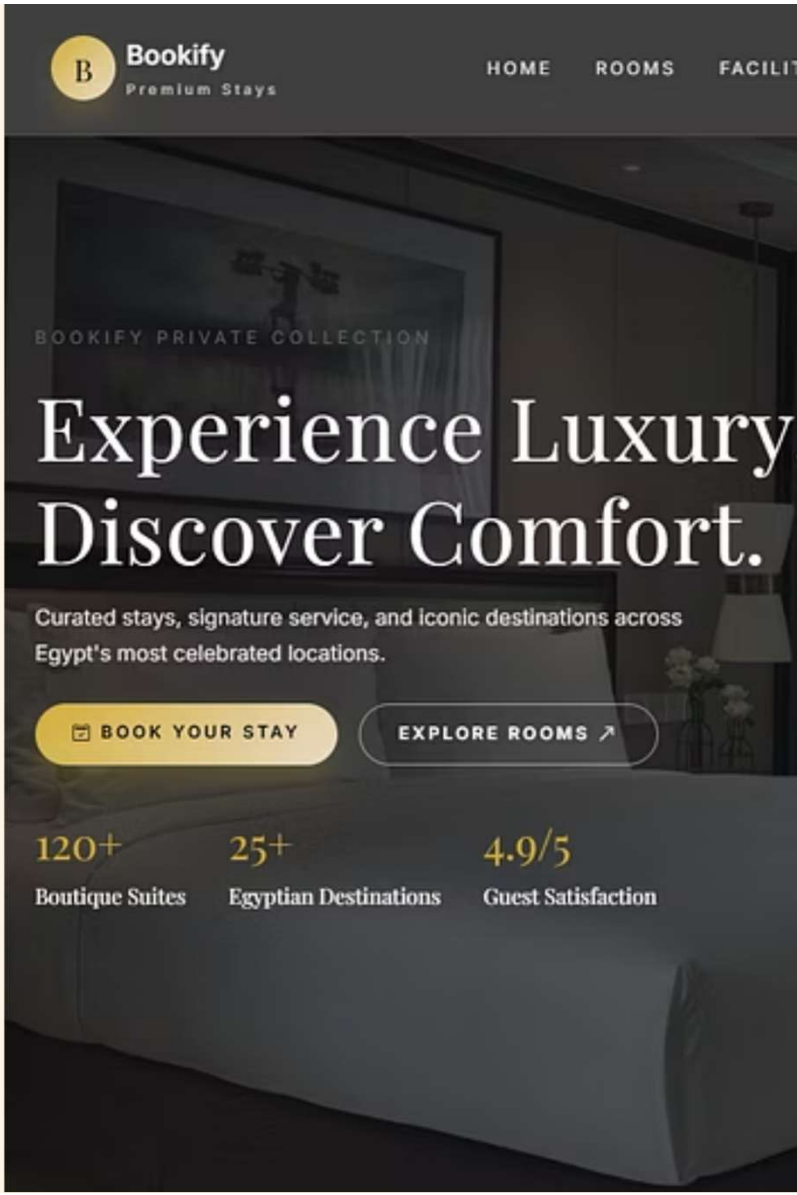
Bookify - Hotel Reservation System

A Full-Stack ASP.NET Core Web Application



Meet the Digital Egyptian Pioneers (DEPI) Team

Youssef Hossam Ibrahim

Full Stack .NET Web Developer

Mohamed Adel

Full Stack .NET Web Developer

Ahmed Ayman Ramadan

Full Stack .NET Web Developer

The DEPI team brings together expertise to deliver robust and scalable software solutions.

Project Overview: Bookify - Your Seamless Hotel Booking Solution

Project Type

Hotel Booking Platform

Target Users

Hotel customers and admin staff

Purpose

Allow customers to search, view, and book hotel rooms with integrated payment.

Key Value Proposition

A robust, scalable, and production-ready reservation system designed for efficiency and reliability.

Core Objectives: Building a World-Class Reservation System

01

Robust & Scalable Web Application

To develop a hotel reservation web application that is both highly reliable and capable of handling increasing user loads.

