Pass Task 2.2 Entity Persistence and ORM

**Task 4. Design justification:**



*Figure 1. Architecture diagram.*

**Database Layer:** This is a MSSQL Server database, effectively SQL Express server which contain the data table MyUser.

**Data Access Layer:** This layer handles all the works related to accessing data. It contains an ORM (effectively Entity Framework) as a mapping between MyUser object and the MyUser table inside the database. It also contains MyUserDao, a data access object which provide the CRUD methods. This DAO exposes an interface to public named: IMyUserDAO. By using the interface to implement the DAO, it will be very easy to change to different implementation of the DAO. For example, when the database table grow very large and the Entity Framework shows inefficient in accessing data (1), we can easily switch the implementation of a DAO into a more effective ORM such as Dapper or even using basic ADO.NET.

**Business Logic Layer:** This layer contains 1 or more service classes which maintain business logics and provide business data to the upper level in the format of data transfer object (DTO). This layer will handle tasks related to the business rules such as: business rule data validation, data processing and manipulating, data mapping from entity object into DTO and vise-versa. This layer use the DAO interface provided by the Data Access Layer to communicate with the persistence storage.

**Presentation Layer:** This layer is also called application layer. It handles all interactions with the users as well as responsible for presenting the data. It can be a console app, mobile app, desktop program or web pages. In my implementation, this is a web application done by using the ASP.NET MVC framework. MyUserController is a Controller of the MVC framework which handles user interaction, load the business data from the service layer and pass it on to the Razor view engine. This engine use the .cshtml files to do the render of html pages. In this case, we have 3 pages: Index.cshtml for user data listing, Create.cshtml for creating new user, and edit.cshtml for editing user information.

**Common Shared Library:** This is a shared library between all layers in this architecture which contains DTO, helper classes, logging tools that can be used by all layers. Data mapping could be considered to put into this layer. However, this library doesn’t have any knowledge about Entity Object, the mapping of Entity Object to DTO cannot be put here and have to put into Business Logic Layer.

