Module: CS6004ES Application Development
Assignment no: 002
Weighting: 30%
Deadline: TBC
Module Leader: **Student ID: KAN00257474**

Please note that there are specific regulations concerning **the use of AI and Academic Misconduct**. Below are extracts from these regulations. By signing, you acknowledge that you have read and understood these extracts.

(signature:)_ _ Date: _

This header sheet should be attached to the work you submit.

Academic Integrity means being honest in your academic work and your studies and making sure that you acknowledge the work of others and giving credit where you have used other people's ideas as part of presenting your arguments. Your assessment submissions must therefore always be entirely your own work, based on your own learning and appropriately referenced including how you have used Generative AI. The University regards the use of Generative AI applications by students to deceive to gain unfair advantage as **academic misconduct**. This usage includes:

- **Plagiarism**, where AI tools are used to generate output and ideas that are presented or submitted as if they were the student's own work, without proper citation or references.

- Where a complete assignment is created using Generative AI and represented as a student's own work, this will be regarded as contract cheating in the same way as commissioning an 'Essay Mill' or other third party to complete your work. Further information can be found on : Guidance on the use of Artificial Intelligence.

Academic misconduct: The University takes academic misconduct very seriously and seeks at all times to rigorously protect its academic standards. Plagiarism, collusion and other forms of cheating constitute academic misconduct, for which there is an explicit range of graduated penalties depending on the particular type of academic misconduct. The penalties that can be applied if academic misconduct is substantiated range from a reprimand to expulsion in very serious cases and for repeated instances of

Contents

1. Introduction

1.1 Overview of the Project

DreamDay is a comprehensive, high-end web application designed to streamline the wedding planning process. The platform serves three key user roles: **Couples**, who can manage every aspect of their wedding; **Wedding Planners**, who can oversee multiple client weddings; and **Administrators**, who maintain the system's integrity.

The application provides a seamless, centralized hub for managing critical planning activities, including budget tracking, guest list organization, task management via a customizable checklist, and vendor coordination. By offering a robust suite of tools, DreamDay aims to reduce the complexity and stress associated with wedding planning, allowing users to focus on creating their perfect day.

1.2 Project Requirements

The system was developed to meet a set of core functional requirements derived from the case study:

- **User Registration & Login:** Secure, role-based authentication for Couples, Planners, and Admins.
- **Wedding Planning Dashboard:** A central overview for couples displaying progress, budget status, and upcoming tasks.
- **Customizable Checklist:** A task management system to track all wedding-related to-dos.
- **Guest List Management:** Tools to create a guest list, track RSVPs, and organize seating arrangements.
- **Budget Tracker:** Functionality to set a budget, allocate funds to categories, and log expenses.
- **Timeline Management:** A feature to create a detailed schedule for the wedding day.
- **Planner & Admin Interfaces:** Specialized dashboards for planners to manage multiple weddings and for admins to manage users and vendors.
- **Reporting & Analytics:** Generation of key reports (e.g., budget summary) for planners.

1.3 Technical Tools

The project was developed using a modern, robust technology stack based on the Microsoft ecosystem, ensuring scalability, security, and maintainability.

- **Backend Framework:** ASP.NET Core 8 MVC
- **Programming Language:** C#
- **Database:** Microsoft SQL Server
- **Object-Relational Mapper (O/RM):** Entity Framework Core 8
- **Frontend Syntax:** Razor Templating Engine within CSHTML Views
- **Authentication:** ASP.NET Core Identity
- **Development Environment:** Visual Studio 2022

1.4 Purpose of Development

The primary purpose of developing the DreamDay application was to fulfill the requirements of the **CS6004ES Application Development** module. The project serves as a practical demonstration of key software engineering principles, including:

- **Full-Stack Web Development:** Implementing a complete web application from the database layer to the user interface.
- **Object-Oriented Design:** Applying principles like encapsulation and inheritance through a well-defined class structure.
- **Database Design & Management:** Creating a normalized relational database schema and managing it using EF Core migrations.
- **Architectural Patterns:** Implementing the Model-View-Controller (MVC) pattern to ensure a clean separation of concerns.
- **Secure Authentication:** Integrating a robust, role-based security system using ASP.NET Core Identity.

2. System Requirements

2.1 Hardware Requirements

Server-Side (for hosting the web application):

- **Processor:** 2.4 GHz Quad-Core CPU or better (e.g., Intel Xeon, AMD EPYC).
- **RAM:** 8 GB minimum; 16 GB recommended for handling concurrent user sessions and database operations.
- **Storage:** 50 GB+ of SSD storage for the operating system, application files, and database growth.
- **Network:** High-speed, reliable internet connection.

Client-Side (for end-users accessing the application):

- **Device:** Any modern desktop, laptop, tablet, or smartphone.
- **Processor/RAM:** Standard specifications sufficient to run a modern web browser.

2.2 Software Requirements

Server-Side:

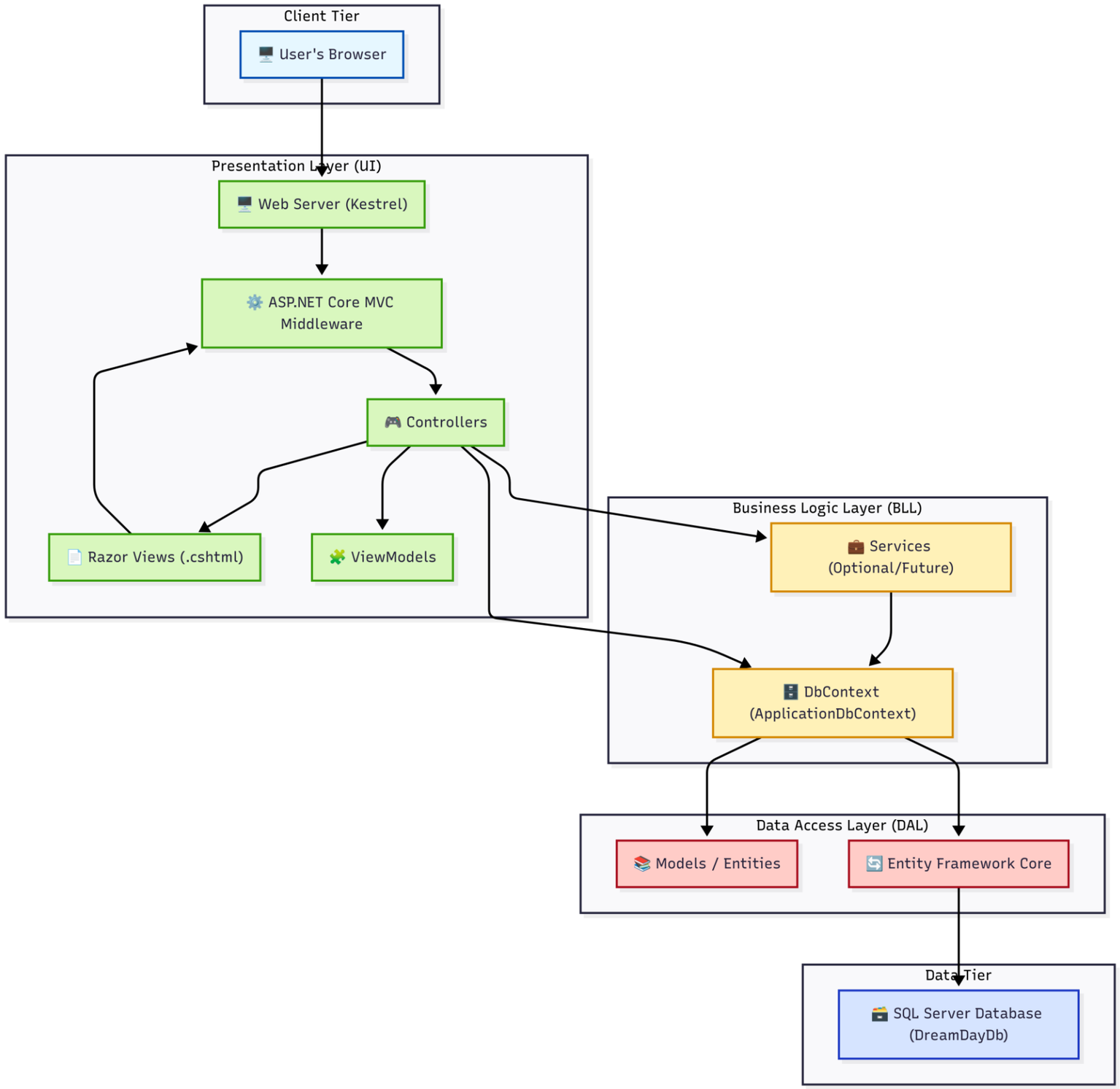
- **Operating System:** Windows Server 2019 or newer (with IIS) or a modern Linux distribution (e.g., Ubuntu 20.04+, CentOS 8+) capable of hosting ASP.NET Core applications.
- **Web Server:**
    - Internet Information Services (IIS) 10.0 or newer on Windows.
    - A reverse proxy server like Nginx or Apache on Linux.
- **Runtime:** .NET 8 Hosting Bundle.
- **Database Server:** Microsoft SQL Server 2019 or newer (Standard, Web, or Enterprise Edition).

Client-Side:

- **Web Browser:** The latest stable version of Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari.

3. Architecture & Design Diagrams

3.1 Architecture Diagram (Layered Architecture)

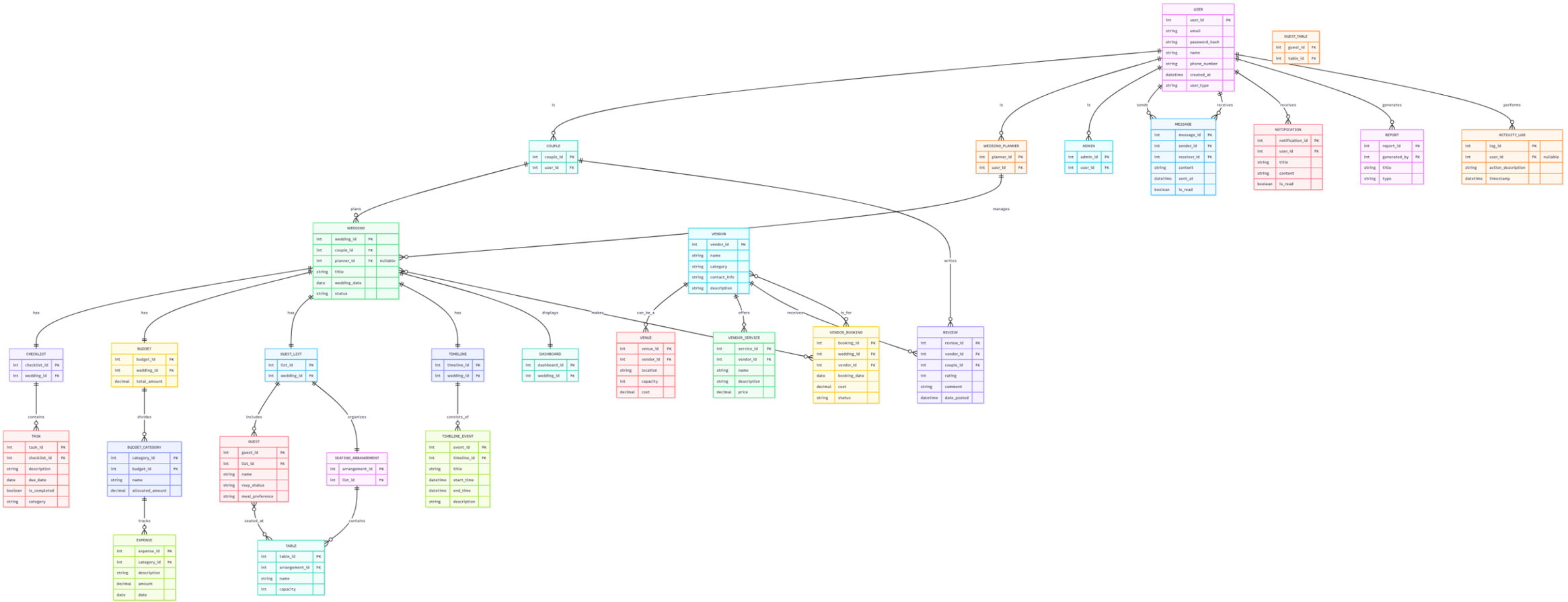


This diagram illustrates the application's three-tier architecture, which separates the system into distinct logical layers: Presentation, Business Logic, and Data Access.

