Task 2:

Security and privacy:

System operators would have access to sensitive user information across multiple restaurants which means its important for this data to be secure. To protect this data, encryption should be used on both data at rest and data in transmission, this would mean any data intercepted would be made unreadable. Data at rest can be encrypted using the AES-256 algorithm and for data in transit, protocols such as TLS or HTTPS can be used.

Strict access control should be used in order to protect data and operational controls. This can be done by implementing role based access control, limiting operators access to only what i deemed necessary for their job. MFA should also be used to further reduce the risk of unauthorized access especially for operators who work with sensitive data. This can be done by requiring the employee to present some sort of identification or biometric information to validate their authorization for a certain operation. All operator actions should also be recorded and stored to ensure no one is taking advantage of their privilege and to solve any security incidents.

Customer data in the operation systems must comply with the correct data regulations. Policies must be put in place to specify how long the data should be stored on the system and when it should be deleted securely. Only necessary customer information should be stored in the system, everything else should be deleted.

Performance:

System operators need access to real time information as they have to generate reports and check on different restaurants statuses which requires efficient data processing. A good way to do this is by implementing caching and database indexing so that during rush hours and weekends where the restaurant is very busy, the response time can still be quick. Another way to speed up data loading time is to use optimised SQL queires espcially for the data that is most accessed.

System operators should have real time account management, this means they should be able to access and manage different restaurant accounts dynamically and have updates confirming that their actions have been processed correctly. This can be through realtime notifications displaying the change the action caused such as activating or deactivating a restaurant account. Efficient and well created APIs should be used to allow system operators to access different restaurant accounts and modify them so that information is quickly available to customers.

Resource management is extremely important during peak restaurant times as operators have to perform data heavy tasks while there is already high traffic. This can be done through dynamic resource scaling, using cloud resouces which automatically scale depending on the demand meaning system operators wont experience slow system processing in peak hours.

Reliability:

The operation system needs to be highly available at all times and especially during periods of high traffic, in order for system operators not to be slowed down and able to keep up at any time. This can be done through load balancing, this way the traffic is distributed across different servers to prevent any one server from being overwhelmed due to too much traffiic. Load balancers also monitor the health of the server and fail over to a backup server automatically if anything goes wrong.

The data should be consistent across all systems and operator interfaces. The use of DDS(distrubuted data synchronization) ensures data synchronization across multiple systems so that any updates such as bookings or customer details are reflected immediately on any operator interface.

Effective recovery plans and regular backups would help system operators maintain data integrity. All customer, restaurant and operation data should be backed up daily to avoid any unwanted data loss.

There should also be automated consistency checks across all operator and restaurant accounts in order to detect any mistakes or synchronization quickly. These checks would verify the integrity of the data and if possible fix them automatically or alert a system operator to prevent any operational issues.

Scalability:

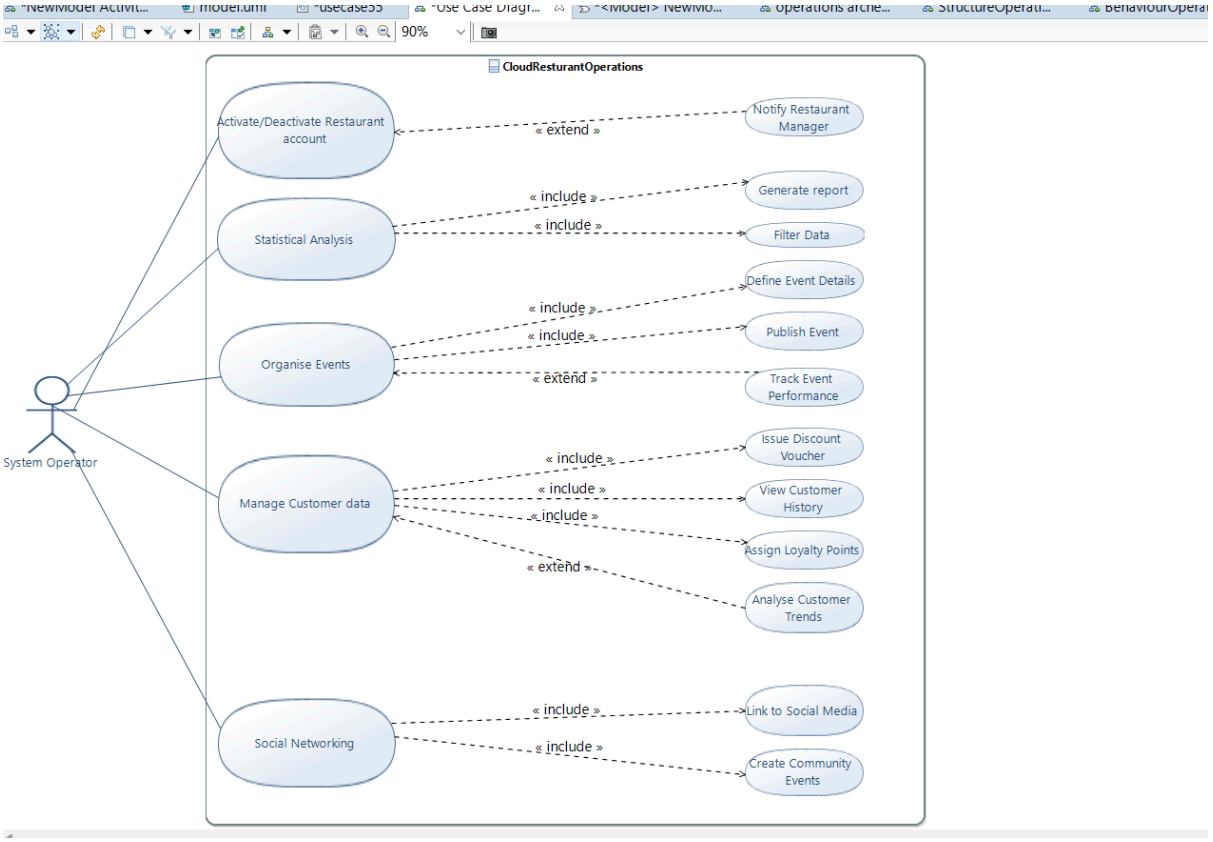
The system can use elastic cloud infrastructure, this would allow the system to automatically change the amount of resources available based on the current state of traffic. This would mean during times of high traffic and peak restaurant hours, the system would increase the amount of resources available to match the need, then scale down automatically when traffic goes down. This would ensure that the operations system stays consistent and responsive at all times.

The use of microservice architecture would also benefit the scalability of the system as it splits up the system into a group of independent services that communicate with each other. This would allow any additions or modifications to be done independently and based on what the system needs. This would also help in the deployment of new services without the need of redeploying the whole system again.

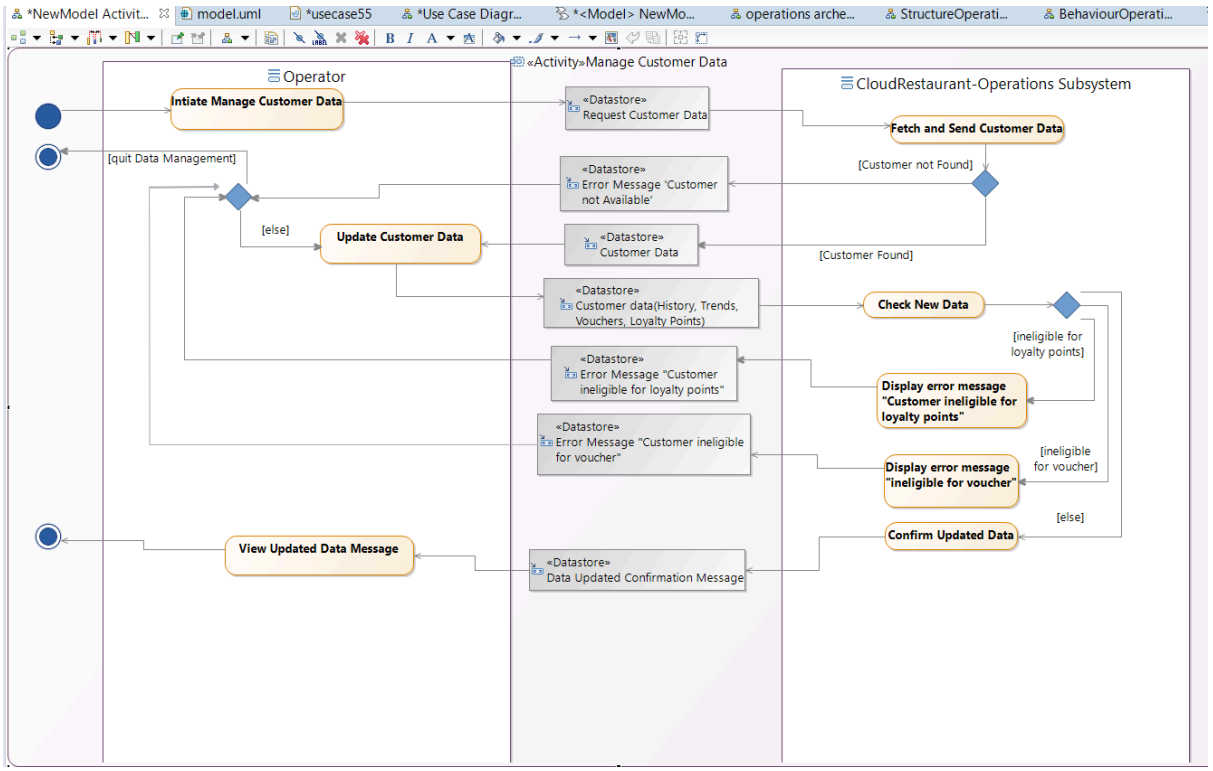
Another way to increase the scalabilty of the operations system is through sharding, this is when a large database would be split into multiple smaller databases which are easier to manage and scale. This would be used to avoid the database getting too large which would cause the response time to be longer as all requests are made to one database. As the requests would be split over different smaller databases this would ensure the system would still have good response times even when many actions are being performed.

