

Assignment 1 Out of 50 Marks

DUE: 29 March 2023

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.
- All assignments are submitted via ClickUP. See the Assignments section.
- You only upload your Angular App (.zip), API (.zip), and video demo (.mp4).
- You may use the L&P: 01 Architecture with TypeScript Source Code shared with you previously. See the code on ClickUP under Course Content. You then build upon it.
- Please execute the API migration before building your angular application. See the Angular and API installation and configuration section.
- If you are caught plagiarising, we will give you zero percent (0%), and you will be reported for plagiarism immediately. We will audit historical assignments throughout the semester. We trust that you understand the importance of this point.

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 Standard Requirements against the Rubric.
- When showing something from the Standard Requirements, show us as much detail as required. See the Rubric for the assessment criteria. For example, when assessing the "Program Functionality," you must show the CRUD functionality is working and that the data is saved and updated in the database. Similarly, for the "Program Output," the correct records are displayed on the related pages per CRUD. The web pages display the right data in a valid format, demonstrating all the pages. Further, for the "Code readability" we expect you to show us your code and display the organization of the code and descriptive names (i.e., all the code used to create the program, not the configuration files like package.json, etc.). The same applies to the rest of the Rubric. See below.
- 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. This is to assess you correctly according to the Rubric.
- See the "Video Recording and Compression and Assignment Upload Guide" in the Assignments section on ClickUP for video recording, compression, and upload assistance.

SUBMISSION INSTRUCTIONS:

- In this assignment, you will be given the requirements to implement.
- Source Code: Zip your source code files together, and for the API name it uXXXXXXX_HW01_API.zip, where the XXXXXXXX is your student number, e.g., u12345678_HW01_API.zip. Further, for the Angular App, name it uXXXXXXXX_HW01_Angular.zip, where the XXXXXXXX is your student number, e.g., u12345678_HW01_Angular.zip.
- Video Demo: <u>Do not</u> zip your video demo. In other words, submit the actual ".mp4" file. Name the video demo uXXXXXXX_HW01.mp4, where the XXXXXXXX is your student number, e.g., u12345678 HW01.mp4.
- If files are uploaded to the wrong upload area, we will not look for the upload. Uploads should be submitted correctly.
- <u>Please Note</u>: If you omit the code (.zip) or the video (.mp4) submission, you will lose <u>50%</u> of your assignment mark. If no files are uploaded (neither the .zip nor .mp4), you lose <u>100%</u>. Please take this seriously and plan accordingly to submit it on time.
- Note: you upload the code (.zip files) and the video demo (.mp4 file) together in the same location in the Assignment 01 Submission section. See the ClickUP information in the Assignments section (when readily available).

- Please <u>do not</u> upload the "node_modules" and ".angular" folders for the Angular App. In other words, once you have completed your program and created your video, delete the "node_modules" and ".angular" folders. As the Lecturing Team, we will reinstall the node_modules folder dependencies using the "npm install" terminal command, where necessary. This is so you do not take long to upload your code with the video demo.
- In addition, do not upload the "bin" and "obj" folders for the .Net API application. In other words, once you have completed your program and created your video, delete the "bin" and "obj" folders.

SUBMISSION DEADLINE: 29 March 2023

- There shall be no extensions to the deadline above.
- If homework submissions are uploaded too late, then upload errors will happen.
- Do not wait until the last minute to complete the assignment.
- Start working on the assignment as soon as possible.
- E-mail submissions will not be accepted.
- Late submissions will not be accepted.
- No exceptions will be made for anyone.

USE CASE:

- A client requests you to create a proof-of-concept application using Angular and .Net 6 Web API. They want to see if software development productivity improves using the Angular framework.
- You are requested to develop the back end using a .Net core 5 API and the front end using Angular.
- For the application, you need to build the capability for users to Create, Read, Update, and Delete (CRUD) courses stored in the SQL Server 2019 database.
- When the application is launched, the landing page must be the Course Listing page, and navigation to all other
 pages must be done via angular routing, subject to the restrictions that will be detailed under "Standard
 Requirements".