02

N-Tier Architecture Implementation

To ensure a clean separation of concerns, enhancing modularity and maintainability across the system layers.

03

Repository & Unit of Work Patterns

To create a maintainable codebase by abstracting data access logic and ensuring atomic transactions.

04

Secure Payment Processing with Stripe

To integrate a secure and reliable payment gateway for seamless and protected transactions.

05

Comprehensive Admin Panel

To provide powerful tools for hotel staff to efficiently manage rooms, bookings, and user data.

06

Production-Ready Quality

To deliver a system with built-in health checks and structured logging for operational excellence and easy debugging.

Customer-Facing Features: Enhancing the User Experience

- User Registration & Authentication

  Seamless onboarding and secure access for all users.

- Room Search & Filtering

  Advanced options by date, type, and price for easy room discovery.

- Detailed Room Viewing

  High-resolution images, comprehensive amenities, and transparent pricing.

- Reservation Cart Management

  Session-based cart to manage multiple room selections before booking.

- Booking Confirmation Process

  Clear steps for reviewing and confirming reservations.

- Secure Payment Processing

  Stripe integration ensures encrypted and reliable transactions.

- Booking History Viewing

  Access past and upcoming reservations with ease.

- Room Reviews & Ratings

  Allow users to share their experiences and help others choose.

- Favorite Rooms Management

  Users can save preferred rooms for future consideration.

Admin Functionality: Empowering Hotel Management

Admin Dashboard

Real-time statistics and overview of hotel operations.

Room Type Management

CRUD operations for efficient management of room categories.

Individual Room Management

Create, read, update, and delete specific room details.

Booking Management

Comprehensive tools to view, update, and cancel bookings.

User Management

Administer user accounts and permissions effectively.

Payment Transaction Monitoring

Track and review all financial transactions securely.

Non-Functional Requirements: Ensuring System Integrity

Performance

Fast page load times and efficient database queries for a smooth user experience.

Security

ASP.NET Identity, role-based authorization, secure payment processing, and SQL injection prevention.

Scalability

N-Tier architecture and repository pattern support horizontal scaling and maintainability.

Reliability

Health checks endpoint, Serilog for structured logging, and robust error handling.

Usability

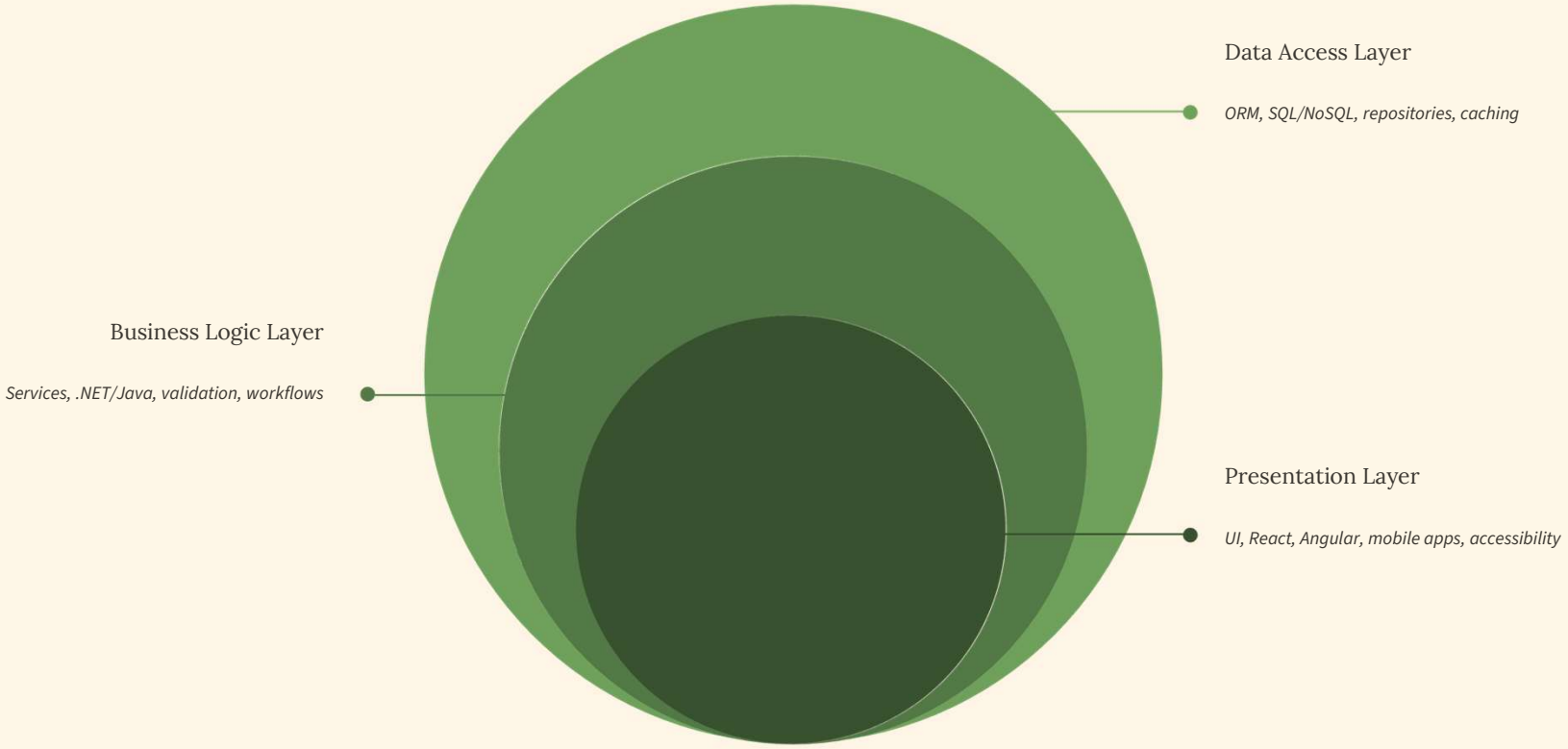
Responsive Bootstrap UI, intuitive navigation, and informative toast notifications.

