**Portfolio**

The final due date is Friday 21 June (11:59pm).

This assessment is worth 75% of your course grade. It is a must-pass assessment.

**Overview**

There are three main websites that need to be created for your portfolio:

1. A website that is built with the Bootstrap framework.

2. An e-commerce store.

3. A Django web application.

Details on each of these will be given here on other subtabs at the appropriate time. Do not start any item until you have read the requirements.

* **Conditions**

Read the main conditions and information about extensions on the Assessments topic (tab) here on Moodle.

A reminder: You must submit your own work. Please do not share your work with others or help them with their assessed work. Similarly, you cannot use AI to help with this assessed work - if we suspect that AI-generated work has been submitted we will follow up with an interview in which you will need to be able to explain your coding, etc.

* **Infrastructure Requirements**

The technical requirements for the assessment are:

* + The websites should be running on an Amazon Web Services EC2 instance.
  + The websites should have their own domain names. You should use subdomains of your main domain, as explained in class.
  + The websites should be served by Apache over HTTPS.
  + The data for the e-commerce store and Django web application should be stored in a MySQL database.
  + The websites should be in their own directories inside /var/www.
  + The Django web application should have its own Python virtual environment and Django installation (do not use any existing ones you might have - this will help prevent problems).
* **Marks Allocation**

1. Bootstrap Website (20 marks)

2. E-Commerce Store (5 marks)

3. Django Web Application (75 marks)

TOTAL: 100 marks

* **Portfolio Submission**

Read all instructions here carefully before submitting your portfolio. If any work is missing you will get 0 marks for that item.

**Overview**

NOTE: You must keep your AWS EC2 instance running until you have received a confirmed course result, with all websites accessible (make sure you renew the domains and SSL certificates if necessary - keep an eye on your email for messages about renewal). Marking is done by assessing your live websites, which requires login access to your instances. This means you must not make any more changes to your websites on your instances (I will be checking for file modification, etc.). What you are submitting as per the instructions below is a copy of your work, so that Ara has evidence of what you did if an audit of assessments is undertaken.

You need to submit the required items described below. The items should all be put into a single folder, which is zipped into an archive and then uploaded to the drop box.

**Folder naming and compression/archiving**

Name the folder **exactly**like the following example (exactly means what it means, even down to commas and spaces!) - your lecturer gets Grumpy (with a capital G) having to correct about half of submitted file/folder names every single time students submit work. Name the **folder**like this **before**zipping it:

YourLastName, YourFirstName BCDE215 Portfolio

Folders can't be uploaded to Moodle, so create a compressed archive of the folder and submit that. The only acceptable file type for the archive is zip. Other file types will be rejected by Moodle.

**Items to submit**

Note that instructions to help you generate and download some required items are given in the next section. This section outlines what you need to submit, not how to get the items.

Put all these items into your submission folder:

* + A copy of your Bootstrap website (the whole directory - it can be zipped, as below, if necessary).
  + A copy of your e-commerce website (the whole directory - it can be zipped, as below, if necessary).
  + A copy of your Django project (the whole directory- it can be zipped, as below, if necessary). Note: if your venv directory is outside the project directory, you do not need to include it; if the venv is inside your project directory, just leave it there and include it in your submission.
  + A copy of the MySQL databases used by your e-commerce and Django websites.
  + Copies of all relevant Apache website configuration files - put these in a subfolder in your main portfolio folder.
  + A copy of your EC2 instance's private key in .ppk format.
  + A document that provides:
    - The public IP address of your AWS EC2 instance.
    - The domain names for each website, clearly indicating which domain name is for which website.
    - The names of the Bootstrap JS interface features (components) you implemented (e.g., "Carousel").
    - The admin login details for your Django web app - that is, the superuser account created for your lecturer. (See the first Django web app requirement.)
    - The Django web app's user account details.
    - A list of resources you used to help you build the Django web app - these could be web pages, books, etc. (See above.)

**How to get some of the items you need to submit**

Copy each of your website directories from your EC2 instance to your computer using WinSCP. The WordPress and Django website directories might take ages to copy because there are a lot of files. If you want to speed this up you could tar and gzip the directories before downloading them, change into your home directory and then use a command like:

sudo tar -zcvf your-project-website-name.gzip /var/www/path/to/your/website

Create dumps (backups) of your MySQL databases (obviously use your own database names and filenames here) -

* + Change into your home directory if you aren't already in it, so that the database dumps will be created there: cd /home/ubuntu
  + Dump a database - use this command if your MySQL root user does not have a password set: sudo mysqldump your-database-name > your-database-name.sql
  + Dump a database - use this command if your MySQL root user has a password set (you used mysql-secure-installation at setup time): sudo mysqldump -u root -p your-database-name > your-database-name.sql
  + Use WinSCP (etc.) to copy the TWO database dump files from your instance to your computer.

