كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

Software Engineering-2 Cover Sheet

Project Title: Library Management System

Row Number (in PDF): 69 Time: 11.30 Am

	Member Name (pri nted in arabic)	Member Id (pri nted)	Handwritten signature
1	عبدالله محمد سعيد	20210556	
2	أندرو شريف شكري	20210190	
3	كريم عمرو امام حسين	20210687	
4	الاء هشام ممدوح ابراهيم	20210173	
5	تقي محمد أحمد السعدي	20210242	
6	رنا عصام الدين عيسي	20210335	
7	مروة عمر محمد محمود	20210900	

Project Requirements(grades)

SRS	2	
SDD	2	
Validation	3	
OCL	4	
AOP	4	
Microservices	1	
Cloud	4	

20

Helwan University Faculty of Computers
and Artificial Intelligence.

Software Engineering-2

Library Management System

Software Requirement Specifications Document

TABLE OF CONTENTS

1 Introduction,Description of project.

Overall Description 4-5

2 System Requirements Specification .

1 Functional Requirements 6-7

2 Non-Functional Requirements 8-9

3 System Design and Models .

1 Class Diagram 10

2 OCl Diagram 11-15

3 ERD diagram 16

4 Activity Diagram 17-24

5 Sequence Diagram 25-33

6 General Use Case Diagram 34

1.Introduction

This document serves as a comprehensive overview of the Library Management System (LMS), This project manages and stores book information electronically according to user's needs. The system helps both users and library manager to keep a constant track of all the books available in the library. It allows both the admin and the users to search for the desired book. an Advanced software solution designed to streamline the operations of libraries and enhance user experiences. This system aims to automate various library processes, including management, user registration, circulation management. The library system facilitates efficient management of library resources.

This document outlines the system's Requirements (functional and nonfunctional), System architectural design and model (class, ERD, use case, activity and sequence, diagrams), technologies used. It is intended to provide valuable insights for stakeholders, including library administrators, staff, and system developers, guiding them through the system's functionalities, interfaces, constraints, and response mechanisms.

Advantages

The system excludes the use of paperwork by managing all the book information electronically.

Admin can keep updating the system by providing the new books arrival in system and their availability thus Users need s.

The system has books well organized and systematically arranged in different categories in the system so that the user can easily search and find the book.

Thus, it saves human efforts and resources.

User Interface Design:

- The user interface is designed to be user-friendly, and accessible.
- It focuses on simplicity, efficiency, and accessibility to users with diverse needs.

Technologies Used:

i. Frontend: -Html, CSS, JavaScript, Bootstrap.
ii. Backend: - Java spring Boot framework.
iii. Database: - MySQL.

2.System Requirements specifications

- ## Functional Requirements

The LMS must have the following functional requirements:

- The LMS should store all information about librarian and other users (Users and Admin) like their login info, books etc.
- The LMS should store all information about the books and users in two separate Tables.
- The admin must be able to add/remove/manage books, users, category.
- LMS system provides suggestions for the user depending on the user's books have been read.

Users shall be able to: -

-Create an account.

-Login the system.

-Logout from the system.

- Update their profile information (name, password, etc...)

- Delete their account.

- Add book to cart.

-pay by credit card.

-export the payment billing.

-listening to eBook.

Admin shall be able to: -

-login.

-logout.

-Create acount.

-update profile.

-Add user.

-Delete user.

-Add category.

-Delete category.

-Add Book.

-Update Book.

-Delete Book.

- ## Non-Functional Requirements

The system is going to be exposed to large traffic. There can be a huge number of requests at the same time at the system and it should tend to respond to all requests in real-time experience. Also, the system should respond with minimum latency, a user should never wait too much time to get the response

as this is a real-time system experience. To illustrate this, when you search for a video in the system, this system should suggest related videos as soon as possible. The system shall be cost effective, stay consistent and reliable, which means there should not be any data loss. Additionally, service will be durable which means all piece of system should exists until they are

Usability Requirements:

- Our user interface should be interactive, simple and easy to understand. The system should prompt the user and administrator to login to the application for proper input criteria.

Security Requirements:

- System should use secure Database.
- Normal users can just read information, but they cannot edit or modify anything except their personal and some other information.
- System will have different types of users and every user will have access constraints.
- Proper user authentication should be provided.
- No one should be able to hack user's password.
- There should be separate accounts for admin and members such that no member can access the database and only admin has the right to update the database.

Availability

- The site is available for users 24/7.

- Pages that may experience problems must display a notification with a timer showing when the system is going to be up again.

Performance Requirements

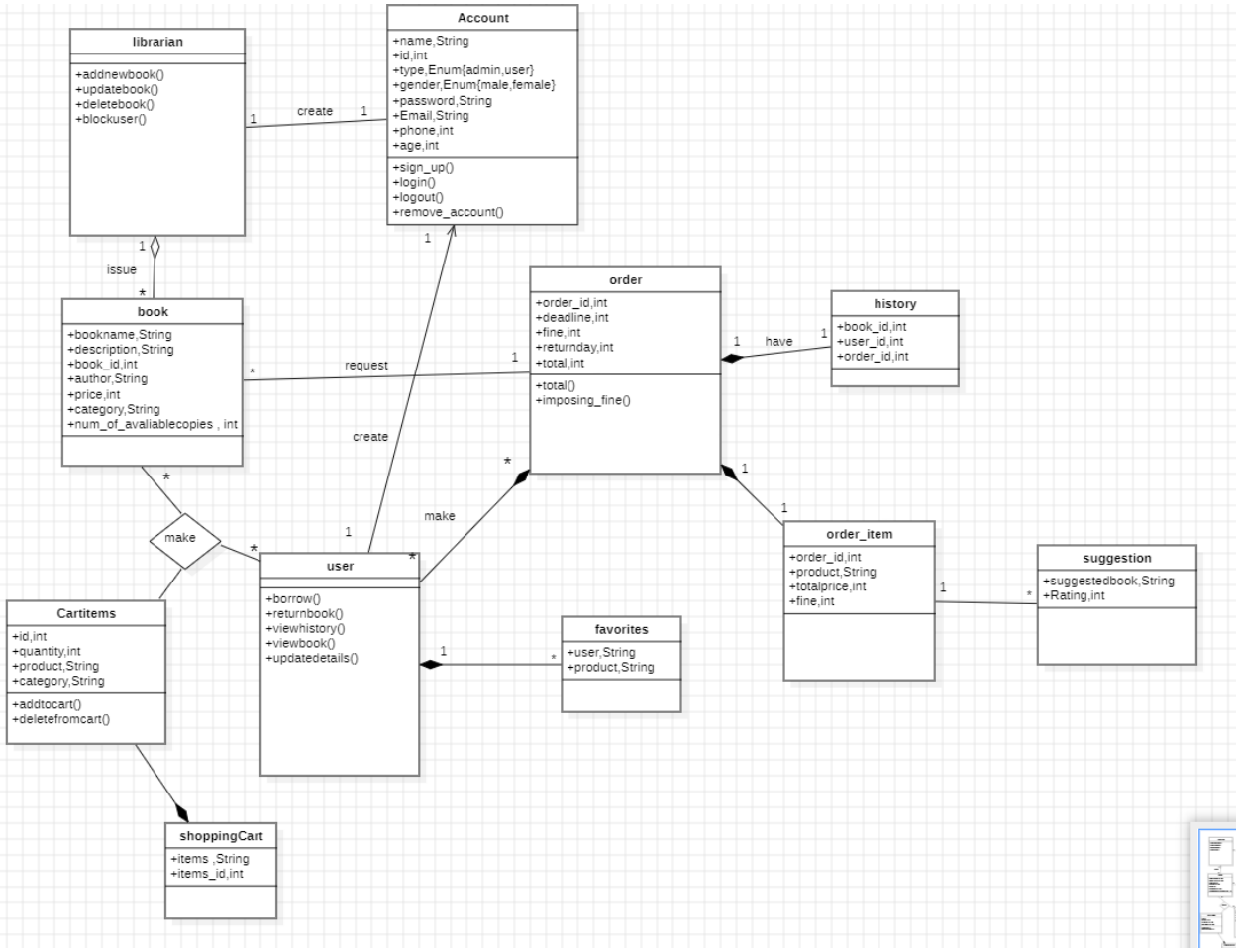
- The system shall accommodate a high number of books and users without any fault.
- Responses to view information shall take no longer than 5 seconds to appear on the screen.

Maintainability

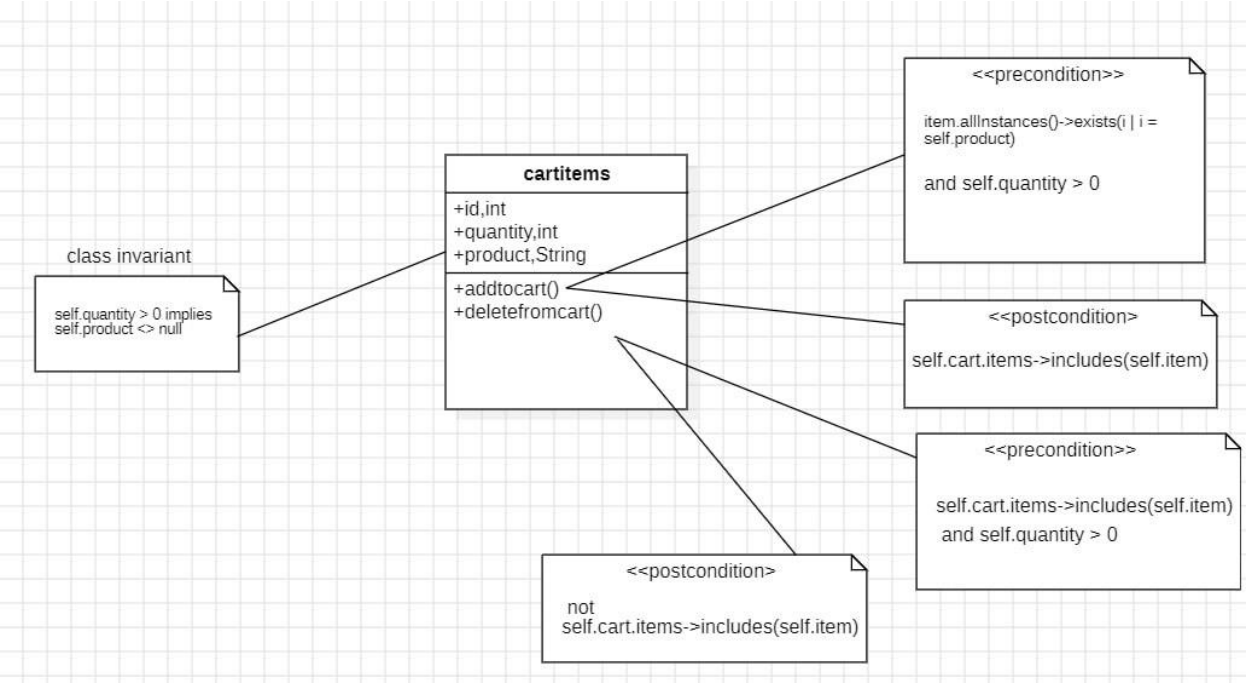
The developers, database administrators and a team of software engineers work on the maintenance and the site.