Maintainability

Clean code architecture, separation of concerns, and dependency injection for long-term viability.

System Architecture Overview: The N-Tier Approach

Bookify employs a robust N-Tier architecture to ensure a clear separation of concerns, promoting modularity, scalability, and maintainability.



Key Design Patterns: Foundations of a Maintainable System

N-Tier Architecture

Achieves clear separation of Presentation, Business Logic, and Data Access layers.

Repository Pattern

Abstracts data operations, providing a clean API for accessing data sources.

Unit of Work Pattern

Ensures atomic transactions for complex operations, maintaining data consistency.

Dependency Injection

Reduces coupling between components, enhancing testability and flexibility through constructor injection.

Technology Stack: Powering Bookify's Performance

Backend Framework

ASP.NET Core 8.0 MVC

Database

SQL Server with Entity Framework Core 8.0

Authentication

ASP.NET Identity with JWT Bearer tokens

Payment Gateway

Stripe.net SDK

Frontend Libraries

Bootstrap 5, jQuery, DataTables, Toaster.js

Logging & Monitoring

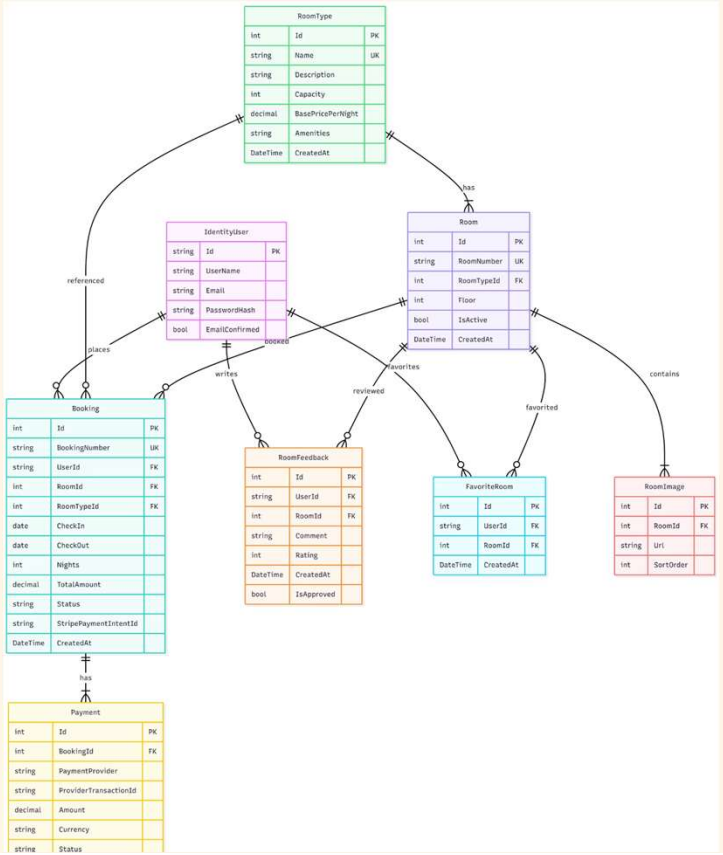
Serilog for structured logging

Development Tools

Visual Studio, SQL Server Management Studio

Database Design: Entity Relationship Diagram (ERD)

A well-structured database is the backbone of any reliable system. The Bookify ERD illustrates the relationships between our key entities, ensuring data integrity and efficient querying.



Database Schema Details

Our database schema is meticulously designed to support Bookify's complex operations while optimising for performance and maintainability. Key elements include:

1

Key Tables

Eight primary tables manage users, room types, rooms, images, bookings, payments, feedback, and favourites, forming the system's core.

2

Relationships

Crucial one-to-many relationships link RoomType to Rooms, Room to Bookings, Room to RoomImages, and Booking to Payments, defining data flow.

3

Indexes

Unique indexes on `RoomNumber` *and* `BookingNumber`*, alongside composite indexes, are implemented to accelerate query performance.*

4

Constraints

Foreign key constraints, with carefully considered delete behaviours, maintain referential integrity. Unique constraints prevent duplicate critical data.