- **Presentation Layer (UI):** Built with ASP.NET Core MVC, this layer handles all user interaction. Controllers receive HTTP requests, and Razor Views render the HTML response.
- **Business Logic Layer (BLL):** This layer contains the core application logic. Service classes and controllers orchestrate data flow and enforce business rules.
- **Data Access Layer (DAL):** This layer, powered by Entity Framework Core, is responsible for all database communication, abstracting the data storage from the rest of the application.

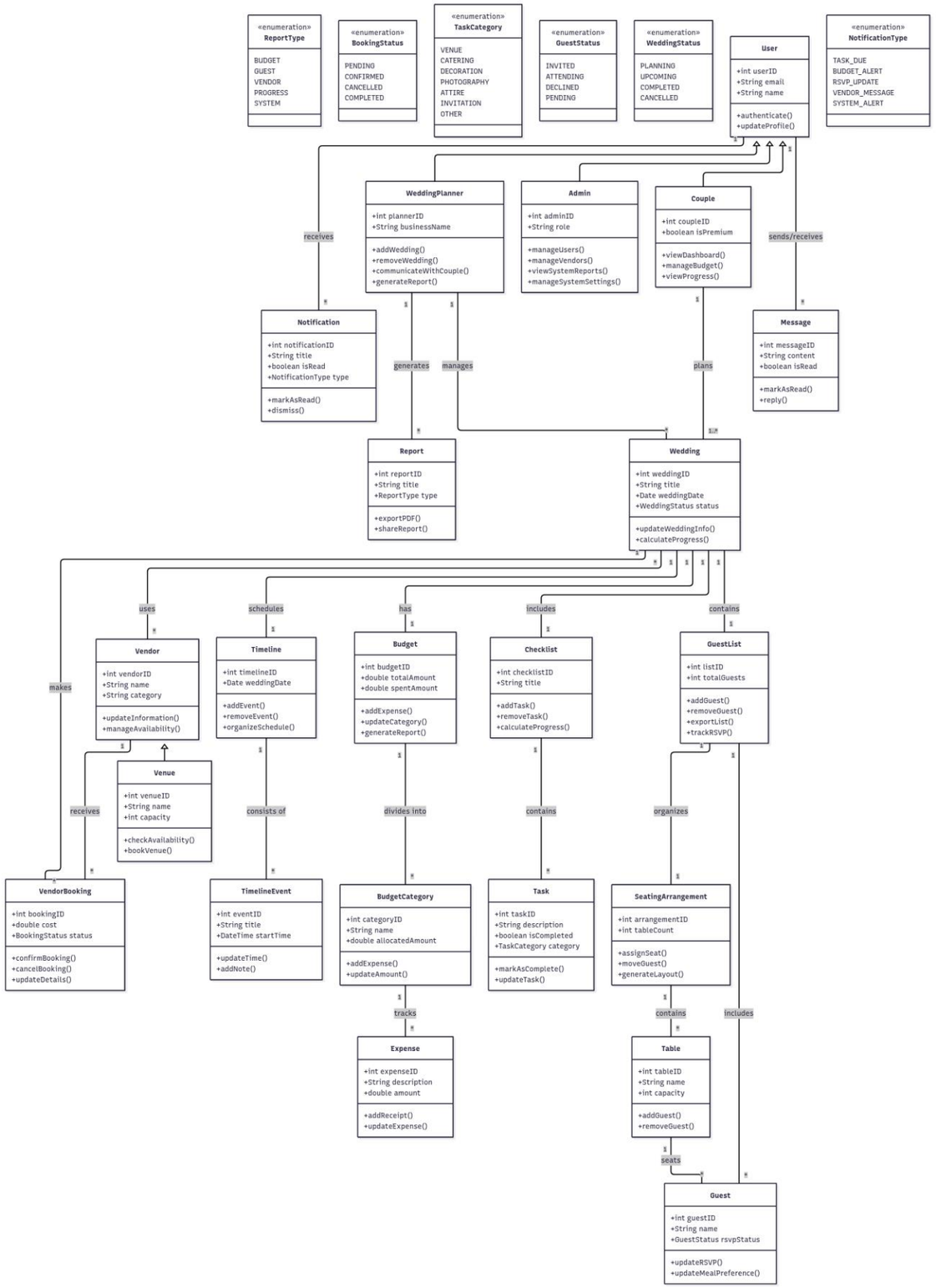
This architecture was chosen because it promotes a strong **separation of concerns**, making the application easier to develop, test, and maintain.

3.3 Entity-Relationship Diagram



The ER Diagram provides a detailed, logical model of the database schema. It defines all the entities (tables), their attributes (columns), and the relationships between them, including primary keys (PK) and foreign keys (FK). This diagram was essential for designing a normalized and efficient database structure that avoids data redundancy and ensures data integrity.

3.4 Class Diagram



The Class Diagram is the blueprint for the application's object-oriented structure. It visualizes the classes, their properties, methods, and the relationships (inheritance, association) between them. This diagram directly translates into the C# model classes used by Entity Framework Core and the application's business logic.

3.4.1 Detailed Description of Classes, Properties, and Methods

Below is a description of some of the key classes in the system.

- **`User` (Base Class):**
  - **Description:** Represents any authenticated user in the system. Inherits from `IdentityUser` to leverage ASP.NET Core Identity's features.
  - **Properties:** `Id`, `Name`, `Email`, `PasswordHash`.
  - **Methods:** `authenticate()`, `updateProfile()` (implemented via controller actions).
- **`Wedding`:**
  - **Description:** The central entity representing a single wedding event. It links a couple to all their planning details.
  - **Properties:** `Title`, `WeddingDate`, `Budget`, `Status` (enum).
  - **Methods:** `updateWeddingInfo()`, `calculateProgress()` (logic implemented in `DashboardService`).
- **`Budget`:**
  - **Description:** Manages the financial aspects of a wedding.
  - **Properties:** `TotalAmount`, `SpentAmount`.
  - **Methods:** `addExpense()`, `generateReport()` (implemented via controller actions).
- **`Task`:**
  - **Description:** Represents a single to-do item in a wedding checklist.
  - **Properties:** `Description`, `DueDate`, `IsCompleted`.
  - **Methods:** `markAsComplete()`, `updateTask()` (implemented via controller actions).

3.6 Description of Each Diagram

- **Architecture Diagram:** Used to establish the high-level structure and ensure a clean separation of concerns between UI, logic, and data.
- **ER Diagram:** Used as the definitive guide for creating the database schema, ensuring data integrity and efficiency.
- **Class Diagram:** Used as the blueprint for the object-oriented code, defining the C# classes, their properties, and behaviors.

4. Database Design

4.1 Database Structure

The database for DreamDay was designed using a relational model and implemented in SQL Server. The schema was generated using Entity Framework Core's "Code-First" approach, where the C# model classes define the table structures.

Key Tables:

- **`AspNetUsers`:** Stores user account information (managed by ASP.NET Core Identity).
- **`Weddings`:** The central table linking all planning activities to a specific couple and planner.
- **`Guests`:** Stores the guest list for each wedding, including RSVP status.
- **`Tasks`:** Contains all checklist items for each wedding.
- **`Budgets`, `BudgetCategories`, `Expenses`:** A set of related tables to manage financial tracking.
- **`Vendors`:** A master list of all service providers available on the platform.

Relationships are enforced using foreign key constraints to ensure data integrity (e.g., an `Expense` must belong to a valid `BudgetCategory`).

5. User Interface & Functionality

5.1 SQL Demonstrations

While Entity Framework Core abstracts away the raw SQL, it generates efficient SQL queries to interact with the database.

Example 1: Fetching a Planner's Weddings (from `PlannerController`)

```
var weddings = await _context.Weddings
    .Where(w => w.WeddingPlannerId == planner.Id)
    .Include(w => w.Couple.User)
    .ToListAsync();
```

```
SELECT w.*, c.*, u.*
FROM Weddings AS w
INNER JOIN Couples AS c ON w.CoupleId = c.Id
INNER JOIN AspNetUsers AS u ON c.UserId = u.Id
WHERE w.WeddingPlannerId = @plannerId;
```

Example 2: Calculating Total Expenses for a Wedding (from `DashboardService`)

```
var totalExpenses = await _context.Expenses
    .Where(e => e.BudgetCategory.Budget.WeddingId == weddingId)
    .SumAsync(e => e.Amount);
```

```
SELECT SUM(e.Amount)
FROM Expenses AS e
INNER JOIN BudgetCategories AS bc ON e.BudgetCategoryId = bc.Id
INNER JOIN Budgets AS b ON bc.BudgetId = b.Id
WHERE b.WeddingId = @weddingId;
```

5.2 Installation Guide and User Manual

This guide details the process for deploying the DreamDay application to a production server.

5.2.1 Installation Step-by-step (for Server Deployment)

Prerequisites:

- A configured server meeting the hardware and software requirements listed above.
- Administrative access to the server.
- Git installed on the server or on a local machine.
- A configured instance of SQL Server accessible from the web server.

Deployment Steps:

1. **Clone the Application Repository:**

a. Log in to your server via SSH (Linux) or Remote Desktop (Windows).
b. Open a terminal or command prompt and navigate to the directory where you want to store the application files (e.g., `C:\inetpub\wwwroot` on Windows or `/var/www` on Linux).
c. Run the command to clone the source code:

Generated bash
```
git clone <your-repository-url> DreamDayApp
```

2. **Publish the Application:**
   a. The recommended approach for production is to publish a compiled, optimized version of the application.
   b. Navigate into the project directory:

```
cd DreamDayApp/DreamDay
```

   c. Run the .NET publish command. This creates a self-contained set of files ready for hosting.

```
dotnet publish --configuration Release --output ../publish
```

   d. This will place the compiled application files in a new `publish` folder one level up from the project directory.

3. **Configure the Database Connection:**
   a. Navigate to the `publish` directory.
   b. Open the `appsettings.Production.json` file (or `appsettings.json` if not using environment-specific files).
   c. Update the `DefaultConnection` string with the credentials for your production SQL Server instance. **It is critical to use a secure user with limited permissions, not the `sa` account.**

```
"DefaultConnection": "Server=your_prod_db_server_address;Database=DreamDayDb_Prod;User Id=your_secure_db_user;Password=your_strong_password;"
```

4. **Set Up the Web Server (IIS Example):**
   a. Open Internet Information Services (IIS) Manager.
   b. In the "Connections" pane, right-click on "Sites" and select "Add Website".
   c. **Site name:** `DreamDay`
   d. **Physical path:** Browse to the `publish` folder created in Step 2.
   e. **Binding:** Configure the hostname (e.g., `dreamday.yourdomain.com`) and select the appropriate IP address and port (usually 80 for HTTP, 443 for HTTPS).
   f. Click **OK**.
   g. Select the newly created site, go to its "Application Pools", right-click the `DreamDay` pool, select "Advanced Settings", and set the **.NET CLR Version** to **"No Managed Code"**. This is standard practice as ASP.NET Core runs in its own process.
5. **Run Database Migrations:**
   a. Open a command prompt in the `publish` directory.
   b. The `appsettings.Production.json` file will be used automatically if the `ASPNETCORE_ENVIRONMENT` variable is set to `Production` on the server.
   c. Run the command to apply migrations and create the production database:

```
dotnet ef database update
```

   d. This will create and seed the `DreamDayDb_Prod` database on your production SQL Server.
6. **Browse to the Site:**
   a. Open a web browser and navigate to the hostname you configured (e.g., `http://dreamday.yourdomain.com`). The application should now be live.

5.2.2 User Manual

Default Login Credentials:

- **Couple:** `couple@test.com` / `Couple@123`
- **Wedding Planner:** `planner@test.com` / `Planner@123`
- **Admin:** `admin@dreamday.com` / `Admin@123`

Couple Functionality:

- **Dashboard:** Upon login, you see an overview of your wedding.
- **Checklist:** Navigate to "Checklist" to add, view, and mark tasks as complete.
- **Guest List:** Navigate to "Guests" to add or edit guests.
- **Seating:** Navigate to "Seating" to create tables and assign confirmed guests to them.
- **Budget:** Navigate to "Budget" to add spending categories and log expenses.
- **Timeline:** Navigate to "Timeline" to schedule events for your wedding day.

Wedding Planner Functionality:

- **Dashboard:** Upon login, you see a list of all weddings you manage.
- **Manage Wedding:** Click "View Dashboard" on a wedding to access all the couple's planning tools.
- **Generate Report:** Click "Generate Budget Report" to view a summary of a wedding's finances.