Apache configuration files - use WinSCP (etc.) to copy the config files to your computer.

**Bootstrap Website**

Download the file below. Use Bootstrap to apply CSS to the HTML file in the download. The page should have an attractive appearance and make use of Bootstrap's responsiveness. Make sure you are using Bootstrap version 5.3 (check your download or <link> carefully).

Implement **two**JavaScript interface features such as mobile menu, modal, carousel, collapse (see <https://getbootstrap.com/docs/5.3/getting-started/javascript/>).

You can add more items (images, text, HTML elements) to the page if you need to in order to implement your JavaScript interface feature.

You can alter (minimal alterations) or add to the structure of the HTML if you need to, but you cannot replace the HTML with a template, such as a Bootstrap starter template.

This is a one-page website and needs to be served by Apache on your AWS EC2 instance - see Infrastructure Requirements on the Overview tab.

**E-Commerce Store**

Create a small web store using WordPress and WooCommerce. The specific requirements are:

1. The web store should be created to sell a particular category of product (e.g., electronic goods).

2. Enter three products in the web store. It would be safest not to have products actually for sale, so make sure it is clear to users that no products will actually be sold (e.g., set the quantity on hand to zero?).

3. Install a theme suitable for a web store and suitable for the type of products in your web store. NOTE: You cannot simply leave the default theme installed.

4. See Infrastructure Requirements on the Overview tab.

**Django Web Application**

* This task is to create a Django-based web application.

The Django web app should have its own Python virtual environment and Django installation (do not use any existing ones you might have - this will help prevent problems).

You are building an online music catalogue for a group of friends called Music Collectors. They have a large collection of music albums. They want friends to be able to see what they have.  
  
Here are the requirements for the web application:  
  
Website Name: Music Collectors Catalogue  
  
You need to build a Django web application that stores information about albums and artists. An album is created by only one artist. An artist can have multiple albums.  
  
An artist has an ID, a name, and a country.  
  
An album has a name, a release date, a genre, a producer name (a person), and a record label name. It also has a comment field, which should allow for a very long comments - over 500 words, for example.  
  
You need to create a custom web application for members of Music Collectors (not just set up the default Django admin web application). The web application must force users to log in to create, update, and delete artists and albums. Other users of the website (the public) have read-only access to the pages of the website. Access to the home page does not require logging in at all.  
  
Requirements for the design and content of the website for the web application:

* + The main menu should have links to Home, Artists, Albums, and Admin Login. These links go to:

        Home: a page that has a welcome message, a relevant image of your choosing and some brief introductory text (of your creation/finding).  
        Artists: a list of all artists.  
        Albums: a list of all albums.  
        Admin Login: the Django admin area.

* + The website name must appear in the header of the web pages.
  + Login/logout links should appear in the header of the web pages.
  + The web application should be constructed with your own HTML and CSS. The code should follow principles shown in class (e.g., CSS grid layout). Do not use frameworks, such as Bootstrap.
  + The web application should have a tidy/professional design.
  + The web application must allow members of Music Collectors to create, update and delete artists and albums. As stated above, these actions are not to be done in the default Django admin app - you need to add the necessary custom views.
  + The list page for artists (which is a list of all artists) should show just the id and name of each artist. None of the other data fields should be on that page. It should be possible to click an artist in the list to go to their detail page.
  + The list page for albums (which is a list of all albums) should have the album name, artist name and release date for each album. None of the other data fields should be on that page.
  + The detail page for an artist should show all of an artist's details and also show a list of their albums. Do not show the album notes, which would be too long to display in a list view. It should be possible to click on an album in the list on this page to go to its detail page.
  + The detail page for an album should show all an album's data and also show the name of the related artist on it.
* **Other Django Web App / Assessment Requirements**
  + You need to provide a list of resources you used to help you (e.g. web page URLs).
  + Create an admin user account for your lecturer - it must be a superuser and be able to log in.
  + Enter five sample artists in the system, each with at least two albums, so that the website's appearance with data can be seen (e.g., its responsiveness).
  + The web app should be responsive and display appropriately on a range of devices.
  + You should use HTML and CSS that follows the principles and methods we have covered in class.