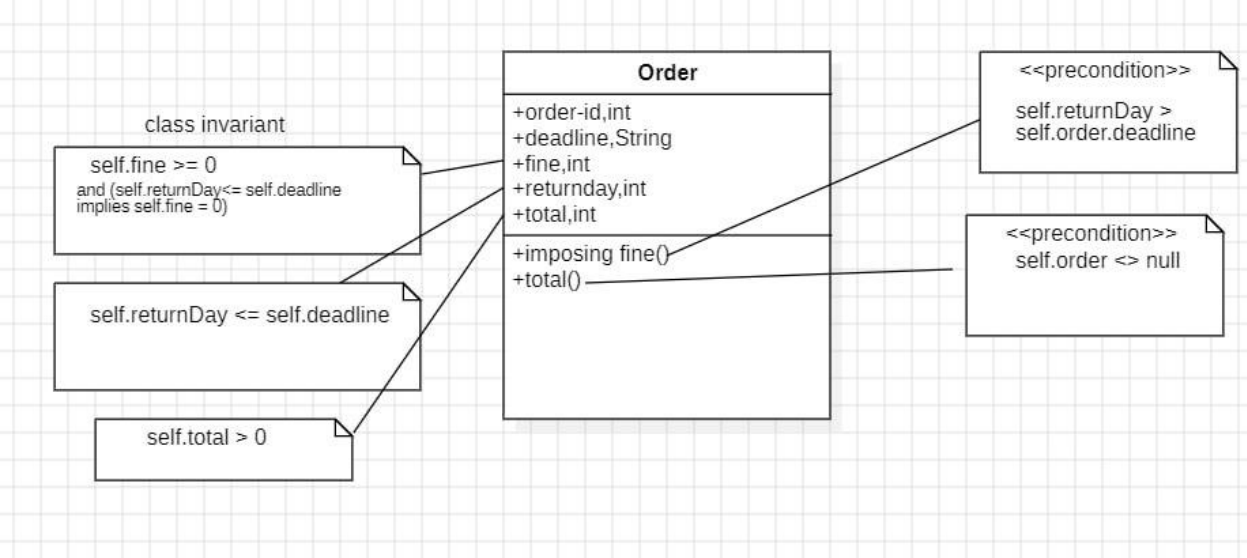
3.System Designs and Models

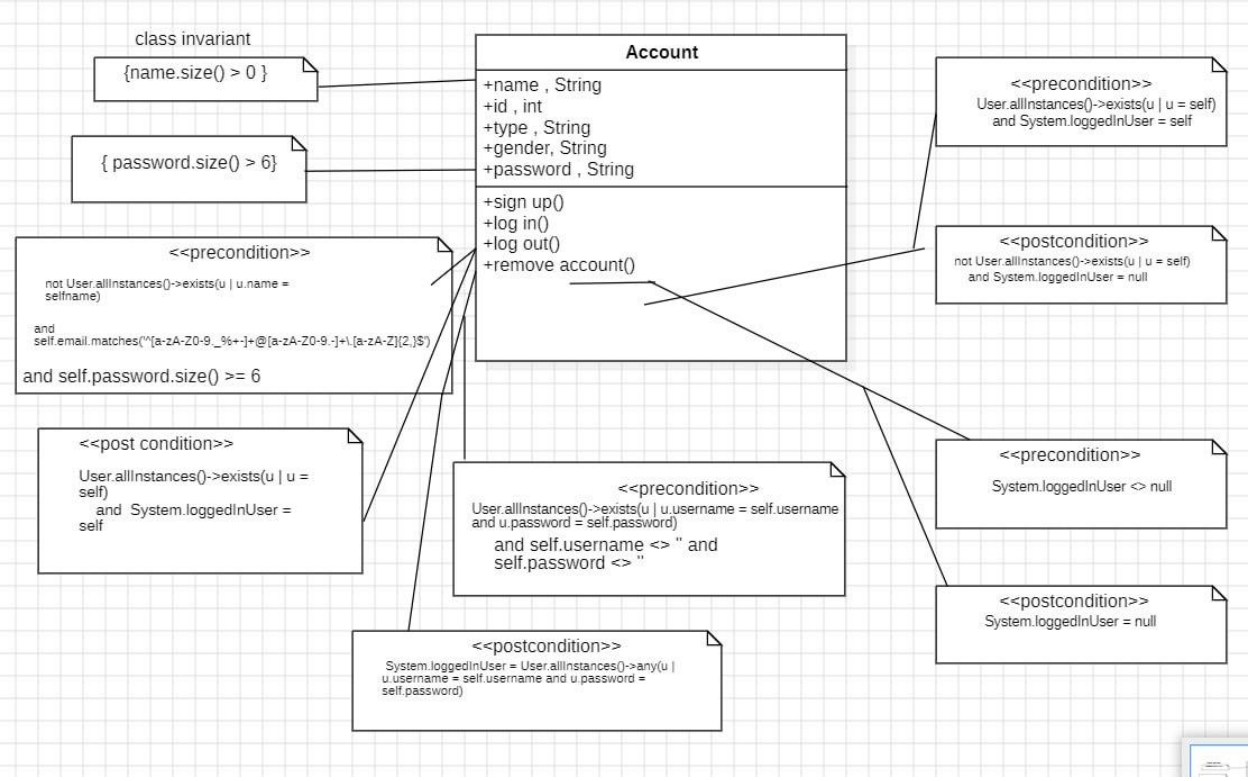
1. Class Diagram

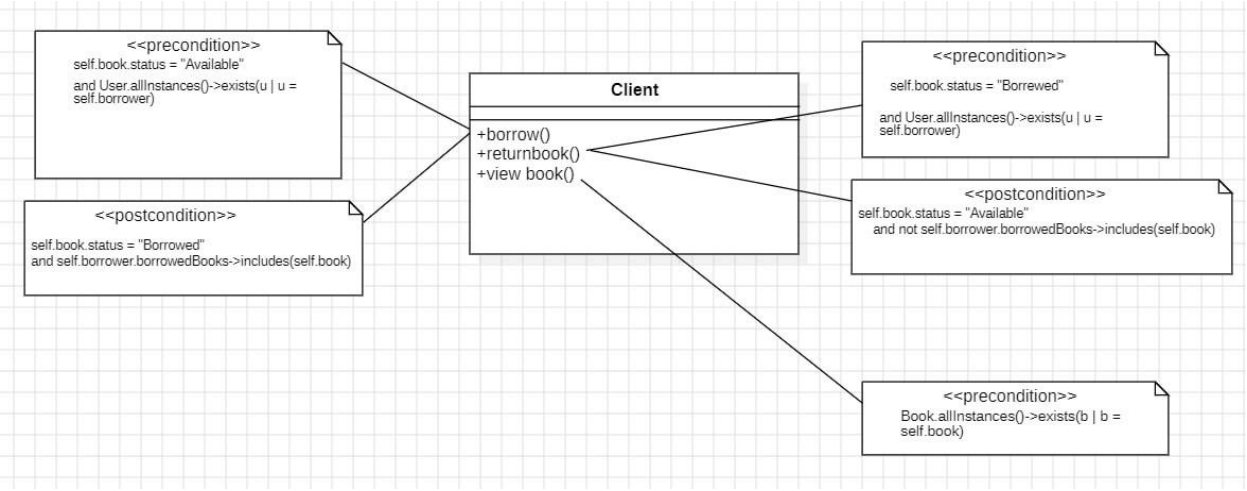


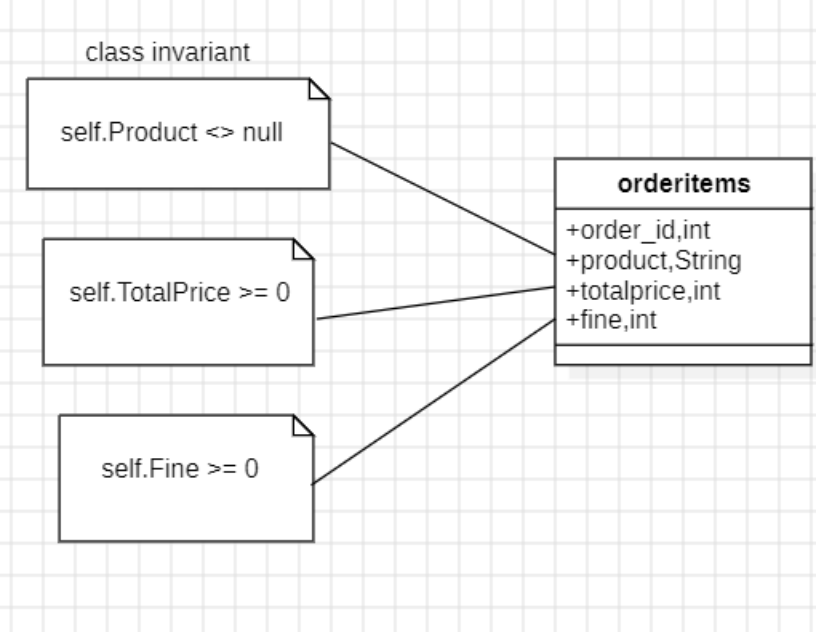
2. OCL Diagram

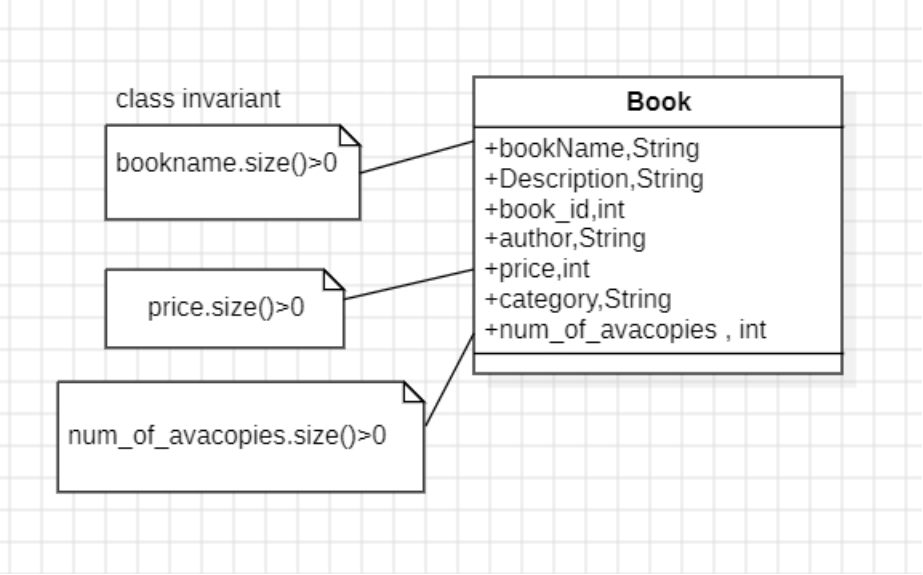


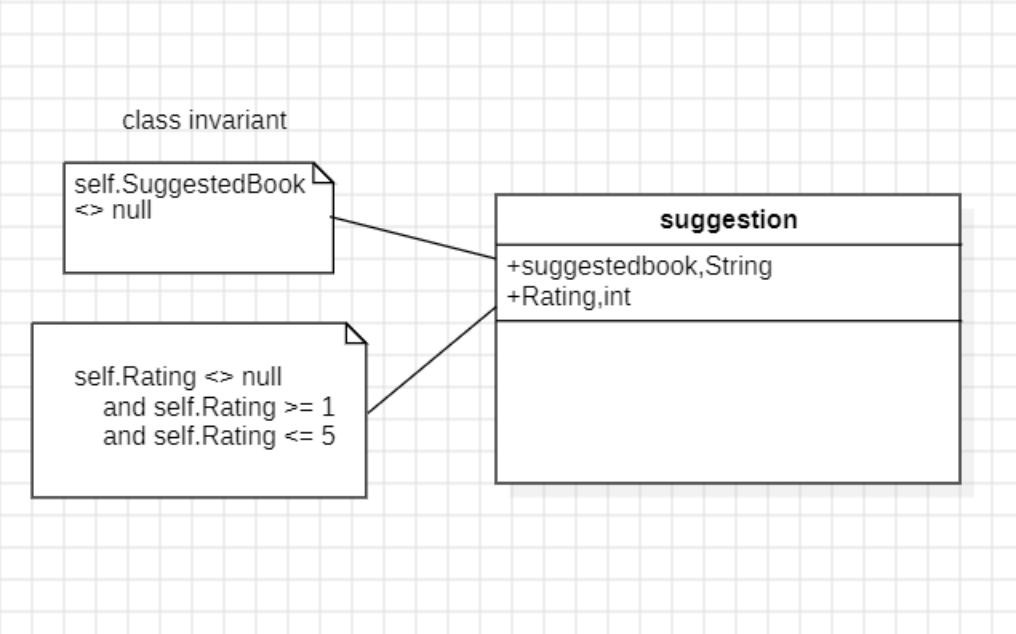

Order
+order-id,int
+deadline,String
+fine,int
+returnday,int
+total,int
+imposing fine()
+total()
class invariant
self.fine >= 0
and (self.returnDay<= self.deadline
implies self.fine = 0)
self.returnDay <= self.deadline
self.total > 0
<<precondition>>
self.returnDay >
self.order.deadline
<<precondition>>
self.order <> null

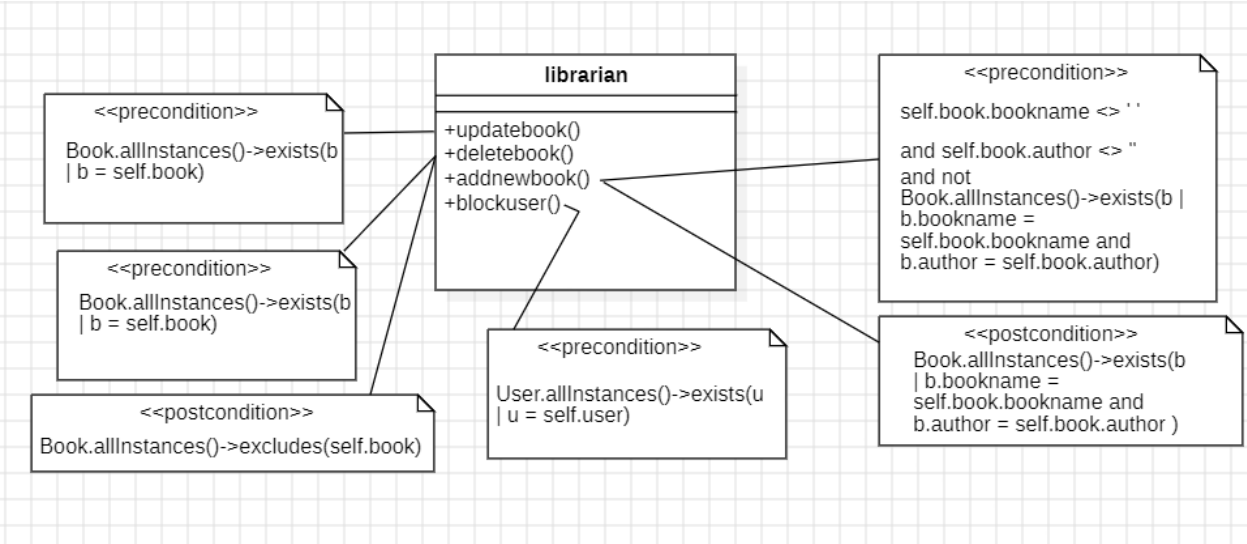

class invariant
{name.size() > 0 }
{ password.size() > 6}
Account
+name , String
+id , int
+type , String
+gender, String
+password , String
+sign up()
+log in()
+log out()
+remove account()
<<precondition>>
User.allInstances()->exists(u | u = self)
and System.loggedInUser = self
<<postcondition>>
not User.allInstances()->exists(u | u = self)
and System.loggedInUser = null
<<precondition>>
not User.allInstances()->exists(u | u.name = selfname)
and
self.email.matches('^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')
and self.password.size() >= 6
<<post condition>>
User.allInstances()->exists(u | u = self)
and System.loggedInUser = self
<<precondition>>
