THIS FORM IS FOR BOTH THE GENERAL & MEDICAL INFORMATICS PROGRAMMES

SE-I COURSE PROJECT (COVER SHEET)

Discussions Scheduled for Week 13 *(Thursday, May 9th, 2024)*.

- Print 1 copy of this cover sheet and attach it to a printed copy of the documentation *(SRS, … etc.)*. You must also submit softcopies of all your documents *(as PDFs)*; details will be announced later.
- Please write all your names in Arabic.
- Please make sure that your students' IDs are correct.
- Handwritten Signatures for the attendance of all team members should be filled in before the discussion.
- Please attend the discussion on time *(announced separately)*; late teams will lose 5 grades.

Project Name: _____

Team Information ***(typed, not handwritten, except for the attendance signature)*:**

	ID [Ordered by ID]	Full Name [In Arabic]	Attendance [Handwritten Signature]	Final Grade
1	20220309	عمر احمد الرفاعي طليس		
2	20220320	عمر عبد الحميد احمد حسين		
3	20220016	أحمد حسام الدين حمدي صالح		
4	20220484	مصطفي عتريس عبدالكريم		
5	20220299	علي محمد جمعه ذكي		
6	20220292	عصام محمد اسماعيل دياب		

Grading Criteria:

14 Items		Grade	Notes
1. Functional Requirements	**1**		
2. Non-Functional Requirements	**1**		
3. Use-Case Diagram(s) *including general use-cases for the system and the detailed use-cases description*	**2.5**		
4. System Architecture – *including applied Architectural Pattern(s)*	**1**		
5. Activity Diagrams	**2**		
6. Object Diagrams *(Including object diagrams that illustrate the preconditions and the post-conditions of selected functions)*	**2**		
7. Package Diagram(s)	**1**		
8. Sequence Diagrams *including* ***System Sequence Diagrams (SSDs)***	**2.5**		
9. Database Specification *(ERD, Tables)*	**2**		
10. Collaboration/Communication Diagrams	**2**		
11. Class Diagram *(3 versions)* **1) An initial version based on the requirements and Use-Case/Activity diagrams.** **2) An intermediate version based on the interaction diagrams.** **3) A final version after applying the design patterns and other modifications.**	**6**		
12. Three Mandatory Design Patterns Applied *(Including a typed description)*	**3**		
13. Front End Design for all Functions *(HTML, Bootstrap)*	**2**		

14. Implementation based on the submitted Requirements & Design. Should include at least 4 of the following modules (in addition, of course, to modules specific to your projects): **1) User Role Management Module.** **2) User manipulation Module** *(Login, Add / Delete / Update / Search, List).* **3) Controlling Resources Module** *(Rooms, Orders, Products, … etc.).* **4) Reservation and Rescheduling Module.** **5) Generating Reports Module** *(PDFs, … etc.).* **6) Sending Emails or Notifications Module.**	7		

Teaching-Assistant's Signature: _____

_____
35

Helwan University

Faculty of Computers & Artificial Intelligence

Module: CS251 Software Engineering 1

Spring "Semester 2" 2023-2024

_____Introduction_____

(A Museum Web App)

A. Our Purpose:

A Digital System for One of the most ancient Museums in The Middle East (Helwan Museum).

Our Major Purpose is to make Our Clients Doing the Booking Process more easily and in a short time and Highly Recommend System for them to choose the approbative visit and event for them.

B. Project Scope: (Visits, Events, Collection, Learn, Membership)

- Visit: This section serves as a personal planner for users, allowing them to schedule and reserve various types of visits. Whether it's a solo trip or a group tour, the app simplifies the process of organizing a visit to the museum.

- Exhibitions and Events: Acting as a virtual guide, this part of the app provides detailed information about all current and upcoming exhibitions and events. Users can effortlessly book their attendance and manage their reservations for specific events. Additionally, the app curates event recommendations tailored to the interests of its members.
- Collection: A digital treasure trove, the Collection section houses online exhibits, galleries, and blogs. Users can navigate through this rich content by filtering via categories, topics, and other criteria, making it easier to find areas of interest.
- Learn: Dedicated to educational services, this segment targets various groups such as schools, young individuals, and community organizations. It offers learning opportunities and resources to engage with the museum's offerings in an educational context.
- Membership: This feature streamlines all aspects of museum memberships, from sign-ups to renewals. It provides members with easy access to their account details and exclusive content.

C. 1: Abbreviations

- **UI**: User Interface
- **UX**: User Experience
- **CMS**: Content Management System
  -

2. Glossary

- **Code of Ethics**: A set of principles and standards of conduct that guide the behaviour of museum staff and volunteers.
- **Collections Plan**: A strategic document that outlines the goals and methods for developing and managing the museum's collection.
- **Interactive Maps**: Digital maps provided within the app to help users navigate the museum space.
- **Membership Management**: The system within the app that handles all aspects of user memberships, including sign-ups, renewals, and benefits.

D. List of the System Stakeholders.

1. The visitor (End User)

2. The System

E) References.

Our References: https://www.britishmuseum.org/

MUSEUM WEP APP REQUIREMENTS

1. *General: *

- **Contact us** (The Visitor can contact the museum for any help or a compliment)

- **About us** (The visitor can see the museum's history and all the information about it)

- **Opening hours (**The visitor can check the Opening hours of the museum)

- **Manage user account** (The Visitor can see his information and manage the profile by changing the username, email and password)

- **User feedback** (the visitor can provide feedback about any topic in the museum)

- **Log in** (The visitor can access his account and log in to the website)

- **Sign out** (The visitor can Sign out of the website)

- **Register** (The visitor can register his info at the website)

- **Checking The museum rules** (The Visitor can see the Museum rules)

- **See the museum map** (The visitor can Check the museum map with a Legend on it)

2. *Visit Requirements: *

- **Book a visitation ticket** (The visitor can book a ticket to visit the museum the ticket can be Economical, Educational or Research)

- **Choose the date and time of the visit** (The visitor can choose the date and the time of his visitation)

- **Pay with various options** (The visitor can choose which way to pay for his ticket. Can be Cash or Master card)

- **Getting a confirmation** (The visitor Gets a Confirmation on his payment the confirmation can be a Notification, Email or Instant feedback)

- **Hire a guide or an assistant** (The visitor can hire a guide to help him with his visitation)

3. Exhibition and Events:

- **Book an event**: Visitors can reserve spots for upcoming events or exhibitions hosted by the museum.

- Manage the booking: Visitors have the flexibility to modify or cancel their event bookings as needed.

- **Guide manual**: Provides detailed instructions or guidelines for attendees regarding event rules and information.

- **Seeing the available Events**: Visitors can browse through a list of upcoming events and exhibitions.

- **Recommendations for the relevant Events for the members**: Members receive personalized recommendations for events based on their interests and preferences.

4. Collection:

- **Online collections and galleries**: Visitors can explore the museum's collection through an online platform, viewing galleries and individual artifacts.

- **Searching by category**: Allows users to search for specific artifacts or collections based on categories such as time period, culture, or type of artifact.

- **Add to favorites**: Users can mark their favorite artifacts or collections for easy access later.

- **Download images from the site**: Provides the option to download images of artifacts for personal or educational use.

5. Learn:

- **Provide Educational resources for students**: Offers educational materials such as articles, videos, and interactive content to enhance students' learning experiences.

- **Include activities and quizzes**: Engages visitors with interactive activities and quizzes related to museum content, promoting active learning.

- **Provide useful links to learn about the Civilizations**: Offers curated links to external resources for further exploration of the civilizations represented in the museum's collection.

6. Membership:

- **See all available packages and their benefits**: Displays information about different membership packages offered by the museum, along with the perks and benefits associated with each.

- **Manage user Membership**: Allows members to handle various membership-related tasks such as registration, renewal, and cancellation.

- **Show recommendations based on the Membership type**: Offers personalized recommendations for events, exhibitions, or additional benefits based on the member's subscription level.

7. Additional facilities:

- **Display information about additional facilities**: Provides details about other amenities available on-site, such as libraries, archives, and cafes.

- **Allow users to reserve study rooms or private spaces**: Enables visitors to book study rooms or private areas for research or meetings.

- **Enable browsing of menus and placing orders for cafes and restaurants**: Allows users to view menus, place orders, and make reservations for dining options within the museum.

- **Provide accessibility** information for each facility: Offers guidance on how to access and utilize various facilities, ensuring inclusivity and convenience for all visitors.

Non-Functional Requirements

1. **Performance: **

- The system should respond to user interactions within 2 seconds under normal load conditions.

- It should be able to handle peak loads during special events without significant degradation in performance.

2. **Scalability: **

- The system should be designed to handle an increasing number of concurrent users as the museum's popularity grows.

- It should support horizontal scaling to add more resources seamlessly.

3. **Reliability: **

- The system should have a high level of availability, aiming for at least 99.9% uptime.

- It should include mechanisms for automatic failover and recovery in case of system failures.

4. **Security: **

- User data should be stored securely and encrypted to protect against unauthorized access.

- The system should enforce proper authentication and authorization mechanisms to ensure that only authorized users can access sensitive information or perform privileged actions.

- Payment transactions should adhere to industry-standard security protocols to prevent fraud.

5. **Usability: **

- The user interface should be intuitive and easy to navigate for users of all technical levels.

- Accessibility features should be implemented to accommodate users with disabilities.

6. **Compatibility: **

- The system should be compatible with a wide range of devices and web browsers to ensure a seamless user experience.

- It should support multiple languages to cater to a diverse audience.

7. **Maintainability: **

- The system should be designed with clean, modular code to facilitate ease of maintenance and future enhancements.

- Proper documentation should be provided for developers and administrators to understand the system architecture and configuration.

8. **Performance Monitoring and Logging: **

- The system should include monitoring tools to track performance metrics and identify potential bottlenecks or issues.

- Comprehensive logging should be implemented to record user actions, system events, and errors for troubleshooting and auditing purposes.

9. **Backup and Disaster Recovery: **

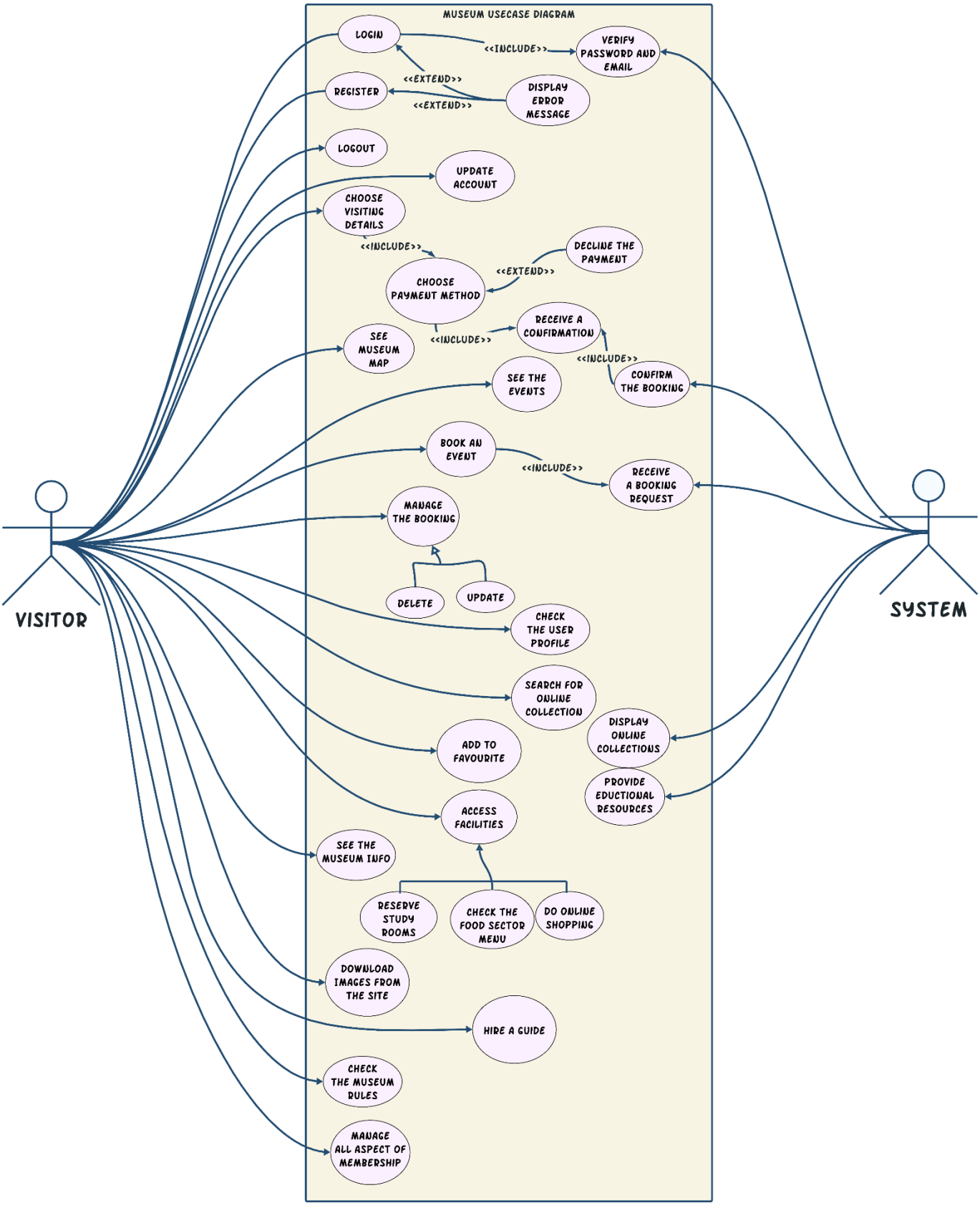
- Regular backups of the system's data should be performed to prevent data loss in case of hardware failures, cyberattacks, or other disasters.

- A disaster recovery plan should be in place to quickly restore the system to a functional state in the event of a major outage.

PART 2: System Design & Models

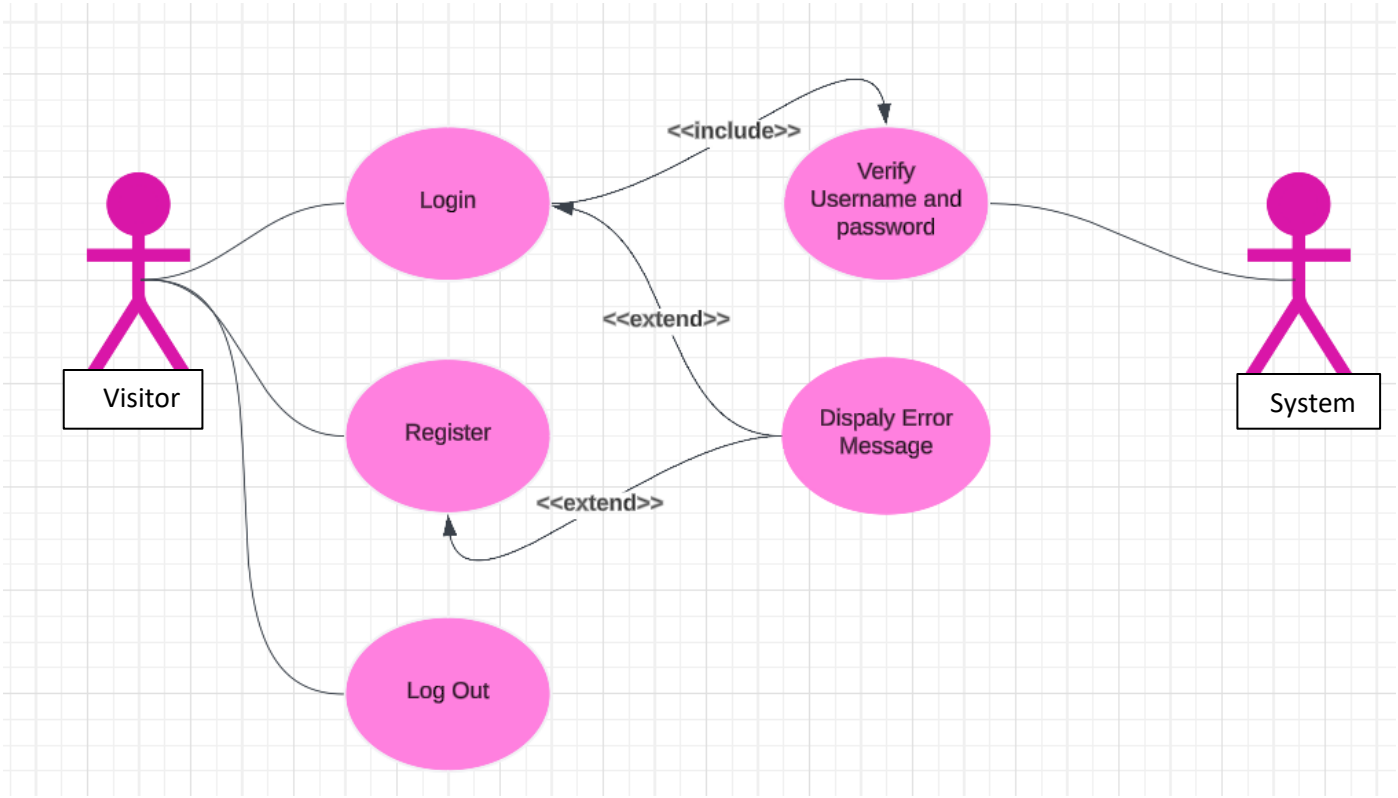
Functional Diagrams:

A. Use-Case Diagram(s)



b) Detailed Use-Cases Description

1:



Initiator: customer

Goals: Visitor login on the site

Main success scenario: the customer could login at the form of the site

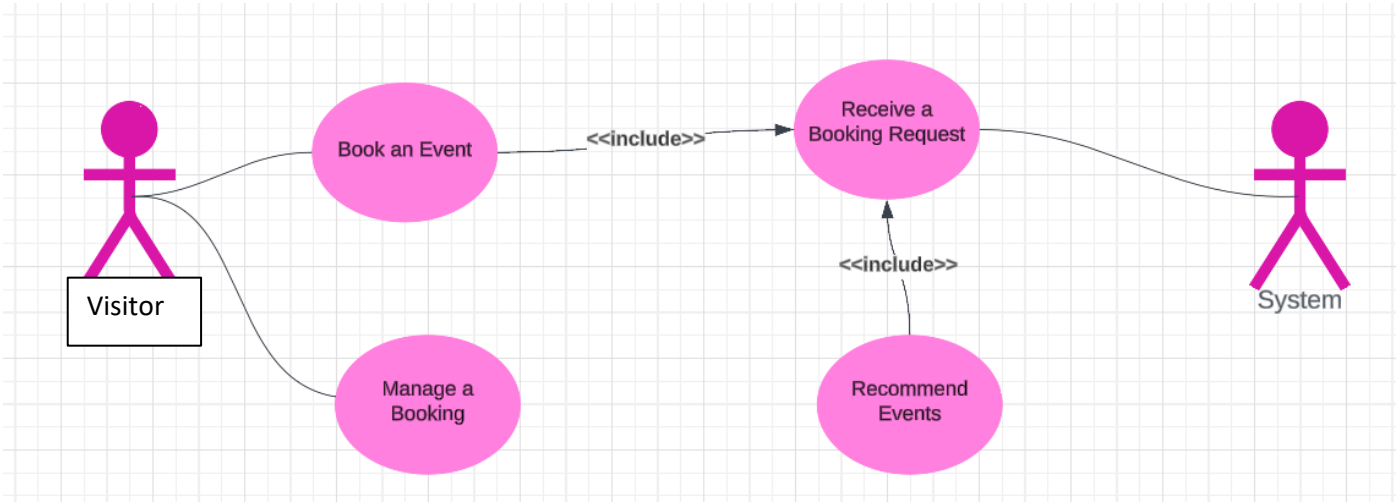
Unsuccessful scenario: the customer can't login on the site because he had forgotten his login password, or the system is not working

Precondition(s): The Form of login is ready to take information

Postcondition(s): the Visitor is login the form and registered

_____-

2:



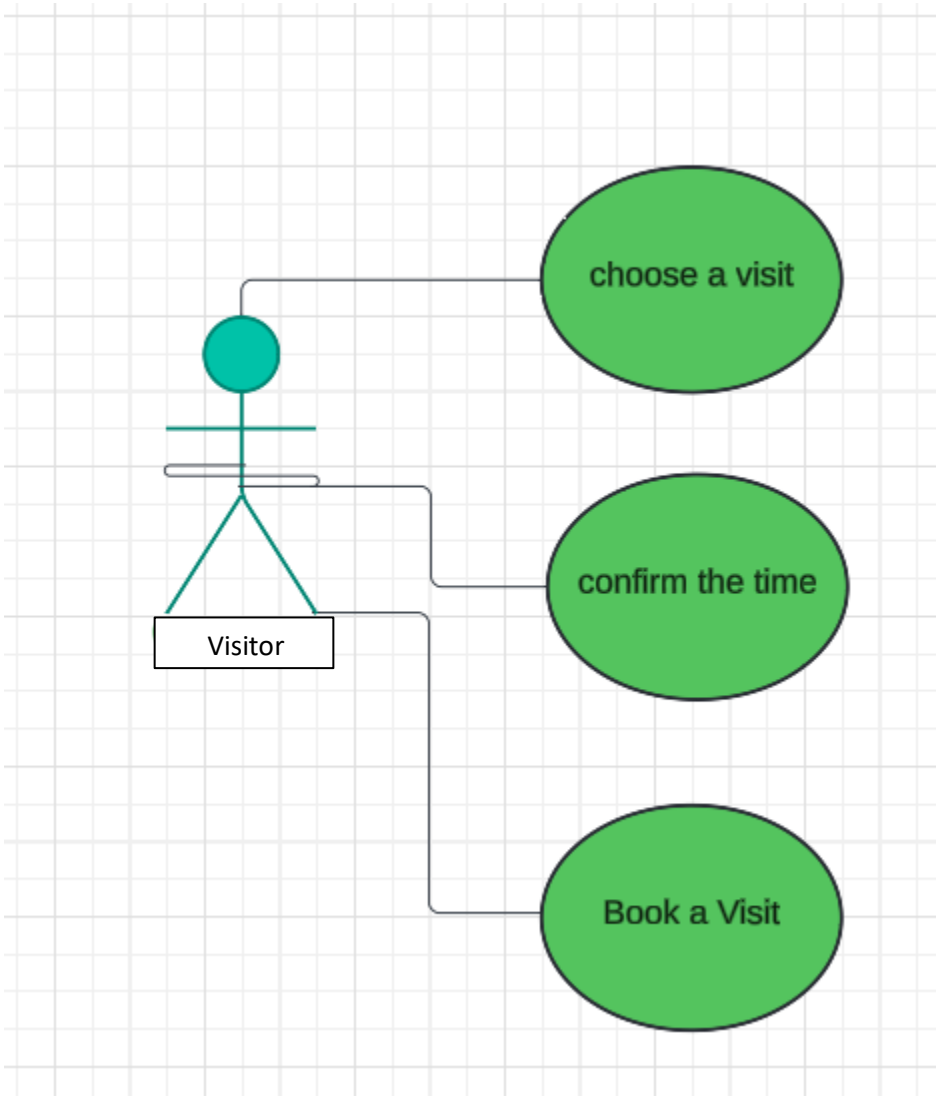
Initiator: customer

Goals: Customer book an event on the site

Main success scenario: the customer could book an event at the form of events of the site

Unsuccessful scenario: the customer can't book at the form on the site because he has chosen an available event at this time or the number of visitors is completed.