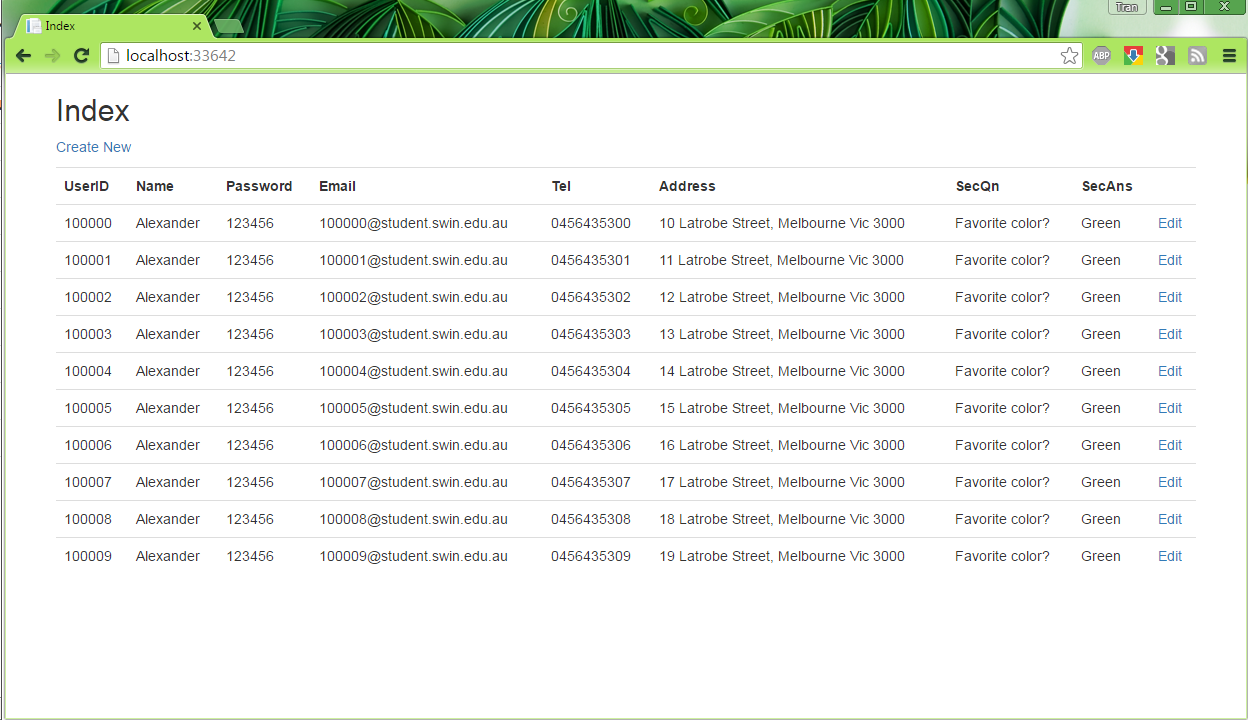
**Task 3. Test cases:**

The following test cases have been taken out to make sure that the whole application works as expected:

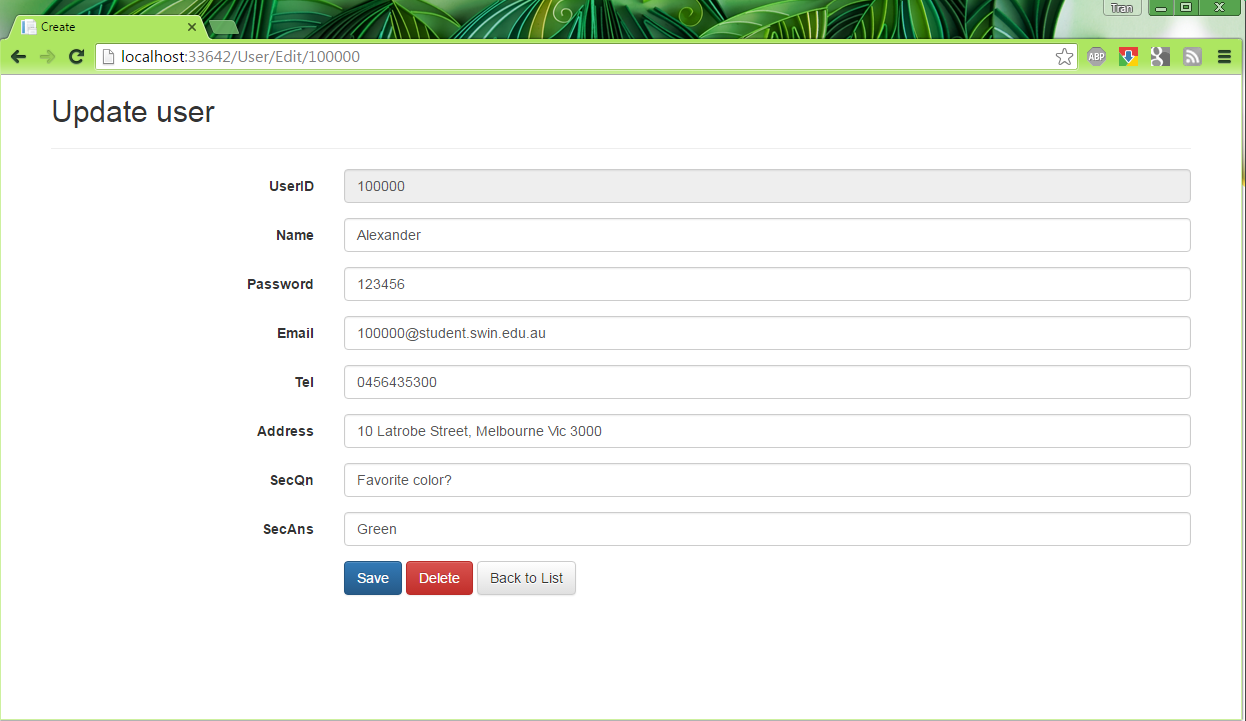
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case Name | Step | Expected Output | Result |
| 01 | Get all users | View the page /User | A list of 10 users is displayed on the screen | Pass |
| 02 | Get one user | Click to the Edit link of one user in the list | Display the detailed information of that user in a form | Pass |
| 03 | Create an user | - Click to the Create New link.  - Enter new user information  - Click Create | The new user is added to the list. The list is now showing 11 users | Pass |
| 04 | Update existing user | - Click to the Edit link of one user in the list  - Change one or more detail of the user using the form  - Click Save | The updated information of that user should be displayed on the list | Pass |
| 05 | Delete existing user | - Click to the Edit link of one user in the list  - Click Delete | The user is removed from the list | Pass |