User.allInstances()->exists(u | u.username = self.username and u.password = self.password)
and self.username <> '' and self.password <> ''
<<precondition>>
System.loggedInUser <> null
<<postcondition>>
System.loggedInUser = null
<<postcondition>>
System.loggedInUser = User.allInstances()->any(u | u.username = self.username and u.password = self.password)


<<precondition>>
self.book.status = "Available"
and User.allInstances()->exists(u | u = self.borrower)
<<postcondition>>
self.book.status = "Borrowed"
and self.borrower.borrowedBooks->includes(self.book)
Client
+borrow()
+returnbook()
+view book()
<<precondition>>
self.book.status = "Borrewed"
and User.allInstances()->exists(u | u = self.borrower)
<<postcondition>>
self.book.status = "Available"
and not self.borrower.borrowedBooks->includes(self.book)
<<precondition>>
Book.allInstances()->exists(b | b = self.book)

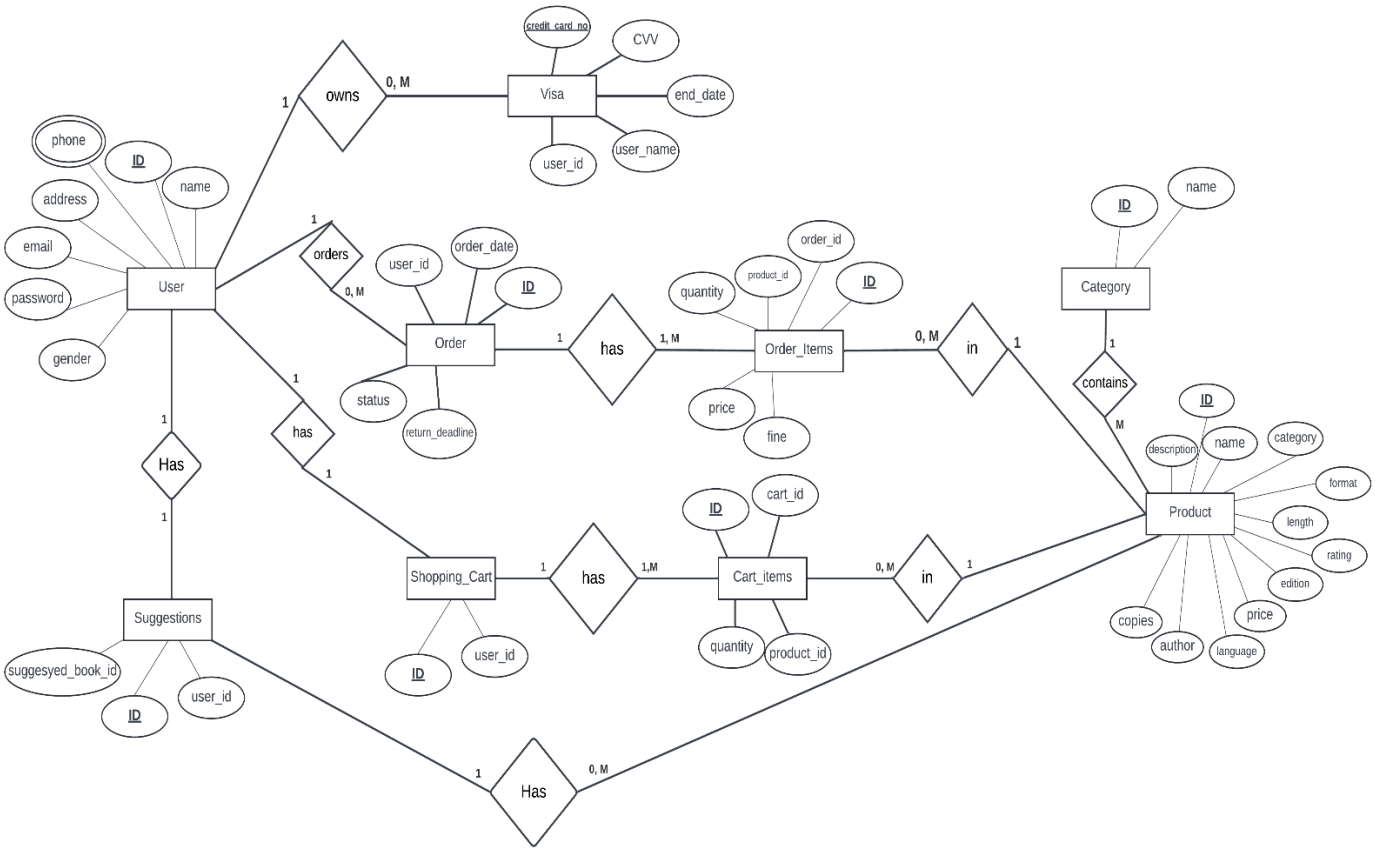

class invariant
self.Product <> null
self.TotalPrice >= 0
self.Fine >= 0
orderitems
+order_id,int
+product,String
+totalprice,int
+fine,int


class invariant
bookname.size()>0
price.size()>0
num_of_avacopies.size()>0
Book
+bookName,String
+Description,String
+book_id,int
+author,String
+price,int
+category,String
+num_of_avacopies , int


class invariant
self.SuggestedBook <> null
suggestion
+suggestedbook,String
+Rating,int
self.Rating <> null
and self.Rating >= 1
and self.Rating <= 5

