National College of Ireland

BSc in Computing

2014/2015

Paul O’Beirne

X10203800

pobeirne@Outlook.ie

GradFolio

Technical Report



**Declaration Cover Sheet for Project Submission**

**SECTION 1**

|  |
| --- |
| **Name: Paul O’Beirne** |
| **Student ID: x10203800** |
| **Supervisor: Mikael Timofeev** |

**SECTION 2 Confirmation of Authorship**

*The acceptance of your work is subject to your signature on the following declaration:*

I confirm that I have read the College statement on plagiarism (summarized overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: 30/07/15

**Table of Contents**

[Executive Summary 4](#_Toc425985764)

[1 Introduction 5](#_Toc425985765)

[1.1 Background 5](#_Toc425985766)

[1.2 Aims 6](#_Toc425985767)

[1.3 Technologies 6](#_Toc425985768)

[1.4 Structure 7](#_Toc425985769)

[2 System 8](#_Toc425985770)

[2.1 Requirements 8](#_Toc425985771)

[2.1.1 Functional requirements 9](#_Toc425985772)

[2.1.2 Data requirements 23](#_Toc425985773)

[2.1.3 User requirements 24](#_Toc425985774)

[2.1.4 Environmental requirements 24](#_Toc425985775)

[2.2 Usability requirements 24](#_Toc425985776)

[2.3 Design and Architecture 25](#_Toc425985777)

[2.4 Implementation 28](#_Toc425985778)

[2.5 Testing & Performance 49](#_Toc425985779)

[2.6 Graphical User Interface (GUI) Layout 51](#_Toc425985780)

[3 Conclusions 63](#_Toc425985781)

[4 Further development or research 64](#_Toc425985782)

[5 References 65](#_Toc425985783)

# Executive Summary

The main objective of GradFolio to is to provide a web application capable of generating a curriculum vitae and portfolio site for the user. GradFolio is aimed at providing a quick and simple solution for current students and or recent graduates to showcase themselves and works through a portfolio site and allow the creation of custom curriculum vitae’s fit for varying potential job application purposes. The web application will allow the user overcome any potential barriers such as lack of knowledge in web technologies.

Website link: http://gradfolio.azurewebsites.net/

# Introduction

## Background

As a current and hopefully soon to be graduate, I will face many new challenges in entering the work environment and looking to begin a new career path. While there is many opportunities in the current economic climate for someone looking for a role in the information sector, the application processes is highly competitive and will remain foreseeable future. This is not something unique to this specific area but more a common trait across many professional fields.

With the digital age in full swing, many of the routes to finding work and applying for jobs has moved online thus professionals also need a digital footprint to showcase themselves. Sites like LinkedIn provide a very good solution for any professional to show their work experience and education credentials to potential employers.

In applying for different jobs I began to notice that I was constantly adjusting and tailoring my curriculum vitae to each job specification. From my own personal experience LinkedIn was a great resource but often fell short in catering to this in any easy way. I also noticed during interviews that employers would view my LinkedIn profile which had more detail information then the curriculum vitae provided but still feel short in terms of showcasing a satisfactory level of my body of work.

With potential employers asking if I had any alternative portfolio of work they can look at beyond that described on LinkedIn. I realised I didn’t have anything beyond a GitHub account holding some projects from college but nothing else I could actually show them. This was not something unique to myself I also observed many of my colleges where also in the same position as myself. For most students and graduates alike it is highly important to convey to employers what you have learnt from your course and provide examples of work you done.

## Aims

GradFolio main aim is to help students or recent graduates gain a competitive edge over others by providing quick and easy solution in which the user can create custom curriculum vitae’s and a portfolio site. The solution will help the user overcome potential barriers which such as time constraints and lack of resources or knowledge in web technologies to develop their own portfolio site.

## Technologies

* **Azure Cloud Services:** provides cloud hosting for the web application and database.
* **Visual Studio:** is an integrated development environment (IDE).
* **ASP.NET MVC 5:** is a web application framework that implements the model–view–controller pattern.
* **Entity Framework 6:** is an object-relational mapping framework for ADO.NET.
* **Ninject:** provides a lightweight dependency injection framework for .NET applications.
* **Twitter Bootstrap:** is a front end framework which provides a collection of HTML and CSS tools used for creating the application interfaces.
* **NLog:** is a logging platform for .NET used to log errors and exceptions to a database or file.
* **MvcRazorToPdf:** used to render razor views to pdf documents in the browser.

## Structure

**Introduction:** provides a summary of the project idea, background and aims.

**System:** describes the project requirements, Design, engineering of the software involved, testing plans, GUI layout, testing and evaluation.

**Conclusions**: provides a summation of the outcome of the project and what was learnt over the course of the development process.

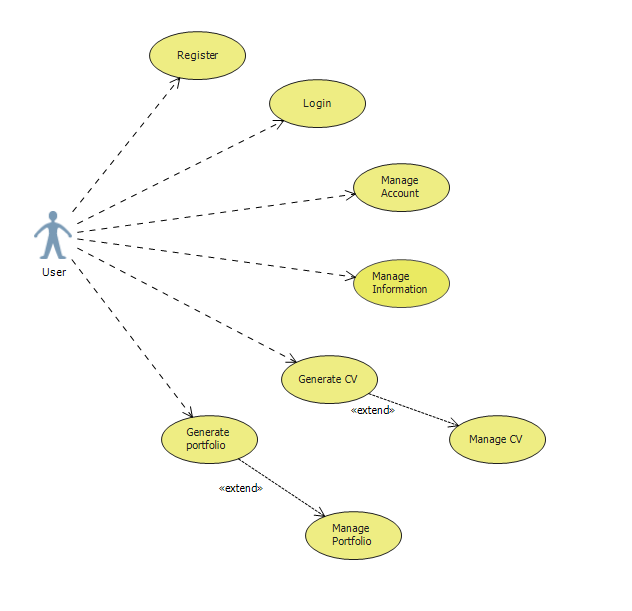
**Further Development or research:** discusses potential additional development features and implementations that could be added to enhance the application.

**References**: provides links to learning, technology resources or any external resources used in the document or development process.

**Appendix**: contains any additional resources code, diagrams or further details.

# System

## Requirements



1. User should be able to register to use the application
2. User should be able to login after registering
3. User should be able to manage their account i.e. update profile
4. User should be able to manage their information
5. User should be able to generate a curriculum vitae
6. User should be able to generate a portfolio

### Functional requirements

**Requirement 1 <Register>**

**Description & Priority**

This function will allow a new user to register for the application.

**Use Case**

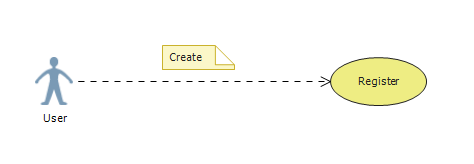
**Scope**

The scope of this use case is to allow the user to register a new account as a new user for the application.

**Description**

This use case describes the user interaction with the “Register” functionality

**Use Case Diagram**



**Flow Description**

**Precondition**

* None

**Activation**

* This use case starts when the user navigates to the registration page

**Main flow**

1. User enters registration credentials.
2. System authenticates user information and adds user to database.
3. System indicates that registration is complete.

**Alternate flow**

* 2a. Failed authentication
* 2a1. System notifies user of failed authentication and re-requests registration credentials.
* 2a2. User re-enters registration credentials.
* 2a3. System authenticates user.

**Termination**

* The System indicates that registration is complete

**Post condition**

* The system goes into a wait state

**Requirement 2 <Login>**

**Description & Priority**

This function will allow a user to log into the application.

**Use Case**

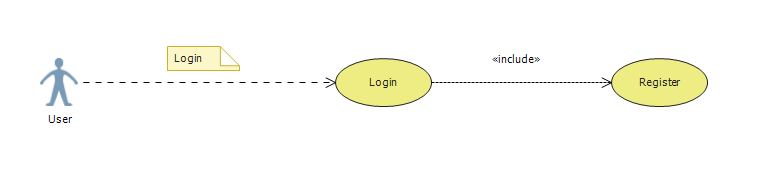
**Scope**

The scope of this use case is to allow the user to log into the application using their account information.

**Description**

This use case describes the user interaction with the “Login” functionality

**Use Case Diagram**



**Flow Description**

**Precondition**

* The user has an account

**Activation**

* This use case starts when the user navigates to the login page

**Main flow**

1. User enters login credentials.
2. System authenticates user information and queries the database.
3. System indicates that login is successful.
4. System redirects the user to the portal overview.

**Alternate flow**

* 2a. Failed authentication
* 2a1. System notifies user of failed authentication and re-requests login credentials.
* 2a2. User re-enters login credentials.
* 2a3. System authenticates user.

**Termination**

* The System redirects the user to the portal overview

**Post condition**

* The system goes into a wait state

**Requirement 3 <Manage Account>**

**Description & Priority**

This function will allow a user to manage their account information.

**Use Case**

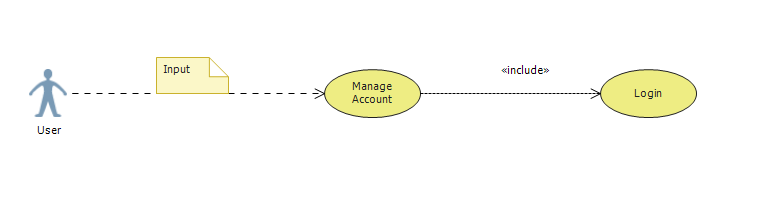
**Scope**

The scope of this use case is to allow the user to manage their account information. I.e. update profile, password and delete account

**Description**

This use case describes the user interaction with the “Manage Account” functionality

**Use Case Diagram**



**Flow Description**

**Precondition**

* The user has an account
* The user has logged in

**Activation**

* This use case starts when the user navigates to the account or profile view

**Main flow**

1. User enters in new information.
2. System authenticates user & information and updates the database.
3. System indicates that changes are successful.
4. System redirects the user back to the page view (Account/Profile).

**Alternate flow**

* 2a. Failed authentication
* 2a1. System notifies user of failed authentication and re-requests login credentials.
* 2a2. User re-enters login credentials.
* 2a3. System authenticates user.
* 2b. Failed information authentication
* 2b1. System notifies user of failed authentication and re-requests correct information.
* 2b2. User re-enters information.
* 2b3. System authenticates user information and updates database.

**Termination**

* The System redirects the user to the page view

**Post condition**

* The system goes into a wait state

**Requirement 4 <Manage Information>**

**Description & Priority**

This function will allow a user to manage their information.

**Use Case**

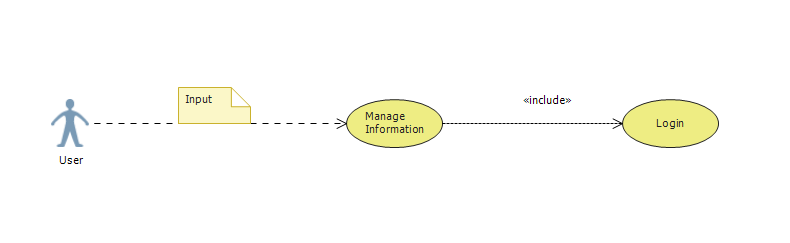
**Scope**

The scope of this use case is to allow the user to manage their information. I.e. Experience, Education, Awards, Interests etc.

**Description**

This use case describes the user interaction with the “Manage Information” functionality

**Use Case Diagram**



**Flow Description**

**Precondition**

* The user has an account
* The user has logged in

**Activation**

* This use case starts when the user navigates to the account info view (Experience, Education, Awards, and Interests etc.).

**Main flow**

1. User enters in new information.
2. System authenticates user & information and updates the database.
3. System indicates that changes are successful.
4. System redirects the user back to the page view (Experience, Education, Awards, and Interests etc.).

**Alternate flow**

* 2a. Failed authentication
* 2a1. System notifies user of failed authentication and re-requests login credentials.
* 2a2. User re-enters login credentials.
* 2a3. System authenticates user.
* 2b. Failed information authentication
* 2b1. System notifies user of failed authentication and re-requests correct information.
* 2b2. User re-enters information.
* 2b3. System authenticates user information and updates database.

**Termination**

* The System redirects the user to the page view

**Post condition**

* The system goes into a wait state

**Requirement 5 <Generate CV>**

**Description & Priority**

This function will allow a user to generate and modify a CV.

**Use Case**

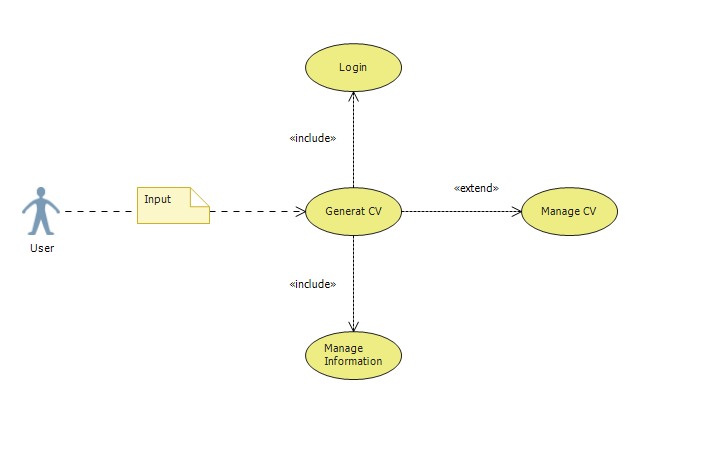
**Scope**

The scope of this use case is to allow the user to generate a CV by selecting a template to use and which information should be shown and modify a CV if the user wishes to.

