PES University, Bengaluru

Department of Computer Science & Engineering

Session: Jan - May 2023

Subject: Object Oriented Analysis and Design with Java

Title: Travel Agency Management System

Team members:

Prashanth Sai Akurathi	***PES1UG20CS525***
Vishal M	***PES1UG20CS508***
Ekanath reddy	***PES1UG20CS548***

Project Description

The Travel Agency Management System is an efficient and effective solution for managing the various processes involved in a travel agency. The proposed system aims to simplify the management of customer information, travel packages, bookings, service and payments for a travel agency.

The system will be developed using the Model-View-Controller (MVC) architecture, which separates the application's data, logic, and presentation layers. The application will be developed using Java using Design principles, as the programming language and MySQL as the database management system.

The Travel Agency Management System will have the following features:

User Management: This module will allow the travel agency to manage Users information, including their personal details, booking history, and payment details.

Package Management: This module will allow the travel agency to manage the travel packages they offer, including the destinations, pricing, and availability.

Booking Management: This module will allow the travel agency to manage the booking process, including creating bookings, modifying bookings, and cancelling bookings.

Payment Management: This module will allow the travel agency to manage payment transactions, including payment processing and payment reconciliation.

The proposed system will be developed using the following technologies:

Java Programming Language: Java is a widely used programming language, known for its platform independence, scalability, and security features.

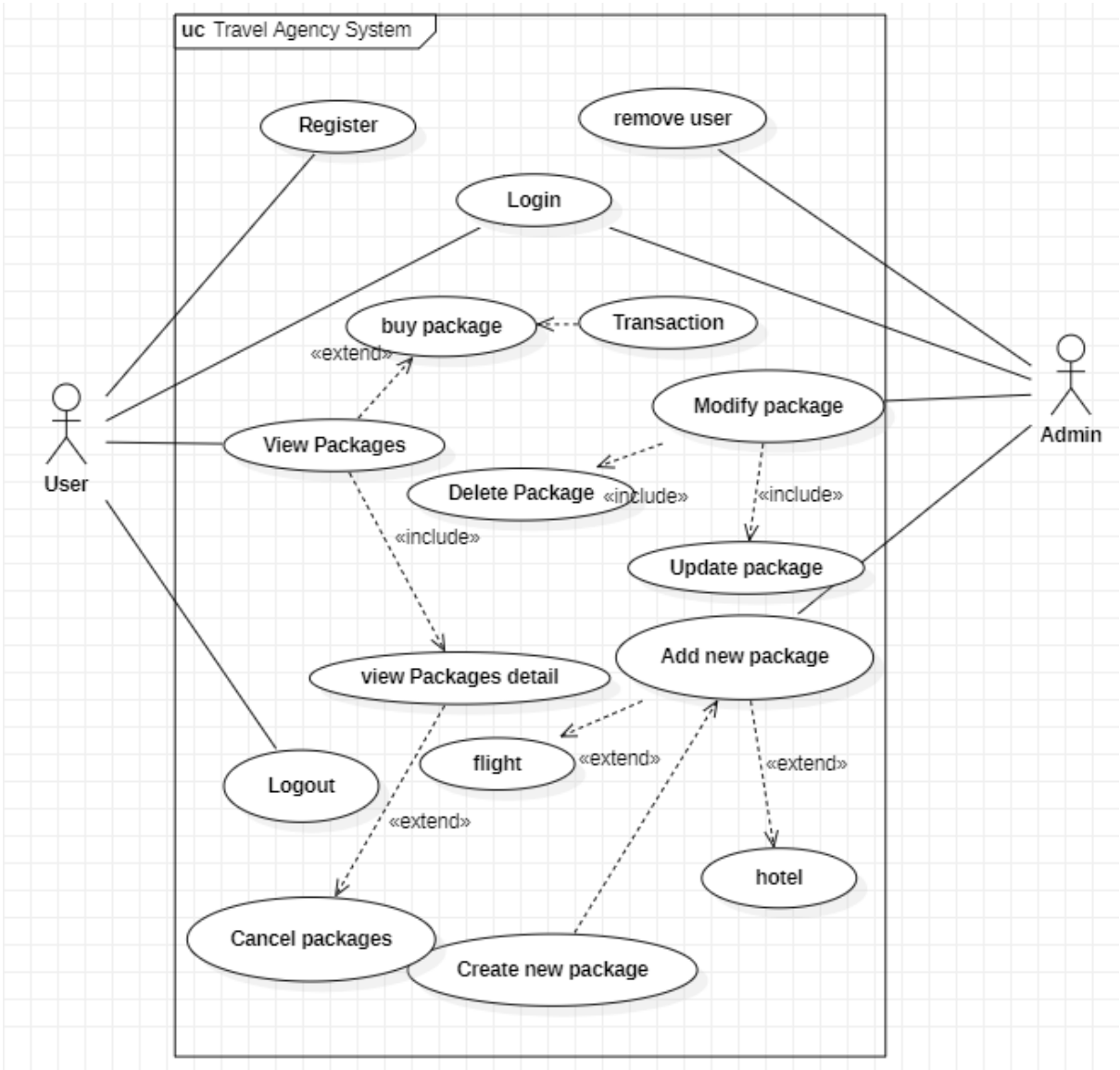
MySQL Database Management System: MySQL is a popular open-source database management system known for its reliability, speed, and flexibility.

