# ASP.NET Fundamentals Regular Exam – 10 December 2024

# A screenshot of a screen Description automatically generated

Problems for the retake exam for the ["ASP.NET Core Fundamentals"](https://softuni.bg/trainings/4707/asp-net-fundamentals-september-2024) course @ [SoftUni](https://softuni.bg)

## Welcome to Horizons

***Horizons*** *is a web application designed to inspire and guide travelers by showcasing the best tourist destinations in Bulgaria. Whether you're looking to explore breathtaking natural landmarks or historical sites, Horizons provides a platform where users can discover, save, and share their favorite destinations.* ***Logged-in users*** *can create and manage their own* ***Favorites list****, while each destination tracks its popularity through a* ***counter for user engagement****.*

## Technological Requirements and Overview

* Use the provided skeleton – [**Horizons-Skeleton.zip**](http://svn.softuni.org/admin/svn/soft-tech/Sept-2024/CSharp/15-CSharp-Web-Oct-2024/01-Regular-Exam/DeskMarket-Skeleton.zip)
* All of the needed packages have been installed.

**The Technological Requirements are ABSOLUTE. If you do not follow them, you will NOT be scored for other Requirements.**

The provided skeleton consists of:

* **Areas/Identity/Pages** – you have to **scaffold Identity** here
* **Controllers** – you should implement the controllers logic here
* **Data** – you should hold the entities models here
* **Models** – you should implement the models here
* **Views** – you are **provided with** the needed **views**. Your task is to implement some logic  
   regarding the **logged-in**/**logged-out** user
* **appsettings.json** – don't forget to change your **connection string**
* **Program.cs** – you should fulfill the **security and password requirements** here

**NOTE:** You should seed the database with the data provided in advance regarding the **Category** entity. To do this, remove the comments from the block of code in the **protected override void OnModelCreating(ModelBuilder builder)** method of the **DbContext.**

**NOTE: Don't forget to uncomment the code inside the views while you implement your logic.**

Now that you know the **Technological Requirements**, let us see what the **Functional Requirements** are.

## Identity Requirements

You should **scaffold** **Identity** and use the **default** **IdentityUser**.

Remove the unnecessary code from the **Login.cshtml** and **Register.cshtml** files and leave only the needed code for the app to function correctly.

**NOTE: Don't worry about the views' style – once you scaffold Identity and remove the unnecessary code, the** [**Login**](#_Login_Page_(logged-out) **and** [**Register**](#_Register_Page_(logged-out) **pages should look like the ones shown below. Don't add any classes to the views of those two pages!**

The **password** requirements for the **IdentityUser** are the following:

* Require confirmed account: **false**
* Require digits: **false**
* Require non-alphanumeric characters: **false**
* Required uppercase letters: **false**
* Required lowercase letters: **false**

## Database Requirements

The **Database** of **Horizons**:

### Destination

* Has Id – a unique **integer, Primary Key**
* Has Name – a string with **min** length **3** and **max** length **80** (**required**)
* Has Description – string with **min** length **10** and **max** length **250** (**required**)
* Has **ImageUrl** – nullable string (**not required**)
* Has PublisherId – string (**required**)
* Has Publisher – IdentityUser (**required**)
* Has PublishedOn – DateTime with format "**dd-MM-yyyy**" (**required**)
  + The **DateTime** format is only recommended
* Has TerrainId – **integer, foreign key (required)**
* Has Terrain – Terrain (**required**)
* Has IsDeleted – bool **(default value == false)**
* Has **UsersDestinations** – a collection of type **UserDestination**

### Terrain

* Has Id – a unique **integer, Primary Key**
* Has Name – a string with min length **3** and max length **20** (**required**)
* Has **Destinations** – a collection of type **Destination**

### UserDestination

* Has UserId – string, PrimaryKey, foreign key (**required**)
* Has User – IdentityUser
* Has DestinationId – integer, PrimaryKey, foreign key (**required**)
* Has Destination – Destination

Implement the entities with the **correct datatypes** and their **relations**.