**Description**

This use case describes the user interaction with the “Generate CV” functionality

**Use Case Diagram**



**Flow Description**

**Precondition**

* The user has an account
* The user has logged in
* The user has created sufficient amounts of information (Experience, Education, Awards, and Interests etc.).

**Activation**

* This use case starts when the user navigates to the CV overview page

**Main flow**

1. User chooses a template to use and clicks the generate button.
2. System authenticates user & information and updates the database.
3. System indicates that changes are successful.
4. System redirects the user back to the CV overview page showing the new CV addition.
5. User chooses to modify existing CV.
6. System authenticates user & information and updates the database.
7. System indicates that changes are successful.
8. System redirects the user back to the CV overview page
9. User chooses to view the CV in full.
10. System authenticates user and retrieves CV information from the database.
11. System opens a new window and displays the CV to the user.

**Alternate flow**

* 2a. Failed authentication
* 2a1. System notifies user of failed authentication and re-requests login credentials or valid information.
* 2a2. User re-enters login credentials or valid information.
* 2a3. System authenticates user or information.
* 6a. Failed authentication
* 6a1. System notifies user of failed authentication and re-requests login credentials or valid information.
* 6a2. User re-enters login credentials or valid information.
* 6a3. System authenticates user or information.
* 9a. Failed authentication
* 9a1. System notifies user of failed authentication and re-requests login credentials or valid information.
* 9a2. User re-enters login credentials or valid information.
* 9a3. System authenticates user or information.

**Termination**

* The System redirects the user to the CV overview page

**Post condition**

* The system goes into a wait state

**Requirement 6 <Generate Portfolio>**

**Description & Priority**

This function will allow a user to generate and modify a portfolio.

**Use Case**

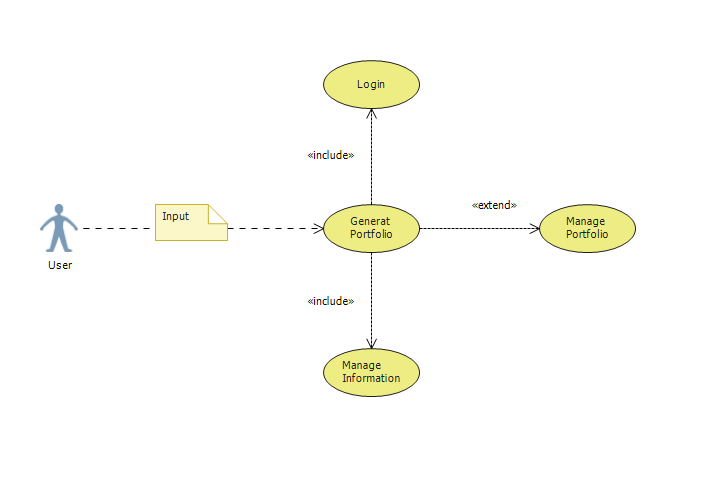
**Scope**

The scope of this use case is to allow the user to generate a portfolio by selecting a template to use and which information should be shown and modify a portfolio if the user wishes to.

**Description**

This use case describes the user interaction with the “Generate portfolio” functionality

**Use Case Diagram**



**Flow Description**

**Precondition**

* The user has an account
* The user has logged in
* The user has created sufficient amounts of information (Experience, Education, Awards, and Interests etc.).

**Activation**

* This use case starts when the user navigates to the portfolio overview page.

**Main flow**

1. User chooses a template to use and clicks the generate button.
2. System authenticates user & information and updates the database.
3. System indicates that changes are successful.
4. System redirects the user back to the portfolio overview page showing the new portfolio addition.
5. User chooses to modify existing portfolio.
6. System authenticates user & information and updates the database.
7. System indicates that changes are successful.
8. System redirects the user back to the portfolio overview page.
9. User chooses to view the portfolio in full.
10. System authenticates user and retrieves portfolio information from the database.
11. System opens a new window and displays the portfolio to the user.

**Alternate flow**

* 2a. Failed authentication
* 2a1. System notifies user of failed authentication and re-requests login credentials or valid information.
* 2a2. User re-enters login credentials or valid information.
* 2a3. System authenticates user or information.
* 6a. Failed authentication
* 6a1. System notifies user of failed authentication and re-requests login credentials or valid information.
* 6a2. User re-enters login credentials or valid information.
* 6a3. System authenticates user or information.
* 9a. Failed authentication
* 9a1. System notifies user of failed authentication and re-requests login credentials or valid information.
* 9a2. User re-enters login credentials or valid information.
* 9a3. System authenticates user or information.

**Termination**

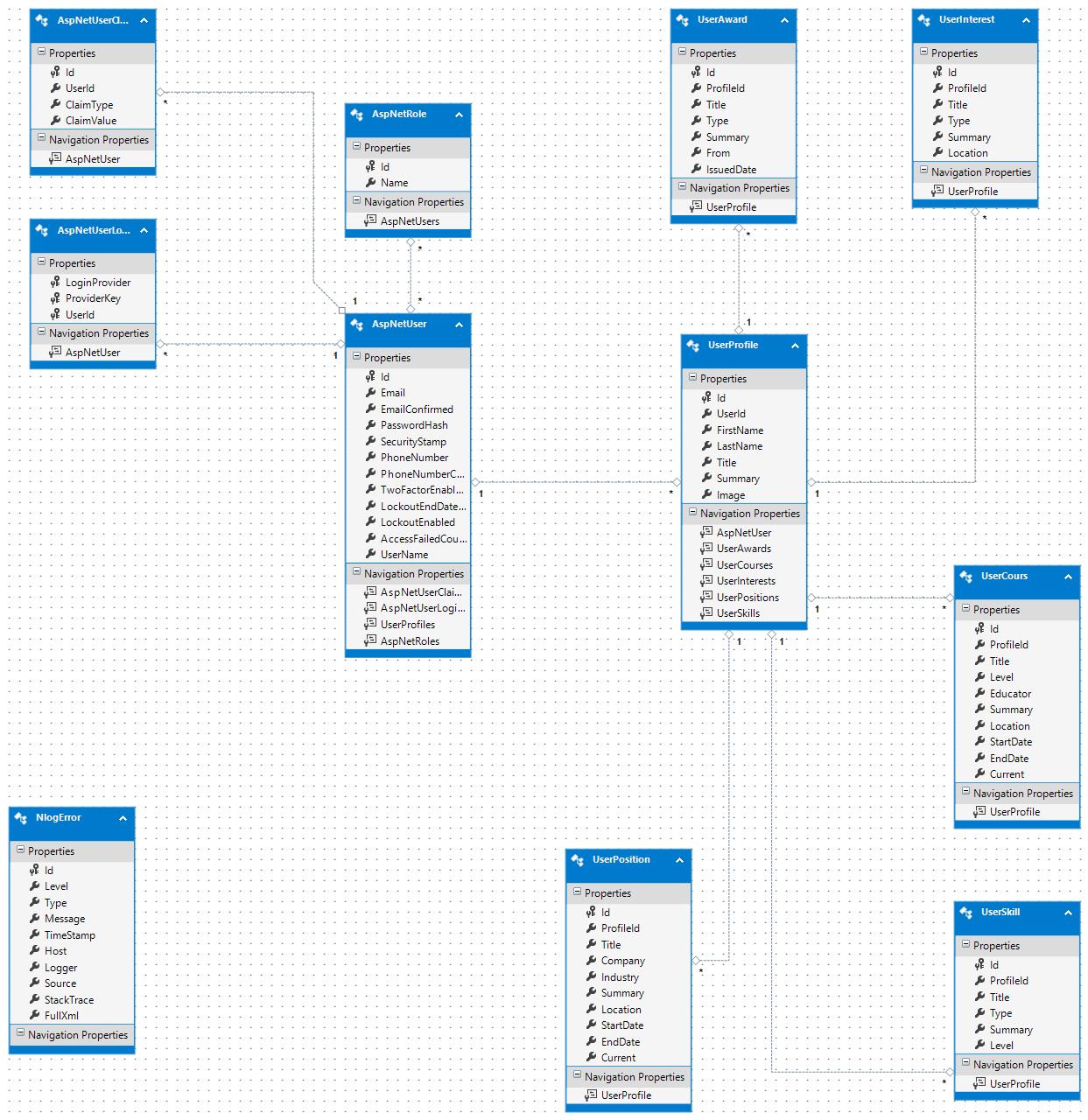
* The System redirects the user to the portfolio overview page or portfolio

**Post condition**

* The system goes into a wait state

### Data requirements

**Database diagram**



The database is required handle user account and information for the application. The application is required to be able to run under multiple environments for example deployed to Azure cloud services and also be able to be used in a local capacity. I enabled database migrations to provide a quick solution which will allow the developer to generate the required tables when changing from one environment to another. I also provided some basic seeding for testing and presentation purposes.

### User requirements

1. User should be able to register to use the application.
2. User should be able to login after registering.
3. User should be able to manage their account i.e. update profile.
4. User should be able to manage their information (Experience, Education, Awards, and Interests etc.).
5. User should be able to generate and modify multiple curriculum vitae’s.
6. User should be able to generate and modify a portfolio.

### Environmental requirements

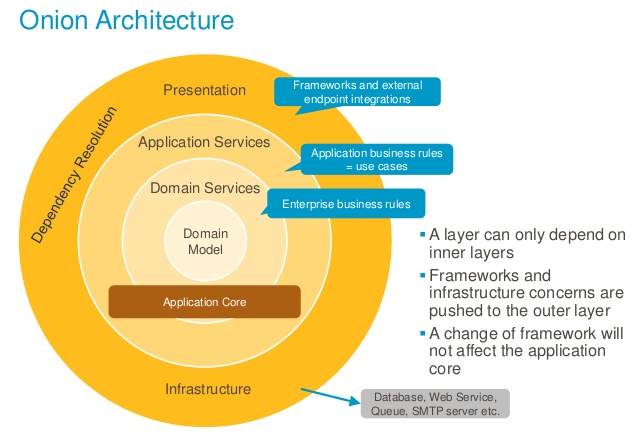
* **Visual Studio:** the application has been developed locally using this IDE.
* **Database:** SQL Server Express Local DB through Visual Studio
* **Git:** provides a local version control for the project solution.
* **Windows OS, Chrome Browser**
* **Azure Cloud Services:** provides cloud hosting for the web application and database.
* **Visual Studio Online:** provides Git version control monitoring tools and deployment options.

## Usability requirements

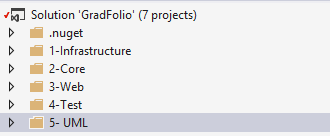
1. The User requires little to no knowledge or training to successfully operate and navigate the different functionality of the system.
2. The User requires that information input should be clear and understandable in what is required.
3. The User should be made aware of the actions performed or the failure of these actions to be performed in a meaningful way.

## Design and Architecture

The applications architecture was implemented based on Onion Architecture design concepts. The main aim of this concept is to create an application that is loosely coupled. This is done by separating different areas of concern into separate folders/applications making each layer independent of each other. Thus making the development process, testing and maintenance easier.

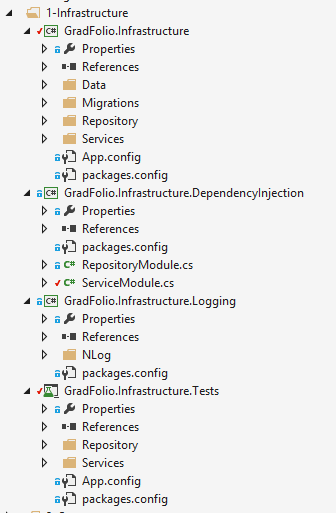
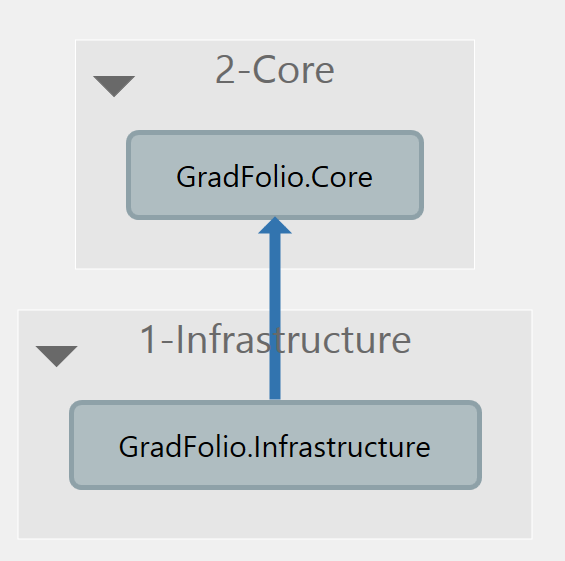


**Project solution structure**



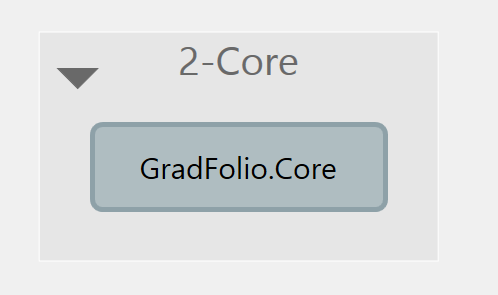
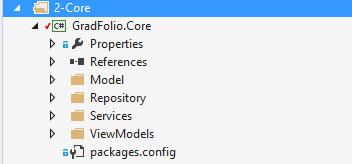
**Description:** The project is divide into multiple layers each layer addressing its own concern. The fundamental rule is that all code can depend on layers more central, but code cannot depend on layers further out from the core.