Model-View-Controller (MVC) Architecture: MVC is a design pattern that separates the application's data, logic, and presentation layers, making the application easy to maintain and update.

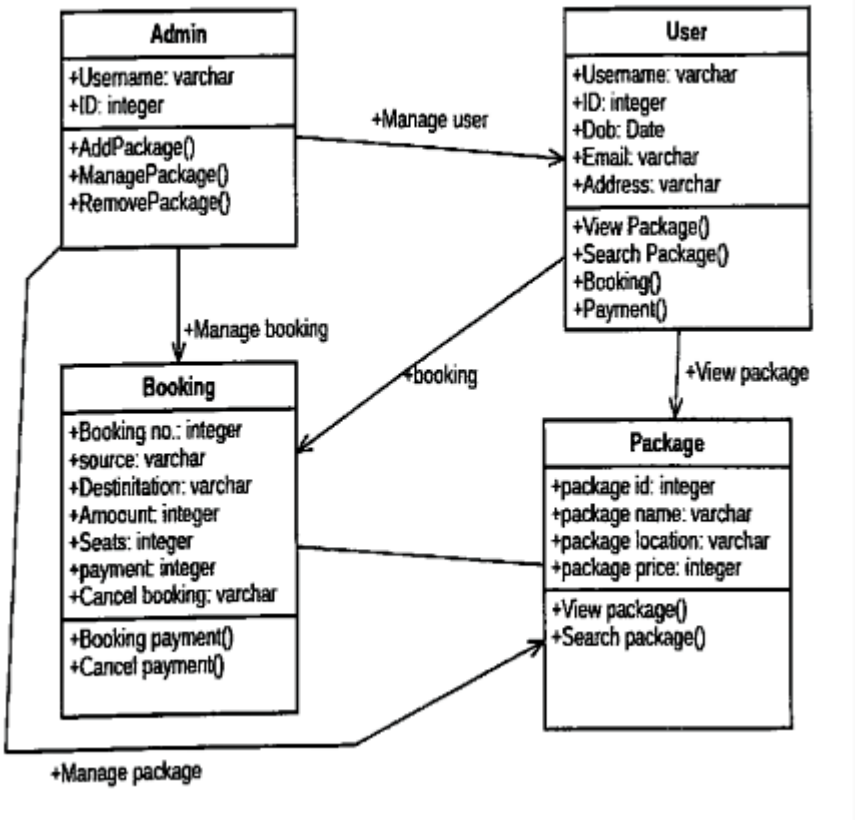
Spring Boot Framework: Spring Boot is a powerful framework that simplifies the development of Spring-based applications. It provides out-of-the-box features such as auto-configuration, embedded server, and dependency injection.

The Travel Agency Management System will provide the travel agency with an efficient and effective way to manage their business processes, improve customer service, and increase revenue.

