

## ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

(Note : This version is to be used for an assignment brief issued to students via Classter)

Course Title	Enterprise Programming			Lecturer Name & Surname	Ryan Attard	
Unit Number & Title		ITSFT-506-1616 - Enterprise Programming				
Assignment Number, Title / Type		1 - Building An Enterprise Application using a clean architecture				
Date Set		15/11/2021	Deadline Date	3/1/2022		
Student Name			ID Number		Class / Group	

	Assessment Criteria	Maximum Mark
KU1.1	Describe what is meant by software architecture and the role that it offers while expressing it in practice	5
KU1.2	Interpret what is meant by refactoring	5
KU2.1	Show correct use of software design pattern	5
KU3.4	Find and select a perspective of versioning, configuration and scalability of the solution	5
AA2.3	Select and implement appropriate design patterns to a solution being implemented	7
AA3.2	Choose and develop the correct delivery model	7
AA4.2	Illustrate what methods can be used to upload content onto cloud services	7
AA4.3	Use and draw practical application to upload application content to cloud services	7
SE1.3	Construct and Ascertain that enterprise standards fit within an enterprise solution	10
SE3.3	Identify and design security in data integrity features	10
	<b>Total Mark:</b>	<b>68</b>

**Notes to Students:**

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (<http://mcast.classter.com>) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Unicheck will be requested to confirm online the following statements:

**Student's declaration prior to handing-in of assignment**

- ❖ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

**Student's declaration on assessment special arrangements**

- ❖ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- ❖ I declare that I refused the special support offered by the Institute.

**Assignment Guidelines**

Read the following instructions carefully before you start the assignment. If you do not understand any of them, ask your invigilator.

- This assignment is a HOME assignment.
- Fill in and print/scan the assignment sheet.
- Copying is **Strictly Prohibited** and will be penalised according to disciplinary procedures.
- Deadline: See front page
- This assignment has a total of 68 marks. Follow the rubric at the back for detailed assessment guidelines.
- Submission must be done by inputting your Git repository link or Moodle.

### Home Assignment: Building an Asynchronous File Transfer Website

Note: The assignment is composed of a number of Tasks each of which is linked to a number of criteria which ultimately all contribute to one final product. It is divided in two parts – Part 1 and Part 2. Part 1 is basic stuff which should be enough for an anonymous person to make use of the basic functionalities, while Part 2 will require authentication to provide more services.

#### Part 1

1. (KU3.4) – *Find and select a perspective of versioning, configuration and scalability of the solution [5]*

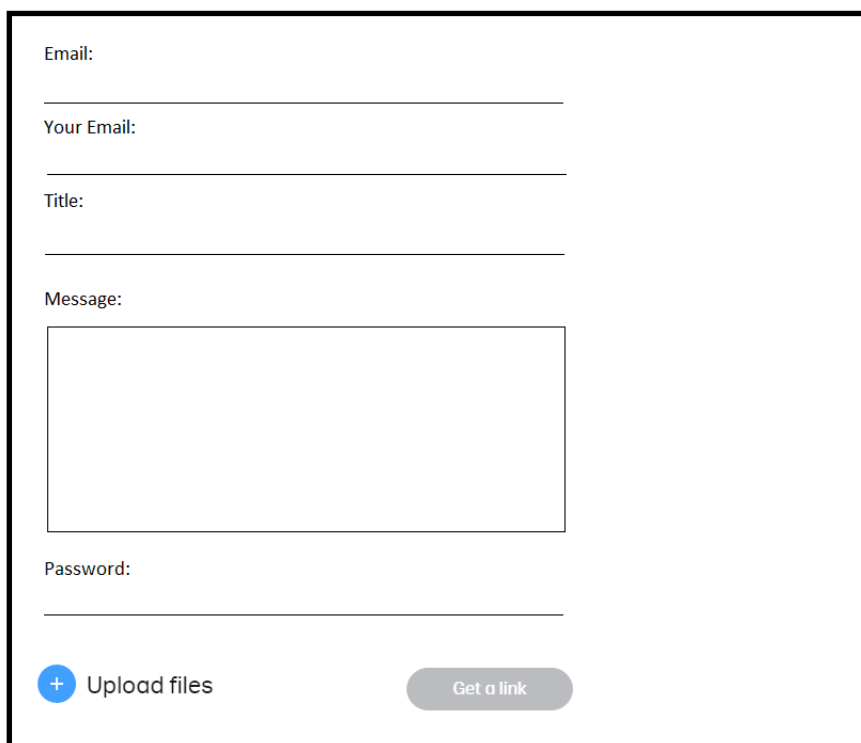
Application has to be configured and stored on a git repository, keeping all versions it went throughout development till deployment. Git repository must show evidence that you have been continuously updating the solution with different versions.

2. (KU1.1) – *Describe what is meant by enterprise software architecture and the role that it offers in practice [5]*

Create a .NET Core Web Application which consists of at least the following projects:

- a. **Domain** contains all the models you will be using in your application
  - b. **Data** contains all the repository and context classes you will make use of to interact directly with your database
  - c. **Application** contains your services and view-models
  - d. **Presentation**
3. (AA4.3) - *Use and draw practical application to upload application content to cloud services [7]*

Look at this sketch



The sketch shows a web form for uploading files. It includes the following elements:

- Email:** A label followed by a text input field.
- Your Email:** A label followed by a text input field.
- Title:** A label followed by a text input field.
- Message:** A label followed by a large text area for a message.
- Password:** A label followed by a text input field.
- Buttons:** At the bottom, there are two buttons: a blue circular button with a white plus sign labeled "Upload files", and a grey rounded rectangular button labeled "Get a link".