**1-Infrastructure layer**

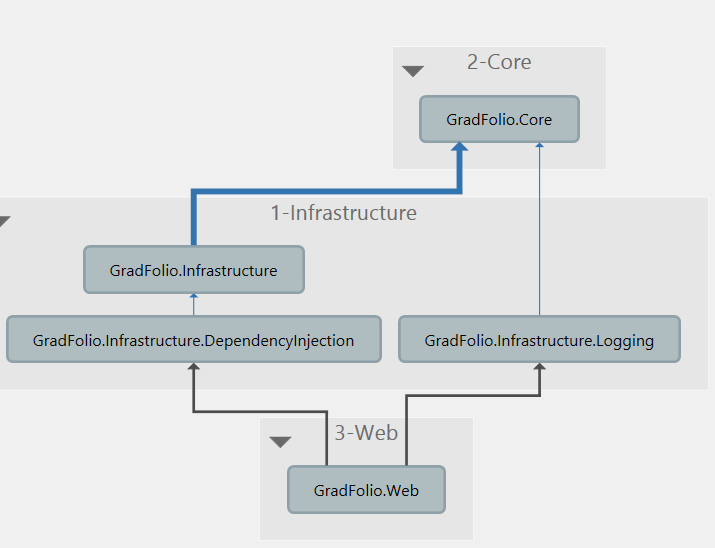
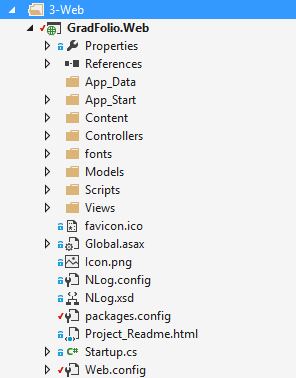
**Description:** This is the outermost layer of onion architecture which deals with Infrastructure needs and provides the implementation of your repositories interfaces. In other words, this is where we hook up the Data access logic or logging logic or Dependency injection.

**2-Core Layer (Domain)**



**Description:** At the very core is the Domain layer which holds all of your domain objects. The idea is to have all of your domain objects at this core. Please restrict yourself by keeping just the properties or definitions inside your domain objects and not any piece of code which talks to database or has any other business functions. Besides the domain objects, you could also have domain interfaces, but just the interfaces and not any kind of implementation.

**3-Web layer (Application)**

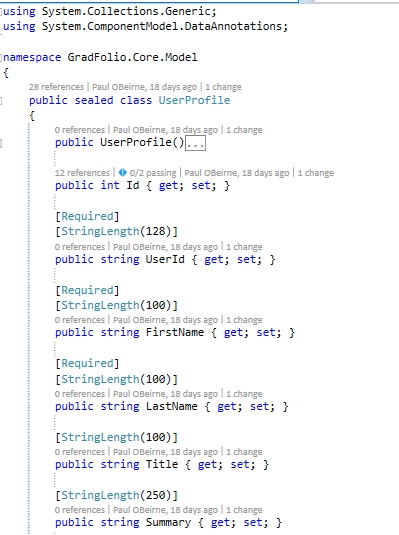
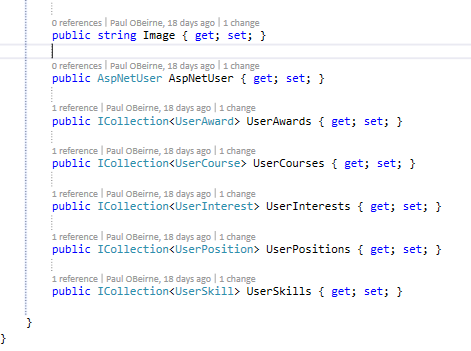


**Description:** Web defines what features your system provide. I think it's highly coupled with the Interfaces in the core layer. If you define a method in Application, often you need to add an Interfaces class/method as well. But several Interfaces class/method may depends on the same Application object.

## Implementation

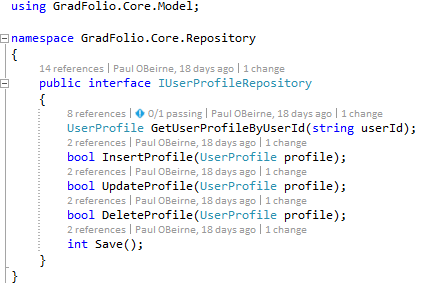
**Sample Model**

This is a direct model translation of the profile table in the database. The model has validation data notations applied.

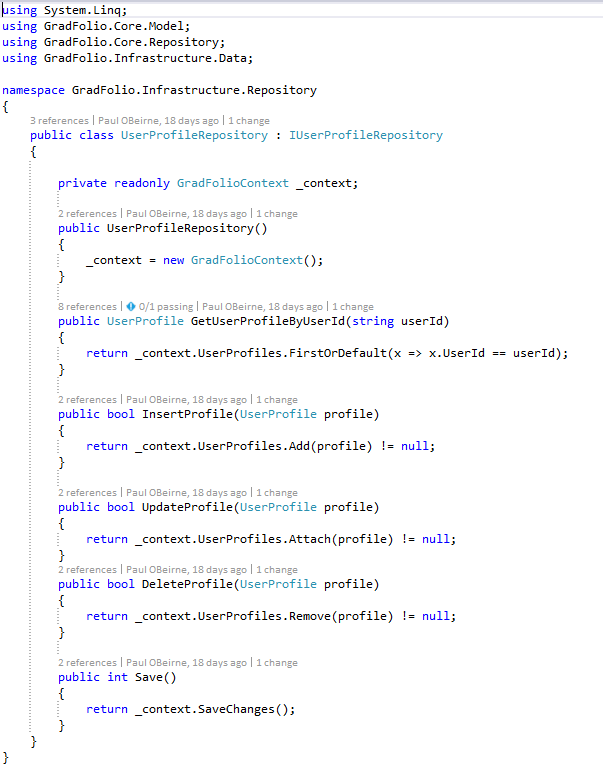
**Sample Repository Interface**

This is an interface class with a set of methods that will provide the functionality for the repository to interact with the database context in retrieving data from the database.



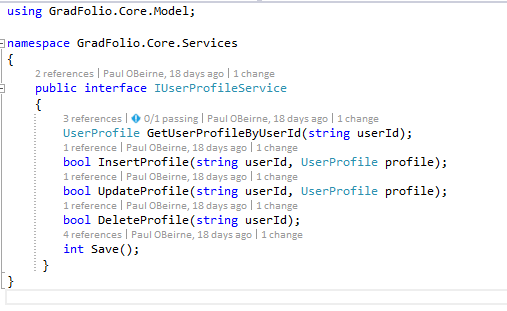
**Sample Repository implementation**

This class implements the sample interface and provides the actual implementation and functionality required to perform CRUD functions on the database.



**Service Interface example**

This is an interface class with a set of methods that will provide the functionality for the service to interact with the repository.



**Service example**

This class implements the sample interface and provides the implementation and functionality required to call the repository methods. Additional validation and logging has been implemented to provide more security before date is retrieved or submitted to the database.

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using GradFolio.Core.Model;

using GradFolio.Core.Repository;

using GradFolio.Core.Services;

using GradFolio.Infrastructure.Services.Helpers;

namespace GradFolio.Infrastructure.Services

{

public class UserProfileService : DataAnnotationsValidator, IUserProfileService

{

private readonly ILoggingService \_loggingService;

private readonly IUserProfileRepository \_profileRepository;

private readonly IUserRepository \_userRepository;

public UserProfileService(ILoggingService loggingService,

IUserProfileRepository profileRepository,

IUserRepository userRepository)

{

\_loggingService = loggingService;

\_profileRepository = profileRepository;

\_userRepository = userRepository;

}

public UserProfile GetUserProfileByUserId(string userId)

{

try

{

//Validate user

if (\_userRepository.IsAuthenticated(userId))

{

//GetUserProfile

var profile = \_profileRepository.GetUserProfileByUserId(userId);

if (profile != null)

{

//Success

return profile;

}

\_loggingService.Info("Not profile for the user found: " + userId);

}

\_loggingService.Info("UserId Authenticated Failed: " + userId);

}

catch (Exception ex)

{

//Error

\_loggingService.Error("An error has occurred", ex);

}

//Fail

return null;

}

public bool InsertProfile(string userId, UserProfile profile)

{

try

{

//Validate user

if (\_userRepository.IsAuthenticated(userId))

{

//Validate Model

ICollection<ValidationResult> results;

if (IsValidModel(profile, out results))

{

//Call Repository

if (\_profileRepository.InsertProfile(profile))

{

//Save

if (Save() != 0)

{

//Success

return true;

}

\_loggingService.Info("Failed To Save");

}

\_loggingService.Info("UserRepository Failed Insert");

}

\_loggingService.Info("Model Validation Failed: " + profile);

}

\_loggingService.Info("UserId Authenticated Failed: " + userId);

}

catch (Exception ex)

{

//Error

\_loggingService.Error("An error has occurred", ex);

}

//Fail

return false;

}

public bool UpdateProfile(string userId, UserProfile profile)

{

try

{

//Validate user

if (\_userRepository.IsAuthenticated(userId))

{

//Validate Model

ICollection<ValidationResult> results;

if (IsValidModel(profile, out results))

{

//Call Repository

if (\_profileRepository.UpdateProfile(profile))

{

//Save

if (Save() != 0)

{

//Success

return true;

}

\_loggingService.Info("Failed To Save");

}

\_loggingService.Info("UserRepository Failed Update");

}

\_loggingService.Info("Model Validation Failed: " + profile);

}

\_loggingService.Info("UserId Authenticated Failed: " + userId);

}

catch (Exception ex)

{

//Error

\_loggingService.Error("An error has occurred", ex);

}

//Fail

return false;

}

public bool DeleteProfile(string userId)

{

try

{

//Validate user

if (\_userRepository.IsAuthenticated(userId))

{

//Validate profile

var profile = GetUserProfileByUserId(userId);

if (profile != null)

{

//Call Repository

if (\_profileRepository.DeleteProfile(profile))

{

//Save

if (Save() != 0)

{

//Success

return true;

}

\_loggingService.Info("Failed To Save");

}

\_loggingService.Info("UserRepository Failed Delete");

}

\_loggingService.Info("No user Profile Exists: " + userId);

}

\_loggingService.Info("UserId Authenticated Failed: " + userId);

}

catch (Exception ex)

{

//Error

\_loggingService.Error("An error has occurred", ex);

}

//Fail

return false;

}

public int Save()

{

try

{

//Success

return \_profileRepository.Save();

}

catch (Exception ex)

{

//Error

\_loggingService.Error("An error has occurred", ex);

}

//Fail

return 0;

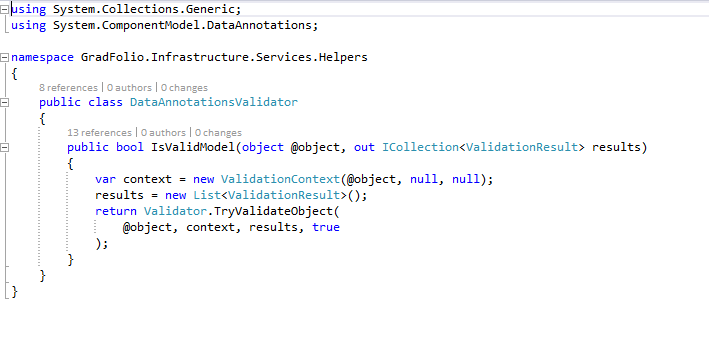
}

}

}

**Service layer model validator helper class**

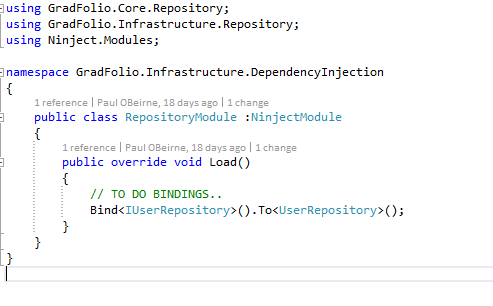
This class is checks the data annotations of a model and validates the model. This is important mainly to make sure the information is correct before calling and repository methods.



