Sultan Qaboos University

College of Science

Department of Computer Science

COMP4701

Project Phase 1
G09 - Tijwal – Customizable Travel Packages Web Application

Team Members:

Salim Amur Al Habsi	135202
Yazan Khamis Al Qanobi	136908

Contents

Part A: Project Specifications

1. Project Information

Objectives:

1. Simplify Travel Planning: Provide a platform where users can easily browse, customize, and book travel packages based on their preferences.
2. Customizable Travel Experience: Allow users to modify pre-existing travel packages by adding or removing destinations, activities, and changing the number of days.
3. Secure and Seamless Transactions: Enable users to book and pay for travel packages securely, with trusted payment gateways.
4. Admin Control & Management: Provide an admin interface where travel agencies can manage, modify, and add new travel packages, update prices, and track bookings.
5. Community-Driven Reviews: Incorporate reviews and ratings from previous travelers to help users make informed decisions about their bookings.

Motivation:

The motivation for creating Tijwal stems from the challenges many travelers face when trying to plan trips. Travel planning can often be overwhelming due to the vast number of choices available—different destinations, activities, price ranges, and travel durations. Travelers want a platform where they can not only find ready-made travel packages but also have the flexibility to personalize their plans.

Furthermore, with the rise of online travel bookings, there is a growing demand for platforms that offer personalized experiences, security in transactions, and real-time feedback from fellow travelers. By addressing these challenges, Tijwal seeks to make travel planning more efficient, flexible, and enjoyable for users.

Real-World Relevance:

- Changing Travel Preferences: Post-pandemic, travelers are seeking more flexible, personalized, and hassle-free ways to plan trips. Platforms that offer customization are increasingly in demand as travelers seek unique experiences.
- Tourism Growth: As global travel rebounds, especially for destinations that have reopened, there is an opportunity to create a seamless, one-stop shop for travel planning and booking.
- Rise of Online Booking: With the growth of digital platforms in various industries, the travel industry is undergoing rapid transformation. Travelers are more likely to

book their trips online, and platforms offering flexibility and ease of use will attract a broad audience.

Benefits of the Project:

1. For Travelers: Tijwal gives travelers the ability to customize their itineraries according to their needs and preferences. Instead of being restricted to fixed packages, users can modify a plan, add or remove destinations, and change the trip's duration. This will make it easier to create a unique, tailored travel experience.
2. For Travel Agencies: Tijwal offers a comprehensive admin panel where agencies can easily manage their travel packages, monitor bookings, and adjust pricing. The platform offers them a streamlined method for handling bookings while reducing the complexities of manual tracking.
3. For the Travel Industry: By creating an app that bridges the gap between ready-made packages and user customization, Tijwal will contribute to a shift toward more personalized travel services, pushing the industry to adopt more dynamic and flexible offerings.

Problems the Project Aims to Solve:

1. Limited Customization: Many existing travel platforms provide fixed packages that do not cater to individual preferences. Travelers often have to stick to a predefined itinerary, which might not meet their exact needs. Tijwal solves this by allowing users to add or remove destinations and adjust their travel dates, activities, and duration.
2. Difficulty in Planning: Trip planning can be complex and time-consuming. By providing all relevant information in one place, such as detailed descriptions of activities, prices, and reviews, Tijwal helps users quickly make decisions, saving time and effort.
3. Lack of Trust: Travelers often hesitate to book packages without knowing whether the service is reliable. Reviews and ratings from past customers can give potential travelers the confidence to proceed with bookings.
4. Unsecure Payment Systems: Many travelers worry about the security of online payments. Tijwal aims to offer a secure and trustworthy payment system, ensuring users feel confident when booking their trips.
5. Admin Control: For travel agencies, managing and updating packages can be a cumbersome process. Tijwal offers an easy-to-use admin panel to streamline package management, pricing adjustments, and booking tracking.

