Homework 03 – HTML, CSS, JavaScript, and Responsive web design WITHIN and MVC project (C#)

Out of 110 Marks

DUE: 2022-06-05 before 23:59

IMPORTANT NOTES:

- This is an individual assignment.
- Homework assignments are based on assessment objectives. If an objective has been achieved a mark will be allocated.

INSTRUCTIONS:

- In this assignment, you will be given high-level requirements and can implement them in a context you see fit.
- Follow the instructions on the Study Guide page listed as Assignment Uploading.
- Refer to the following two documents:
  - How to upload a video to YouTube Guide.pdf
  - How to use GitHub in Visual Studios 2019 Guide.pdf
- Name the Repository uXXXXXXXX_HW03 where uXXXXXXXX is your student number.
- Make sure to share the repository with inf272marker as a collaborator (use the following email: **inf272marker@gmail.com**). If you do not share the repository with inf272marker as collaborator, then we will not be able to mark your submission.

NOTE ON DEMO SOFTWARE

- Please change the compression ratio and Frames Per Second (FPS) of the desktop recoding software that you use to reduce the file size of your demo. These details have been noted at the top of the homework submission page.
- Please use desktop recoding software as suggested on the homework submission page.
- DO NOT use your phone to record your demo as it is extremely difficult to follow what is going on in your demo and it creates unnecessarily large files making the video demo upload problematic. We suggested desktop recording software as it simply streamlines the process and creates smaller files that are easier to upload.

SUBMISSION DEADLINE: 2022-06-05 before 23:59

- There shall be no extensions to the deadline.
- If homework submissions are uploaded too late then upload errors WILL happen.
- Do not wait to the last minute to complete the assignment.
- Start working on the assignment as soon as it is posted.
- Verify the completeness of your upload.
- There are multiple upload opportunities enabled if your upload is incomplete.
- Incomplete uploads will be considered unsubmitted work.
- E-mail submissions WILL NOT be accepted.
- Late submissions WILL NOT be accepted.
- **NO EXCEPTIONS WILL BE MADE FOR ANYONE.**

IMPORTANT

- We download your program source code to make sure that you do not upload empty programs.
- In the past we have encountered people demoing other people's programmes. That is considered academic dishonesty and there are severe consequences if you are found to have committed academic dishonestly. Please refer to the student code of conduct regarding the consequences of academic dishonestly.

1. TOOLS, SKILLS, KNOWLEDGE REQUIRED:

- To complete his homework assignment, you will need to make use of MVC, HTML, JavaScript, CSS, and Bootstrap.
- You must create a MVC project using Visual Studio to complete the project.
- You will need to make use of Bootstrap for the content and Layout of the pages.

2. USE CASE:

- For this assignment you will be creating a media uploading system. You will need to code four main sections of the program:
    - Home page: The home page will have a form where users will need to select whether they want to upload a **File**, **Image**, or **Video**. Users will then be able to select the file on their computer which will then be uploaded on submit of the form.
    - Files page: The files page must display a list of all the **Files** that a user has uploaded with the functionality to **delete** and **download** the file.
    - Images page: The images page must display all the images (**the images must be viewable**) that the user uploaded, in an orderly way with the functionality to **delete** and **download** the images.
    - Videos page: The videos page must display all the videos (**the videos must be playable**) that the user uploaded in an orderly way, with the functionality to **delete** and **download** the videos.
    - About Me: The about me page should display a biography along with a picture (image) of yourself (this is helpful as this project will be uploaded to your GitHub repository that you will be able to share down the line).
- All the files you upload (files, images, and videos) should be stored in the main MVC project solution with the following folder structure:
  >> Media
  >> Documents
  >> Images
  >>Videos

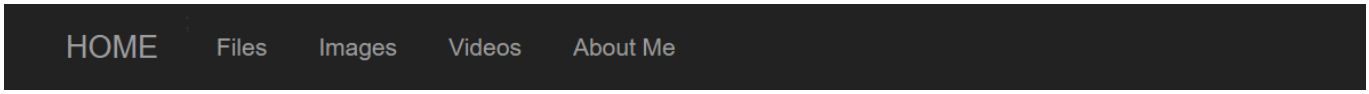
3. STANDARD REQUIREMENTS:

The following is a list of mandatory requirements.