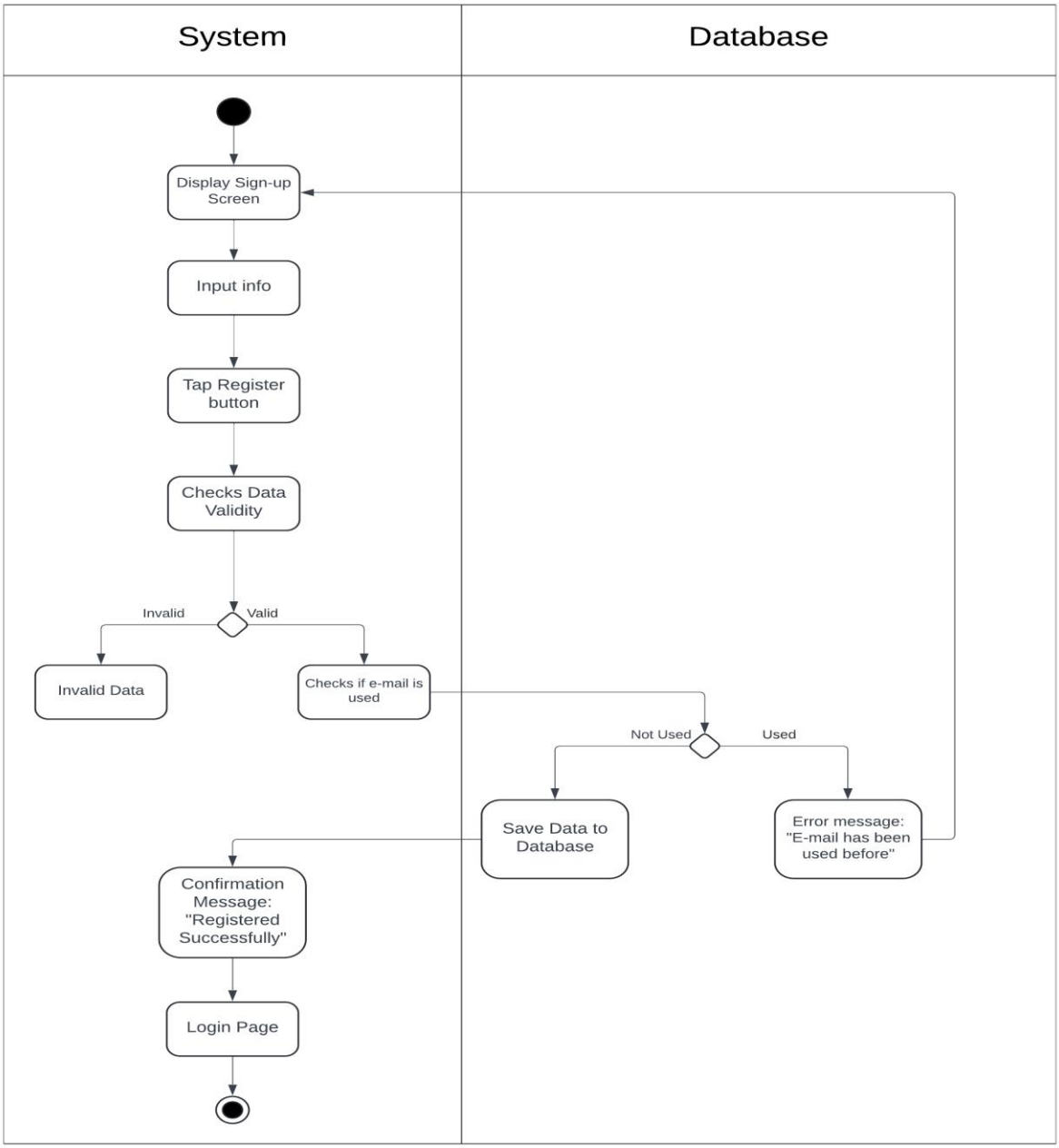

librarian
+updatebook()
+deletebook()
+addnewbook()
+blockuser()
<<precondition>>
Book.allInstances()->exists(b | b = self.book)
<<precondition>>
Book.allInstances()->exists(b | b = self.book)
<<postcondition>>
Book.allInstances()->excludes(self.book)
<<precondition>>
User.allInstances()->exists(u | u = self.user)
<<precondition>>
self.book.bookname <> ' '
and self.book.author <> ''
and not Book.allInstances()->exists(b | b.bookname = self.book.bookname and b.author = self.book.author)
<<postcondition>>
Book.allInstances()->exists(b | b.bookname = self.book.bookname and b.author = self.book.author)