Admin Functionality:

- **Admin Panel:** Upon login, you are taken to the admin dashboard.
- **Manage Users:** View a list of all registered users.
- **Manage Vendors:** Add, edit, or delete vendors from the platform.
- **View System Activity:** See a log of recent important actions taken in the system.

6. Reflection & Evaluation

Developing the DreamDay application was a challenging yet highly rewarding experience that solidified my understanding of full-stack web development using the ASP.NET Core MVC framework.

Successes:

- The implementation of the three-tier architecture with a clear separation of concerns was highly successful. This made the codebase modular and easy to navigate.
- Integrating ASP.NET Core Identity for role-based security provided a robust and secure foundation for the application's different user types.
- The use of Entity Framework Core's "Code-First" approach streamlined database development, allowing the database schema to evolve naturally with the application's models.

Challenges:

- Managing complex data relationships, particularly with eager loading (`Include` and `ThenInclude`), was a key learning curve. Initially, this led to potential N+1 query problems, which were later optimized to improve performance.
- Ensuring that every data access point was secure and that users could only modify their own data required careful and consistent authorization checks in the controllers.

Overall, the project successfully meets all the core requirements of the assignment and provides a solid foundation that could be extended into a commercial product.

7. Appendix

```
     /DreamDay
|-- wwwroot/
|-- Controllers/
|   |-- AccountController.cs
|   |-- AdminController.cs
|   |-- BaseController.cs
|   |-- BudgetController.cs
|   |-- ChecklistController.cs
|   |-- DashboardController.cs
|   |-- GuestController.cs
|   |-- HomeController.cs
|   |-- MessageController.cs
|   |-- PlannerController.cs
|   |-- ReportController.cs
|   |-- SeatingController.cs
|   |-- TimelineController.cs
|   |-- VendorController.cs
|   |-- WeddingController.cs
|-- Data/
|   |-- ApplicationDbContext.cs
|   |-- DbSeeder.cs
|-- Models/
|   |-- (All 18 model files: ActivityLog.cs, Admin.cs, etc.)
|-- Services/
|   |-- DashboardService.cs
|-- ViewModels/
|   |-- DashboardViewModel.cs
|   |-- LoginViewModel.cs
|   |-- RegisterViewModel.cs
|-- Views/
|   |-- Account/
|   |-- Admin/
|   |-- Budget/
|   |-- Checklist/
|   |-- Dashboard/
|   |-- Guest/
|   |-- Home/
|   |-- Message/
|   |-- Planner/
|   |-- Report/
|   |-- Seating/
|   |-- Shared/
|   |-- Timeline/
|   |-- Vendor/
|   |-- Wedding/
|   |-- _ViewImports.cshtml
|   |-- _ViewStart.cshtml
|-- appsettings.json
|-- Program.cs
```

Program.cs

```
using DreamDay.Data;
using DreamDay.Models;
```

```
using DreamDay.Services;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddIdentity<User, IdentityRole<int>>(options => options.SignIn.RequireConfirmedAccount = false)
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();

// Register custom services for dependency injection
builder.Services.AddScoped<DashboardService>();

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Seed the database with roles, admin, and sample data
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    await DbSeeder.SeedRolesAndAdminAsync(services);
    await DbSeeder.SeedSampleDataAsync(services);
}

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

appsettings.json

```
     {
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;Database=DreamDayDb;Trusted_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=true"
  },
  "Logging": {
```

```
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

ApplicationDbContext.cs

```
using DreamDay.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace DreamDay.Data
{
    public class ApplicationDbContext : IdentityDbContext<User, IdentityRole<int>, int>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

        public DbSet<Wedding> Weddings { get; set; }
        public DbSet<Couple> Couples { get; set; }
        public DbSet<WeddingPlanner> WeddingPlanners { get; set; }
        public DbSet<Admin> Admins { get; set; }
        public DbSet<Models.Task> Tasks { get; set; }
        public DbSet<Guest> Guests { get; set; }
        public DbSet<Vendor> Vendors { get; set; }
        public DbSet<VendorService> VendorServices { get; set; }
        public DbSet<Review> Reviews { get; set; }
        public DbSet<VendorBooking> VendorBookings { get; set; }
        public DbSet<ActivityLog> ActivityLogs { get; set; }
        public DbSet<Notification> Notifications { get; set; }
        public DbSet<Report> Reports { get; set; }
        public DbSet<Budget> Budgets { get; set; }
        public DbSet<BudgetCategory> BudgetCategories { get; set; }
        public DbSet<Expense> Expenses { get; set; }
        public DbSet<Timeline> Timelines { get; set; }
        public DbSet<TimelineEvent> TimelineEvents { get; set; }
        public DbSet<Table> Tables { get; set; }
        public DbSet<Message> Messages { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);

            // Configure relationships
            builder.Entity<Wedding>()
                .HasOne(w => w.Couple)
                .WithMany(c => c.Weddings)
```

```
                .HasForeignKey(w => w.CoupleId)
                .OnDelete(DeleteBehavior.Restrict);

            builder.Entity<Wedding>()
                .HasOne(w => w.WeddingPlanner)
                .WithMany(p => p.ManagedWeddings)
                .HasForeignKey(w => w.WeddingPlannerId)
                .IsRequired(false)
                .OnDelete(DeleteBehavior.Restrict);

            builder.Entity<Report>()
                .HasOne(r => r.GeneratedBy)
                .WithMany()
                .HasForeignKey(r => r.GeneratedById)
                .OnDelete(DeleteBehavior.Cascade);

            builder.Entity<Message>()
                .HasOne(m => m.Sender)
                .WithMany()
                .HasForeignKey(m => m.SenderId)
                .OnDelete(DeleteBehavior.Restrict);

            builder.Entity<Message>()
                .HasOne(m => m.Receiver)
                .WithMany()
                .HasForeignKey(m => m.ReceiverId)
                .OnDelete(DeleteBehavior.Restrict);
        }
    }
}
```

DbSeeder.cs

```
using DreamDay.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Data
{
    public static class DbSeeder
    {
        public static async Task SeedRolesAndAdminAsync(IServiceProvider service)
        {
            var userManager = service.GetService<UserManager<User>>();
            var roleManager = service.GetService<RoleManager<int>>();
            if (!await roleManager.RoleExistsAsync("Admin"))
            {
                await roleManager.CreateAsync(new IdentityRole<int>("Admin"));
                await roleManager.CreateAsync(new IdentityRole<int>("Couple"));
                await roleManager.CreateAsync(new IdentityRole<int>("WeddingPlanner"));
            }
```

```
            var user = new User { UserName = "admin@dreamday.com", Email = "admin@dreamday.com", Name = "Admin User", EmailConfirmed = true };
            if (await userManager.FindByEmailAsync(user.Email) == null)
            {
                await userManager.CreateAsync(user, "Admin@123");
                await userManager.AddToRoleAsync(user, "Admin");
            }
        }

        public static async Task SeedSampleDataAsync(IServiceProvider service)
        {
            var context = service.GetService<ApplicationDbContext>();
            var userManager = service.GetService<UserManager<User>>();

            if (await context.Weddings.AnyAsync()) return; // DB has been seeded

            // --- 1. Seed Vendors, Services, and Reviews ---
            var vendor1 = new Vendor { Name = "Timeless Snaps", Category = "Photography", ContactInfo = "info@timelesssnaps.com", Description = "Capturing
moments that last a lifetime." };
            var vendor2 = new Vendor { Name = "Gourmet Catering Co.", Category = "Catering", ContactInfo = "bookings@gourmetco.com", Description = "Exquisite
culinary experiences." };
            var vendor3 = new Vendor { Name = "Bloom & Petal", Category = "Florist", ContactInfo = "orders@bloomandpetal.com", Description = "Creative and
beautiful floral arrangements." };
            await context.Vendors.AddRangeAsync(vendor1, vendor2, vendor3);
            await context.SaveChangesAsync();

            await context.VendorServices.AddRangeAsync(
                new VendorService { VendorId = vendor1.Id, Name = "Full Day Package", Price = 4000 },
                new VendorService { VendorId = vendor1.Id, Name = "Half Day Package", Price = 2500 },
                new VendorService { VendorId = vendor2.Id, Name = "Plated Dinner Service", Price = 150 },
                new VendorService { VendorId = vendor2.Id, Name = "Buffet Service", Price = 120 }
            );

            // --- 2. Seed Couple & Planner Users ---
            var coupleUser = new User { UserName = "couple@test.com", Email = "couple@test.com", Name = "Alex & Jordan", EmailConfirmed = true };
            await userManager.CreateAsync(coupleUser, "Couple@123");
            await userManager.AddToRoleAsync(coupleUser, "Couple");
            var coupleProfile = new Couple { UserId = coupleUser.Id };
            context.Couples.Add(coupleProfile);
            await context.SaveChangesAsync();

            var plannerUser = new User { UserName = "planner@test.com", Email = "planner@test.com", Name = "Jane Doe", EmailConfirmed = true };
            await userManager.CreateAsync(plannerUser, "Planner@123");
            await userManager.AddToRoleAsync(plannerUser, "WeddingPlanner");
            var plannerProfile = new WeddingPlanner { UserId = plannerUser.Id, BusinessName = "Dream Day Planners Inc." };
            context.WeddingPlanners.Add(plannerProfile);
            await context.SaveChangesAsync();

            // --- 3. Seed the Wedding and link it to Couple & Planner ---
            var wedding = new Wedding
            {
                Title = "Alex & Jordan's Summer Wedding",
                WeddingDate = DateTime.Today.AddDays(120),
                Budget = 25000,
                CoupleId = coupleProfile.Id,
                WeddingPlannerId = plannerProfile.Id
```

```
        };
        context.Weddings.Add(wedding);
        await context.SaveChangesAsync();

        // --- 4. Seed data that depends on the Wedding ---
        // Budget
        var budget = new Budget { WeddingId = wedding.Id };
        context.Budgets.Add(budget);
        await context.SaveChangesAsync();

        var budgetCat1 = new BudgetCategory { BudgetId = budget.Id, Name = "Venue", AllocatedAmount = 10000 };
        var budgetCat2 = new BudgetCategory { BudgetId = budget.Id, Name = "Catering", AllocatedAmount = 8000 };
        var budgetCat3 = new BudgetCategory { BudgetId = budget.Id, Name = "Photography", AllocatedAmount = 4000 };
        await context.BudgetCategories.AddRangeAsync(budgetCat1, budgetCat2, budgetCat3);
        await context.SaveChangesAsync();

        await context.Expenses.AddRangeAsync(
            new Expense { BudgetCategoryId = budgetCat1.Id, Description = "Venue Deposit", Amount = 5000, Date = DateTime.Today.AddDays(-10) },
            new Expense { BudgetCategoryId = budgetCat3.Id, Description = "Photographer Booking Fee", Amount = 2000, Date = DateTime.Today.AddDays(-9) }
        );

        // Timeline
        var timeline = new Timeline { WeddingId = wedding.Id };
        context.Timelines.Add(timeline);
        await context.SaveChangesAsync();

        await context.TimelineEvents.AddRangeAsync(
            new TimelineEvent { TimelineId = timeline.Id, Title = "Guests Arrive", StartTime = wedding.WeddingDate.AddHours(14) },
            new TimelineEvent { TimelineId = timeline.Id, Title = "Ceremony Begins", StartTime = wedding.WeddingDate.AddHours(14).AddMinutes(30) },
            new TimelineEvent { TimelineId = timeline.Id, Title = "Cocktail Hour", StartTime = wedding.WeddingDate.AddHours(15).AddMinutes(30) },
            new TimelineEvent { TimelineId = timeline.Id, Title = "Dinner Service", StartTime = wedding.WeddingDate.AddHours(17) }
        );

        // Seating Tables
        await context.Tables.AddRangeAsync(
            new Table { WeddingId = wedding.Id, Name = "Head Table", Capacity = 10 },
            new Table { WeddingId = wedding.Id, Name = "Family Table 1", Capacity = 8 },
            new Table { WeddingId = wedding.Id, Name = "Friends Table 1", Capacity = 8 }
        );

        // Guests
        await context.Guests.AddRangeAsync(
            new Guest { WeddingId = wedding.Id, Name = "Michael Scott", RsvpStatus = GuestStatus.ATTENDING, MealPreference = "Chicken" },
            new Guest { WeddingId = wedding.Id, Name = "Pam Beesly", RsvpStatus = GuestStatus.ATTENDING, MealPreference = "Vegetarian" },
            new Guest { WeddingId = wedding.Id, Name = "Jim Halpert", RsvpStatus = GuestStatus.ATTENDING, MealPreference = "Beef" },
            new Guest { WeddingId = wedding.Id, Name = "Dwight Schrute", RsvpStatus = GuestStatus.PENDING, MealPreference = "Beets" }
        );

        // --- 5. Seed Inter-user data ---
        // Messages
        await context.Messages.AddRangeAsync(
            new Message { SenderId = coupleUser.Id, ReceiverId = plannerUser.Id, Content = "Hi Jane, have you had a chance to look at the catering
options?" },
            new Message { SenderId = plannerUser.Id, ReceiverId = coupleUser.Id, Content = "Hi Alex & Jordan! Yes, I've sent over the top three proposals.
Let me know your thoughts!" }
        );
```

```
            // Reviews
            context.Reviews.Add(new Review { VendorId = vendor1.Id, CoupleId = coupleProfile.Id, Rating = 5, Comment = "So professional and easy to work with
during our engagement shoot!" });

            // --- 6. Seed Activity Logs ---
            context.ActivityLogs.Add(new ActivityLog { ActionDescription = $"User '{coupleUser.Email}' registered.", UserId = coupleUser.Id });
            context.ActivityLogs.Add(new ActivityLog { ActionDescription = $"User '{plannerUser.Email}' registered.", UserId = plannerUser.Id });
            context.ActivityLogs.Add(new ActivityLog { ActionDescription = $"Wedding '{wedding.Title}' created.", UserId = coupleUser.Id });
            context.ActivityLogs.Add(new ActivityLog { ActionDescription = $"Vendor '{vendor1.Name}' created by system.", UserId = null });

            // --- 7. Final Save ---
            await context.SaveChangesAsync();
        }
    }
}
```

