System Requirements Specification Index

For

School Admission Management System(MS SQL)

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 ADMISSION MANAGEMENT SYSTEM

System Requirements Specification

1. Project Abstract

1.1 Problem Statement:

School Admission Management System: Develop application using C#, .NET Core 3.1 and RESTful WebApi with MS SQL database. The purpose of this application is to allow the school management to add classrooms,display available classrooms.Also the system allows the students to register and login themselves, students are also able to apply for admission.

1.2 Following is the requirement specifications:

	School Admission Management System
SchoolAdmission Controller	
1	Register a student .
2	Login a student.
3	Add ClassRoom Details.
4	Display available ClassRooms having availability 'yes' .
5	Apply for admission into the ClassRoom.
6	Display list of all admissions.

2. Template Code Structure

2.1 Package: SchoolAdmission

Resource

Names	Resource	Remarks	Status
Package Structure/Project			
Controllers	SchoolAdmissionController	SchoolAdmission controller handles all action methods for the respective api.	Partially Implemented
	appsettings.json	Contains connection string for database	Already Implemented
Properties	launchsettings.json	Contain URL settings for API	Already Implemented

2.2 Package: SchoolAdmission.BusinessLayer

Resource

Names	Resource	Remarks	Status
Package Structure			
Interfaces	Interface directory contains all interfaces for Services class. IAdmissionServices, IClassroomServices, IStudentServices	Interfaces directory contains all interfaces for services class.	Already Implemented
Services, Repository	AdmissionServices, ClassroomServices, StudentServices, IClassroomRepository, IStudentRepository , IAdmissionRepository, ClassroomRepository, AdmissionRepository, StudentRepository	Inside this directory contains all business logic code and CURD Operation Logic method.	Partially Implemented
ViewModels	AdmissionViewModel, ClassroomViewModel, StudentViewModel	Cs file for represent all view entities	Already Implemented

2.3 Package: SchoolAdmission.DataLayer

Resource

Names	Resource	Remarks	Status
Package Structure			
SchoolDbContext	Contain all Database setting	Using this cs file performed all Database related settings operations.	Already Implemented

2.4 Package: SchoolAdmission.Entities

Resources

Names	Resource	Remarks	Status
Package Structure			
Entities Class	Student,Admission,Classroom Class List cs file	Contain all entities property for application	Already Implemented

2.5 Package: SchoolAdmission.Tests

Resources

The SchoolAdmission.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

3. Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created

3.1 SchoolAdmissionController

URL Exposed		Purpose
/student/register-student		Register a Student
Http Method	POST	
Parameter 1	StudentViewModel model	
Return	HTTP Status Code	
/student/login-student		Login a Student
Http Method	POST	
Parameter 1	String EmailId	
Parameter 2	String Password	
Return	HTTP Status Code	
/classroom/add-classroom		Add ClassRoom Details, default ClassRoom availability should be yes
Http Method	POST	
Parameter 1	ClassroomViewModel model	
Return	HTTP Status Code	

<table>
<tr>
<td>/classroom/list-available-classrooms
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Classroom>></td></tr></table></td>
<td>Display available ClassRooms having availability ‘yes’</td>
</tr>
<tr>
<td>/admission/add-admission
<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>AdmissionViewModel model</td></tr><tr><td>Return</td><td>HTTP Status Code</td></tr></table></td>
<td>Apply for admission into the ClassRoom</td>
</tr>
<tr>
<td>/admission/list-all-admissions
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Admission>></td></tr></table></td>
<td>Display list of all admissions</td>
</tr>
</table>

4.Business Validations

4.1 Admission Entity:

- All the value for Admission details: must not be null
- AdmissionId (int) is a key and must not be null
- PersonId (int) must not be null
- ClassroomId (int) must not be null
- NumberOfStudents (int) must not be null

4.2 Classroom Entity:

- All the value for Classroom details: must not be null
- ClassroomId (int) is a key and must not be null
- StudentCapacity (int) must not be null
- ClassType(string) must not be null
- Availability (string) must not be null
- PerClassFee (int) must not be null

4.3 Person Entity:

- All the value for Student details: must not be null
- PersonId (int) is a key and must not be null
- PersonName (string) must not be null
- Password(string) must not be null
- EmailId (string) must not be null
- PersonCity (string) must not be null

4.4 Common Constraints:

- For all receiving Url parameter, validation check must be done and must throw custom exception if data is invalid
- Must not go and touch the test resources, as they will be used for Auto-Evaluation.

5. CONSIDERATIONS

1. Your code will also be evaluated for code quality, naming conventions, readability etc.

2. Make sure you do not modify existing class and method names and their signatures, else it would severely affect the final result.

3. Make sure you do not add any new class or methods, else it would severely affect the final result.

4. Make sure you do not modify any test cases, else it would severely affect final result

6. 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 connect SQL server from terminal:**
   (SchoolAdmission/sqlcmd -S localhost -U sa -P pass@word1)
   - **To create database from terminal -**
     1. **Create Database SchoolAdmission_Db**
     2. **Go**

5. **Steps to Apply Migration(Code first approach):**
   - **Press Ctrl+C to get back to command prompt**
   - **Run following command to apply migration-**
     (SchoolAdmission /dotnet-ef database update)

6. **To check whether migrations are applied from terminal:**
   (SchoolAdmission /sqlcmd -S localhost -U sa -P pass@word1)

   ```
   1> Use SchoolAdmission_Db
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go
   ```

7. **To build your project use command:**
   (SchoolAdmission /dotnet build)

8. **To launch your application, Run the following command to run the application:**
   (SchoolAdmission/dotnet run)

9. **This editor Auto Saves the code.**

10. **To run the test cases in CMD, Run the following command to test the application:**
    (SchoolAdmission.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).

11. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment 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.**

12. **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.**

13. **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.**