STANDARD REQUIREMENTS:

- Course Listing page:
 - The course listing page should pull through the ten records you created when you have done the API migration using the seed data provided in the **AppDbContext.cs** file (Fig. 1).
 - o In addition, the table that lists the ten records should have an Edit and Delete button per record (Fig. 1).
 - The table should display only the Name, Duration, and Description data from the Courses table.
 - Clicking on the Delete button should delete the relevant record and display the updated listing (Fig. 2).
 - Clicking the Edit button should take you to the "Edit Course" page to edit the existing database record (Fig. 3).
 - The Navigation Bar displayed in Fig.1. should be shared across all the created pages. The "Course Listing" link takes you to the Course Listing Page (current page), and the "Add Course" link routes you to the "Add Course" page (Fig. 6).

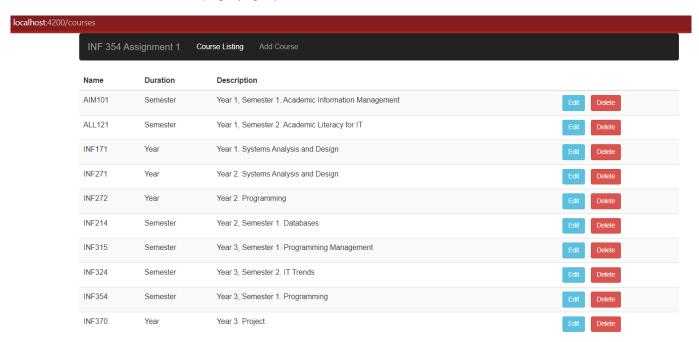


Fig. 1

INF 354 As	INF 354 Assignment 1 Course Listing Add Course							
Name	Duration	Description						
AIM101	Semester	Year 1, Semester 1. Academic Information Management	Edit Delete					
ALL121	Semester	Year 1, Semester 2. Academic Literacy for IT	Edit Delete					
INF171	Year	Year 1. Systems Analysis and Design	Edit Delete					
INF271	Year	Year 2. Systems Analysis and Design	Edit Delete					
INF272	Year	Year 2. Programming	Edit Delete					
INF214	Semester	Year 2, Semester 1. Databases	Edit Delete					
INF315	Semester	Year 3, Semester 1. Programming Management	Edit Delete					
INF324	Semester	Year 3, Semester 2. IT Trends	Edit Delete					
INF354	Semester	Year 3, Semester 1. Programming	Edit Delete					

Fig. 2

Edit Course page:

- The edit course page should allow the user to change the current course's Name, Duration, and Description. The user can update any one, two, or all of the control values (Fig. 4).
- After clicking the submit button, the record updates in the database, and the user is routed to the Course Listing page (Fig. 5)
- Clicking on the cancel button returns the user to the Course Listing page without making any changes to the data.

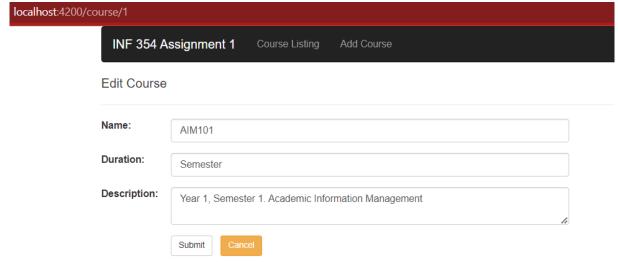


Fig. 3

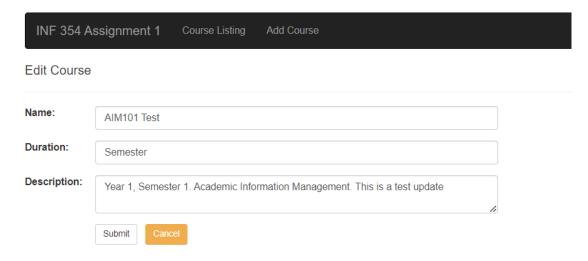
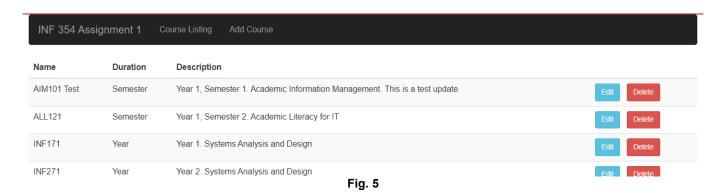


Fig. 4



Add Course page:

- o Once the user clicks on the "Add Course" link, they should be routed to the "Add Course" page (Fig. 6)
- The submit button is disabled and will only be enabled once values are provided for all three Form Controls (Fig. 7).
- After clicking the submit button, the record is created in the database, and the user is routed to the Course Listing page (Fig. 8)
- Clicking on the cancel button returns the user to the Course Listing page without making any changes to the data.