ActivityLog.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class ActivityLog {
        [Key] public int Id { get; set; }
        public string ActionDescription { get; set; }
        public DateTime Timestamp { get; set; } = DateTime.UtcNow;
        public int? UserId { get; set; }
        public virtual User? User { get; set; }
    }
}
```

Admin.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public class Admin {
        [Key] public int Id { get; set; }
        public string Role { get; set; } = "SystemAdmin";
        public int UserId { get; set; }
        [ForeignKey("UserId")] public virtual User User { get; set; }
    }
}
```

Budget.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class Budget {
        [Key] public int Id { get; set; }
        public int WeddingId { get; set; }
        public virtual Wedding Wedding { get; set; }
        public virtual ICollection<BudgetCategory> BudgetCategories { get; set; } = new List<BudgetCategory>();
    }
```

```
}
```

BudgetCategory.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public class BudgetCategory {
        [Key] public int Id { get; set; }
        [Required] public string Name { get; set; }
        [Column(TypeName = "decimal(18,2)")] public decimal AllocatedAmount { get; set; }
        public int BudgetId { get; set; }
        public virtual Budget Budget { get; set; }
        public virtual ICollection<Expense> Expenses { get; set; } = new List<Expense>();
    }
}
```

Couple.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public class Couple {
        [Key] public int Id { get; set; }
        public bool IsPremium { get; set; }
        public int UserId { get; set; }
        [ForeignKey("UserId")] public virtual User User { get; set; }
        public virtual ICollection<Wedding> Weddings { get; set; } = new List<Wedding>();
        public virtual ICollection<Review> Reviews { get; set; } = new List<Review>();
    }
}
```

Expense.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public class Expense {
        [Key] public int Id { get; set; }
        [Required] public string Description { get; set; }
        [Column(TypeName = "decimal(18,2)")] public decimal Amount { get; set; }
        public DateTime Date { get; set; }
        public int BudgetCategoryId { get; set; }
        public virtual BudgetCategory BudgetCategory { get; set; }
    }
}
```

Guest.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
```

```
    public enum GuestStatus { INVITED, ATTENDING, DECLINED, PENDING }
    public class Guest {
        [Key] public int Id { get; set; }
        [Required] public string Name { get; set; }
        public string? Email { get; set; }
        public GuestStatus RsvpStatus { get; set; } = GuestStatus.PENDING;
        public string? MealPreference { get; set; }
        public int WeddingId { get; set; }
        public virtual Wedding Wedding { get; set; }
        public int? TableId { get; set; }
        public virtual Table? Table { get; set; }
    }
}
```

Message.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class Message {
        [Key] public int Id { get; set; }
        [Required] public string Content { get; set; }
        public DateTime SentAt { get; set; } = DateTime.UtcNow;
        public bool IsRead { get; set; } = false;
        public int SenderId { get; set; }
        public int ReceiverId { get; set; }
        public virtual User Sender { get; set; }
        public virtual User Receiver { get; set; }
    }
}
```

Notification.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public enum NotificationType { TASK_DUE, BUDGET_ALERT, RSVP_UPDATE, VENDOR_MESSAGE, SYSTEM_ALERT }
    public class Notification {
        [Key] public int Id { get; set; }
        [Required] public string Title { get; set; }
        [Required] public string Content { get; set; }
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
        public bool IsRead { get; set; } = false;
        public NotificationType Type { get; set; }
        public int UserId { get; set; }
        public virtual User User { get; set; }
    }
}
```

Report.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public enum ReportType { BUDGET, GUEST, VENDOR, PROGRESS, SYSTEM }
    public class Report {
```

```
        [Key] public int Id { get; set; }
        [Required] public string Title { get; set; }
        public DateTime GeneratedAt { get; set; } = DateTime.UtcNow;
        public ReportType Type { get; set; }
        public string? ContentUrl { get; set; }
        public int GeneratedById { get; set; }
        public virtual User GeneratedBy { get; set; }
    }
}
```

Review.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class Review {
        [Key] public int Id { get; set; }
        [Range(1, 5)] public int Rating { get; set; }
        public string? Comment { get; set; }
        public DateTime DatePosted { get; set; } = DateTime.UtcNow;
        public int VendorId { get; set; }
        public int CoupleId { get; set; }
        public virtual Vendor Vendor { get; set; }
        public virtual Couple Couple { get; set; }
    }
}
```

Table.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class Table {
        [Key] public int Id { get; set; }
        [Required] public string Name { get; set; }
        public int Capacity { get; set; }
        public int WeddingId { get; set; }
        public virtual Wedding Wedding { get; set; }
        public virtual ICollection<Guest> SeatedGuests { get; set; } = new List<Guest>();
    }
}
```

Task.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public enum TaskCategory { VENUE, CATERING, DECORATION, PHOTOGRAPHY, ATTIRE, INVITATION, OTHER }
    public class Task {
        [Key] public int Id { get; set; }
        [Required] public string Description { get; set; }
        public DateTime DueDate { get; set; }
        public bool IsCompleted { get; set; }
        public TaskCategory Category { get; set; }
        public int WeddingId { get; set; }
        public virtual Wedding Wedding { get; set; }
```

```
    }
}
```

Timeline.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class Timeline {
        [Key] public int Id { get; set; }
        public int WeddingId { get; set; }
        public virtual Wedding Wedding { get; set; }
        public virtual ICollection<TimelineEvent> TimelineEvents { get; set; } = new List<TimelineEvent>();
    }
}
```

TimelineEvent.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class TimelineEvent {
        [Key] public int Id { get; set; }
        [Required] public string Title { get; set; }
        public string? Description { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public int TimelineId { get; set; }
        public virtual Timeline Timeline { get; set; }
    }
}
```

User.cs

```
using Microsoft.AspNetCore.Identity;
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class User : IdentityUser<int> {
        [Required] public string Name { get; set; }
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
    }
}
```

Vendor.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public class Vendor {
        [Key] public int Id { get; set; }
        [Required] public string Name { get; set; }
```

```
        [Required] public string Category { get; set; }
        public string? ContactInfo { get; set; }
        public string? Description { get; set; }
        public virtual ICollection<VendorService> Services { get; set; } = new List<VendorService>();
        public virtual ICollection<Review> Reviews { get; set; } = new List<Review>();
        public virtual ICollection<VendorBooking> Bookings { get; set; } = new List<VendorBooking>();
    }
}
```

VendorBooking.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public enum BookingStatus { PENDING, CONFIRMED, CANCELLED, COMPLETED }
    public class VendorBooking {
        [Key] public int Id { get; set; }
        public DateTime BookingDate { get; set; }
        [Column(TypeName = "decimal(18,2)")] public decimal Cost { get; set; }
        public BookingStatus Status { get; set; }
        public int WeddingId { get; set; }
        public int VendorId { get; set; }
        public virtual Wedding Wedding { get; set; }
        public virtual Vendor Vendor { get; set; }
    }
}
```

VendorService.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public class VendorService {
        [Key] public int Id { get; set; }
        [Required] public string Name { get; set; }
        public string? Description { get; set; }
        [Column(TypeName = "decimal(18,2)")] public decimal Price { get; set; }
        public int VendorId { get; set; }
        public virtual Vendor Vendor { get; set; }
    }
}
```

Wedding.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.Models {
    public enum WeddingStatus { PLANNING, UPCOMING, COMPLETED, CANCELLED }
    public class Wedding {
        [Key] public int Id { get; set; }
        [Required] public string Title { get; set; }
        public DateTime WeddingDate { get; set; }
        public string? Theme { get; set; }
        public string? Location { get; set; }
```

```
        public decimal Budget { get; set; }
        public WeddingStatus Status { get; set; }
        public int CoupleId { get; set; }
        public int? WeddingPlannerId { get; set; }
        public virtual Couple Couple { get; set; }
        public virtual WeddingPlanner? WeddingPlanner { get; set; }
        public virtual ICollection<Task> Tasks { get; set; } = new List<Task>();
        public virtual ICollection<Guest> Guests { get; set; } = new List<Guest>();
        public virtual ICollection<VendorBooking> VendorBookings { get; set; } = new List<VendorBooking>();
    }
}
```

WeddingPlanner.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace DreamDay.Models {
    public class WeddingPlanner {
        [Key] public int Id { get; set; }
        public string? Experience { get; set; }
        public string? Specialization { get; set; }
        public string? BusinessName { get; set; }
        public int UserId { get; set; }
        [ForeignKey("UserId")] public virtual User User { get; set; }
        public virtual ICollection<Wedding> ManagedWeddings { get; set; } = new List<Wedding>();
    }
}
```

DashboardService.cs

```
using DreamDay.Data;
using DreamDay.Models;
using DreamDay.ViewModels;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Services
{
    public class DashboardService
    {
        private readonly ApplicationDbContext _context;

        public DashboardService(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<DashboardViewModel> GetDashboardViewModelAsync(int weddingId)
        {
            var wedding = await _context.Weddings
                .Include(w => w.Tasks)
```

```
                .FirstOrDefaultAsync(w => w.Id == weddingId);

            if (wedding == null)
            {
                return null;
            }

            var totalExpenses = await _context.Expenses
                .Where(e => e.BudgetCategory.Budget.WeddingId == weddingId)
                .SumAsync(e => e.Amount);


            var viewModel = new DashboardViewModel
            {
                CurrentWedding = wedding,
                DaysRemaining = (wedding.WeddingDate - DateTime.Today).Days,
                UpcomingTasks = wedding.Tasks.Where(t => !t.IsCompleted && t.DueDate >= DateTime.Today).OrderBy(t => t.DueDate).Take(5).ToList(),
                BudgetTotal = wedding.Budget,
                BudgetSpent = totalExpenses,
                TotalTasks = wedding.Tasks.Count(),
                CompletedTasks = wedding.Tasks.Count(t => t.IsCompleted)
            };

            if (viewModel.BudgetTotal > 0)
                viewModel.BudgetProgressPercentage = (viewModel.BudgetSpent / viewModel.BudgetTotal) * 100;

            if (viewModel.TotalTasks > 0)
                viewModel.TaskCompletionPercentage = ((double)viewModel.CompletedTasks / viewModel.TotalTasks) * 100;

                return viewModel;
            }
        }
    }
```

DashboardViewModel.cs

```
using DreamDay.Models;
namespace DreamDay.ViewModels {
    public class DashboardViewModel {
        public Wedding CurrentWedding { get; set; }
        public int DaysRemaining { get; set; }
        public List<Models.Task> UpcomingTasks { get; set; }
        public decimal BudgetTotal { get; set; }
        public decimal BudgetSpent { get; set; }
        public decimal BudgetProgressPercentage { get; set; }
        public int TotalTasks { get; set; }
        public int CompletedTasks { get; set; }
        public double TaskCompletionPercentage { get; set; }
    }
}
```