Justification for the Proposed Solution:
The Tijwal platform addresses a gap in the travel industry by combining flexibility, ease of use, and trust. As travelers seek more personalized experiences, they are also more concerned about security and reliability when booking online. Tijwal offers an integrated solution that meets these needs, making travel planning more accessible and enjoyable.
Potential Users Who Will Benefit from Tijwal:

1. Individual Travelers: Anyone looking for a personalized vacation experience, whether it's a family trip, solo adventure, or group tour.
2. Tourists Seeking Flexibility: Travelers who want to tweak existing travel packages according to their preferences (e.g., extending a stay, removing destinations, or adding specific activities).
3. Adventure Seekers: Those who are interested in offbeat destinations or specific activities (e.g., hiking, cultural tours, food exploration) can tailor their itinerary to suit their needs.
4. Travel Agencies/Operators: Travel companies or tour operators can use Tijwal to list and manage their packages while offering users the flexibility to modify the tours.
5. Group Travelers: Families or groups of friends who need to collaborate on planning a trip, ensuring that everyone’s preferences are considered in a single, modified package.

In summary, Tijwal is designed to cater to the modern traveler who values both convenience and personalization. It solves real-world issues of limited customization, lack of flexibility, and the need for secure and trustworthy online travel bookings.

2. User Interviews and Findings

1. Tourist

- Interview Summary: The customer wants an easy way to find travel packages that match their budget and time. They prefer clear package details, including total price, places to visit, activities, and user reviews. They also want to customize the package by adding or removing places.
- Key Needs Identified:
    1. Simple and clear interface.

2. Online booking and secure payment.
3. Option to customize trip packages.
4. Real reviews from other users.

2. Travel Agency Staff

1. Interview Summary: The admin wants a simple dashboard to add or edit packages, change prices, and manage bookings. They also need to see customer details and booking history easily.
2. Key Needs Identified:
    1. Admin panel to manage packages and bookings.
    2. Option to update prices and availability quickly.
    3. View reports about user bookings and feedback.

How the Interviews Helped Define Requirements

The interviews helped shape the main features of the system. From the customer, we focused on usability, customization, and secure payment. From the admin, we learned the importance of easy management and real-time updates. These insights guided the design of both the user and admin interfaces and ensured the system meets real user needs.

3. Project Requirements

Functional Requirement

1 Package Management (User & Admin):

1. Users must be able to search, filter, and view a list of travel packages.
2. Admins must be able to log in to a separate dashboard to create, update (e.g., change prices), and delete packages.

2 Itinerary Customization:

- Users must have the ability to modify a selected package by adding or removing specific places or activities from the default itinerary. The price should update accordingly.

• Booking and Payment System:

- Users must be able to select a package (standard or customized) and complete a full booking process.
- The system must integrate a secure online payment gateway to process transactions.
- Admins must be able to view a list of all successful bookings.

• Information Display and Interaction:

- The system must display package details, including the itinerary, price, duration, included activities, a map visualization of the locations, and user-submitted reviews.

Non-functional requirement:

• Security**:**

- The system must be secure, especially the payment process ("pay safely"). This includes using SSL encryption (HTTPS) across the entire site and ensuring all user data (personal information, passwords) is securely stored.

• Usability**:**

- The application must be easy to use ("easy for tourists to plan"). The user interface (UI) should be intuitive, clean, and responsive, allowing users to find, customize, and book packages with minimal effort on both desktop and mobile devices.

• Reliability:

- The system must be highly reliable, especially the booking and payment modules. It should handle transaction errors gracefully (e.g., not charge a user for a failed booking).

4. Application Design

Tigwal web application follows a three-layer architecture consisting of a front-end, a back-end, and a database.

Front-End:

1. Developed using ASP.NET Core Razor Pages integrated with jQuery.
2. Offers administrators and users a responsive and dynamic interface.
3. Search bars, booking pages, and administrators' dashboards are included.
4. Manages user interactions and validated input.

Back-End:

1. Uses ASP.NET Core (C#) for handling requests, processing data, and managing the logic of the application.
2. Handles core system logic such as:
    1. Validating user input.
    2. Updating total price when users customize their travel package.
3. The back-end links between the front-end and the database.