Precondition(s): The Visitor is login before
Postcondition(s): the Visitor has booked the event he has chosen.



3.

Initiator: customer

Goals: Visitors book a visit on the site

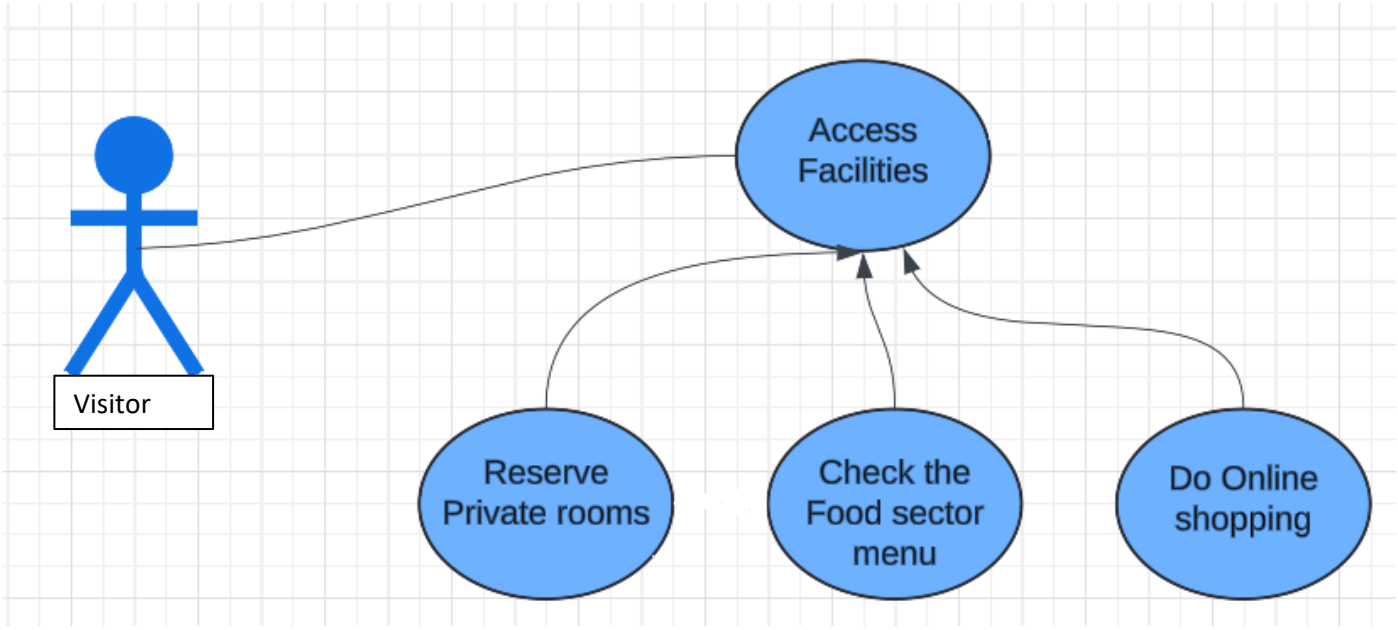
Main success scenario: the customer could book a visit at the form of events of the site

Unsuccessful scenario: the customer can't book at the form on the site because he has chosen an available visit, or the number of visitors is completed.

Precondition(s): The Visitor is login before
Postcondition(s): the Visitor has booked the event he has chosen.

---

4 .



Initiator: customer

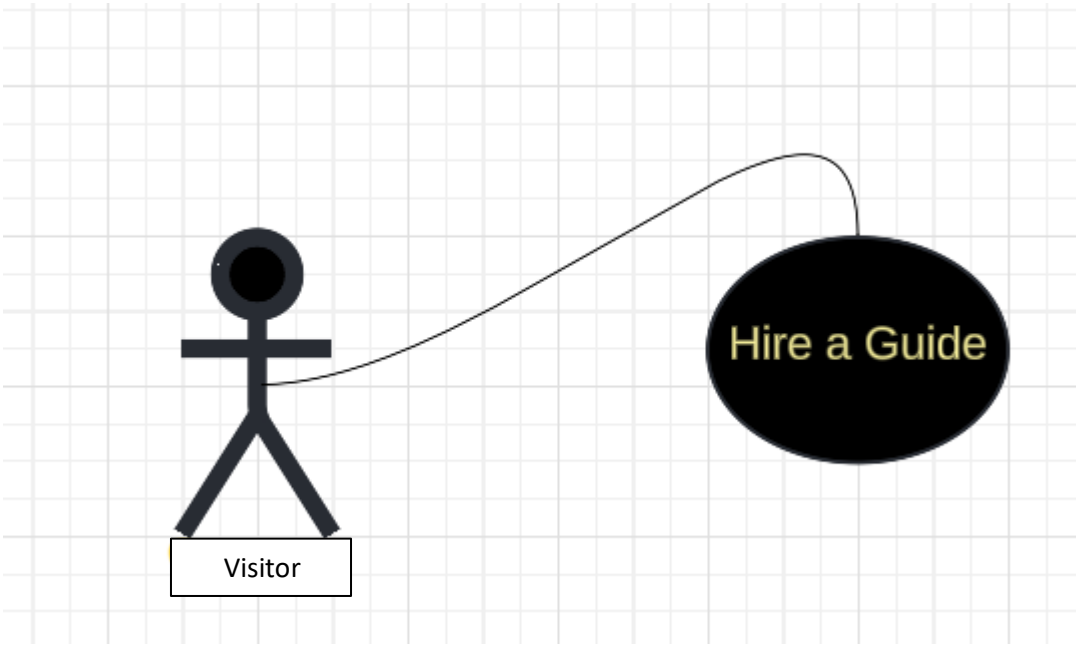
Goals: Visitors access facilities on the site

Main success scenario: the Visitor could access facilities of the site

Unsuccessful scenario: the Visitor can't access facilities on the site

Precondition(s): The Visitor is login before

Postcondition(s): the Visitor has booked the event he has chosen.



5 .

Initiator: customer

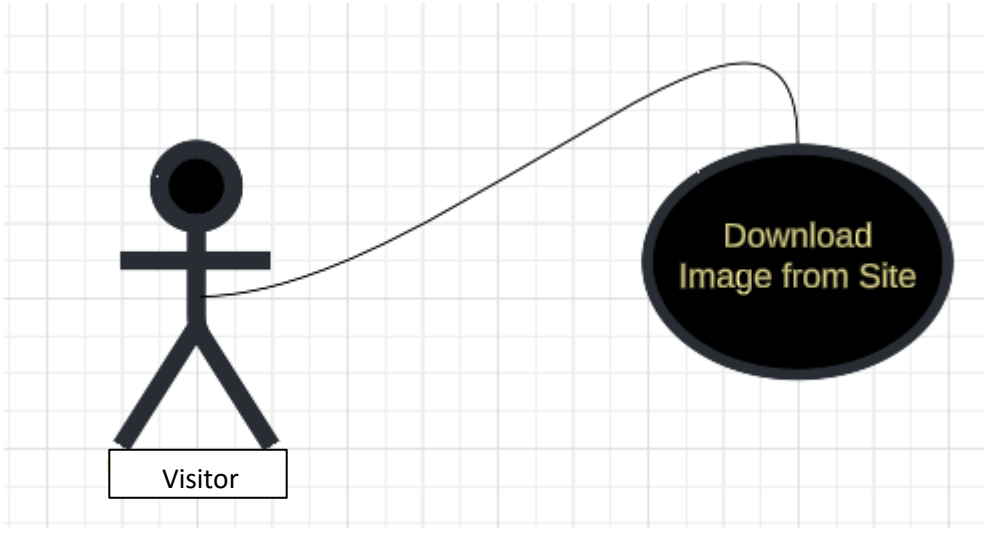
Goals: Customers hire a guide on the site

Main success scenario: the customer could hire an available guide of the site

Unsuccessful scenario: the customer can't hire a guide on the site because no available guide at this time or the number of guides are completed .

Precondition(s): The Visitor is login before
Postcondition(s): the Visitor has booked the Visit he has chosen.

6.



Initiator: customer

Goals: Customers Download image from the site

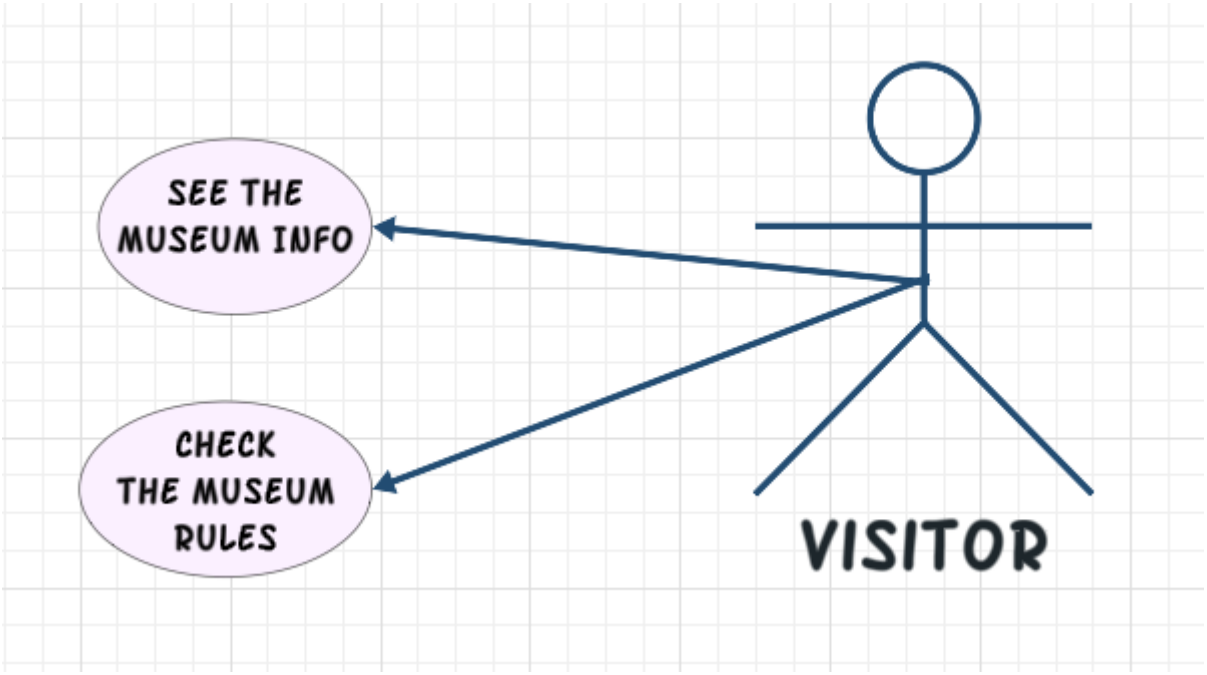
Main success scenario: the customer could Customers Download image from the site

Unsuccessful scenario: the customer can't Customers Download image from the site

Precondition(s): The Customer is login before

Postcondition(s): the customer has Downloaded image from.

7.



Initiator: customer

Goals: Visitors see the museum info and rules

Main success scenario: the Visitor could see the museum info and rules

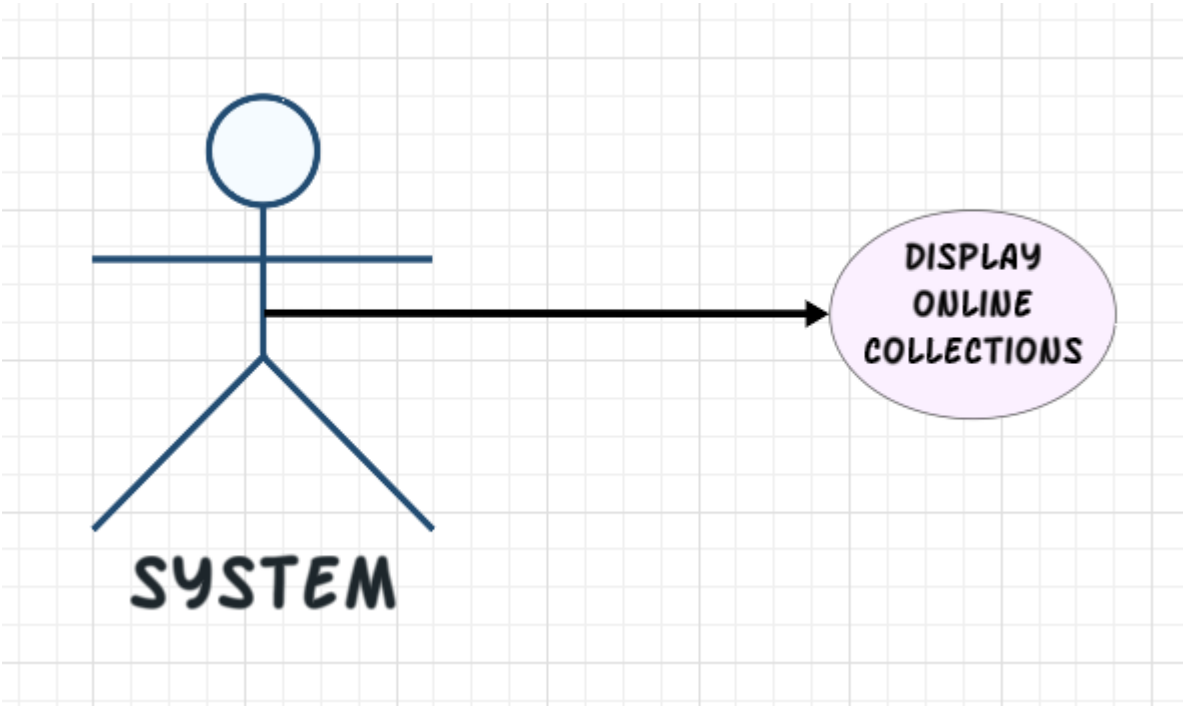
Unsuccessful scenario: the Visitor can't see the museum info and rules

Precondition(s): The Visitor is login before

Postcondition(s): the Visitor has seen the museum info and rules

---

8.



Initiator: System

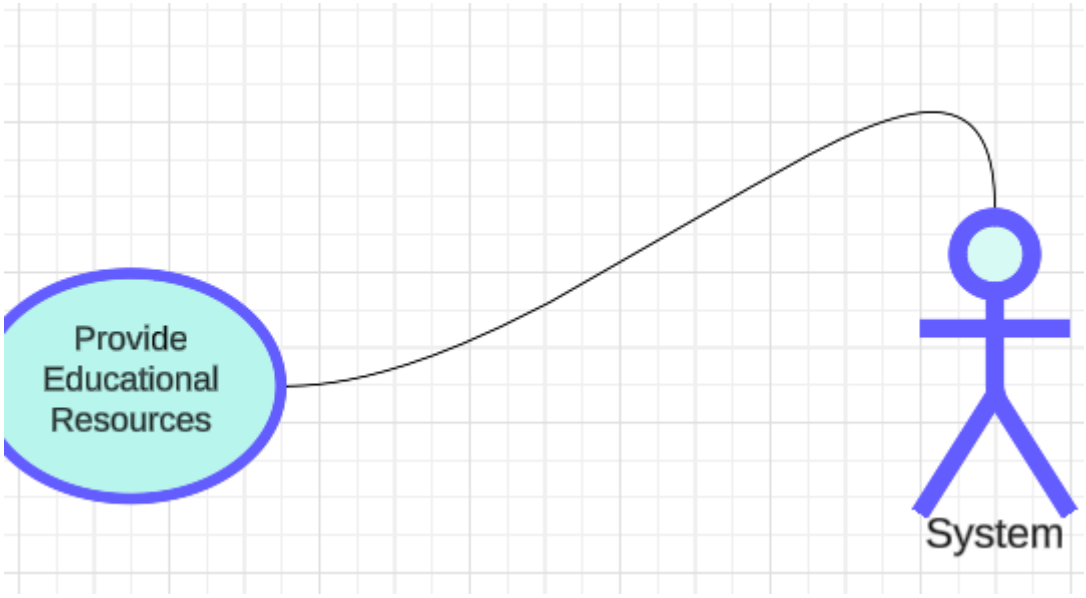
Goals: System Display online collection.

Main success scenario: the System can Display online collection.

Unsuccessful scenario: the Visitor can't see online collection.

Precondition(s): The Database is available

Postcondition(s): The Database Collection is Displayed



9.

Initiator: System

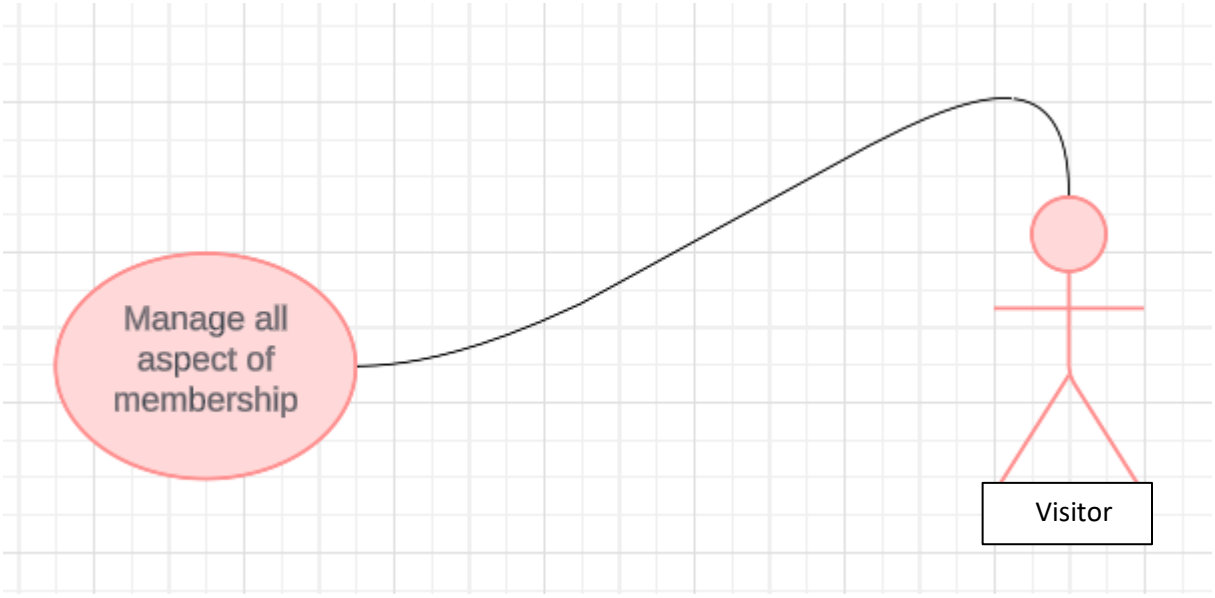
Goals: System provide Educational Resources

Main success scenario: the System can provide Educational Resources

Unsuccessful scenario: the customer can't provide Educational Resources

Precondition(s): The Database is available

Postcondition(s): The Database Resources is Displayed



10.