I. Use Case Diagram:

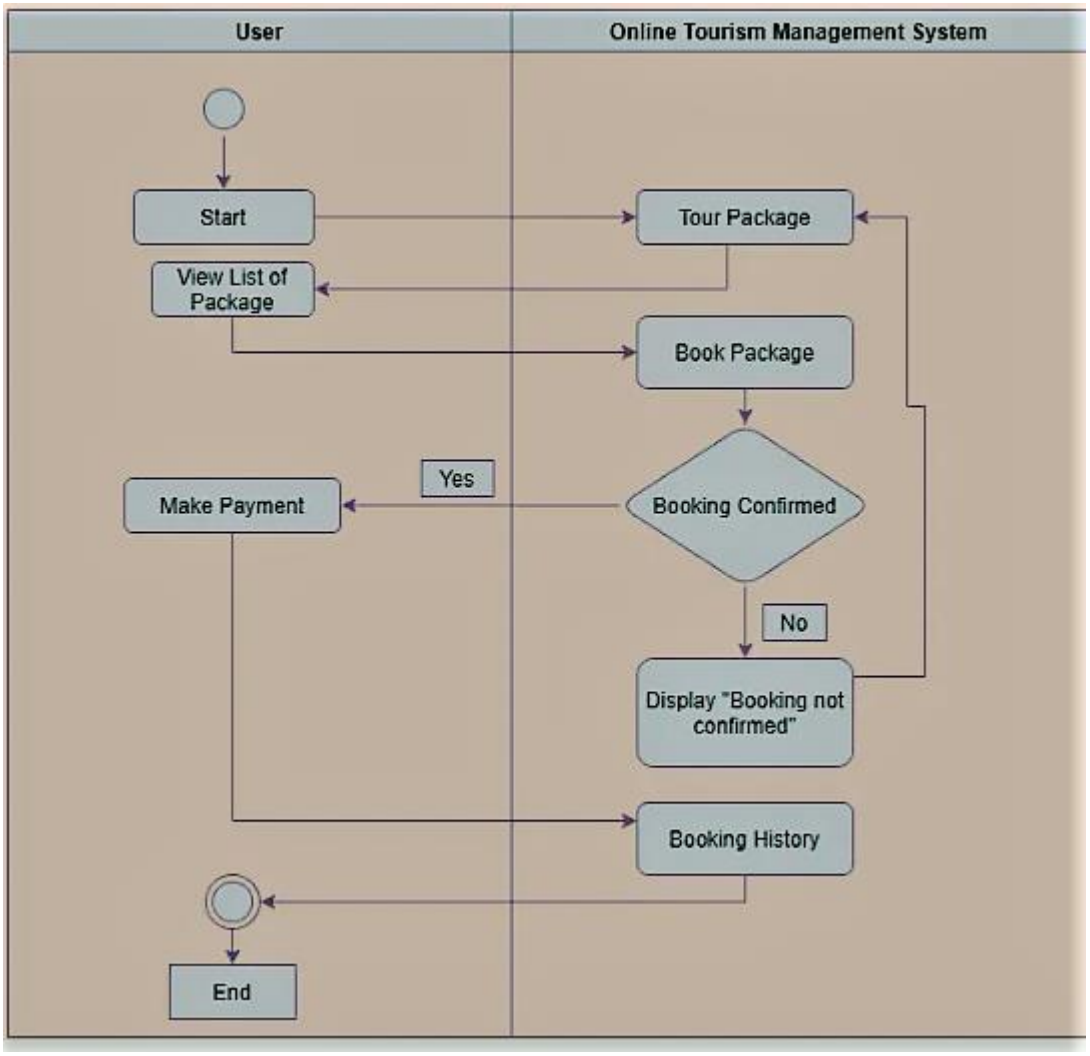


II. Class Diagram:

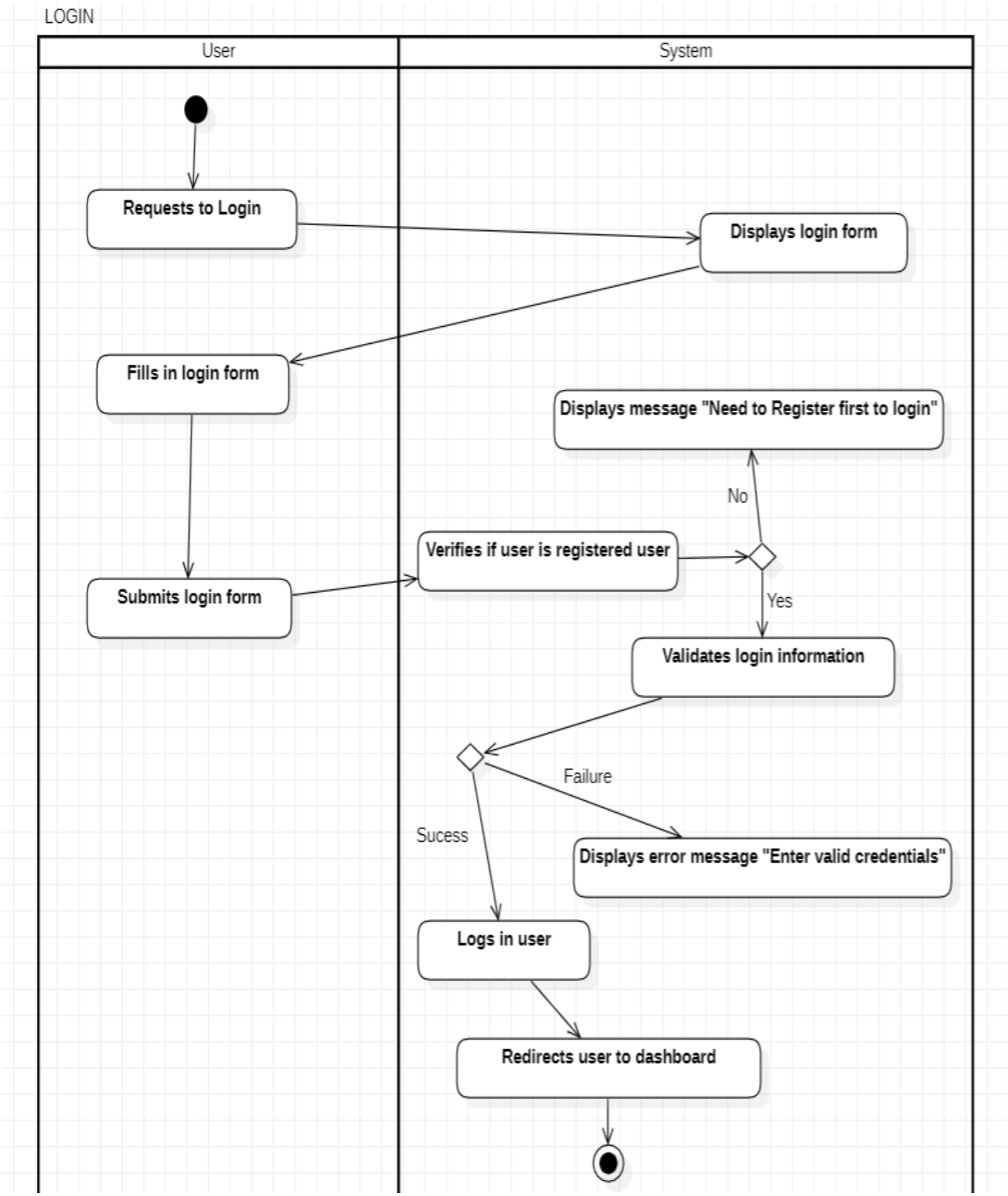


III. Activity Diagram:

Name: Register


User
Online Tourism Management System
Start
Tour Package
View List of Package
Book Package
Booking Confirmed
Yes
Make Payment
No
Display "Booking not confirmed"
Booking History
End

Name: Login


LOGIN
User
System
Requests to Login
Displays login form
Fills in login form
Displays message "Need to Register first to login"
No
Verifies if user is registered user
Submits login form
Yes
Validates login information
Failure
Sucess
Displays error message "Enter valid credentials"
Logs in user
Redirects user to dashboard

Architecture Pattern

Architecture Pattern used by Application is **Model View Controller (MVC)** which separates the application into three interconnected parts: the model (data), the view (user interface), and the controller (business logic).

Design Principles :

1) **Single Responsibility Principle (SRP)**

- ★ The User Class Has A Single Responsibility, Which Is To Represent The Data Of A User.
- ★ The Authority class has a single responsibility of defining the authority string for a user.
- ★ The JwtRequest class has only one responsibility, which is to represent a JWT authentication request.
- ★ The Quiz class has a single responsibility, which is to represent the quiz data model.
- ★ The Question class has a single responsibility, which is to represent a question in an exam.
- ★ The Category class has a single responsibility,

2) **Open/Closed Principle (OCP)**

- The Quiz class is open for extension, but closed for modification.
- The Category class.

3) **Liskov Substitution Principle (LSP)**

The Quiz class can be used wherever the base class Object is expected, without affecting the correctness of the program.

Design Pattern

Domain Model Pattern

The Role class has been implemented as a domain model, which encapsulates the behavior and attributes of a specific concept in the application domain. The class contains fields and methods to manage role-related data.

The Java Persistence API (JPA) design pattern: JPA is a Java-based ORM framework that provides a set of interfaces and annotations to map Java objects to a relational database. The @ManyToOne and @Id annotations in this code are part of the JPA API.(User-Role.java)

The Data Access Object (DAO) pattern:

DAO is a design pattern that separates the database access logic from the business logic of an application.

Object-Relational Mapping (ORM) -

The application is using the JPA (Java Persistence API) to map Java objects to database tables, allowing for seamless integration between the application's Java code and the database.

The class Quiz is annotated with the JPA annotations @Entity, @Id, @GeneratedValue, @Column, @ManyToOne, and @OneToMany. These annotations are used to define the mapping between the Java class Quiz and the relational database table.