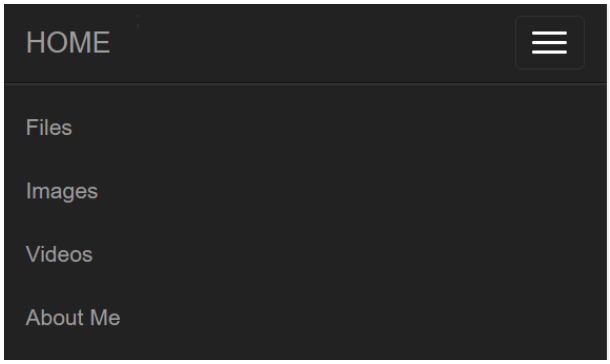
- You must make use of one **Model** in this project.
    - Create a model called **FileModel** which must include the following attribute:
        - FileName (string)
    - The Model should make use of decorators to ensure that the data is displayed as aesthetically pleasing text and not the variable names used in the Model.

- NavBar
    - The NavBar must include links to all the different **views** of the system. You must include a **link** to the **Home** view, **Files** view, **Images** view, **Videos** view, and the **About Me** view.
    - You must make use of **Boostrap** to make the NavBar **responsive** (it should be **collapsible**).

See the following image below for an example of what the NavBar could look like:



See the following image for an example of what the **collapsed** NavBar could look like:

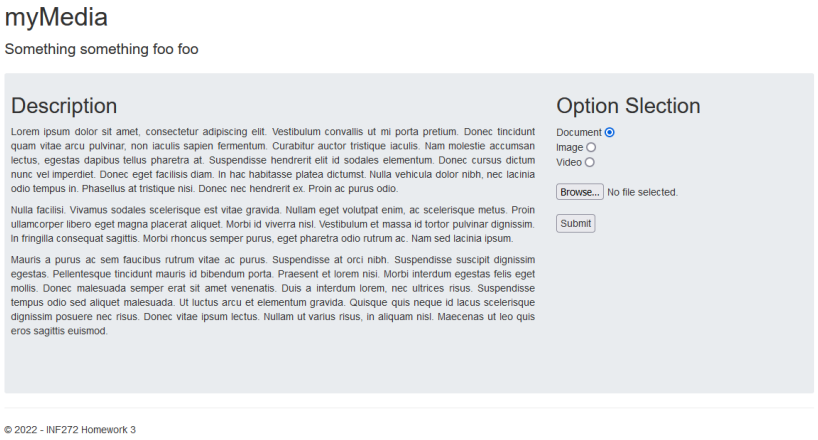


- Home Page
    - Index view requirements
        - A heading for the view
        - A sub-heading for the view
        - A description for the view which should explain the system you are building.
        - A form which should include the following:
            - Three radio buttons (one for **Document**, one for **Image**, and one for **Video**) which the user will use to select the type of file they want to upload to the system.
            - A **Choose File** button which the user will use to select the file they want to upload from their computer.
            - A **Submit** button which the user will use to submit the form and upload the selected file to the system.
        - The **Option Selection** form must be displayed next to the description (see the example image below)
        - The view must be **Responsive** i.e., the page must resize properly to fit different screen types (make use of **Bootstrap)**

    - About view requirements
        - A heading for the view
        - A sub-heading for the view
        - A biography paragraph. This should be your biography (this is helpful as this project will be uploaded to your GitHub repository that you will be able to share down the line)
        - A picture (image) of you (this is helpful as this project will be uploaded to your GitHub repository that you will be able to share down the line).
        - Your biography and picture should be displayed side-by-side (make use of **Bootstrap**)
        - See the example image of the About View below

    - Home controller requirements
        - The Home controller must have the following Action methods:
            - Index ActionResult (GET): This action must return the Index view of the Home controller.
            - Index ActionResult (POST): This action must receive the **file** and the **radio button option selected** that has been posted by the form on submit in the Home view. You must then save the file in the correct folder (see the folder structure discussed above) and then redirect to the Home page view of the Home controller.
            - About ActionResult: This action must return the About page view (discussed below).

