Project Description

GENERAL INSTRUCTIONS:

Please carefully read the below instructions

The objective of this assessment is to check your ability to complete a project as per the provided "Project Design".

You are expected to –

1. Write the source code for the classes, methods and packages EXACTLY as mentioned in the "Project Design" section.
2. Ensure that the names of the packages, classes, methods and variables EXACTLY MATCH with the names specified in the "Project Design" section.
3. Understand the project requirements and ACCORDINGLY WRITE the code and logic in the classes and methods so as to meet all given requirements.

Creating the project and testing it –

1. Download the Project Template from the Assessment Portal and extract it on your local machine
2. Extract the Downloaded ZIP file. It imports the Visual Studio Solution containing the Project(s) and the related source files. You need not to create a separate application on your location machine, instead you are supposed to work on the Solution / Project / Source downloaded / extracted from the downloaded ZIP File.
3. Define the methods as mentioned in the Project Design Below.
4. Build and test your application locally before uploading it for evaluation.
5. Before uploading, make sure to delete all the PDB and DLL files from the Project directory. For this delete the content from BIN and OBJ directories.
6. Close the IDE, Create a .ZIP file by right clicking on the Root directory of the Project and Send to Compressed Folder option.
7. To upload, select the File, and click on Upload

NOTE that –

8. The assessment engine will create objects and invoke methods as per the project design, and while doing so, it will use your packages, classes and methods. If your namespaces, classes and methods have a name mismatch or method prototype mismatch w.r.t the expected "Project Design", the tool will show it as an ERROR. If your packages OR classes OR methods OR Script match as per the names but do not perform the expected functionality, the tool will show it as a FAILURE.
9. Unless specified in the Project Design, DO NOT use System. Environment. Exit (0) anywhere in your code. Using System. Environment. Exit (0) in your project code will cause the CPC test engine to exit and it will not be able to run all test-cases.

Asset Management System

Project Design

The XYC Inc. is a leading IT Organization. The employees are provided with the assets like, Computer Machines, Laptops etc. The Organization want to replace their existing manual asset tracking system with a robust Software application. As a developer you are required to design and develop a software solution that handles the asset tracking features that includes, Managing assets, Tagging assets and d-tagging the assets.

A. Database Design

1. **Create a database with the name, “AssetManagement” using the MS SQL Server on your machine locally.**
2. **Create a table, “Employee” in the above database with the following specifications.**

Column Name	**Data Type**	**Constrains / Specification**
EmployeeID	**VARCHAR(10)**	**PRIMARY KEY**
EmployeeName	**VARCHAR(20)**	Not Null
DateOfJoining	**DATE**	Not Null
Location	**VARCHAR(20)**	Not Null
Email	**VARCHAR(30)**	Not Null
Phone	**VARCHAR(10)**	Not Null.

3. **Insert the Following records into Employee Table.**

EmployeeID	**EmployeeName**	**DateOfJoining**	**Location**	**Email**	**Phone**
JO123456	John	01/01/2011	HYDERABAD	john@xyz.com	123456789
SM234567	Smith	15/11/2011	CHENNAI	smith@xyz.com	123456789

4. **Create a Table, “Asset” in the “AssetManagement” database with the following specifications.**

Column Name	**Data Type**	**Constrains / Specification**
ASSETID	**INT**	**IDENTITY(1000,1) PRIMARY KEY**
ASSETTYPE	**VARCHAR(20)**	LAPTOP / DESKTOP / VOIP / DATACARD
SERIALNO	**VARCHAR(20)**	Not Null
PROCUREMENTDATE	**DATE**	Not Null
TAGGINGSTATUS	**VARCHAR(9)**	Default “FREE POOL”

5. **Create a Table, “AssetTagging” in the “AssetManagement” database with the following specifications.**

Column Name	**Data Type**	**Constrains / Specification**
EMPLOYEEID	**VARCHAR(10)**	FOREIGN KEY EMPLOYEE : EMPLOYEEID
ASSETTID	**INT**	FOREIGN KEY ASSET : ASSETID
TAGGINGDATE	**DATE**	Not Null
RELEASEDATE	**DATE**	Can take null values. Null value indicates the asset is not released. If the release date is mentioned then the asset is in free pool.

B. Application Design

1. Define the below properties in the Employee Class

   ✓ EmployeeID as string

   ✓ EmployeeName as string

   ✓ DateofJoining as DateTime

   ✓ Email as string

- ✓ Location as string
- ✓ Phone as long

2. Define the below properties in “Asset” as per the following specifications.
   - ✓ AssetID as int
   - ✓ AssetType as string
   - ✓ SerialNo as string
   - ✓ ProcurementDate as DateTime
   - ✓ TaggingStatus as string.
3. Define the below properties in “AssetTagging” as per the following specifications.
   - ✓ AssetID as int
   - ✓ EmployeeID as string
   - ✓ TaggingDate as DateTime
   - ✓ ReleaseDate as DateTime
4. Define the below methods in AssetManagement Class as per the below specifications

Class	Method	Description
Asset Tracking	public bool AddAsset(Asset obj)	The AddAsset method should be defined as per the below specifications. ✓ The method should take the reference of the Asset object containing the asset details to be added into the database. ✓ The Serial Number of the asset should be generated as per the below procedure. **Step 1:** Extract first two characters of the Asset type. **Step 2:** Generate a random number between 1 and 1000 and append it to the first two characters of the asset type. The asset ID should be auto generated on the database. After generating the Serial number of the asset, insert these details into the database. If the asset details are added successfully, return “true” else return “false”. **Note : The method should return false if the object reference is null**
	public bool ModifyAsset(Asset obj)	The ModifyAsset method should modify the asset details based on the asset ID. Follow the below procedure to update the asset information. ✓ If the asset details are existing on the database, the Serial Number, Asset Type and procurement date should be updated based on the Asset ID. ✓ Upon successful update, the method should return “true”. ✓ If the update fails OR the asset reference is pointing to null, must return “false”
	public bool TagAsset(AssetTagging obj)	This method should tag an asset to the defined employee. Follow the below procedure to tag the asset. ✓ To tag an asset, insert the asset tagging details into the AssetTagging table of the Asset management database. ✓ While inserting, check if the asset is already tagged to an employee. Same asset can’t tagged to multiple employees. For tagging any asset the asset must be in free pool. If the tagging status of the asset is “Free Pool”, then insert the record into AssetTagging table of the asset management database. Also change the tagging status of the asset in the asset table to “Tagged”. Upon successful tagging, return “true”. ✓ If either the asset is already tagged OR asset tagging reference is pointing to null, return “false”
	Public bool DeTagAsset(int intAssetId)	This method should De-tag a tagged asset. The method should take the asset ID as an argument, and make the below modifications in the corresponding tables.

Table	**Action required**
AssetTagging	Update the Release date of the associated Asset ID to the current system date
Asset	Update the Tagging Status of the associated Asset to Free Pool