System Requirements Specification Index

For

Track Management Application (Console)

Version 4.0

IIHT Pvt. Ltd.
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

TRACK MANAGEMENT CONSOLE APPLICATION

System Requirements Specification

1.BUSINESS-REQUIREMENT:

Name of the Course : C#

Tool Stack : .net core console application

Problem Statement :

Build an application to work with Track handling like Dotnet, Java etc.

Description :

1. Take console input from the user in the Main method of the program.

2. Take input of details of each track in the following format.
   Track Id, Track Name, Track Duration, Track Lead,NumberofCampusMinds
   Eg: 2, MyTrack, 12-12-2020,MyTrackLead,3

3. Create Track class(Entity) with
   a. TrackId as int
   b. TrackName as string
   c. TrackDuration as datetime
   d. TrackLead as string
   e. NumberofCampusMinds as int
   f. Create one static method as below:
      i. static IEnumerable<Object> AddTrack(Track track)
      which accepts Entity Object as input, add the input to the list

   g. Create another static method as below:
      i. static IEnumerable<Object> UpdateTrack(Track track)
      which accepts Entity Object as input, check TrackId and update the details to the list

   h. Create another static method as below:
      i. static IEnumerable<Object> ShowAllTracks()
      Display all the list objects in the console

   i. Create another static method as below:
      i. static IEnumerable<Object> GetTrackById(TrackId)
      Return Track details by trackId

   j. Create another static method as below:
      i. static IEnumerable<Object> RemoveTrackById(TrackId)
      ii. Remove track details from list by trackId

2.Resources available:

2.1 Package: TrackHandling

Names	Resource	Remarks	Status
Package Structure/Project			
TrackHandling	Track.cs	This file contains all the properties related to Track.	Already Implemented
	Program.cs	Program.cs is an important class that is the entry point of application and contains business logic of application.	Partially Implemented

2.2 Package: TrackHandling.Tests

Resources

Note: - Under the TrackHandling.Tests contain all test cases for code evaluation, please don't try to alter and edit it.

3. Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**

2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

3. **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

4. **To build your project use command:**
   (TrackManagement /dotnet build)

5. **To launch your application, Run the following command to run the application:**

(TrackManagement /dotnet run)

6. This editor Auto Saves the code.

7. To run the test cases in CMD, Run the following command to test the application: (TrackManagement.Tests/dotnet test --logger "console;verbosity=detailed")

   (You can run this command multiple times to identify the test case status,and refactor code to make maximum test cases passed before final submission).

8. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Track Landing Page) then you need to use CTRL+Shift+B - command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.

9. These are time bound Tracks the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

10. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.