LoginViewModel.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.ViewModels {
    public class LoginViewModel {
        [Required] [EmailAddress] public string Email { get; set; }
        [Required] [DataType(DataType.Password)] public string Password { get; set; }
        [Display(Name = "Remember me?")] public bool RememberMe { get; set; }
    }
}
```

RegisterViewModel.cs

```
using System.ComponentModel.DataAnnotations;
namespace DreamDay.ViewModels {
    public class RegisterViewModel {
        [Required] public string Name { get; set; }
        [Required] [EmailAddress] public string Email { get; set; }
        [Required] [DataType(DataType.Password)] public string Password { get; set; }
        [DataType(DataType.Password)] [Display(Name = "Confirm Password")] [Compare("Password", ErrorMessage = "The password and confirmation password do not
match.")] public string ConfirmPassword { get; set; }
        [Required] public string Role { get; set; }
    }
}
```

AccountController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using DreamDay.ViewModels;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;

namespace DreamDay.Controllers
{
    public class AccountController : Controller
    {
        private readonly UserManager<User> _userManager;
        private readonly SignInManager<User> _signInManager;
        private readonly ApplicationDbContext _context;

        public AccountController(UserManager<User> userManager, SignInManager<User> signInManager, ApplicationDbContext context)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _context = context;
        }

        public IActionResult Login() => View(new LoginViewModel());

        [HttpPost]
        public async Task<IActionResult> Login(LoginViewModel model)
        {
            if (!ModelState.IsValid) return View(model);
```

```
        var user = await _userManager.FindByEmailAsync(model.Email);
        if (user != null)
        {
            var passwordCheck = await _userManager.CheckPasswordAsync(user, model.Password);
            if (passwordCheck)
            {
                var result = await _signInManager.PasswordSignInAsync(user, model.Password, model.RememberMe, false);
                if (result.Succeeded)
                {
                    if (await _userManager.IsInRoleAsync(user, "Admin")) return RedirectToAction("Index", "Admin");
                    if (await _userManager.IsInRoleAsync(user, "WeddingPlanner")) return RedirectToAction("Index", "Planner");
                    return RedirectToAction("Index", "Dashboard");
                }
            }
        }
        TempData["Error"] = "Wrong credentials. Please, try again!";
        return View(model);
    }

    public IActionResult Register() => View(new RegisterViewModel());

    [HttpPost]
    public async Task<IActionResult> Register(RegisterViewModel model)
    {
        if (!ModelState.IsValid) return View(model);

        var user = await _userManager.FindByEmailAsync(model.Email);
        if (user != null)
        {
            TempData["Error"] = "This email address is already in use";
            return View(model);
        }

        var newUser = new User() { Name = model.Name, Email = model.Email, UserName = model.Email };
        var result = await _userManager.CreateAsync(newUser, model.Password);

        if (result.Succeeded)
        {
            await _userManager.AddToRoleAsync(newUser, model.Role);

            if (model.Role == "Couple")
            {
                _context.Couples.Add(new Couple { UserId = newUser.Id });
            }
            else if (model.Role == "WeddingPlanner")
            {
                _context.WeddingPlanners.Add(new WeddingPlanner { UserId = newUser.Id });
            }
            await _context.SaveChangesAsync();

            return RedirectToAction("Login");
        }

        foreach (var error in result.Errors)
        {
```

```
                ModelState.AddModelError(string.Empty, error.Description);
            }

            return View(model);
        }

        [HttpPost]
        public async Task<IActionResult> Logout()
        {
            await _signInManager.SignOutAsync();
            return RedirectToAction("Index", "Home");
        }
    }
}
```

AdminController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Admin")]
    public class AdminController : Controller
    {
        private readonly ApplicationDbContext _context;

        public AdminController(ApplicationDbContext context)
        {
            _context = context;
        }

        public IActionResult Index() => View();

        public async Task<IActionResult> Users() => View(await _context.Users.ToListAsync());

        public async Task<IActionResult> Vendors() => View(await _context.Vendors.ToListAsync());

        public IActionResult CreateVendor() => View();

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> CreateVendor([Bind("Name,Category,ContactInfo,Description")] Vendor vendor)
        {
            if (ModelState.IsValid)
            {
                _context.Add(vendor);
                await _context.SaveChangesAsync();
```

```
                return RedirectToAction(nameof(Vendors));
            }
            return View(vendor);
        }

        public async Task<IActionResult> EditVendor(int? id)
        {
            if (id == null) return NotFound();
            var vendor = await _context.Vendors.FindAsync(id);
            if (vendor == null) return NotFound();
            return View(vendor);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> EditVendor(int id, [Bind("Id,Name,Category,ContactInfo,Description")] Vendor vendor)
        {
            if (id != vendor.Id) return NotFound();
            if (ModelState.IsValid)
            {
                _context.Update(vendor);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Vendors));
            }
            return View(vendor);
        }

        public async Task<IActionResult> DeleteVendor(int? id)
        {
            if (id == null) return NotFound();
            var vendor = await _context.Vendors.FirstOrDefaultAsync(m => m.Id == id);
            if (vendor == null) return NotFound();
            return View(vendor);
        }

        [HttpPost, ActionName("DeleteVendor")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteVendorConfirmed(int id)
        {
            var vendor = await _context.Vendors.FindAsync(id);
            _context.Vendors.Remove(vendor);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Vendors));
        }

        public async Task<IActionResult> ActivityLog()
        {
            var logs = await _context.ActivityLogs
                .Include(l => l.User)
                .OrderByDescending(l => l.Timestamp)
                .Take(100)
                .ToListAsync();
            return View(logs);
        }
    }
}
```

BaseController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    public abstract class BaseController : Controller
    {
        protected readonly ApplicationDbContext _context;
        protected readonly UserManager<User> _userManager;

        protected BaseController(ApplicationDbContext context, UserManager<User> userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        protected async Task<Wedding> GetCurrentUserWeddingAsync()
        {
            var user = await _userManager.GetUserAsync(User);
            if (user == null) return null;

            var couple = await _context.Couples.FirstOrDefaultAsync(c => c.UserId == user.Id);
            if (couple == null) return null;

            return await _context.Weddings
                .Include(w => w.Couple)
                .FirstOrDefaultAsync(w => w.CoupleId == couple.Id);
        }
    }
}
```

BudgetController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple")]
    public class BudgetController : BaseController
    {
```

```
        public BudgetController(ApplicationDbContext context, UserManager<User> userManager)
            : base(context, userManager)
        {
        }

        public async Task<IActionResult> Index()
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return RedirectToAction("Create", "Wedding");

            var budget = await _context.Budgets
                .Include(b => b.BudgetCategories)
                    .ThenInclude(bc => bc.Expenses)
                .FirstOrDefaultAsync(b => b.WeddingId == wedding.Id);

            if (budget == null)
            {
                budget = new Budget { WeddingId = wedding.Id };
                _context.Budgets.Add(budget);
                await _context.SaveChangesAsync();
            }

            ViewBag.TotalBudget = wedding.Budget;
            return View(budget);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> AddCategory(string name, decimal allocatedAmount)
        {
            var wedding = await GetCurrentUserWeddingAsync();
            var budget = await _context.Budgets.FirstOrDefaultAsync(b => b.WeddingId == wedding.Id);

            var category = new BudgetCategory
            {
                BudgetId = budget.Id,
                Name = name,
                AllocatedAmount = allocatedAmount
            };
            _context.BudgetCategories.Add(category);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index");
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> AddExpense(int budgetCategoryId, string description, decimal amount)
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return Forbid();

            var category = await _context.BudgetCategories
                .FirstOrDefaultAsync(c => c.Id == budgetCategoryId && c.Budget.WeddingId == wedding.Id);

            if (category == null) return Forbid();
```

```
            var expense = new Expense
            {
                BudgetCategoryId = budgetCategoryId,
                Description = description,
                Amount = amount,
                Date = DateTime.UtcNow
            };
            _context.Expenses.Add(expense);
            await _context.SaveChangesAsync();

            await CheckForBudgetOverrunAndNotify(category, wedding.Couple.UserId);

            return RedirectToAction("Index");
        }

        private async Task CheckForBudgetOverrunAndNotify(BudgetCategory category, int coupleUserId)
        {
            var totalSpentInCategory = await _context.Expenses
                .Where(e => e.BudgetCategoryId == category.Id)
                .SumAsync(e => e.Amount);

            if (totalSpentInCategory > category.AllocatedAmount)
            {
                var existingNotification = await _context.Notifications
                    .AnyAsync(n => n.UserId == coupleUserId && n.Content.Contains($"'{category.Name}' category"));

                if (!existingNotification)
                {
                    var notification = new Notification
                    {
                        UserId = coupleUserId,
                        Title = "Budget Alert!",
                        Content = $"You have exceeded your budget for the '{category.Name}' category.",
                        Type = NotificationType.BUDGET_ALERT
                    };
                    _context.Notifications.Add(notification);
                    await _context.SaveChangesAsync();
                }
            }
        }
    }
}
```

ChecklistController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple")]
    public class ChecklistController : BaseController
    {
        public ChecklistController(ApplicationDbContext context, UserManager<User> userManager)
            : base(context, userManager)
        {
        }

        public async Task<IActionResult> Index()
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return RedirectToAction("Create", "Wedding");

            var tasks = await _context.Tasks
                .Where(t => t.WeddingId == wedding.Id)
                .OrderBy(t => t.DueDate)
                .ToListAsync();

            ViewBag.WeddingId = wedding.Id;
            return View(tasks);
        }

        public async Task<IActionResult> Create()
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return NotFound();
            return View(new Models.Task { WeddingId = wedding.Id });
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Description,DueDate,Category,WeddingId")] Models.Task task)
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null || wedding.Id != task.WeddingId) return Forbid();

            if (ModelState.IsValid)
            {
                _context.Add(task);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(task);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> ToggleComplete(int id)
        {
            var task = await _context.Tasks.FindAsync(id);
            if (task == null) return NotFound();

            var wedding = await GetCurrentUserWeddingAsync();
```

```
            if (wedding == null || task.WeddingId != wedding.Id) return Forbid();

            task.IsCompleted = !task.IsCompleted;
            _context.Update(task);
            await _context.SaveChangesAsync();

            return RedirectToAction(nameof(Index));
        }
    }
}
```

DashboardController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using DreamDay.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple, WeddingPlanner")]
    public class DashboardController : BaseController
    {
        private readonly DashboardService _dashboardService;

        public DashboardController(ApplicationDbContext context, UserManager<User> userManager, DashboardService dashboardService)
            : base(context, userManager)
        {
            _dashboardService = dashboardService;
        }

        public async Task<IActionResult> Index(int? weddingId)
        {
            Wedding wedding;
            var user = await _userManager.GetUserAsync(User);
            var plannerProfile = User.IsInRole("WeddingPlanner")
                ? await _context.WeddingPlanners.FirstOrDefaultAsync(p => p.UserId == user.Id)
                : null;

            if (weddingId.HasValue)
            {
                wedding = await _context.Weddings.FindAsync(weddingId.Value);
                if (wedding == null) return NotFound();

                bool isAuthorized = (User.IsInRole("Couple") && (await GetCurrentUserWeddingAsync())?.Id == weddingId) ||
                                    (User.IsInRole("WeddingPlanner") && wedding.WeddingPlannerId == plannerProfile?.Id);

                if (!isAuthorized) return Forbid();
            }
            else
```

```
            {
                if (!User.IsInRole("Couple")) return Forbid();
                wedding = await GetCurrentUserWeddingAsync();
            }

            if (wedding == null)
            {
                return RedirectToAction("Create", "Wedding");
            }

            var viewModel = await _dashboardService.GetDashboardViewModelAsync(wedding.Id);
            return View(viewModel);
        }
    }
}
```

GuestController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple")]
    public class GuestController : BaseController
    {
        public GuestController(ApplicationDbContext context, UserManager<User> userManager)
            : base(context, userManager)
        {
        }

        public async Task<IActionResult> Index()
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return RedirectToAction("Create", "Wedding");

            var guests = await _context.Guests
                .Where(g => g.WeddingId == wedding.Id)
                .ToListAsync();

            return View(guests);
        }

        public IActionResult Create() => View();

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Name,Email,MealPreference")] Guest guest)
        {
```

```
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return NotFound();

            if (ModelState.IsValid)
            {
                guest.WeddingId = wedding.Id;
                guest.RsvpStatus = GuestStatus.INVITED;
                _context.Add(guest);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(guest);
        }

        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null) return NotFound();
            var guest = await _context.Guests.FindAsync(id);
            if (guest == null) return NotFound();

            var wedding = await GetCurrentUserWeddingAsync();
            if (guest.WeddingId != wedding.Id) return Forbid();

            return View(guest);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Email,RsvpStatus,MealPreference,WeddingId")] Guest guest)
        {
            if (id != guest.Id) return NotFound();

            var wedding = await GetCurrentUserWeddingAsync();
            if (guest.WeddingId != wedding.Id) return Forbid();

            if (ModelState.IsValid)
            {
                _context.Update(guest);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(guest);
        }

        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null) return NotFound();
            var guest = await _context.Guests.FirstOrDefaultAsync(m => m.Id == id);
            if (guest == null) return NotFound();

            var wedding = await GetCurrentUserWeddingAsync();
            if (guest.WeddingId != wedding.Id) return Forbid();

            return View(guest);
        }
```

```
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var guest = await _context.Guests.FindAsync(id);
            var wedding = await GetCurrentUserWeddingAsync();
            if (guest.WeddingId != wedding.Id) return Forbid();

            _context.Guests.Remove(guest);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
    }
}
```