See the following image for an example of what the Home controller Index view could look like:

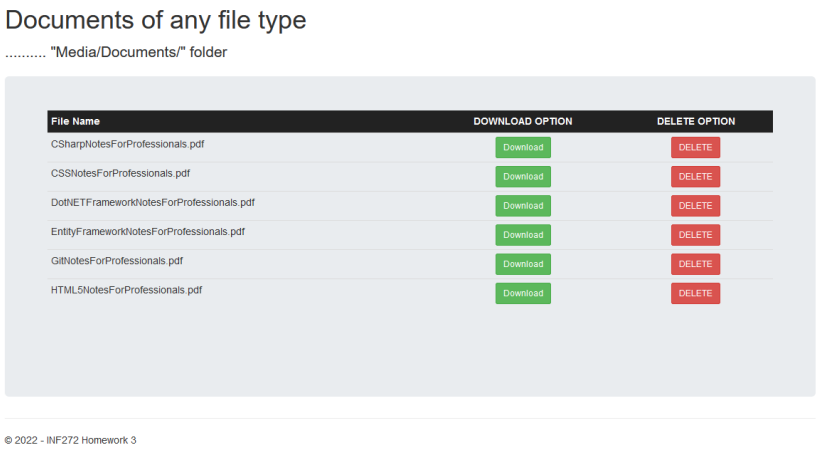


See the image below for an example of what the Home controller About view could look like:

- Files Page
    - Index view requirements
        - This view should receive a list of **FileModel**.
        - A heading for the view.
        - A sub-heading for the view.
        - A **Bootstrap** table with the following headings: **File Name**, **Download Option**, and **Delete Option**. The table must display all the **File** types that the user has uploaded using the system.
            - The **File Name** column should display the name of the file uploaded including the extension.
            - The **Download Option** column must have a **Download** button for each file name.
            - The **Delete Option** column must have a **Delete** button for each file name.
        - When the **Download** button is clicked the file must be downloaded to the user's computer.
        - When the **Delete** button is clicked the file must be deleted from the system (the file will be removed from the folder in the project solution).

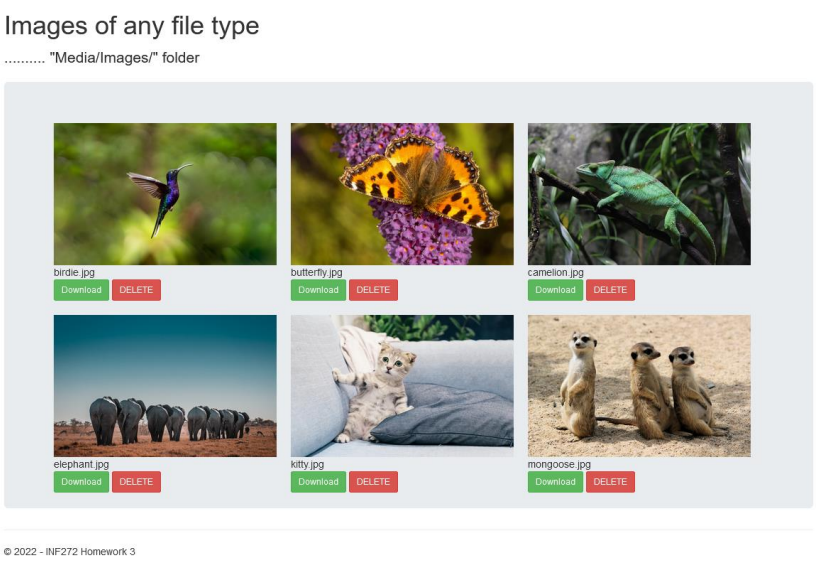
    - File controller requirements
        - The File Controller must have the following Action methods:
            - Index ActionResult: This action should retrieve all the files that have been uploaded by the user and return them as a list to the Index view of the File controller for display in the table (you will need to make use of the **FileModel** to accomplish this).
            - DownloadFile FileResult: This action should receive the file name of the file that the user would like to download and should then download the file onto the user's computer. The action should return a File.
            - DeleteFile ActionResult: This action should receive the file name of the file that the user would like to delete and should then delete the file from the system. The action should redirect to the index view of the File controller.