3. Erd Diagram

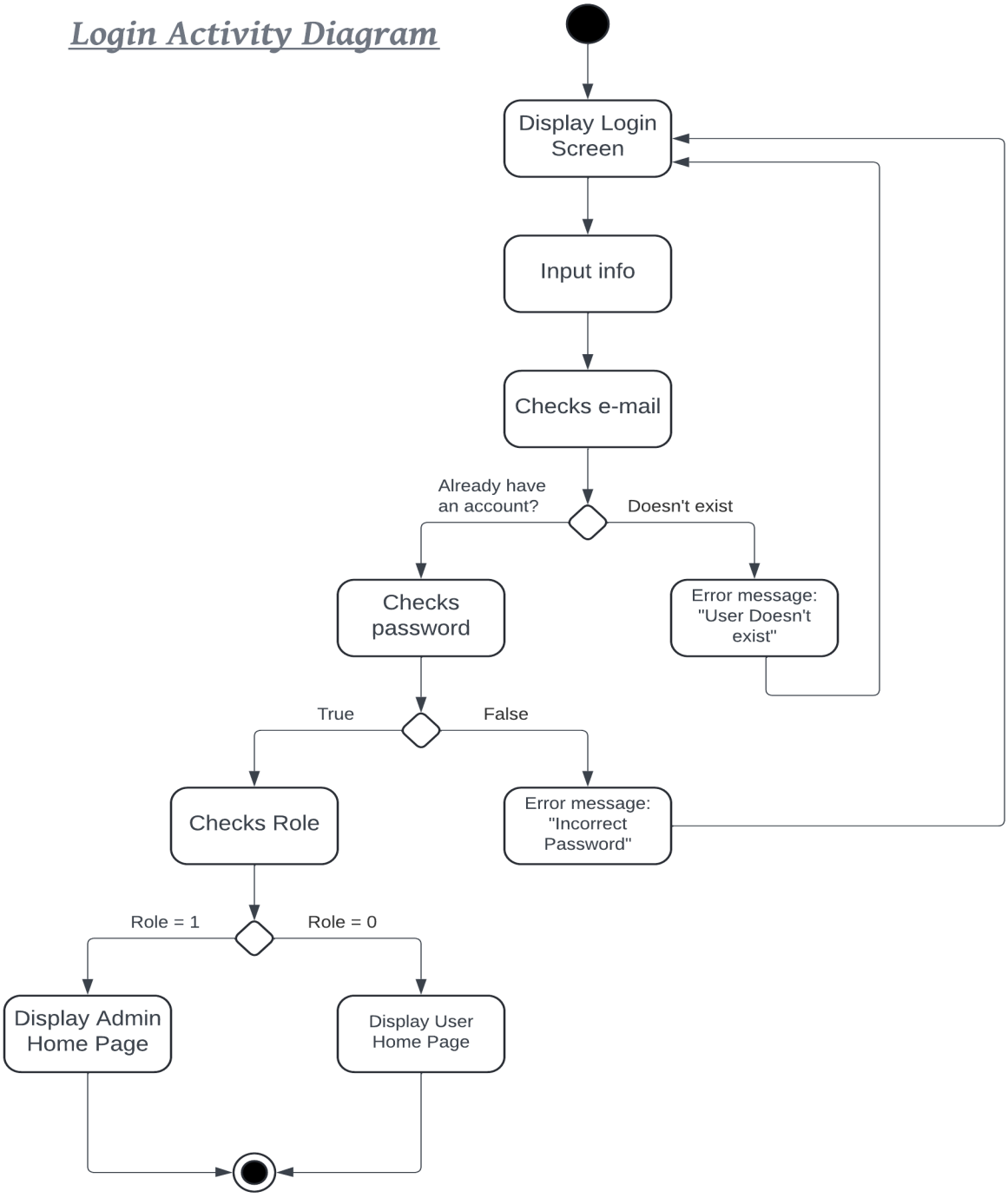


4. Activity Diagram

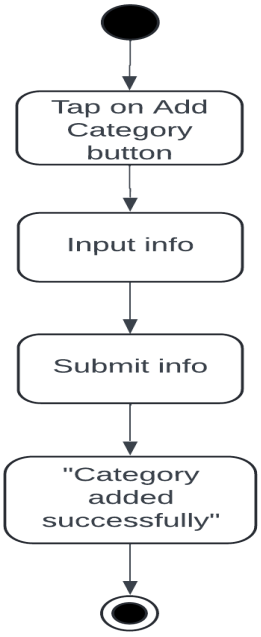
Sign-up Activity Diagram



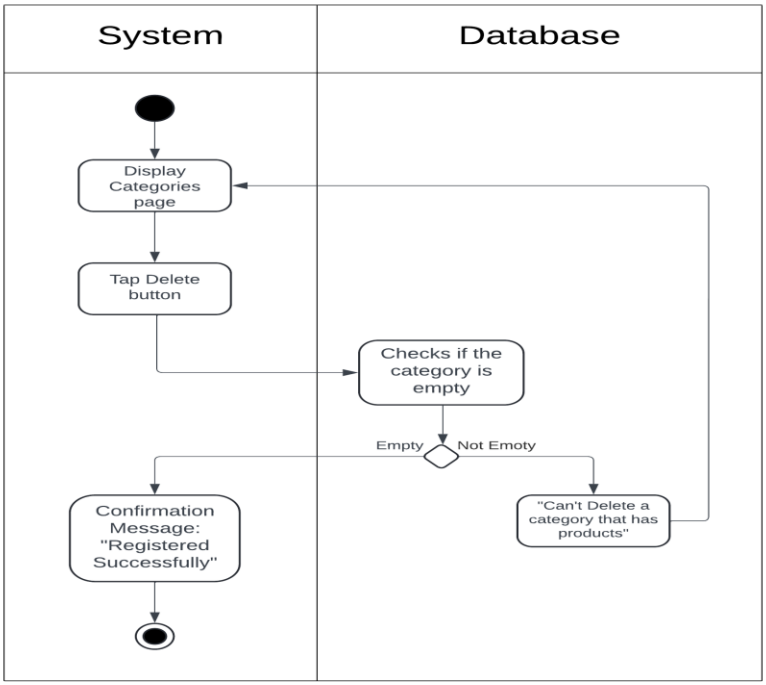
Login Activity Diagram



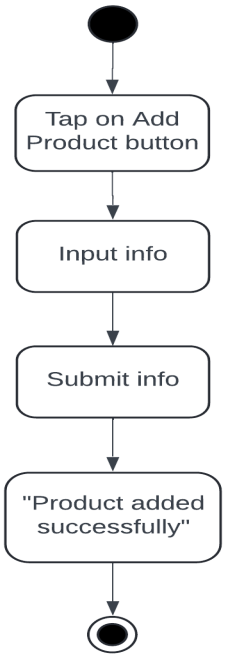
Add Category Activity Diagram



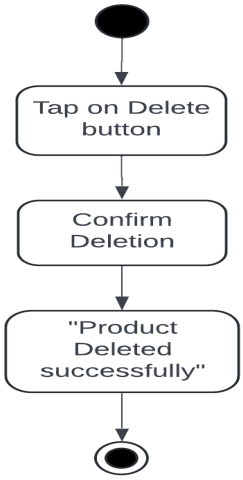
Delete Category Activity Diagram



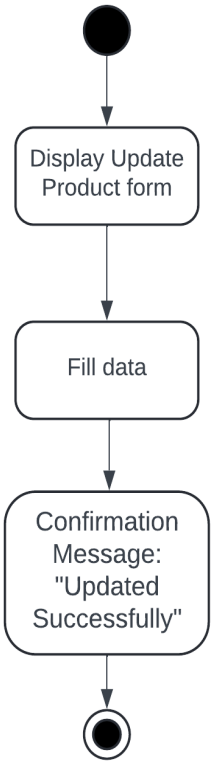
Add Product Activity Diagram



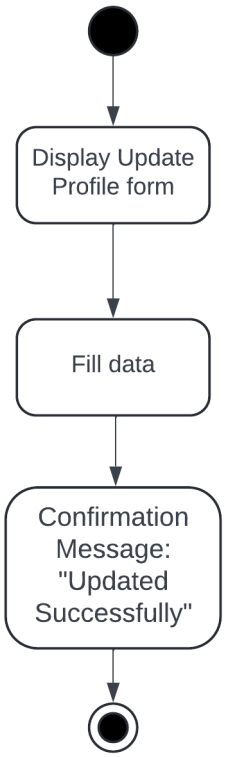
Delete Product Activity Diagram



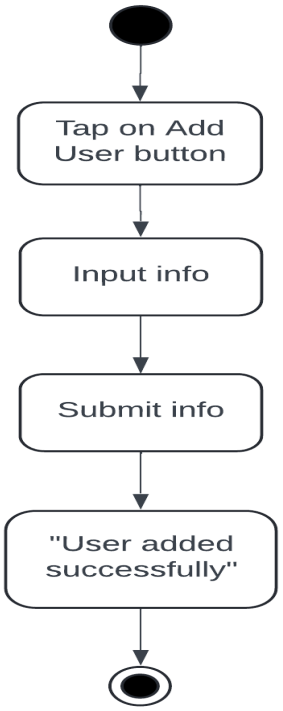
-Update book



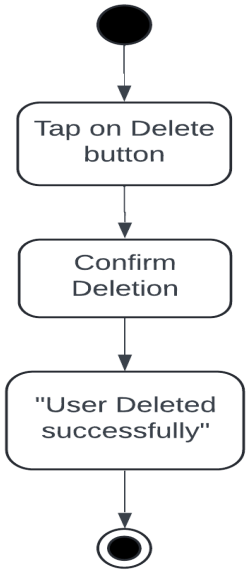
-Update User



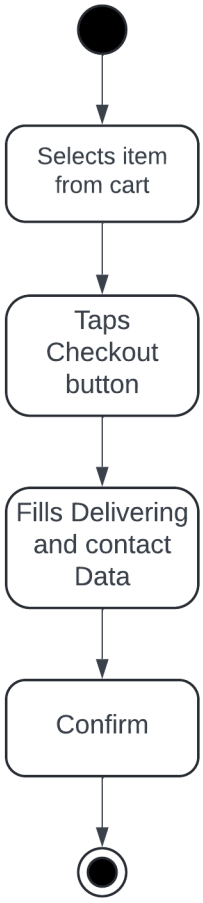