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
namespace DreamDay.Controllers {
    public class HomeController : Controller {
        public IActionResult Index() {
            if (User.Identity.IsAuthenticated) {
                if (User.IsInRole("Admin")) return RedirectToAction("Index", "Admin");
                if (User.IsInRole("WeddingPlanner")) return RedirectToAction("Index", "Planner");
                return RedirectToAction("Index", "Dashboard");
            }
            return View();
        }
    }
}
```

MessageController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize]
    public class MessageController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly UserManager<User> _userManager;

        public MessageController(ApplicationDbContext context, UserManager<User> userManager)
        {
            _context = context;
            _userManager = userManager;
```

```
        }

        public async Task<IActionResult> Index()
        {
            var currentUser = await _userManager.GetUserAsync(User);
            var wedding = await _context.Weddings
                .FirstOrDefaultAsync(w => w.Couple.UserId == currentUser.Id || w.WeddingPlanner.UserId == currentUser.Id);

            if (wedding == null) return View(null);

            var couple = await _context.Couples.Include(c => c.User).FirstOrDefaultAsync(c => c.Id == wedding.CoupleId);
            var planner = await _context.WeddingPlanners.Include(p => p.User).FirstOrDefaultAsync(p => p.Id == wedding.WeddingPlannerId);

            User otherUser = null;
            if (User.IsInRole("Couple")) otherUser = planner?.User;
            else if (User.IsInRole("WeddingPlanner")) otherUser = couple?.User;

            if (otherUser == null) return View(null);

            var messages = await _context.Messages
                .Where(m => (m.SenderId == currentUser.Id && m.ReceiverId == otherUser.Id) ||
                            (m.SenderId == otherUser.Id && m.ReceiverId == currentUser.Id))
                .OrderBy(m => m.SentAt)
                .ToListAsync();

            ViewBag.OtherUser = otherUser;
            return View(messages);
        }

        [HttpPost]
        public async Task<IActionResult> Send(string content, int receiverId)
        {
            var sender = await _userManager.GetUserAsync(User);
            var message = new Message { Content = content, SenderId = sender.Id, ReceiverId = receiverId };
            _context.Messages.Add(message);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index");
        }
    }
}
```

PlannerController.cs

```
using DreamDay.Data;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
```

```
{
    [Authorize(Roles = "WeddingPlanner")]
    public class PlannerController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly UserManager<Models.User> _userManager;

        public PlannerController(ApplicationDbContext context, UserManager<Models.User> userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        public async Task<IActionResult> Index()
        {
            var user = await _userManager.GetUserAsync(User);
            var planner = await _context.WeddingPlanners.FirstOrDefaultAsync(p => p.UserId == user.Id);
            if (planner == null) return Unauthorized();

            var weddings = await _context.Weddings
                .Where(w => w.WeddingPlannerId == planner.Id)
                .Include(w => w.Couple)
                    .ThenInclude(c => c.User)
                .ToListAsync();

            return View(weddings);
        }
    }
}
```

ReportController.cs

```
using DreamDay.Data;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "WeddingPlanner, Admin")]
    public class ReportController : Controller
    {
        private readonly ApplicationDbContext _context;

        public ReportController(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> BudgetReport(int weddingId)
        {
            var wedding = await _context.Weddings
```

```
                .Include(w => w.Couple.User)
                .FirstOrDefaultAsync(w => w.Id == weddingId);

            var budget = await _context.Budgets
                .Include(b => b.BudgetCategories)
                .ThenInclude(bc => bc.Expenses)
                .FirstOrDefaultAsync(b => b.WeddingId == weddingId);

            ViewBag.Wedding = wedding;
            return View(budget);
        }
    }
}
```

SeatingController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple")]
    public class SeatingController : BaseController
    {
        public SeatingController(ApplicationDbContext context, UserManager<User> userManager)
            : base(context, userManager)
        {
        }

        public async Task<IActionResult> Index()
        {
            var wedding = await GetCurrentUserWeddingAsync();
            if (wedding == null) return RedirectToAction("Create", "Wedding");

            ViewBag.Tables = await _context.Tables
                .Where(t => t.WeddingId == wedding.Id)
                .Include(t => t.SeatedGuests)
                .ToListAsync();

            ViewBag.UnseatedGuests = new SelectList(
                await _context.Guests.Where(g => g.WeddingId == wedding.Id && g.RsvpStatus == GuestStatus.ATTENDING && g.TableId == null).ToListAsync(),
                "Id", "Name");

            ViewBag.TableList = new SelectList(
                await _context.Tables.Where(t => t.WeddingId == wedding.Id).ToListAsync(),
                "Id", "Name");
```

```
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> AddTable(string name, int capacity)
        {
            var wedding = await GetCurrentUserWeddingAsync();
            var table = new Table { Name = name, Capacity = capacity, WeddingId = wedding.Id };
            _context.Tables.Add(table);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index");
        }

        [HttpPost]
        public async Task<IActionResult> AssignGuest(int guestId, int tableId)
        {
            var guest = await _context.Guests.FindAsync(guestId);
            var table = await _context.Tables.Include(t => t.SeatedGuests).FirstOrDefaultAsync(t => t.Id == tableId);

            if (guest != null && table != null && table.SeatedGuests.Count < table.Capacity)
            {
                guest.TableId = tableId;
                _context.Update(guest);
                await _context.SaveChangesAsync();
            }
            return RedirectToAction("Index");
        }
    }
}
```

TimelineController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple")]
    public class TimelineController : BaseController
    {
        public TimelineController(ApplicationDbContext context, UserManager<User> userManager)
            : base(context, userManager)
        {
        }

        public async Task<IActionResult> Index()
        {
            var wedding = await GetCurrentUserWeddingAsync();
```

```
            if (wedding == null) return RedirectToAction("Create", "Wedding");

            var timeline = await _context.Timelines
                .Include(t => t.TimelineEvents.OrderBy(te => te.StartTime))
                .FirstOrDefaultAsync(t => t.WeddingId == wedding.Id);

            if (timeline == null)
            {
                timeline = new Timeline { WeddingId = wedding.Id };
                _context.Timelines.Add(timeline);
                await _context.SaveChangesAsync();
            }

            return View(timeline);
        }

        [HttpPost]
        public async Task<IActionResult> AddEvent(string title, DateTime startTime)
        {
            var wedding = await GetCurrentUserWeddingAsync();
            var timeline = await _context.Timelines.FirstOrDefaultAsync(t => t.WeddingId == wedding.Id);

            var newEvent = new TimelineEvent
            {
                TimelineId = timeline.Id,
                Title = title,
                StartTime = startTime,
                EndTime = startTime.AddHours(1)
            };
            _context.TimelineEvents.Add(newEvent);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index");
        }
    }
}
```

VendorController.cs

```
using DreamDay.Data;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize]
    public class VendorController : Controller
    {
        private readonly ApplicationDbContext _context;

        public VendorController(ApplicationDbContext context)
        {
            _context = context;
        }
```

```
        public async Task<IActionResult> Index() => View(await _context.Vendors.ToListAsync());

        public async Task<IActionResult> Details(int? id)
        {
            if (id == null) return NotFound();

            var vendor = await _context.Vendors
                .Include(v => v.Services)
                .Include(v => v.Reviews)
                .ThenInclude(r => r.Couple.User)
                .FirstOrDefaultAsync(m => m.Id == id);

            if (vendor == null) return NotFound();

            return View(vendor);
        }
    }
}
```

WeddingController.cs

```
using DreamDay.Data;
using DreamDay.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;

namespace DreamDay.Controllers
{
    [Authorize(Roles = "Couple")]
    public class WeddingController : BaseController
    {
        public WeddingController(ApplicationDbContext context, UserManager<User> userManager)
            : base(context, userManager)
        {
        }

        public IActionResult Create() => View();

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Wedding wedding)
        {
            if (ModelState.IsValid)
            {
                var couple = await _context.Couples.FirstOrDefaultAsync(c => c.UserId == int.Parse(_userManager.GetUserId(User)));
                if (couple == null) return Unauthorized();

                wedding.CoupleId = couple.Id;
                _context.Add(wedding);
                await _context.SaveChangesAsync();
```

```
                return RedirectToAction("Index", "Dashboard");
            }
            return View(wedding);
        }
    }
}
```

ViewImports.cshtml

```
@using DreamDay
@using DreamDay.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - DreamDay</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">DreamDay</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item"><a class="nav-link text-dark" asp-controller="Home" asp-action="Index">Home</a></li>
                        @if (User.Identity.IsAuthenticated)
                        {
                            if (User.IsInRole("Couple"))
                            {
                                <li class="nav-item"><a class="nav-link text-dark" asp-controller="Dashboard" asp-action="Index">Dashboard</a></li>
                                <li class="nav-item"><a class="nav-link text-dark" asp-controller="Checklist" asp-action="Index">Checklist</a></li>
```

```
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Guest" asp-action="Index">Guests</a></li>
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Seating" asp-action="Index">Seating</a></li>
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Budget" asp-action="Index">Budget</a></li>
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Timeline" asp-action="Index">Timeline</a></li>
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Vendor" asp-action="Index">Vendors</a></li>
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Message" asp-action="Index">Messages</a></li>
                        }
                        if (User.IsInRole("WeddingPlanner"))
                        {
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Planner" asp-action="Index">Planner Dashboard</a></li>
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Message" asp-action="Index">Messages</a></li>
                        }
                        if (User.IsInRole("Admin"))
                        {
                            <li class="nav-item"><a class="nav-link text-dark" asp-controller="Admin" asp-action="Index">Admin Panel</a></li>
                        }
                    }
                </ul>
                <ul class="navbar-nav">
                    @if (!User.Identity.IsAuthenticated)
                    {
                        <li class="nav-item"><a class="btn btn-outline-primary me-2" asp-controller="Account" asp-action="Login">Login</a></li>
                        <li class="nav-item"><a class="btn btn-primary" asp-controller="Account" asp-action="Register">Register</a></li>
                    }
                    else
                    {
                        <li class="nav-item"><span class="nav-link text-dark">Hello, @User.Identity.Name</span></li>
                        <li class="nav-item">
                            <form asp-controller="Account" asp-action="Logout" method="post">
                                <button type="submit" class="btn btn-link nav-link text-dark">Logout</button>
                            </form>
                        </li>
                    }
                </ul>
            </div>
        </div>
    </nav>
</header>
<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>
<footer class="border-top footer text-muted">
    <div class="container">© 2024 - DreamDay</div>
</footer>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

ValidationScriptsPartial.cshtml

```
<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

Login.cshtml

```
@model DreamDay.ViewModels.LoginViewModel
@{ ViewData["Title"] = "Login"; }
<div class="row"><div class="col-md-6 offset-md-3"><h2>Login to your account</h2>
@if (TempData["Error"] != null){<div class="alert alert-danger">@TempData["Error"]</div>}
<form asp-action="Login"><div asp-validation-summary="ModelOnly" class="text-danger"></div>
<div class="form-group mb-3"><label asp-for="Email" class="control-label"></label><input asp-for="Email" class="form-control" /><span asp-validation-for="Email"
class="text-danger"></span></div>
<div class="form-group mb-3"><label asp-for="Password" class="control-label"></label><input asp-for="Password" class="form-control" /><span asp-validation-
for="Password" class="text-danger"></span></div>
<div class="form-group"><input type="submit" value="Login" class="btn btn-primary" /></div></form></div></div>
```