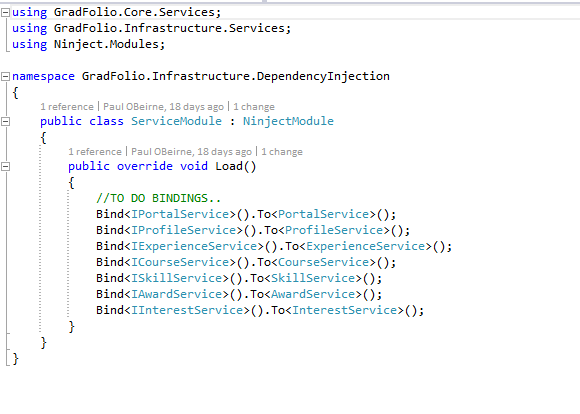
**Dependency Injection**

These methods a initialized from the web UI layer

Creates the required decencies for the repositories

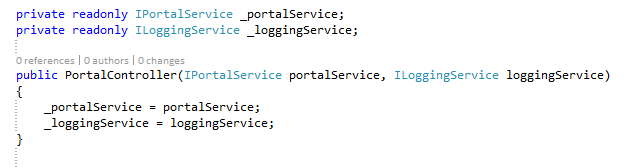


Creates the required decencies for the services



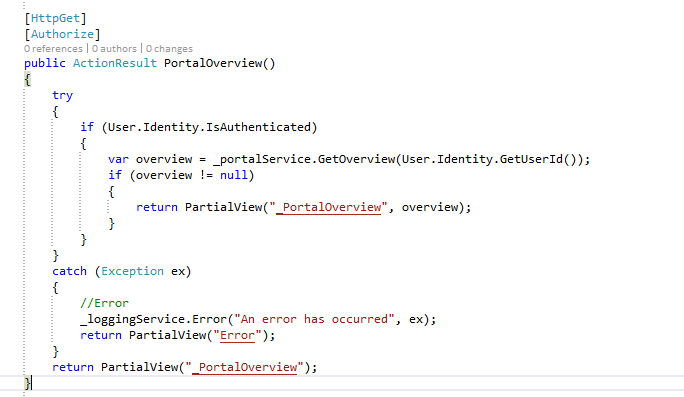
**Controller (Portal)**

Initialise required services (Dependency Injection)



**Controller method to return Portal Overview**

This method will return the users portal overview information if the user is valid and the retrieved data is not null else the user will be returned the original page or if an error occurs then to an error page.



**Portal Overview view**

This contains the basic container for the content section which will be loaded when the page is loaded. The actual overview information will be in the loaded partial view.



**Partial view load into Overview view**

Display the users overview information

@model GradFolio.Core.ViewModels.PortalOverviewViewModel

<div class="row">

<div class="col-lg-12 col-md-12 col-xs-12" style="max-height: 660px; overflow-y: auto; margin-bottom: 20px;">

<div class="row">

<div class="col-lg-2 col-md-4 col-xs-6">

<div class="thumbnail">

<p align="center">

Experience

@if (Model.PortalOverviewStats.ExperienceCount > 0)

{

<span class="label label-success">

<span>@Model.PortalOverviewStats.ExperienceCount</span>

</span>

}

else

{

<span class="label label-danger">

<span>0</span>

</span>

}

</p>

<div style="text-align: center;">

<a href="@Url.Action("Experience", "Portal")" style="text-align: center;">

<h1 class="glyphicon glyphicon-briefcase" aria-hidden="true"></h1>

</a>

</div>

</div>

</div>

<div class="col-lg-2 col-md-4 col-xs-6">

<div class="thumbnail">

<p align="center">

Courses

@if (Model.PortalOverviewStats.CourseCount > 0)

{

<span class="label label-success">

<span>@Model.PortalOverviewStats.CourseCount</span>

</span>

}

else

{

<span class="label label-danger">

<span>0</span>

</span>

}

</p>

<div style="text-align: center;">

<a href="@Url.Action("Courses", "Portal")" style="text-align: center;">

<h1 class="glyphicon glyphicon-education" aria-hidden="true"></h1>

</a>

</div>

</div>

</div>

<div class="col-lg-2 col-md-4 col-xs-6">

<div class="thumbnail">

<p align="center">

Skills

@if (Model.PortalOverviewStats.SkillCount > 0)

{

<span class="label label-success">

<span>@Model.PortalOverviewStats.SkillCount</span>

</span>

}

else

{

<span class="label label-danger">

<span>0</span>

</span>

}

</p>

<div style="text-align: center;">

<a href="@Url.Action("Skills", "Portal")" style="text-align: center;">

<h1 class="glyphicon glyphicon-sunglasses" aria-hidden="true"></h1>

</a>

</div>

</div>

</div>

<div class="col-lg-2 col-md-4 col-xs-6">

<div class="thumbnail">

<p align="center">

Awards

@if (Model.PortalOverviewStats.AwardCount > 0)

{

<span class="label label-success">

<span>@Model.PortalOverviewStats.AwardCount</span>

</span>

}

else

{

<span class="label label-danger">

<span>0</span>

</span>

}

</p>

<div style="text-align: center;">

<a href="@Url.Action("Awards", "Portal")" style="text-align: center;">

<h1 class="glyphicon glyphicon-certificate" aria-hidden="true"></h1>

</a>

</div>

</div>

</div>

<div class="col-lg-2 col-md-4 col-xs-6">

<div class="thumbnail">

<p align="center">

Interests

@if (Model.PortalOverviewStats.InterestCount > 0)

{

<span class="label label-success">

<span>@Model.PortalOverviewStats.InterestCount</span>

</span>

}

else

{

<span class="label label-danger">

<span>0</span>

</span>

}

</p>

<div style="text-align: center;">

<a href="@Url.Action("Interests", "Portal")" style="text-align: center;">

<h1 class="glyphicon glyphicon-book" aria-hidden="true"></h1>

</a>

</div>

</div>

</div>

<div class="col-lg-2 col-md-4 col-xs-6">

<div class="thumbnail">

<p align="center">

Projects

@if (Model.PortalOverviewStats.ProjectCount > 0)

{

<span class="label label-success">

<span>@Model.PortalOverviewStats.ProjectCount</span>

</span>

}

else

{

<span class="label label-danger">

<span>0</span>

</span>

}

</p>

<div style="text-align: center;">

<a href="@Url.Action("Projects", "Portal")" style="text-align: center;">

<h1 class="glyphicon glyphicon-blackboard" aria-hidden="true"></h1>

</a>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-lg-6 col-md-6 col-xs-12">

<div class="thumbnail">

<div class="list-group" style="margin-bottom: 0;">

<div class="list-group-item list-group-item-info">

<h4>

Profile

</h4>

</div>

@if (Model.ProfileOverview != null)

{

<a href="@Url.Action("ProfileInfo", "Portal")" class="list-group-item">

<div class="row">

<div class="col-lg-3 col-md-3 col-xs-12">

<div class="thumbnail">

<img class="img-responsive" src="@Model.ProfileOverview.ImageUrl" alt="">

</div>

</div>

<div class="col-lg-9 col-md-9 col-xs-12">

<h4 class="list-group-item-heading">@Model.ProfileOverview.Name</h4>

<br/>

<p class="list-group-item-text">

Title :@Model.ProfileOverview.Title <br/>

Join Date: @Model.ProfileOverview.JoinDate.ToShortDateString()

</p>

<br/><br/>

</div>

</div>

</a>

}

else

{

<a href="@Url.Action("ProfileInfo", "Portal")" class="list-group-item">

<div class="alert alert-danger" role="alert">...Error</div>

</a>

}

<div class="list-group-item list-group-item-info">

<h4>

Latest CV

</h4>

</div>

@if (Model.CvOverviews != null)

{

<a href="@Url.Action("CurriculumVitae", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Title :@Model.CvOverviews.First().FileName </h4>

<span class="list-group-item-text">

Type: @Model.CvOverviews.First().Type<br />

Create Date: @Model.CvOverviews.First().CreateDate.ToShortDateString()

</span>

</a>

}

else

{

<a href="@Url.Action("CurriculumVitae", "Portal")" class="list-group-item">

<div class="alert alert-danger" role="alert">...Error</div>

</a>

}

<div class="list-group-item list-group-item-info">

<h4>

Portfolio

</h4>

</div>

@if (Model.PortfolioOverview != null)

{

<a href="@Url.Action("Portfolio", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">My Portfolio</h4>

<span class="list-group-item-text">

Type: @Model.PortfolioOverview.Type<br />

Create Date: @Model.PortfolioOverview.CreateDate.ToShortDateString()

</span>

</a>

}

else

{

<a href="@Url.Action("Portfolio", "Portal")" class="list-group-item">

<div class="alert alert-danger" role="alert">...Error</div>

</a>

}

</div>

</div>

</div>

<div class="col-lg-6 col-md-6 col-xs-12">

<div class="thumbnail">

<div class="list-group" style="margin-bottom: 0;">

@if (Model.ExperienceOverviews != null)

{

<a href="@Url.Action("Experience", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Latest Experience</h4>

<p class="list-group-item-text">

Title :@Model.ExperienceOverviews.First().Title <br/>

Company: @Model.ExperienceOverviews.First().Company

</p>

</a>

}

@if (Model.CourseOverviews != null)

{

<a href="@Url.Action("Courses", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Latest Course</h4>

<p class="list-group-item-text">

Title :@Model.CourseOverviews.First().Title <br/>

Company: @Model.CourseOverviews.First().College

</p>

</a>

}

@if (Model.SkillOverviews != null)

{

<a href="@Url.Action("Skills", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Latest Skill</h4>

<p class="list-group-item-text">

Title :@Model.SkillOverviews.First().Title <br/>

Company: @Model.SkillOverviews.First().Type

</p>

</a>

}

@if (Model.AwardOverviews != null)

{

<a href="@Url.Action("Awards", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Latest Award</h4>

<p class="list-group-item-text">

Title :@Model.AwardOverviews.First().Title <br/>

Company: @Model.AwardOverviews.First().Type

</p>

</a>

}

@if (Model.InterestOverviews != null)

{

<a href="@Url.Action("Interests", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Latest Interest</h4>

<p class="list-group-item-text">

Title :@Model.InterestOverviews.First().Title <br/>

Company: @Model.InterestOverviews.First().Type

</p>

</a>

}

@if (Model.ProjectOverviews != null)

{

<a href="@Url.Action("Projects", "Portal")" class="list-group-item">

<h4 class="list-group-item-heading">Latest Project</h4>

<p class="list-group-item-text">

Title :@Model.ProjectOverviews.First().Title <br/>

Company: @Model.ProjectOverviews.First().Type

</p>

</a>

}

</div>

</div>

</div>

</div>

</div>

</div>

**Script**

This script is the basic portal script that handle all the page load reload and model popups for each section in the portal.

