MOVIE BOOKING APP

IIHT

CONTENTS

1 PROBLEM STATEMENT

Movie Booking App is an application that allows the users to register and login and search movies released. User can book the tickets for the movies they wish for. Admin shall view the tickets booked and update the pending tickets available to the system.

Guest users cannot book any tickets.

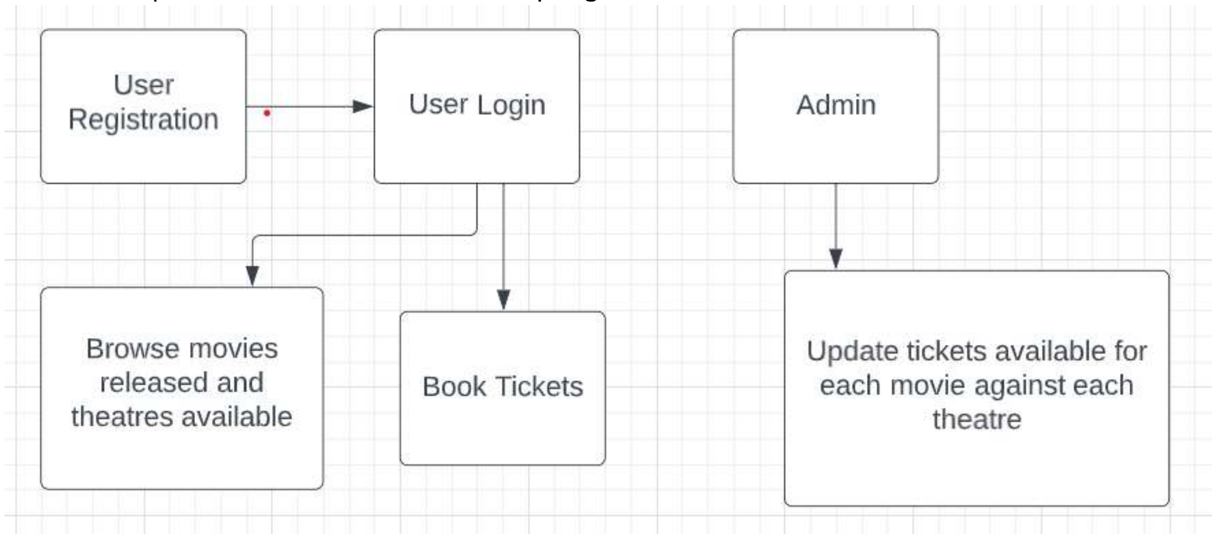
The core modules of movie booking app are:

1. User registration and login
2. Browse the movies released and list of theatres
3. User can book the tickets
4. Admin to update the available ticket status based on the tickets booked

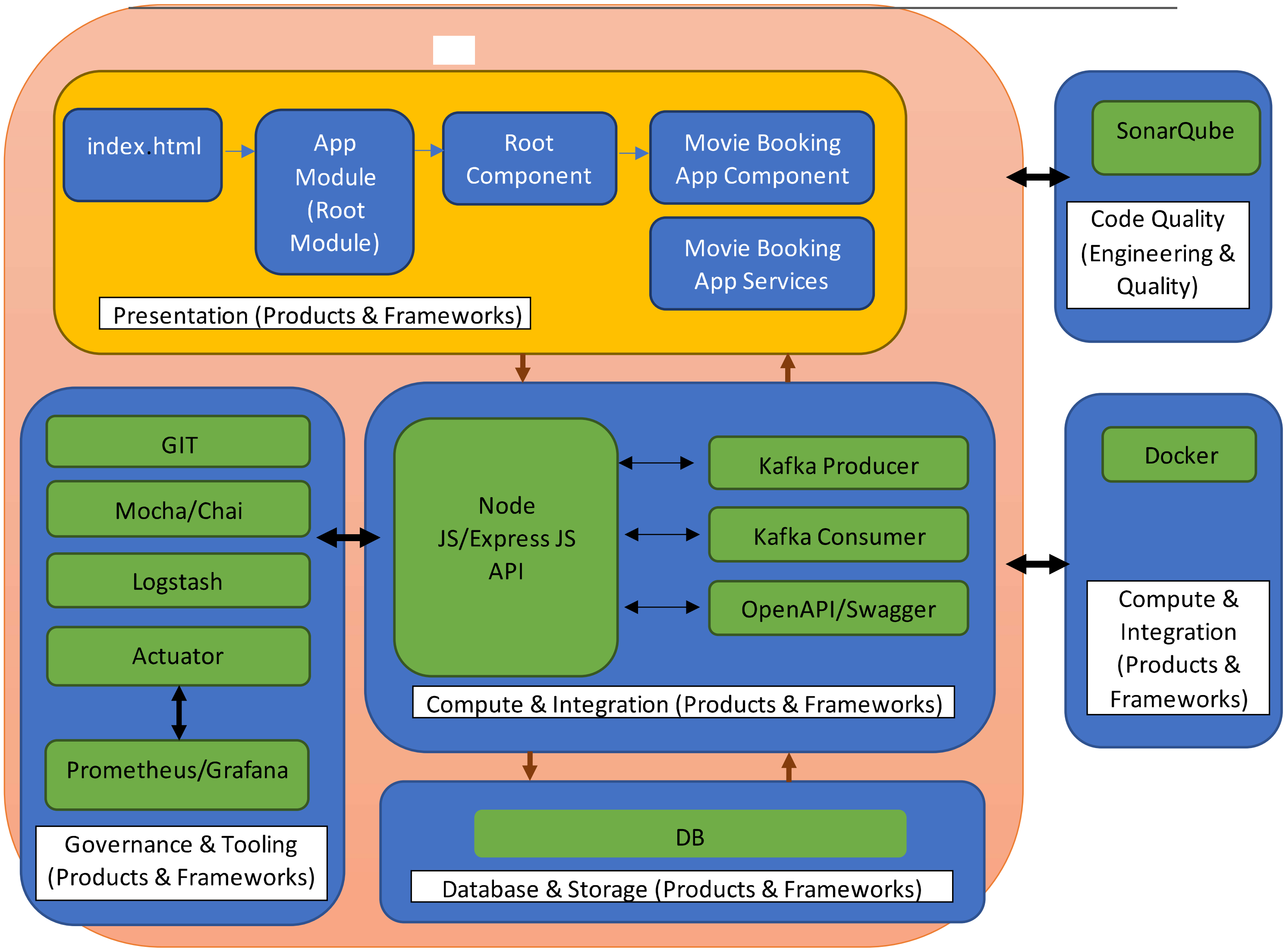
The scope includes developing the application using toolchain mentioned below.