See the following image for an example of what the File controller Index view could look like:

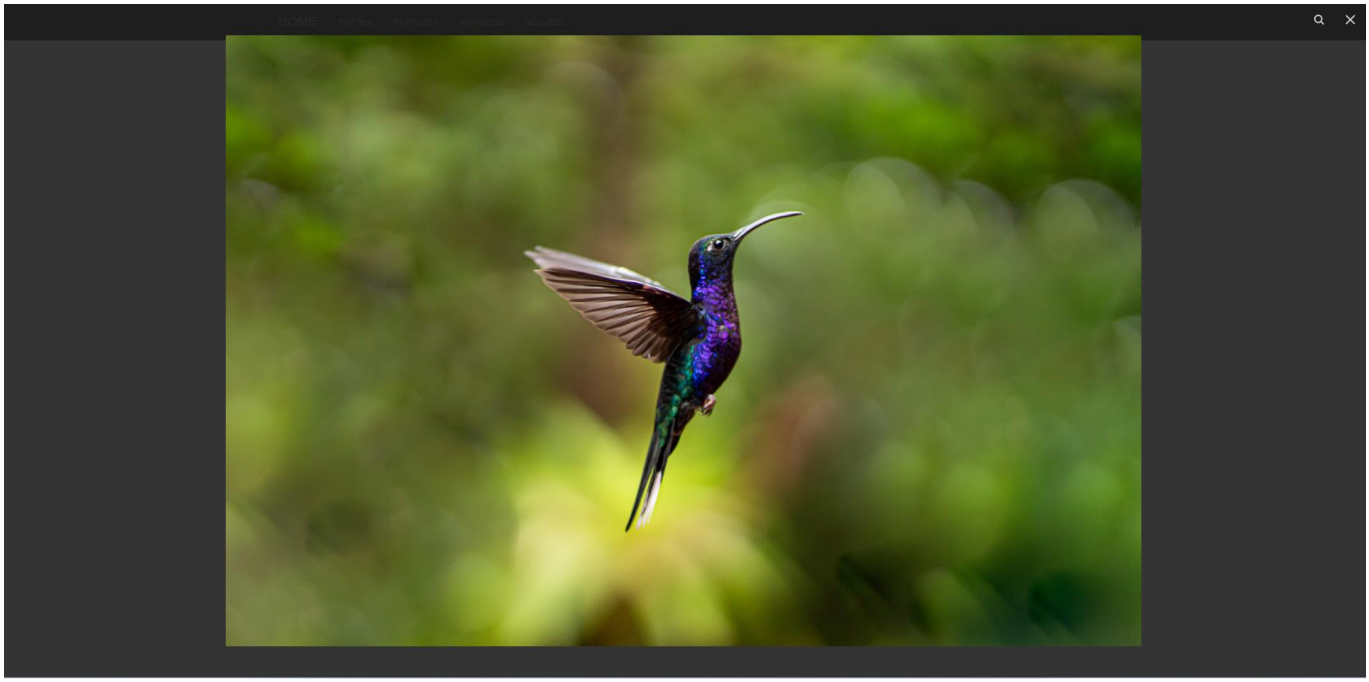


- Images Page
    - Index view requirements
        - The view should receive a list of **FileModel**.
        - A heading for the view
        - A sub-heading for the view
        - All the image files that the user has uploaded to the system should be retrieved and displayed in the view in an ordered way.
        - The filename of the image (including the file extension) should be displayed below the image.
        - Below the filename of each image, you need two buttons:
            - A **Download** button that will download the image to the user's computer.
            - A **Delete** button that will delete the image from the user's computer (the image file will be removed from the folder in the project solution)
        - A scroll bar should appear if there are too many images displayed (see the example image below).
        - When a user clicks on an image, the image should pop out onto the screen by making use of Fancy Box (see the example image below).
    - Image controller requirements
        - The Image controller should have the following Action methods:
            - Index ActionResult: This action should retrieve all the image files that have been uploaded by the user and return them as a list to the Index view of the Image controller for display in the view (you will need to make use of the **FileModel** to accomplish this).
            - DownloadFile FileResult: This action should receive the file name of the image file that the user would like to download and should then download the image file onto the user's computer. The action should return a File.
            - DeleteFile ActionResult: This action should receive the file name of the image that the user would like to delete and should then delete the image file from the system. The action should redirect to the index view of the Image controller.