Task 3:

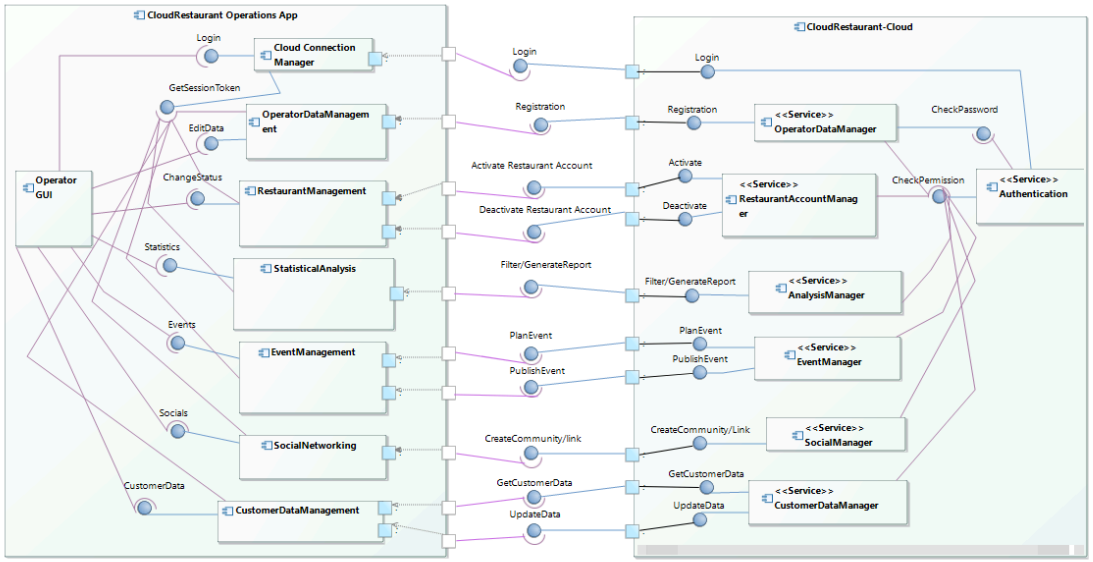
Use Case Diagram:



Activity Diagram:



Architecture Diagram:



Microservice Specification:

Component	OperatorDataManager
Description	This microservice manages operator account data and verification.

Stereo Type	Microservice
Required Interfaces	CheckPermission
Provided Interfaces	Registration

Component	RestaurantAccountManager
Description	This microservice manages restaurant accounts, allowing for activation and deactivation of accounts
Stereo Type	Microservice
Required Interfaces	CheckPermission
Provided Interfaces	Activate, Deactivate

Component	AnalysisManager
Description	This microservice processes statistical data and generates reports for restaurant and customer activities.
Stereo Type	Microservice
Required Interfaces	CheckPermission
Provided Interfaces	Filter/GenerateReport

Component	EventManager
Description	This microservice handles the planning and publishing of promotional events across restaurants
Stereo Type	Microservice
Required Interfaces	CheckPermission
Provided Interfaces	PlanEvent, PublishEvent

Component	SocialManager
Description	This microservice handles customer social networking, including creating communities and providing links to social media applications.