//Turn off caching

$.ajaxSetup({ cache: false });

//URL Parameters

//Sort Parameters

var sort\_1 = document.getElementById("sortbydate");

var sort\_2 = document.getElementById("sortbyaward");

var sort\_3 = document.getElementById("sortbyissuer");

//################ Content Load/Reload/Sort Functions

//Initial Page Load

$(function() {

loadContent(window.loadContent\_url);

});

function reloadBtnClick() {

loadContent(window.loadContent\_url);

}

//Sort

function itemMenuBtnClick(action) {

//Remove active css

removeClass("active", sort\_1);

removeClass("active", sort\_2);

removeClass("active", sort\_3);

switch (action) {

case "add":

//alert("view");

loadViewModel(window.addContent\_url);

break;

case "reload":

//alert(action);

loadContent(window.loadContent\_url);

break;

case "sortby1":

//alert(action);

addClass("active", sort\_1);

loadContent(window.loadContent\_url + "?sort=" + action);

break;

case "sortby2":

//alert(action);

addClass("active", sort\_2);

loadContent(window.loadContent\_url + "?sort=" + action);

break;

case "sortby3":

//alert(action);

addClass("active", sort\_3);

loadContent(window.loadContent\_url + "?sort=" + action);

break;

case "":

alert("empty");

break;

}

}

//Loading Function

function loadContent(url) {

var content = $("#content");

content.fadeOut(400, function() {

$("#container").append("<div id=\"loading\">Loading..." +

" <img src=\"../Content/Images/loading1.gif\" /></div>");

var loading = $("#loading");

loading.fadeIn(400, function() {

content.load(url, function() {

loading.fadeOut(400, function() {

$(this).remove();

content.fadeIn(400);

});

});

});

});

}

//################ Content Form Functions

function itemBtnClick(action, id) {

switch (action) {

case "view":

//alert("view");

loadViewModel(window.viewContent\_url + "?itemId=" + id);

break;

case "edit":

//alert("edit");

loadViewModel(window.editContent\_url + "?itemId=" + id);

break;

case "remove":

//alert("remove");

loadViewModel(window.removeContent\_url + "?itemId=" + id);

break;

case "":

//alert("empty");

break;

}

}

$(document).on("click", "#btn-content-add-submit", function() {

var self = $(this);

$.ajax({

url: window.addContent\_url,

type: "POST",

data: self.closest("form").serialize(),

success: function(data) {

if (data.success === true) {

$("#popupModal").modal("hide");

loadContent(window.loadContent\_url);

return false;

} else {

$("#popupcontainer").html(data);

}

return false;

}

});

});

$(document).on("click", "#btn-content-edit-submit", function() {

var self = $(this);

$.ajax({

url: window.editContent\_url,

type: "POST",

data: self.closest("form").serialize(),

success: function(data) {

if (data.success === true) {

$("#popupModal").modal("hide");

loadContent(window.loadContent\_url);

return false;

} else {

$("#popupcontainer").html(data);

}

return false;

}

});

});

$(document).on("click", "#btn-content-remove-submit", function() {

var self = $(this);

$.ajax({

url: window.removeContent\_url,

type: "POST",

data: self.closest("form").serialize(),

success: function(data) {

if (data.success === true) {

$("#popupModal").modal("hide");

loadContent(window.loadContent\_url);

return false;

} else {

$("#popupcontainer").html(data);

}

return false;

}

});

});

//################ Model Function

function loadViewModel(url) {

$("#popupcontainer").html("");

$("#popupcontainer").load(url);

$("#popupModal").modal("show");

}

//################ Helper Functions

function addClass(classname, element) {

var cn = element.className;

//test for existance

if (cn.indexOf(classname) !== -1) {

return;

}

//add a space if the element already has class

if (cn !== "") {

classname = " " + classname;

}

element.className = cn + classname;

}

function removeClass(classname, element) {

var cn = element.className;

var rxp = new RegExp("\\s?\\b" + classname + "\\b", "g");

cn = cn.replace(rxp, "");

element.className = cn;

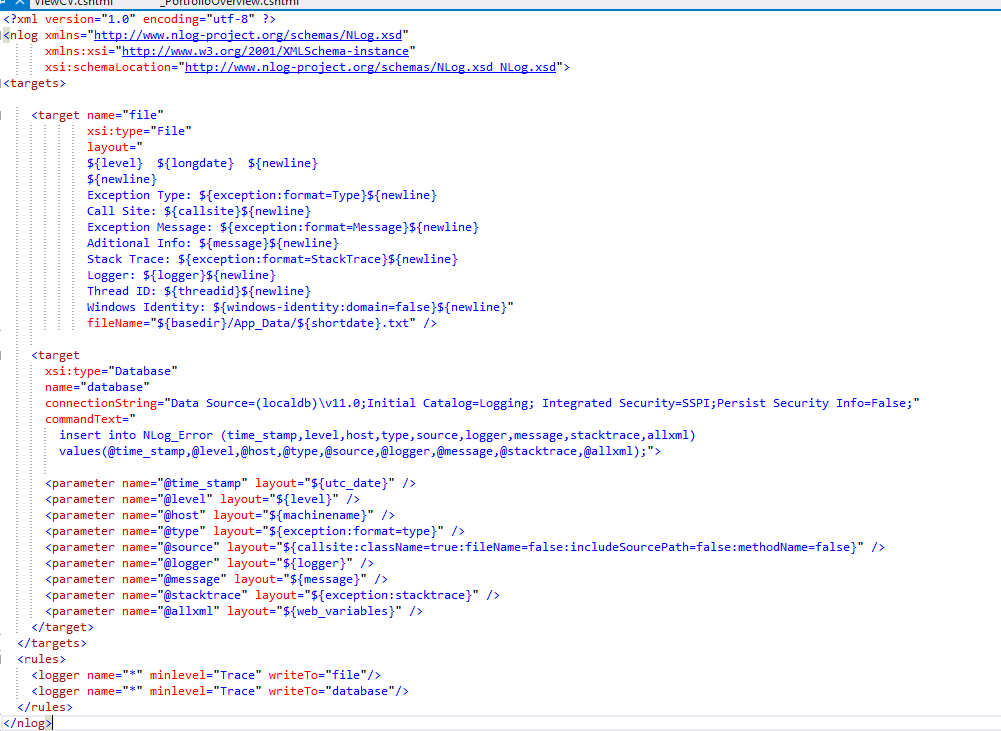
}

function hasClass(classname, element) {

return (" " + element.className + " ").indexOf(" " + classname + " ") > -1;}

**Nlog**

Sends exceptions, errors and other information to a txt file in the project and also to the database table. The database will be the main storage when deployed to the cloud as it will provide a centralised place for logging information regardless of how many instances of the application exists

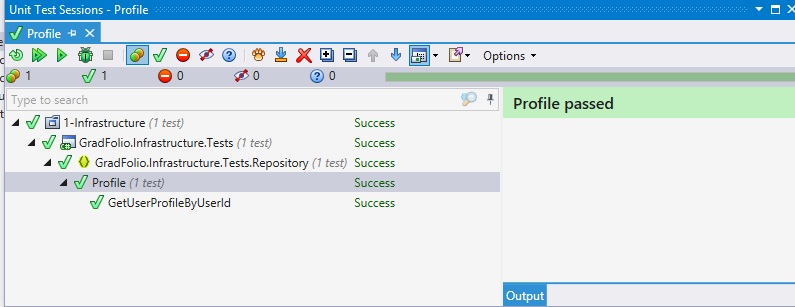


## Testing & Performance

Describe any testing tools, test plans and test specifications used in the project

