

Project Documentation: Movie Ticket Booking System

Hrushikesh Adya (700749931)

Big Data Analytics and IT, University of Central Missouri

Readings in Computer Information Systems

Professor: Silvana Faja

Index

1. Project Overview
2. Technology Stack
3. Folder Structure
4. Technical Implementation Details
5. AWS Infrastructure Details
6. Screenshots of UI
7. Code Explanation

Project Overview:

The Movie Ticket Booking System will serve as a bridge between movie enthusiasts and the Ticket Selling Company through multiple theatre chains. A user will be able to see the available shows in the theatre and book a movie show. The admin will have access to control the available movies on the platform. All the ticket booking and selling details will be stored in the database for multiple use cases.

Technology Stack:**Front End Development:**

- Scripting Language: HTML, CSS
- Programming Language: Typescript
- Framework: React JS

Back End Development:

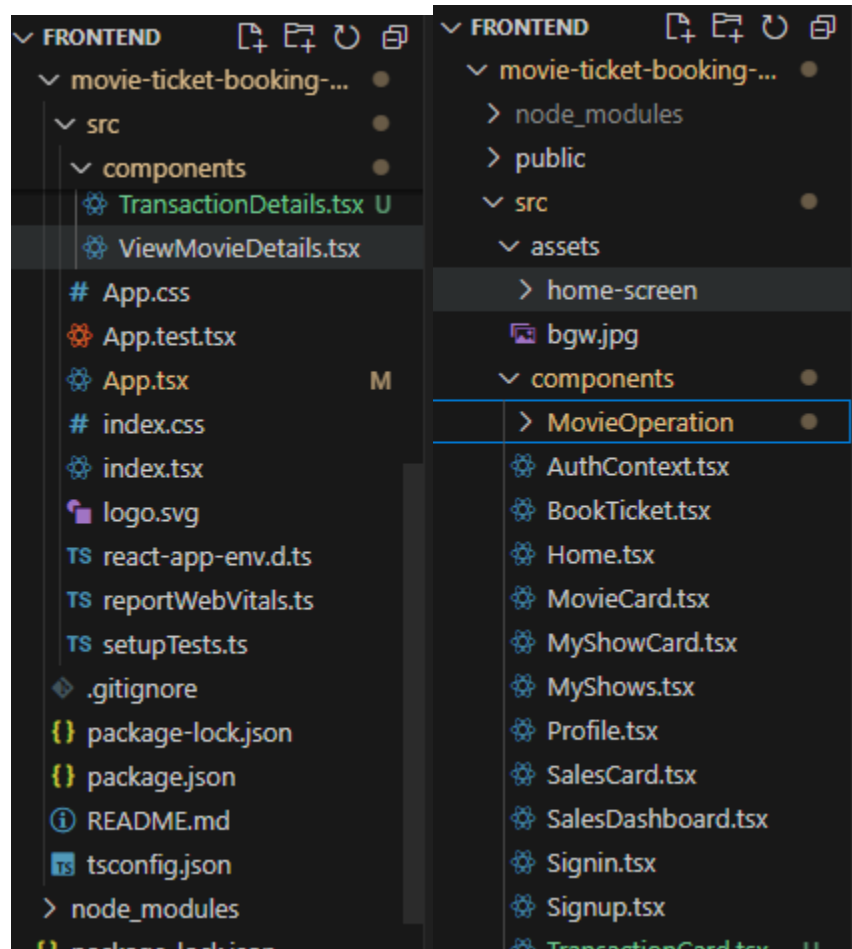
- Programming Language: Python

Cloud Platform:

- Amazon Web Services (AWS)

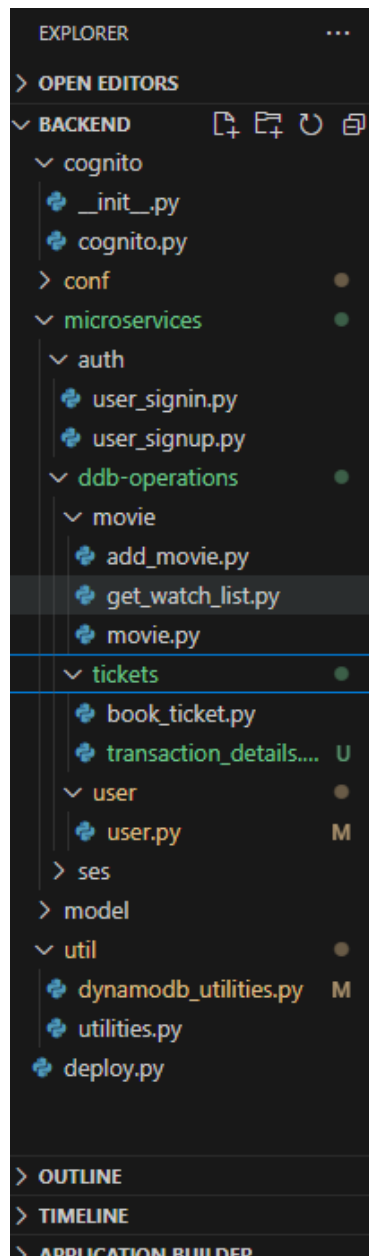
Folder Structure:

Front End Folder Structure:



Above screenshots show folder structure for front end code development done with React JS Framework.

Back End Folder Structure:



Here is the folder structure for backend code development done with Python and AWS boto3 SDKs.

Technical Implementation Details:

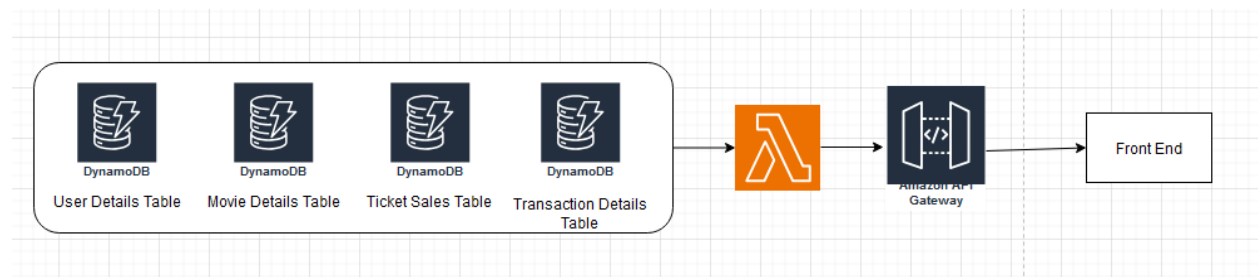
Movie ticket booking system app is developed using React JS Framework, HTML, CSS, and Python for backend development with boto3 AWS SDKs.

For every screen in the UI where data needs to be pulled from the backend or database, we are calling HTTP API using `fetch ()` function with desired HTTP method. The response is carefully filtered and showcased into UI.

This API was created by using AWS API Gateway Service. API Gateway service in AWS allows you to create an API Endpoint which can be used to link other AWS services to third party applications.

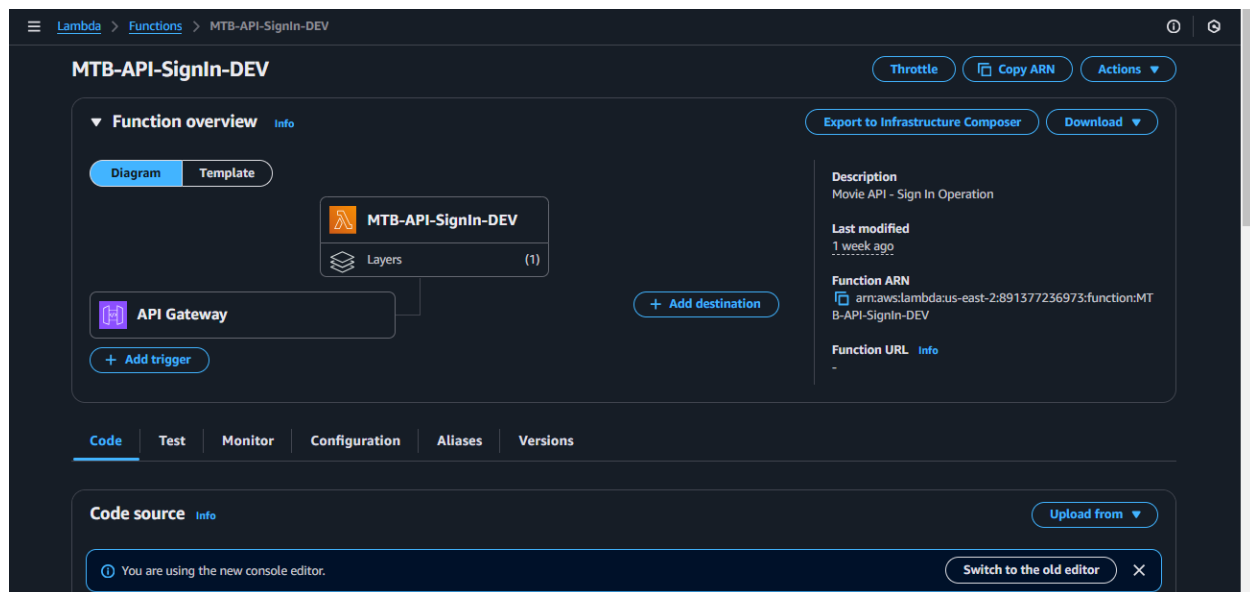
In our case, we are using AWS Lambda function written in Python behind every API which performs multiple DB operations on DynamoDB Tables.

