Addis Ababa Institute of Technology

School of Information Technology and Engineering

Tourism Management System

Software Design Specification

Team Members

NAME	ID	SECTION
ABEL NIGUS	UGR/8843/13	2
HANA GUTA	UGR/2919/13	2
MIHIRET SHIMELS	UGR/2221/13	2
NETSANET TEWODROS	UGR/3048/13	2
ROBEL TESFAYE	UGR/8429/13	1

Date:27/12/2023

Table of contents

List of Tables

List of figures

1. Introduction

1.1 Purpose

This document outlines the system design for a tourism management system. The system is designed to provide a comprehensive solution for managing the various aspects of a tourism business, such as customer(visitor) management,package management, and booking management. The system will be used by tourism businesses to manage their operations and provide a better customer experience.

1.2 General Overview

TOUR.ET(Tourism management system) is a comprehensive system designed to manage the operations of a tourism business. It is designed to provide a comprehensive view of the business, from package management to customer relation. The system is organised into modules that cover different aspects of the business, such as customer relation, booking, payment, and operation. Each module is designed to provide the necessary tools and information to manage the business effectively. The system also provides a centralised platform for data storage and analysis, allowing for better decision-making. The system is designed to be user-friendly and intuitive, allowing for easy access to the necessary information.

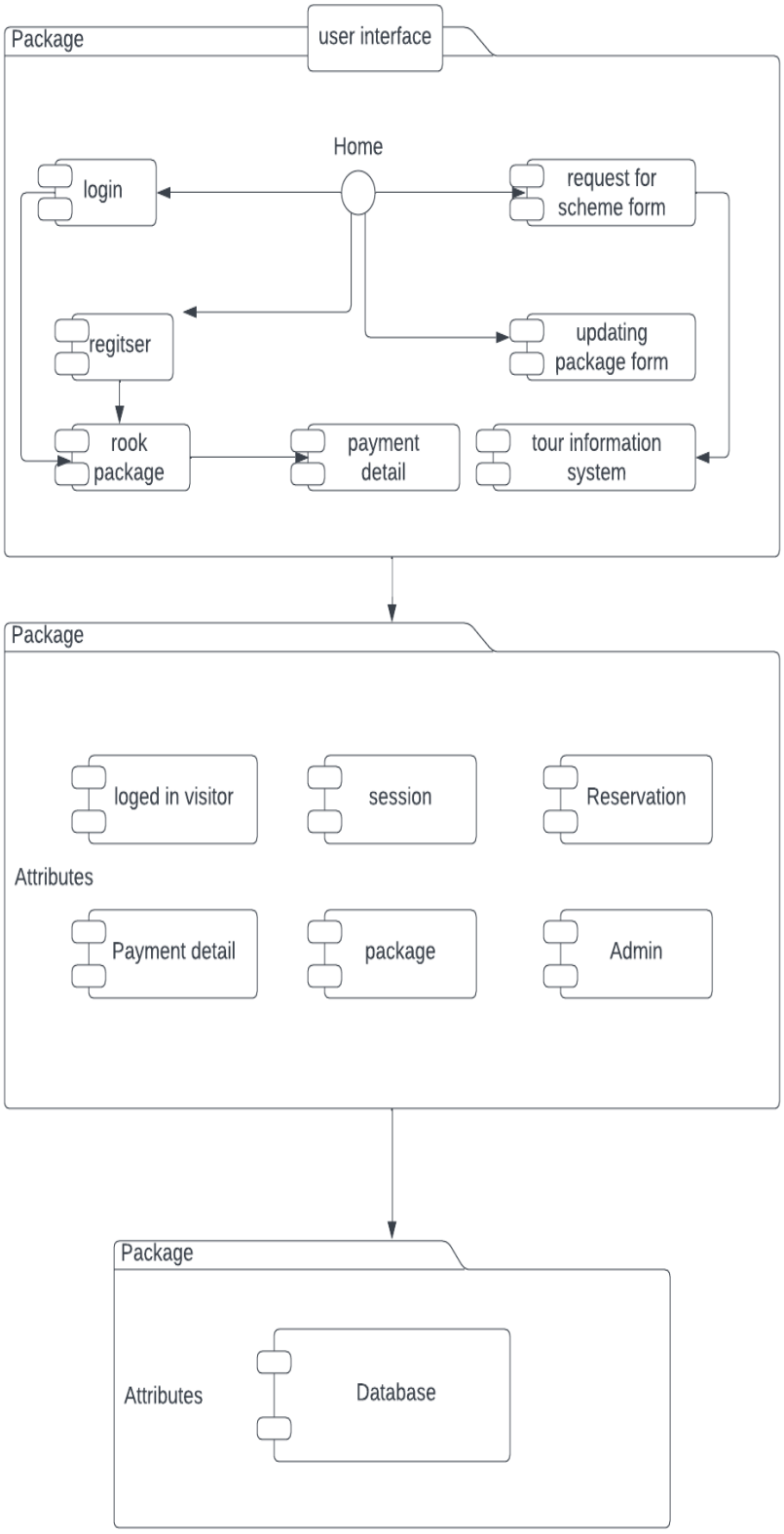
It's software architecture designed to provide a comprehensive solution for managing the operations of a tourism business. The system will be built using a three-tier architecture, consisting of a presentation layer, a business layer, and a data access layer. The presentation layer will be responsible for displaying the user interface and handling user interactions. The business logic layer will be responsible for processing user requests and performing business log. The data access layer will be responsible for connection to the database and performing data operations. It is designed to provide a unified platform for managing all aspects of the tourism business. The system is designed to be highly scalable and customizable, allowing businesses to tailor the system to their specific needs. The goal of it is to provide a comprehensive, integrated solution for managing the operations of a tourism business.