See the following image for an example of what the Image controller Index view could look like:

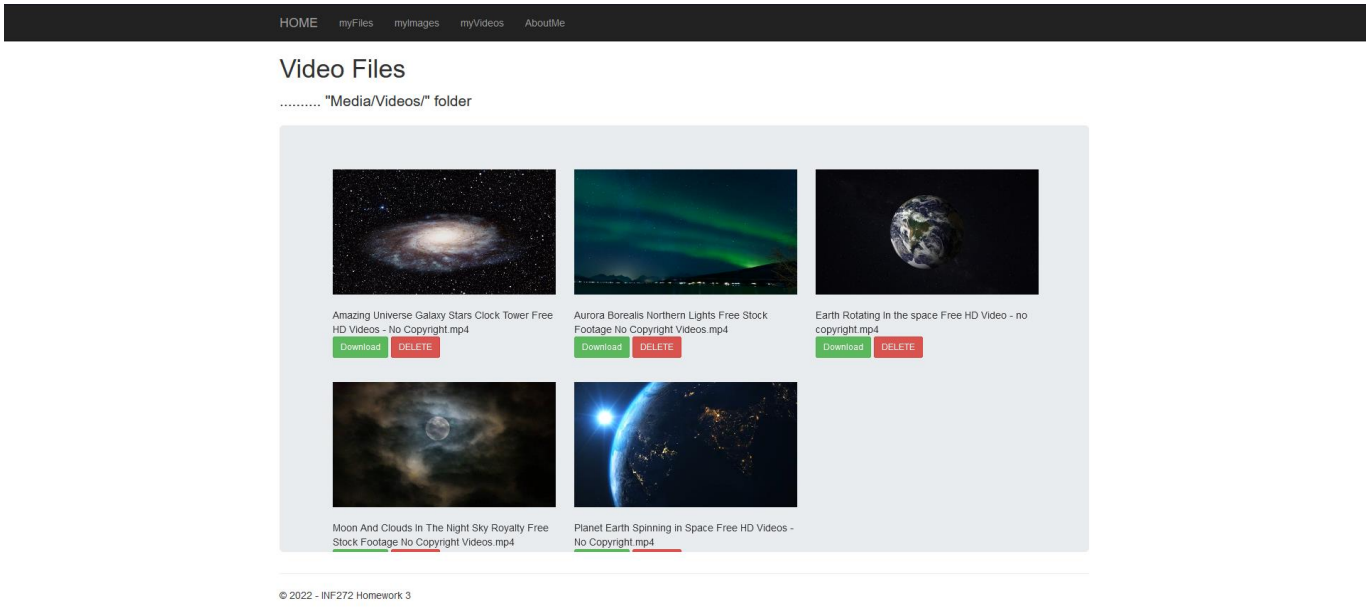


See the following image for an example of what the Fancy Box Image on the Index view could look like:



- Videos Page
    - Index view requirements
        - The view should receive a list of **FileModel.**
        - A heading for the view.
        - A sub-heading for the view.
        - All the video files that the user has uploaded to the system should be retrieved and displayed in the view in an ordered way.
        - The filename of the video (including the file extension) should be displayed below the video.
        - Below the filename of each video, you need two buttons:
            - A **Download** button that will download the video to the user's computer.
            - A **Delete** button that will delete the video from the user's computer (the video file will be removed from the folder in the project solution)
        - A scroll bar should appear if there are too many videos displayed (see the example image below).
        - When a user clicks on a video, the video should pop out onto the screen by making use of Fancy Box (see the example image below).
    - Video controller requirements
        - The Video controller should have the following Action methods:
            - Index ActionResult: This action should retrieve all the video files that have been uploaded by the user and return them as a list to the Index view of the Video controller for display in the view (you will need to make use of the **FileModel** to accomplish this).
            - DownloadFile FileResult: This action should receive the file name of the video file that the user would like to download and should then download the video file onto the user's computer. The action should return a File.
            - DeleteFile ActionResult: This action should receive the file name of the video that the user would like to delete and should then delete the video file from the system. The action should redirect to the index view of the Video controller.