Database:

1. Managed using Microsoft SQL Server.
2. Stores and retrieves data about users, packages, bookings, reviews, and transactions.
3. Use foreign key constraints for data consistency.

Main Pages:

1. Home Page: Display packages, and search filters.

2. Package Details Page: Show price, activities, and user reviews.

3. Customization Page: Lets the user add/remove destinations or activities.

4. Booking Page: Collects customer details and payments information.

5. Admin Dashboard: Accessible to admins for adding/removing packages, viewing booking, and changing prices.

Data Flow:

1. Users interact with a page.
2. Requests are sent to the back-end.
3. Back-end validates and processes the requests.
4. Results are retrieved from or stored in the database.
5. Responses are sent back to the front-end and rendered by the user's browser.

5. Database Design

The database of Tijwal is designed to keep track of users, travel packages, bookings, and reviews. It follows a relational database model.

Entities:

1. User: Customers or administrators who interact with the application.

2. Travel Package: Contains information about each package.

3. Booking: Stores user reservation details and payment information.

4. Review: Stores user reviews and feedback.

Relationships:

1. A user can have many bookings.
2. A user can have many reviews.
3. A booking belongs to a travel package.
4. A travel package can have many reviews.

Tables and Attributes:

User table:

Attribute	Date Type	Constrains	Description
UserID	INT	Primary Key	Unique user ID
FullName	VARCHAR(100)	Not Null	User's Full Name
Email	VARCHAR(100)	Unique, Not Null	User's email
PasswordHash	VARCHAR(255)	Not Null	Encrypted Password
Role	VARCHAR(20)	Customer or Admin	User Type

TravelPackages Table:

Attribute	Date Type	Constrains	Description
PackageID	INT	Primary Key	Unique package ID
Title	VARCHAR(150)	Not Null	Package Name
Description	TEXT	Not Null	Overview of the trip
DurationDays	INT	DurationDays > 0	Trip duration
BasePrice	DECIMAL(10,2)	Not Null	Base Price of the trip
Destinations	VARCHAR(255)	NotNull	List of included destinations
CreatedBy	INT	Foreign Key	Admin who created the package

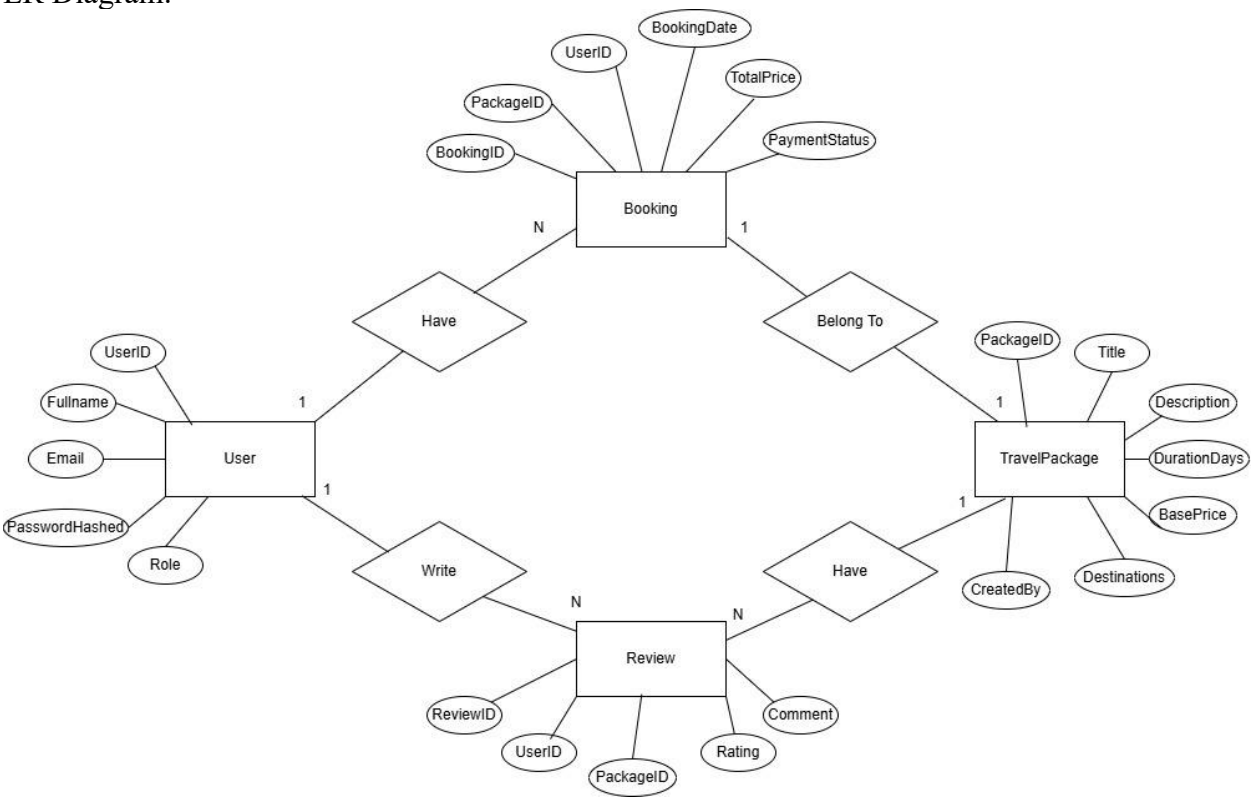
Booking Table:

Attribute	Date Type	Constrains	Description
BookingID	INT	Primary Key	Unique booking ID
UserID	INT	Foreign Key	Linked user
PackageID	INT	Foreign Key	Linked package
BookingDate	DATETIME	Default Current_Timestamp	Time of booking
TotalPrice	DECIMAL(10,2)	Not Null	Final price
PaymentStatus	VARCHAR(20)	Pending, Completed, or Failed	Payment state

Review Table:

Attribute	Date Type	Constrains	Description
ReviewID	INT	Primary Key	Unique review ID
UserID	INT	Foreign Key	Linked user
PackageID	INT	Foreign Key	Linked package
Rating	INT	$0 \leq \text{Rating} \leq 5$	User rating
Comment	TEXT	Null	User feedback

ER Diagram:



Part B: Implementation

6. Enumeration

The code above illustrates the definition of two enumerations: PaymentStatus and UserRole. They are used throughout the web application to enhance code clarity and consistency.

PaymentStatus represents the state of a booking's payment and has three possible values:

4. Pending (0): Payment has not been completed yet.
5. Completed (1): Payment was completed.
6. Failed (2): Payment attempt failed.

UserRole represents the access level for users:

7. Customer (0): Regular users with access to booking and travel package customization.
8. Admin (1): Administrative users with access to travel package management.

