.NET Core Sprint

Tourism Management System

Table of Contents

INTRODUCTION

This document outlines a project for the .NET Line of Technology (LOT). The project is to develop **Tourism Management System**. This document contains the requirements, workflow of the system and gives guidelines on how to build the functionality gradually in each of the course modules of the .NET LOT.

SETUP CHECKLIST

Minimum System Requirements

- Intel Pentium 4 and above Windows 7
- Memory 8 GB
- Edge / Chrome browser
- SQL Server 2019 and SQL Server Management Studio
- Visual Studio 2022
- Visual Studio Code
- Git

INSTRUCTIONS

- The code modules in the Sprint should follow all the coding standards.
- High Level Design document should be created (HLD).
- You can refer to your course material.
- Git must be used for integration of Modules
- Since this project work will span for 4-5 days, you will need to take care of maintaining the code.

PROBLEM STATEMENT

OBJECTIVE

The objective of the **Tourism Management System** is to provide a comprehensive solution for managing tourism services. This includes user management, tour package handling, bookings, payments, and user feedback. The system aims to:

1. **Facilitate seamless user registration and authentication**: Allow users to create accounts, log in, and manage their profiles.
2. **Streamline tour package management**: Enable tour operators to create, edit, and manage tour packages.
3. **Simplify booking processes**: Allow users to search for tours, book them, and manage theirbooking history, including cancellations.
4. **Integrate secure payment processing**: Provide secure payment gateways (e.g., PayPal, Stripe) to process payments for bookings.
5. **Enhance user experience with reviews and ratings**: Allow users to leave reviews and rate tours to help future customers make informed decisions.
6. **Provide an admin interface for efficient system management**: Equip administrators with tools to manage users, bookings, reviews, and tour packages.

Abstract of the project:

The **Tourism Management System** is an online platform designed to streamline the process of managing and booking tour packages. The system allows users to search for, book, and review tour packages while enabling administrators to manage tours, users, bookings, and reviews. The platform features secureuser authentication, real-time payment processing, and an intuitive user interface for seamless interactions. The system is intended to improve the experience for both travelers and tour organizers by providing efficient management tools and a user-friendly booking system.

MODULE LIST and MODULE DETAILS

Modules:

Requirements:

1. **i. User Management:** Registration, login, and profile management.
2. **Tour Package Management:** Add, edit, delete, and list tour packages.
3. **Booking Management:** Book tours, view booking history, and manage cancellations.
4. **Payment Integration:** Process payments using a payment gateway.

5. **Search and Filter:** Search tour packages and apply filters.
6. **User Reviews and Ratings:** Allow users to review and rate tour packages.
7. **Admin Panel:** Manage users, tour packages, bookings, and reviews.

High-Level Design (HLD):

- **Frontend:** ASP.NET Core MVC for creating a responsive user interface.
- **Backend:** ASP.NET Core Web API for handling business logic and data access.
- **Database:** SQL Server (accessed through SSMS) for storing user, tour package, booking, and revie w data.
- **Authentication:** JWT (JSON Web Tokens) for secure user authentication.
- **Payment Gateway:** Integration with a third-party payment gateway like PayPal or Stripe.

Use Cases:

1. **User Registration:** A user can register by providing personal details and email verification.
2. **Tour Package Search:** A user can search for tour packages using keywords and apply filters.
3. **Book Tour:** A user can book a tour package.
4. **Checkout:** A user can proceed to checkout, enter payment details, and make a payment.
5. **Booking History:** A user can view their booking history and manage cancellations.
6. **Tour Review:** A user can write a review and rate a tour package after completion.

Endpoints:

User Management:

- **POST /api/users/register - Register a new user.**
- **POST /api/users/login - User login.**
- **GET /api/users/profile - Get user profile details.**
- **PUT /api/users/profile - Update user profile details.**
- **POST /api/users/logout - User logout.**
- **GET /api/users - List all users.**
- **GET /api/users/{id} - Get details of a specific user.**

- **DELETE /api/users/{id} - Delete a user.**

Tour Package Management:

- **POST /api/tours - Add a new tour package.**
- **PUT /api/tours/{id} - Update a tour package.**
- **DELETE /api/tours/{id} - Delete a tour package.**
- **GET /api/tours - List all tour packages.**
- **GET /api/tours/{id} - Get details of a specific tour package.**
- **GET /api/tours/category/{category} - List tour packages by category.**
- **GET /api/tours/location/{location} - List tour packages by location.**
- **GET /api/tours/search -**
  Search tour packages based on various filters (name, price, category, location).

Booking Management:

- **POST /api/bookings - Book a tour package.**
- **PUT /api/bookings/{id}/update - Update a booking.**
- **DELETE /api/bookings/{id} - Cancel a booking.**
- **GET /api/bookings - View all bookings.**
- **GET /api/bookings/{id} - View details of a specific booking.**
- **GET /api/bookings/user/{userId} - View bookings made by a specific user.**
- **GET /api/bookings/tour/{tourId} - View bookings for a specific tour package.**
- **GET /api/bookings/date/{date} - View bookings by date.**

Payment Integration:

- **POST /api/payments - Process payment.**
- **GET /api/payments/{id} - Get payment details.**
- **GET /api/payments/booking/{bookingId} - Get payment details for a specific booking.**
- **GET /api/payments/user/{userId} - View all payments made by a specific user.**

Reviews and Ratings:

- **POST /api/reviews - Add a new review for a tour package.**

- **PUT /api/reviews/{id} - Update a review.**
- **DELETE /api/reviews/{id} - Delete a review.**
- **GET /api/reviews - List all reviews.**
- **GET /api/reviews/{id} - View details of a specific review.**
- **GET /api/reviews/tour/{tourId} - List reviews for a specific tour package.**
- **GET /api/reviews/user/{userId} - List reviews made by a specific user.**

