MTA Software Development Worksheet

Students who attended the 93-361 course might attempt the below worksheet. As discussed in class this does not feature all the technologies covered during the course, however is an attempt to create an architecture that connects various technologies learnt. Particularly the creation of an OO model, a database table with a number of stored procedures and a variety of visual studio project types:

- A database that holds a table of products and a number of stored procedures to select, insert, and update records.
- A web service will be used to receive product creation requests and insert them into a database table through stored procedures.
- A console application to input a list of computer products.
- A windows application to input a list of car products.
- A windows service to randomly update one of the products every five minutes.
- A web application to display the list of products by type.

A challenge is also provided at the end of the worksheet to enhance your architecture by further research.

Database

- Install an MS SQL Server Express edition
- Create a table to store a list of products and a last updated date.
    - tblProduct
        - ProductName varchar(50)
        - ProductType int
        - LastUpdatedDate datetime
- Create a stored procedure to insert a product
    - USPInsProduct(@ProductName, @ProductType)
- Create a stored procedure to update a product
    - USPUpdProductDate(@ProductName)
- Create a stored procedure to retrieve a list of products
    - USPGetProduct(@ProductType) returns ProductName and LastUpdatedDate
- Hint: Use the getdate() function

Web Service

- Install IIS with ASP.NET support enabled, create a virtual folder for your service in the default website.
- Create a data model as an Abstract class ProductDM:
    - two internal members ProductName and ProductType
    - a constructor that sets both data members
- Create a derived class ComputerDM:
    - inherits the ProductDM
    - a constructor that calls its base and hard codes ProductType to 1
- Create a derived class CarDM:
    - inherits the ProductDM
    - a constructor that calls its base and hard codes ProductType to 2
- Create a data layer as a static class ProductDL with two static methods:
    - Insert(ProductDM) which calls USPInsProduct
    - Update(ProductName) which calls USPUpdProductDate

- Create a web service ProductService.asmx with four web methods:
    - CreateComputer(ComputerName) and CreateCar (CarName) which creates and populates an instance of the respective class and subsequently send it to ProductDL.Insert(ProductDM) method
    - UpdateProduct(ProductName) which calls ProductDL.Update(ProductName) method directly
    - GetProducts(ProductType) which calls USPGetProduct() and returns an XML file with the list of products

Console Application

- ComputerApplication.exe with a text-based menu:
    - Option 1: Create Computer – which takes the user input and passes it to the CreateComputer() web service method
    - Option 2: Exit

Windows Application

- CarApplication.exe with a WinForm that contains:
    - One ListBox: lstCar
    - One TextBox: txtCar
    - One Button: Add – which takes non-empty inputs and adds them to the list
    - One Button: Upload – which loops the list and calls the CreateCar() web service method per list item and removes each item on success

Windows Service

- Every 5 minutes:
    - randomises a product type (1 or 2) and retrieves the list of products through the web service method GetProducts(ProductType)
    - randomly selects a product from the retrieved list and calls the web service method Update(ProductName)
- Hint: Use while(true) and Thread.Sleep()

Web Application

- On IIS, in the default website create another virtual folder for your site.
- Create a page Default.aspx
    - Include two image links that represent computers and cars and link to Products.aspx by passing the correct ProductType as a value through a QueryString parameter
- Create a page Products.aspx
    - Retrieve the QueryString value "type"
    - Call the web service method GetProducts(ProductType) to retrieve the list of products and display them in a GridView
    - Display an error message if: the type parameter is not supplied, or the type is not a valid number, or the list of products is empty
    - Add a link to the home page
- Hint: Load XML using DataSet.ReadXML() for GridView.DataSource purposes.

Challenge!

- Upgrade the tblProduct and related stored procedures to include additional fields.
- Refactor the ProductDM and ProductDL classes to reflect the addition of such fields.
- Upgrade the console and windows application to retrieve the list of respective products and enable the modification of existing products as well.
- Add a van data model and distinguish data members between the car and van models through inheritance.
- Add a tblProductType(ProductTypeId, ProductType) table to contain the reference list of product types and add a relationship to the product table.
- Add an automatic refresh on Products.aspx web page to reload the page every 6 minutes either through META tags or AJAX.