
System Requirements Specification Index

For

School 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

SCHOOL MANAGEMENT APPLICATION SYSTEM

System Requirements Specification

1. BUSINESS-REQUIREMENT:

1.1 PROBLEM STATEMENT:

School Management System: Develop School Management System console application using C#, .NET Core 3.1. Implements all the checks so that there are no errors when School is added, sort from collections and use File/IO. Use proper coding standards with proper class name and method names as per given functionalities.

1.2 FOLLOWING IS THE REQUIREMENT SPECIFICATION:

	School Management Application
	Implement the following Functionalities using a menu driven program.
1	Add at least 5 School details using List generic collection. <user input>
2	Save all School details in excel file.
3	Serialize School List object in JSON format and save it in text file.
4	Fetch all School details from the saved text file.
5	Deserialize the fetched School list object.
6	Show details of School in descending order of name.

2. RESOURCES AVAILABLE:

2.1 PACKAGE: SCHOOLMANAGEMENT

Names	Resource	Remarks	Status
Package Structure/Project			
Assets	Test.txt ResultSheet.xlsx	Test file contains serialized object in JSON format, ResultSheet file will be created automatically after running the application which contains School details.	Already Implemented
	Program.cs	Program.cs is an important class that is the entry point of application.	Partially Implemented
	School.cs	School class is an entity class that consists of field properties of school.	Already Implemented

2.2 PACKAGE: SCHOOLMANAGEMENT.TESTS

Resources

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

3. ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

3.1 Common Constraints:

- Develop application School Management System using Collections, Classes, Exception handling, C# new features File I/O in C#, .NET.
- Implement all Functionalities using a menu driven program.
- Include C# New Features wherever required.

3.2 Exception Handling Specifications:

- Handle Exceptions in the main function by adding try/catch blocks.
- Implement FileNotFoundException, NullReferenceException built-in exception class for all scenarios with separate catch blocks.
- Implement custom exception DuplicateIdException to handle duplicate School id.

4. BUSINESS VALIDATIONS

4.1 Entity Class Specifications:

Create a class School that consists of field properties given below. Implement IComparable<> generic interface to sort the School details by name.

Class name: School

Data Member (Properties):

- int SchoolId
- String SchoolName
- String Address
- long Number of students
- Date DateofInauguration

5. 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. Install the .NET SDK 6.0 by running:
sudo apt install dotnet-sdk-6.0
***If it asks for the password, provide password : pass@word1**

If it asks: Do you want to continue? [Y/n]

Type **Y** and press **Enter**.
 4. On command prompt, cd into your project folder (**cd <Your-Project-folder>**).
 5. To build your project use command:
(SchoolManagement / **dotnet build**)
 6. To launch your application, Run the following command to run the application:
(SchoolManagement / **dotnet run**)
 7. This editor Auto Saves the code.
 8. To run the test cases in CMD, Run the following command to test the application:
(SchoolManagement / **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)
 9. These are time bound assessments 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.
-