Initiator: Visitor

Goals: Visitors manage all aspect of membership

Main success scenario: the Visitor can manage all aspect of membership

Unsuccessful scenario: the Visitor can't manage all aspect of membership

Precondition(s): The membership packages are available

Postcondition(s): The Vistor can be able to manage all the aspects of the membership

---

4. System Architecture – *including applied Architectural Pattern(s)*

The architecture pattern that are we used is Model-View-Controller because:

1. Simultaneous development: Multiple developers can work simultaneously on the model, controller and views.

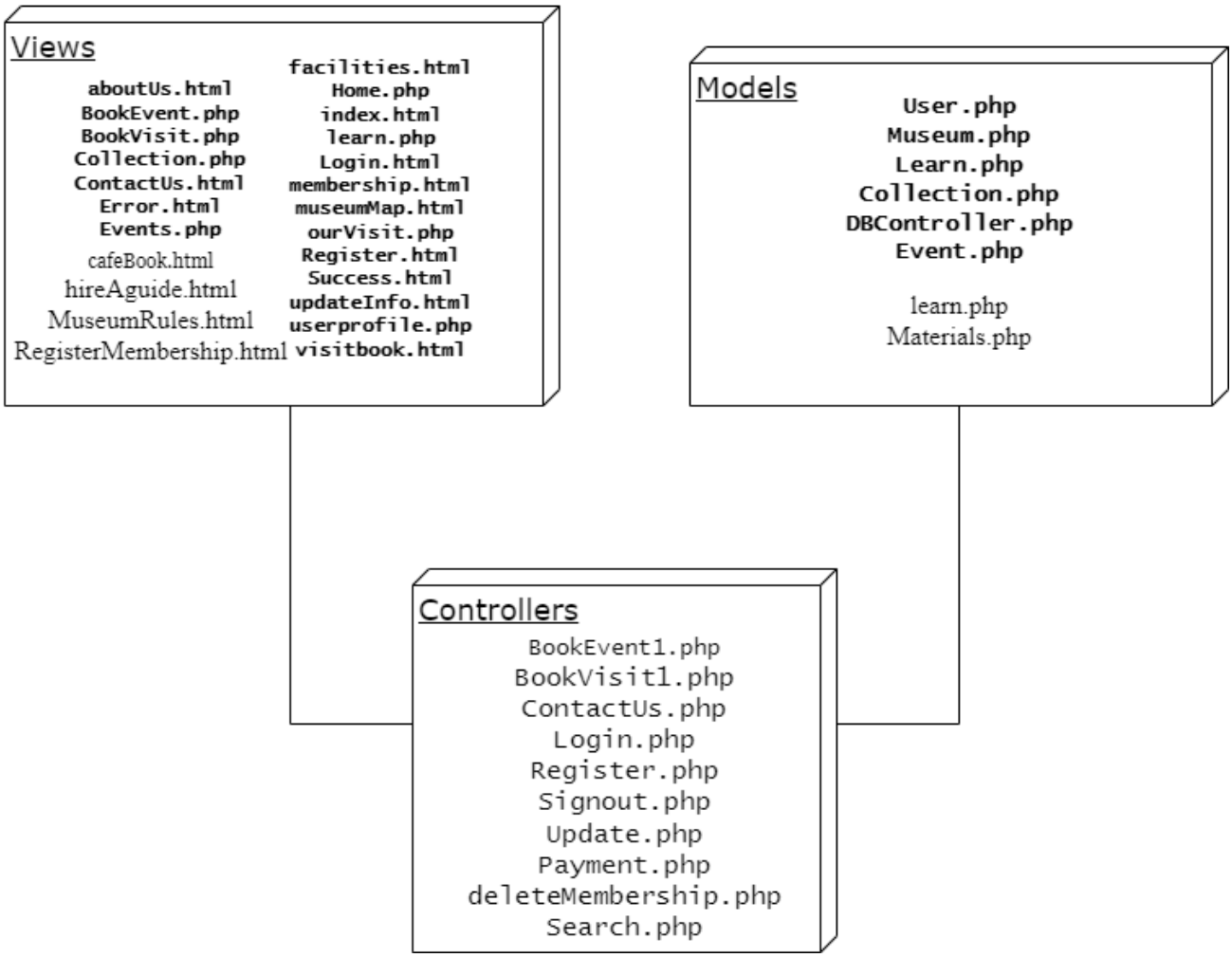
2. High cohesion: MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together. 3. Loose coupling: The very nature of the MVC framework is such that there is low coupling among models, views or controllers

4. Ease of modification: Because of the separation of responsibilities, future development or modification is easier

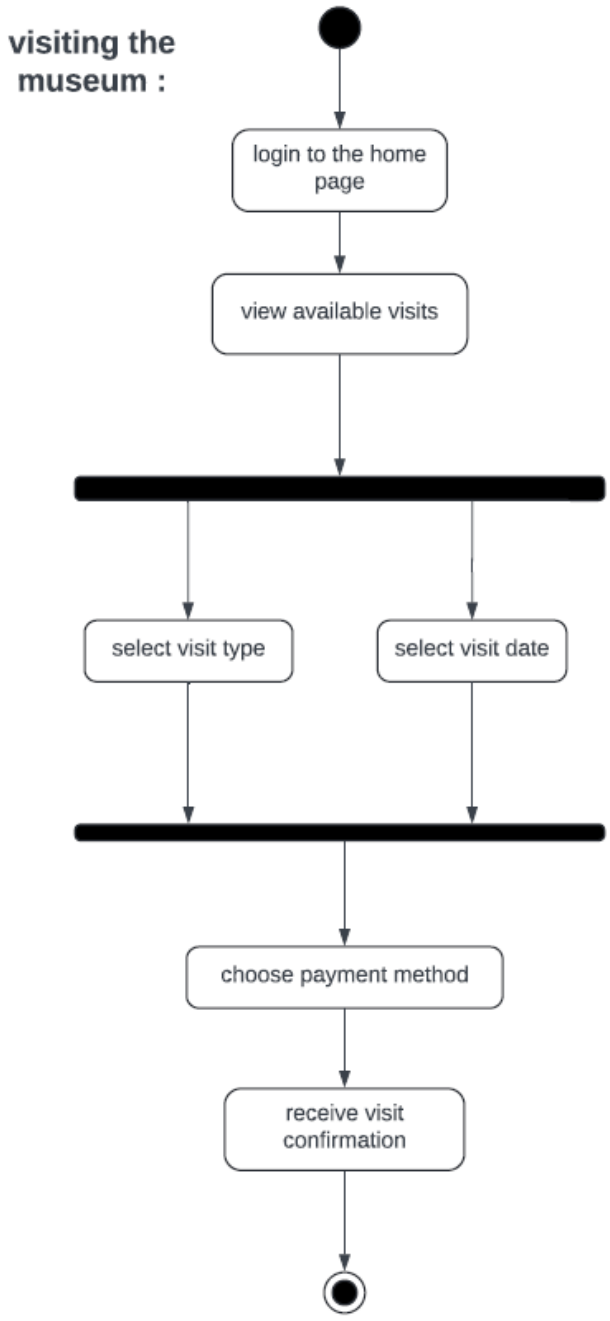
5. Multiple views for a model: Models can have multiple views

6. Testability: with the clearer separation of concerns, each part can be better tested independently

Helwan Muesum - System Architecture : MVC



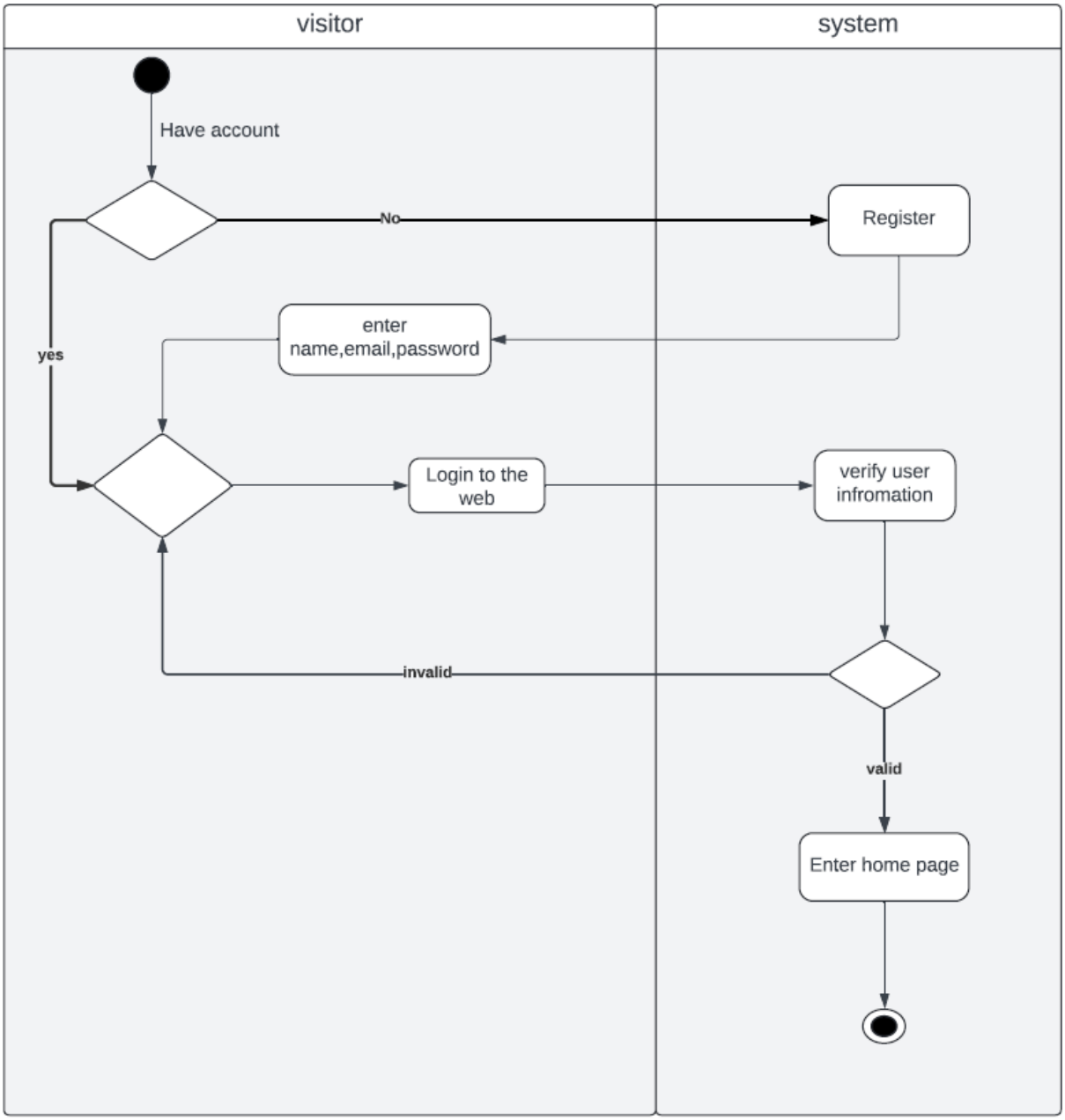
5. Activity Diagrams



The visitor will be able to Book a ticket the museum. Moreover the visitor will interact with system when the visitor decide to book an ticket After booking the Ticket, the visitor will be able to choose the payment method. Then the system will send confirmation Message to the visitor .

The interface will be essential when the visitor books a Ticket.

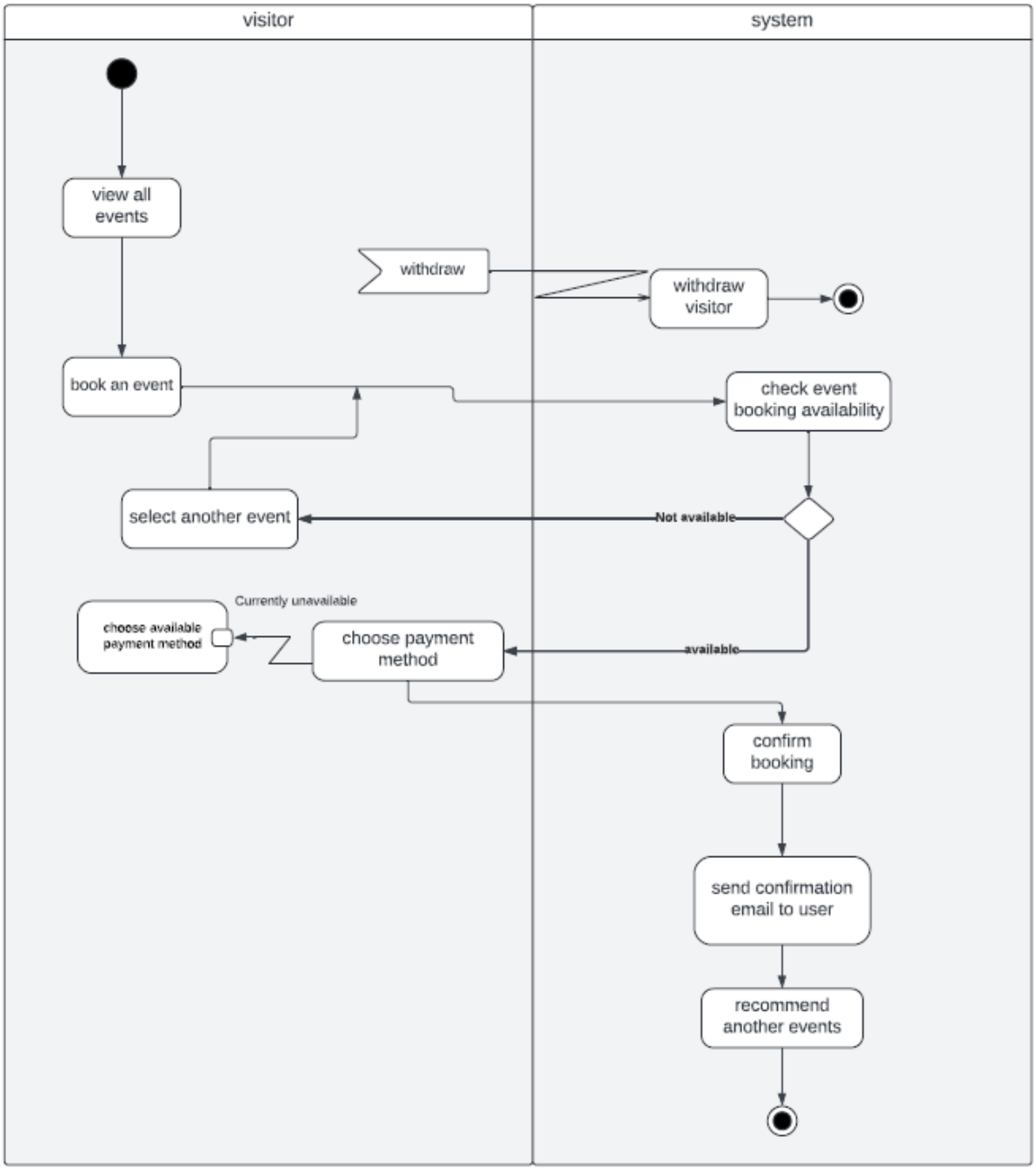
Login :


visitor
system
Have account
No
Register
enter
name,email,password
yes
Login to the
web
verify user
infromation
invalid
valid
Enter home page

1. A Visitor interacts with the system. The Visitor enter logging data [e-mail and password and sign in.

2. This interface will be needed when a Visitor want to log in

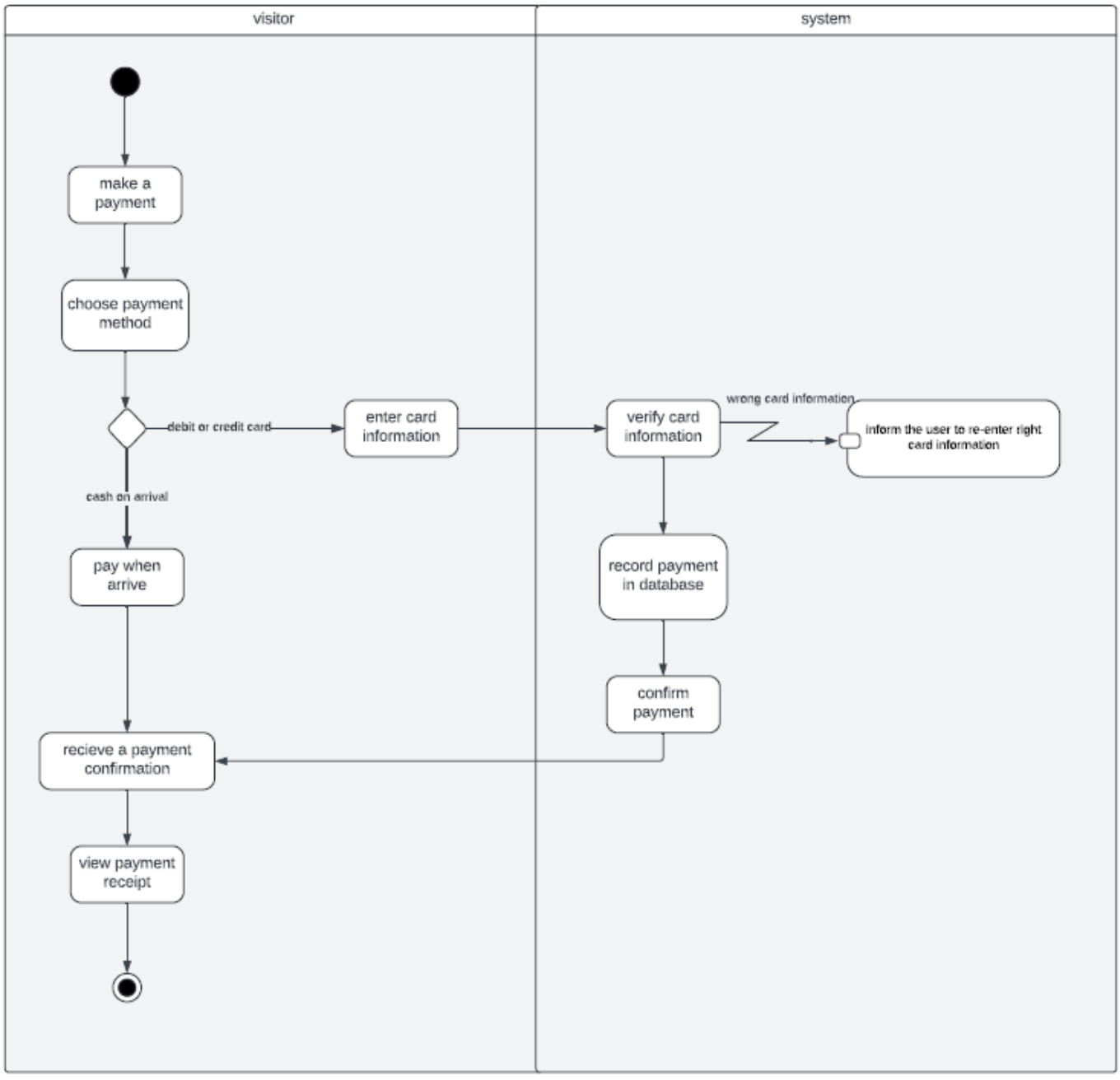
Booking event :



The visitor will be able to see all the events that related to the museum. Moreover the visitor will interact with system when the visitor decide to book an event. The system will check if this event is available or not. furthermore if the event is not available, the visitor can book another event. After booking the event, the visitor will be able to choose the payment method. Then the system will send confirmation email to the visitor and recommend another event.