**Feel free to use the new syntax for the realization of the many-to-many relation without a mapping table.**

## Page Requirements

### Index Page (logged-out user)

A screenshot of a website

Description automatically generated

A screenshot of a website

Description automatically generated

### Index Page (logged-in user)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Note: The Destination Index Page** will have **different behavior** for **logged-in** and **not-logged-in** users. Also, **different destinations' cards** will behave differently for **publishers** and **non-publishers**.

A screenshot of a computer

Description automatically generated

### /Destination/Index (logged-out user)

A screenshot of a computer

Description automatically generated

**NOTE**: If the current user is not **logged in** they will be still able to access the **/Destination/Index** page

### Login Page (logged-out user)

A screenshot of a login form

Description automatically generated

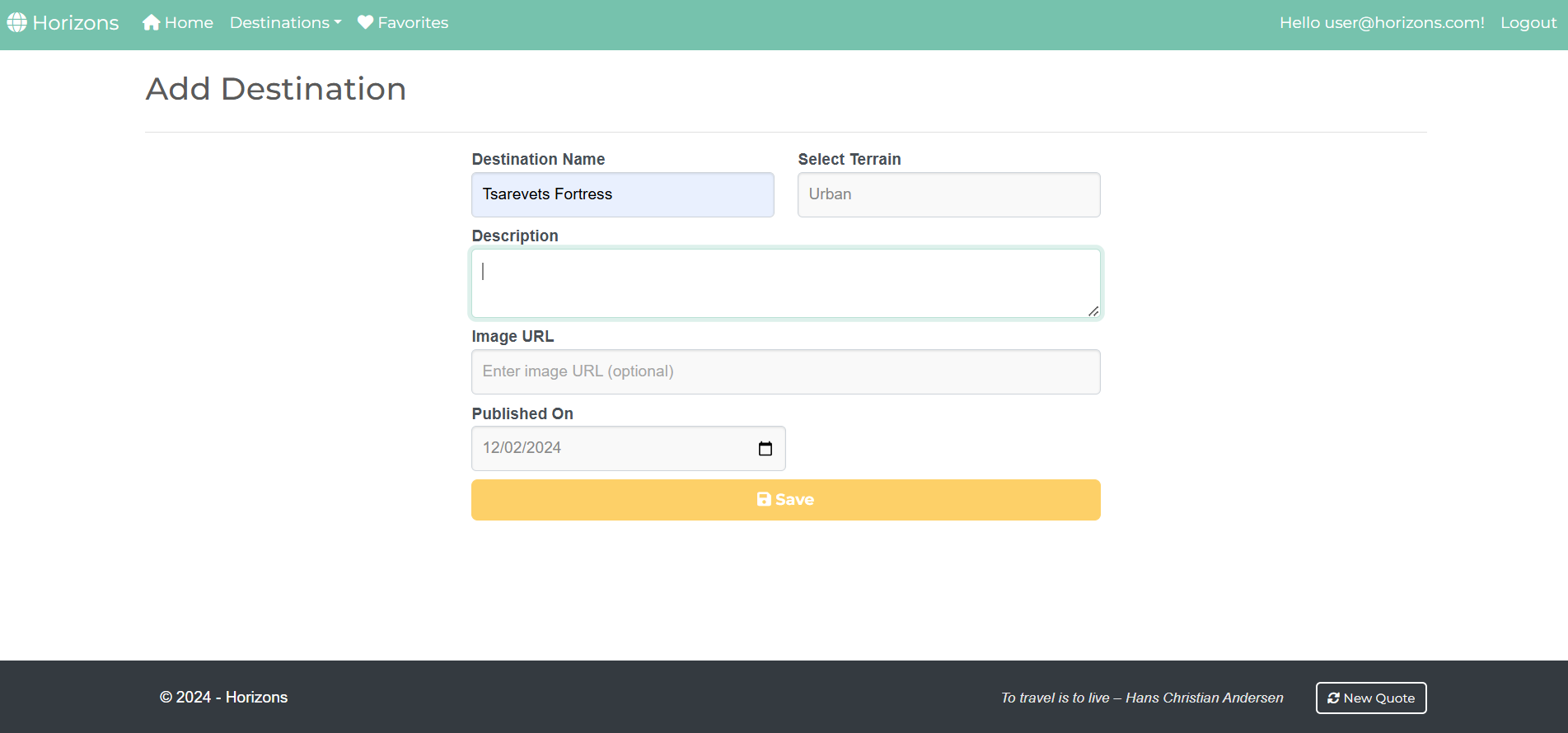
### Register Page (logged-out user)

