Project Management

Explanation of GitHub

Ben's diagrams – full papyrus project.

Ben's images – screenshots of all diagrams, as my Papyrus wouldn't allow me to properly link to the GitHub.

This document contains all written work – discussion on teamwork, quality requirements and discussion/ explanations on diagrams.

Team Structure

The team for this Project consists of four members, each of whom is focusing on one of USU's subsystems. And I believe this method is optimal considering the fact that the main system has already been split into four more simple subsystems in the brief. The development of each subsystem is as follows:

- Member A – USU Student App (mobile application for students).
- Member B – Student Union Management System (web interface for university-specific student unions).
- Author - USU Operation System (federation management subsystem).
- Member C – Society Leader App (mobile and web interface for society leaders).

Collaboration

Despite working on four different subsystems, we are still focusing on having a high level of collaboration by attending weekly meetings to share options on others subsystem designs. Along with the weekly meetings, we also have other ways to communicate, including a GitHub page to share files and information like our weekly meeting notes or our meeting agendas as well as our design documents. We also have a WhatsApp group to update each other on where we are on our design documents and to confirm if we will all be able to attend the weekly meetings.

We also decided to assign a team leader to manage the organisation of the meetings and take meeting notes. Tania Reyes was eager to take the role, so this was decided without conflict.

Meetings/schedule

Meeting	Date	Topic	Attendance	Time and Place
1	23/09	Organising group roles, assigning a leader, and setting up collaborative media.	All member	12:00–13:00, AB-115, Headington Campus
2	07/10	Review case study, discuss quality attributes, review Project management notes, and set initial deadlines.	Tania Reyes Malachai Broderick	12:00–13:00, AB-115, Headington Campus
3	14/10	Finished write-up on early group collaboration and shared progress on quality requirements, as well as planning use case diagrams.	Tania Reyes Ben Crosfield	12:00–13:00, AB-115, Headington Campus
4	21/10	Sharing finished quality requirements as well as starting on the use case diagram.	Tania Reyes Ben Crosfield	12:00–13:00, AB-115, Headington Campus
5	28/10	Uploaded and shared my finished use case diagram and my activity diagram. Issues with papyrus were slowing progress; however, we helped each other fix them.	Tania Reyes Ben Crosfield	12:00–13:00, AB-115, Headington Campus
6	4/11	Discussed plans for the component diagrams and began working on them.	Tania Reyes Ben Crosfield	12:00–13:00, AB-115, Headington Campus
7	11/11	Shared progress on component diagrams.	Tania Reyes Ben Crosfield	12:00–13:00, AB-115, Headington Campus
8	18/11	Completed component diagrams and began group implementation of the component diagrams	Tania Reyes Ben Crosfield	12:00–13:00, AB-115, Headington Campus
9	25/11	Shared our combined component diagram	Tania Reyes Ben Crosfield Lucas Olivera	12:00–13:00, AB-115, Headington Campus
10	25/11	Continued work on component diagrams ensured all content of part 4 was completed to a good standard.	Tania Reyes Ben Crosfield Lucas Olivera	12:00–13:00, Online

11	02/12	Shared our UML diagrams and behaviour diagrams before working to prepare our final submission	Tania Reyes Ben Crosfield Lucas Olivera	12:00–13:00, AB-115, Headington Campus

Project Management Approach

We have decided the best approach to this project is an agile approach. We achieve this by having a focus on collaboration at our weekly meeting. This helps us keep our work flexible while giving us clear roles on what work we need to carry out.

Quality requirements of USU’s Operating System

Overview

USU is a UK-based organisation that is looking to upgrade its current system to a cloud computing-based system. Their goal is to provide software and assist all student unions and societies in organising events, as well as software to assist student unions and societies in managing their members. To do this, they plan to create four pieces of software that link to a backend system called the United Students that shares the data from these 4 pieces of software to carry out live updates. The system I will be looking at the quality requirements for is a web app for the USU officers to manage the student unions connected to the USU and the USU national events.