1.3 Development Methods & Contingencies

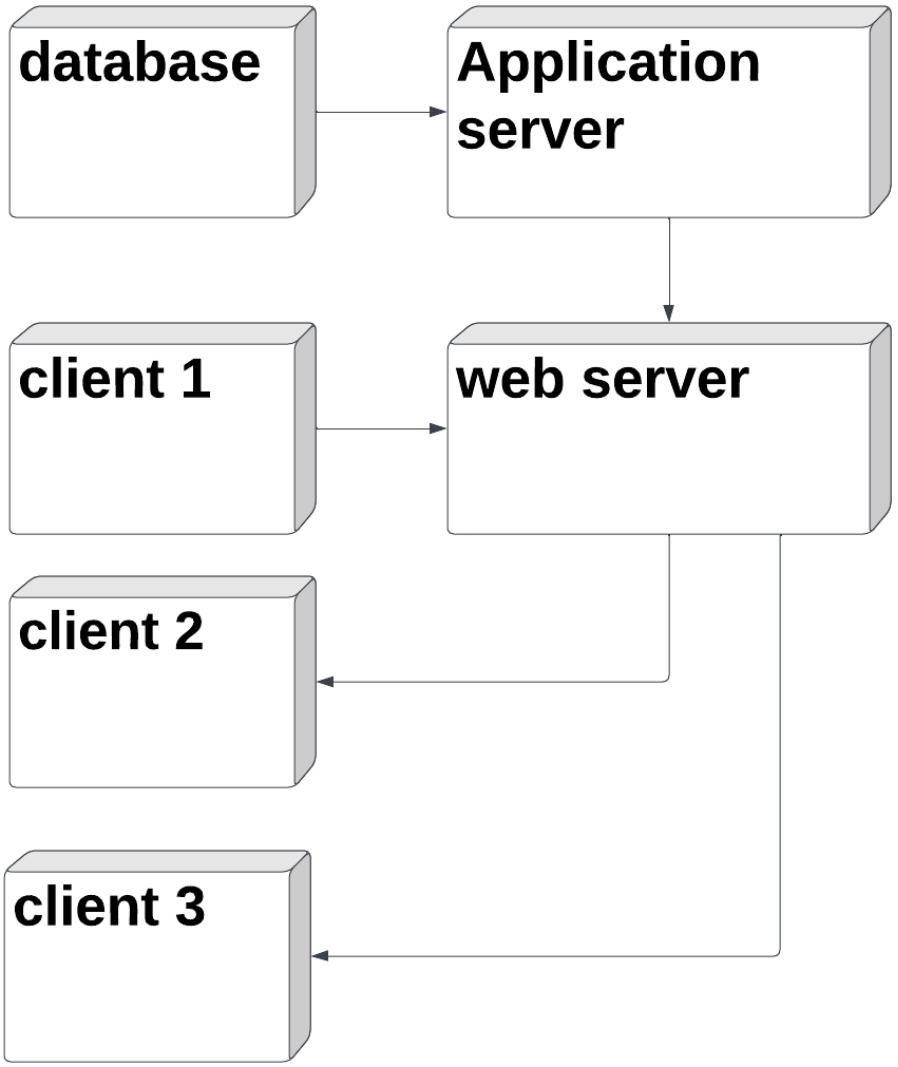
The method used for the system and software design of the TOUR.ET is UML(Unified Modeling Language). This method is used to create a visual representation of the system and its components. It is used to model the system's structure, behaviour, and interactions. UML includes diagrams such as class diagrams, sequence diagrams, and deployment diagrams.These diagrams are used to represent the system's components, their relationships, and the flow of data and control. The diagrams are used to identify the system's objects, their attributes, and the operations that can be performed on them. The diagrams also help to identify the system's use case and the interactions between the system's use cases and the interactions between the system's components. It is also used to identify the system's architecture and the components that make up the system. The diagrams allow to identify the system's requirements and the design of the system, and the system's interfaces and the interactions between the system's components.

2. System Architecture

2.1 Subsystem decomposition



2.2 Hardware/software mapping



3. Object Model

3.1 Class Diagram

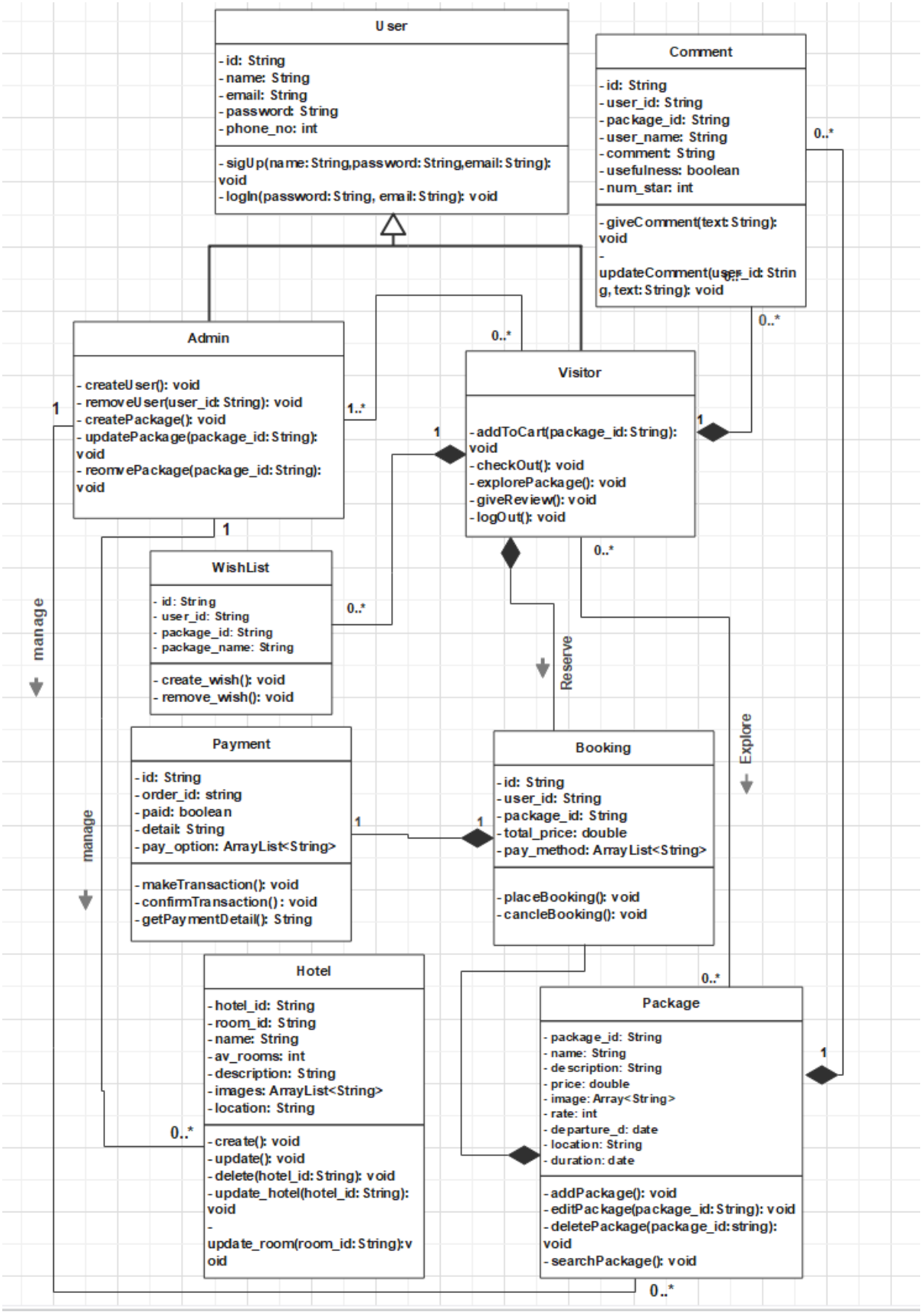


Figure. Class diagram

3.2 Sequence Diagram

Show how processes operate with one another and in what order. Depict the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