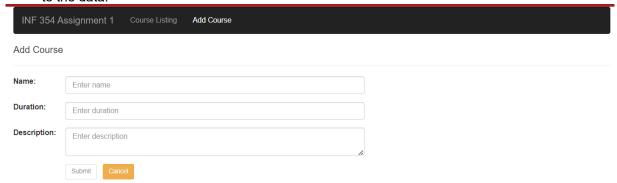


Fig. 6



Fig. 7

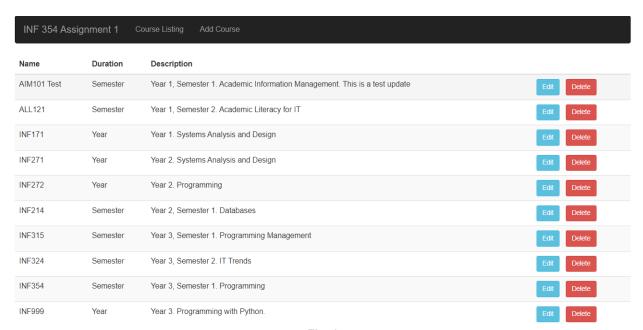


Fig. 8

ANGULAR AND API INSTALLATION AND CONFIGURATION:

API:

- An "API Template" has been created with the default configuration. In other words, the Cors, Database Connection, the Course entity, and .Net Core installations to get you started. This has been shared via the L&P: 02 Architecture with TypeScript source code.
- o Open the Assignment01 .Net Core application in Visual Studio 2022.
- Once the application loads, open the "appsettings.json" file in Solution Explorer.
- o Change and save the **Server** location, pointing to your *SQL Server Server Name*). Alternatively, you can just replace the server name with a period (.). See the example below.
- Example:
 - optionsBuilder.UseSqlServer("Server=.;Database=Assignment1;Trusted_Connection=True;Mult
 ipleActiveResultSets=True");
- Next, open the Package Manager Console (View > Other Windows > Package Manager Console) and run the following 2 commands individually to create the database tables from the abovementioned entities.
 - add-migration initial
 - update-database
- The Assignment1 MS SQL Server 2019 database will create the Courses table.
- Now, run the API and have it running when you are trying to connect your Angular App to it. In other words, the API and the Angular App must be running for the application to work correctly.

Angular:

An "**Angular Template**" has been created with the default configuration. In other words, a Course component with relevant files, routing, and service to get you started. This has been shared via the L&P: 02 Architecture with TypeScript source code.

- Open the Angular application in Visual Studio Code, and run "npm install" in the terminal window in the correct folder. I.e., the folder where you run "npm install" must have, for example, the "angular.json" file directly in it or you will run into problems.
- o To run the application, type "**ng serve**" in the terminal window.

SUGGESTIONS:

- o For the API, you will likely have 1 controller (the API controller with endpoints (functions) to talk to the database and Angular App).
 - For example, a CourseController with at least 4 endpoints (functions) to GetAllCourses (GET), AddCourse (POST), EditCourse (PUT), and DeleteCourse (POST).
- You can design your UI any way you want, so long it has all the controls and output required as specified in the Standard Requirements.
- You can develop your API any way you want, so long as it can perform the functionality required as specified in the Standard Requirements.