A screenshot of a computer

Description automatically generated

**Note: The Register Page** should be **accessible** from the **"Start Your Adventure" button** from the Index Page for  
(logged-out user)

### /Destination/Add (logged-in user)



**NOTE:** the **DateTime** format is recommended, if you are having trouble with the one in the image, you are **free** to use **another** one.

### /Destination/Index (logged-in user)

A screenshot of a computer

Description automatically generated

**NOTE**: If a user is **logged in** and tries to go to the **Home page**, the application must redirect them to the **/Destination/Index**

### /Destination/Favorites (logged-in user)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**NOTE**:

* When a user **removes a favorite destination** from the list, the application **should update the view** **without redirecting the user**
* If **no favorites are left after a removal action**, the **user should remain on the same page** and see the "You don't have any favorite destinations yet!" message

**Destination/Details/{id} (logged-in user, publisher of a destination)**

A screenshot of a computer

Description automatically generated

**Destination/Details/{id} (logged-in user, not-publisher of a destination, the destination is not added to favorites)**

A screenshot of a computer

Description automatically generated

**Destination/Details/{id} (logged-in user, not-publisher of a destination, the destination is already added to favorites)**

A screenshot of a computer

Description automatically generated

**Destination/Details/{id} (guest user)**

**A screenshot of a computer

Description automatically generated**

**NOTE**:

* If the **currently logged-in user is the publisher of the destination**, the **"Delete"** and **"Edit"** buttons should be **visible**, allowing them full access to the destinations functionalities.
* If the **logged-in user is** **not the publisher** of the destination, the **"Delete"** and **"Edit"** buttons should **NOT** be **visible**. The **"Favorites"** button should be **visible**. This ensures that the **destination publisher** and   
  **Not-publisher-users** have different application functionalities access.
* Guests (**unauthenticated users**) should be able to **access the destination's details page** but **won't have access to actions** like **adding to favorites**, **editing**, or **deleting** a destination.

### /Destination/AddToFavorites?id={id} (logged-in user)

A screenshot of a phone

Description automatically generated

This **feature is accessible by the Favorites button** in every destination card and **adds the selected destination to the user's Favorites collection**. If the destination is **already in the collection**, it **should not be added again**. If the **addition is successful**, the users will **remain on the current page**, (either **Destination/Index** or **Destination/Details**) rather than being redirected to the favorites page.

### /Destination/RemoveFromFavorites?id={id} (logged-in user)

A screenshot of a computer

Description automatically generated

This **feature is accessible by the Remove button** in the "**/Destination/Favorites**" page and **removes the selected destination** from the user's **favorites collection**. If the **removal is successful**, the user will **remain on the current page** "**/Destination/Favorites**" page.

**NOTE**: The templates should look **EXACTLY** as shown above.

### /Destination/Edit/{id} (logged-in user) A screenshot of a computer Description automatically generated

**NOTE:** the **DateTime** format is recommended, if you are having trouble with the one in the image, you are **free** to use **another** one.

**Destination/Delete/{id} (logged-in user, seller of a product)**

When a **logged-in user** who is the **publisher of a destination** accesses this page, they can initiate the deletion of the destination. This process uses **soft delete functionality**, meaning the **destination will remain in the database** but will **no longer appear in destination listings** or be accessible to users.

**After confirming the deletion**, the publisher will be **redirected to the Destination controller's Index action**. This operation is **one-directional and irreversible through the user interface**. Once deleted, the **destination will not be displayed** in the application unless it is manually restored via the database.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functionality

The functionality of the **Horizons** Platform is a very simple application.

### Users

Guests can Register, Login and view the **Home Page**, Index Page and Details Page.

**Logged-in users** can **publish** destinations and **edit** destinations **they have created**.

Users (**guest-users + logged-in-users**) can **view all published destinations**, including those by other users, on the **Home Page** (**/Destination/Index**).