AWS Infrastructure Details:

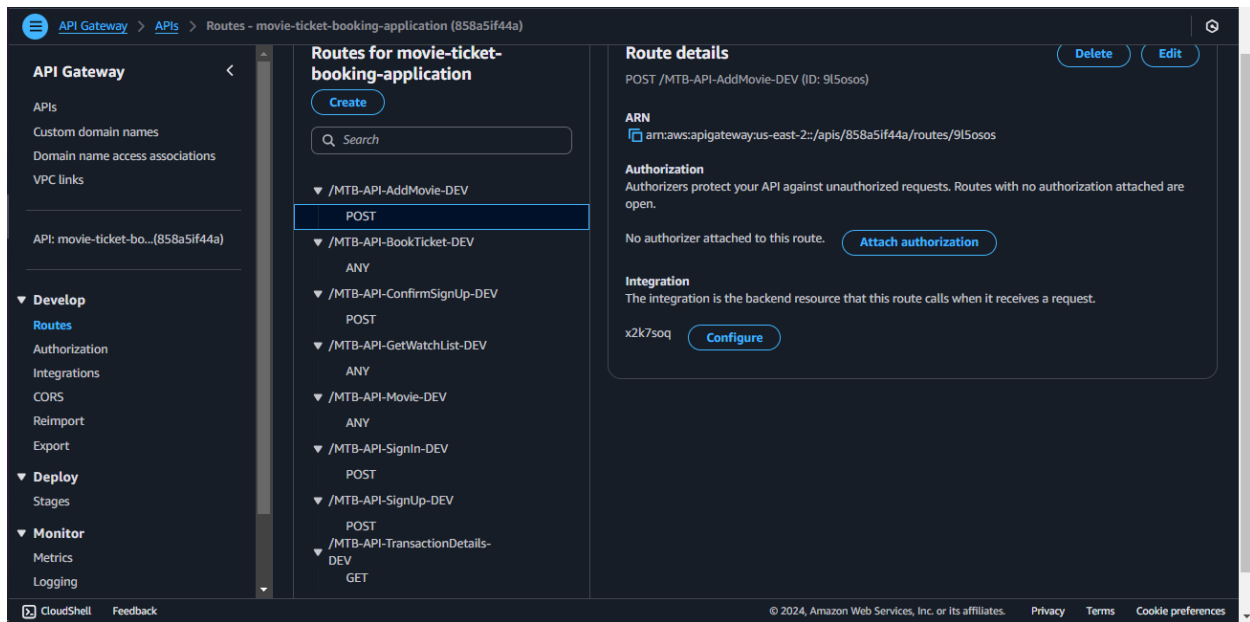


All the details are stored in DynamoDB Tables as shown in the above architecture diagram. These DynamoDB tables are linked with Lambda Functions which are triggered by APIs created with API Gateway service.