Stereo Type	Microservice
Required Interfaces	CheckPermission
Provided Interfaces	CreateCommunity/Link

Component	CustomerDataManager
Description	This microservice manages customer data, including bookings, orders, feedback and incentives like loyalty points and vouchers
Stereo Type	Microservice
Required Interfaces	CheckPermission
Provided Interfaces	GetCustomerData, UpdateCustomerData

Component	Authentication
Description	This microservice provides authentication and permission for system operators by validating login details.
Stereo Type	Microservice
Required Interfaces	None
Provided Interfaces	Login, CheckPermission

Interface Specification:

Name		Login
Provider		Authentication
Operation	Signature	operatorLogin(op_id: String, password:

		String): sessionKey: String
	Function	Checks if the password entered is correct, if it is then the login is successful and it returns a sessionKey for the operators active session, if its incorrect then an error message is returned.

Name		Login
Provider		Authentication
Operation	Signature	operatorLogin(op_id: String, password: String, ip_Address: ip_Address): accessCode: String
	Function	Checks if the password entered is correct, if it is then the login is successful and it returns an accessCode, if its incorrect then an error message is returned.

Name		Registration
Provider		OperatorDataManager
Operation	Signature	registerOperator(name: String, password: String): registered: Boolean
	Function	Registers a new operator with their details and returns a message, true if successful and an error message if failed.

Name		Login
Provider		Authentication
Operation	Signature	operatorLogin(op_id: String, password: String, ip_Address: ip_Address): accessCode: String
	Function	Checks if the password entered is correct, if it is then the login is successful and it returns an accessCode, if its incorrect then an error message is returned.

Name	Activate

Provider		RestaurantAccountManager
Operation	Signature	activateRestaurantAccount(restaurantId: String): Boolean
	Function	Activates a restaurant account by setting its status to active. Returns true if successful and an error message if failed.

Name		Deactivate
Provider		RestaurantAccountManager
Operation	Signature	deactivateRestaurantAccount(restaurantId: String): Boolean
	Function	Deactivates a restaurant account by setting its status to inactive. Returns true if successful and an error message if failed.

Name		Filter/GenerateReport
Provider		AnalysisManager
Operation	Signature	-Operation 1: filterReport(filterCriteria: String, startDate: Date, endDate: Date): Report[] -Operation 2: generateReport(reportType: String, startDate: Date, endDate: Date): Report
	Function	Function 1: Filters the report based on a filterCriteria such as performance metrics and a date range, returns an array of matching reports Function 2: generates a report based on the specified report type and date range, returning the report or an error message if failed

Name		PlanEvent
Provider		EventManager
Operation	Signature	planEvent(eventName: String, eventDate: Date, eventLocation: String): Boolean

	Function	Plans a new event with the provided name, date and location, returning true if successful or an error message if failed

Name		PublishEvent
Provider		EventManager
Operation	Signature	publishEvent(eventId: String): Boolean
	Function	Publishes the event identified by its eventID, allowing it to be visible across all restaurants. If the event is published successfully it returns true otherwise it returns a error message

Name		CreateCommunity/Link
Provider		SocialManager
Operation	Signature	createCommunityLink(communityName: String, description: String): String
	Function	Creates a community for the restaurant customers with the specified name and description, returns the link for the community or returns an error message if the creation failed

Name		GetCustomerData
Provider		CustomerDataManager
Operation	Signature	getCustomerData(customerID: String): String[]
	Function	Gets the details for the specified customer such as their history, loyalty points and feedback. Returns the customer data or an error message

Name	UpdateData
Provider	CustomerDataManager

Operation	Signature	updateCustomerData(customerID: String, dataType: String, newValue: String): Boolean
	Function	Updates the selected customer data depending on the specified dataType. Returns true if the update was successful or an error message if failed

Name		CheckPermission
Provider		Authentication
Operation	Signature	checkPermission(op_ID: String, action: String): permissionStatus: Boolean
	Function	Checks if the operator has permission to perform a specific action. Returns true if permission is granted and an error message of false.