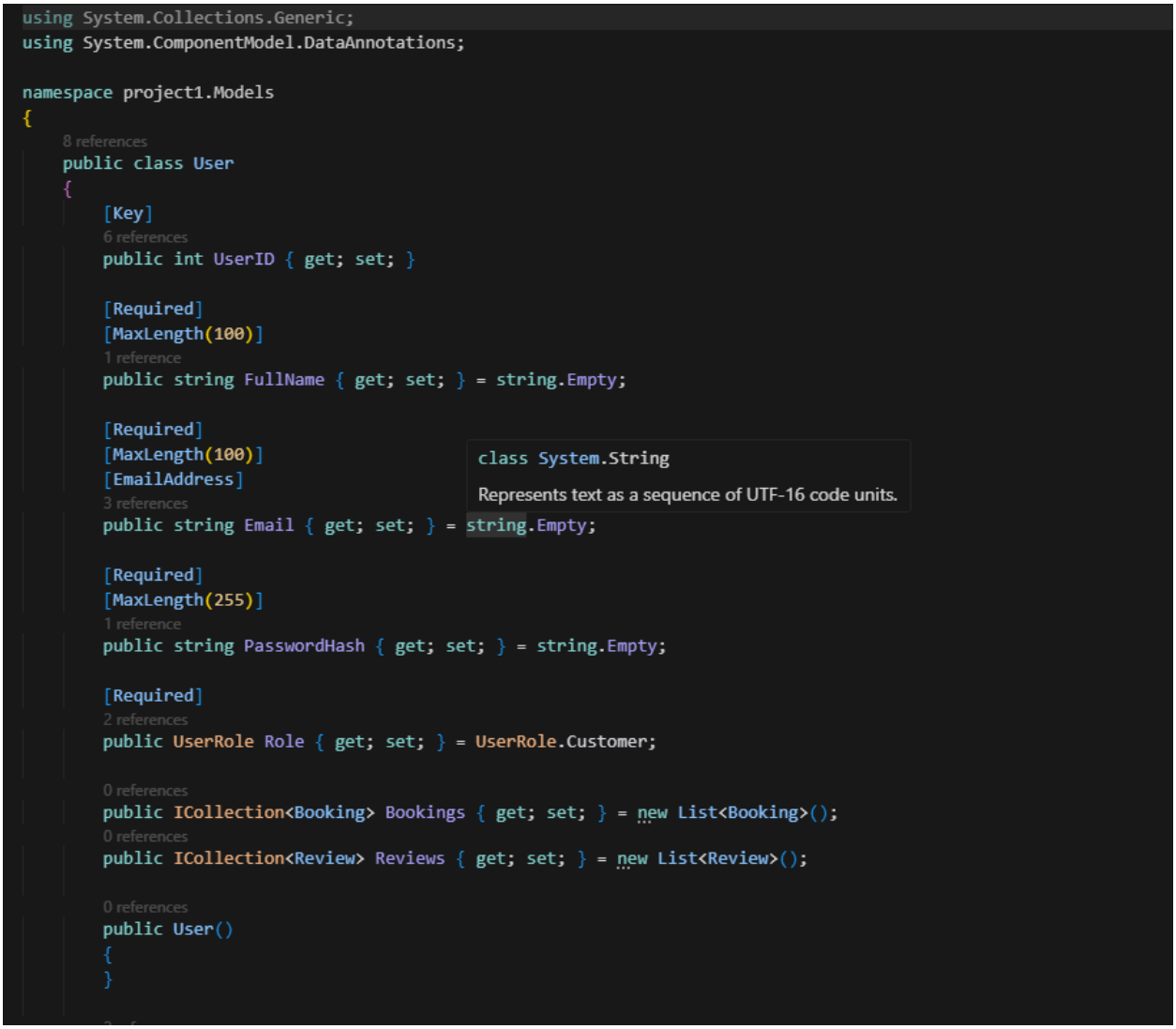
7. C# Classes

TravelPackage Class:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace project1.Models
{
    18 references
    public class TravelPackage
    {
        [Key]
        13 references
        public int PackageID { get; set; }

        [Required]
        [MaxLength(150)]
        13 references
        public string Title { get; set; } = string.Empty;

        [Required]
        5 references
        public string Description { get; set; } = string.Empty;

        [Range(1, int.MaxValue)]
        7 references
        public int DurationDays { get; set; }

        [Column(TypeName = "decimal(10,2)")]
        [Range(0, 99999999.99)]
        8 references
        public decimal BasePrice { get; set; }

        [Required]
        [MaxLength(255)]
        5 references
        public string Destinations { get; set; } = string.Empty; // comma-separated list for v1

        [MaxLength(500)]
        14 references
        public string? ImageUrl { get; set; }