Functionality of the system

This web app will have a few different functions that the users should be able to carry out:

- Receiving a USU application and reviewing the application data before rejecting or accepting the union's USU application.
- Upon a successful application, union leaders should receive a login to the student union management app, and students in that Union can access the USU app, and their data will be stored. As well as this, union members can promote events via USU and attend USU events.
- Receiving requests to edit student union data and then accepting or rejecting them, saving the changed data over the old data.
- Receiving a request to terminate membership and then deleting that union's details and blocking their access to USU tools.

- USU officers should be able to create events containing all event details. This information will go on USU's portal.
- A USU officer should be able to make announcements about existing events. This will show on the portal and email those attending.
- USU officers should be able to see those who have registered for one of their events.

Quality priorities

In the brief, USU has outlined 5 key features of their system, these are:

- KF-1: Dynamic Update of Information.
- KF-2: Services Across University-Specific Student Unions.
- KF-3: Automated Workflow.
- KF-4: User Experience.
- KF-5: Security and Privacy Protection.

From these attributes, we can understand what Quality Attributes they want to focus on.

KF-1: Dynamic Update of Information.

Here, USU talk about dynamically updating data and pushing the updated data to users in real time. This is most important for the mobile app; however, it still means that data entered on the USU web app should be pushed into the cloud database quickly, so they want a focus on efficiency.

KF-2: Services Across University-Specific Student Unions.

Here, the USU talk about how they must collect data from many different universities and push that relevant data to students at that university. This means that the system must be able to handle a large amount of data. This shows the USU also want to focus on scalability, which can have an inverse effect on efficiency. This is very Important for the operating system as it needs to collect and store all the data from Student Unions across the country.

KF-3: Automated Workflow.

Here, USU specify how important it is for all the systems to be linked together and how the connection between the four systems needs to be seamless. I'd say this is once again demanding high levels of efficiency in all 4 pieces of software, so I will remember efficiency is a high priority when designing the application.

KF-4: User Experience.

Here USU demand an excellent user experience taking advantage of all technologies available to them they want features like graphical user friendly interfaces and would even like to provide an AI tool to assist with queries on the app these user experience features will definitely have inverse effect on the efficiency but shouldn't massively effect scalability useless they are unprepared for large amounts of traffic on features such as AI tools. These user experience demands show a high priority on usability; however, I will not need to worry about issues with the AI tool as the USU web application I'm designing isn't designed to be accessed by all users only the USU admins so while the interface still needs to be easy to use the amount of graphics and effects in the software can be minimal.

KF-5: Security and Privacy Protection.

This is the final key feature USU list, and I believe that, as the people on my web application have access to a lot of the data submitted by societies, there will need to be a lot of features in place to prevent unauthorised access, the simplest of which would be something like password protection. Once again, security will have an inverse effect on efficiency. Despite this, I am going to put it as the most important feature for the software I am designing.

Quality objectives

QUALITY ATTRIBUTES	Desired Outcome
Functionality	For functionally the main focus will the Interoperability as systems must link to the other three systems created along with this, the security on the website should be very high as the system provides tools to access and edit a lot of data stored in the USU's database.
Reliability	Reliability is required in any system and is very important for the web app, as if there are any changes in USU events or USU's connected societies, they may need to be able to change their database quickly or use the website to quickly update the information on an event. So, we cannot have an unreliable system, and even in the event of one bug, we must have high fault tolerance, so it doesn't affect other users for the web app.
Usability	The number of people with access to the web app is limited, as only USU officers can use it. However, it will still be important to have a simple, easy-to-use interface. To help with usability, learnability and operability.
Efficiency	We should be able to create an efficient website, as it is purely for data management; we don't need any resource-intensive

	images or effects, so the overall efficiency of the web app should be high. This will help ensure updates can go out quickly.
Maintainability	The system shouldn't be overly complicated, as the main system has been split into four sub-systems, and this web app sub-system should be easy to maintain. The main goal will be for it to be able to deal with the scale of information. Future changes to the software may involve entering more information to create a society, and small adaptations like this won't be difficult to add to the system.
Portability	Being a web app, the software should be highly portable; it should work on all operating systems and devices, so this will be something that needs to be considered during design.

ISO 9126 Model

Quality prioritization

While all quality attributes are very important, all should be considered in the design stages. I have listed them here in levels of priority for the system requirements USU have shared with us.