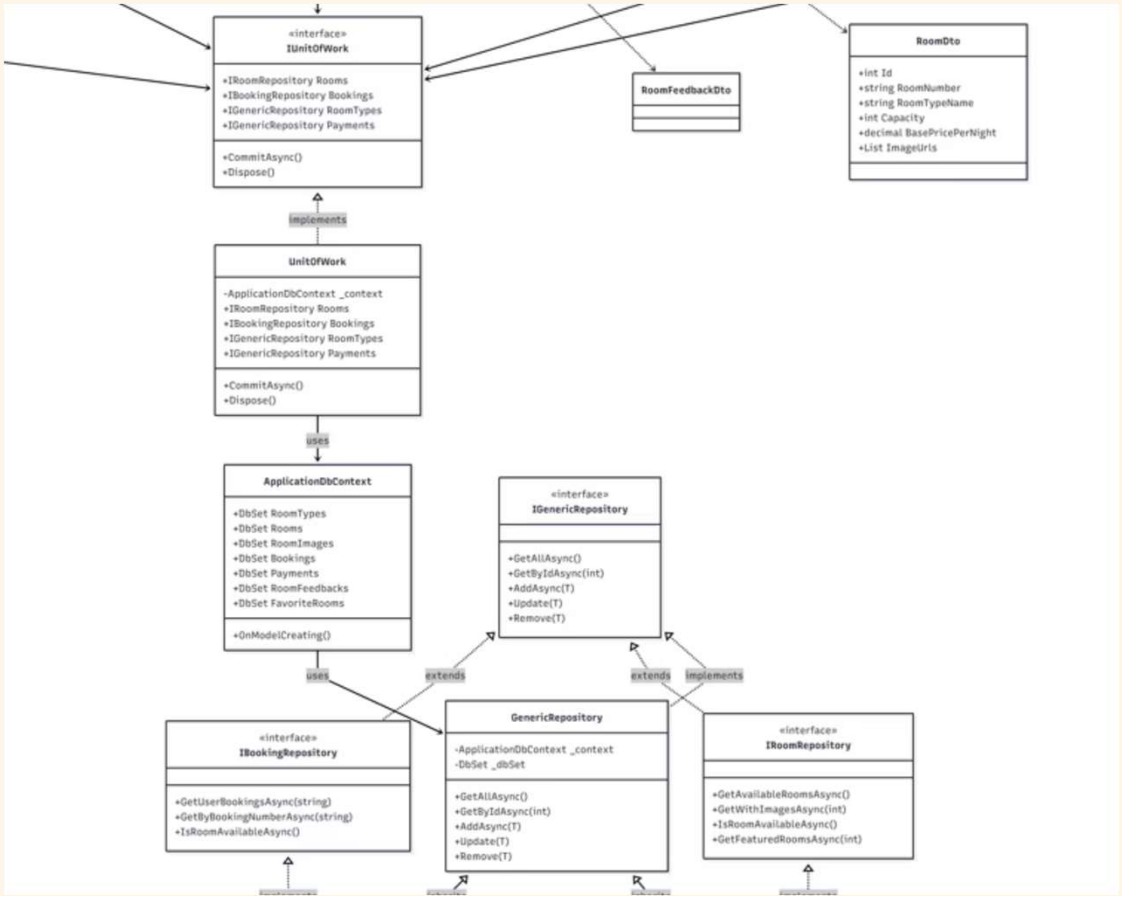
5

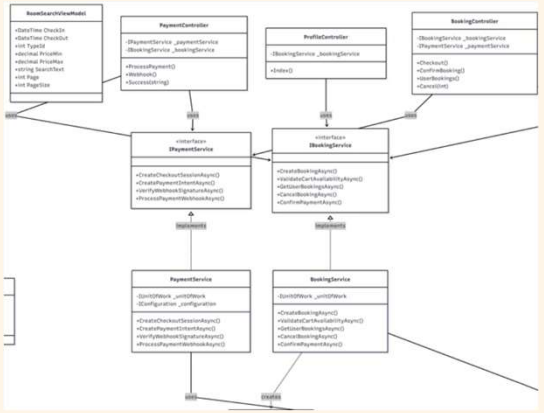
Data Types

Strict data type enforcement, including `Decimal` *for prices,* `Date` *for check-in/out, and* `String` *with defined max lengths, ensures data consistency.*

