IRunes

You have to implement the **IRunes application** – a simple **music store application**.

Technological Requirements

- Use the **SIS**
- Use **Entity Framework Core**

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

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

Database Requirements

You have been tasked to implement a simple application, using the Web Server. The application imitates a **store** for **Music Albums** and **Music Tracks**. You will see the functionality – described below.

The first thing you need to do is implement the Database entities. Use Entity Framework Core, and implement the following entities:

User

- `Id` – a `string` (**GuID**), Primary key
- `Username` – a `string` with **min length 4** and **max length 10 (inclusive) (required)**
- `Password` – a `string` with **min length 6** and **max length 20 (inclusive) - hashed** in the database (**required**)
- `Email` – a `string` **(required)**

Album

- `Id` – a `string` (**GuID**), Primary key
- `Name` – a `string` with **min length 4** and **max length 20 (inclusive) (required)**
- `Cover` – a `string` (a **link** to an **image**) **(required)**
- `Price` – a `decimal` (**sum** of all **Tracks**' **prices**, **reduced** by **13%**) **(required)**
- `Tracks` – a **collection** of `Tracks`

Track

- `Id` – a `string` (**GuID**), Primary key
- `Name` – a `string` with **min length 4** and **max length 20 (inclusive) (required)**
- `Link` – a `string` (a **link** to a **video**) **(required)**
- `Price` – a `decimal` **(required)**

Templates

You will be shown how the **template looked before**, and **how it should look now**. Style it **using Bootstrap**, to match the given screenshot. Be as precise as you can.

1. Index (guest, logged-out) (route="/")

2. Login (guest, logged-out) (route="/Users/Login")

3. Register (guest, logged-out) (route="/Users/Register")

IRunes Home Login Register

Register

Username

Username...

Password

Password...

Confirm Password

Confirm Password...

Email

Email...

Register

4. Index (user, logged-in) (route="/Home/Index", route="/")

IRunes Home Albums Logout

Welcome, Pesho

IRunes wishes you a fun experience.

5. All Albums (user, logged-in) (route="/Albums/All")

IRunes Home Albums Logout

Albums

Create Album

Kamikaze
Koncerta Na Slavi

Follow us:

6. Album Create (user, logged-in) (route="/Albums/Create")

IRunes   Home   Albums   Logout

Create Album

Name

> Name...

Cover

> Cover...

Create   Back To All

CopyRight © Sanity Design Studios. All rights reserved.

7. Album Details (user, logged-in) (route="/Albums/Details?id={albumId}")

IRunes   Home   Albums   Logout

Tracks
- 1. *Fall*
- 2. *Lucky You*

Album Name: Kamikaze
Album Price: $5,84

Create Track   Back To All

CopyRight © Sanity Design Studios. All rights reserved.

8. Track Create (user, logged-in) (route="/Tracks/Create?albumId={albumId}")

IRunes   Home   Albums   Logout

Create Track

Name

Name...

Link   Price

Link...   Price...

Create   Back To Album
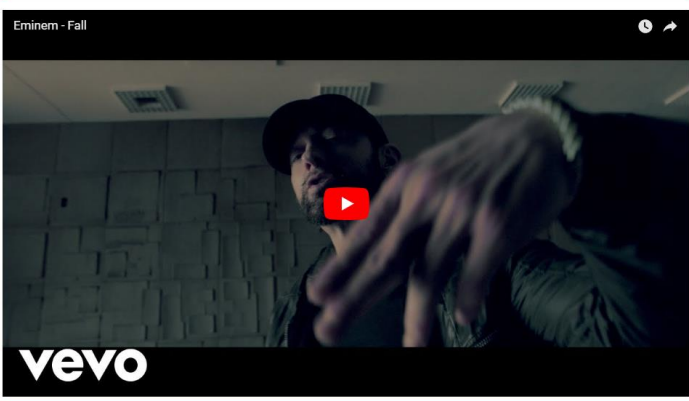
CopyRight © Sanity Design Studios. All rights reserved.

9. Track Details (user, logged-in) (route="/Tracks/Details?albumId={albumId}&trackId={trackId}")

IRunes   Home   Albums   Logout

Track Name: Fall
Track Price: $3,59

Back To Album

CopyRight © Sanity Design Studios. All rights reserved.

Functional Requirements

The functional requirements are quite simple.

Users

The application should provide **guests** (logged-out) with the **functionality** to access:

- The **Guest Index Page**

Follow us:

- The **Login Page** and **Functionality**
- The **Register Page** and **Functionality**

The application should provide **users** (logged-in) with the **functionality** to access:

- The **User Index Page**
- The **All Albums Page**
- The **Album Create Page**
- The **Album Details Page**
- The **Track Create Page**
- The **Track Details Page**

Albums

The **Albums** are **created** and **presented** on the **All Albums Page**, in a **listed format** with only their **names** as elements. Each **album name** should be a **link** which **leads** to the corresponding **Album**'s **Details Page**.

If there **are no Albums currently** in the Database, a message "**There are currently no albums.**" should be printed.

On the **Album**'s **Details Page**, its tracks should be **listed**, in an **indexed list**, **starting** from **1**. The **order** of **data** is **not mandatory**.

Each **track name** should be a link which leads to the corresponding **Track**'s **Details Page**.

Logged in **User** should be able to create a new **Album**.

Tracks

The **Tracks** are **created** and **presented** on their **Album's Details Page**. **Tracks** are created using the **Album**'s **id** which is passed through the **query parameters**.

When you **create** a **Track**, you can pass it the **iframe-ready url**, in order to make your work easier.

Logged in **User** should be able to create a new **Track**.

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.

Data Validation

- In case of invalid data during the **login**, redirect to **"/Users/Login"**
- In case of invalid data during the **registration**, redirect to **"/Users/Register"**
- In case of invalid data during the **album creation**, redirect to **"/Albums/Create"**
- In case of invalid data during the **track creation**, redirect to **"/Tracks/Create?albumId={albumId}"**

Scoring

- Database Requirements – 10 points.
- Template Requirements – 10 points.
- Functionality – 50 points.

- Security – 10 points.
- Application Structure – 10 points.
- Data Validation – 10 points.