Add User Activity Diagram



Delete User Activity Diagram

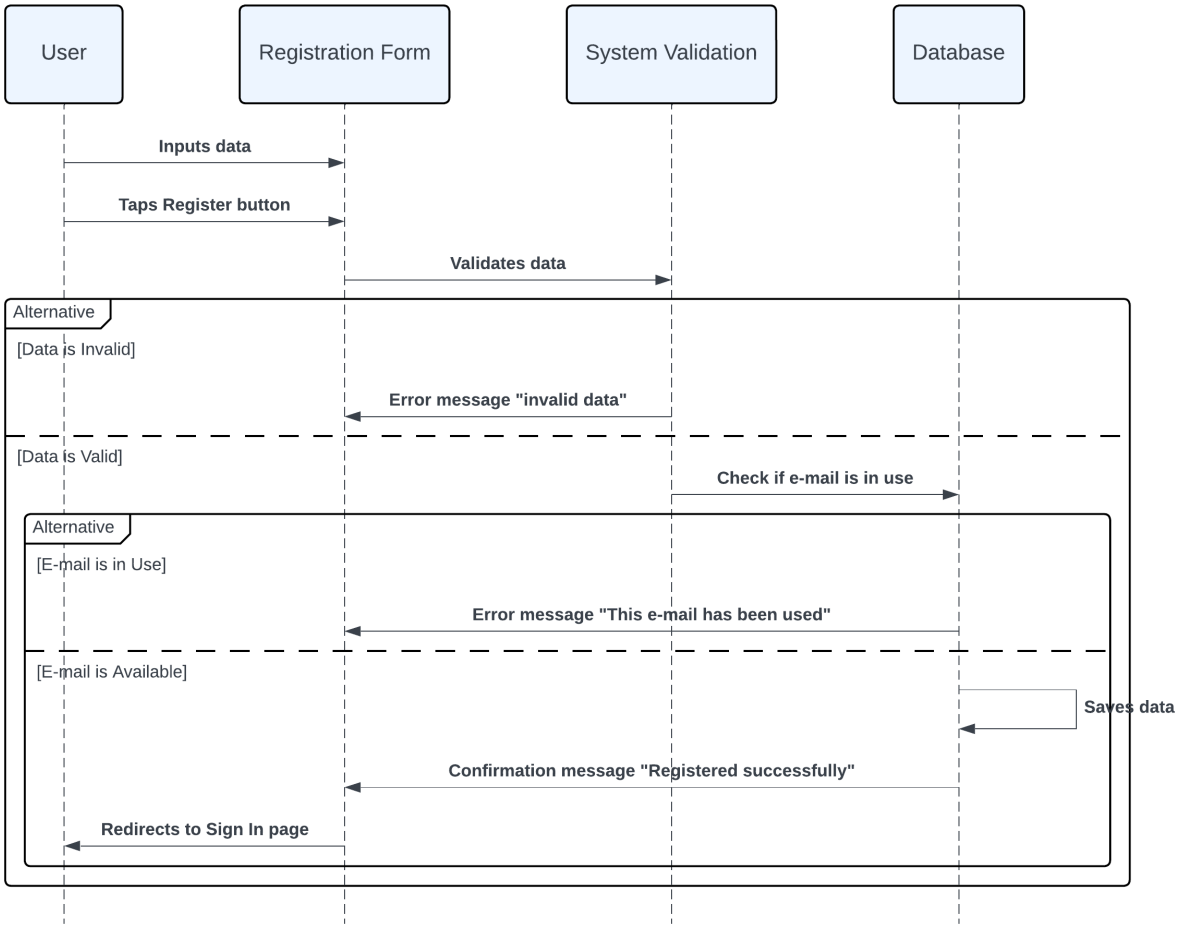


Checkout Activity Diagram

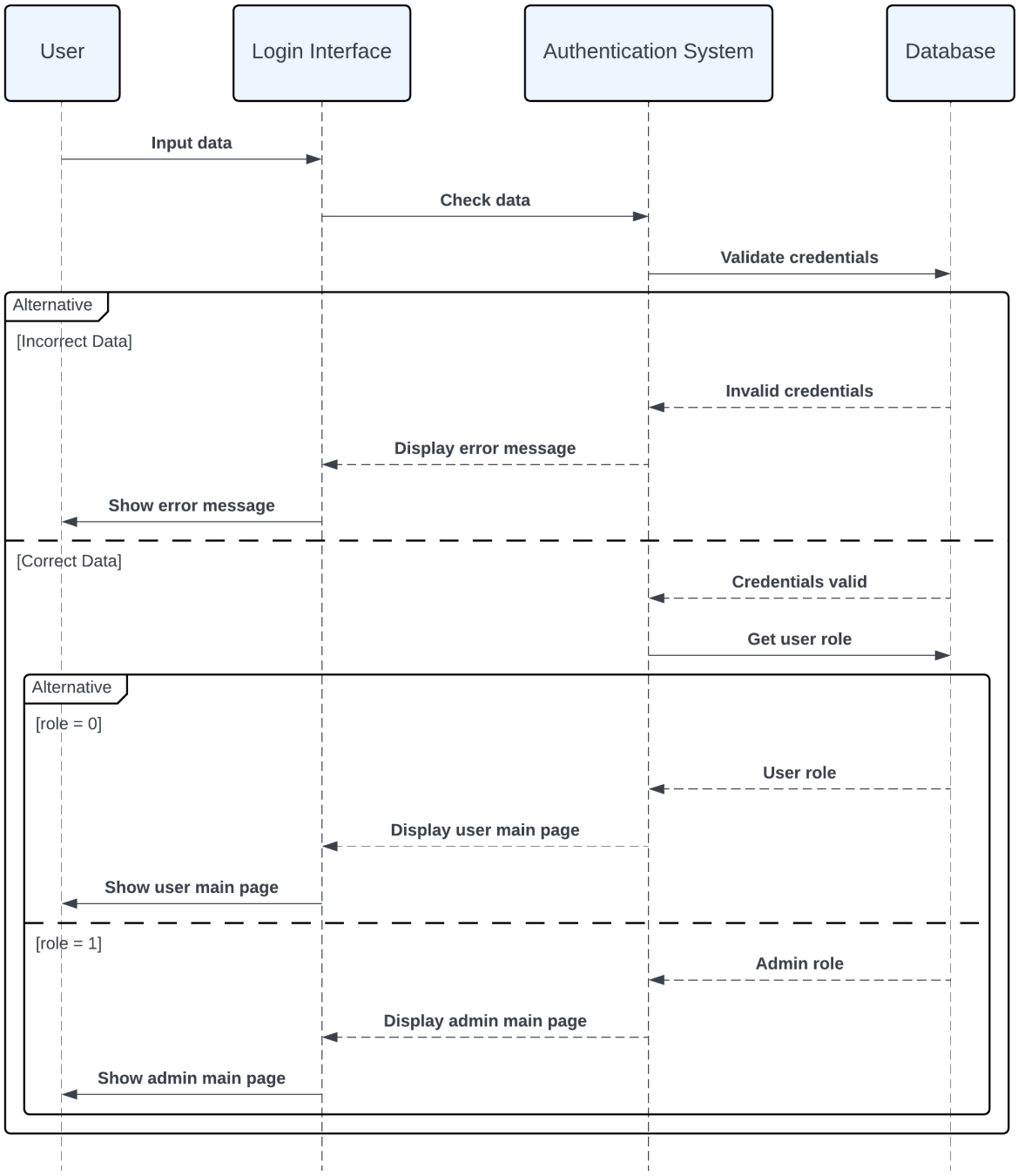


5. Sequence Diagram

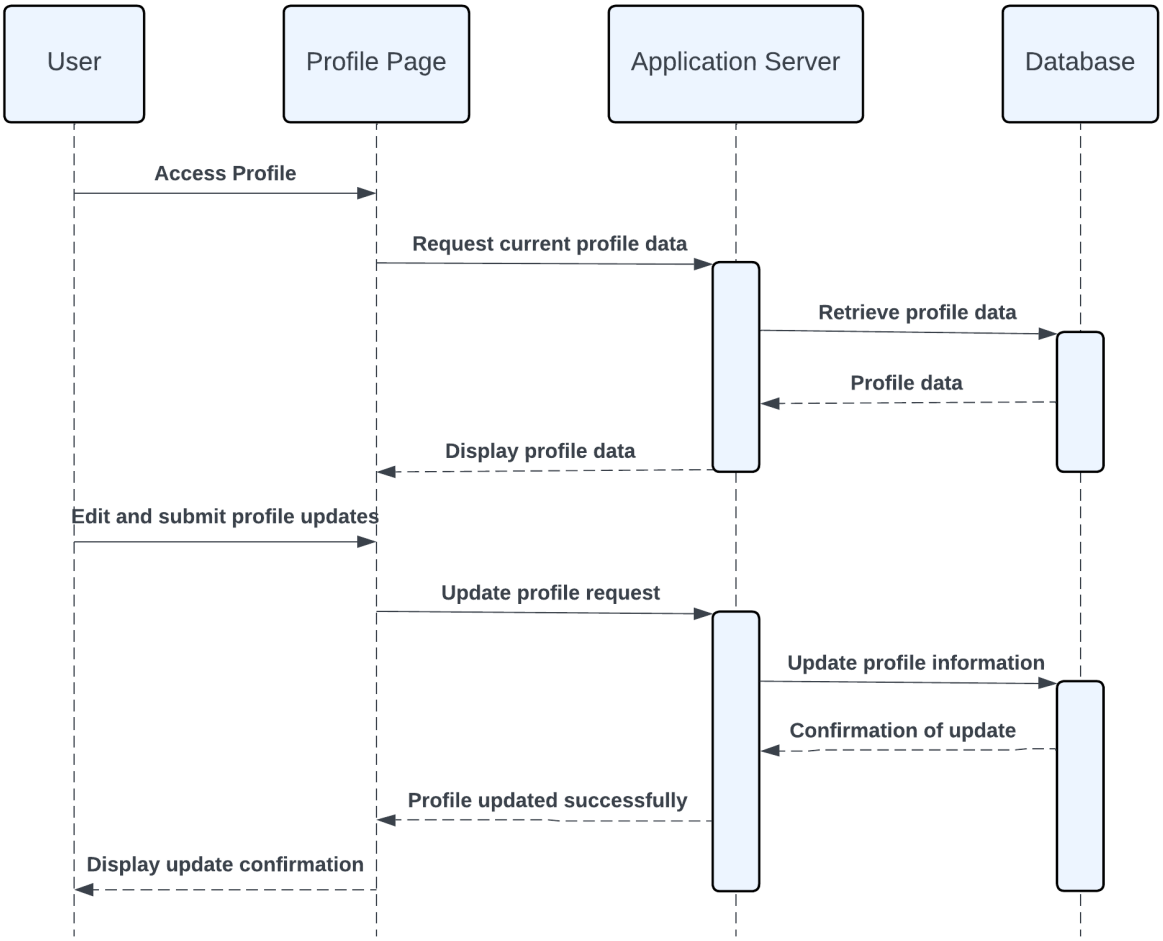
1. Sign Up



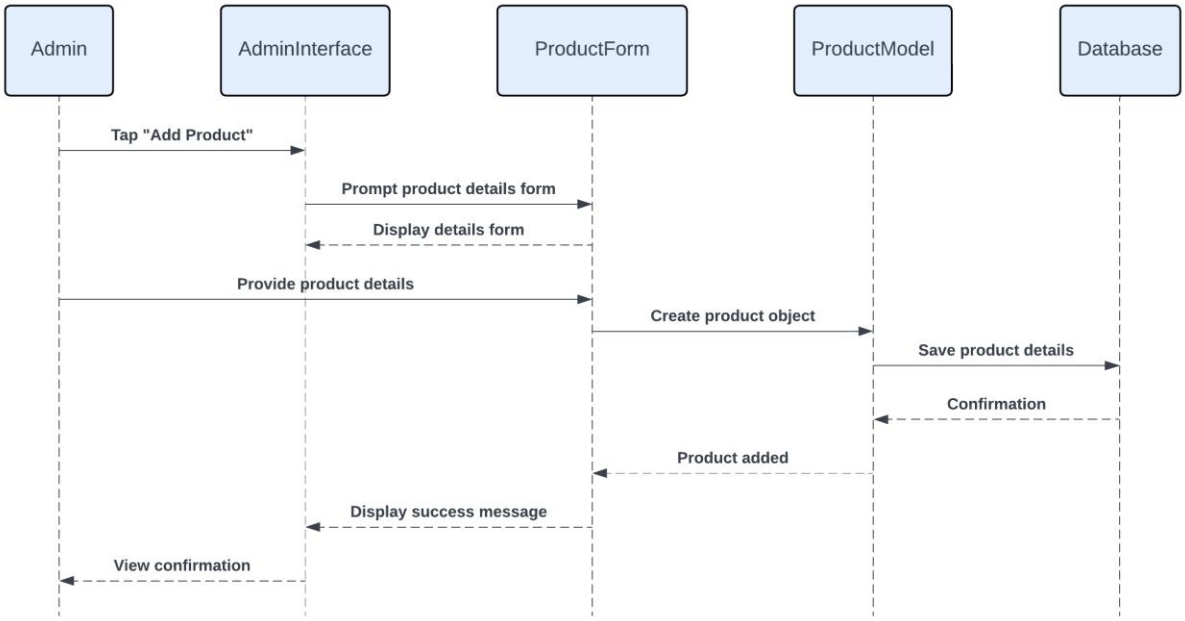
2. Sign In


User
Login Interface
Authentication System
Database
Input data
Check data
Validate credentials
Alternative
[Incorrect Data]
Invalid credentials
Display error message
Show error message
[Correct Data]
Credentials valid
Get user role
Alternative
[role = 0]
User role
Display user main page
Show user main page
[role = 1]
Admin role
Display admin main page
Show admin main page