Class Diagram: Data Layer

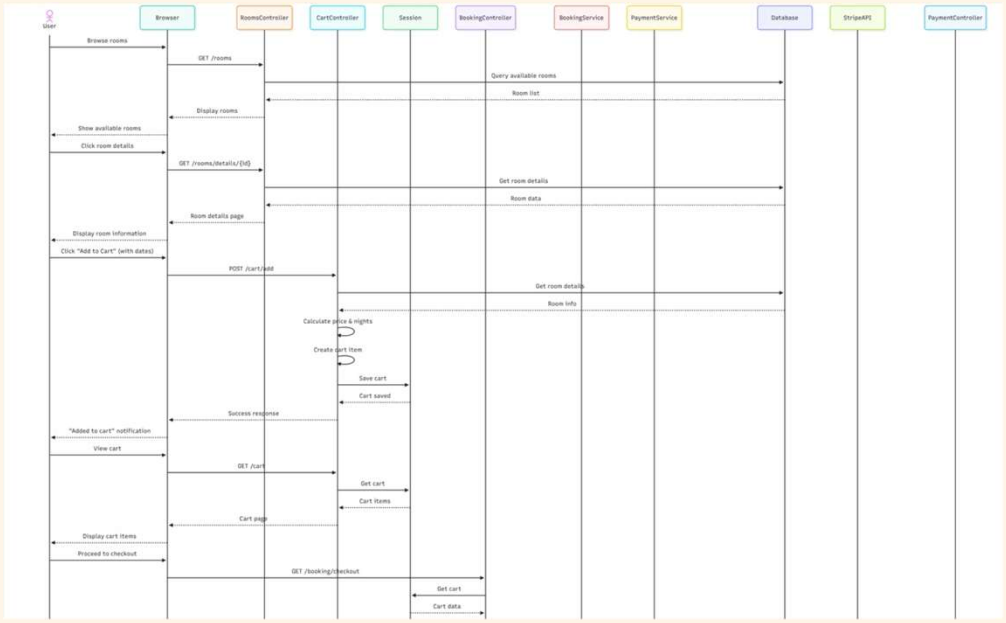
The Data Access Layer abstracts database interactions, promoting a clean architecture. It leverages Entity Framework Core with generic and specialised repositories.