Functionality

In my quality report I have decided that the most important quality requirements is the functionality this is down two main reasons the first being the requirement for the system to be connected to 3 other systems so a high interoperability is required also with the level of access to the USU database this piece of software offers I have decided that security should also be a high priority which in the model I looks at comes under functionality.

Reliability

I have system reliability as the second most important quality. This is due to the fact that USU want real-time updates from this software to their other pieces of software, and without a reliable system, it may be hard to achieve this.

Portability

While portability wasn't specified in the brief when designing a web application, it is important that it can be loaded on a selection of devices, as we don't know what devices the USU officers will have access to, so I think the website should be designed to work on all Operating Systems and devices, from laptops to mobile devices.

Efficiency

While this software isn't going to be resource-intensive, I think that to carry out the real-time updates across their three other pieces of software, I will have to ensure the website is efficient.

Usability

As the website has many different tools that can affect the database drastically, I must ensure the website is easy to use to prevent management errors that could damage USU's database.

Maintainability

Despite having it last, maintainability is still very important. I need to ensure the software I design will be easy to change in the future and easy to fix so USU can continue to add new features to their system.

Quality relations

Focusing on one quality Attribute can have inverse effects on others. With the project I must focus on this, I will have to remember that high levels of security can have large impacts on many other quality attributes. The main one it affects is the efficiency, as encrypting data will slow down the speed of the website. And it can have effects on the functionality, as there will need to be security features in place, like password protection and verification on certain actions, like deleting data, which does add an extra step to doing everything, but I believe it is necessary.

Another quality relationship I will need to keep in mind when designing the software is the fact that improving all quality attributes will influence the efficiency, so I will need to find a balance between focusing on the key quality attributes without having too much efficiency.

I will also need to find out about how reliability, which is one of the quality attributes I want to focus on when designing the software, can have a direct impact on maintainability and usability.

Zhu, H. (2025)

Success indicators

QUALITY ATTRIBUTES	Success indicator
Functionality	Data will be shared between the four different systems, and my system will be able to access and edit data stored in USU's database. The data will also be secure with encryption and password protection, so no unauthorised users can gain access to the data or web app.

Reliability	**Failure rate:** The USU web app system should be operated continuously with a failure rate lower than 3 times in a year of operation. **Mean time to recovery**: When a failure occurs, the USU web app system should, on average, recover to normal operation no more than 12 hours of downtime, as a longer downtime may result in USU officers being unable to carry out updates on live events. **Availability:** The USU web app system should not be out of service more than 50 hours per year in any consecutive operation period of one calendar year.
Usability	The system should have a simple easy to easy-to-understand graphical interface that will be easy for USU officers to understand.
Efficiency	When a user clicks a button, the system should take no longer than 2 seconds to respond. When a user deletes, edits or adds data, the system should take no longer than 4 seconds to respond.
Maintainability	The software should require minimal maintenance and be easy for someone with low-level IT skills to maintain, and carrying out software updates should be easy.
Portability	The software should be highly portable, and it should work on all operating systems and devices. The system should be ready to deal with a large amount of data and be ready to deal with traffic from having up to 500 users, which will consist of all the USU officers across the country.

Zhu, H. (2025)

Preliminary quality strategies

Some of the strategies I think we should consider in the design stage are:

Secure by Design- this involves starting the design process by thinking of security as a foundational property of the software instead of an afterthought. (Niri, M. (2025))

Encryption/Password Protection – as mentioned earlier in the report, I am very keen to have encryption and password protection to add more levels of security to the software.