If the **user is the publisher of the destination**

* They **can't see** the **[Favorites]** button
* They can see the **[Details]** button

If the user is **not the publisher** of the destination, they can add the destination to their collection [**Favorites**].

### Destination

Destinations can be **added** by **logged-in-**users. All added destinations are visualized on the Home Page (/Destination/Index) with **some** of their associated **information**.

Destinations are visualized on the Index Page (/Destinations/Index) with **one or two buttons**:

* If the user **IS NOT** the **publisher** of the destination – **[Details][Favorites]**
* If the user **IS** the **publisher** of the destination – **[Details]**

The **[Favorites]** button adds the destination to the user's collection of favorite destinations (Favorites) **unless it is already added**.

The **[Details]** button displays a new page with the full information for the selected destination. The actions and buttons displayed on this page will vary depending on the user's identity and role:

* **Guest (Not Logged In)**:  
  The user can see the **full details of the destination** but will **not be able to add to favorites**, **edit**, or **delete** the destination. No action buttons (**[Favorites]**, **[Edit]**, **[Delete]**) will be shown.
* **Logged-in User (Not the Publisher)**:  
  The user will **see the full details** of the destination and will **have access to the [Favorites]** button, allowing them to **add the destination to their collection of favorite destinations** (if it has not already been added). The **[Edit]** and **[Delete]** buttons will not be displayed as they are **not the publisher** of the destination.
* **Publisher of the Destination**:  
  The publisher will have **full access to the destination's details** and will be able to see both the **[Edit]** and **[Delete]** buttons. However, the **[Favorites]** button will not be displayed since the publisher cannot add their own destination.

The **[Edit]** button displays a new page with a form, filled in with **all** of the **info** for the **selected** **destination**. Users can **change this info and save it**.

Usershave a Favorites page where only the destinationsthey have selected as favorites are displayed.

* The **[Remove]** button **removes** the selected **destination** from the **user's** collection of **favorite destinations**.

### Redirections

* Upon successful Login of an **IdentityUser**, you should be redirected to the /Destination/Index
* Upon successfully Publishing a Destination, you should be redirected to the /Destination/Index
* Upon successfully Adding a **Destination** to the **User's Favorites**, should be redirected to /Destination/Index
* Upon successful **Editing** of a **Destination,** you should be redirected to **/Destination/Details/{product\_id}**
* Upon successful Removal of a **Destination** from the **User's Favorites**, should be redirected to the /Destination/Favorites
* If a User **adds** a destination to the **Favorites** collection, the **[Favorites]** button **disappears**, and the user should not be able to duplicate destinations in the **User's Favorites.**
* Upon successful Logout of a User, you should be redirected to the Home Page
* If any of the **validations** in the POST forms **don't pass**, **redirect** to the **same page** (**reload/refresh** it)

## Security

The Security section mainly describes access requirements.

**Users-only** pages (Add, Edit, Favorites, Delete)

Configurations about which users can access specific functionalities and pages:

* Guest (not logged-in) users can access the Home page, **Destination Index** page, and **Destination Details** page
* Guest (not logged-in) users can access the Login page
* Guest (not logged-in) users can access the Register page
* Guests (not logged in) cannot access Users-only pages
* Users (logged in) cannot access Home page
* Users (logged in) can access the Destination/Add page and functionality
* Users (logged in) can access the Destination/Edit page and functionality
* Users (logged in) can access theDestination**/Favorites** page
* Users (logged in) can access Logout functionality
* Users (logged in) cannot access the Destination/Edit page of a destination that has another user as a publisher
* Users (logged in) cannot access the Destination/Delete page of a destination that has another user as a publisher

## Code Quality

Make sure you provide the best architecture possible. Structure your code into different classes, and follow the principles of high-quality code (**SOLID**). You will be scored for the Code Quality and Architecture of your project.

## Scoring

### Identity Requirements – 5 points

### Database Requirements – 10 points

### Template Requirements – 10 points

### Functionality – 50 points

### Security – 5 points

### Code Quality – 10 points

### Data Validation – 10 points