IMPORTANT INSTRUCTIONS

- This is exam **VERSION A: Northern Star Recruitment**
- Ensure your exam preparation is uploaded before you begin the exam.
- Create a folder with your initials, followed by **-exam** and put all your work done during the exam in that folder. So if your name is **Jane Doe**, your folder will be named **JD-exam**. At the end of the exam, run Build clean and zip this folder (using .zip or .7z) and upload it to Blackboard in the exam submission section.
- **Questions for the exam appear after the exam preparation.**

IMPORTANT: VB.Net code for the following scenario must be prepared and uploaded to Blackboard prior to the exam otherwise you will not have sufficient time to complete the exam.

Exam Preparation for Northern Star Recruitment [10 Marks]

Northern Star Recuritment is a company specializing in helping companies shortlist potential employees based on their experience in well known tech companies. They want your help in creating an ASP.Net MVC5 based website to handle potential employees. They give you a JSON file[1].

Business Logic

They want you to do the following:

- Create a VB.Net DLL to handle the business logic of the ASP.Net application. The VB.Net DLL must throw exceptions on invalid data. The exception messages thrown by the DLL must show up on the ASP.Net website when invalid data is entered after it was seeded. Your website may, however, ignore all exception messages thrown while the database is being seeded and silently drop the invalid records.
- Create an ASP.Net MVC5 website that uses the above DLL and creates a LocalDB database using CodeFirst method. This website should do the following:
  - The database name should be named after you, so for example if your name is Jane Doe - the database name should be: **JaneDoeStore**
  - When the website starts up it drops the existing database and reads data from the JSON file to seed the LocalDB database. Put the JSON file in the **App_Data** folder and read it from there.
  - Create MVC controller and views to handle the following Create and Display Details scenario for a Person:
    1. The database should only store the following information for each person: *Name, Email, Companies*, and *Specialization*. This means when seeding the database, only populate these **FOUR 4** fields.
    2. Name: Should consist of **alphabets and spaces only** and must be a **non-empty string**.
    3. Email: Usual rules for validating emails apply, only check that an email has @ and does not start or end with a .
    4. Companies: must **be non-empty** and must have expertise in *Microsoft* or *Apple* or both at some time in their career.
    5. Specialization: can be zero or more, no need to check values.

Security

- Create an Admin Role and admin User with password *123456*. Admin can create new records and see **all** the details of the Person. When admin creates a new record, you do not need to show a drop down list of specializations, but if you do then let the user leave it with a default value or select a value from the drop down list if you provide one for specializations, however, the new record must follow the rules of valid companies, so when creating a new record you must follow these rules:
  - do not display one of the companies as the default drop down value, instead provide a help message indicating that one of the valid company values must be selected
  - force the admin user to select one company from that list

[1] dps916vba544fe_w14.zip is available in the exam section on Moodle

  - do not let the admin user select the default drop down help message

- Non-admin users can only see Name and Email details of a Person. You can show this on the Home Details Page of the website.
- Newly registered users automatically get added into the admin role.
- Routes:

```
/Home:  Admin and non-admin users can access.
/Home/Details:  Admin and non-admin users can access.
/Person/Index: Only Admin can access.
/Person/Create: Only Admin can access
/Person/Details: Only Admin can access
```

Website

- Show your Full Name and Student Id in the About page of the website
- The website interface lets the user create, and display records that meet the above business logic.
- Lightly edit the default ASP.Net MVC 5 settings of the website to make it look and feel like its purpose of business. Keep changes to the default ASP.Net MVC 5 View minimal yet sufficient to convey the purpose of the website.
- When displaying the records for a person, show all the details for that person as appropriate, namely show *Email, Companies*, and *Specialization* when showing the person's *Name* to the admin but only *Email* and *Name* to non-Admin users.

Unit Tests

- Write unit tests for your DLL to demonstrate that it throws exceptions when invalid data is entered, these exception messages thrown by the library appear on the ASP.Net MVC 5 website too.
- Write ASP.Net MVC unit tests to demonstrate the following works by calling the Person controller of your website:
  - prove a valid record matching the business criteria from the JSON file actually shows up in the database after being seeded. This means the unit test checks that the JSON file actually seeds correctly.
  - prove that create new Person works, so demonstrate that valid records get added to the database and invalid records throw exceptions and are not added to the database. The exception messages thrown by the DLL are matched in the Unit Tests.

Architecture

There is no design imposed on how to create and design the website, but keep the following in mind when writing your code:

- Your application mostly uses VB.Net, optionally with some C# code, to create an ASP.Net MVC 5 website.
- You may reuse code taken from your assignments, or code released under an open source licence that is publicly available on the Internet, however, **avoid** using code written by another student as multiple versions of the final exam exist. Each version has its own requirement criteria that are different from other versions.
- Keep your design flexible so it can respond to changes quickly, for example: during the online exam you will be expected to respond to one or more of the following changes in business requirements:
  - like seeding the database only with values based on some additional validation criteria, for example: accepting employees from other companies.
  - Create and Details View changes for the change in the seeding scenarios described above, or after seeding all new records might impose additional requirements on the companies, for example all new people must have their companies previously held position categorized as: Staff, Management, and Director.
  - Unit tests to demonstrate that the new requirements actually work.
  - Adding another role to permit actions between anonymous users and admin users.