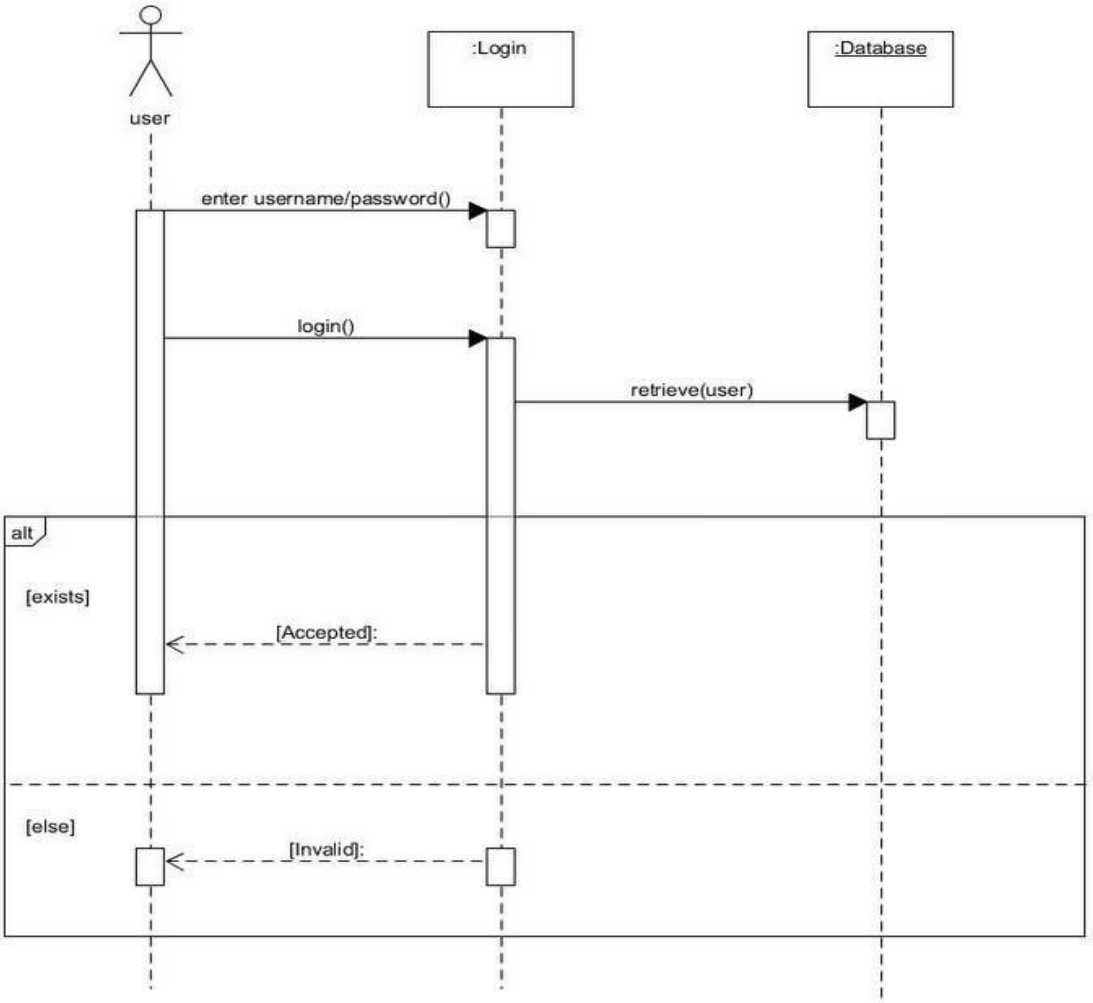


Figure:Sequence diagram for login

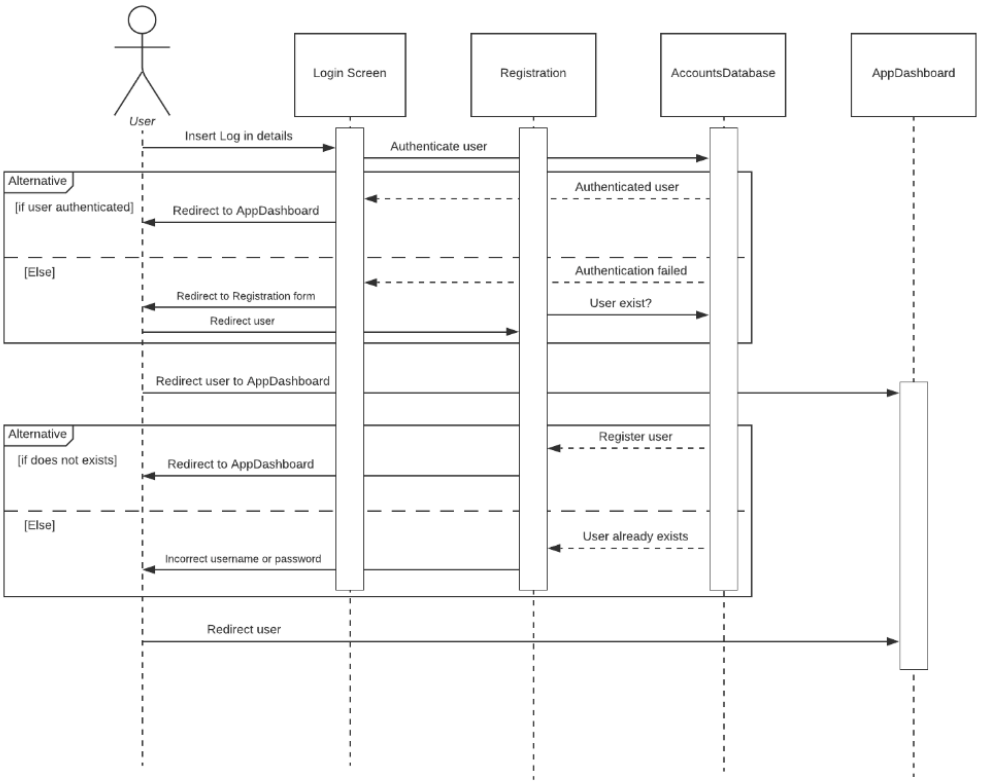


Figure:Sequence diagram for sign up

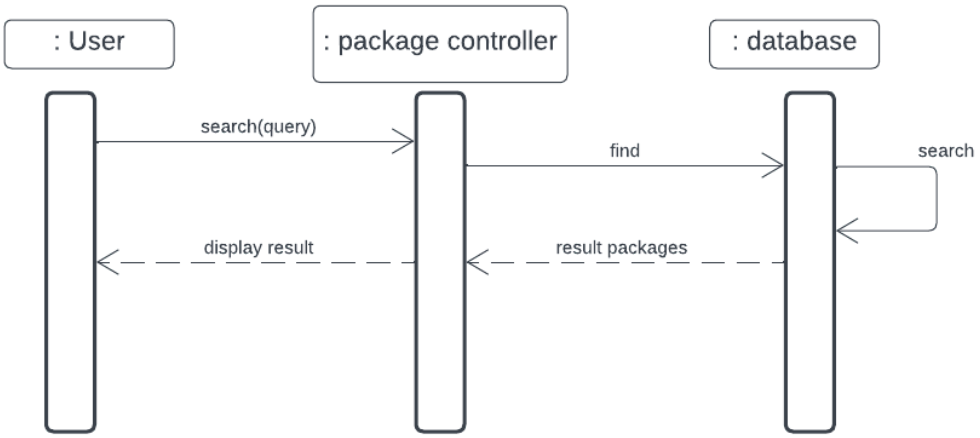


Figure: Sequence Diagram for `Search Package`

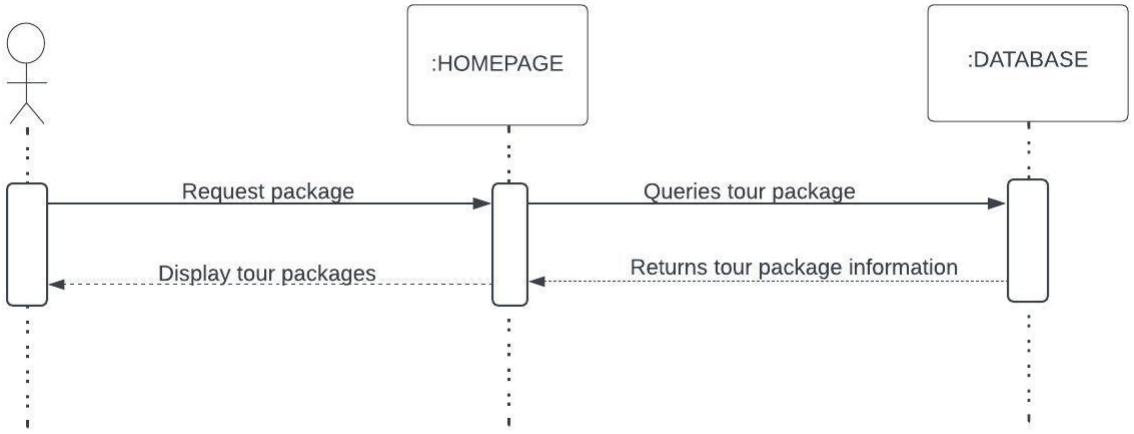


Figure: Sequence Diagram for `Explore Package`

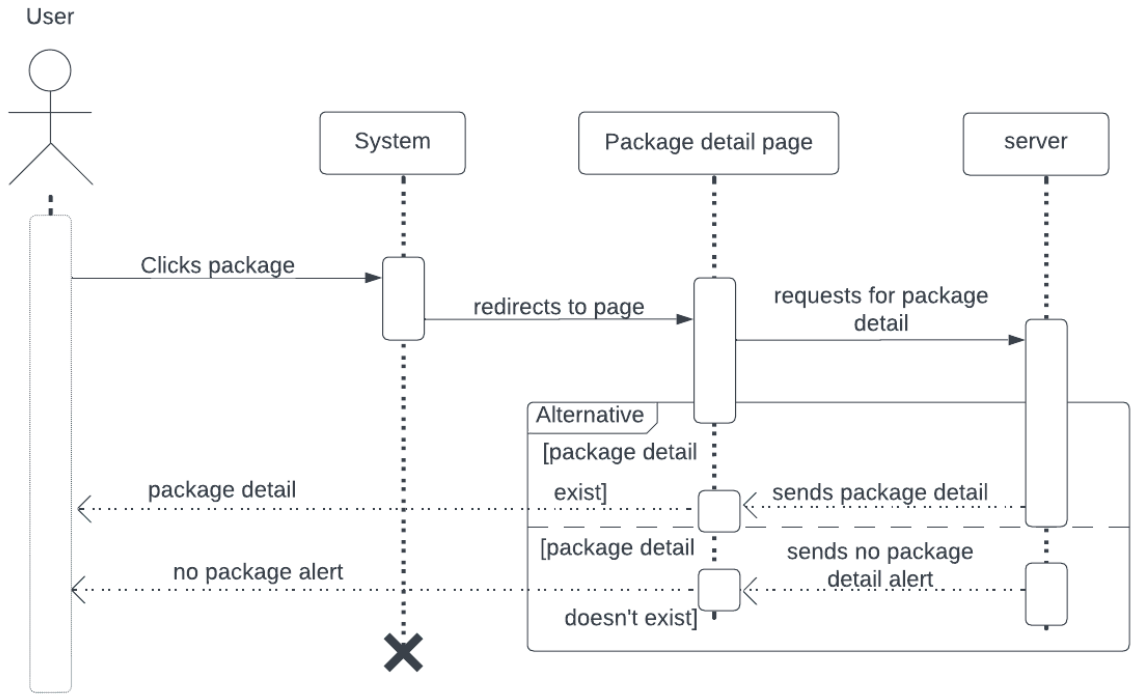


Figure: Sequence Diagram for `view Package`

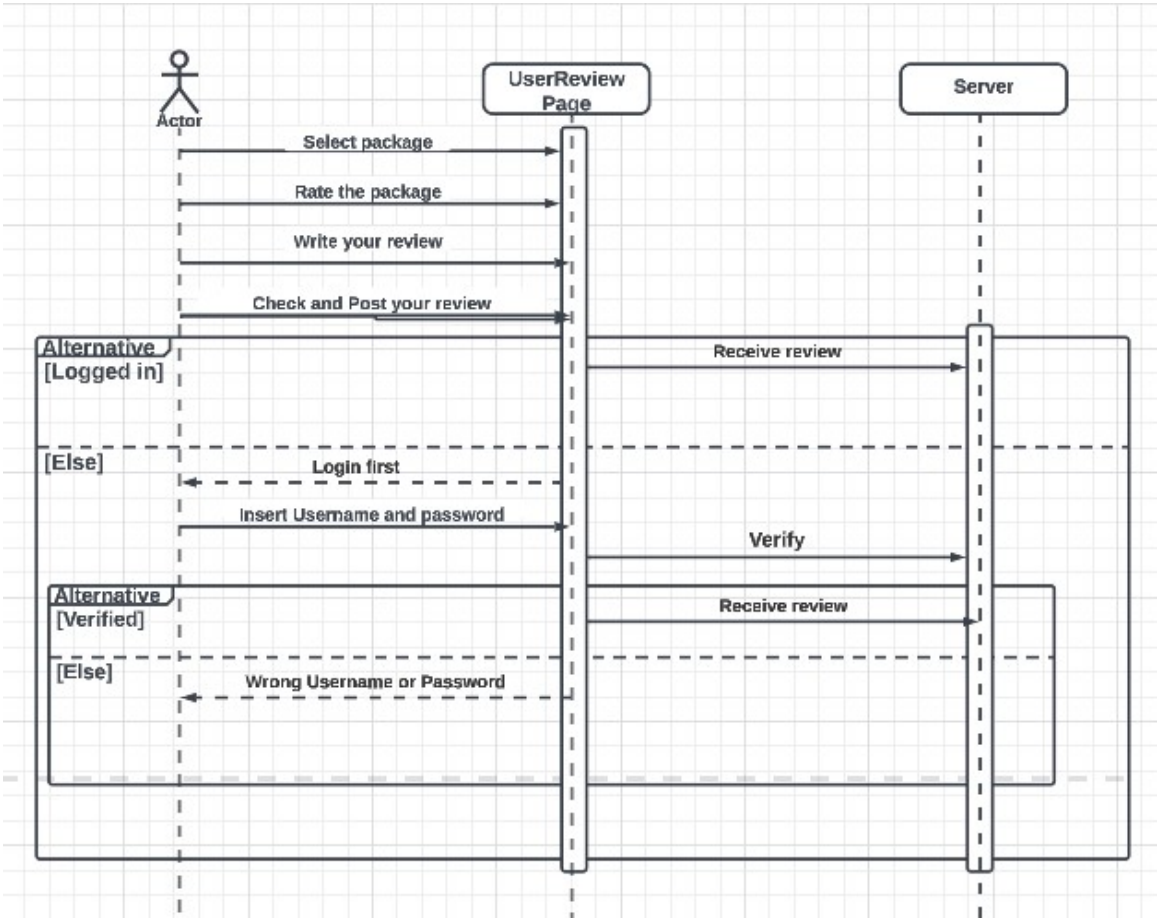


Figure: Sequence Diagram for `Write Review`

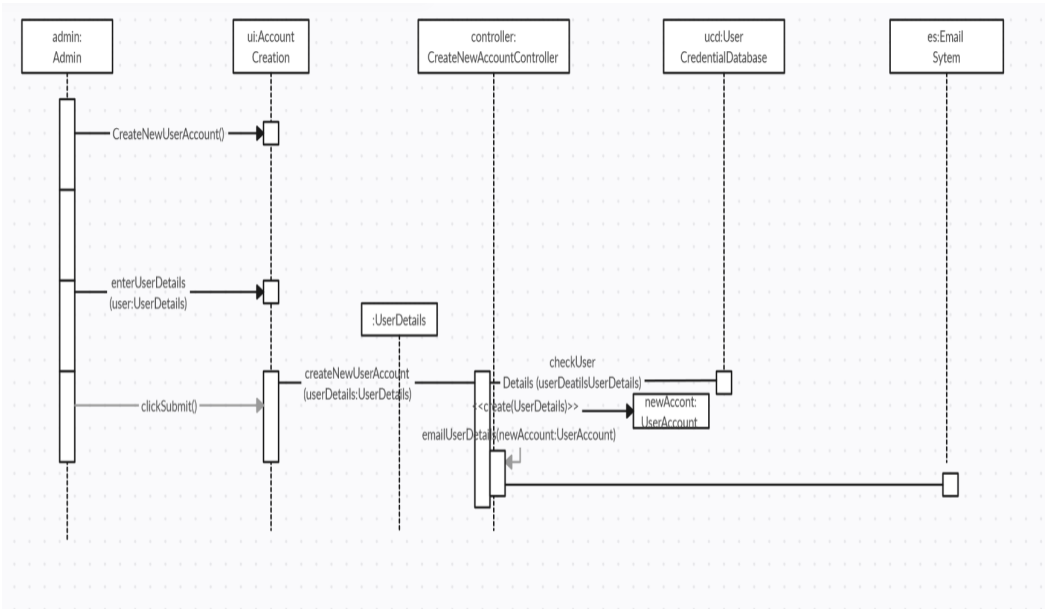


Figure: Sequence Diagram for `Create User`

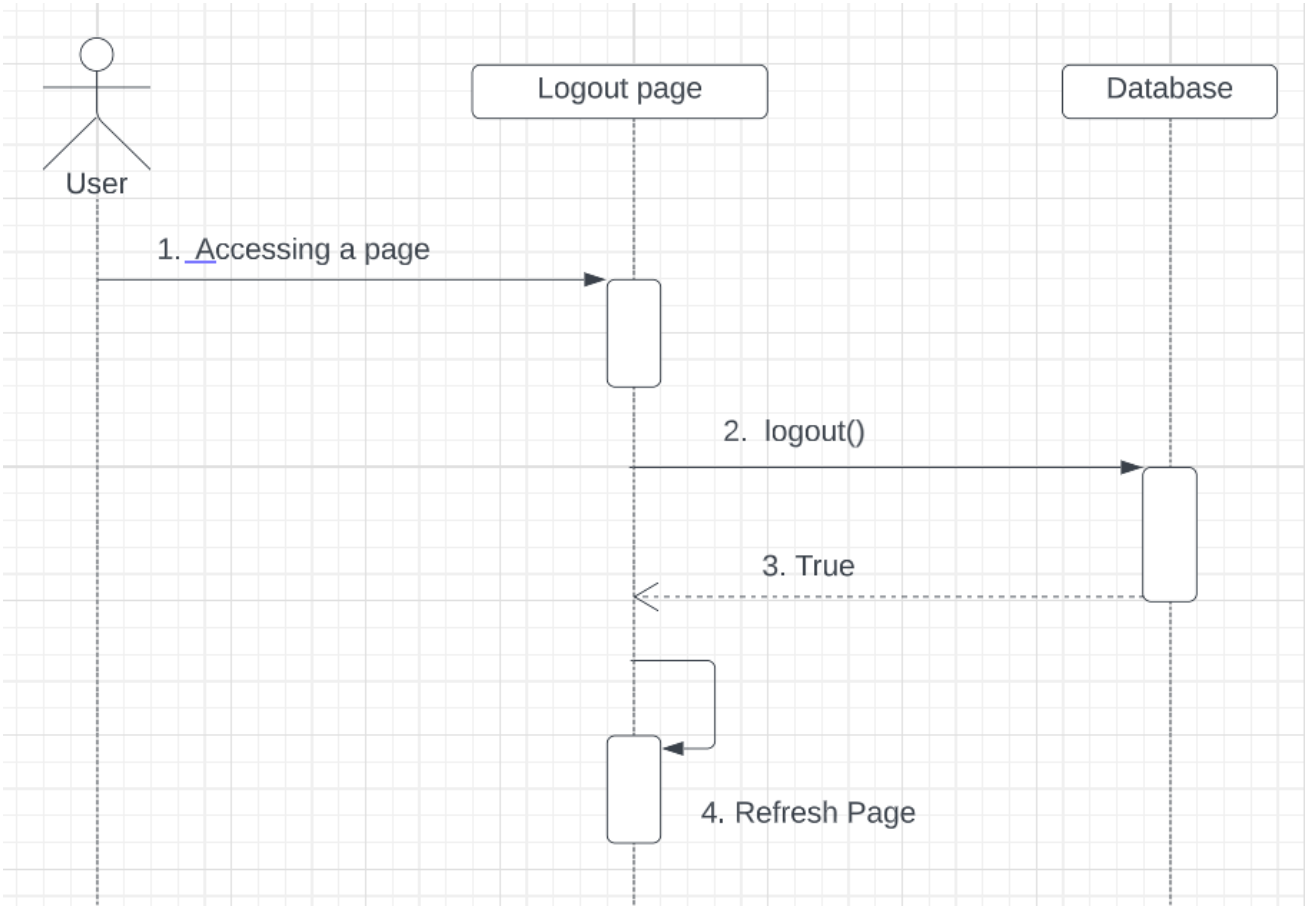


Figure: Sequence Diagram for `Logout`

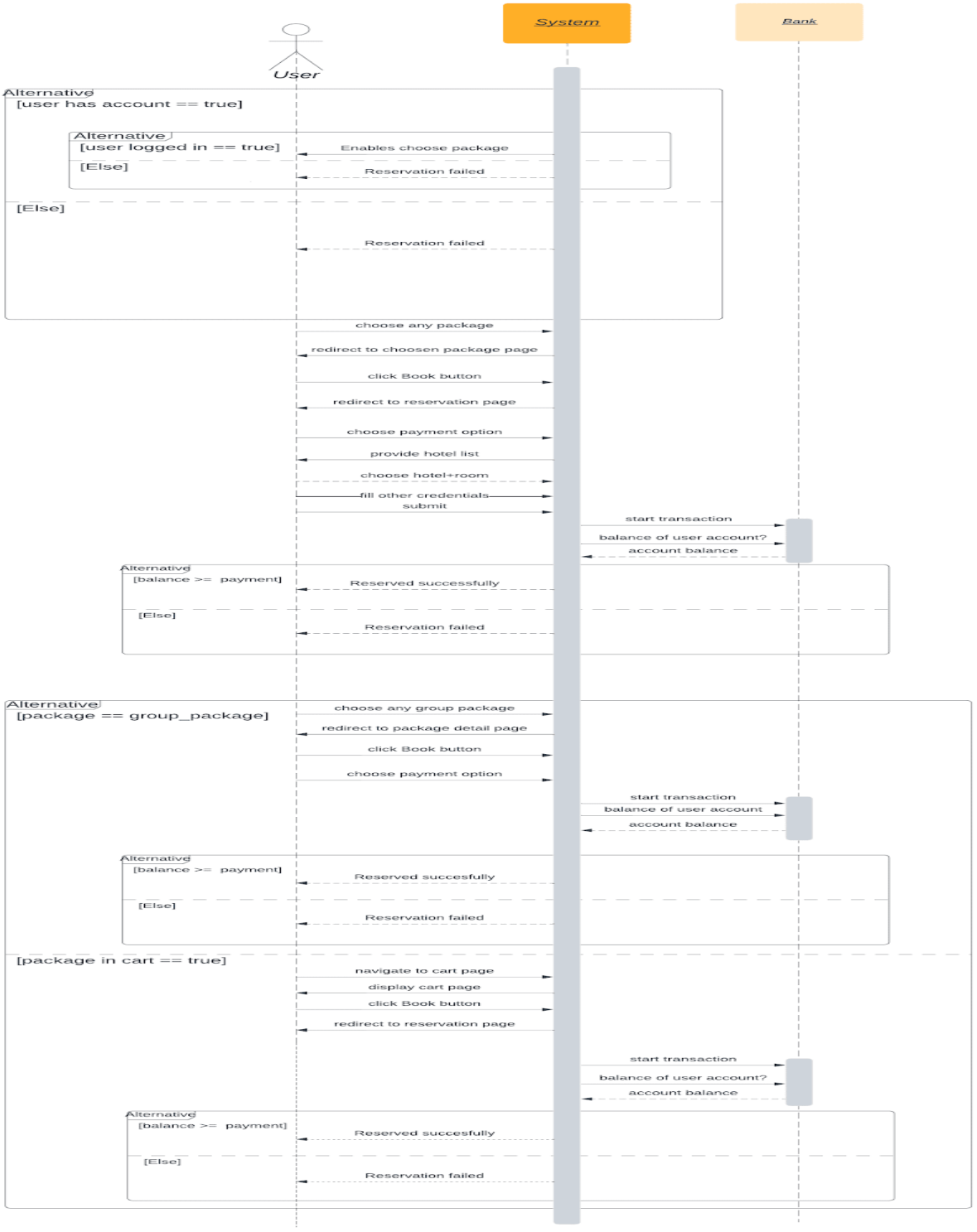