Database Specification:

Database Name	Operator Database
Description	This database stores all system operator information and their permissions for managing the restaurant operations system
Type	Relational database
Design(tables)	-Operators -permissions
Table(Fields): Operators	operator_ID, Operator_Name, Operator_Email, password, status
Table(Fields): Permissions	Permission_ID, operatorI_D, module, Access_Level

Database Name	Restaurant Database
Description	This database stores the restaurants information, accounts and its status
Type	Relational database

Design(tables)	-Restaurants -RestaurantAccounts
Table(Fields): Restaurants	restaurantID, name, location ,status
Table(Fields): Permissions	Account_ID, Restuarant_ID, Manager_ID, Creation_Date

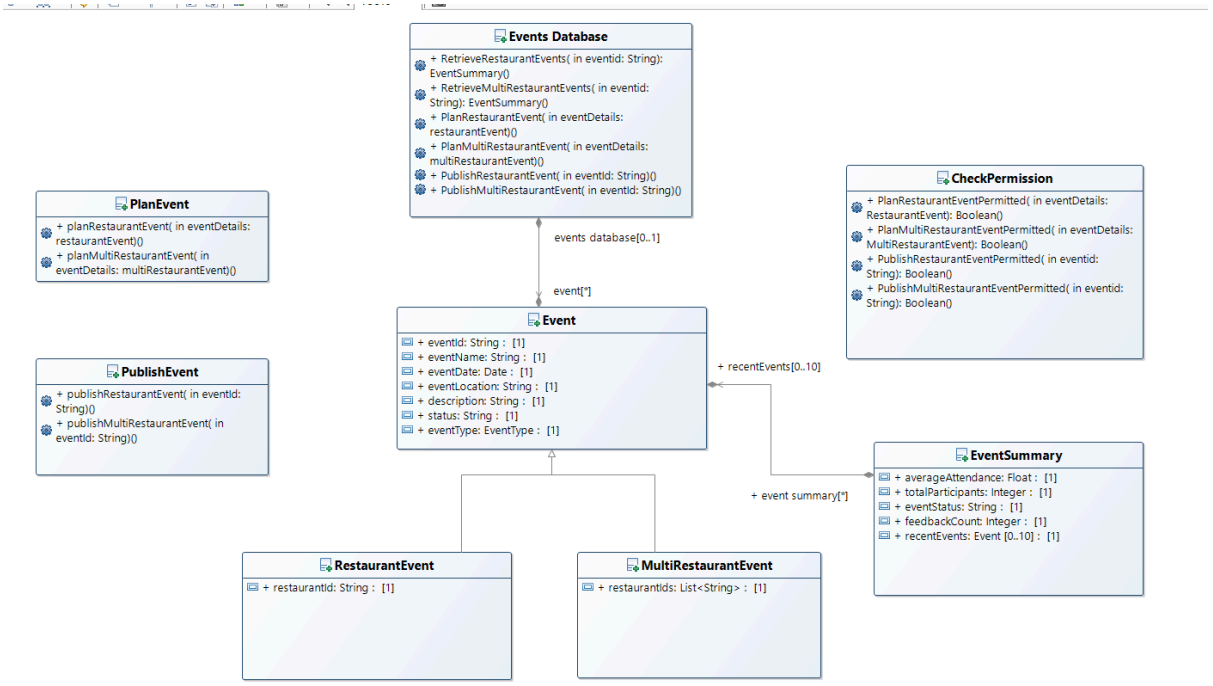
Database Name	Reports Database
Description	This database stores reports generated by system operators, including statistical and analytical information.
Type	Relational database
Design(tables)	-Reports
Table(Fields): Restaurants	report_ID, Report_Type, Generated_By, Start_Date, End_Date, content

Database Name	Events Database
Description	This database stores information about promotional events planned and managed by system operators.
Type	Relational database
Design(tables)	-Events
Table(Fields): Events	event_ID, Event_Name, Event_Description, Date, Time, Location, Created_By, Status