RUBRIC: Your assignment submission will be marked according to the following rubric:

Program (50 pts)	(Exceptional)	(Very good)	(Good)	(Satisfactory)	(Poor)	(Very poor)
Program	The program	The program	The program executes		The program executes with	
Execution	executes correctly	executes with one or	with a few syntax or		major errors. <i>E.g. The</i>	
	with no syntax or	two syntax or runtime	runtime errors. E.g. A		program can execute,	application fails to run. (0)
	runtime errors. <i>I.e.</i>	errors. <i>E.g.</i> the	couple of runtime		however, it is plagued with	
	the program has no	program loads with	errors and/or the		runtime or syntax errors, or	
	execution issues. (10)	no crashing but displays minor bugs	program crashes at one screen/section. (6)	crashes at two screens/sections. (5)	the program keeps crashing during use. (3)	
	(10)	in the debugger. (8)	One screen/section. (0)	Screens/Sections. (3)	crashing during use. (3)	
Program	Program	Program functionality	Program functionality	Program functionality	Program functionality has	Program functionality is
Functionality	functionality is in	has one minor	has a few minor		major inconsistencies. <i>E.g.</i>	
	line with the	inconsistency. <i>E.g.</i>	inconsistencies. <i>E.g.</i>		Most of the functional	
	requirements. I.e.	One of the functional	Two of the functional		requirements is incorrect or	
	the program has all	requirements is	requirements are	requirements are		the functionality is
	the correct	incorrect. (8)	incorrect or one is	incorrect or half is		missing. (0)
	functionality		missing. (6)	missing. (5)		
	implemented. (10)			- .		
Program Output	The program	The program has one	The program has a few		The program has major	
	displays correct output in line with	or two very minor output discrepancies.	output discrepancies. I.e. It produces output		output discrepancies. <i>I.e.</i> The output is plagued with	
	the requirements.	I.e. It produces output	with easily noticeable		inconsistencies. E.g. The	
	I.e. It produces the	with barely noticeable	inconsistencies. E.g.		program does not return	
	same output as	inconsistencies. E.g.	The program does not		most of the data or there	
	required. (10)	one or two formatting	return some of the data		are substantial formatting	
	, , ,	issues. (8)	or there are a few	or there are plenty of		, , ,
			formatting issues. (6)	formatting issues. (5)		
Program Interface	The program	The program	N/A	The program interface	The program interface is	
(UI)	interface is	interface is done well.		is good enough. I.e.	poorly done. <i>I.e. The</i>	
	professionally done.	I.e. The interface is		The interface is	interface is mostly incorrect	
	I.e. The interface is	implemented		implemented correctly	or looks poorly done. E.g.	
	implemented correctly and looks	correctly and looks good. E.g. One or two			The layout is mostly incorrect or has plenty of	
	very good. (5)	styling/layout issues.		few styling/layout issues. (3)		the styling is missing. (0)
	vory good. (0)	(4)		133463. (0)	Dryming 133003. (2)	the styling is impoing. (0)
Code Readability	The program code	Program code is	N/A	Program code is	Program code is somewhat	Program code is difficult
1	is well organized	organized and makes		mostly organized and	organized, and not easy to	
	and makes good	use of white space.		makes use of white	read and understand. <i>E.g.</i>	•
	use of white space.	Variables have		space. Most variables	There are plenty of variable	_
	Variables have	descriptive names.		have descriptive	naming convention issues	

Program (50 pts)	(Exceptional)	(Very good)	(Good)	(Satisfactory)	(Poor)	(Very poor)
	descriptive names.	E.g. There are one or			or the code is challenging	missing or the code is
	I.e. There is nothing	two variable naming		a few variable naming	to follow. (2)	hard to follow. (0)
	to fault on. (5)	convention issues or		convention issues or		
		white space issues.		program code		
		(4)		organization that could		
				be improved. (3)		
Video Demonstration	The program is exceptionally well presented. I.e. The student demonstrated and displayed all the required functionality, output, interfaces, and code. (10)	The program is well presented. E.g. The student demonstrated and displayed all the required functionality, output, interfaces, and code. However, one of the descriptions or illustrations was lacking. (8)	presentation is good. E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, two of	presentation is adequate. E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, a few to half of the functionality,	displayed a few of the required functionality, output, interfaces, and code. However, most functionality, output,	been presented or has been presented very poorly. E.g. The student failed to demonstrate and display the required functionality, output, interfaces, and code or it was missing. (0)