Builder design pattern:

The Quiz class could potentially be used with a builder pattern to provide more readable and maintainable code when creating new instances of the Quiz class.

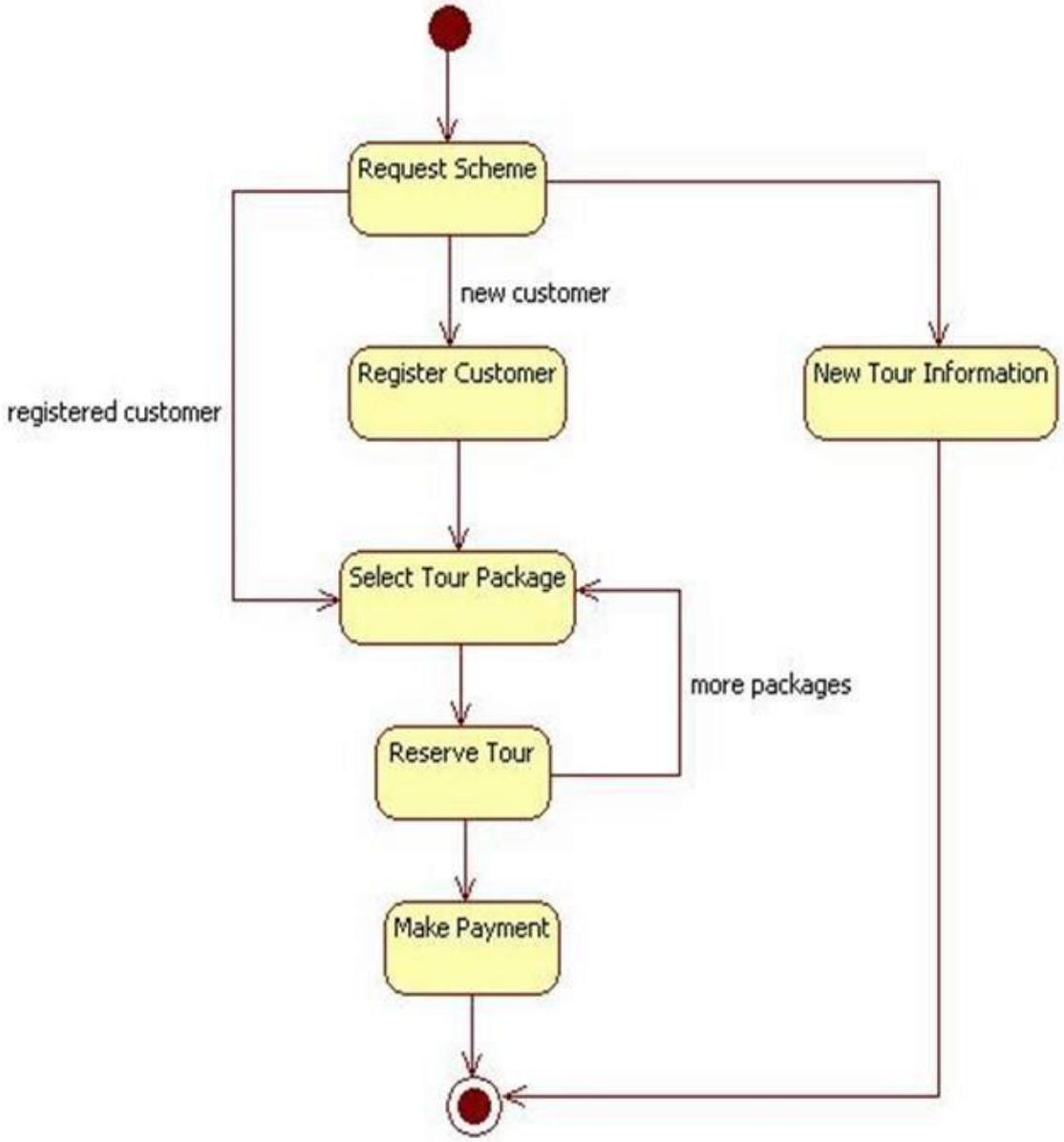
The Question class has a default constructor and getter/setter methods, which can be used to build objects.

Repository pattern:

The Question class does not directly interact with the database. Instead, it is likely used by a repository or service layer to persist and retrieve data.

The Category class acts as a Data Access Object by providing methods to access and manipulate the data stored in the category table.

IV. State Diagram:



Team Contributions:

PES1UG20CS525	Front end and Controller
PES1UG20CS508	Creation of Model
PES1UG20CS548	Transcation Class and add packages class