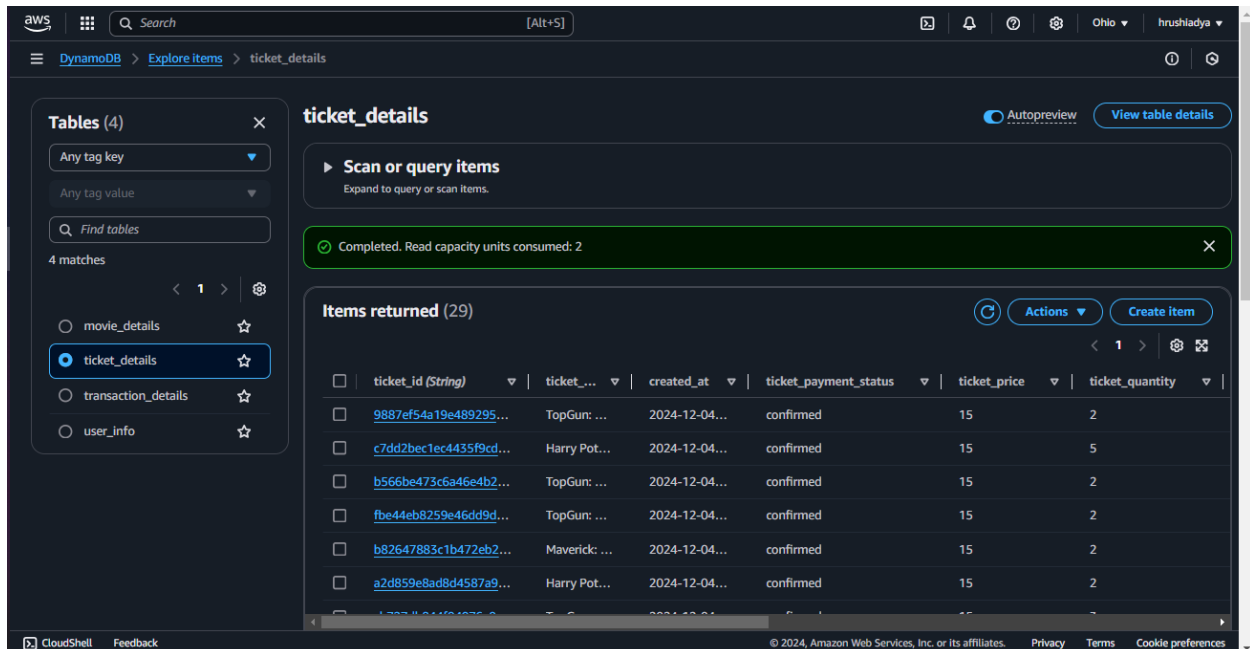
Lambda Function Integration with API Gateway:



API Gateway Bundles:



DynamoDB Tables:



Screenshots of User Interface:

Sign Up Page:

UCM Movie Theatre

Home Sign In Sign Up

Sign Up

Email ID:

Password:

First Name:

Last Name:

Phone:

Profile Type: ☐ Admin ☐ User

Sign In Page:

UCM Movie Theatre

HomeSign InSign Up

Login

Username:

hrushikesh.adya@gmail.com

Password:

Login

Home Page:

UCM Movie Theatre

HomeSales DashboardMovie OperationTransaction DetailsMy ShowsSign OutMy Profile

Harry Potter and Goblet of Fire

Harry Potter and Goblet of Fire

Fiction Movie

Rating:

BookSelect

TopGun: Maverick

TopGun: Maverick

Airforce Action Movie.

Rating:

BookSelect

Book Ticket Page:

UCM Movie Theatre

HomeSales DashboardMovie OperationTransaction DetailsMy ShowsSign OutMy Profile

Book Ticket for

Theatre:

Select Theatre

Your Name:

hrushikesh.adya@gmail.com

Number of Seats:

Movie Name:

Harry Potter and Goblet of Fi

Show Time:

Ticket Price:

15

Confirm Booking

Sales Dashboard:

Harry Potter and Goblet of Fire

7 Revenue: 105

TopGun: Maverick

25 Revenue: 375

Add Movie Operation Page:

Movie Operations[Add Movie](#)[Edit Movie](#)[Search Movie](#)[Remove Movie](#)**Add Movie**

Title:

Description:

Genre:

Director:

Release Date:

Ticket Price:

Length:

Thumbnail URL:

Available:

Showtimes:

[Submit](#)

Update Movie Operation:

Movie Operations[Add Movie](#)[Edit Movie](#)[Search Movie](#)[Remove Movie](#)**Update Movie Details**

Title:

Description:

Genre:

Director:

Release Date:

Ticket Price:

Length:

Thumbnail URL:

Available:

Showtimes:

[Submit](#)

Search Page:

UCM Movie Theatre

Home

Sales Dashboard

Movie Operation

Transaction Details

My Shows

Sign Out

My Profile

Movie Operations

Add Movie

Edit Movie

Search Movie

Remove Movie

Search Movie

Enter movie title

Search

Delete Movie Page:

UCM Movie Theatre

Home

Sales Dashboard

Movie Operation

Transaction Details

My Shows

Sign Out

My Profile

Movie Operations

Add Movie

Edit Movie

Search Movie

Remove Movie

Enter Movie Name:

Submit

Transaction Details between two dates feature:

UCM Movie Theatre

Home

Sales Dashboard

Movie Operation

Transaction Details

My Shows

Sign Out

My Profile

Start Date: 12/04/2024

End Date: 12/09/2024

Submit

9887ef54a19e48929574de55d74

Movie: TopGun: Maverick

Amount: 15

Date: 12-05-2024

b566be473c6a46e4b2a2a62806f

Movie: TopGun: Maverick

Amount: 15

Date: 12-06-2024

fbe44eb8259e46dd9da191693e4

Movie: TopGun: Maverick

Amount: 15

Date: 12-05-2024

My Shows Page:

UCM Movie Theatre

[Home](#) [Sales Dashboard](#) [Movie Operation](#) [Transaction Details](#) [My Shows](#) [Sign Out](#) [My Profile](#)

TopGun: Maverick
4:00 PM
Ticket Quantity 2
Transaction ID
9887ef54a19e48929574de55d740c227

Harry Potter and Goblet of Fire
4:00 PM
Ticket Quantity 5
Transaction ID
76f816180a71418fbc2730c040e4a9f0

TopGun: Maverick
4:00 PM
Ticket Quantity 3
Transaction ID
cb727db844f04076a9c7e221f6fe90e2

My Profile Page:

UCM Movie Theatre

[Home](#) [Sales Dashboard](#) [Movie Operation](#) [Transaction Details](#) [My Shows](#) [Sign Out](#) [My Profile](#)

My Profile

Welcome, hrushikesh.adya@gmail.com!

First Name:

Last Name:

Email:

Phone:

Update Profile

Code Explanation:

In this section we will explore functionality of one API from front end to back end.

UCM Movie Theatre

[Home](#) [Sales Dashboard](#) [Movie Operation](#) [Transaction Details](#) [My Shows](#) [Sign Out](#) [My Profile](#)

TopGun: Maverick
4:00 PM
Ticket Quantity 2
Transaction ID
9887ef54a19e48929574de55d740c227

Harry Potter and Goblet of Fire
4:00 PM
Ticket Quantity 5
Transaction ID
76f816180a71418fbc2730c040e4a9f0

TopGun: Maverick
4:00 PM
Ticket Quantity 3
Transaction ID
cb727db844f04076a9c7e221f6fe90e2

My shows tab shows the user all the shows user has booked on movie ticket booking system.

```

const MyShows: React.FC = () => {
  const { isLoggedIn, username } = useAuth();
  const location = useLocation();
  const [responseMessage, setResponseMessage] = useState<string | null>(null);
  const [isLoading, setIsLoading] = useState<boolean>(false);
  const [errorMessage, setErrorMessage] = useState<string | null>(null);
  const navigate = useNavigate();

  const apiGatewayUrl = 'https://858a51f44a.execute-api.us-east-2.amazonaws.com/dev/MTB-API-GetWatchList-DEV';
  const [movies, setMovies] = useState<any[]>([]);

  console.log("Username: ", username);
  const fetchData = async () => {
    try {
      const response = await fetch(`${apiGatewayUrl}?user_id=${username}`, {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
        },
      });

      const data = await response.json();
      console.log('Data:', data);
      if (response.ok) {
        console.log('Data:', data);
        setMovies(data);
      } else {
        setErrorMessage(data.message || 'Failed to get WatchList from API.');
```

In MyShows.tsx file we have used GetWatchList API which is returning a list of shows watched by logged in user.

```

return (
  <div>
    {isLoggedIn ? (
      <div>
        <div style={gridStyle}>
          {movies.map((movie, index) => (
            <MyShowsCard
              key={index}
              title={movie.movie_name}
              show_date={movie.show_date}
              ticket_quantity={movie.ticket_quantity}
              transaction_id={movie.transaction_id}
            />
          ))}
        </div>
      </div>
    )}
  </div>
)

```

The data returned with API is showcased in front end UI using HTML and CSS.

```

def lambda_handler(event, context):
    print("Event: ", event)
    if 'httpMethod' not in event:
        raise RuntimeError('No HttpMethod')
    logger.info("Event:")
    logger.info(json.dumps(event))
    http_method = event["httpMethod"]

    try:
        if http_method == "GET":
            if event['queryStringParameters'] is not None:
                user_id = event['queryStringParameters']['user_id']
                print("user_id ", user_id)
                user = get_user_by_key(user_id, user_id)
                print("user: ", user)
                response = user['watch_list']
                print("response: ", response)
                return {
                    'statusCode': 200,
                    'body': json.dumps(response, cls=DecimalEncoder)
                }
    except JSONDecodeError as e:
        raise JSONDecodeError('Error when decoding json body', inner=e)
    except Exception as e:
        raise Exception('Error when performing GET API', e)

```

This is the backend lambda function code for get_watch_list functionality which is helping us get the users watch list from DynamoDB table.