        7 references
        public bool IsFeatured { get; set; }
```

The class above represents the travel package entity in the database. It defines the attributes of a package, which are its ID, title, duration, destinations, price, description, and an imageURL.

The annotations [Required], [MaxLength], and [Range] are used to validate the values of the attributes. The [Column(TypeName = "decimal(10,2)")] ensures that price values are stored in the database accurately.
User Class:



The class above defines the structure for storing user data in the application. It holds properties like UserID, FullName, Email, PasswordHash, and role which determines the user access level.

[Required], [MaxLength], and [Email] are attributes used for validation to ensure data consistency and accuracy. The Role property uses the UserRole enumeration to distinguish between customers and administrators.

Furthermore, relationships with the Booking and Review entities are maintained to link users to their bookings and reviews.

Booking Class:

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
 Ctrl+L to chat, Ctrl+K to generate
namespace project1.Models
{
    8 references
    public class Booking
    {
        [Key]
        1 reference
        public int BookingID { get; set; }

        [Required]
        1 reference
        public int UserID { get; set; }
        0 references
        public User? User { get; set; }

        [Required]
        1 reference
        public int PackageID { get; set; }
        0 references
        public TravelPackage? Package { get; set; }

        0 references
        public DateTime BookingDate { get; set; } = DateTime.UtcNow;

        [Column(TypeName = "decimal(10,2)")]
        [Range(0, 99999999.99)]
        1 reference
        public decimal TotalPrice { get; set; }

        [Required]
        3 references
        public PaymentStatus PaymentStatus { get; set; } = PaymentStatus.Pending;

        0 references
        public Booking()
        {
        }

        1 reference
        public Booking(int userId, int packageId, decimal totalPrice)
        {
            UserID = userId;
            PackageID = packageId;
            TotalPrice = totalPrice;
            PaymentStatus = PaymentStatus.Pending;
        }
```

Review next file ›

This class defines the structure that handles users' bookings in the Tijwal web application. It connects users to their travel packages and other details related to bookings such as BookingID, TotalPrice, and PaymentStatus.

Attributes like [Required], [Range], and [Column(TypeName = "decimal(10,2)")] are used to ensure that all data are valid and stored correctly. The class also uses the PaymentStatus enumeration to keep track of the payment process.

8. Front-End Framework

Dynamic Price Calculation:

```
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script>
    (function () {
        function recalc() {
            var base = parseFloat($("#BasePrice").val() || 0);
            var days = parseInt($("#DurationDays").val() || 1);
            var price = base;
            if (days > 1) {
                price = Math.round((base * (1 + (days - 1) * 0.05)) * 100) / 100;
            }
            $("#EstimatedPrice").val(price);
        }
        $("#DurationDays").on("input", recalc);
        $(document).ready(recalc);
    })();
</script>
```

Review next file ›

Customize.cshtml is razor page that allows users to personalize their travel package by changing destinations and trip duration. It uses jQuery to dynamically recalculate the estimated cost in real time as the customer customize the travel package.

The function listens for input on the DurationDays and updates the EstimatedPrice by applying a 5% increase for each additional day.

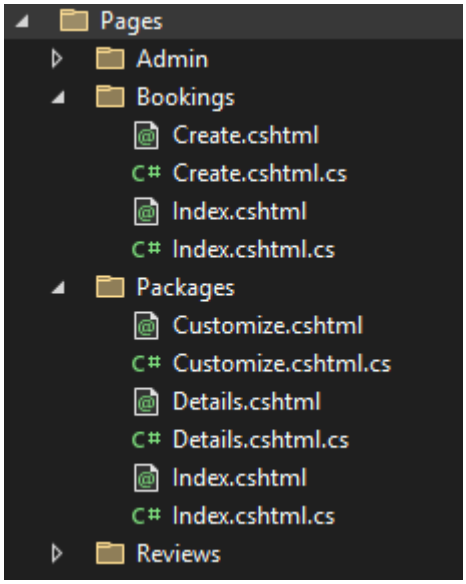
Auto Focus:

```
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script>
    $(function(){
        $("input[name='Search']").focus();
    });
</script>
```

This script automatically focuses on the searching input field when the page loads, which enhances the user experience.