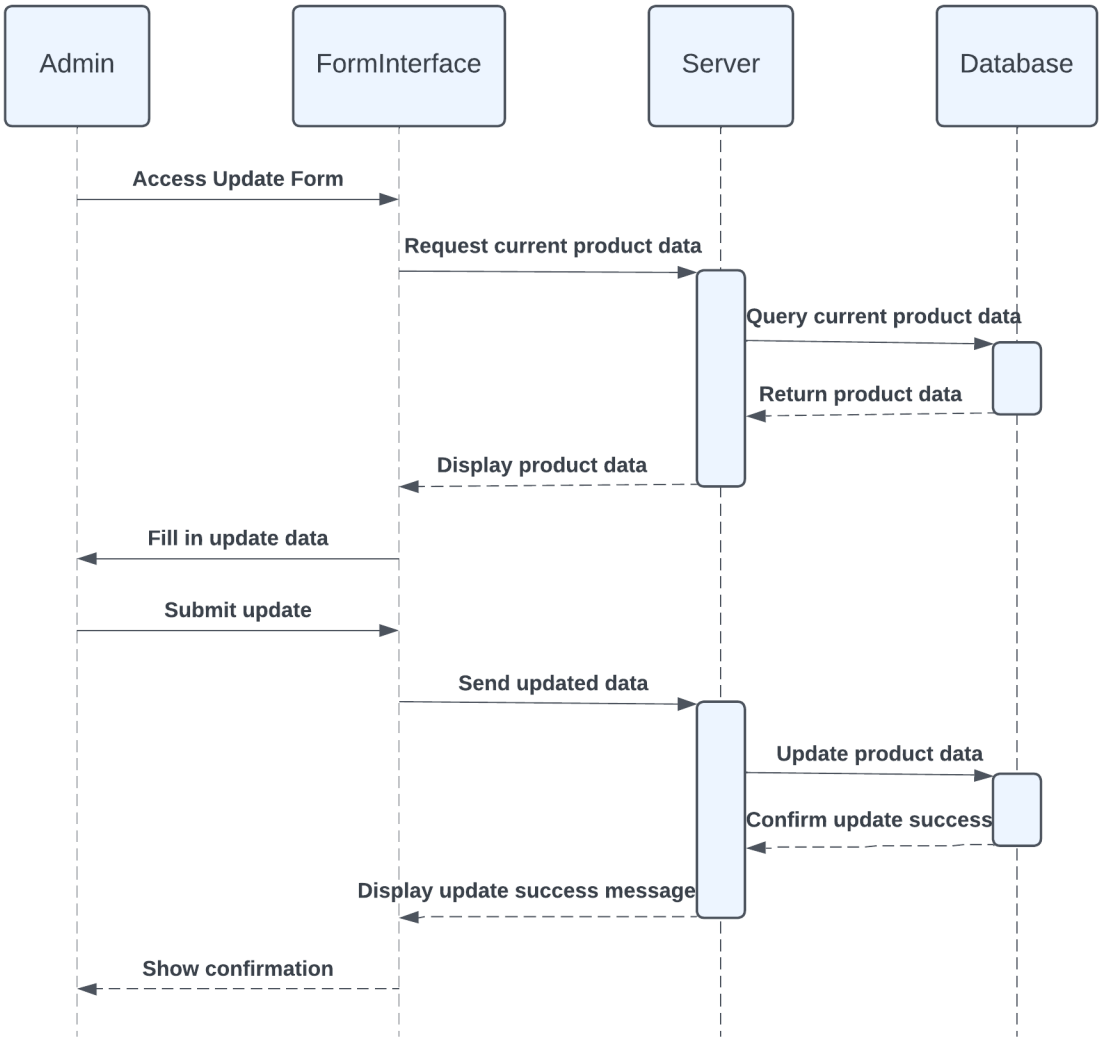
3.Update Profile



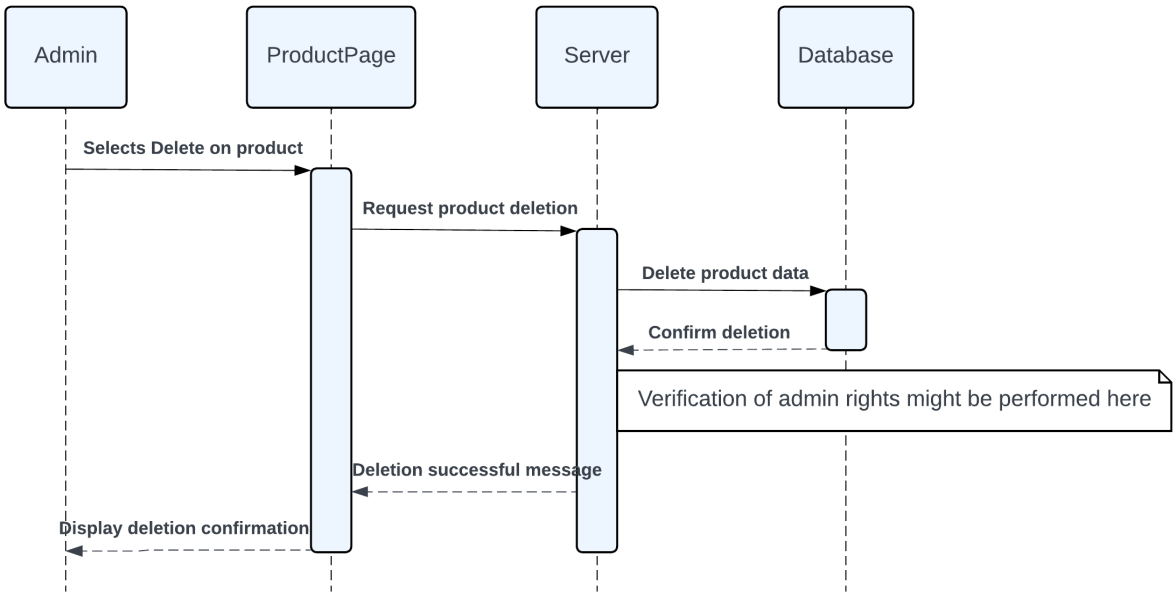
4. Add Product



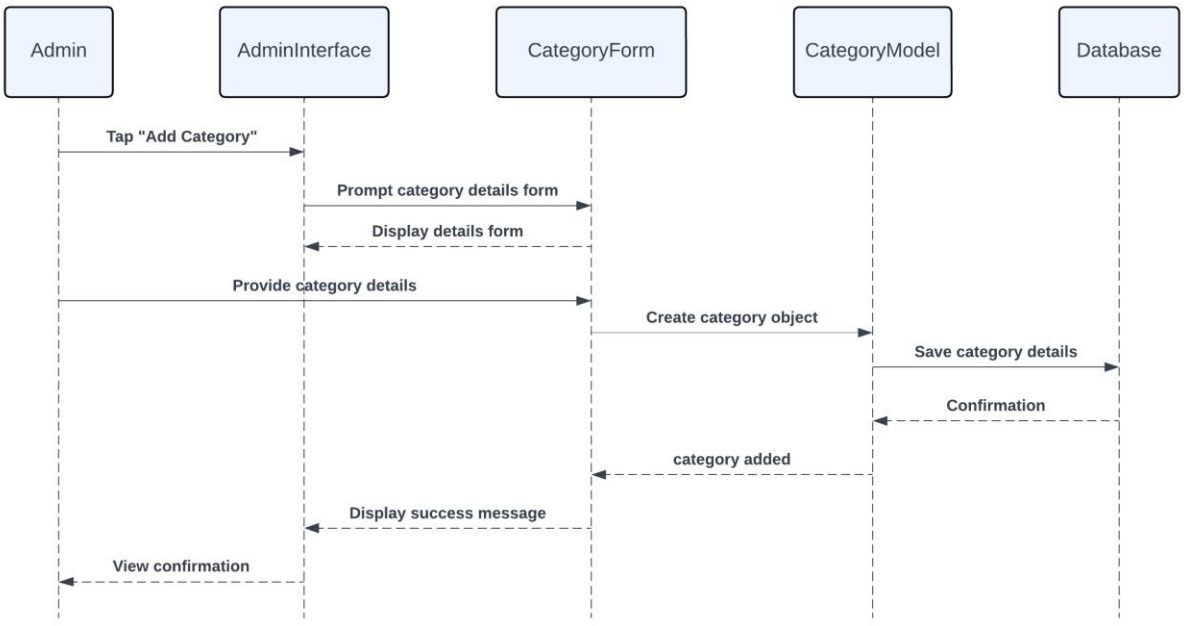
5. Update Product



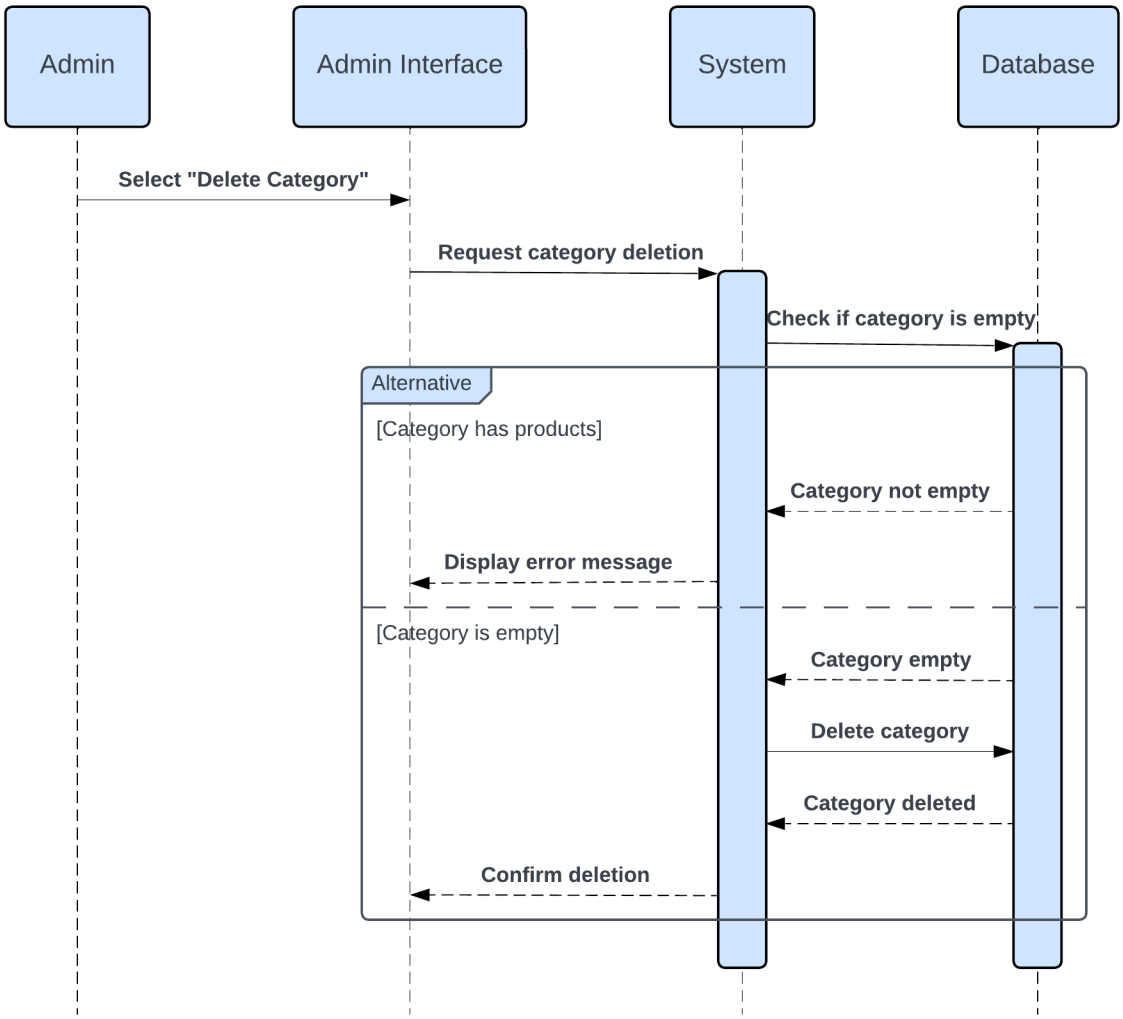
6. Delete Product



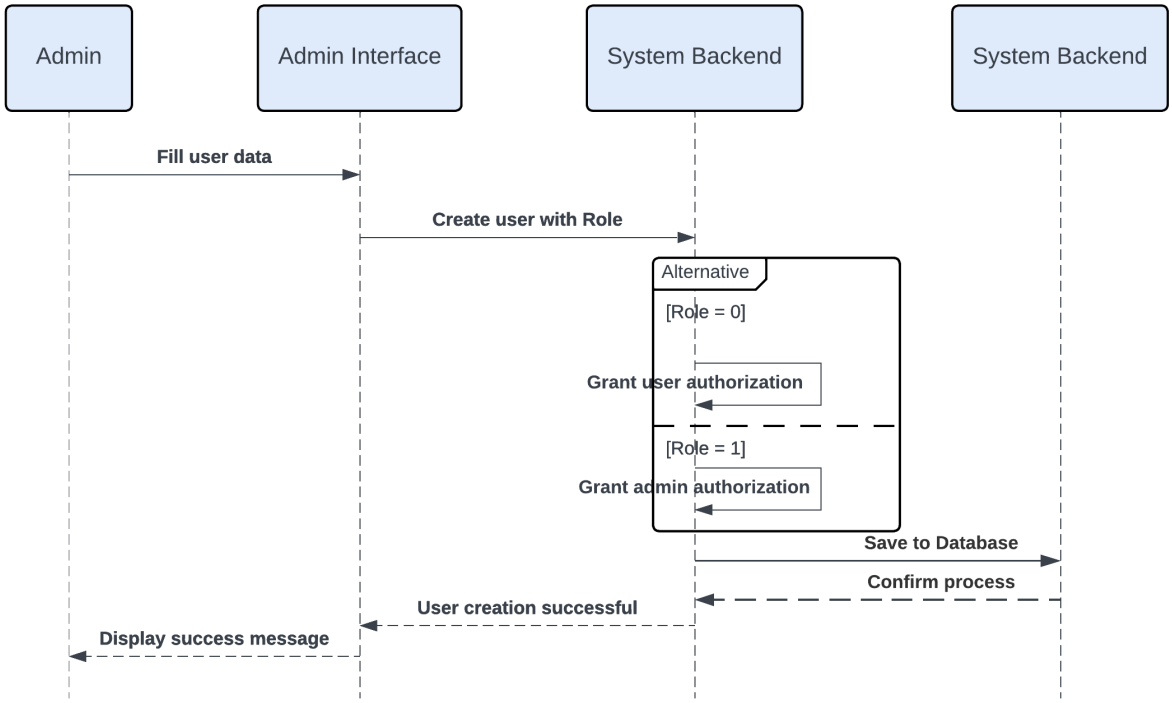
7. Add Category



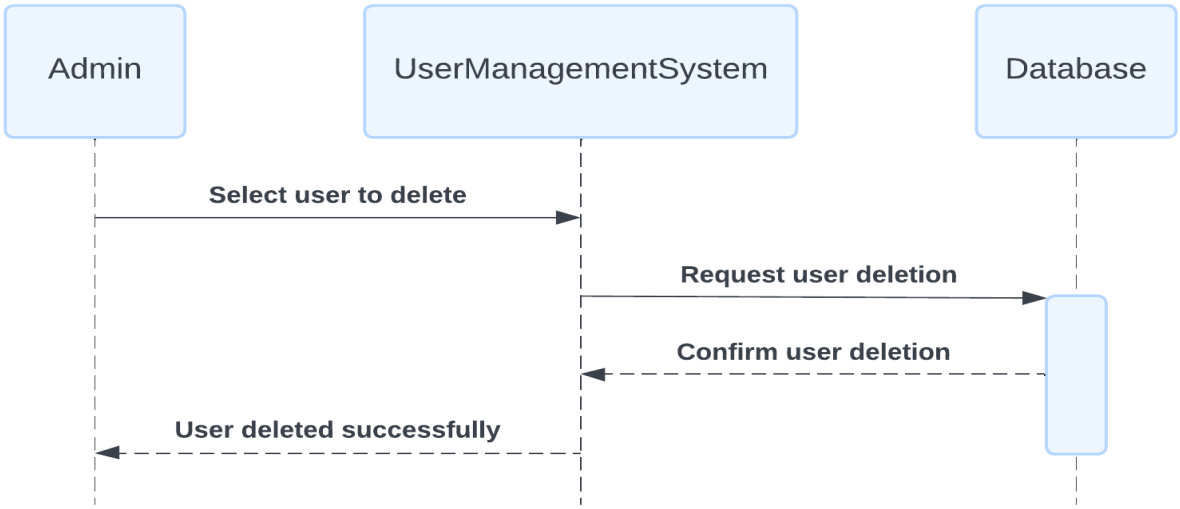