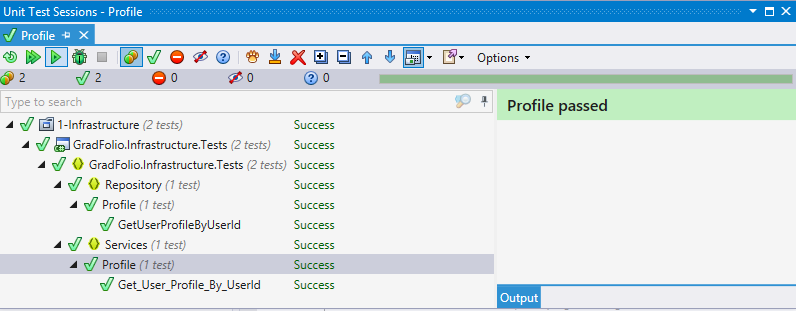
**Repository**





**Services**

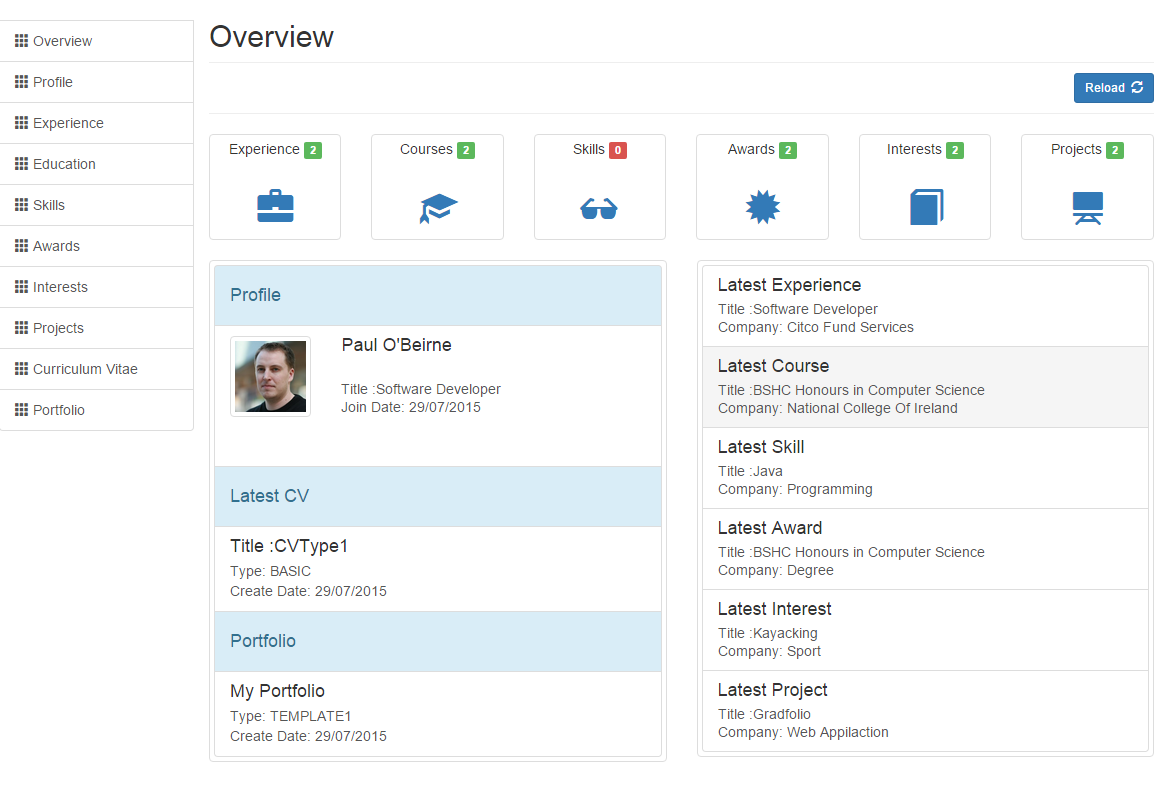




## Graphical User Interface (GUI) Layout

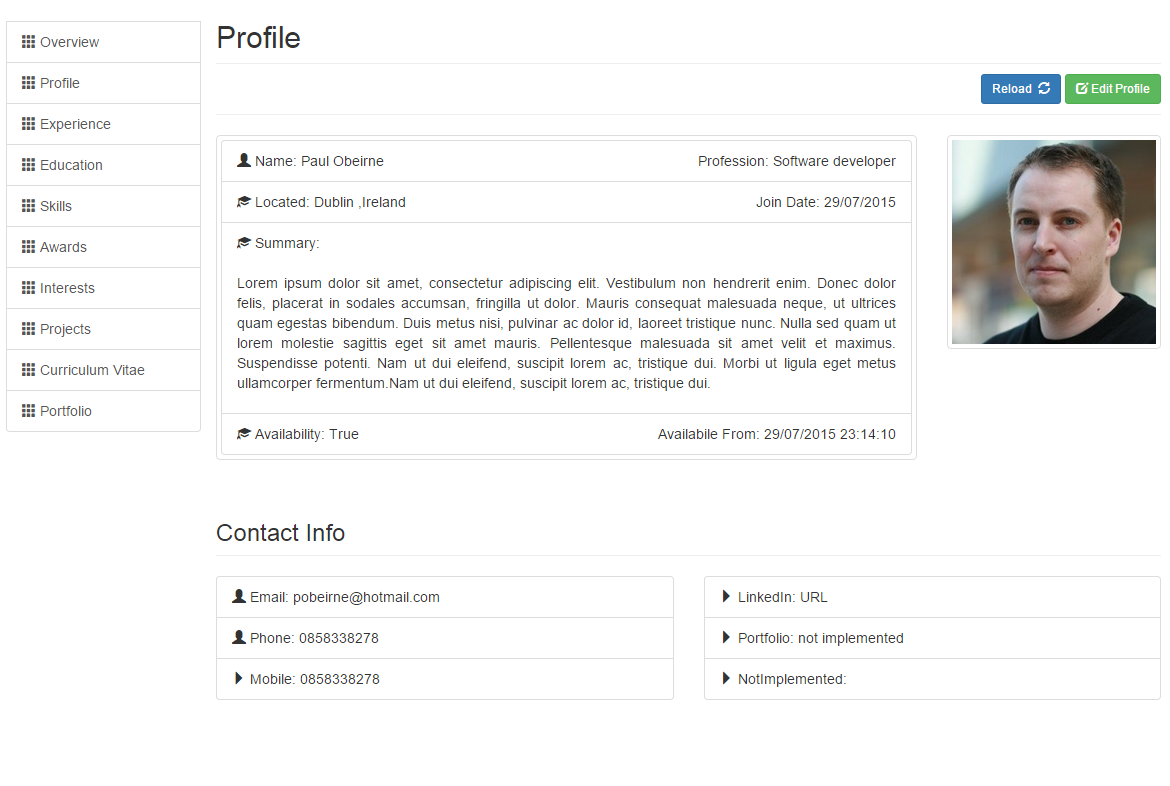
**GradFolio Portal**

**Overview**



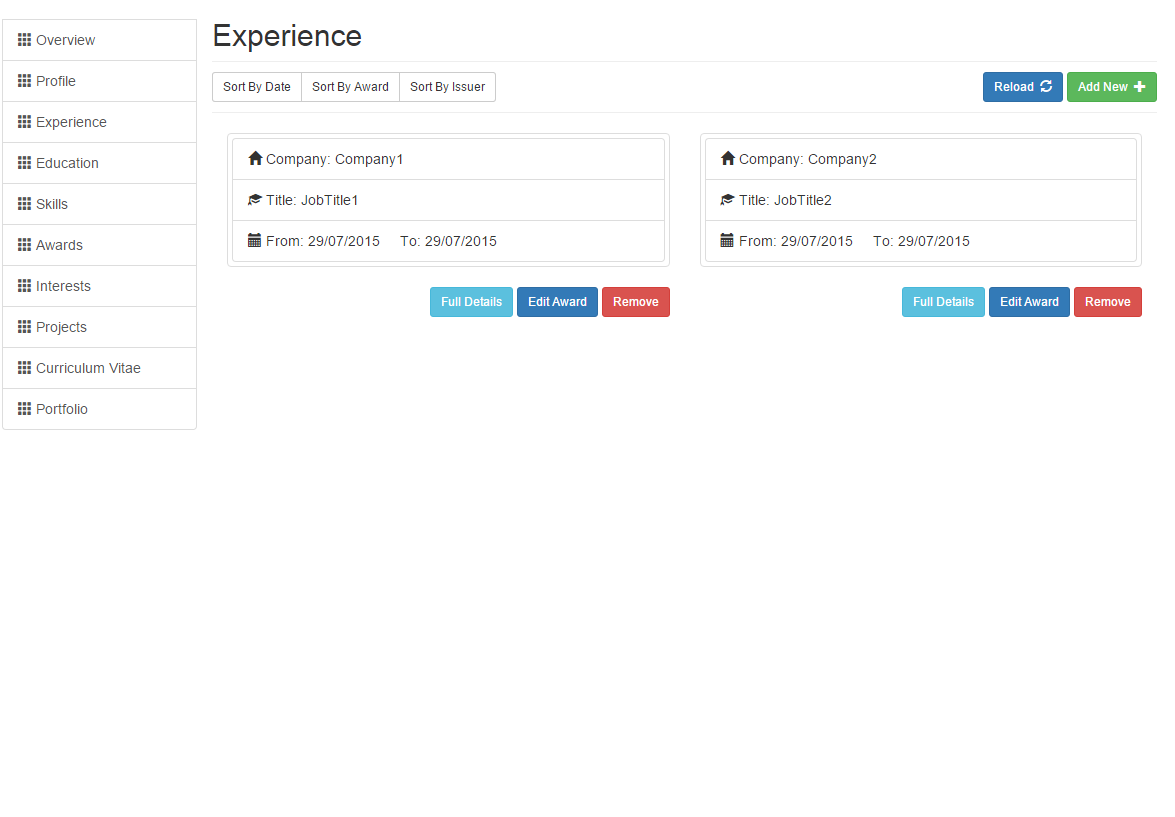
Provides the user with an overview of their information and generated content that they have added to the system. The top area shows the number of items added for section while below shows a brief overview of the users profile, current portfolio and latest additions. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. The user can navigate to different sections using the sidebar navigation or by clicking on the item.

**Profile**



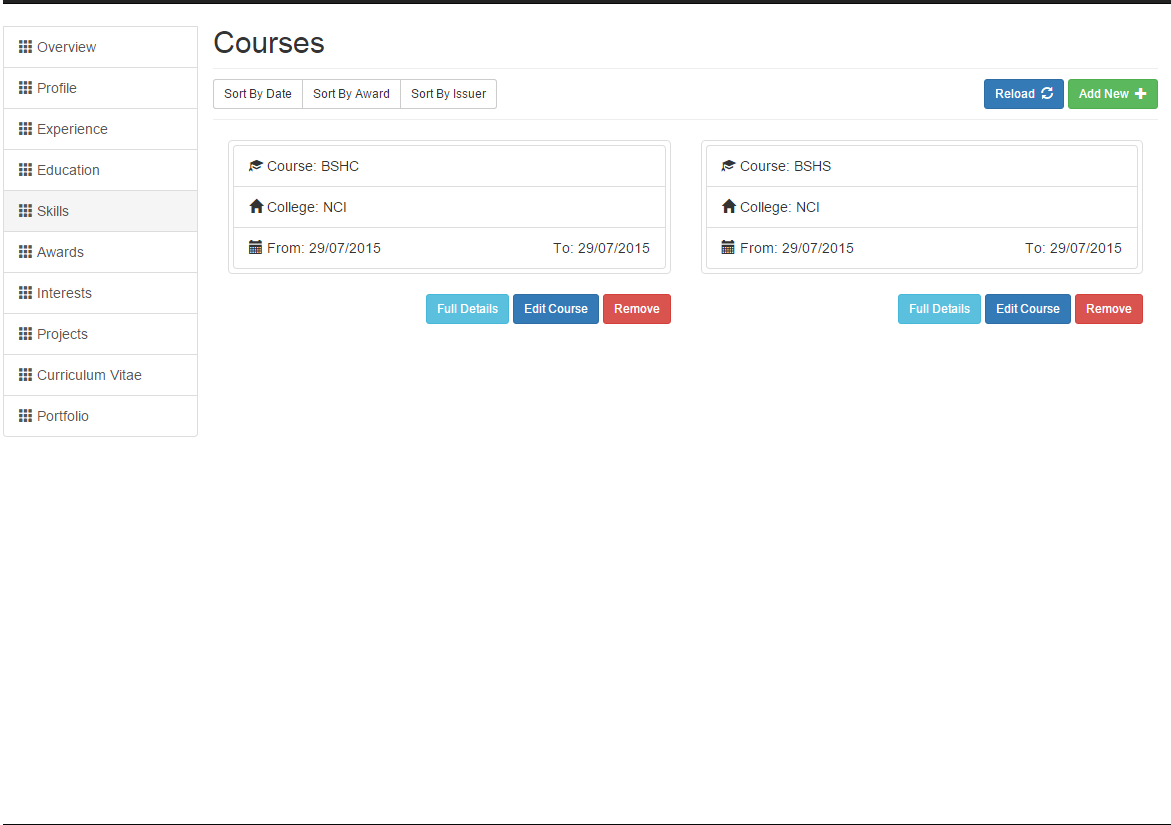
Displays the users current profile information. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. The user can click the edit button and a new model window form will popup allowing the user to update their information.

**Experience**



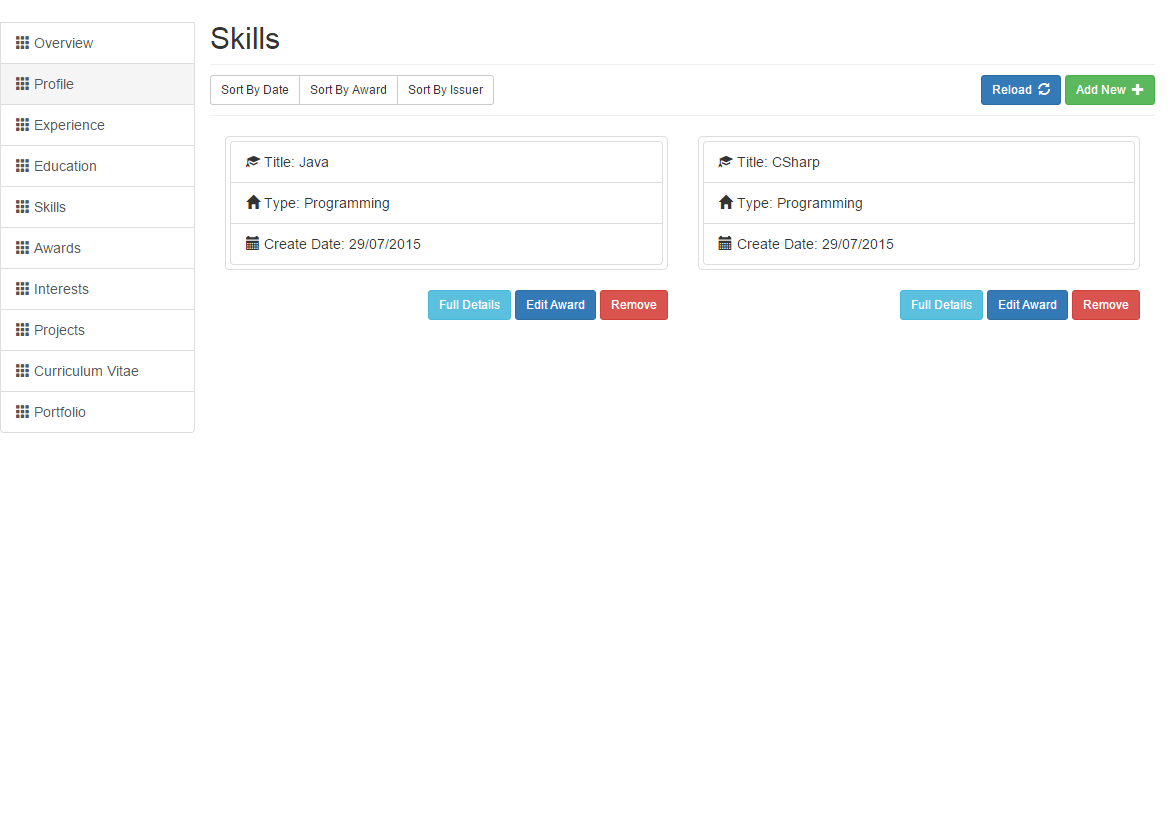
Displays a list of the users work experiences. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full experience by clicking the Full Details button in which a model window will popup displaying the full information. Other actions can be take such as adding, editing and removing experiences.

**Education**



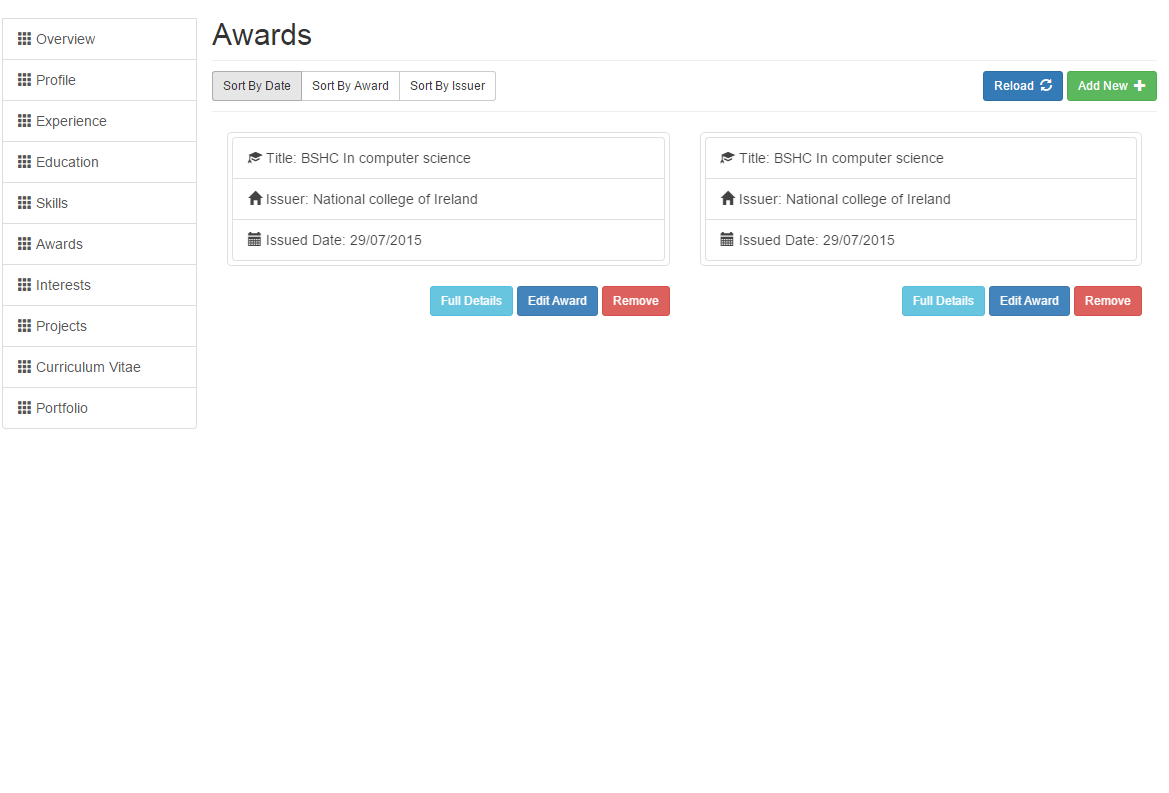
Displays a list of the user’s educational courses. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full course by clicking the Full Details button in which a model window will popup displaying the full information. Other actions can be take such as adding, editing and removing courses.

