Distributed Systems

(Assignment-II)

Movies.me

Reg No	
Name	rama41222

Table of Contents

Introduction

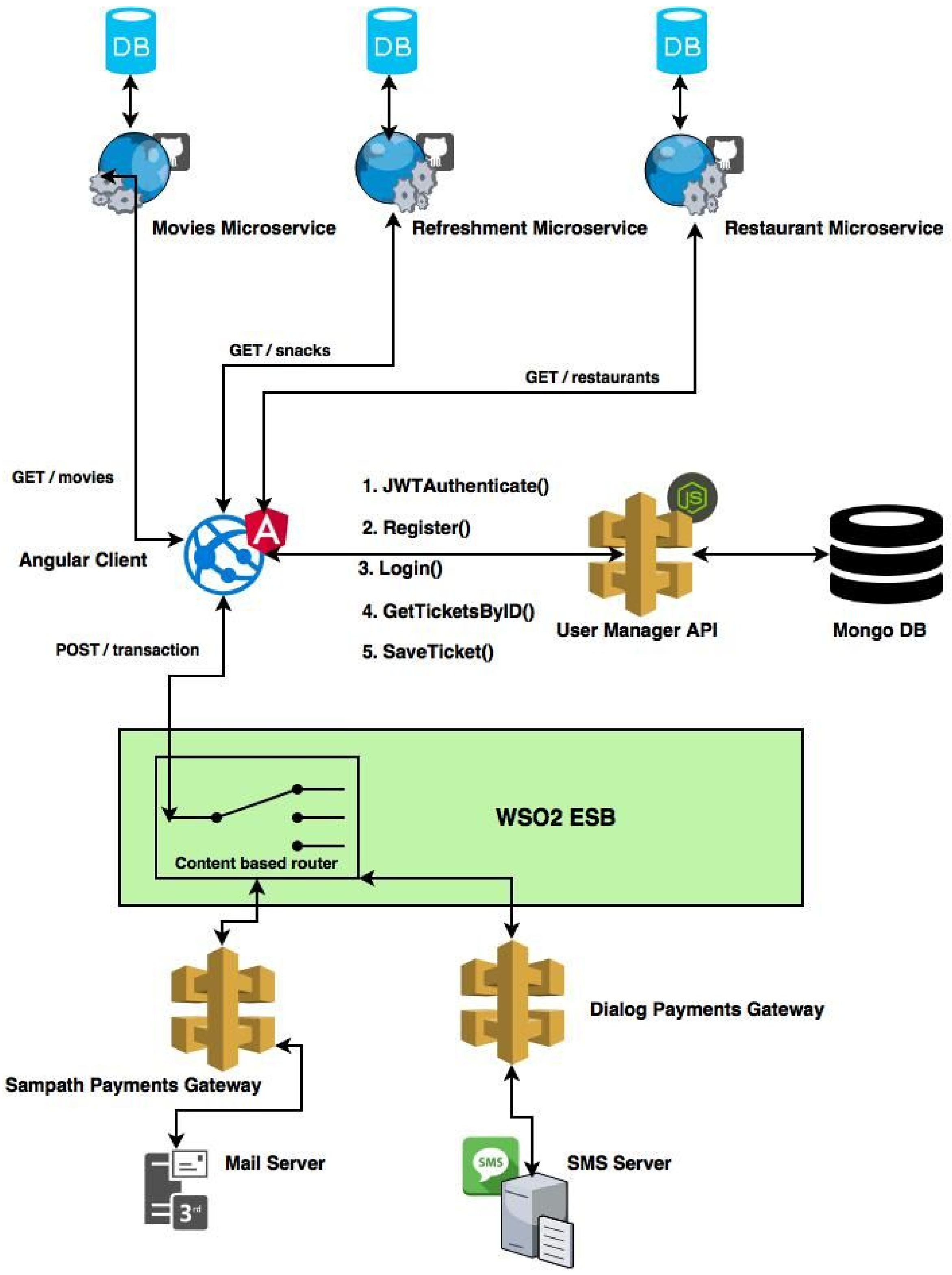
Movies.me is a online movie ticketing system integrated with a refreshment service and a restaurant service where the user can buy snacks during movie time and spend the time in a nearby restaurant.

Movies.me is integrated with the Sampath easypay payments gateway and the Dialog easycash payments gateway. This feature enables the user to buy movie tickets and other services easily.

Client side web application is designed in a way to depart from conventional design and improve the user experience.

Architectural Design

Architecture Diagram



Review

Architectural influence for the Movies.me is taken from both Service Oriented architecture and microservices architecture (In some books it's mentioned that microservices is the "SOA done right!")