8. Delete Category



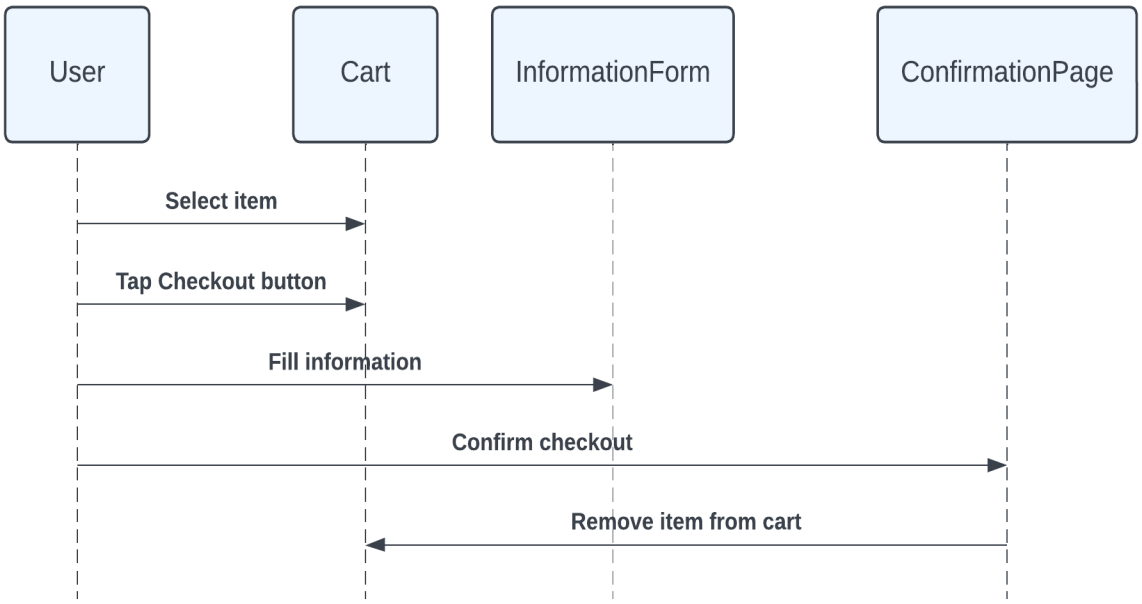
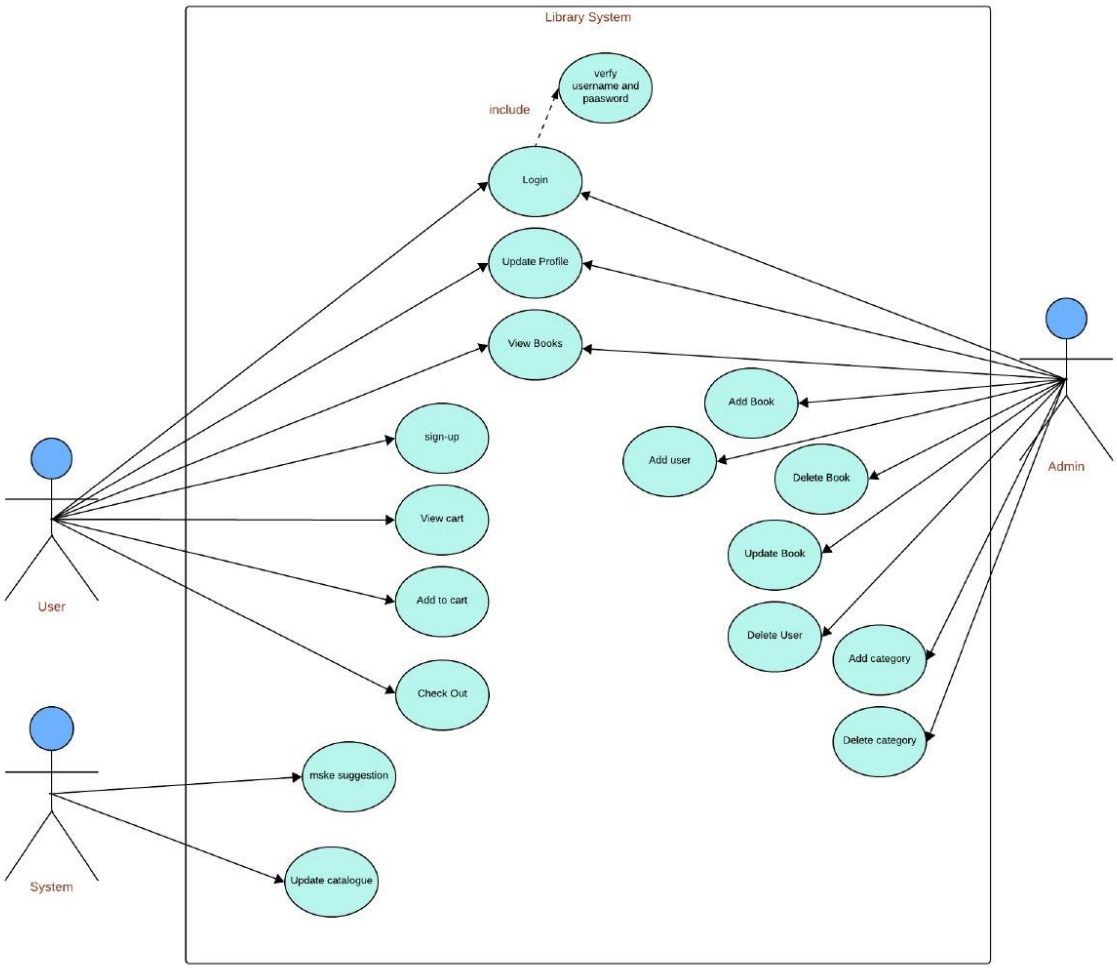
9. Create User



10. Delete User



11. Check Out


User
Cart
InformationForm
ConfirmationPage
Select item
Tap Checkout button
Fill information
Confirm checkout
Remove item from cart

6. Use Case Diagram