Register.cshtml

```
@model DreamDay.ViewModels.RegisterViewModel
@{ ViewData["Title"] = "Register"; }
<div class="row"><div class="col-md-6 offset-md-3"><h2>Create a new account</h2>
<form asp-action="Register"><div asp-validation-summary="ModelOnly" class="text-danger"></div>
<div class="form-group mb-3"><label asp-for="Name" class="control-label"></label><input asp-for="Name" class="form-control" /><span asp-validation-for="Name"
class="text-danger"></span></div>
<div class="form-group mb-3"><label asp-for="Email" class="control-label"></label><input asp-for="Email" class="form-control" /><span asp-validation-for="Email"
class="text-danger"></span></div>
<div class="form-group mb-3"><label asp-for="Password" class="control-label"></label><input asp-for="Password" class="form-control" /><span asp-validation-
for="Password" class="text-danger"></span></div>
<div class="form-group mb-3"><label asp-for="ConfirmPassword" class="control-label"></label><input asp-for="ConfirmPassword" class="form-control" /><span asp-
validation-for="ConfirmPassword" class="text-danger"></span></div>
<div class="form-group mb-3"><label asp-for="Role" class="control-label">I am a...</label><select asp-for="Role" class="form-control"><option
value="Couple">Couple</option><option value="WeddingPlanner">Wedding Planner</option></select><span asp-validation-for="Role" class="text-danger"></span></div>
<div class="form-group"><input type="submit" value="Register" class="btn btn-primary" /></div></form></div></div>
```

Index.cshtml

```
@{ ViewData["Title"] = "Admin Dashboard"; }
<h1>Admin Dashboard</h1><p>Welcome, Administrator. Use the links below to manage the system.</p>
<div class="list-group">
    <a asp-action="Users" class="list-group-item list-group-item-action">Manage Users</a>
    <a asp-action="Vendors" class="list-group-item list-group-item-action">Manage Vendors</a>
    <a asp-action="ActivityLog" class="list-group-item list-group-item-action">View System Activity</a>
</div>
```

ActivityLog.cshtml

```
@model IEnumerable<DreamDay.Models.ActivityLog>
@{ ViewData["Title"] = "System Activity Log"; }
<h1>System Activity Log</h1><p>Showing the last 100 actions performed in the system.</p>
<table class="table table-striped"><thead><tr><th>Timestamp</th><th>User</th><th>Action</th></tr></thead>
<tbody>@foreach (var log in Model){<tr><td>@log.Timestamp.ToString("g")</td><td>@(log.User?.Email ??
"System")</td><td>@log.ActionDescription</td></tr>}</tbody></table>
```

CreateVendor.cshtml

```
@model DreamDay.Models.Vendor
@{ ViewData["Title"] = "Create Vendor"; }
<h1>Create Vendor</h1><hr /><div class="row"><div class="col-md-6"><form asp-action="CreateVendor">
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<div class="form-group mb-3"><label asp-for="Name" class="control-label"></label><input asp-for="Name" class="form-control" /></div>
<div class="form-group mb-3"><label asp-for="Category" class="control-label"></label><input asp-for="Category" class="form-control" placeholder="e.g.,
Photographer, Florist" /></div>
<div class="form-group mb-3"><label asp-for="ContactInfo" class="control-label"></label><input asp-for="ContactInfo" class="form-control" /></div>
<div class="form-group mb-3"><label asp-for="Description" class="control-label"></label><textarea asp-for="Description" class="form-control"></textarea></div>
<div class="form-group"><input type="submit" value="Create" class="btn btn-primary" /></div></form></div></div>
```

DeleteVendor.cshtml

```
@model DreamDay.Models.Vendor
@{ ViewData["Title"] = "Delete Vendor"; }
<h3>Are you sure you want to delete @Model.Name?</h3>
<div><hr /><form asp-action="DeleteVendor" asp-route-id="@Model.Id" method="post"><input type="submit" value="Delete" class="btn btn-danger" /> | <a asp-
action="Vendors">Back to List</a></form></div>
```

EditVendor.cshtml

```
@model DreamDay.Models.Vendor
@{ ViewData["Title"] = "Edit Vendor"; }
<h1>Edit Vendor</h1><hr /><div class="row"><div class="col-md-4"><form asp-action="EditVendor">
<input type="hidden" asp-for="Id" />
<div class="form-group mb-2"><label asp-for="Name"></label><input asp-for="Name" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="Category"></label><input asp-for="Category" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="ContactInfo"></label><input asp-for="ContactInfo" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="Description"></label><textarea asp-for="Description" class="form-control"></textarea></div>
<div class="form-group mt-3"><input type="submit" value="Save" class="btn btn-primary" /></div></form></div></div>
```

Users.cshtml

```
@model IEnumerable<DreamDay.Models.User>
@{ ViewData["Title"] = "Manage Users"; }
<h1>Manage Users</h1>
<table class="table"><thead><tr><th>@Html.DisplayNameFor(model => model.Name)</th><th>@Html.DisplayNameFor(model =>
model.Email)</th><th>@Html.DisplayNameFor(model => model.CreatedAt)</th><th></th></tr></thead>
<tbody>@foreach (var item in Model){<tr><td>@Html.DisplayFor(modelItem => item.Name)</td><td>@Html.DisplayFor(modelItem =>
item.Email)</td><td>@Html.DisplayFor(modelItem => item.CreatedAt)</td><td><a href="#" class="btn btn-sm btn-info">Details</a></td></tr>}</tbody></table>
```

Vendors.cshtml

```
@model IEnumerable<DreamDay.Models.Vendor>
@{ ViewData["Title"] = "Manage Vendors"; }
<h1>Manage Vendors</h1><p><a asp-action="CreateVendor" class="btn btn-primary">Create New Vendor</a></p>
<table class="table"><thead><tr><th>@Html.DisplayNameFor(model => model.Name)</th><th>@Html.DisplayNameFor(model => model.Category)</th><th></th></tr></thead>
<tbody>@foreach (var item in Model){<tr><td>@Html.DisplayFor(modelItem => item.Name)</td><td>@Html.DisplayFor(modelItem => item.Category)</td><td><a asp-
action="EditVendor" asp-route-id="@item.Id" class="btn btn-sm btn-secondary">Edit</a> | <a asp-action="DeleteVendor" asp-route-id="@item.Id" class="btn btn-sm
btn-danger">Delete</a></td></tr>}</tbody></table>
```

Index.cshtml

```
@model DreamDay.Models.Budget
@{ ViewData["Title"] = "Budget Tracker"; var totalAllocated = Model.BudgetCategories.Sum(c => c.AllocatedAmount); var totalSpent =
Model.BudgetCategories.SelectMany(c => c.Expenses).Sum(e => e.Amount); }
<h1>Budget Tracker</h1><h4>Total Budget: @ViewBag.TotalBudget.ToString("C")</h4>
<div class="row mb-4"><div class="col"><strong>Total Allocated:</strong> @totalAllocated.ToString("C")</div><div class="col"><strong>Total Spent:</strong> <span
class="text-danger">@totalSpent.ToString("C")</span></div><div class="col"><strong>Remaining Budget:</strong> @(ViewBag.TotalBudget -
totalSpent).ToString("C")</div></div>
<div class="row"><div class="col-md-8"><h3>Spending Categories</h3>
@foreach (var category in Model.BudgetCategories){ var categorySpent = category.Expenses.Sum(e => e.Amount);
<div class="card mb-3"><div class="card-header"><h5>@category.Name <small class="text-muted">(@categorySpent.ToString("C") /
@category.AllocatedAmount.ToString("C"))</small></h5></div>
<div class="card-body"><h6>Expenses</h6><ul>@foreach (var expense in category.Expenses){<li>@expense.Description: @expense.Amount.ToString("C")</li>}</ul><hr/>
<form asp-action="AddExpense" method="post"><div asp-validation-summary="ModelOnly" class="text-danger"></div><input type="hidden" name="budgetCategoryId"
value="@category.Id" />
<div class="row g-3"><div class="col-auto"><input type="text" name="description" class="form-control" placeholder="Expense Description" required></div>
<div class="col-auto"><input type="number" step="0.01" name="amount" class="form-control" placeholder="Amount" required></div>
<div class="col-auto"><button type="submit" class="btn btn-sm btn-outline-primary">Add Expense</button></div></div></form></div></div>}</div>
<div class="col-md-4"><h3>Add New Category</h3><form asp-action="AddCategory" method="post"><div asp-validation-summary="ModelOnly" class="text-danger"></div>
<div class="form-group mb-2"><label>Category Name</label><input type="text" name="name" class="form-control" required /></div>
<div class="form-group mb-2"><label>Allocated Amount</label><input type="number" step="0.01" name="allocatedAmount" class="form-control" required /></div>
<button type="submit" class="btn btn-success">Add Category</button></form></div></div>
```

Index.cshtml

```
@model IEnumerable<DreamDay.Models.Task>
@{ ViewData["Title"] = "Wedding Checklist"; var categories = Model.Select(m => m.Category).Distinct().OrderBy(c => c.ToString()); }
```

```
<h1>@ViewData["Title"]</h1><p><a asp-action="Create" class="btn btn-primary">Add New Task</a></p>
@foreach (var category in categories){<h3>@category.ToString()</h3>
<ul class="list-group mb-4">@foreach (var task in Model.Where(t => t.Category == category)){
<li class="list-group-item d-flex justify-content-between align-items-center @(task.IsCompleted ? "list-group-item-success" : "")">
<div><strong class="@(task.IsCompleted ? "text-decoration-line-through" : "")">@task.Description</strong><br /><small>Due:
@task.DueDate.ToShortDateString()</small></div>
<form asp-action="ToggleComplete" asp-route-id="@task.Id" method="post"><button type="submit" class="btn @(task.IsCompleted ? "btn-sm btn-warning" : "btn-sm
btn-success")">@(task.IsCompleted ? "Mark as Incomplete" : "Mark as Complete")</button></form></li>}</ul>}
```

Create.cshtml

```
@model DreamDay.Models.Task
@{ ViewData["Title"] = "Create Task"; }
<h1>Create a New Task</h1><hr /><div class="row"><div class="col-md-6"><form asp-action="Create">
<div asp-validation-summary="ModelOnly" class="text-danger"></div><input type="hidden" asp-for="WeddingId" />
<div class="form-group mb-3"><label asp-for="Description" class="control-label"></label><input asp-for="Description" class="form-control" /></div>
<div class="form-group mb-3"><label asp-for="DueDate" class="control-label"></label><input asp-for="DueDate" type="date" class="form-control" /></div>
<div class="form-group mb-3"><label asp-for="Category" class="control-label"></label><select asp-for="Category" asp-
items="Html.GetEnumSelectList<DreamDay.Models.TaskCategory>()" class="form-control"></select></div>
<div class="form-group"><input type="submit" value="Create" class="btn btn-primary" /> <a asp-action="Index" class="btn btn-secondary">Back to
List</a></div></form></div></div>
```

Index.cshtml

```
@model DreamDay.ViewModels.DashboardViewModel
@{ ViewData["Title"] = "My Dashboard"; }
<h1>@Model.CurrentWedding.Title</h1><p class="lead">Wedding Date: @Model.CurrentWedding.WeddingDate.ToShortDateString()</p>
<div class="row"><div class="col-md-4"><div class="card text-center"><div class="card-body"><h5 class="card-title">Days Remaining</h5><p class="card-text fs-
1">@Model.DaysRemaining</p></div></div></div>
<div class="col-md-4"><div class="card"><div class="card-body"><h5 class="card-title">Task Progress</h5><p>@Model.CompletedTasks of @Model.TotalTasks tasks
completed.</p><div class="progress"><div class="progress-bar" role="progressbar" style="width: @Model.TaskCompletionPercentage.ToString("0.00")%;" aria-
valuenow="@Model.TaskCompletionPercentage" aria-valuemin="0" aria-valuemax="100">@Model.TaskCompletionPercentage.ToString("0")%</div></div></div></div></div>
<div class="col-md-4"><div class="card"><div class="card-body"><h5 class="card-title">Budget Overview</h5><p>@Model.BudgetSpent.ToString("C") spent of
@Model.BudgetTotal.ToString("C")</p><div class="progress"><div class="progress-bar bg-success" role="progressbar" style="width:
@Model.BudgetProgressPercentage.ToString("0.00")%;" aria-valuenow="@Model.BudgetProgressPercentage" aria-valuemin="0" aria-
valuemax="100"></div></div></div></div></div></div>
<div class="row mt-4"><div class="col-md-12"><h3>Upcoming Tasks</h3><ul class="list-group">
@foreach (var task in Model.UpcomingTasks){<li class="list-group-item">@task.Description - <small>Due: @task.DueDate.ToShortDateString()</small></li>}
@if (!Model.UpcomingTasks.Any()){<li class="list-group-item">No upcoming tasks.</li>}</ul></div></div>
```