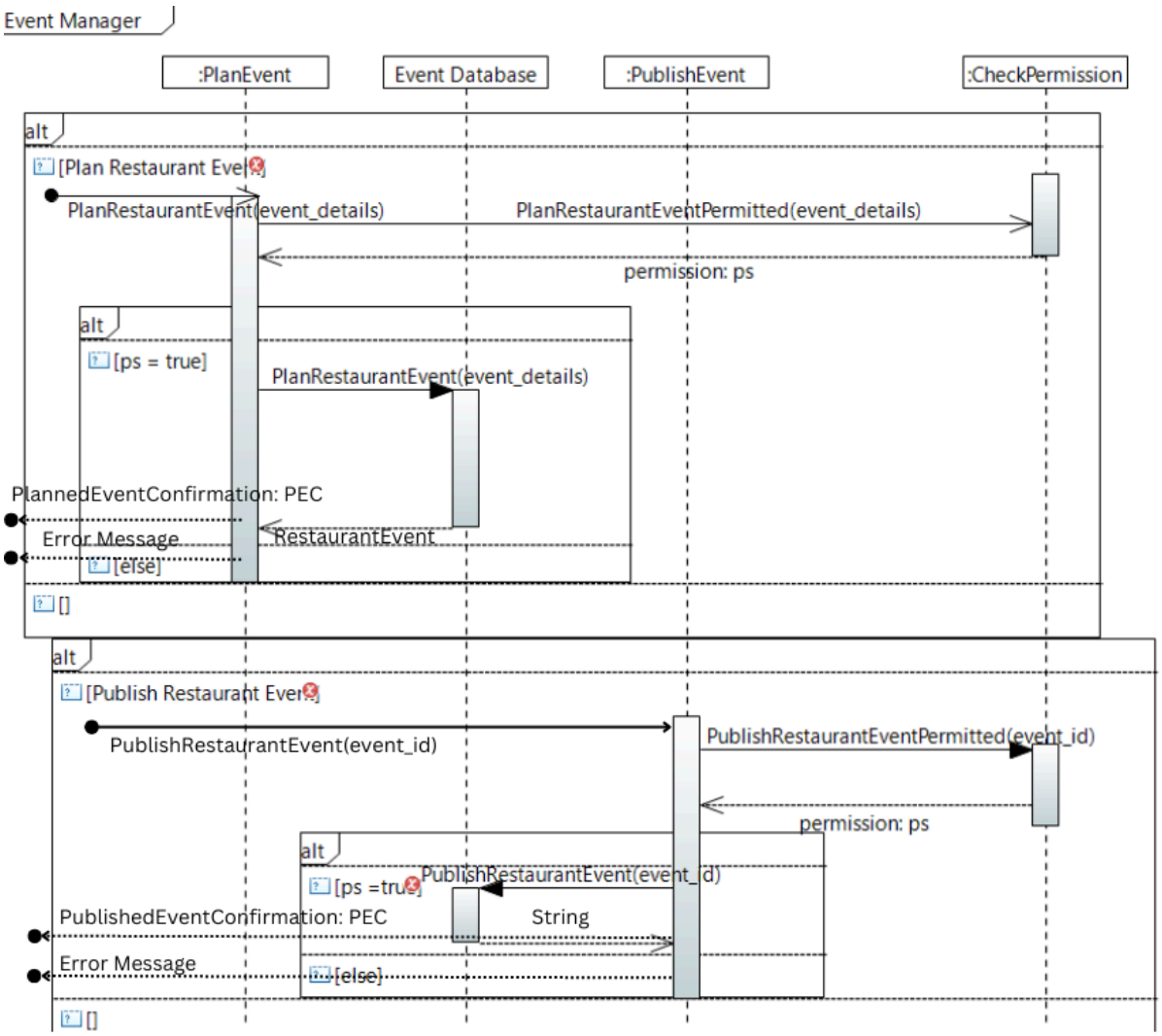
Database Name	Socials Database
Description	This database manages information about communities and social links created for customer engagement.
Type	Relational database
Design(tables)	-Socials
Table(Fields): Socials	Community_ID, Community_Name, Description, Link, Created_By