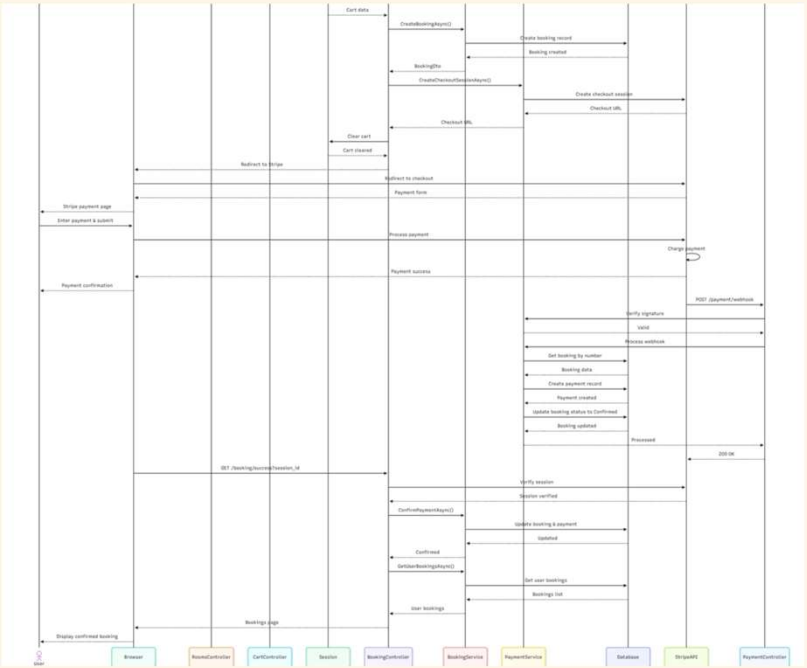




Sequence Diagram: Booking Flow

This diagram illustrates the step-by-step interaction during a typical booking process, from initial room search to final booking creation, highlighting key system components.





Core Features: Room Management

Bookify offers comprehensive room management capabilities, designed for both end-users searching for accommodation and administrators maintaining the inventory.

Room Catalogue Display

Browse all available rooms with intuitive filtering options.

Advanced Search

Filter rooms by date, type, price, and capacity for precise results.

Room Details Page

Detailed view including image galleries, amenities, pricing, availability, and guest reviews.

Room Type Management

Admin functions to create, edit, and delete room types with associated pricing structures.

Individual Room Management

Admins can add/edit rooms, assign numbers, and manage individual room availability.

Image Management

Support for multiple images per room with robust upload functionality.

Core Features: Reservation System, Payment & Admin

Bookify provides a seamless reservation experience, secure payment processing, and a powerful administrative panel for system oversight.

Session-Based Cart

Temporary selection of rooms, persisting throughout the user's session.

Availability Validation

Real-time checks to prevent double bookings and ensure accuracy.

Booking Creation & Confirmation

Atomic transactions generate unique booking numbers, followed by detailed confirmations.

Booking History & Cancellation

Users can view past/upcoming bookings and manage cancellations.

Stripe Integration

Secure payment processing with payment intent creation and webhook handling.

Admin Dashboard

Statistics, recent bookings, and revenue summaries for quick insights.

Admin Room Management

CRUD operations for room types and rooms, including bulk actions.

Role-Based Access Control

Secure access for Admin and Customer roles, protecting sensitive routes.

Implementation Details: The Repository Pattern

`GenericRepository`

Implements common CRUD operations: `Add`, `GetById`, `GetAll`, `Update`, `Delete`, *and* `Find` *with predicates.*

Specific Repositories

- `RoomRepository`*:* `IsRoomAvailableAsync`, `GetWithImagesAsync`, `GetAvailableRoomsAsync`.
- `BookingRepository`*:* `GetByBookingNumberAsync`, `GetUserBookingsAsync`.