Figure: Sequence Diagram for `Reserve package`

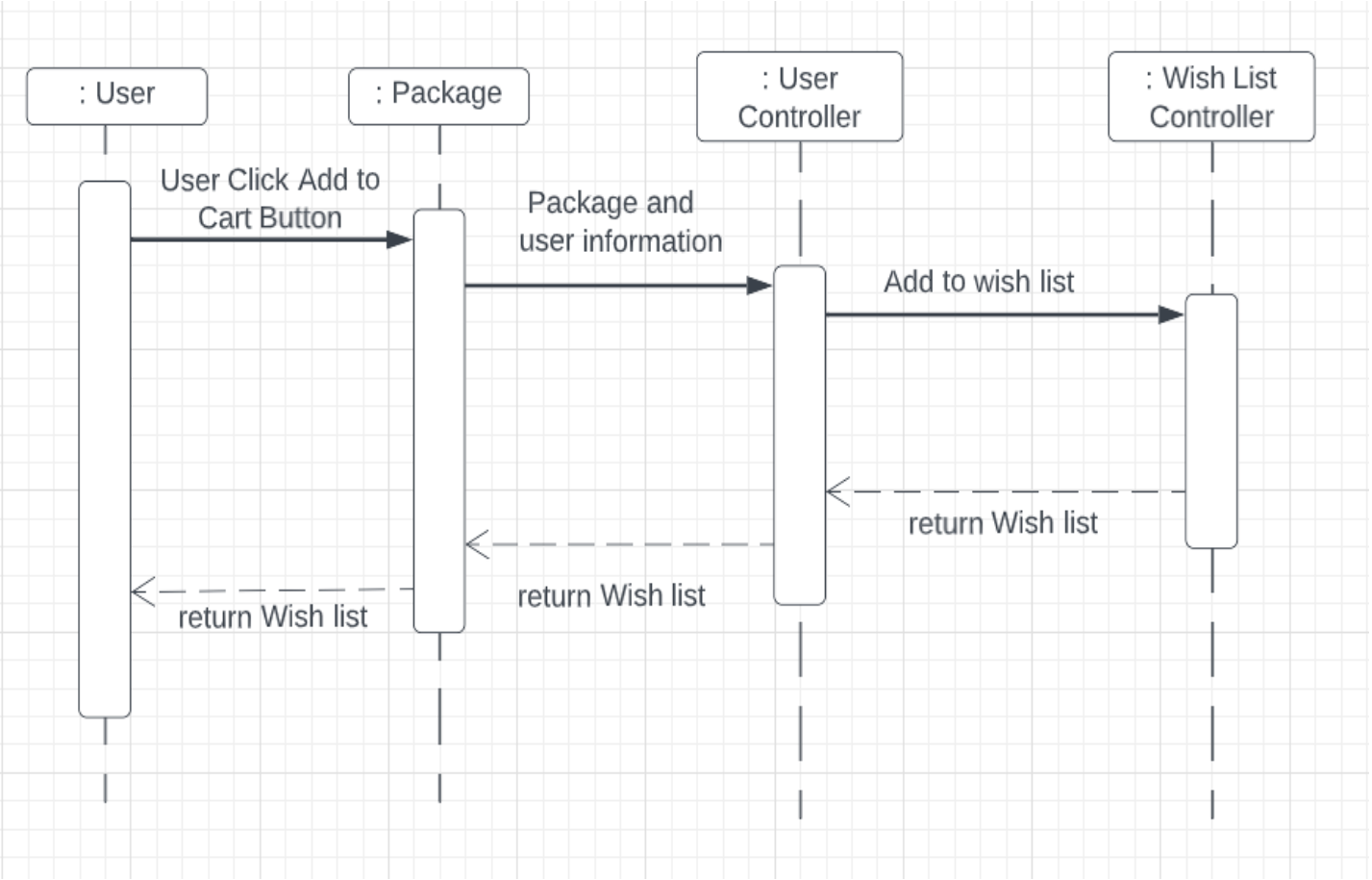


Figure: x Figure: Sequence Diagram for `Add to Cart`

4. Detailed Design

Here show the identified class in detail for example: -Assume a system has a Mobile Client Class then describe about the class in the following manner.

The classes represented here are the ones identified on your class diagram. But you must add the methods and classes identified in sequence and state chart diagrams.

Table: Admin class

Admin class
- Name:String - id:String - email:String - password: String - phone_no: int
- logIn(password: String, email: String):void - signUp(Name, Password, email):void - createUser():void - removeUser(user_id:String):void - createPackage():void - updatePackage(package_id:String):void - removePackage(package_id:String):void - logOut(); void

Table: x Attributes description for Admin class

Attribute	Type	Visibility	Invariant
Name	String	Private	Name <> NULL and must contain first, middle and last name and shouldn't contain special characters and integers.
id	String	Private	Id<> NULL and must be unique.
email	String	Private	Email <> NULL ✔ Must contain @ ✔ Must contain. (dot) ✔ Position of @ >1 ✔ Position of (dot)>position of @ + 2 ✔ Position of (dot)+3<= total length of email address and the total character of the Email is at least 5 characters
password	String	Private	Password <> NULL # Must contain letters # Must contain numbers

			# Must contain special characters # Must be 8 or above characters in total
phone_no	int	Private	PhoneNumber <> NULL must be 10 digits and must start by +251/09

Table: Operation description for Admin class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
logIn	Public	void	Email and Password	The Admin's personal information should exist	Admin should access admin page entirely
signUp	Public	void	Name, Password, email	Admi should have internet connection	Admin will access admin page
createUser	Public	void	Username, Email, Password	The Admin should log in.	Admin will be able to create new users
removeUser	Public	void	user_id	The Admin should log in.	Admin could delete existing users
createPackage	Public	void	-	The Admin should log in.	Admin will be able to create new packages
updatePackage	Public	void	package_id	The Admin should log in.	Admin will be able to send patch requests to the server
removePackage	Public	void	package_id	The Admin should log in.	Existing packages will be removed by the admin
logOut	Public	void	-	The Admin should log in.	Admin will be redirect to the login page

Table: Visitors class

Visitors
+ ID:String + Name:String + Username:String + PhoneNumber:Integer - Email:String - Password:String
logIn(Email, Password):void signUp(Username, Password, email):void addToCart(Package_id):void checkout():void logOut():void explorePackage():void giveReview():void