See the following image for an example of what the Video controller Index view could look like:



See the following image for an example of what the Fancy Box Video on the Index view could look like:

4. RESEARCH / READING REQUIREMENTS

The following is required reading and / or research:

- How to pass a File type from a view to a controller Action method (what should be in the parameters?).
- How to save a file to specific path in the controller.
- C# lists.
- How to download (return) a file to a user's computer from a controller.
- Using model lists in MVC views.
- Setting MIME types in MVC.
- Configuring httpRuntime for downloads.
- Fancy Box
    - Below is the jQuery for the fancy box. This will assist you in implementing Fancy Box in the Image and Video index view.

```
<script>
    $(document).ready(function ($) {
        $(".gallery a").fancybox();
    });
</script>
```

Homework 03 – HTML, CSS, JavaScript, and Responsive web design WITHIN and MVC project (C#)

Out of 110 Marks

DUE: 2022-06-05 before 23:59

VIDEO INSTRUCTIONS:

- Make sure that everything is running when you start recording the video. The video should not be longer than **15** minutes showing the items in the Checklist.
- **Simply scrolling through the program will be considered not presenting a demo which would equal to you immediately losing 50% of your mark. You need to explain your reasoning.** It is part of what is known as "reflective learning" and helps you solve problems when you encounter then during coding. It is extremely useful. Do not neglect your own leaning process by skipping out on this important step
- When showing something in the Checklist, show us your code and explain why you did it in that way. If you prefer to demo everything first then explain code, you can do it that way, but we need an explanation of why you coded it that way and not just functionality. The explanation does not need to be too in-depth.
- If something did not work in your code, in the video explain to us what you wanted to do and what you wanted to achieve with your approach. **We will not mark your code that you show us unless you explain this**.

IMPORTANT NOTES:

- Both the Video Demo and Homework Source Code should be submitted in the correct upload area. **If the Video demo or the Homework Source Code is missing, you will get a zero.**
- If files are upload to the wrong upload area, we will not go and look for the upload. Uploads should be submitted correctly as indicated online. Incorrect uploads will lead to a zero being allocated.
- If you are caught for plagiarism, we will give you 0 and you will be reported for plagiarism immediately. There are no more warnings this year. We will audit historical assignments throughout the semester. You are here to learn, the more you know, the better for you.

Checklist Use the checklist as a script that would allow you to sequence your demonstration.	**MAX**
1. Requirement 1 – Functionality • Start by showing the functionality of the program. • Run the program and then go though one complete action. **Demonstration time allocation = 1 minutes**	**10**
2. Requirement 2 – Model • FileModel (**2**: 2 marks for model correctly created according to instructions) **Demonstration time allocation = 1 minutes**	**2**
3. Requirement 3 – NavBar • NavBar (**3**: 1 mark for navbar being responsive (collapsible), 1 mark for navbar having correct links according to instructions, 1 mark for links navigating to correct view) **Demonstration time allocation = 1 minutes**	**3**

4. Requirement 4 – Home & About
• Controller (**7**)
  ○ Index ActionResult (GET) (**1**: 1 mark for returning the Index view)
  ○ Index ActionResult (POST) (**5**: 2 marks for receiving the **option** and the **file** from the view, 3 marks for saving the file in the correct folder)
  ○ About ActionResult (**1**: 1 mark for returning the About view)
• Index view (**9**)
  ○ Layout & Content (**3**: 1 mark for description and form positioned next to one another, 1 mark for correct content according to instructions, 1 mark for page being responsive and resizing to fit different screen types)
	19

