# C# Web Basics Exam – 19 April 2022

# Bus Station

Exam problems for the [C# Web Basics course @ SoftUni](https://softuni.bg/trainings/3593/csharp-web-basics-basics-january-2022). Submit your solutions in the **SoftUni judge** system (delete all "**bin**"/"**obj**" folders).

**Bus Station** is an online platform that is used to create destinations, tickets, and book trips.

## Technological Requirements

* Use the **MyWebServer – 5.0**
* Use the **BasicWebServer.Server – 6.0**

The Technological Requirements are **ABSOLUTE**. If you do not follow them, you will **NOT** be scored for other Requirements. All the **NuGet packages** which are **necessary** to solve the exam are installed.

When the **final work is submitted**, make sure that the project could be **run (build) successfully**. When starting the submitted project a **compile-time error occurs**, the project will receive **zero points**, without further evaluation.

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

## Database Requirements

The **Database** of **Buss Station**:

### User

* Has an Id – an **string, Primary Key**
* Has a Username – a string with **min length** **5** and **max length 20** (**unique**, **required**)
* Has an Email – a string with **min length** **10** and **max length 60** (**unique**, **required**)
* Has a Password – a string with **min length** **5** and **max length 20 (before hashed)** - no max length required for a hashed password in the database (**required**)
* Has **Tickets** collection

### Destination

* Has Id – an **int, Primary Key**
* Has DestinationName – a string with **min length** **2** and **max length 50** (**required**)
* Has **Origin** – a string with **min length** **2** and **max length 50** (**required**)
* Has **Date** – a string with **max length 30** (**required**). We recommend using **"d"** as a date format. The **date** cannot be smaller than the date of the creation of the destination.
* Has **Time** – a string with **max length 30** (**required**). We recommend using **"t"** as a time format.
* Has ImageUrl – a string (**required**)
* Has **Tickets** collection

**Note:** The recommended **date** and **time** format are **optional**, you can use any other format.

### Ticket

* Has Id – an **int, Primary Key**
* Has Price – a **decimal,** between 10 and 90
* Has **UserId** – an **int**
* Has DestinationId – an **int**

Implement the entities with the **correct data types** and their **relations**.

## Page Requirements

### Index Page (logged-out user)

Graphical user interface, application

Description automatically generated

### Register Page (logged-out user)

Graphical user interface, application

Description automatically generated

### Login Page (logged-out user)

Graphical user interface, application, website

Description automatically generated

### /Destinations/All (logged-in user)

Graphical user interface, application

Description automatically generated

**NOTE**: If the user is logged in and tries to go to the home page, the application must redirect him to the **/Destinations/All**

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

Graphical user interface, application, Word

Description automatically generated

### /Tickets/Create?destinationId={destinationId} (logged-in user)

Graphical user interface, application

Description automatically generated

Adds tickets to a current destination. The price of tickets should be between **10 and 90** euros. A user can add up to **10 tickets** at a time. If everything is successful, the user must be redirected to the home (**"/Destinations/All"**) page.

### /Tickets/MyTickets (logged-in user) Graphical user interface, website Description automatically generated

Here is where all the user's booked trips are displayed.

**Note:** A user can add **more than one** ticket for a destination.

**Note:** The **date** and **hour** should be passed to the view as an **interpolated** **string**.

## Functionality

The functionality of the **Bus Station** platform is very simple.

### Users

Guests can Register, Login, and view the Index Page.

Users can AddDestinations, Add Tickets, see all added Destinations on the Home Page, and see all user's tickets. From the Home Page, they can also view My Tickets and Destinations.

### Destinations

Destinations can be Added by Users. All created Destinations are visualized on the Home Page, each one in its separate rectangular element.

Destinations are visualized on the Home Page with all their information, including the **number of available** tickets they have. When a destination is added, it starts with **zero tickets**.

Destinations are visualized on the Home Page with two buttons **[Add Tickets]** and **[Book a Trip]**:

* The [**Add Tickets**] button adds tickets to the given destination. А user can add a ticket to any destination. The ticket has a **Price** and **Tickets count**. Up to **10 tickets** can be added at a time and the price of tickets should be between **10 and 90** euros.
* The [**Book a Trip**] button reserve a ticket for the given destination and adds it to the user's booked trips. When a ticket is added, it is no longer available for other users. A user **cannot** book a trip to a destination without available tickets. If the user tries to book a trip for a destination without available tickets, redirect the user to **"Destinations/All"**.

### Redirections

* Upon successful Registration of a User, you should be redirected to the Login Page.
* Upon successful Login of a User, you should be redirected to the **/Destinations/All**.
* Upon successful Adding of a Destinations, you should be redirected to the **/Destinations/All**.
* Upon successful Create Tickets to the Destinations, the user should be redirected to the **/Destinations/All**.
* Upon successful Book Trip**,** the usershould be redirected to the **/Destinations/All**.
* If a User tries to **add** a Ticket to Destination, which does not have any available tickets, they should be redirected to /Destinations/All (or just a page refresh).
* Upon successful Logout of a User, you should be redirected to the Index 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. Configurations about which users can access specific functionalities and pages:

* Guest (not logged in) users can access the Index 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 Guest pages.
* Users (logged in) can access the Destinations All page and its functionality.
* Users (logged in) can access the Destinations Add the page and its functionality.
* Users (logged in) can access the Tickets Create page and its functionality.
* Users (logged in) can access the Tickets MyTickets page.
* Users (logged in) can access Logout functionality.

## Code Quality

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

## Scoring

### Database Requirements – 10 points.

### Template Requirements – 10 points.

### Functionality – 50 points.

### Security – 10 points.

### Code Quality – 10 points.

### Data Validation – 10 points.