**2025SP Team Project – submission 1**

Your team is to build an OpenTable web app using the DOTNET Core MVC framework. The commercial website is [www.opentable.com](http://www.opentable.com), but you will build a much simpler version.

In Submission 1, your team project submission should meet the following function and technical requirements.

**Function requirement:**

1. Your web application is to serve two major categories of users: regular users and administrators.

2. Build 3 layouts for your web application.

* \_Layout: the outermost layout. It contains the resources common to all razor views in this app, especially a footer that stays the same for all users. The below screenshot is for your reference, but you are NOT expected to build these many links. Instead, you should at least include ‘About’, ‘Contact’, ‘Privacy Policy’, ‘Term of Use’, and ‘Cookie and Interest-Based Ads’. A black and white website

  AI-generated content may be incorrect.
* \_MainLayout: an intermediary layout with navigations for regular users. The navigation should include the OpenTable logo, a drop-down list for Metropolis including Chicago, San Francisco, Los Angeles, Phoenix, Houston, Atlanta, Miami, Charlotte, New York, and Boston, and sorting in alphabetical order. Additionally, this layout should include links for ‘Mobile’, ‘Admin (replacing For Business)’, and ‘FAQs’. When clicking ‘Admin’ link, it enters the home page for Admin area.

A screenshot of a computer

AI-generated content may be incorrect.

* \_AdminLayout: another intermediary layout with navigation for administrators. This layout should include OpenTable logo, and a link ‘Site’. Upon clicking ‘Site’, it returns to the home page for main area (regular user). You can reference GuitarShop\_C in the textbook chapter 7.

3. Two home pages for regular users and administrators, respectively.

* Home page for regular users: include links for ‘Fina a Table’ and ‘Manage Reservation’.
* Home page for administrators: include links for ‘Manage Metropolis’, ‘Manage Restaurant’, and ‘Manage Users’.

**Technical Requirement:**

1. With learned skills from chapters 6 and 7:

* Create OpenTable app structure with two areas: Admin and main. Two areas will share the same model classes but have different controllers and views.
* At this time, you just need to create the views to the home page level. For the links included in the navigations, return a Content object stating ‘Area:xxx’, ‘Controller:xxx’ , ‘Action:xxx’ (xxx is a placeholder to be replaced properly in your app). For routing in the Admin area, use ‘attribute routing .’ For routing in the main area, use regular routing.
* For main area, create at least two controllers: HomeController handles non-business-essential functions such as Index(), About(), Privacy(), etc, and ReservationController handles List(), Add(), Update(), and Delete(). For Admin area, create at least four controllers: HomeController handles Index(), MetropolisController handles metropolis add/update/delete/list, RestaurantController handles restaurant add/update/delete/list, and UserController handles user add/update/delete/list. Note, you are not required to implement these actions, because we will build them in subsequent submissions incrementally.

1. Create 3 layouts.
   1. The outermost layout stores CSS links and javascript libraries, and it contains a header and footer that are universal to all views. This layout should be placed in the main area.
   2. The two intermediary layouts are for Admin and regular users, respectively. They should be placed in Admin and main area accordingly. Clicking the ‘Admin’ link in the \_MainLayout will access to the Admin area, and clicking ‘View Site’ link in the Admin area will return to the main area.
   3. \_MainLayout is nested in \_Layout, and \_AdminLayout is also nested in \_Layout.
2. In this project submission, the focus is to **construct the app structure with two areas and nested layouts, a simple home page for each area, and two different routing mechanism that can be easily tested with a Content object**. Nothing else. Please do not move advanced; do not build things that are not required (for example, authentication/authorization for admin access or actions for adding/deleting/listing vaccines, etc.). At this point, there are no other razor views except for layouts and the home page. I need to see this app incrementally constructed so that you can practice the skills you freshly learn each week. If you jump ahead with functions that are not required (or never required), you will lose points and reverse engineering them.

**Review Summary**

This summary is finished by the designated review team but submitted by the author team. The review should cover both functionalities and code qualities. If necessary, the reviewing team can use snapshots of screens and source code, with proper text, to describe the flaws and strengths identified in the reviewed project based on the requirements.

**Submission Requirement**

1. Zipped source code (at project root level).
2. Screenshots with major functionality features (no source code).
3. Review Summary