9. Razor Pages

1. **Home page:** Landing page showing 6 featured travel packages in Bootstrap cards with images, prices, and quick action buttons.
2. **booking form:** Form to create bookings with inputs for Full Name, Email, Destinations, and Duration. Includes validation and error handling.
3. **review form:** Review submission form with email input, rating dropdown and a checkbox.
4. **Customization form:** Customize package with real-time price calculation (jQuery). Users can modify destinations and duration, price updates dynamically.
5. **Packages:** Shows all packages in a card grid with a search function.
6. **Admin: Admin dashboard for managing travel packages and viewing** booking information.



10. Business Logic

```
namespace project1.Services
{
    6 references
    public interface IPricingService
    {
        3 references
        decimal CalculateEstimatedPrice(TravelPackage package, int durationDays);
    }

    1 reference
    public class PricingService : IPricingService
    {
        3 references
        public decimal CalculateEstimatedPrice(TravelPackage package, int durationDays)
        {
            if (package == null) return 0m;
            return package.CalculateEstimatedPrice(durationDays);
        }
    }
}
```

Tijwal's business logic is handled by IPricingService interface and its implementation PricingService. This service is responsible for calculating the estimated price of the customized travel packages based on the selected trip duration. CalculateEstimatedPrice() method checks for valid package information passes the price to the related travel package.

12. Error Handling

```
63          public IActionResult OnPost()
64          {
65              try
66              {
67                  if (!ModelState.IsValid)
68                  {
69                      return Page();
70                  }
71
72                  var user = _store.Users.FirstOrDefault(u => u.Email == Email) ?? new User(FullName, Email);
73                  if (user.UserID == 0)
74                  {
75                      user.UserID = _store.Users.Count + 1;
76                      _store.Users.Add(user);
77                  }
78                  var booking = new Booking(user.UserID, PackageId, Price ?? 0)
79                  {
80                      BookingID = _store.Bookings.Count + 1
81                  };
82                  _store.Bookings.Add(booking);
83
84                  TempData["Success"] = "Booking created. Proceed to secure payment (coming soon).";
85                  return RedirectToPage("/Packages/Details", new { id = PackageId });
86              }
87              catch
88              {
89                  TempData["Error"] = "Could not create booking. Please try again.";
90                  return Page();
91              }
92          }
93      }
94  }
95
```

The code above demonstrates the error handling used in the booking process, input is validated by the OnPost() method and then it attempts to create a booking. It manages runtime error by utilizing try-catch, so if an error occurs, the system will catch it and display a message through TempData["Error"]. This ensures that the application remains stable and prevent crashes.

13. Bootstrap Styling

```
11  <style>
12      :root { --brand: #0d6efd; --brand-dark: #0b5ed7; }
13      .navbar { background: var(--brand); }
14      .navbar .nav-link, .navbar .navbar-brand { color: #fff !important; }
15      .btn-primary { background-color: var(--brand); border-color: var(--brand); }
16      .btn-primary:hover { background-color: var(--brand-dark); border-color: var(--brand-dark); }
17      .form-label { font-weight: 600; }
18  </style>
```

This web application uses bootstrap for clean and consistent styling throughout the pages.

14. Form Validation

This razor page has form validation to make sure that user input is correct and accurate before a booking is processed. asp-validation-form links ASP.NET core built-in validation attributes to this page.

Part C: Collaboration and Submission

1.Teamwork

Task	Done By
Project information	Salim
User Interviews	Salim
Project Requirements	Salim
Application Design	Yazan
Database Design	Yazan
Enumeration	Salim
C# Classes	Salim
Front-End Framework	Yazan
Razor Web Pages	Yazan
Business Logic	Salim
Custom Layout	Salim
Error Handling	Yazan
Bootstrap Styling	Yazan
Form Validation	Yazan
Report	Salim
Video Demo	Yazan
GitHub Repository	Yazan
Moodle Submission	Salim

2.Github Repository

https://github.com/Y3908/COMP4701Project