The interface will be essential when the visitor books an event.

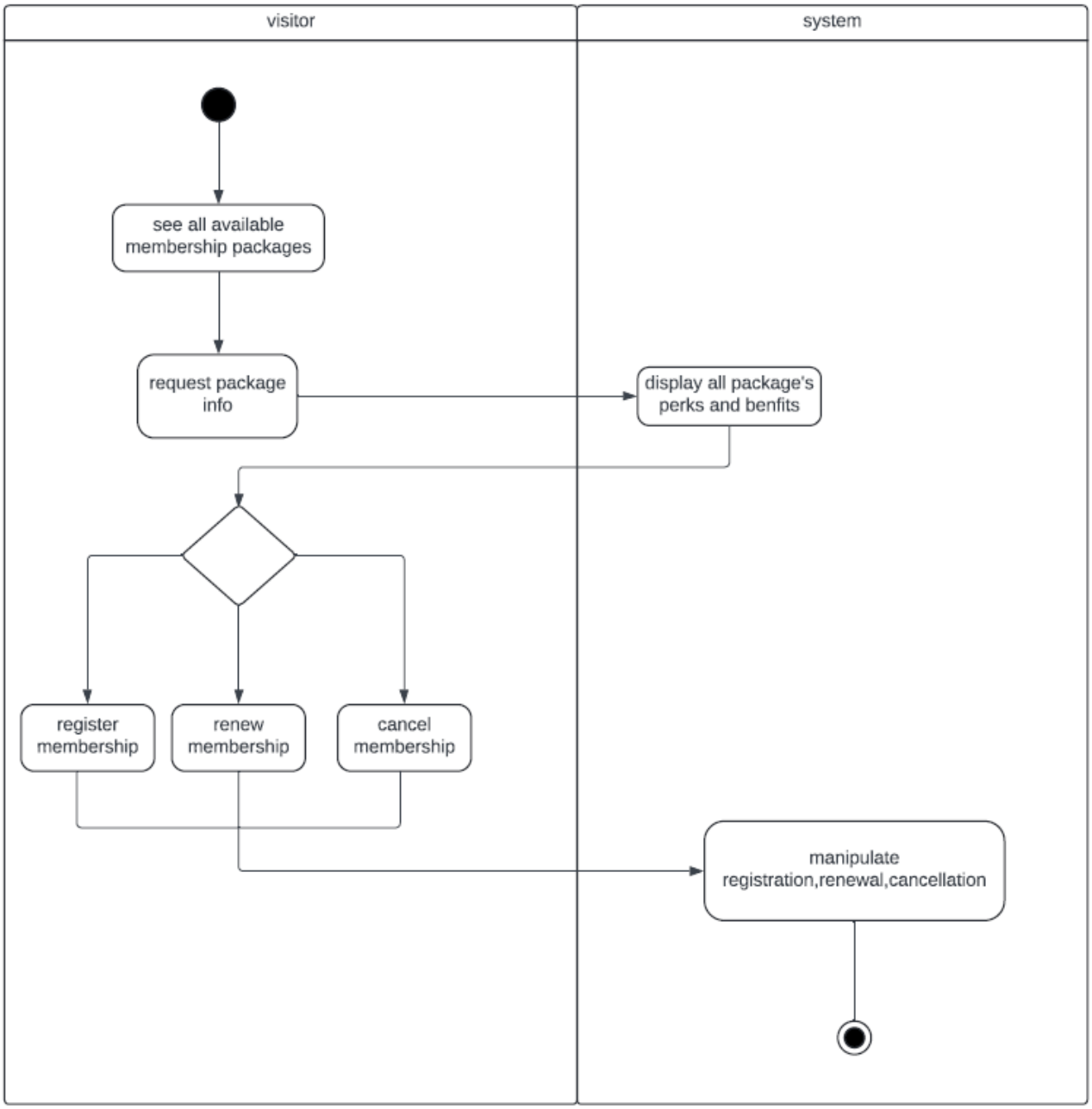
payment:



The visitor will interact with the system when he make a payment. The visitor will be able to choose to the payment method whether it's a credit card or cash , then if the visitor decides to pay by credit card the system will verify the card information. So if the information is right the system will record this information in the database. Finally the visitor will receive the payment confirmation.

The interface will be needed when the visitor wants to make a payment process.

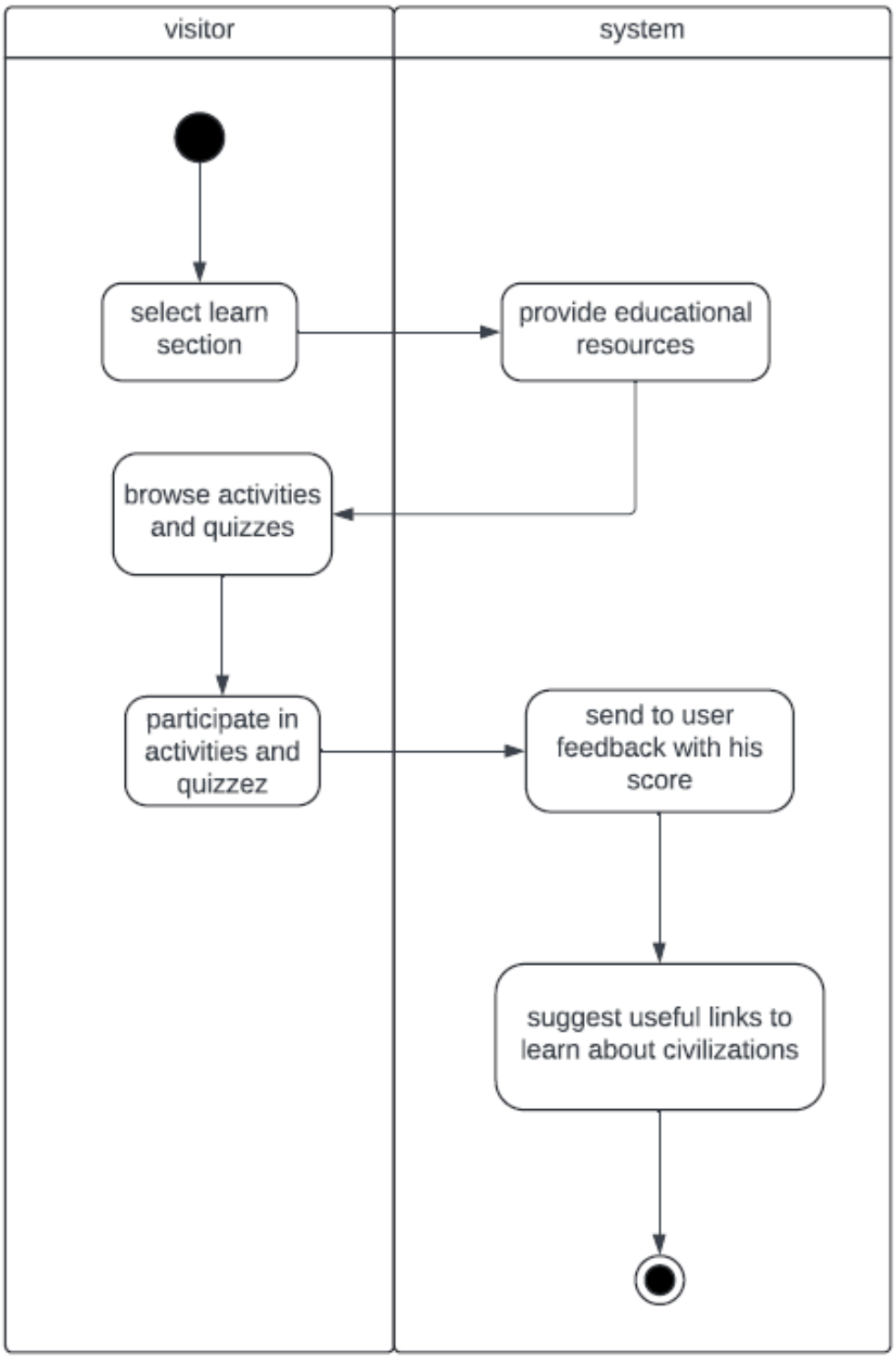
membership :



The visitor will be able to see all available membership packages, The visitor will interact with the system by sending request package info. The system will display all package benfits. Also, The visitor will be able to manage his membership.

The interface will be needed when the visitor tries to see the membership packages

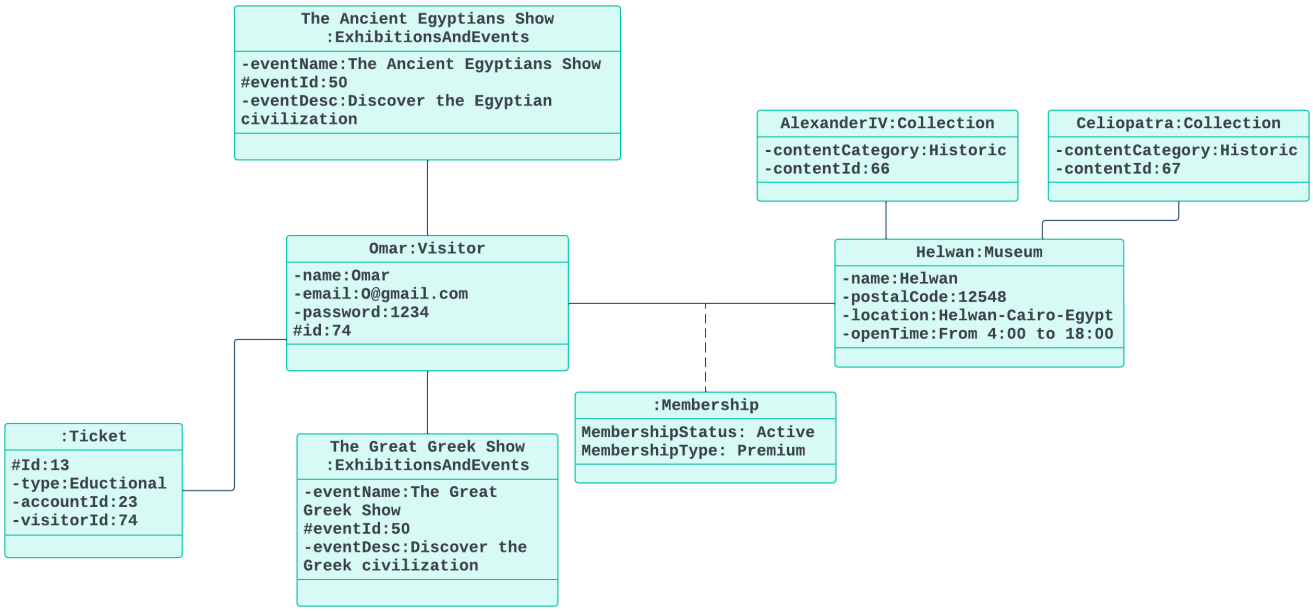
Learn :



A visitor interacts with the system by choosing the learn section. The visitor will be able to enter activities and participate in quizzes, Also the system will send the feedback quiz scores and suggest useful links for the visitor.

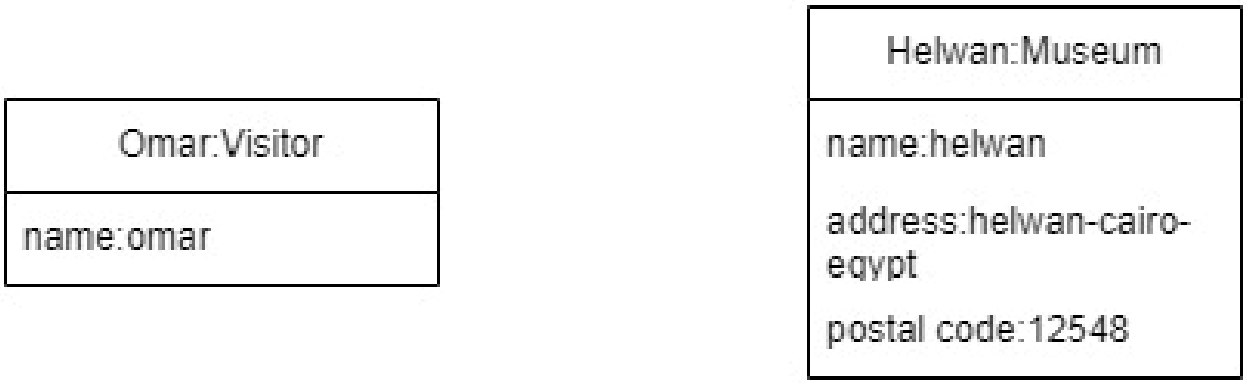
The interface will be needed if the visitor choose the learn section to see the resources.

6. Object Diagrams *(Including object diagrams that illustrate the preconditions and the post-conditions of selected functions*