*Table 1. Test cases and test results*

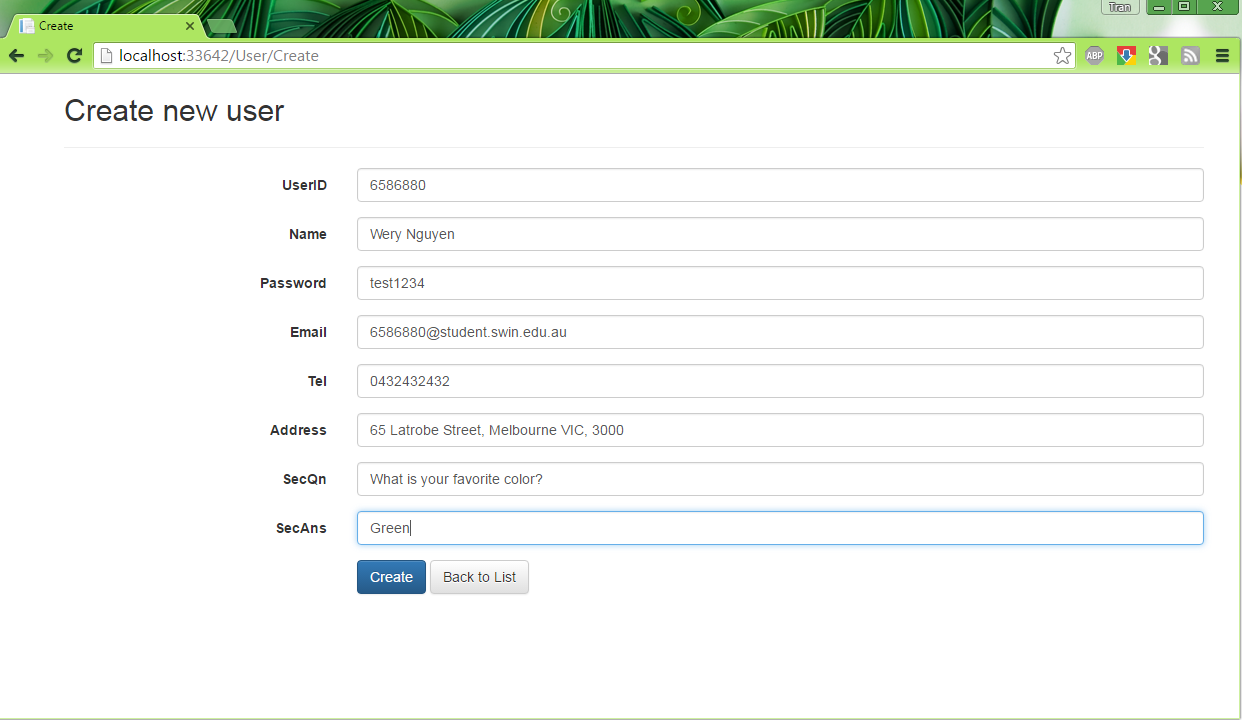
Test case screen shots and success screen shots:



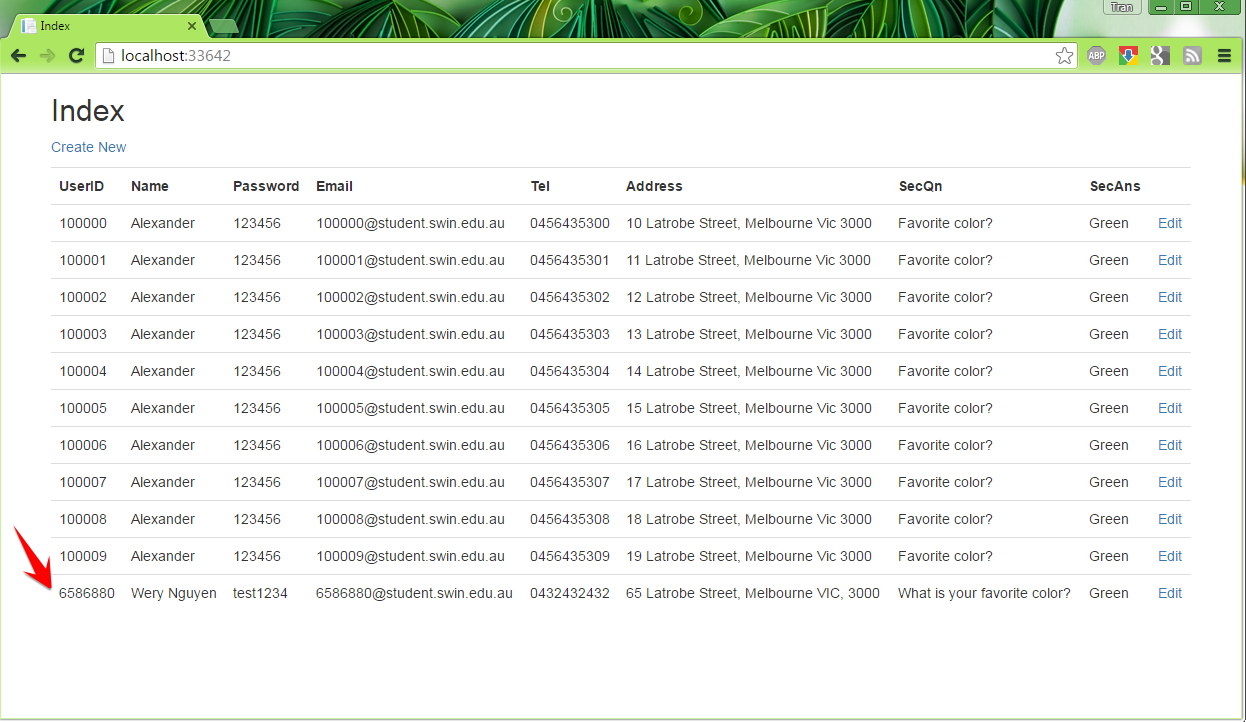
Test ID 01 - Show all users



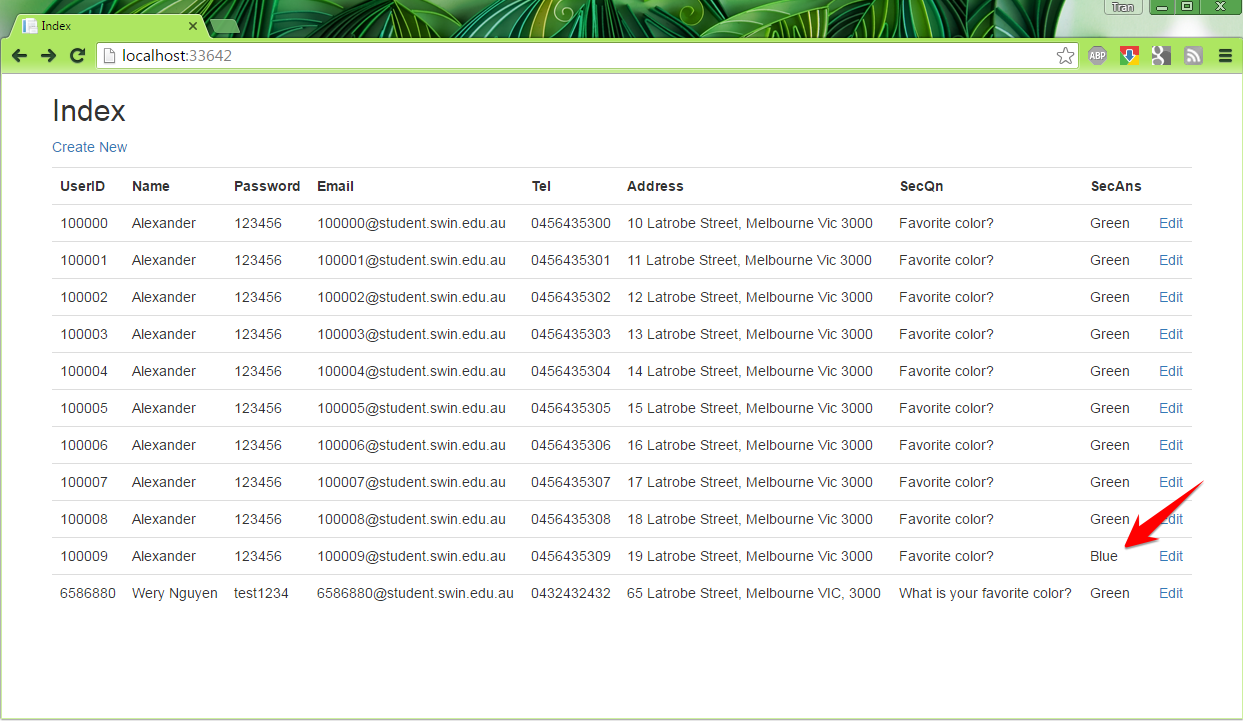
Test ID 02 - Show one user



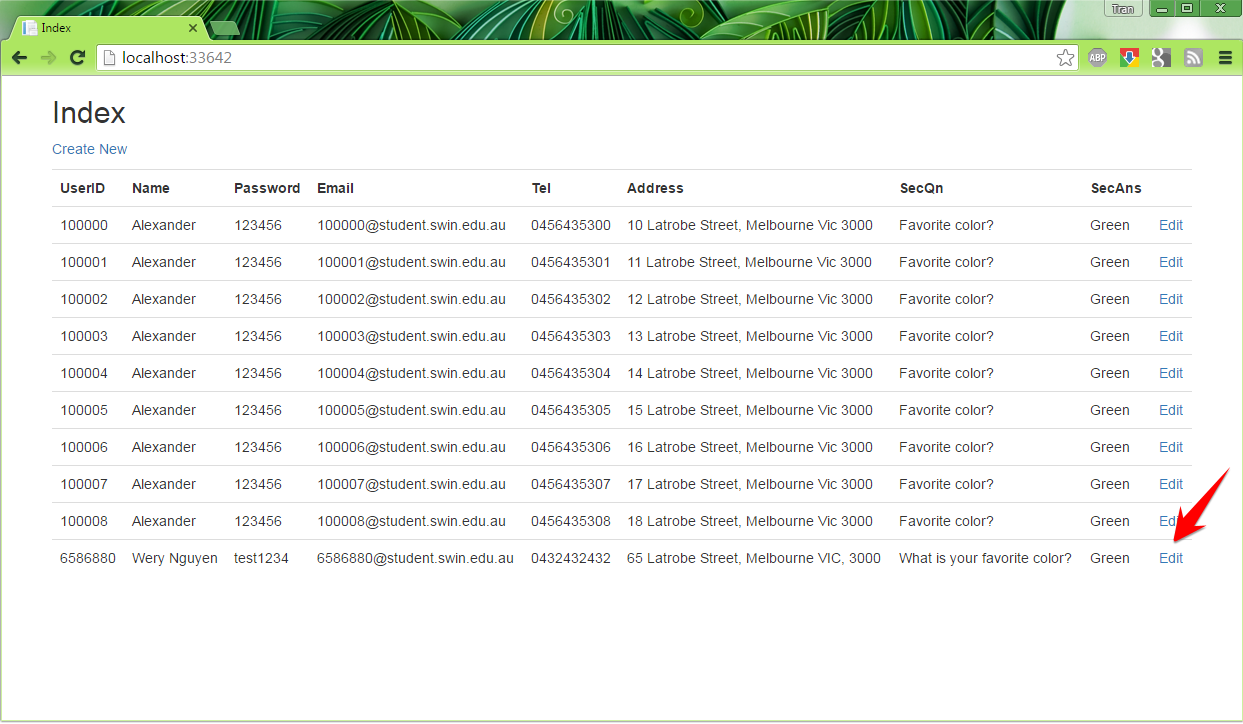
Test ID 03 – Create new user screen.



Test ID 03 - Create one user success



Test 04 - Edit one user detail success



Test 05 - Delete one user success

**References:**

1. ORM performance comparation: https://github.com/StackExchange/dapper-dot-net