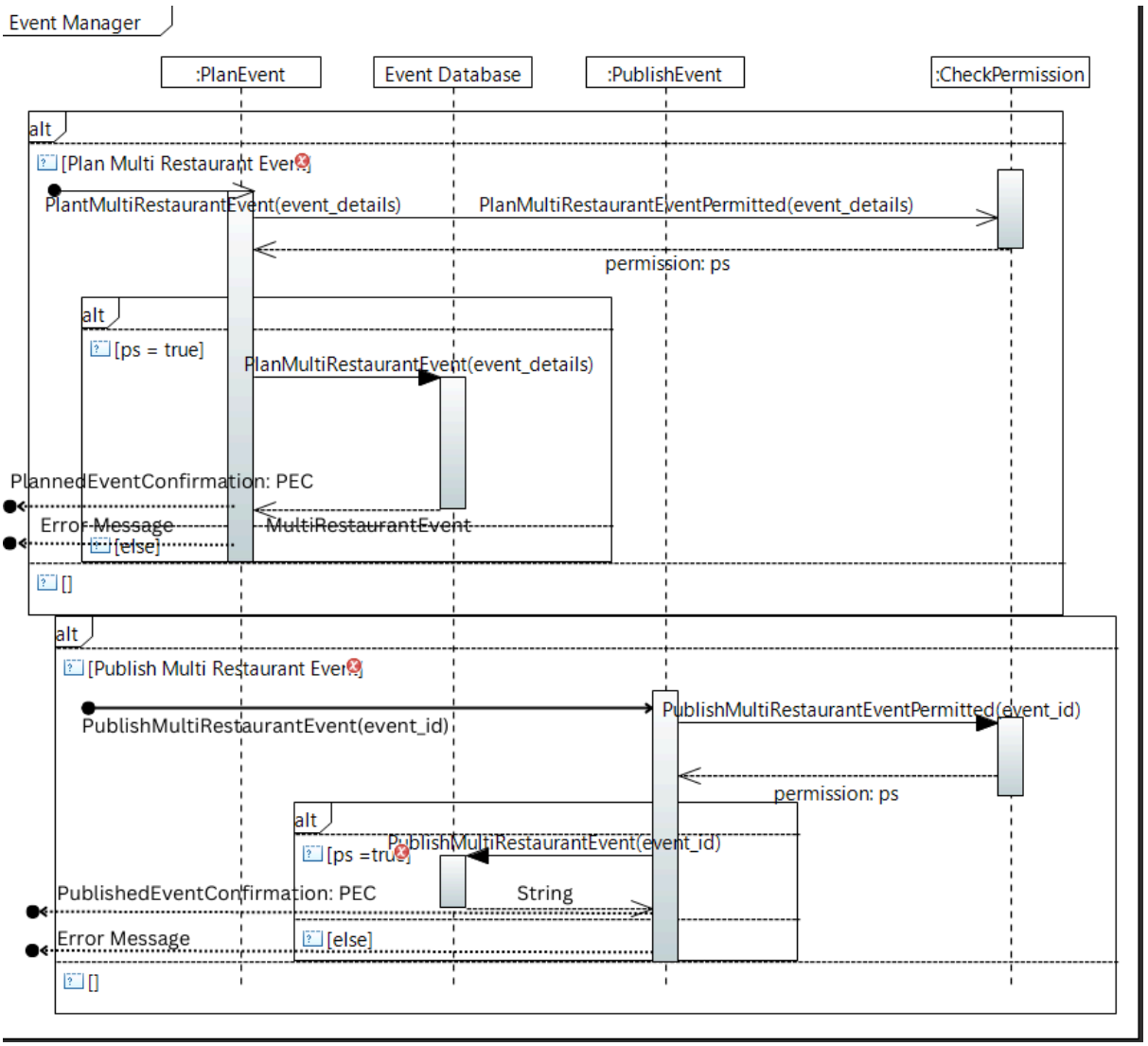
Database Name	Customers Database
Description	This database manages customer information and their activities (bookings, orders, feedback, and vouchers)
Type	Relational database
Design(tables)	-Customers -Activities
Table(Fields): Customers	Customer_ID, Customer_Name, Customer_Email, Customer_Phone, Loyalty_Points
Table(Fields): Activities	Activity_ID, Customer_ID, Activity_Type(Booking, Order, Feedback, Vouchers), Activity_Details, Activity_Date

Structure Diagram:



Behaviour Diagram:


Event Manager
:PlanEvent
Event Database
:PublishEvent
:CheckPermission
alt
[Plan Restaurant Event]
PlanRestaurantEvent(event_details)
PlanRestaurantEventPermitted(event_details)
permission: ps
alt
[ps = true]
PlanRestaurantEvent(event_details)
PlannedEventConfirmation: PEC
Error Message
RestaurantEvent
[else]
[]
alt
[Publish Restaurant Event]
PublishRestaurantEvent(event_id)
PublishRestaurantEventPermitted(event_id)
permission: ps
alt
[ps =true]
PublishRestaurantEvent(event_id)
PublishedEventConfirmation: PEC
String
Error Message
[else]
[]



Github Repository:

https://github.com/OsamaKhaled04/comp5047