Object diagram - Register

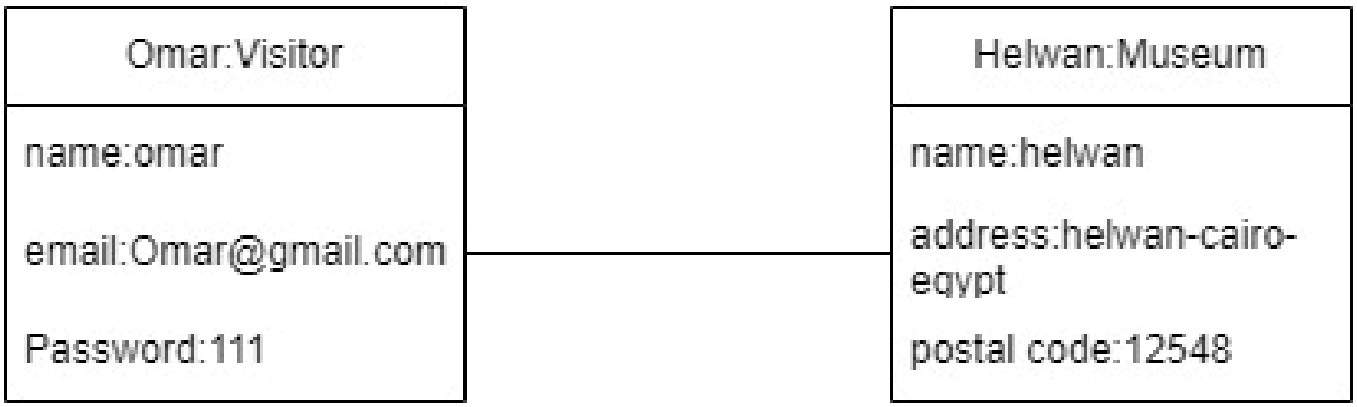
Pre



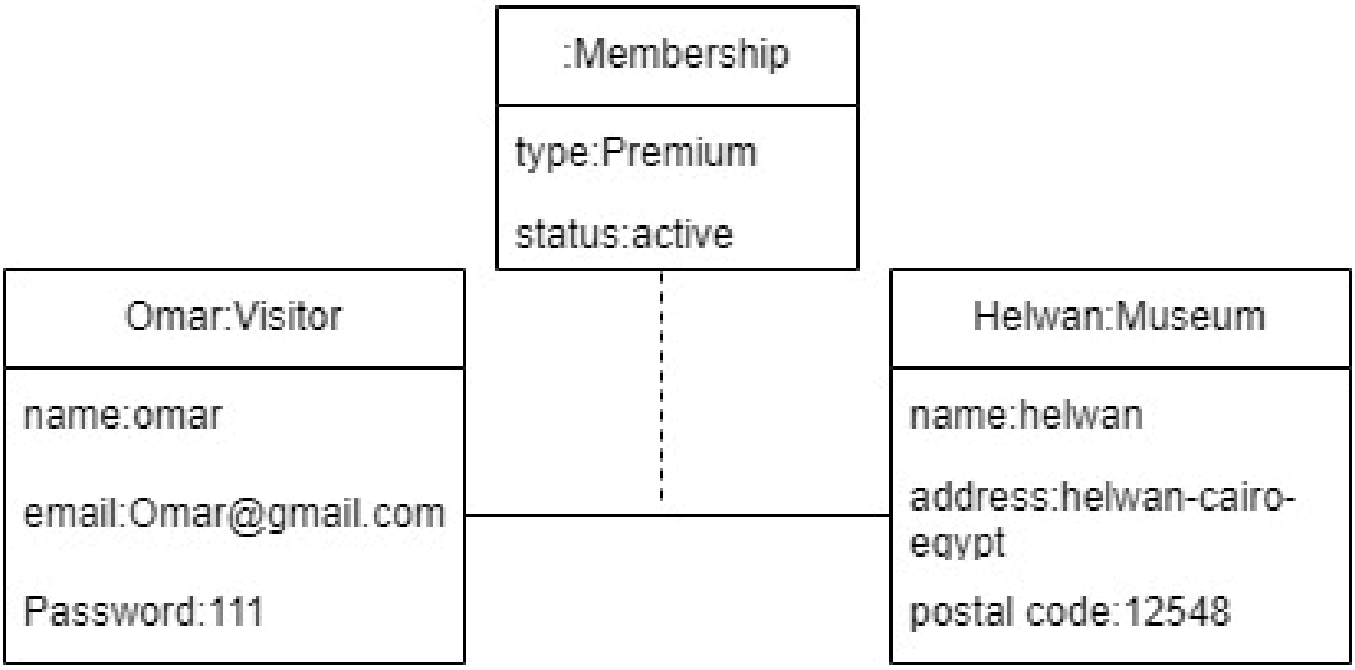
Post

Object diagram - Manage membership

Pre

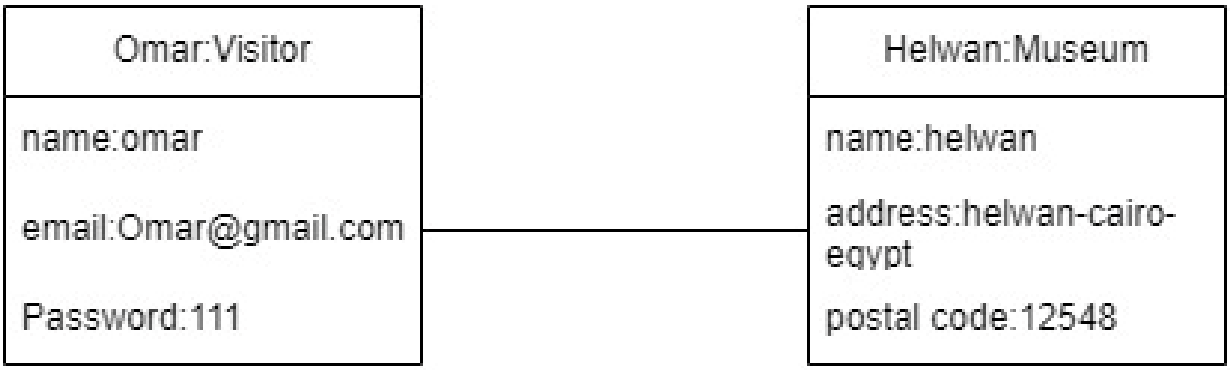


Post

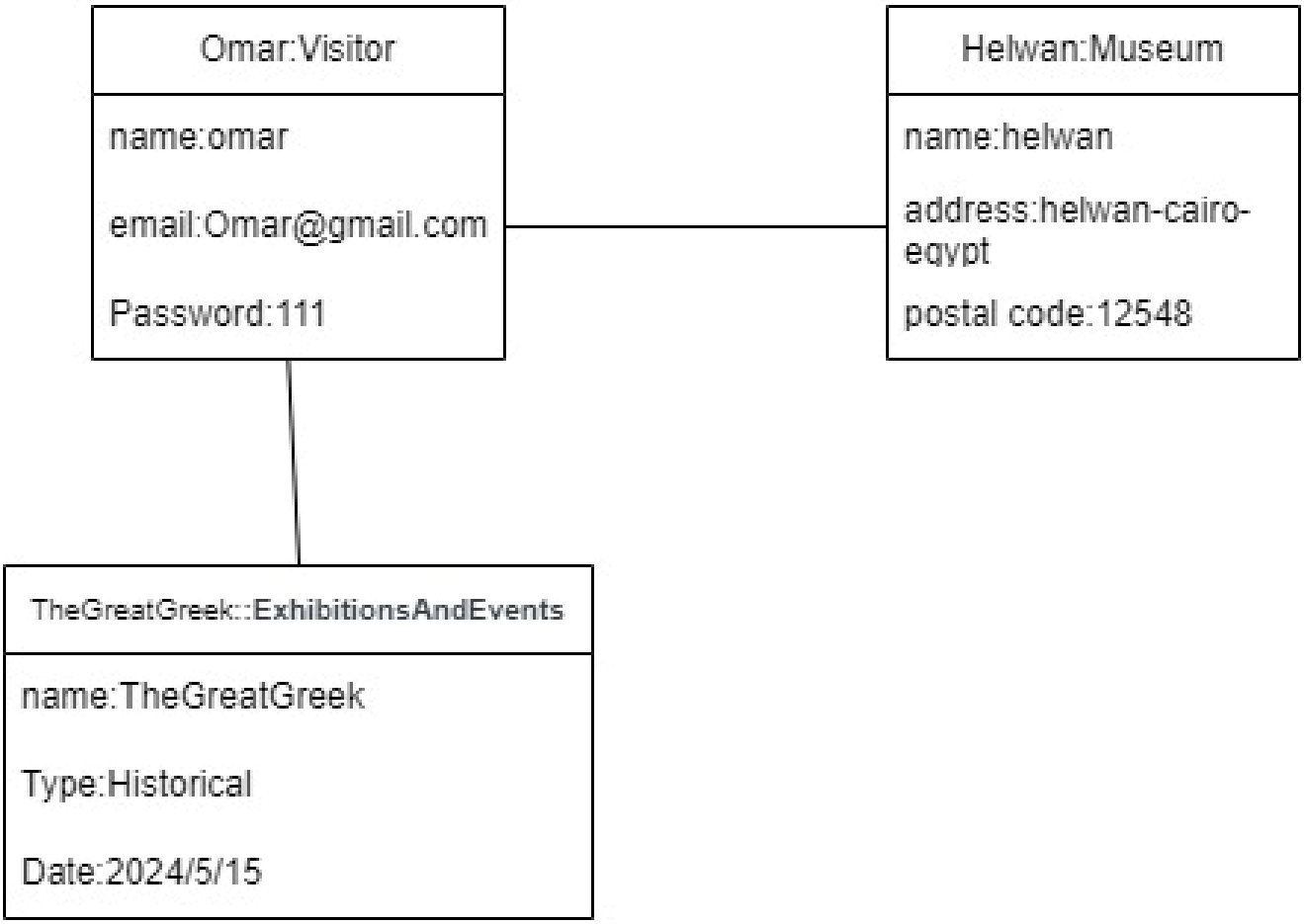


Object diagram - Book a ticket

Pre



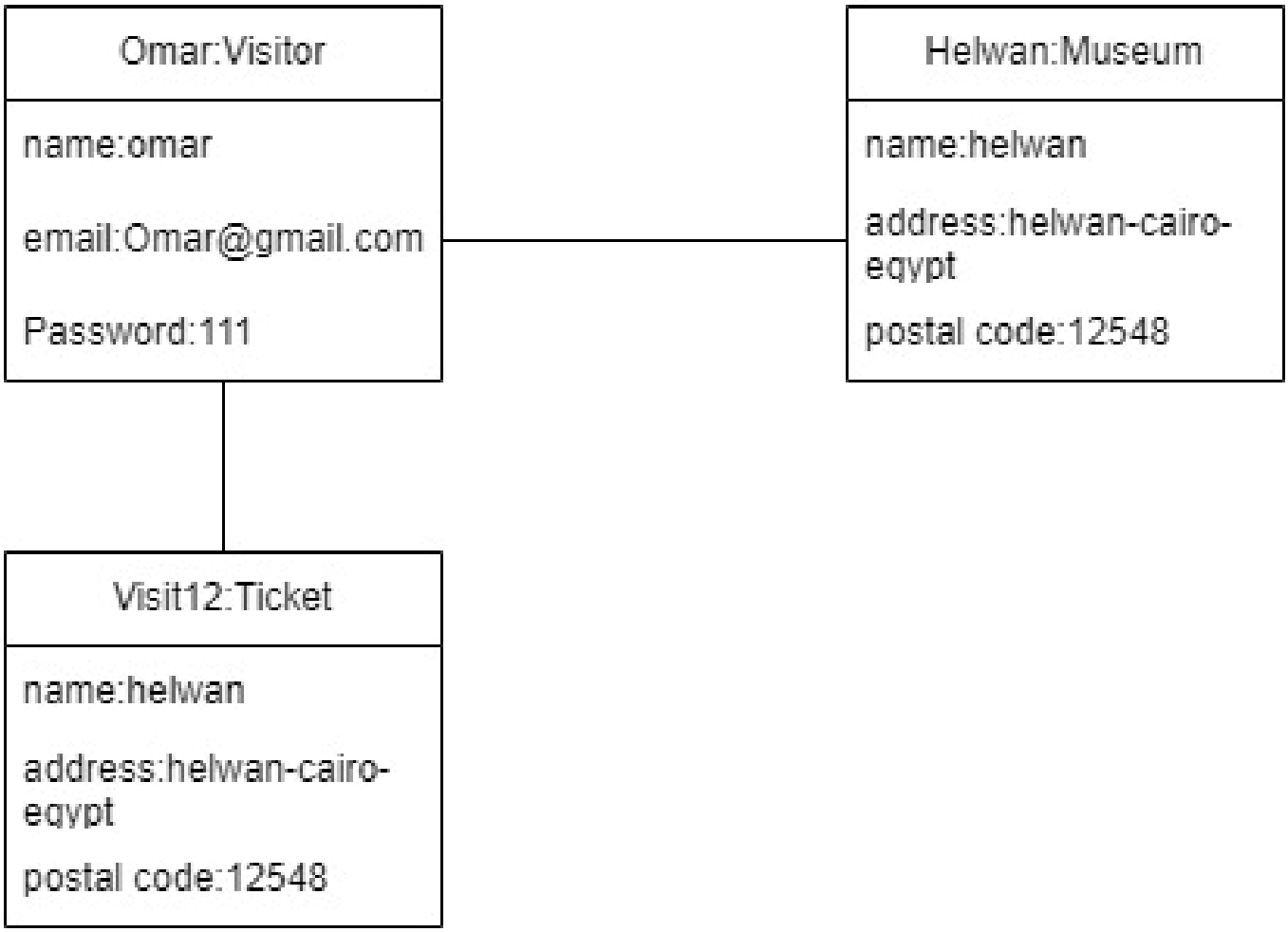
Post



Object diagram - Book a ticket

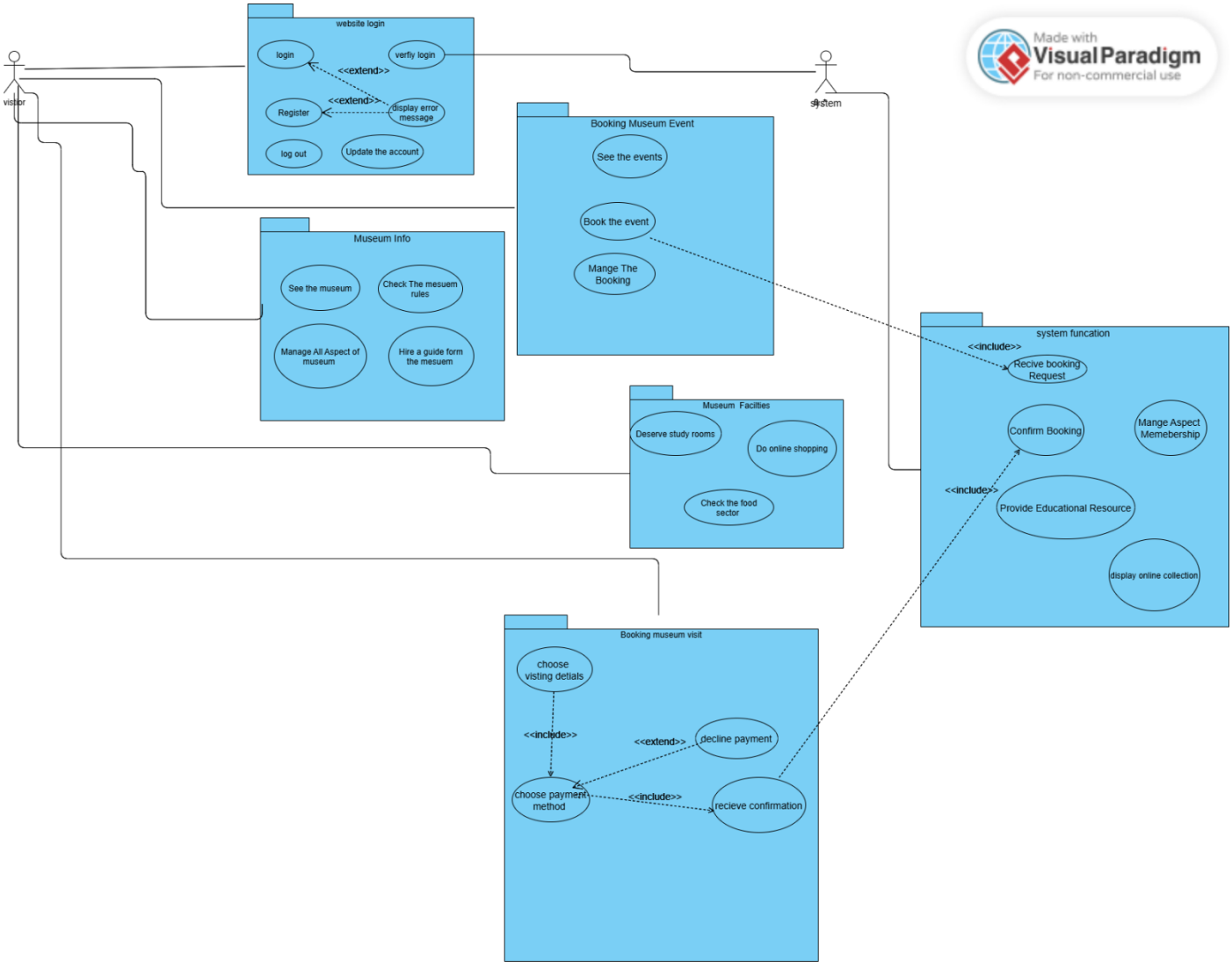
Pre

Post



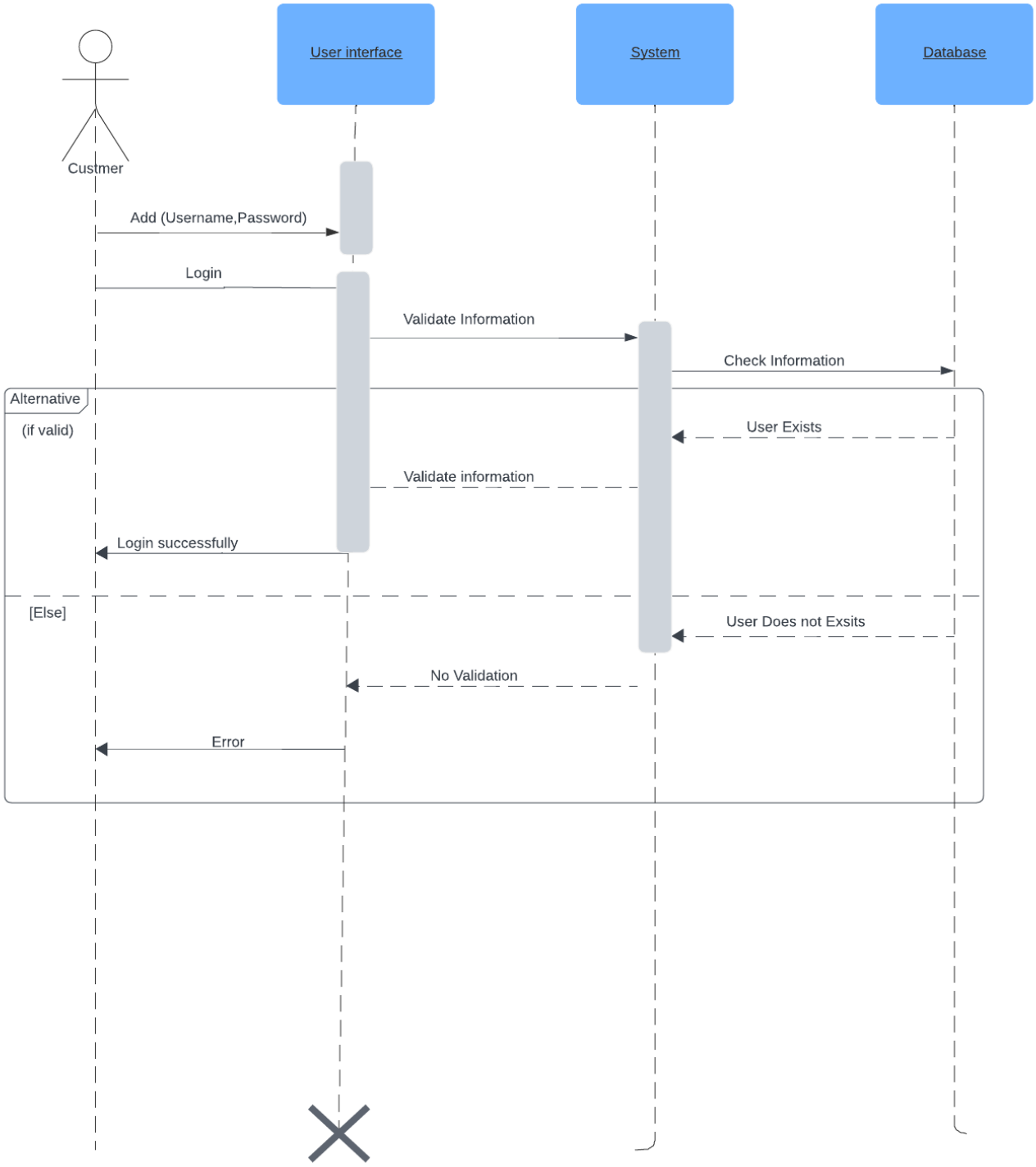
7. Package Diagram(s)

Usecase package diagram



8. Sequence Diagrams *including* ***System Sequence Diagrams (SSDs)***

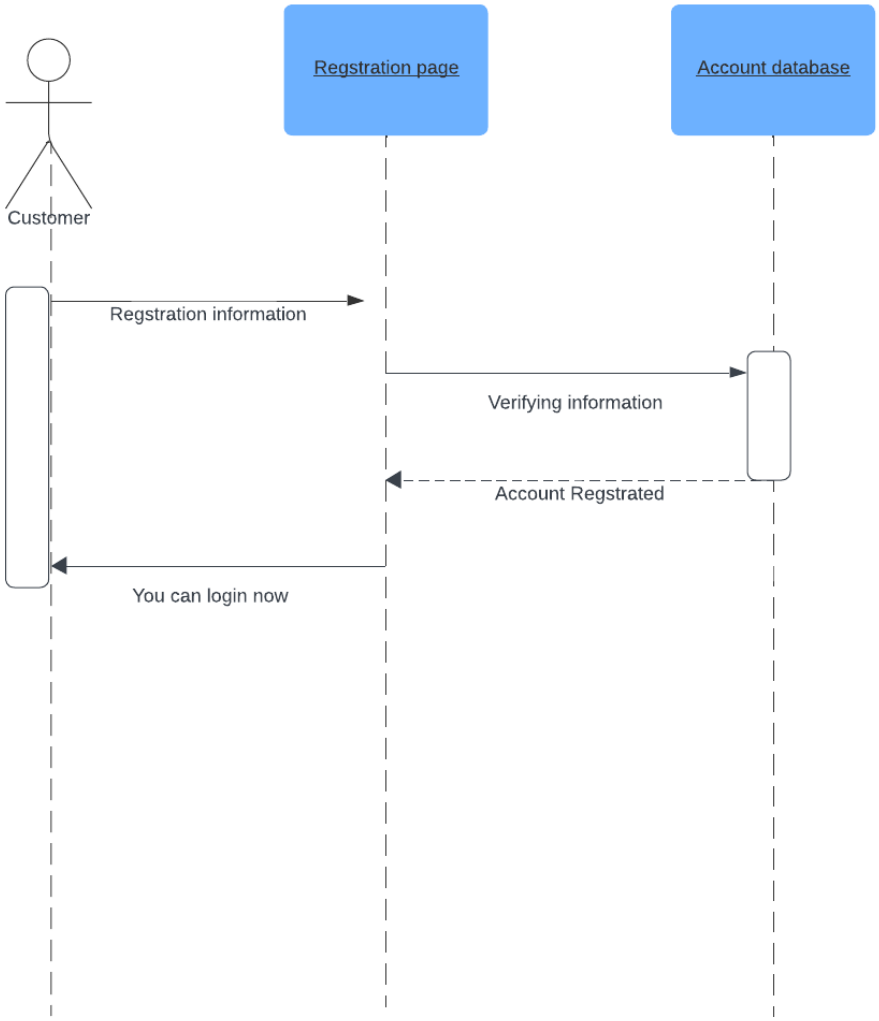
Login


User interface
System
Database
Custmer
Add (Username,Password)
Login
Validate Information
Check Information
Alternative
(if valid)
User Exists
Validate information
Login successfully
[Else]
User Does not Exsits
No Validation
Error

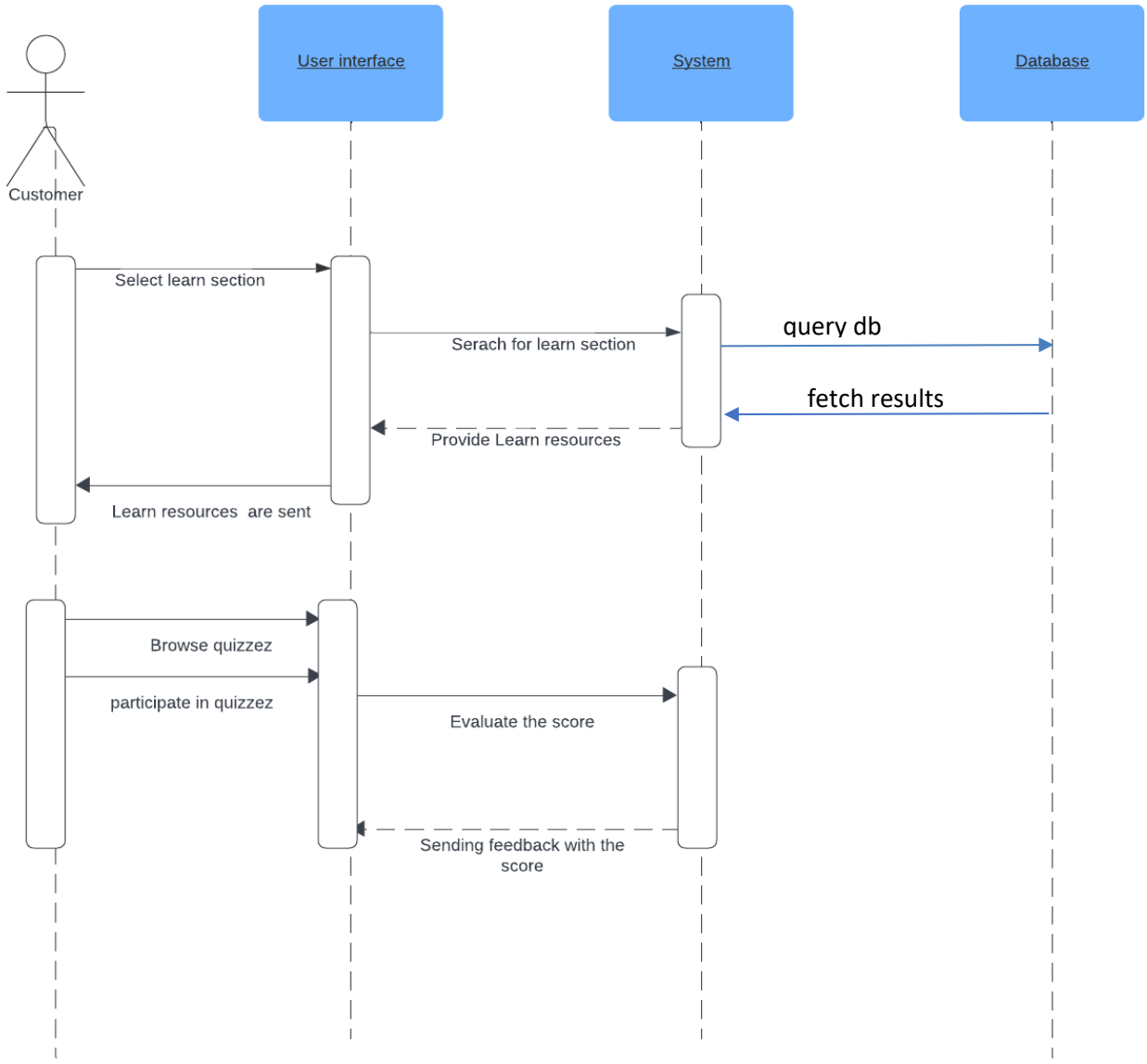
Collection


User interface
System
Database
Custmer
See all the collection
borwseCollection()
Get the collection
sending the collection to the system
getOnlineCollection()
collection is sent successfully
Search an item
searchByCategory()
check the item in the database
Alternative
If exsists
the item exsists
send the item
item is sent
[Else]
the item does not exists
failed to send the item
item is not sent
show detailed infotmation about item
sending to the system to display the information
the information
the information is sent
give a feedback
sending the feedback
the feedbac is sent to the database