Wishlist Management:

- **POST /api/wishlist/add - Add tour package to wishlist.**
- **DELETE /api/wishlist/remove/{tourId} - Remove tour package from wishlist.**
- **GET /api/wishlist - View tour packages in the wishlist.**

Itinerary Management:

- **POST /api/itineraries - Add an itinerary for a tour package.**
- **PUT /api/itineraries/{id} - Update an itinerary.**
- **DELETE /api/itineraries/{id} - Delete an itinerary.**
- **GET /api/itineraries - List all itineraries.**
- **GET /api/itineraries/{id} - View details of a specific itinerary.**
- **GET /api/itineraries/tour/{tourId} - List itineraries for a specific tour package.**

Notification Management:

- **POST /api/notifications - Send notification to users (e.g., booking confirmation, tour updates).**
- **GET /api/notifications - List all notifications.**
- **GET /api/notifications/{id} - View details of a specific notification.**

Admin Management:

- **POST /api/admin - Add a new admin.**
- **PUT /api/admin/{id} - Update admin details.**
- **DELETE /api/admin/{id} - Delete an admin.**
- **GET /api/admin - List all admins.**

Search and Filter:

- **GET /api/search/tours - Search for tour packages based on various criteria.**
- **GET /api/search/bookings - Search for bookings based on date, user, tour package, etc.**
- **GET /api/search/users - Search for users based on name, email, role, etc.**

Properties and Data Types:
User Management:

- **UserID:** int
- **Username:** string
- **Password:** string
- **Email:** string
- **FirstName:** string
- **LastName:** string
- **PhoneNumber:** string
- **Address:** string
- **Role:** string

Tour Package:

- **TourID:** int
- **TourName:** string
- **Description:** string
- **Price:** decimal
- **Category:** string
- **Duration:** int (in days)
- **ImageURL:** string
- **Rating:** double
- **Reviews:** List<Review>

Booking:

- **BookingID:** int
- **UserID:** int
- **TourID:** int
- **BookingDate:** DateTime
- **Status:** string (e.g., confirmed, cancelled)

Admin:

- **AdminID:** int
- **Username:** string
- **Password:** string
- **Email:** string
- **Permissions:** List<string>

Review:

- **ReviewID:** int
- **UserID:** int
- **TourID:** int
- **Rating:** int
- **Comment:** string
- **ReviewDate:** DateTime

Implementation (Expectation):

Frontend:

- Develop the user interface using **ASP.NET** Core MVC.
- Create views for user registration, login, profile management, tour listing, booking, and order pag es.

Backend:

- Implement the Web API using **ASP.NET** Core.
- Create controllers for user, tour package, booking, and payment management.

Database:

- Design and create the database schema in SQL Server using SSMS.

- Implement Entity Framework Core for data access.

Authentication:

- Implement JWT-based authentication for secure user access.

Payment Gateway:

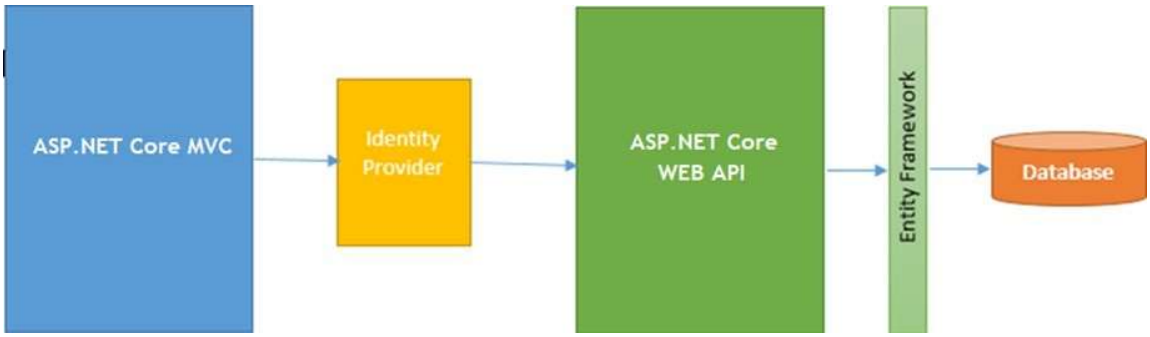
- Integrate with a payment gateway for processing payments.

Testing:

- Conduct thorough testing to ensure all functionalities work as expected.

PROJECT STRUCTURE

The project should have minimum target framework – .NET 6 for Asp.net Core MVC for UI Design and Asp.net Core Web API



Testing

- Unit and integration tests for Web API endpoints.
- End-to-end testing for frontend interactions.
- Load testing to ensure performance under high user traffic.

EXPECTATION

Create Asp.net Core MVC and Web API using

- Visual Studio 2022 IDE

- NET 6.0 / C# 10.0
- SQL Server 2019
- Entity Framework Core

REQUIREMENT

In this sprint, you must use:

- Asp.net Core MVC
- Use Entity Framework Core
- Asp.net Core Web API

Design guidelines

- All the exceptions/errors to be captured and user-friendly message to be displayed on the Common Error page.
- Asp.net Core MVC must be created using Code First Approach.

IMPLEMENTATION

SUMMARY OF THE FUNCTIONALITY TO BE BUILT:

The participants need to develop the Bike Store App by building the functionality incrementally in each of the course modules of .NET LOT.

Sr. No	Course	Duration (in PDs)	Functionality to be built
1	ASP.NET Core MVC, Entity Framework Core, SQL Server, ASP.NET Core Web API	3	Developing client app using Asp.net Core MVC , Services with Asp.net Core Web API, SQL Server database.