2 PROPOSED MOVIE BOOKING APP WIREFRAME

1. UI needs improvisation and modification as per given use case.

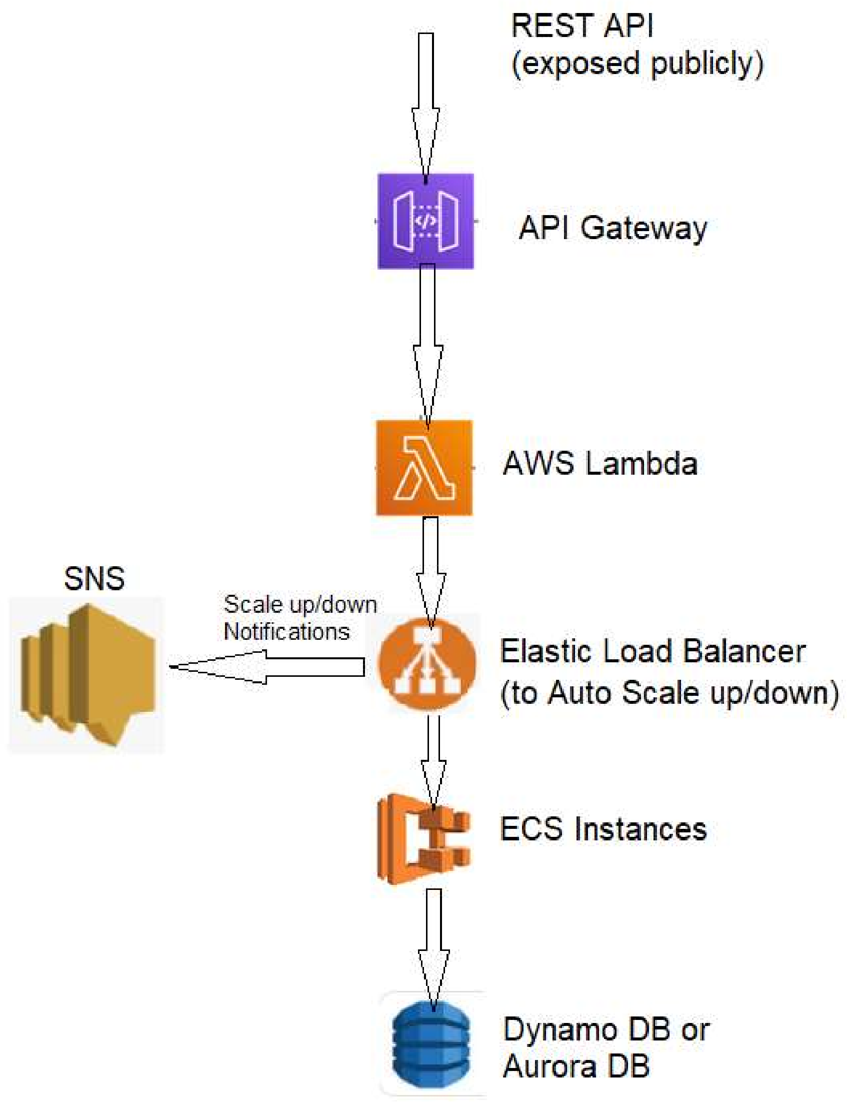

User Registration
User Login
Admin
Browse movies released and theatres available
Book Tickets
Update tickets available for each movie against each theatre