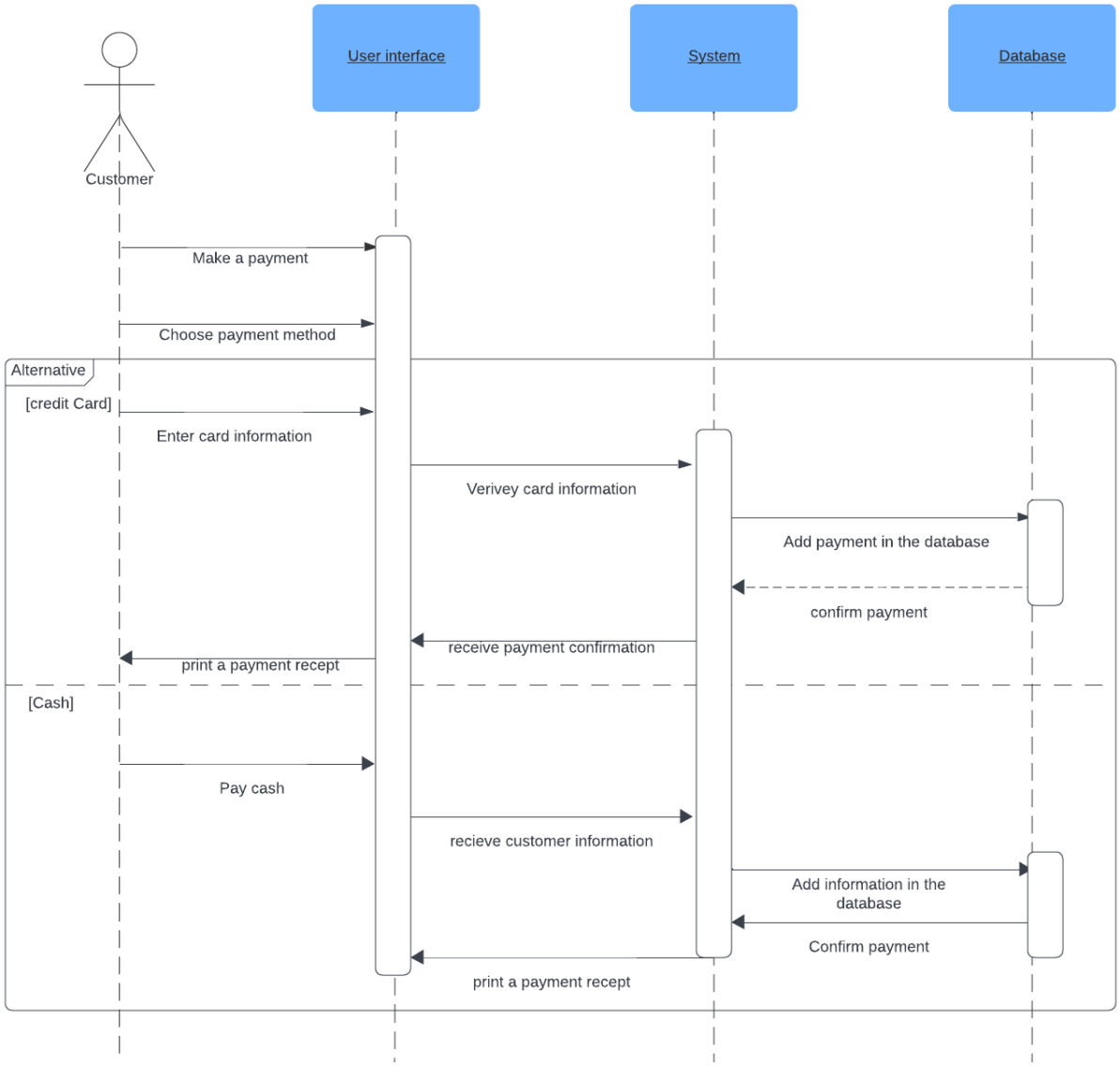
Register


Regstration page
Account database
Customer
Regstration information
Verifying information
Account Regstrated
You can login now

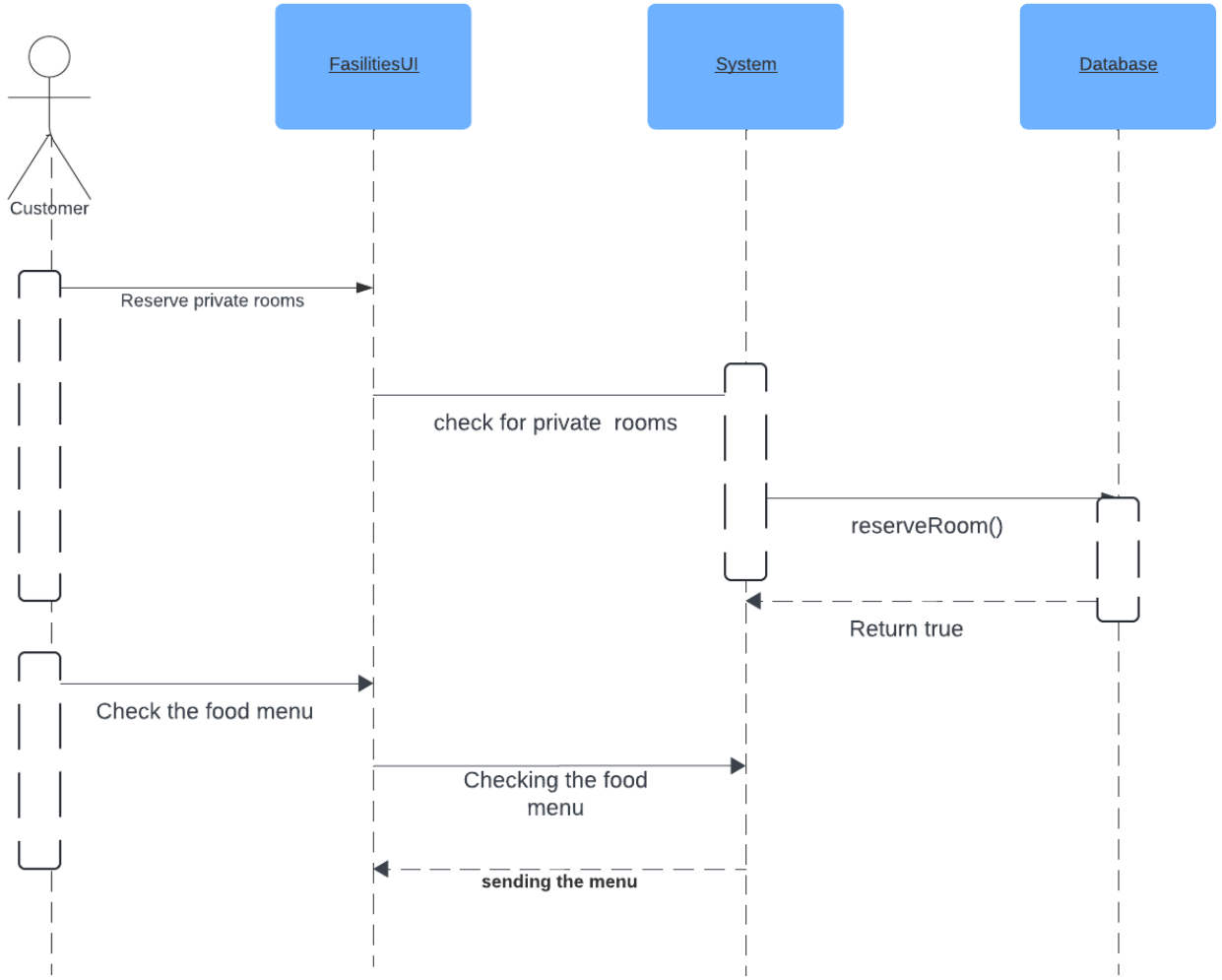
Learn


User interface
System
Database
Customer
Select learn section
Serach for learn section
query db
fetch results
Provide Learn resources
Learn resources are sent
Browse quizzez
participate in quizzez
Evaluate the score
Sending feedback with the score

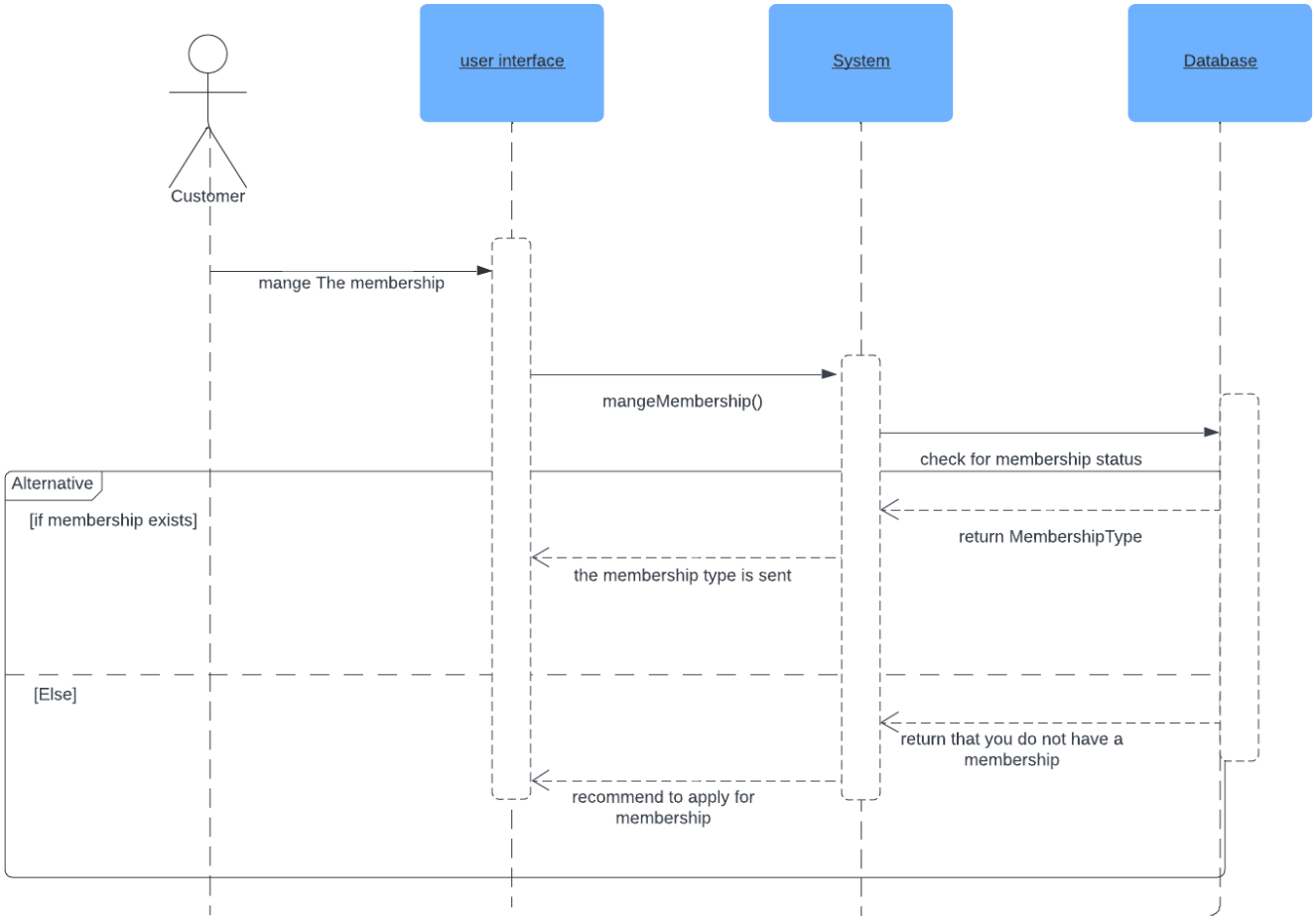
Payment


User interface
System
Database
Customer
Make a payment
Choose payment method
Alternative
[credit Card]
Enter card information
Verivey card information
Add payment in the database
confirm payment
receive payment confirmation
print a payment recept
[Cash]
Pay cash
recieve customer information
Add information in the database
Confirm payment
print a payment recept

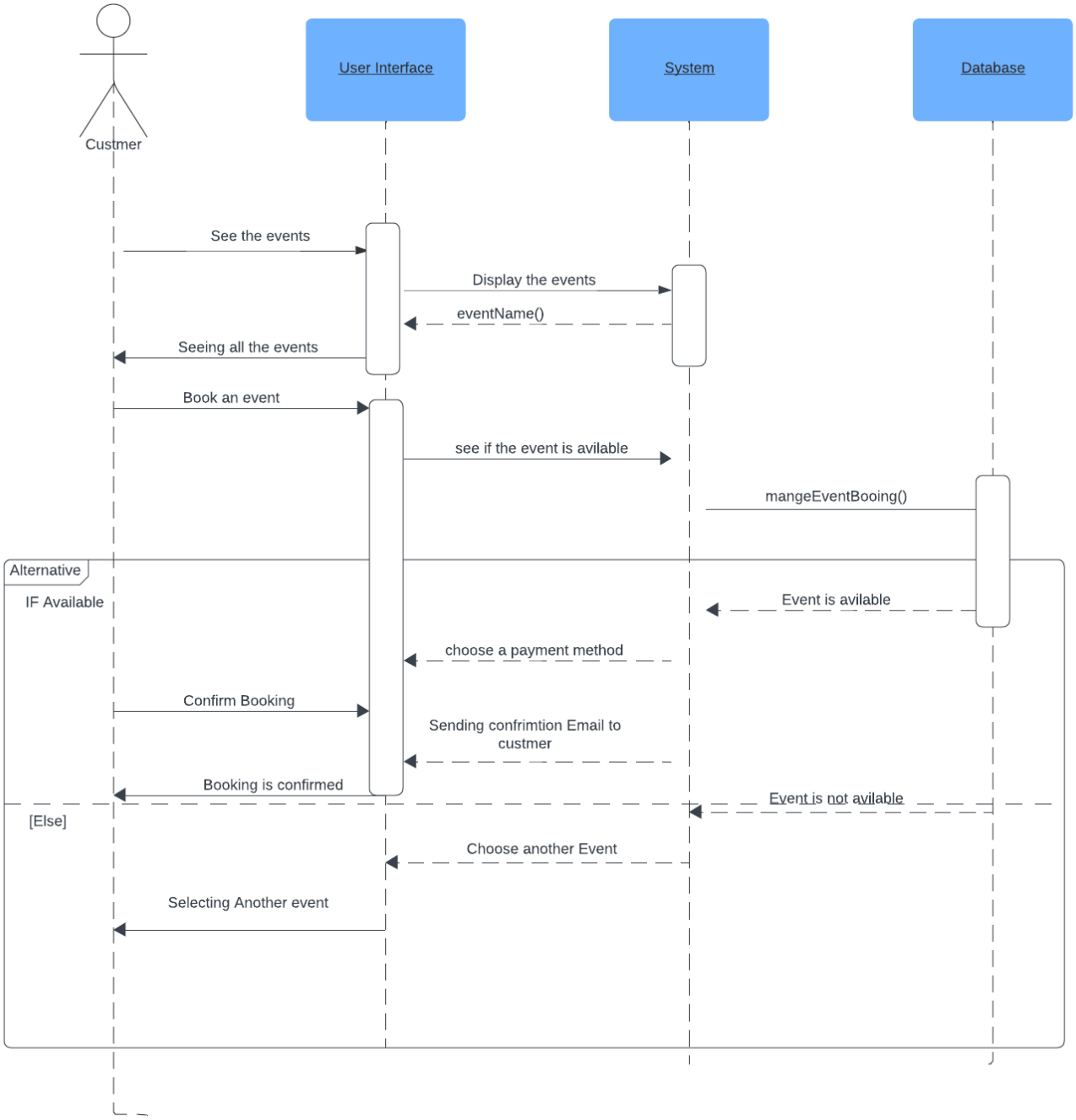
Facilities


FasilitiesUI
System
Database
Customer
Reserve private rooms
check for private rooms
reserveRoom()
Return true
Check the food menu
Checking the food menu
sending the menu

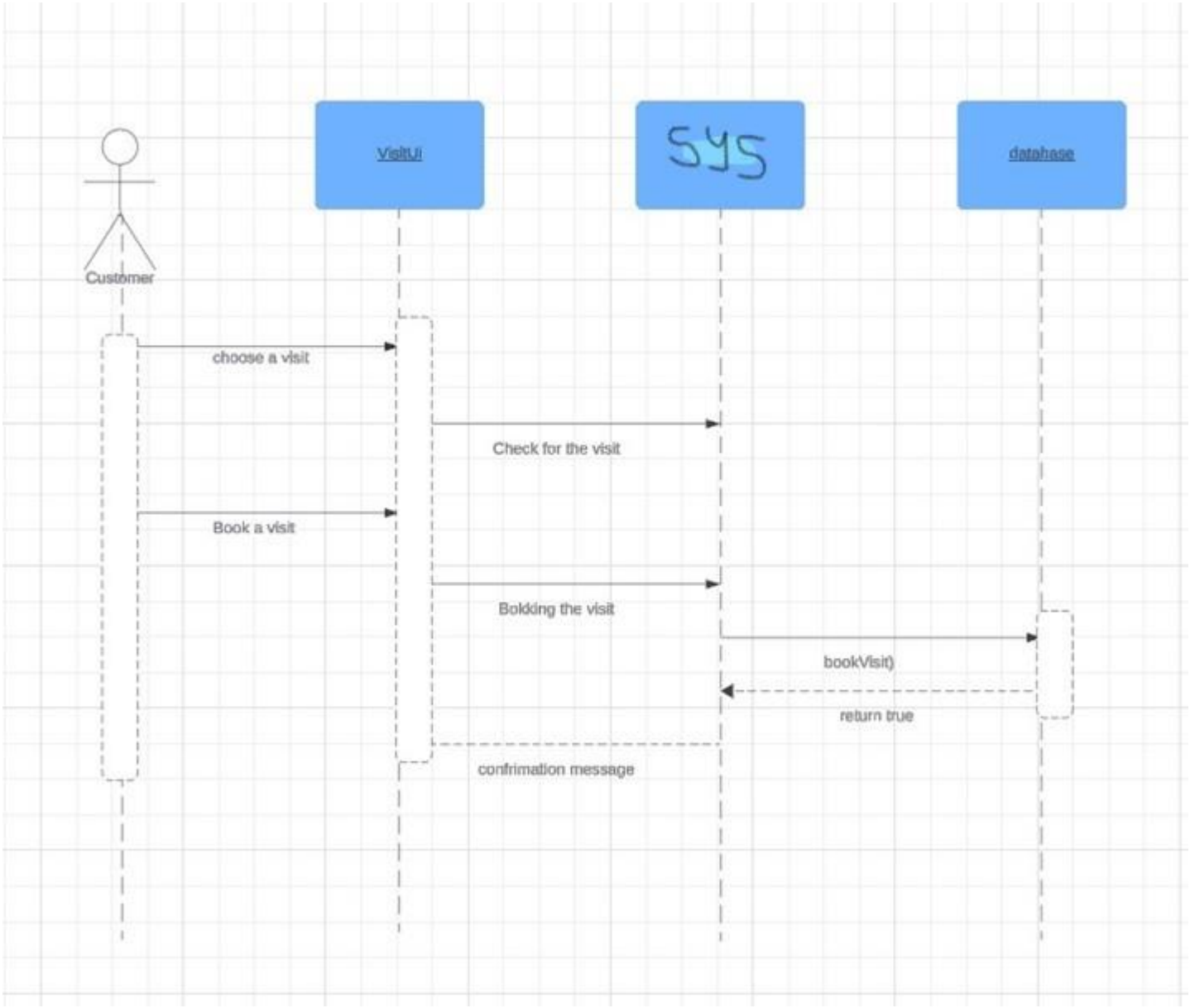
membership


user interface
System
Database
Customer
mange The membership
mangeMembership()
check for membership status
Alternative
[if membership exists]
return MembershipType
the membership type is sent
[Else]
return that you do not have a membership
recommend to apply for membership