**Skills**



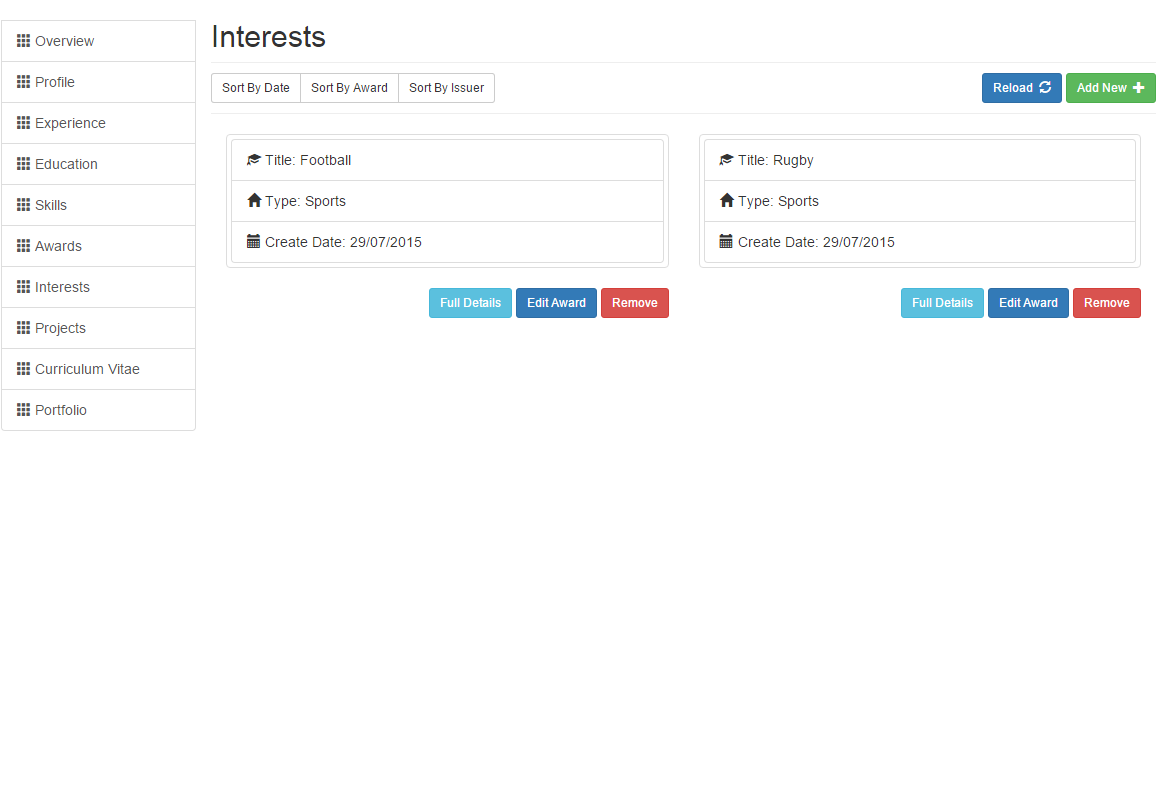
Displays a list of the user’s skills. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full skill by clicking the Full Details button in which a model window will popup displaying the full information. Other actions can be take such as adding, editing and removing skills.

**Awards**



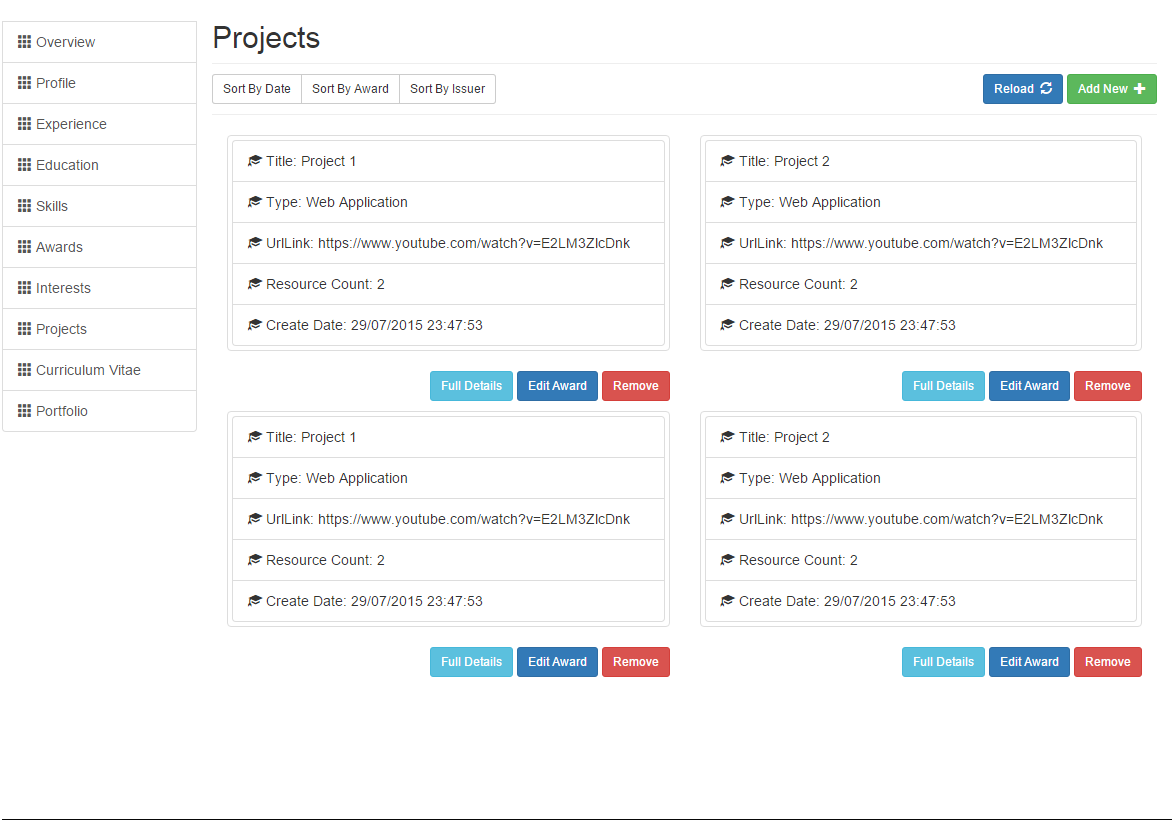
Displays a list of the user’s awards. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full award by clicking the Full Details button in which a model window will popup displaying the full information. Other actions can be take such as adding, editing and removing skills.

**Interests**



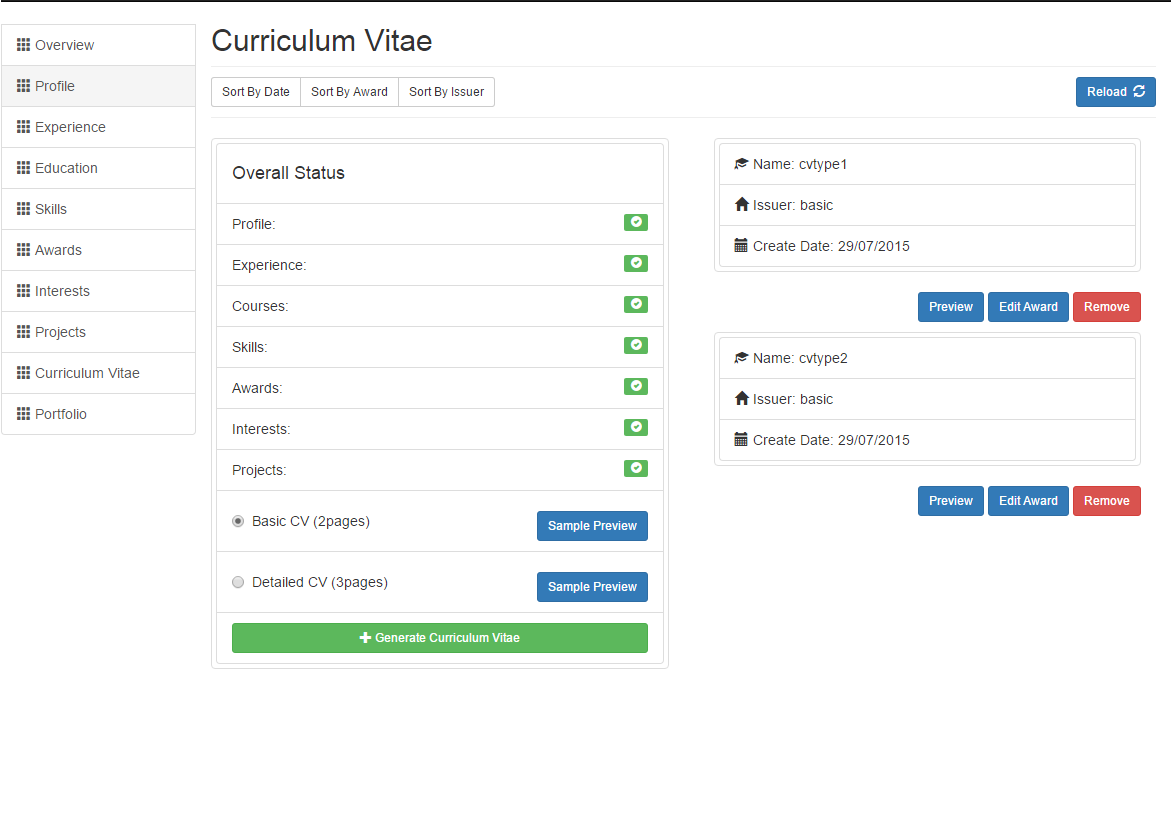
Displays a list of the user’s interests. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full interest by clicking the Full Details button in which a model window will popup displaying the full information. Other actions can be take such as adding, editing and removing interests.

**Projects**



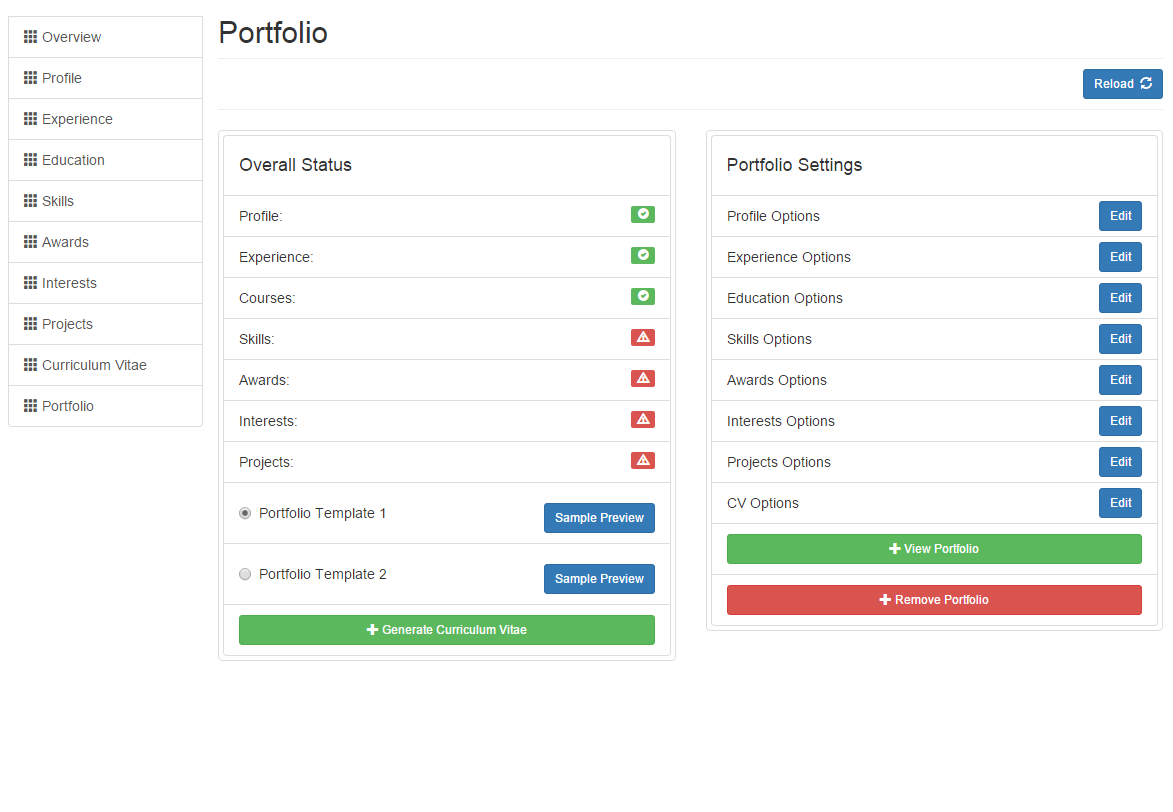
Displays a list of the user’s projects. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full projects by clicking the Full Details button in which a model window will popup displaying the full information. Other actions can be take such as adding, editing and removing projects.