Index.cshtml

```
@model IEnumerable<DreamDay.Models.Guest>
@{ ViewData["Title"] = "Guest List"; }
<h1>Guest List</h1><p><a asp-action="Create" class="btn btn-primary">Add Guest</a></p>
<table class="table"><thead><tr><th>Name</th><th>RSVP Status</th><th>Meal Preference</th><th></th></tr></thead>
<tbody>@foreach (var item in Model){<tr><td>@Html.DisplayFor(modelItem => item.Name)</td><td>@Html.DisplayFor(modelItem =>
item.RsvpStatus)</td><td>@Html.DisplayFor(modelItem => item.MealPreference)</td><td><a asp-action="Edit" asp-route-id="@item.Id">Edit</a> | <a asp-
```

```
action="Delete" asp-route-id="@item.Id">Delete</a></td></tr>}</tbody></table>
```

Create.cshtml

```
@model DreamDay.Models.Guest
@{ ViewData["Title"] = "Add Guest"; }
<h1>Add Guest</h1><hr /><div class="row"><div class="col-md-4"><form asp-action="Create">
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<div class="form-group mb-2"><label asp-for="Name" class="control-label"></label><input asp-for="Name" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="Email" class="control-label"></label><input asp-for="Email" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="MealPreference" class="control-label"></label><input asp-for="MealPreference" class="form-control" /></div>
<div class="form-group mt-3"><input type="submit" value="Create" class="btn btn-primary" /></div></form></div></div>
<div><a asp-action="Index">Back to List</a></div>
```

Edit.cshtml

```
@model DreamDay.Models.Guest
@{ ViewData["Title"] = "Edit Guest"; }
<h1>Edit Guest</h1><hr /><div class="row"><div class="col-md-4"><form asp-action="Edit">
<input type="hidden" asp-for="Id" /><input type="hidden" asp-for="WeddingId" />
<div class="form-group mb-2"><label asp-for="Name" class="control-label"></label><input asp-for="Name" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="Email" class="control-label"></label><input asp-for="Email" class="form-control" /></div>
<div class="form-group mb-2"><label asp-for="RsvpStatus" class="control-label"></label><select asp-for="RsvpStatus" asp-
items="Html.GetEnumSelectList<DreamDay.Models.GuestStatus>()" class="form-control"></select></div>
<div class="form-group mb-2"><label asp-for="MealPreference" class="control-label"></label><input asp-for="MealPreference" class="form-control" /></div>
<div class="form-group mt-3"><input type="submit" value="Save" class="btn btn-primary" /></div></form></div></div>
<div><a asp-action="Index">Back to List</a></div>
```

Delete.cshtml

```
@model DreamDay.Models.Guest
@{ ViewData["Title"] = "Delete Guest"; }
<h3>Are you sure you want to delete this guest?</h3>
<div><h4>@Model.Name</h4><hr /><form asp-action="Delete"><input type="hidden" asp-for="Id" /><input type="submit" value="Delete" class="btn btn-danger" /> | <a
asp-action="Index">Back to List</a></form></div>
```

Index.cshtml

```
@{ ViewData["Title"] = "Welcome"; }
<div class="text-center">
    <h1 class="display-4">Welcome to DreamDay</h1>
    <p>Your personal wedding planning assistant.</p>
    <div class="mt-4">
        <a class="btn btn-primary btn-lg" asp-controller="Account" asp-action="Register">Get Started</a>
        <a class="btn btn-secondary btn-lg" asp-controller="Account" asp-action="Login">Login</a>
```

```
    </div>
</div>
```

Index.cshtml

```
@model IEnumerable<DreamDay.Models.Message>
@{ ViewData["Title"] = "Messages"; var otherUser = ViewBag.OtherUser as DreamDay.Models.User; }
@if (otherUser != null){<h1>Conversation with @otherUser.Name</h1>
<div class="border p-3 mb-3" style="height: 400px; overflow-y: scroll;">
@foreach (var message in Model){<p class="@(message.SenderId == otherUser.Id ? "text-start" : "text-end")"><strong>@(message.SenderId == otherUser.Id ?
otherUser.Name : "Me"):</strong> @message.Content<br /><small class="text-muted">@message.SentAt.ToString("g")</small></p>}</div>
<form asp-action="Send" method="post"><input type="hidden" name="receiverId" value="@otherUser.Id" />
<div class="input-group"><input type="text" name="content" class="form-control" placeholder="Type your message..." /><button type="submit" class="btn btn-
primary">Send</button></div></form>}
else {<p>You do not have anyone to message yet.</p>}
```

Index.cshtml

```
@model IEnumerable<DreamDay.Models.Wedding>
@{ ViewData["Title"] = "Planner Dashboard"; }
<h1>My Managed Weddings</h1><p>Select a wedding to view its dashboard and manage its details.</p>
<div class="list-group">@foreach (var wedding in Model){
<div class="list-group-item list-group-item-action"><div class="d-flex w-100 justify-content-between"><h5 class="mb-
1">@wedding.Title</h5><small>@wedding.WeddingDate.ToShortDateString()</small></div>
<p class="mb-1">Client: @wedding.Couple.User.Name</p><small>Status: @wedding.Status</small>
<div class="mt-2"><a asp-controller="Dashboard" asp-action="Index" asp-route-id="@wedding.Id" class="btn btn-sm btn-primary">View Dashboard</a> <a asp-
controller="Report" asp-action="BudgetReport" asp-route-weddingId="@wedding.Id" class="btn btn-sm btn-info" target="_blank">Generate Budget
Report</a></div></div>}</div>
```

BudgetReport.cshtml

```
@model DreamDay.Models.Budget
@{ Layout = null; var wedding = ViewBag.Wedding as DreamDay.Models.Wedding; var totalSpent = Model.BudgetCategories.SelectMany(c => c.Expenses).Sum(e =>
e.Amount); }
<!DOCTYPE html><html><head><title>Budget Report - @wedding.Title</title><link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" /><style> body { padding: 20px; } </style></head>
<body><h1>Budget Report</h1><h2>@wedding.Title</h2><p><strong>Client:</strong> @wedding.Couple.User.Name</p><p><strong>Report Generated:</strong>
@DateTime.Now.ToShortDateString()</p><hr />
<h3>Summary</h3><p><strong>Total Wedding Budget:</strong> @wedding.Budget.ToString("C")</p><p><strong>Total Spent:</strong> <span
style="color:red;">@totalSpent.ToString("C")</span></p><p><strong>Remaining:</strong> @(wedding.Budget - totalSpent).ToString("C")</p><hr />
<h3>Breakdown by Category</h3>
@foreach (var category in Model.BudgetCategories){ var categorySpent = category.Expenses.Sum(e => e.Amount);
<h4>@category.Name</h4><p>Allocated: @category.AllocatedAmount.ToString("C") | Spent: @categorySpent.ToString("C")</p>
<table class="table table-sm table-bordered"><thead><tr><th>Expense</th><th>Amount</th></tr></thead><tbody>
@foreach (var expense in category.Expenses){<tr><td>@expense.Description</td><td>@expense.Amount.ToString("C")</td></tr>}</tbody></table>}</body></html>
```

Index.cshtml

```
@{ ViewData["Title"] = "Seating Arrangement"; }
<h1>Seating Arrangement</h1>
<div class="row"><div class="col-md-8"><h3>Tables</h3><div class="row">
@foreach (var table in ViewBag.Tables){<div class="col-md-6 mb-3"><div class="card"><div class="card-header">@table.Name (@table.SeatedGuests.Count /
@table.Capacity)</div>
<ul class="list-group list-group-flush">@foreach (var guest in table.SeatedGuests){<li class="list-group-item">@guest.Name</li>}</ul></div></div>}</div></div>
<div class="col-md-4"><h4>Add Table</h4><form asp-action="AddTable" method="post" class="mb-4">
<div class="form-group mb-2"><input type="text" name="name" class="form-control" placeholder="Table Name" required /></div>
<div class="form-group mb-2"><input type="number" name="capacity" class="form-control" placeholder="Capacity" required /></div>
<button type="submit" class="btn btn-success">Add Table</button></form><hr />
<h4>Assign Guest</h4><form asp-action="AssignGuest" method="post">
<div class="form-group mb-2"><label>Guest</label><select name="guestId" asp-items="ViewBag.UnseatedGuests" class="form-control"><option>-- Select Guest --
</option></select></div>
<div class="form-group mb-2"><label>Table</label><select name="tableId" asp-items="ViewBag.TableList" class="form-control"><option>-- Select Table --
</option></select></div>
<button type="submit" class="btn btn-primary">Assign</button></form></div></div>
```

Index.cshtml

```
@model DreamDay.Models.Timeline
@{ ViewData["Title"] = "Wedding Timeline"; }
<h1>Wedding Day Timeline</h1>
<div class="row"><div class="col-md-8"><ul class="list-group">
@foreach (var item in Model.TimelineEvents){<li class="list-group-item"><strong>@item.StartTime.ToString("hh:mm tt")</strong> - @item.Title</li>}</ul></div>
<div class="col-md-4"><h4>Add Event</h4><form asp-action="AddEvent" method="post">
<div class="form-group mb-2"><label>Title</label><input type="text" name="title" class="form-control" required /></div>
<div class="form-group mb-2"><label>Start Time</label><input type="datetime-local" name="startTime" class="form-control" required /></div>
<button type="submit" class="btn btn-primary">Add Event</button></form></div></div>
```

Index.cshtml

```
@model IEnumerable<DreamDay.Models.Vendor>
@{ ViewData["Title"] = "Vendor Catalog"; }
<h1>Vendor Catalog</h1><p>Browse our curated list of professional vendors.</p>
<div class="row">@foreach (var item in Model){<div class="col-md-4 mb-4"><div class="card"><div class="card-body">
<h5 class="card-title">@item.Name</h5><h6 class="card-subtitle mb-2 text-muted">@item.Category</h6>
<p class="card-text">@item.Description</p><a asp-action="Details" asp-route-id="@item.Id" class="card-link">View Details</a></div></div></div>}</div>
```

Details.cshtml

```
@model DreamDay.Models.Vendor
@{ ViewData["Title"] = "Vendor Details"; }
<h1>@Model.Name</h1><h4>@Model.Category</h4><hr />
<div><p><strong>Description:</strong> @Model.Description</p><p><strong>Contact:</strong> @Model.ContactInfo</p></div>
<div class="mt-4"><h3>Services Offered</h3><ul class="list-group">
@foreach(var service in Model.Services){<li class="list-group-item">@service.Name - @service.Price.ToString("C")</li>}</ul></div>
```

```
<div class="mt-4"><h3>Reviews</h3>
@foreach(var review in Model.Reviews){<div class="card bg-light mb-2"><div class="card-body">
<p><strong>Rating: @review.Rating / 5</strong></p><p>"@review.Comment"</p>
<footer class="blockquote-footer">@review.Couple.User.Name on @review.DatePosted.ToShortDateString()</footer></div></div>}</div>
<div class="mt-4"><a asp-action="Index">Back to Catalog</a></div>
```

Create.cshtml

```
@model DreamDay.Models.Wedding
@{ ViewData["Title"] = "Create Your Wedding"; }
<h2>Let's Plan Your Dream Day!</h2><p>Fill out some basic details to get started.</p>
<div class="row"><div class="col-md-8"><form asp-action="Create">
<div class="form-group mb-3"><label asp-for="Title" class="control-label"></label><input asp-for="Title" class="form-control" placeholder="e.g., The Smith &
Jones Wedding" /></div>
<div class="form-group mb-3"><label asp-for="WeddingDate" class="control-label"></label><input asp-for="WeddingDate" type="date" class="form-control" /></div>
<div class="form-group mb-3"><label asp-for="Budget" class="control-label"></label><input asp-for="Budget" class="form-control" /></div>
<div class="form-group"><input type="submit" value="Start Planning" class="btn btn-success" /></div></form></div></div>
```