Table: Attributes description for Visitors class

Attribute	Type	Visibility	Invariant
ID	String	Public	ID <> NULL and must be numbers to identify the visitor clearly from others.
Name	String	Public	Name <> NULL and must contain first, middle and last name and shouldn't contain special characters and integers.
Username	String	Public	Username <> NULL and must contain a name or a string and shouldn't contain special characters.
PhoneNumber	Integer	Public	PhoneNumber <> NULL must be 10 digits and must start by +251/09
Email	String	Private	Email <> NULL ✔ Must contain @ ✔ Must contain. (dot) ✔ Position of @ >1 ✔ Position of (dot)>position of @ + 2 ✔ Position of (dot)+3<= total length of email address and the total character of the Email is at least 5 characters

Password	String	Private	Password <> NULL # Must contain letters # Must contain numbers # Must contain special characters # Must be 8 or above characters in total

Table: Operation description for Visitors class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
logIn	Public	void	Email/User name and Password	The clients personal information should exist	The clients should access booking and add to cart
signUp	Public	void	Username, Email, Password	The clients personal information shouldn't exist	The clients personal information should exist
addToCart	Public	void	Package_id	The clients should select a package	The package should be stored as a potential booking
checkout	Public	void	-	The client should log in.	The client should access packages
logOut	Public	void	-	The client should log in.	The client shouldn't able to access bookings and add to cart.
explorePackage	Public	void	-	The client should log in.	The client should access packages
giveReview	Public	void		The client should log in	The review should be recieved

Table: Hotel class

Hotel class
-Name:String - Location:String -description:String -Image:Arraylist<String> -Available room:Integer -Services:String
-create ():void -update ():void -delete hotel ():void -update room():void

Table: x Attributes description for Hotel class

Attribute	Type	Visibility	Invariant
Name	String	Private	Name <> NULL and must contain first, middle and last name and shouldn't contain special characters and integers.
Location	String	Private	Address <>NULL and it must be between 6 to 20 characters
Description	String	Private	Description <>NULL and it must be between 20 to 120 characters
Available room	Integer	Private	Available room <> NULL and it contain digit between 1 to 50
Service	String	Private	Service <> NULL and it must contain 4 to 20 list it consists

Table: 7 Operation description for Hotel class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition

Create hotel	private	void	.	No inserted value to the hotel	Add hotel or hotel exist
Update hotel	private	void	.	No change to the hotel or change does not exist	Change is exist in the hotel
Delete hotel	private	void	-	Hotel is available to the customer	Hotel is not available to the customer
Update room	private	void		The room is still available or the room is still booked	The booked room are available or the available room are booked

Table: Booking class

Booking class
- Id:String - userId:String + date:Date + packageId:String + totalPrice :Integer + pay_method: Array List<string>
- placeBooking ():void - cancelBooking ():void

Table: x Attributes description for Booking class

Attribute	Type	Visibility	Invariant
Id	String	Private	Id<> NULL and must be unique.

userId	String	Private	userId<>NULL and must refer to a valid user.
date	Date	Public	date<>NULL must be a valid date, must be in the future.
packageId	Integer	Public	packageId <> NULL and must correspond to a valid package.
totalPrice	Double	Public	totalPrice <> NULL and totalPrice >= 0
pay_method	Array List<String>	Public	pay_Method <> NULL , must be one of accepted payment methods.

Table: 7 Operation description for Booking class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
placeBooking	public	booking	Booking: booking	A package with Package Id exists in the database.	A `Booking` object is created and added to the database.
cancelBooking	public	void	bookingId	A booking with a BookingId exists in database	The booking is removed from storage.

Table: Payment class

Payment class

- Id:String - order_id:String - paid:boolean - detail:String - pay_option :Array List<string>
- makeTransaction ():void - confirmTransaction ():void - getPaymentDetail():String

Attributes description for Payment class

Attribute	Type	Visibility	Invariant
id	string	private	id <> NULL and must be unique and shouldn't contain special characters
order_id	string	private	Order_id <> NULL and must be unique and shouldn't contain special characters
paid	boolean	private	Paid <> NULL and it can be whether true or false
detail	string	private	Detail <> NULL and it must be between 10 to 50 characters
Pay_option	string	private	Pay_option <> NULL, must be one of the accepted payment options.

Table: 7 Operation description for Payment class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
makeTransaction()	private	void	-	User must provide payment method and personal information	Payment has been accepted and successfully processed
getPaymentDetail()	Private	void	-	Payment request must have been made	User can see the payment detail that have been stored securely
confirmTransaction()	private	String	-	Payment must have been processed successfully and payment detail must have been successfully saved	Confirmation message sent to the user using email or sms

Table: wishlist class

Wishlist
-Id:String *-UserId:String* *+packageId:String* *+packageName:String*
- create_wishlist ():void - remove_wishlist ():void

Attributes description for wishlist class

Attribute	Type	Visibility	Invariant
Id	String	Private	id <> NULL and must be unique and shouldn't contain special characters
userId	String	Private	Order_id <> NULL and must be unique and shouldn't contain special characters
packageId	String	Public	packageId <> NULL and must be unique and shouldn't contain special characters
packageName	String	Public	packageName <> NULL and shouldn't contain special characters and integers.

Table: 7 Operation description for wishlist class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
create_wishlist	public	void	-	A package with Package Id exists in the database.	A `wishlist` object is created and added to the database.
remove_wishlist	public	Void	-	A wishlist exists in database	The wishlist is removed from storage.

Table: x Comment class

Comment
-id:String -user_id: String -package_id: String -user_name: String -comment: String -usefulness: boolean

-num_star: integer
-giveComment (text: String): void -updateComment (user_id: String, text: String): void

Table: x Attributes description for Comment class

Attribute	Type	Visibility	Invariant
id	String	Private	id <> NULL Is required to be distinct for every remark, which means that no two comments may share the same ID..
user_id	String	Private	user_id <> NULL Must belong to an actual user in the system.
package_id	String	Private	package_id <>NULL Must correspond to an existing package in the system.
user_name	String	Private	user_name <> NULL Must be a non-empty string Must correspond to the user which the user_id is stated
comment	String	Private	comment <> NULL Must be a non-empty string
usefulness	boolean	Private	usefulness

			Must be boolean
num_star	Integer	Private	num_star Must be an integer

Table: 7 Operation description for Comment class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
giveComment	Private	void	text: String	The comment that corresponds to the specified package_id and user_id shouldn't exist	The comment that corresponds to the specified package_id and user_id should exist
updateComment	Private	void	user_id:String, text:String	The comment that corresponds to the specified package_id and user_id should'n exist without updated content.	The comment that corresponds to the specified package_id and user_id should exist with updated content.

References

Bibliography

(list of book used for referance)

Web resource

(list of web pages you used as reference//address + access date)

http://www.tutorialspoint.com/uml/index.htm at May 5,2020(use this link for uml diagrams)