**Curriculum vitae**



Displays a status overview which determines if the user has sufficient information to be able to generate a CV. If the user doesn’t the generate button will be disable preventing the user from creating a new CV. If the user has sufficient information the user can then select a profile template and generate a new CV. The user will be able to initially name the new CV and select which information will be shown based on the restrictions each template has. The users existing CV’s will be displayed on the right side. The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. Each list item only display’s a small amount of information but the user can choose to view each full cv by clicking the preview button in which a new window will popup displaying the full information. Other actions can be take such as adding, editing and removing CV’s.

Portfolio

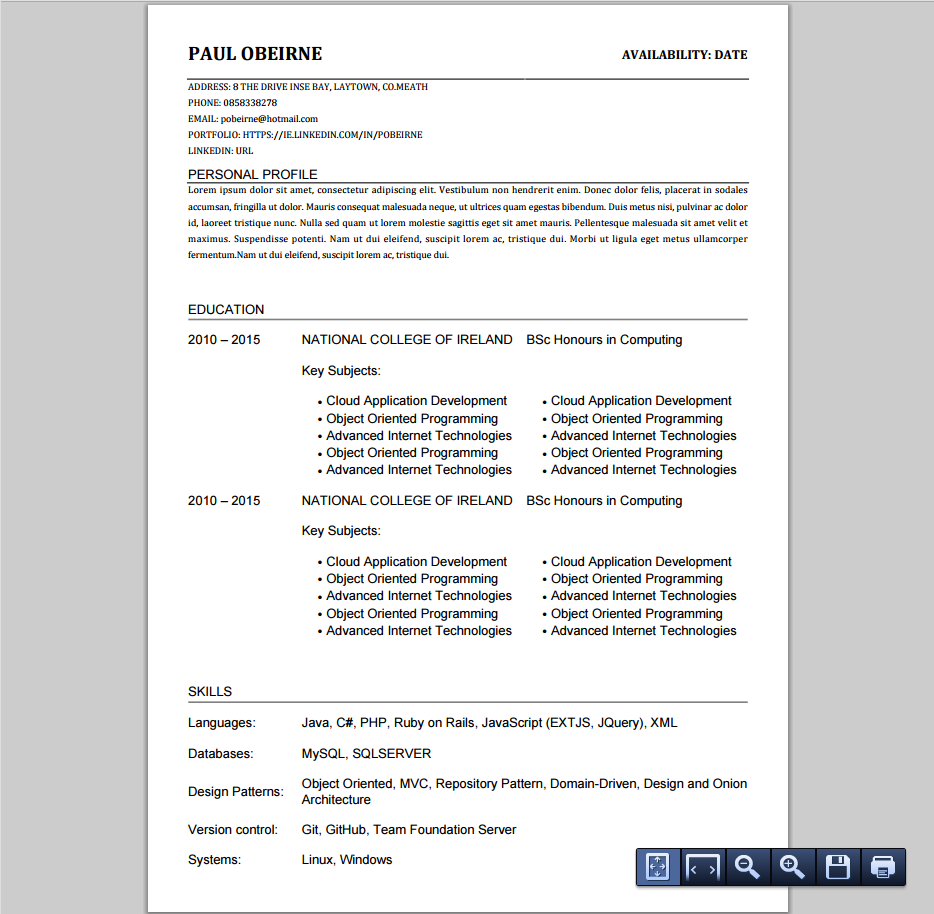


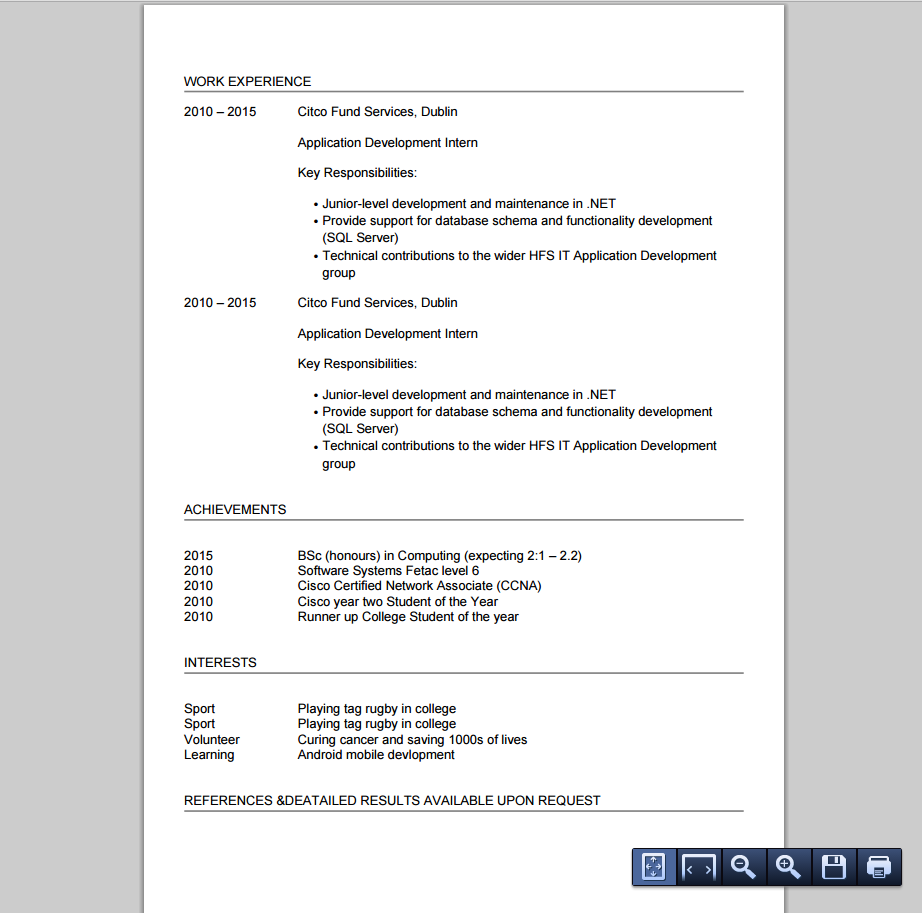
Displays a status overview which determines if the user has sufficient information to be able to generate a Portfolio. If the user doesn’t the generate button will be disable preventing the user from creating a new Portfolio. If the user has sufficient information the user can then select a portfolio template and generate a new portfolio. The initially generate portfolio will be generate based on the restrictions each template has. The users existing portfolio settings will be displayed on the right side. This section will allow the user to change various options on what is displayed on the portfolio web site.

The content section is initially loaded through JavaScript call to the server and can be reloaded using the reload button without reloading the entire page. The user will be able view the portfolio by clicking the view button in which a new window will popup displaying the full site. Other actions can be take such as editing and removing portfolio.

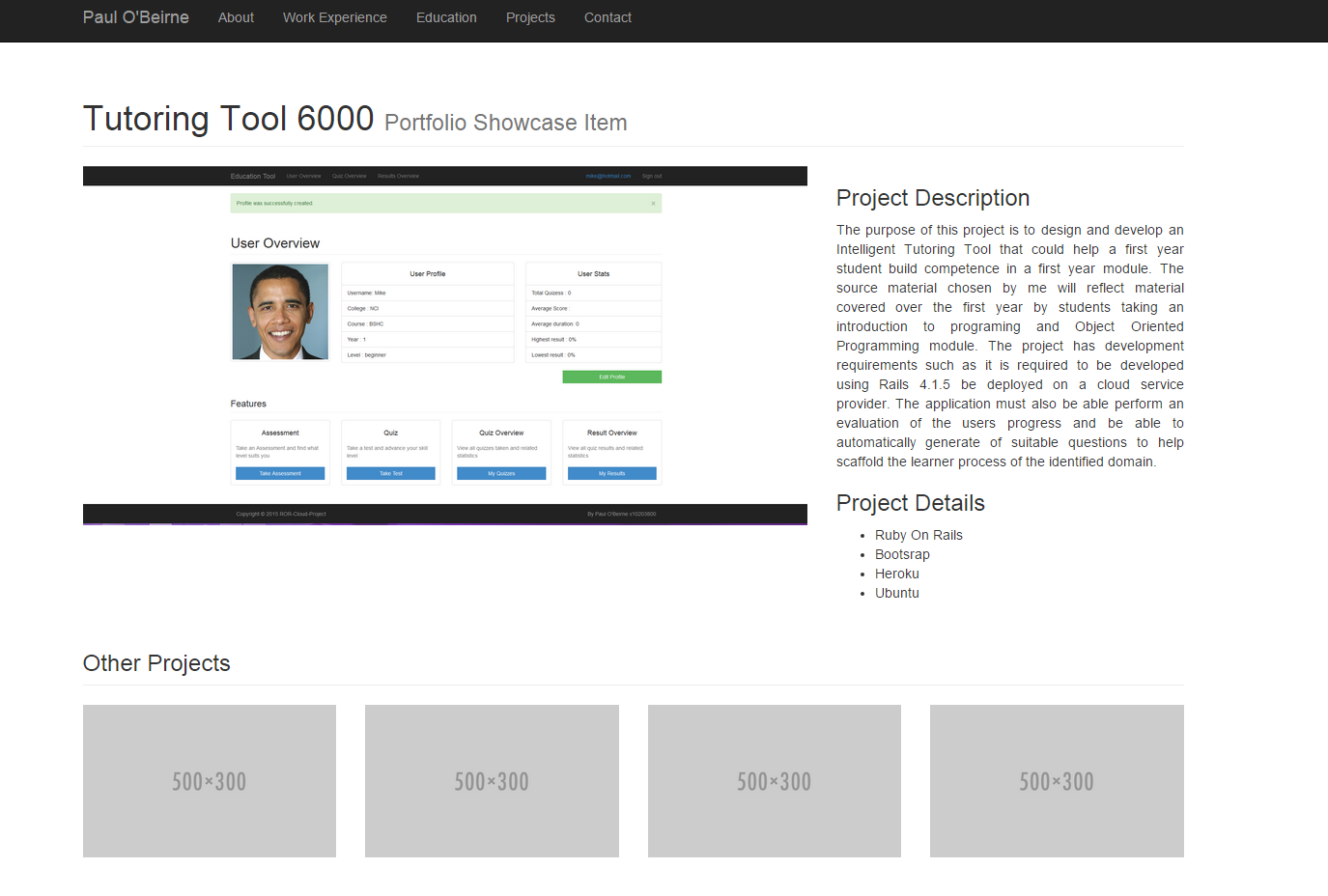
**Generated Content**

CV example 1





Portfolio Example 1



# Conclusions

Overall the development of this application was interesting on a personal level with learning to uses some new and further understand familiar technologies. The advantages of the application is relative to the user and their own expectations of how the wish to be seen by potential employers. The application offers a very simple implementation that can greatly help those with sufficient knowledge to create their own portfolio site as well as help those who simply wish to tailor multiple versions of CV’s.

The development process of this project was under a very short time constraint which limited me in creating more robust functionality and a better user experience. The customisation could be future developed to allow more options to the user and additional templates catering to specific field would likely attract more users from a broader range.

# Further development or research

The application has potential if developed with more customization and functionality added. With additional resource’s the project could be incorporated into Moodle or as part of a college’s service and made available to all students. The application would allow the college help all students with their Personal & Professional Development. It could also lesson some of the user input burdens as college information could help populate some of the required data.

The application could also allow the college to control and create custom templates, monitor all students CV and portfolios in one place. The possibility of providing relevant and up to date CV and portfolios for every graduating student will likely increase the opportunities for students in finding work.

# References

Bootstrap · The world's most popular mobile-first and responsive front-end framework. . 2015. Bootstrap · The world's most popular mobile-first and responsive front-end framework. . [ONLINE] Available at: <http://getbootstrap.com/>. [Accessed 30 July 2015].

Microsoft Azure: Cloud Computing Platform & Services. 2015. Microsoft Azure: Cloud Computing Platform & Services. [ONLINE] Available at: <https://azure.microsoft.com/en-us/>. [Accessed 30 July 2015].

NLog. 2015. NLog. [ONLINE] Available at: <http://nlog-project.org/>. [Accessed 30 July 2015].

Ninject - Open source dependency injector for .NET. 2015. Ninject - Open source dependency injector for .NET. [ONLINE] Available at: <http://www.ninject.org/>. [Accessed 30 July 2015].

andyhutch77/MvcRazorToPdf · GitHub. 2015. andyhutch77/MvcRazorToPdf · GitHub. [ONLINE] Available at: <https://github.com/andyhutch77/MvcRazorToPdf>. [Accessed 30 July 2015].

Understanding Onion Architecture - Chetan Vihite's Blog. 2015. Understanding Onion Architecture - Chetan Vihite's Blog. [ONLINE] Available at: <http://blog.thedigitalgroup.com/chetanv/2015/07/06/understanding-onion-architecture/>. [Accessed 30 July 2015].

Onion Architecture in ASP.Net MVC - CodeProject. 2015. Onion Architecture in ASP.Net MVC - CodeProject. [ONLINE] Available at: <http://www.codeproject.com/Articles/808400/Onion-Architecture-in-ASP-Net-MVC>. [Accessed 30 July 2015].