Benefits

- *Abstraction of data access logic.*
- *Enhanced testability through mocking.*
- *Centralised data access management.*
- *Facilitates easy swapping of data sources.*

Implementation Details: The Unit of Work Pattern

`UnitOfWork` Class

Manages all repositories under a single `DbContext` *instance, providing a* `CommitAsync` *method for transaction handling.*

Transaction Management

Ensures atomic operations, for instance, booking creation and room availability updates occur together, with rollbacks on failure.

Repository Access & Benefits

- *Lazy initialization of repositories.*
- *Single point of access for all repository interactions.*
- *Ensures data consistency and reduces database round trips.*
- *Maintains a clean separation of concerns.*

Implementation Details: Dependency Injection

1

Service Registration

All services are registered in `Program.cs` *with a Scoped lifetime. Repositories are registered via the* `IUnitOfWork` *interface.*

2

Constructor Injection

Controllers and services receive their dependencies (e.g., `IUnitOfWork`*) through constructor parameters, eliminating hard dependencies.*

3

Benefits

- *Promotes loose coupling between components.*
- *Enhances testability and maintainability.*
- *Strictly adheres to SOLID principles, especially the Dependency Inversion Principle.*

Robust Security Implementation

Authentication

Utilises ASP.NET Identity for user management, JWT Bearer tokens for API authentication, and Cookie authentication for MVC views.

Authorisation

Role-based access control (Admin/Customer roles) enforced with `[Authorize]` *attributes and dedicated Admin area protection.*

Password Security

Strong password requirements (8+ chars, diverse characters) and secure password hashing are implemented.

Payment Security

Stripe is used for secure payment processing, including webhook signature verification and zero sensitive data storage.

SQL Injection Prevention

Entity Framework Core's parameterized queries protect against SQL injection; raw SQL with user input is strictly avoided.

Logging and Monitoring

Serilog Integration

Structured logging is configured for console and file output, with daily log file rotation to manage storage.

Log Levels

Comprehensive logging across the application at Information, Warning, Error, and Fatal levels for detailed insights.

Health Checks

ASP.NET Core Health Checks endpoint at `/health` *monitors SQL Server connectivity and overall application status.*

Error Handling & Benefits

Global exception handling provides user-friendly error pages and detailed error logging for easy debugging, production monitoring, and audit trails.

User Interface Design & Experience

Frontend Technologies

- ***Bootstrap 5**: For responsive design across devices.*
- ***jQuery**: For dynamic DOM manipulation and AJAX calls.*
- ***DataTables**: Enhances table functionality with sorting and pagination.*
- ***Toaster.js**: Provides non-intrusive notifications for user feedback.*

Key Design Principles & Pages

- ***Design Principles**: Responsive, intuitive navigation, consistent styling, and user-friendly forms.*
- ***Key Pages**: Home, Room Listing, Room Details, Cart, Checkout, and Admin Dashboard.*
- ***User Experience**: Toast notifications, loading indicators, form validation, and smooth transitions ensure a positive user journey.*

Key Achievements and Project Highlights

Architecture Excellence

Successful implementation of N-Tier architecture with clean separation of concerns, resulting in a highly maintainable codebase.

Design Patterns

Proper implementation of Repository, Unit of Work, and Dependency Injection patterns.

Production Ready

Integrated health checks, structured logging, global error handling, and robust security best practices.

Feature Complete

Delivered a full booking flow, secure payment integration, a comprehensive admin panel, and user management.

Code Quality

Adherence to SOLID principles, clean code practices, and comprehensive documentation.

Conclusion & Future Enhancements

Project Summary

Bookify has successfully delivered a production-ready hotel reservation system, adhering to best practices, design patterns, and a comprehensive feature set for both customers and administrators.

Technologies Mastered

ASP.NET Core MVC, Entity Framework Core, ASP.NET Identity, Stripe integration, and Serilog logging.

Future Enhancements

- *Email notifications for bookings.*
- *Advanced reporting and analytics.*
- *Mobile application development.*
- *Multi-language support.*
- *Real-time availability updates.*
- *Loyalty program integration.*

> *Thank you for your attention.*