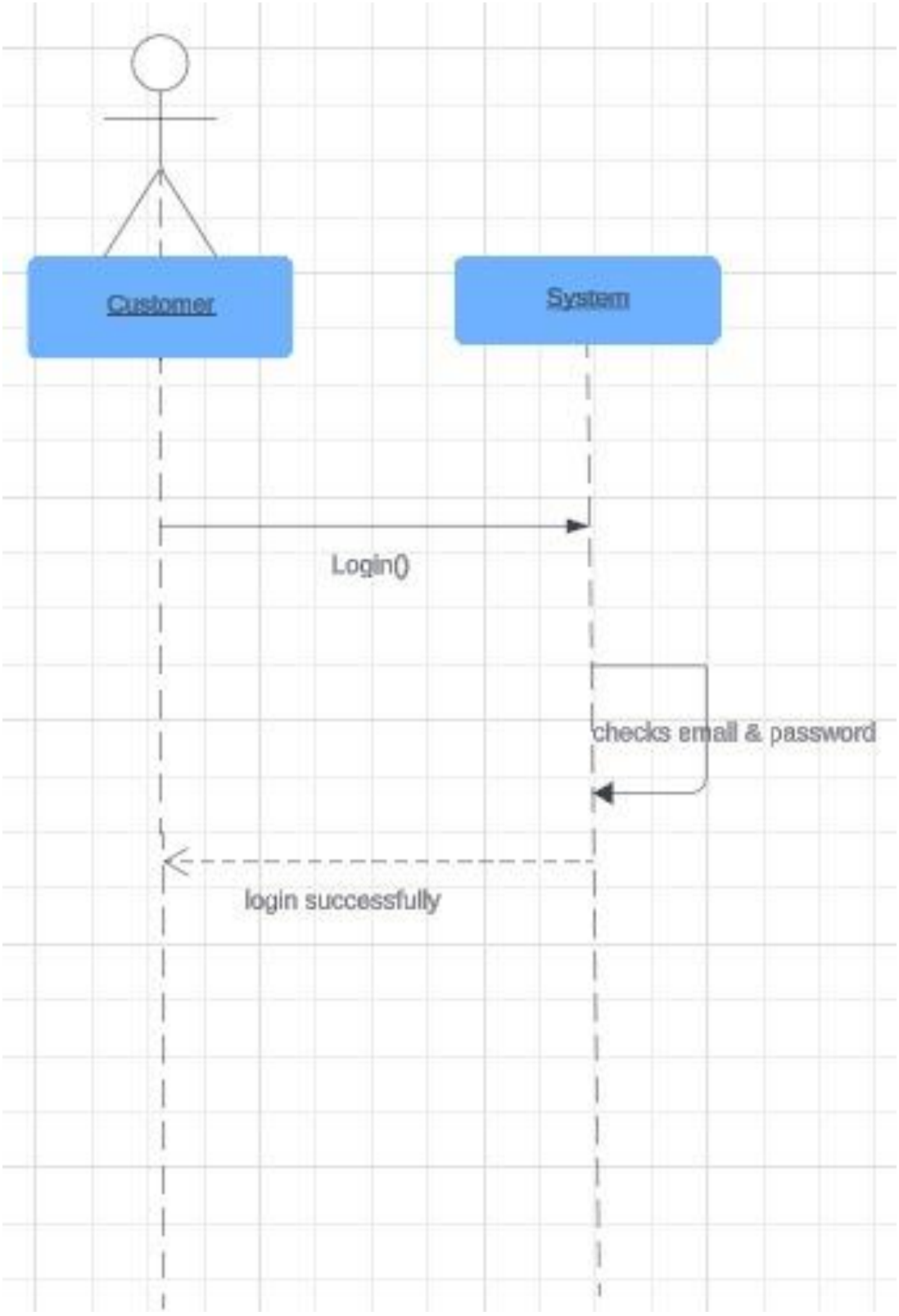
Book event

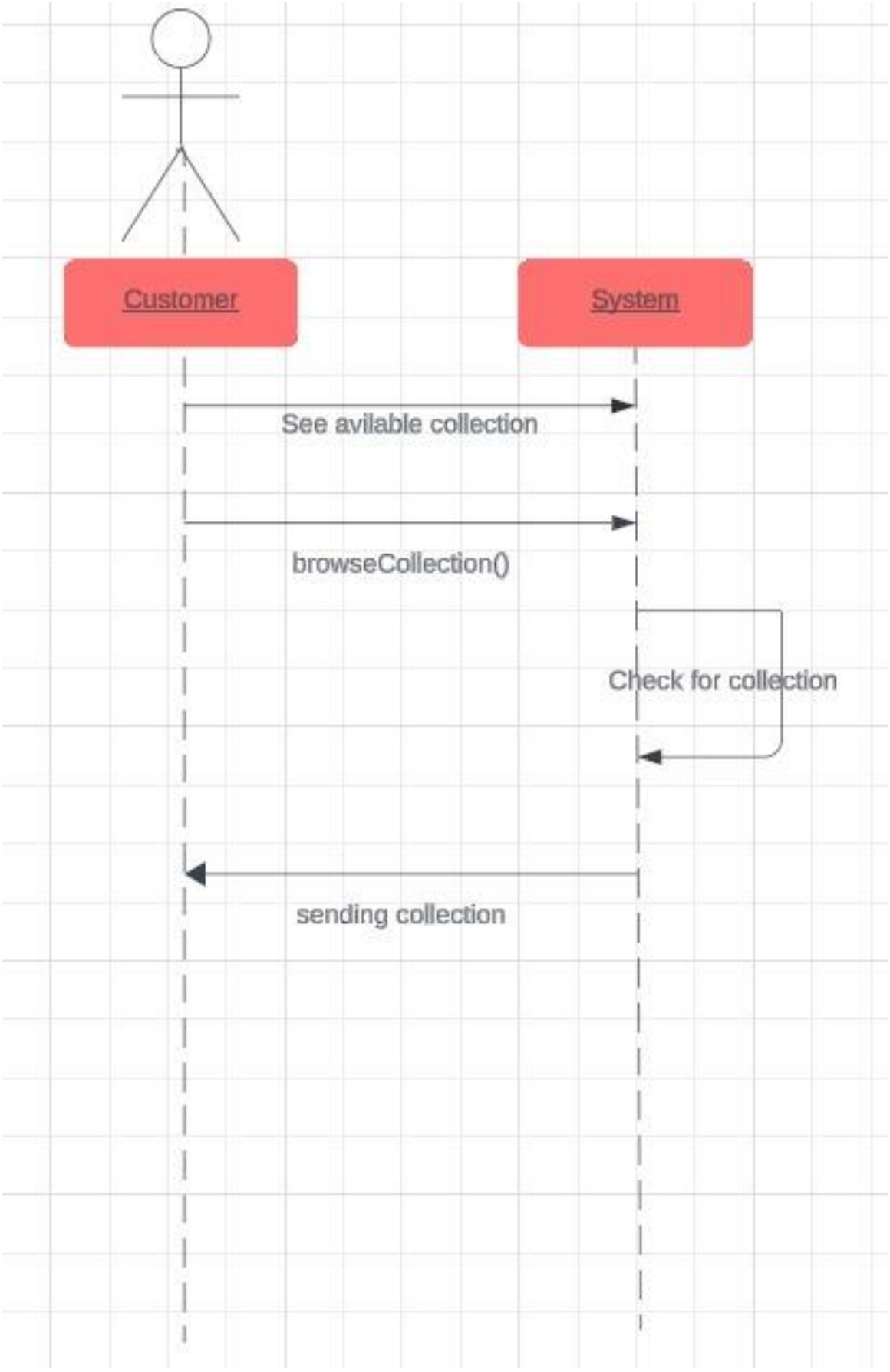

Custmer
User Interface
System
Database
See the events
Display the events
eventName()
Seeing all the events
Book an event
see if the event is avilable
mangeEventBooing()
Alternative
IF Available
Event is avilable
choose a payment method
Confirm Booking
Sending confrimtion Email to custmer
Booking is confirmed
Event is not avilable
[Else]
Choose another Event
Selecting Another event

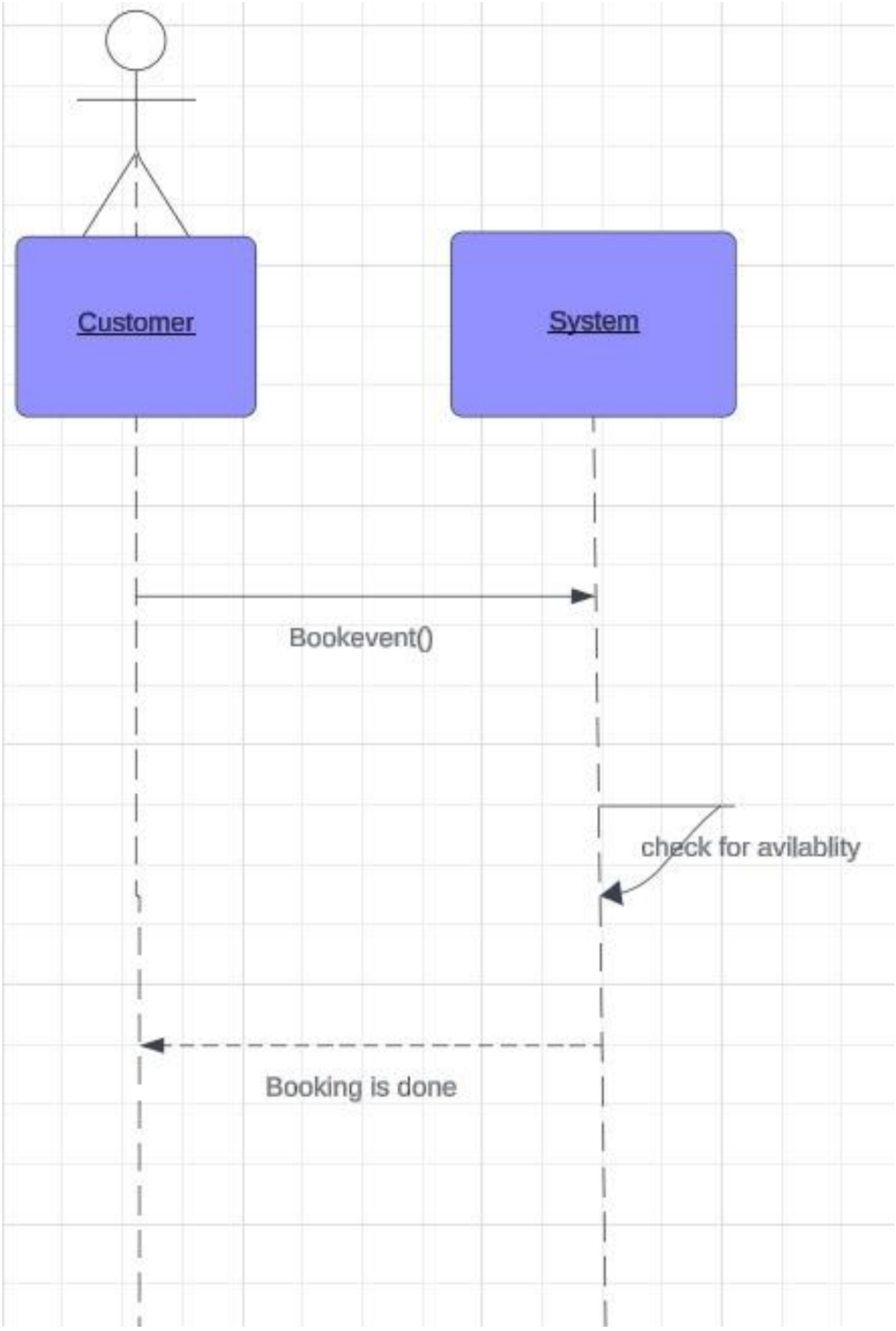
Book Visit

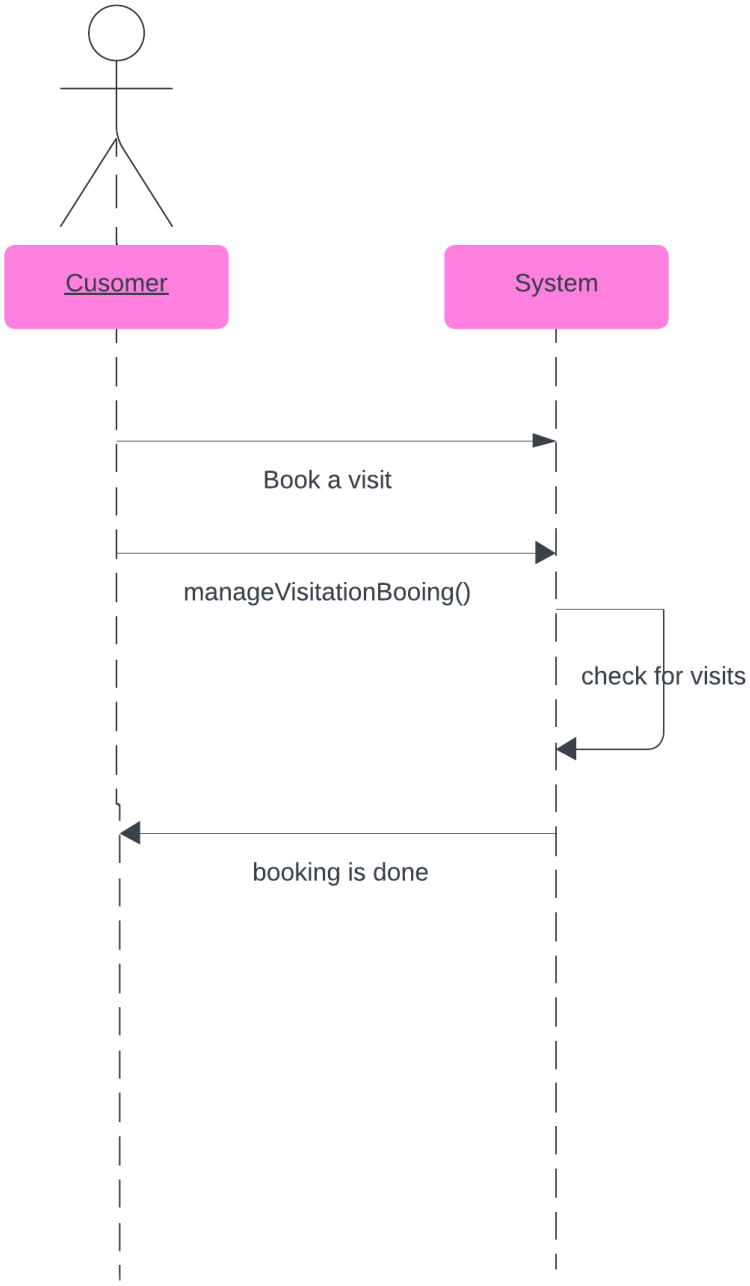

VisitUi
SYS
database
Customer
choose a visit
Check for the visit
Book a visit
Bokking the visit
bookVisit)
return true
confrimation message

SSD's

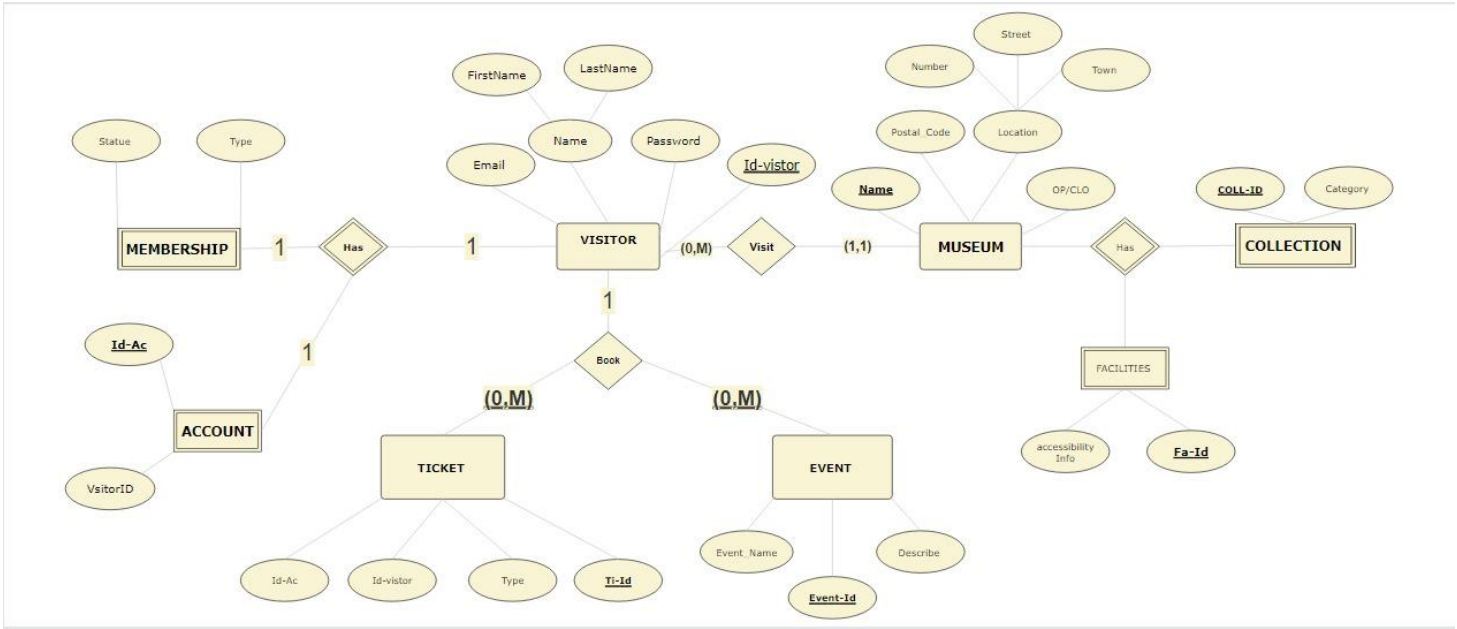

Customer
System
Login()
checks email & password
login successfully

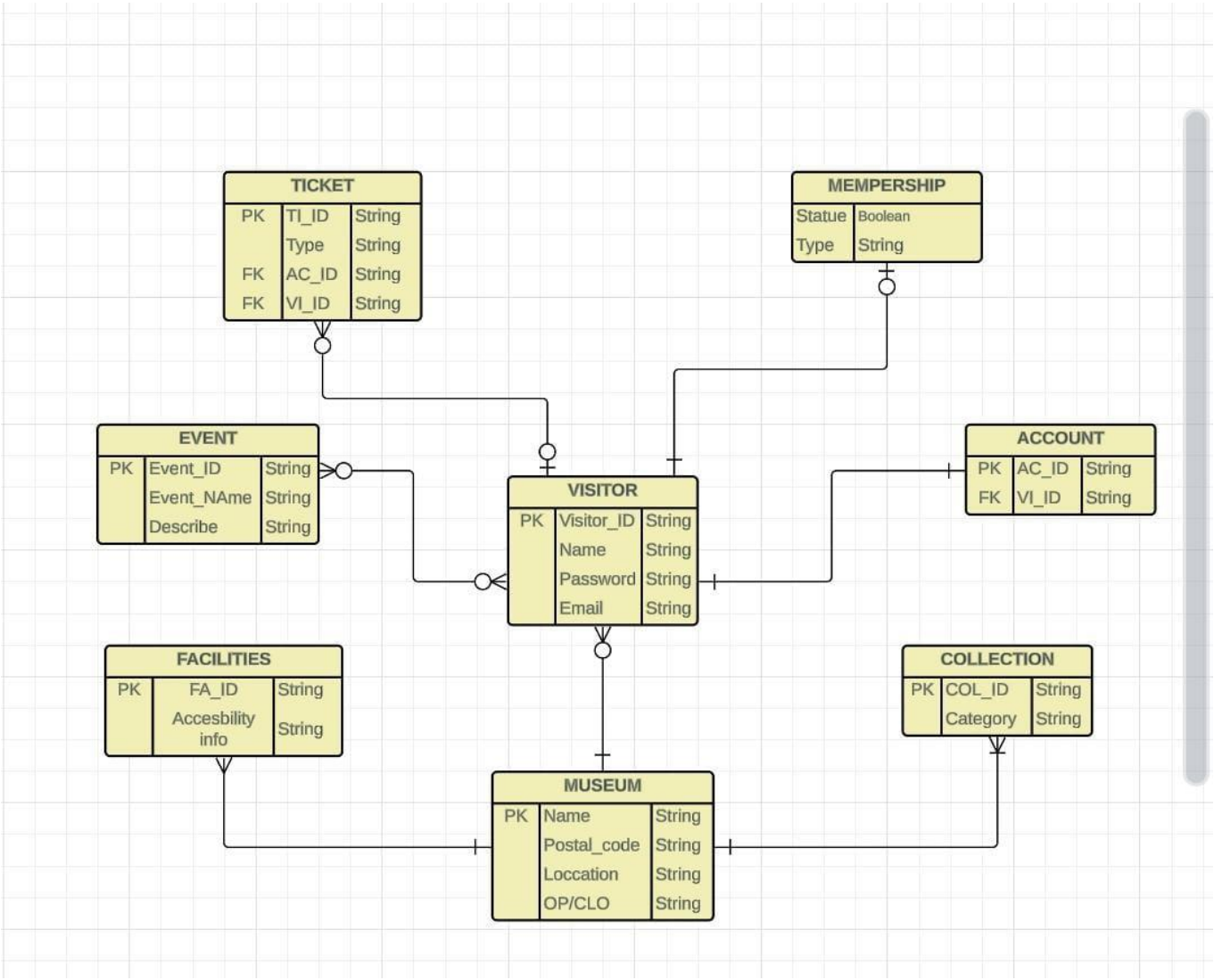

Customer
System
See avilable collection
browseCollection()
Check for collection
sending collection