1. An ESB is used to deal with POST /transaction route which will be handled automatically by wso2 ESB for content based routing.
2. A User management API was deployed since JWT token management and user management has to be done in a similar location
    a. Shortcomings :
        i. Identity server could be used for JWT token management thereby other microservices can talk to the Identity server before handling a request. (Federated Identity Management).
3. Different Micro Services are used for movies, refreshments and Restaurants.
    a. Usage:
        i. Movies microservices is used to get a list of movies
        ii. Refreshment microservice is used to get a list of available refreshments during the movie.
        iii. Restaurant microservice is used to get a list of **nearby** restaurants
    b. Advantages:
        i. If a microservice fails, it doesn't affect the functionality of the whole system.

Improvements

1. An Api gateway should have been used to integrate the microservices. Therefore all the api calls from the angular client could be routed to microservices through the gateway.
    a. Advantage
        i. Increases the reusability of the code.
        ii. Adherence to SOLID principles.

API Documentation

Movie Service [SpringBoot]

URI	Method	Request	Response

/movies	GET	-	[{
			"id": "m1",
			"name": "John Wick",
			"price": 650,
			"locations": [
			{
			"name": "Savoy 3D",
			"location": "Wellawatte",
			"maxNoOfSeats": 300
			},
			{
			"name": "Sigiri",
			"location": "Katugastota",
			"maxNoOfSeats": 500
			}
			],
			"showtimes": [
			"9AM",
			"1PM",
			"6PM"
			]
			}
			]

/movies/:id	GET	{“id”:”m1”}	{ "id": "m1", "name": "John Wick", "price": 650, "locations": [{ "name": "Savoy 3D", "location": "Wellawatte", "maxNoOfSeats": 300 }, { "name": "Sigiri", "location": "Katugastota", "maxNoOfSeats": 500

			}], "showtimes": ["9AM", "1PM", "6PM"] }

/movies	POST	{
		"id": "m20",
		"name": "John Wick",
		"price": 650,
		"locations": [
		{
		"name": "Savoy 3D",
		"location": "Wellawatte",
		"maxNoOfSeats": 300
		},
		{
		"name": "Sigiri",
		"location": "Katugastota",
		"maxNoOfSeats": 500
		}
		],
		"showtimes": [
		"9AM",
		"1PM",
		"6PM"
		]
		}
			{“msg”:“Success”}

/movies/:id	PUT	{“id”:”m1”} , { "id": "m1", "name": "John Wick", "price": 650,	{“msg”:“Success”}

		"locations": [{ "name": "Savoy 3D", "location": "Wellawatte", "maxNoOfSeats": 300 }, { "name": "Sigiri", "location": "Katugastota", "maxNoOfSeats": 500 }], "showtimes": ["9AM", "1PM", "6PM"] }	
/movies/:id	DELETE	{“id”:”23232433”}	{“msg”:“Success”}

Refreshment Service [SpringBoot]

URI	Method
/snacks	GET
/snacks/:id	GET
/snacks	POST
/snacks/:id	PUT
/snacks/:id	DELETE

Restaurant Service [SpringBoot]

URI	Method
/restaurants	GET
/restaurants/:id	GET
/restaurants	POST
/restaurants/:id	PUT
/restaurants/:id	DELETE

User Management API [NodeJS]

URI	Method	Request	Response
/users	GET		{UserLIst}
/users/:id	GET	{“id”:”23232433”}	{user}
/users	POST		{“mgs:”“Success”}
/users/:id	PUT	{“id”:”23232433”}	{“mgs:”“Success”}

Sampath Gateway Simulator [NodeJS]

URI	Method	Request	Response
/transactions	POST	{ "ccno":"2323232323", "cvv":"232", "amt" : 2000, "email": "dinushankanrg@gmail.com" }	{“msg”:“Success”}

Dialog EasyCash Simulator [Nodejs]

URI	Method	Request	Response
/transactions	POST	{ "mob":"+94711570067","amt": 4000 }	{“msg”:”Success”}

Security Mechanisms

1. JWT token based strategy was used for the Movies.me system as it contains independent self contained services.
2. Client side basic security client has to login by entering username and password.
3. Important routes are protected.
   a. router.get('/:id', **passport.authenticate('jwt',{session:false}),**(req,res)=>{}

Known Issues

1. Since the ESB is originally designed for XML and movies.me is based on JSON there were transformation issues.
2. ESB throws a cross domain issue when the micro services are directly connected to the ESB

References

[1] "WSO2 ESB documentation", https://docs.wso2.com/display/ESB500/WSO2+Enterprise+Service+Bus+Documentation

[2]"Microservices", https://smartbear.com/learn/api-design/what-are-microservices/

[3]"Adopting Microservices at Netflix: Lessons for Architectural Design",https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/