- a) Create a model that will represent all this information in the database. Model should be created in the *Domain* project from KU1.1. Password and Message are not a compulsory field.
  - b) Create a repository class (name it FileTransferRepository.cs) that will hold all your operations/methods that you will do to and from the database, and place it in the *Data* project from KU1.1.
  - c) Create a Controller with the required method(s) to service the above user interface in the *Presentation* project from KU1.1. Note: Correct coding for the internal working of the upload and sending of email feature is not assessed here.
  - d) Create the respective View/Page as designed above in the *Presentation* project from KU1.1.
4. (KU1.2) - *Interpret what is meant by refactoring [5]*
- a) Create and implement a FileTransferContext.cs class which follows the same principles of ApplicationDbContext.cs (the default class) and place it in the *Data* project from KU1.1. Make sure there is a relation between FileTransferContext.cs and the model you created in AA4.3
  - b) Do the necessary changes so that every file transfer that takes place has an automated unique primary key.
  - c) Migrate and Update database accordingly, after having set up the proper connection string in a configuration file.
  - d) Apply Lazy Loading
5. (KU2.1) - *Show correct use of a software design pattern [5]*
- a) Make the required changes to any of the projects mentioned in KU1.1, so that Dependency Injection is applied throughout the whole architecture especially where there are calls to action that are going to affect the database somehow;
6. (AA4.2) - *Illustrate what methods can be used to upload content onto cloud services [7]*
- a) Make the website function properly when it comes to a user uploading a file and storing that file in a folder on the webserver/cloud bucket by calling the classes you created in the above steps;
  - b) Updates should be done in the database as well;
  - c) Filenames should be unique;
7. (AA2.3) - *Select and implement appropriate design patterns to a solution being implemented [7]*
- a) Make use of a 3<sup>rd</sup> party API which allows you to send emails without any issues. Email should be sent to whom the file is intended to be transferred to. Email should not contain the file attached but a link to the file;

- b) Use a 3<sup>rd</sup> party library to zip the file and protect it with a password, only if the password has been set on the input screen; Password should be sent in the email in 6.a;
- c) The recipient can then download the file. If a password was set then he has to input a correct password to open the zip file.

8. (AA3.2) - *Choose and develop the correct delivery model [7]*

Website should be deployed on the cloud or any hosting provider of your choice [e.g. [Asp Hosting \(Trial 60 Days free\)](#)] and configured to run properly.

Part 2

9. (SE1.3) - *Construct and ascertain that enterprise standards fit within an enterprise solution [10]*

- a) Create a model called Log where you have a number of properties which you consider important to compose a log (e.g. message, ip address, user, ...)
- b) Create two Repository classes: LogInDbRepository and LogInFileRepository. The difference is that one will store the logs to the database and one will store the logs in a file, so both will have the same method but different implementation.
- c) Create an interface for both and register both services with the Dependency Injection.
- d) Configure the application in such a way that it allows you (as a developer/system admin) to choose one of the two implementations at any time (even while the application has been deployed to the web server). Hint: this has to be done through appsettings.json.

10. (SE3.3) - *Identify and design basic security in data integrity features [10]*

- a) Make use of Identity features provided to by inheriting IdentityDbContext to activate authentication and registration.
- b) Implement a feature while respecting the design founded in Part 1, whereby an authenticated user can check all the files transferred in the past from or to him/her.
- c) Feature can only be accessed by authenticated users to check their files only.

Criterion	Description	Met/Not Met
KU1.1	Architecture must be composed of at least the projects mentioned above. Not following the architecture devised in class will result in loss of all marks	
KU1.2	<ul style="list-style-type: none"> <li>• FileTransferContext.cs [2]</li> <li>• Unique primary key [1]</li> <li>• Database structure is acceptable [1]</li> <li>• Apply Lazy Loading [1]</li> </ul>	
KU2.1	<ul style="list-style-type: none"> <li>• Dependency Injection applied properly everywhere. 1 mark will be deducted for every instance created not respecting DI design pattern</li> </ul>	
KU3.4	Git repository must show a minimum of 7 commits illustrating the following information. Not having at least 7 commits will result in loss of all marks	
AA2.3	<ul style="list-style-type: none"> <li>• Emails successfully sent using 3<sup>rd</sup> party API; [3]</li> <li>• Password protected files (upload and download) [4]</li> </ul>	
AA3.2	Deployment done and website works <ul style="list-style-type: none"> <li>• Deployment [3]</li> <li>• Fully functional website on the server [4] – failure to fully configure the website to be working properly remotely will result in loss of 4 marks</li> </ul>	
AA4.2	<ul style="list-style-type: none"> <li>• Upload works! [3]</li> <li>• Updates in db work as well! [3]</li> <li>• Filenames should be unique [1]</li> </ul>	
AA4.3	<ul style="list-style-type: none"> <li>• Model representing File (transfer) ok! [2]</li> <li>• Repository class ok! [2]</li> <li>• Controller-View with validation ok! [3]</li> </ul>	
SE1.3	<ul style="list-style-type: none"> <li>• Class Hierarchy acceptable [2.5]</li> <li>• Change has to be done from appsettings.json [2.5]</li> <li>• It works! [5]</li> </ul>	
SE3.3	<ul style="list-style-type: none"> <li>• IdentityDbContext inherited (Db populated with tables) [2.5]</li> <li>• Authentication works [2.5]</li> <li>• History of Transfers works &amp; access controlled [2, 3]</li> </ul>	