Customer
System
Bookevent()
check for avilablity
Booking is done

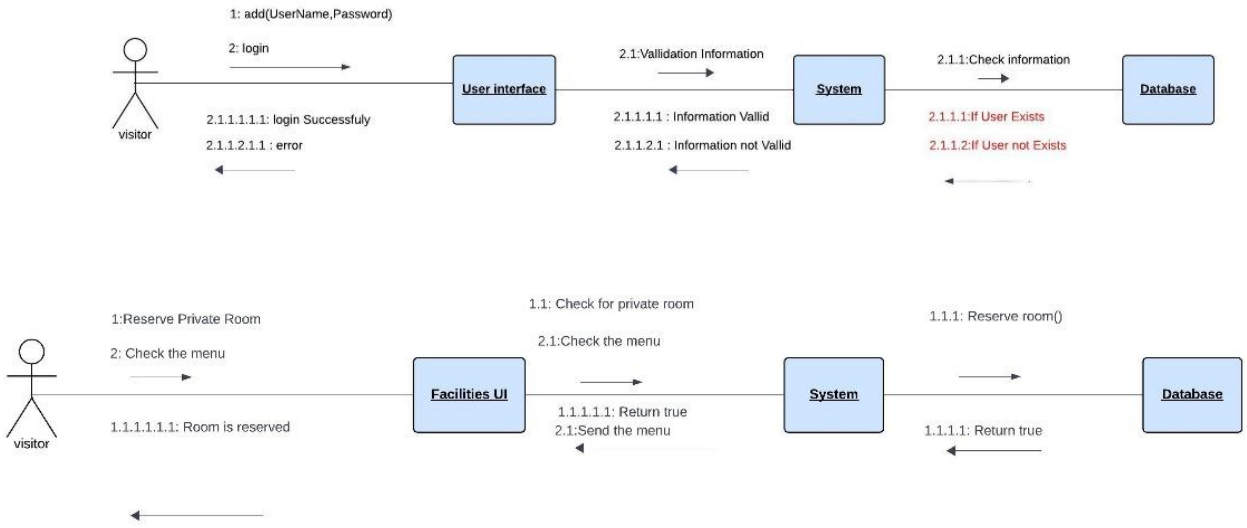

Cusomer
System
Book a visit
manageVisitationBooing()
check for visits
booking is done

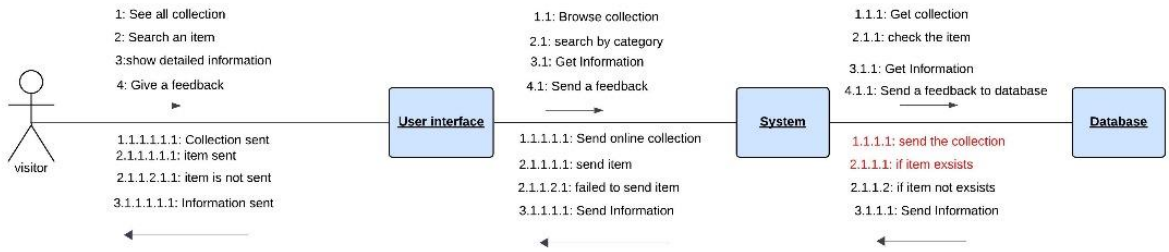
9. Database Specification *(ERD, Tables)*

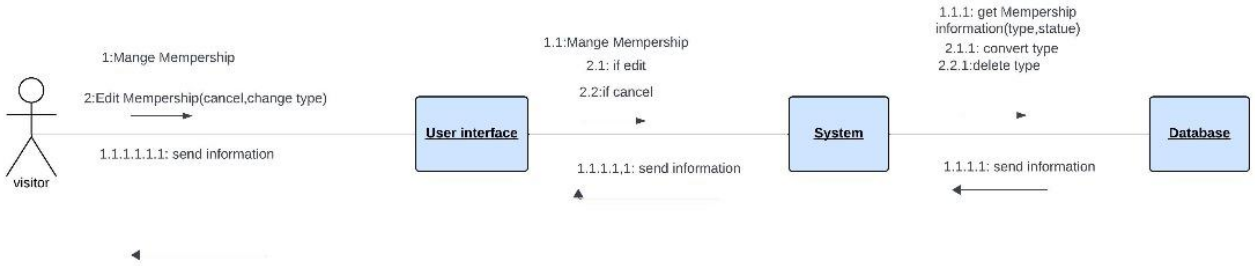


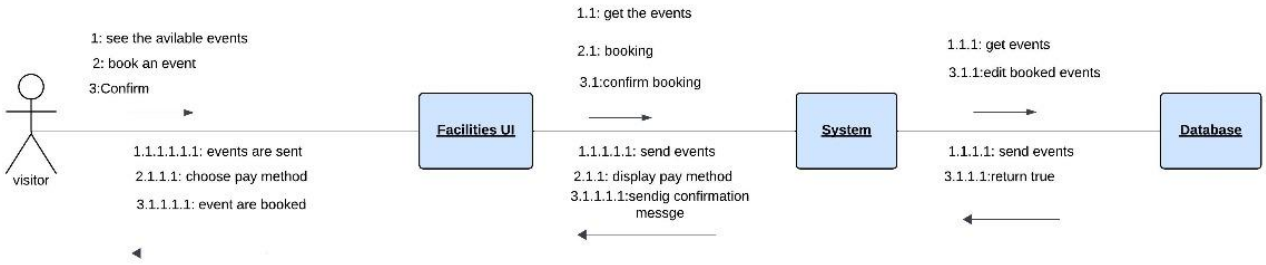


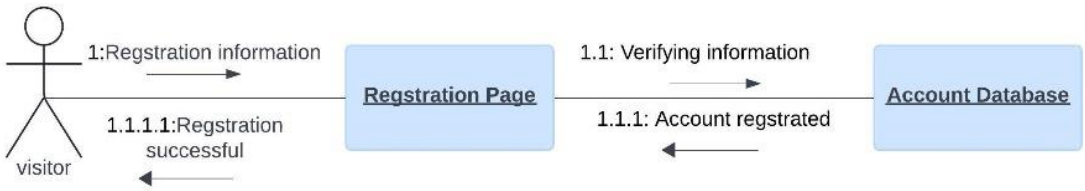
10. Collaboration/Communication Diagrams





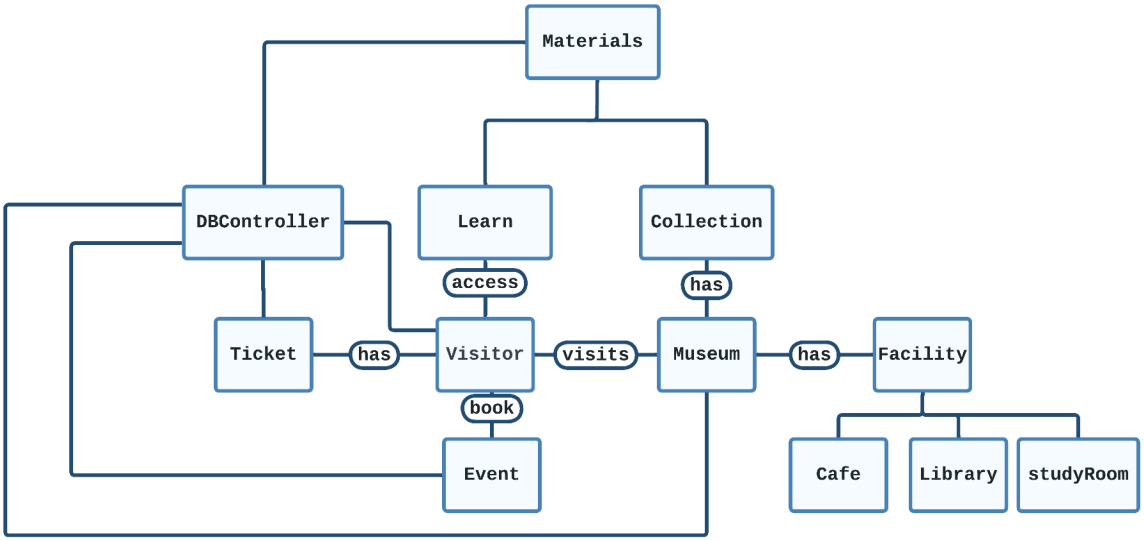




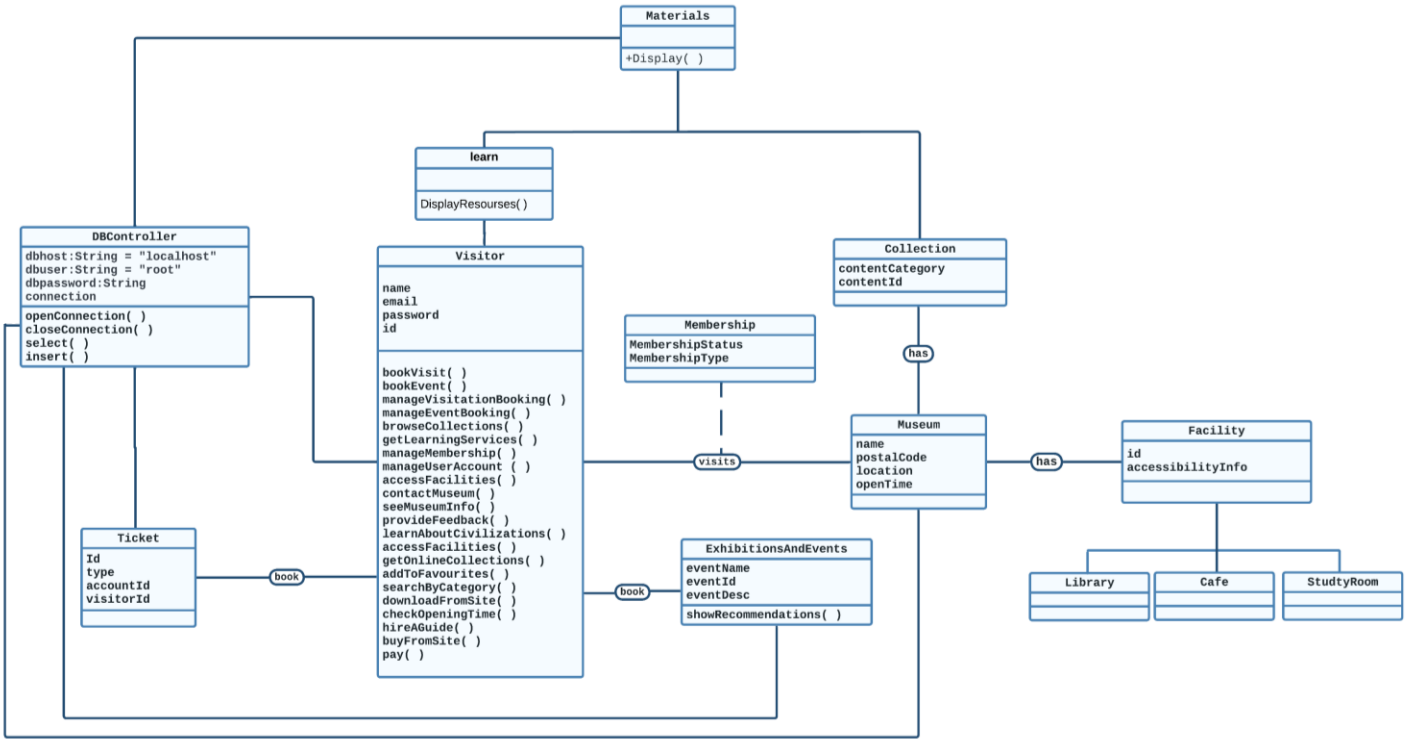


11. Class Diagram *(3 versions)*

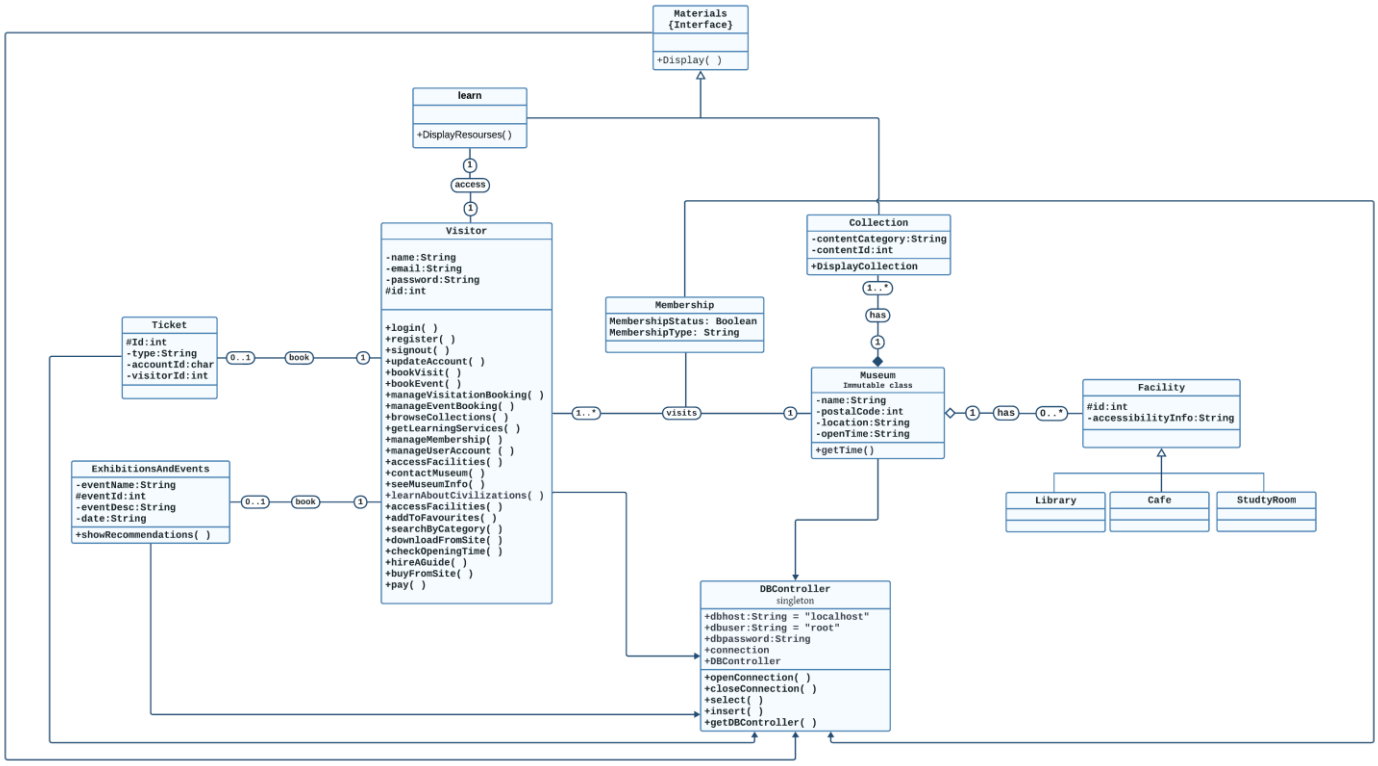
1) An initial version based on the requirements and Use-Case/Activity diagrams.


Materials
DBController
Learn
Collection
access
has
Ticket
has
Visitor
visits
Museum
has
Facility
book
Event
Cafe
Library
studyRoom

2)An intermediate version based on the interaction diagrams



3)A final version after applying the design patterns and other modifications.



12. Three Mandatory Design Patterns Applied *(Including a typed description)*

IMMUTABLE PATTERN:

Context:

An immutable object is an object that has a state that never changes after creation.

Problem:

How do you create a class whose instances are immutable?

Forces:

There must be no loopholes that would allow 'illegal' modification of an immutable object

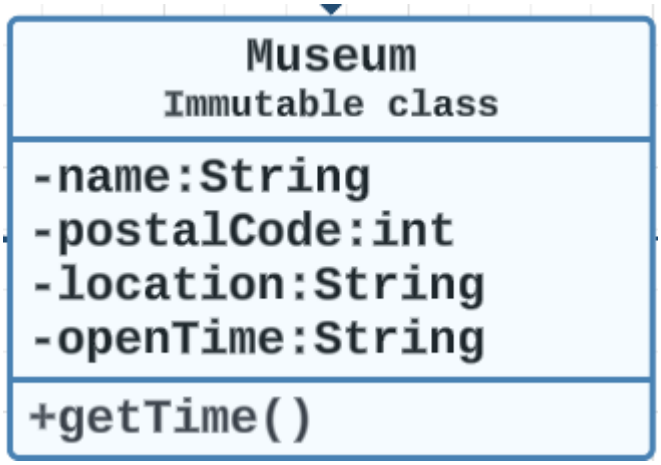
Solution:

- Ensure that the constructor of the immutable class is the only place where the values of instance variables are set or modified.

- Instance methods which access properties must not change instance variables.

Example:

In our system we used IMMUTABLE PATTERN at Museum class to make sure that the museum information is static and avoid that it's information will changes after creation



There's NO Setters!

==========

Delegation PATTERN:

context:

Delegation Design Pattern is all about Aggregation and Composition.

The BookVisit() Method in Visitor class is Used in Ticket class.

problem:

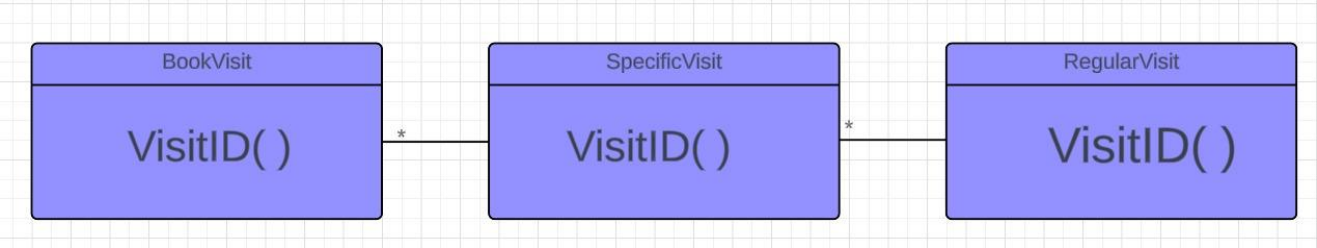
How can we do the aggregation and composition between classes more efficiently.

Forces:

Permissions in the Database Table

Solution:

Use Relational Database and Primary and Foreign Key



SINGLETON PATTERN

Context:

It is very common to find classes for which only one instance should exist (singleton).

Problem:

How do you ensure that it is never possible to create more than one instance of a singleton class. And provide a global point of access to it.

Forces:

The use of a public constructor cannot guarantee that no more than one instance will be created.

The singleton instance must also be accessible to all classes that require it; therefore, it must often be public.

Solution:

- Have the constructor private to ensure that no other class will be able to create an instance of the class singleton.

- Define a public static method, The first time this method is called, it creates the single instance of the class

"singleton" and stores a reference to that object in a static private variable.

Example:

In our system we used SINGLETON PATTERN at DBController class to avoid that no more than one instance will be created for database controller