Checklist Use the checklist as a script that would allow you to sequence your demonstration.	**MAX**
○ Form (**6**: 3 marks for working form, 1 mark for radio buttons, 1 mark for working file upload button, 1 mark for working submit button) • About view (**3**) ○ Layout & Content (**3**: 1 mark for biography and picture positioned next to one another, 1 mark for correct content according to instructions, 1 mark for page being responsive and resizing to fit different screen types) **Demonstration time allocation = 3 minutes**	
5. Requirement 5 – Files • Controller (**12**) ○ Index ActionResult (**4**: 1 mark for retrieving the files and storing the data in an array, 2 marks for adding to list of FileModel using the retrieved data, 1 mark for returning the list to the view) ○ DownloadFile FileResult (**4**: 1 mark for retrieving the filename to be downloaded from the view, 3 marks for downloading the correct file) ○ DeleteFile ActionResult (**4**: 1 mark for retrieving the filename to be downloaded from the view, 3 marks for deleting the correct file) • Index view (**11**) ○ Layout & Content (**2**: 1 mark for page being responsive and resizing to fit different screen types, 1 mark for correct content according to instructions) ○ Model (**1**: 1 mark for making use of model in view) ○ Table (**3**: 1 mark for table, 2 marks for displaying the files passed to the view in the table using model) ○ Buttons (**5**: 1 mark for download and delete button displayed for each file in the table, 2 marks for buttons calling correct action [1 mark per button], 2 marks for buttons working [1 mark per button]) **Demonstration time allocation = 3 minutes**	**23**
6. Requirement 6 – Images • Controller (**12**) ○ Index ActionResult (**4**: 1 mark for retrieving the image files and storing the data in an array, 2 marks for adding to list of FileModel using the retrieved data, 1 mark for returning the list to the view) ○ DownloadFile FileResult (**4**: 1 mark for retrieving the filename to be downloaded from the view, 3 marks for downloading the correct image file) ○ DeleteFile ActionResult (**4**: 1 mark for retrieving the filename to be downloaded from the view, 3 marks for deleting the correct image file) • Index view (**14**) ○ Layout & Content (**2**: 1 mark for page being responsive and resizing to fit different screen types, 1 mark for correct content according to instructions) ○ Model (**1**: 1 mark for making use of model in view) ○ Images displayed (**4**: 2 marks for displaying the images in an orderly manner using model (see example image), 1 mark for scroll bar if images overflow, 1 mark for all images being the same size) ○ Buttons (**5**: 1 mark for download and delete button displayed for each image file, 2 marks for buttons calling correct action [1 mark per button], 2 marks for buttons working [1 mark per button]) ○ Fancy Box (**2**: 2 marks for implementing fancybox in the view. (See example image). **Demonstration time allocation = 3 minutes**	**26**
7. Requirement 7 – Videos • Controller (**12**) ○ Index ActionResult (**4**: 1 mark for retrieving the video files and storing the data in an array, 2 marks for adding to list of FileModel using the retrieved data, 1 mark for returning the list to the view) ○ DownloadFile FileResult (**4**: 1 mark for retrieving the filename to be downloaded from the view, 3 marks for downloading the correct video file) ○ DeleteFile ActionResult (**4**: 1 mark for retrieving the filename to be downloaded from the view, 3 marks for deleting the correct video file) • Index view (**15**)	**27**

Checklist Use the checklist as a script that would allow you to sequence your demonstration.	**MAX**
○ Layout & Content (**2**: 1 mark for page being responsive and resizing to fit different screen types, 1 mark for correct content according to instructions) ○ Model (**1**: 1 mark for making use of model in view) ○ Videos displayed (**4**: 2 marks for displaying the videos in an orderly manner using model (see example image), 1 mark for scroll bar if videos overflow, 1 mark for all videos being the same size) ○ Buttons (**5**: 1 mark for download and delete button displayed for each video file, 2 marks for buttons calling correct action [1 mark per button], 2 marks for buttons working [1 mark per button]) ○ Fancy Box (**3**: 3 marks for implementing fancybox in the view. (See example image). **Demonstration time allocation = 3 minutes**	
TOTAL MARK ALLOCATION	**110**