confirmation

Your ticket is booked!

Back to Home

Conclusion

The Movie Magic website has been successfully developed and deployed using a robust, cloud-native architecture. By leveraging key AWS services such as EC2 for hosting, DynamoDB for real-time data management, and SNS for instant booking confirmations, the platform delivers a scalable and user-friendly movie ticketing experience. This solution addresses the limitations of traditional ticket booking systems by enabling users to seamlessly search for movies, select seats, and receive digital tickets—all from a single platform.

The cloud-based infrastructure ensures that the system can handle a high volume of users and transactions, especially during peak movie release times, without compromising speed or reliability. The integration of Flask with AWS services enables smooth backend operations, including user authentication, movie/event browsing, seat availability checks, and secure ticket booking.

Comprehensive testing has confirmed that all core functionalities—from user registration and login to booking confirmation via email—operate seamlessly. The platform's modern interface and responsive design further enhance the user experience, making movie booking quick, efficient, and enjoyable.

In conclusion, Movie Magic stands as a powerful demonstration of how cloud technologies can modernize traditional services. It provides an efficient, scalable, and intuitive solution for movie ticket booking, showcasing the potential of full-stack cloud applications in solving real-world user experience challenges in the entertainment industry.

THANK YOU