3 Application Architecture


index.html
App Module (Root Module)
Root Component
Movie Booking App Component
Movie Booking App Services
Presentation (Products & Frameworks)
SonarQube
Code Quality (Engineering & Quality)
GIT
Mocha/Chai
Logstash
Actuator
Prometheus/Grafana
Governance & Tooling (Products & Frameworks)
Node JS/Express JS API
Kafka Producer
Kafka Consumer
OpenAPI/Swagger
Compute & Integration (Products & Frameworks)
Docker
Compute & Integration (Products & Frameworks)
DB
Database & Storage (Products & Frameworks)

4 Cloud Architecture

Below Diagram shows how Online Application App can be deployed on AWS Cloud



5 Tool Chain

Competency	Skill	Skill Detail
Engineering Mindset	Networking and Content Delivery	
	Ways of Working	
	Consulting Mindset	
	DevOps	
Programming Languages	Application Language	JavaScript
Products & Frameworks	Presentation	Angular
		Karma & Jasmine
	Compute & Integration	Node JS/Express JS

		Kafka/Kube MQ/Active MQ/RabbitMQ
		Docker
	Database & Storage	MongoDB/Cassandra/Redis
	Governance & Tooling	Git
		Mocha/Chai
		Jest
		Mockito
		Logstash
		Prometheus & Grafana
Engineering Quality	Code Quality	Sonar Cube
Platform	Cloud Tools	AWS ECS
		AWS DynamoDB/Aurora
		AWS Lambda
		AWS Elastic Cache
		AWS Code Deploy
		AWS API Gateway
		AWS ELB (Elastic Load Balancer)
		AWS SNS

6 DEVELOPMENT FLOW

No	MC	Competency	Section	Indicative Mechanism for Evaluation (Passing score of 60% in each MC)
Business Requirement				
1	Backend	Rest API, Database, Messaging, Log/Monitoring, Debugging & Troubleshooting	Click here	Code Submission and Evaluation, Panel Presentation
2	Front End	Angular/React	Click here	Code Submission and Evaluation, Panel Presentation
3	Cloud	Compute, Identity, Compliance, Security and Content Delivery	Click here	Code Submission and Evaluation, Panel Presentation

7 BUSINESS-REQUIREMENT:

As an application developer, develop frontend, middleware and deploy the Movie Booking App with below guidelines:

Pre-requisite before implementing the user story:

Create a static database as “Movie” and add the below fields

- Movie name, Total number of tickets allotted and theatre name (Movie name and theatre name should be of composite primary key)

There should be a predefined set of tickets assigned to each theatre against each movie. Just create sample 2 movies with 2 theatres assigned. This will be used for the upcoming user story to fetch the data.

User Story #	User Story Name	User Story
US_01	Registration and Login	As a user I should be able to login/Register in the movie booking application Acceptance criteria: 1. A logged-in user can reset their password so they can login, even if they forget their password. 2. A logged-in user: a. Cannot change their username. b. Can logout from their account. 3. As a customer I should be able to furnish following details at the time of registration a. First Name

		b. Last Name c. Email d. Login Id e. Password f. Confirm Password g. Contact Number 4. All details fields must be mandatory 5. Login Id and Email must be unique 6. Password and Confirm Password must be same 7. If any constraint is not satisfied, validation message must be shown
US_02	View &Search Movies	As a user I should be able to view all the recent movies opened for booking. User can search any particular movies as well Acceptance criteria: a. User can view all the existing released movies. b. User can search any particular movie based on the movie name
US_03	Book Tickets for a movie	As a user I should be able to book tickets for a movie. Add this booking to a database table "Tickets". Assign movie name and theatre name as a foreign key to database table "Movie" which is already created as a pre-requisite Acceptance criteria: a. Book a movie ticket b. Below are the details to be added • Movie Name • Theatre name • Number of tickets • Seat Number
US_04	View booked tickets and Update ticket status	As an admin I should be able to view booked tickets and update the balance tickets available for a movie Acceptance criteria: a. View number of booked tickets for a particular movie from table "Tickets" b. Check the tickets available from table "Movie" by calculating the total tickets booked c. If the quantity is 0, update the ticket status as 'SOLD OUT', else update as 'BOOK ASAP'

8 RUBRICS/EXPECTED DELIVERABLES

8.1 REST API (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

a. Use Node JS/Express JS to version and implement the REST endpoints.

b. Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

POST	**/api/v1.0/moviebooking/register**	**Register as new user**
GET	**/api/v1.0/ moviebooking /login**	**Login**
GET	**/api/v1.0/ moviebooking /<username>/forgot**	**Forgot password**
GET	**/api/v1.0/ moviebooking /all**	**View all movies**
GET	**/api/v/1.0/moviebooking/movies/search/moviename***	**Search by movie name**
POST	**/api/v1.0/ moviebooking /<moviename>/add**	**Book tickets for a movie**
PUT	**/api/v1.0/moviebooking /<moviename>/update/<ticket>**	**Update ticket status**
DELETE	**/api/v1.0/ moviebooking /<moviename>/delete/<id>**	**Delete movie**

c. ***Movie name may be partial or complete username**

d. Use necessary configuration in place for REST API in application. Properties or bootstrap. Properties or application.yml; whichever is applicable.

e. Package Structure for Node JS/Express JS Project will be like com. moviebookingapp. * With proper naming conventions for package and beans.

f. Use configuration class annotated with @Configuration and @Service for business layer.

g. Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.

h. Follow Node JS/Express JS Bean Naming Conventions

8.2 DATABASE (PRODUCTS & FRAMEWORKS -> DATABASE & STORAGE):

1. As an application developer:
   a. Implement ORM with Node JS/Express JS MongoRepository and MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (Aggregation Operation, Match Operation, Aggregation Results), implementation of MongoTemplateetc as necessary.
   b. Have necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml OR annotation based configuration; whichever is applicable.

8.3 TOOLING:

1. As an application developer:
   a. Create the Node JS/Express JS project using CLI
   b. Generate Sure-fire test reports and share it as a part of deliverables

8.4 MESSAGING (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

1. As an application developer:

   a. Have a centralized logging system

b. Be able to communicate using a messaging infrastructure.
c. Use Kafka Template for communication with Node JS/Express JS and topics in Kafka.
d. Use Kafka for messaging infrastructure and implement admin to read the total number of tickets booked for a movie and available quantity from movie table and admin to write the ticket availability status as provided in the user story
e. Configure Node JS/Express JS app to log all logging messages to Kafka.
f. Configure all Kafka related configuration needed for Node JS/Express JS in *.properties or *.yml file.

8.5 Log/ Monitoring (Products & Frameworks -> Governance & Tooling):

1. As an application developer:

   a. Containerize the complete application, which includes front-end, middleware and Kafka (consumers and producers) using docker and Dockerfile.
   b. Use. dockerignore as necessary to avoid containerizing un-necessary packages.
   c. Integrate Node JS/Express JS Actuator with Prometheus and Grafana to monitor middleware.
   d. Implement logs with Logstash.
   e. Open the preconfigured Logstash in Kibana and check if it successfully connects to Elasticsearch Server.
   f. Perform unit and integration testing of your application and do proper CI/CD

8.6 Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linked with final deliverables which should also suggest the resolution for the encountered bugs and errors.

9 Frontend

1. Develop the front end for all user stories.
2. Implement using either Angular or React
3. Implement all the Front-End validation rules
4. Proper naming conventions and folder structures
5. Implement using proper SOLID design principles
6. Perform unit and integration testing for the front-end application

10 Platform

10.1 Compute

1. Use ECS CLI (as an alternative to AWS Management Console) for container management and deployment of Node JS/Express JS application. You should be able to explain and demonstrate the same in interview.

2. Use NoSQL instance of AWS DynamoDB/Aurora (SQL) as a database for the Movie Booking App Application

10.2 Compute, Identity & Compliance, Security& Content Delivery

1. Use AWS Lambda and AWS Aurora to build a backend process for handling requests for Movie Booking App.
2. Use Serverless Container using AWS ECS and run the Movie Booking app created with Node JS/Express JS inside AWS Lambda.
3. Use Amazon API Gateway to expose the lambda functions built in the previous step to be accessible on public internet.
4. Use AWS ELB to configure the auto-scaling container instances.
5. Configure AWS SNS to issue messages whenever a ELB scales-up and scale-down container instances

Note – Minimum two rest endpoints should be hosted in cloud

10.3 Frontend Deployment

Deploy the Frontend solution as follows:

1. Maintain the production build of Frontend application on S3 bucket
2. Configure an EC2 instance to access Frontend artifacts from S3 bucket and expose it as dynamic web application
3. Configure the S3 to cache the JavaScript build files
4. Configure the Route 53 to register domain name to expose the Frontend solution.
5. Ensure that Privacy Protection feature is enabled for Amazon Route 53 domain.

11 Methodology

11.1 Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
    a. Q/A
    b. New Ideas, New feature implementations and estimation.
    c. Any development related challenges
    d. Skill Gaps
    e. Any other pointers key to UI/UX and Middleware Development