Vanier College

Computer Science Department

420-301-VA Programming Patterns

Fall 2024

Submitted by:

Joey Ayoub-Di Salvo & Danny Zhou

Project Scenario

Scenario: Creating a travel agency management system that allows users/customers to:

- Search for available travel packages (flights, hotels, tours)
- Book travel packages

- Manage bookings (registration, updating details)
- Manage customer accounts (registration, updating details)

Admin functionality:

- Add, update, and remove travel packages
- View all customer bookings and manage inventory

Desing Paradigm

The travel agency system will demonstrate the following key functionalities:

- **User Authentication**: Register and login functionality
- **Search & Book**: Users can search for travel packages based on destination, price, and availability
- **Booking Management**: View, update, or cancel bookings
- **Admin Features**: Add or manage travel packages, view user activity

The project will follow the **MVC (Model-View-Controller)** pattern:

- **Model**: Represents the data structure and interacts with the database (e.g., TravelPackage, User, Booking)
- **View**: Graphical User Interface (GUI) for users and admins (with internationalization)
- **Controller**: Manages user requests, performs CRUD operations using JDBC, and updates the view

Expected Output

The user will be able to:

- **Search travel packages** based on filters such as price, destination, and date
- **Book a travel package** and receive booking details
- **Manage their bookings** (view, modify, or cancel)
- **Admins** will be able to add, update, and remove travel packages and manage customer bookings

Design Requirements

- **Database Design**: We will use SQLite to store travel packages, user accounts, and bookings
- **Class Design**: Two hierarchies of classes:
- **User hierarchy**: User (abstract) -> Customer, Admin
- **TravelPackage hierarchy**: TravelPackage -> Flight, Hotel, Tour
- **Data Structures**: Lists or Maps to store travel packages and bookings in memory for efficient access

- **Design Patterns**: We will Implement the **Factory pattern** for creating travel packages (Flight, Hotel, Tour) and **Singleton pattern** for database connectivity

ER (Entity-Relationship) Diagram

The ER diagram will represent the relationships between the database entities such as User, TravelPackage, Booking, etc.

Entities:

- **User**: Attributes → userID, name, email, password, role (Customer/Admin)
- **TravelPackage**: Attributes → packageID, destination, price, availability, type (Flight/Hotel/Tour)
- **Booking**: Attributes → bookingID, userID, packageID, date, status

Relationships:

- A **User** can have multiple **Bookings** (one-to-many relationship)

- A **Booking** refers to one **TravelPackage** (many-to-one relationship)

ER Diagram Visualization:

- **User** (1) ←——— (N) **Booking** (N) ———→ (1) **TravelPackage**

Class Diagram

The class diagram will represent the structure of our Java classes and their relationships, showing inheritance and composition where relevant.

Key Classes:

- **User (abstract class)**:
  - Attributes: userID, name, email, password
  - Methods: login(), logout()
  - Inheritance: **Customer** and **Admin** classes extend **User**.
- **Customer** (inherits User):

    - Attributes: bookings (list of Booking)
    - Methods: searchPackage(), makeBooking(), cancelBooking()
- **Admin** (inherits User):
    - Methods: addTravelPackage(), updateTravelPackage(), removeTravelPackage(), printBookings(String UserID)
- **TravelPackage (abstract class)**:
    - Attributes: packageID, destination, price, availability
    - Inheritance: **Flight**, **Hotel**, and **Tour** extend **TravelPackage**
- **Flight**, **Hotel**, **Tour** (concrete classes):
    - Additional specific attributes like flightNumber (for Flight), hotelName (for Hotel), tourGuide (for Tour)
- **Booking**:
    - Attributes: bookingID, user, travelPackage, date, status
    - Methods: confirmBooking(), cancelBooking()