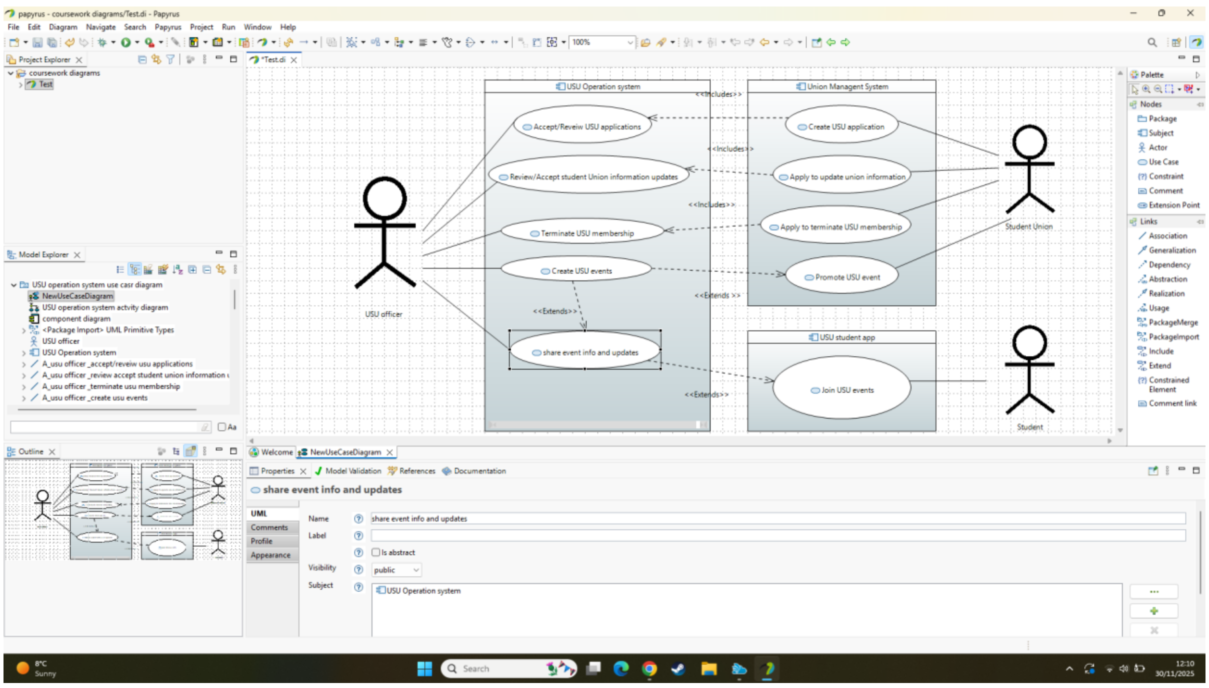
User Testing – to assist with usability, I want to have USU officers carry out testing through the design process so they can give feedback on the graphical interface.

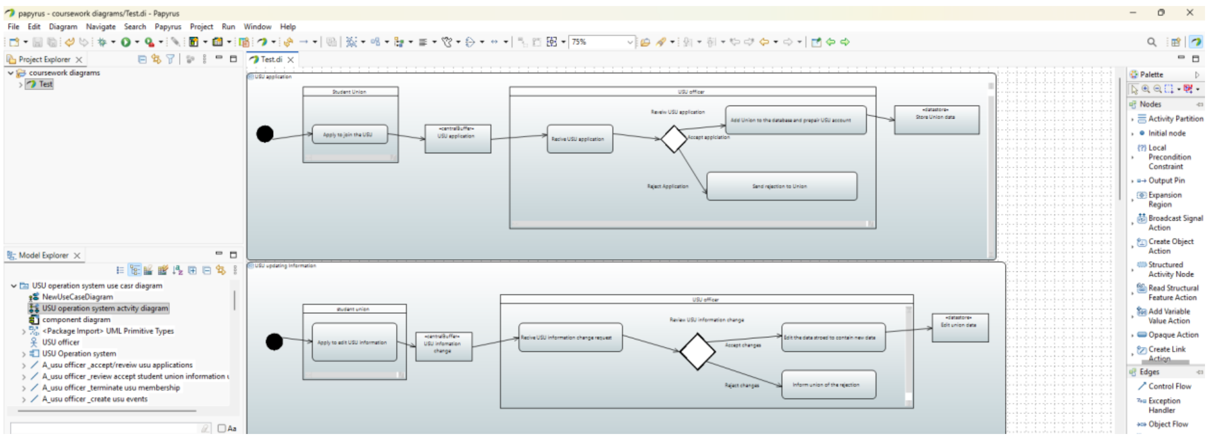
Backups – as this software is linked most closely to USU's database, I want to ensure the database carries out frequent backups to assist with reliability.

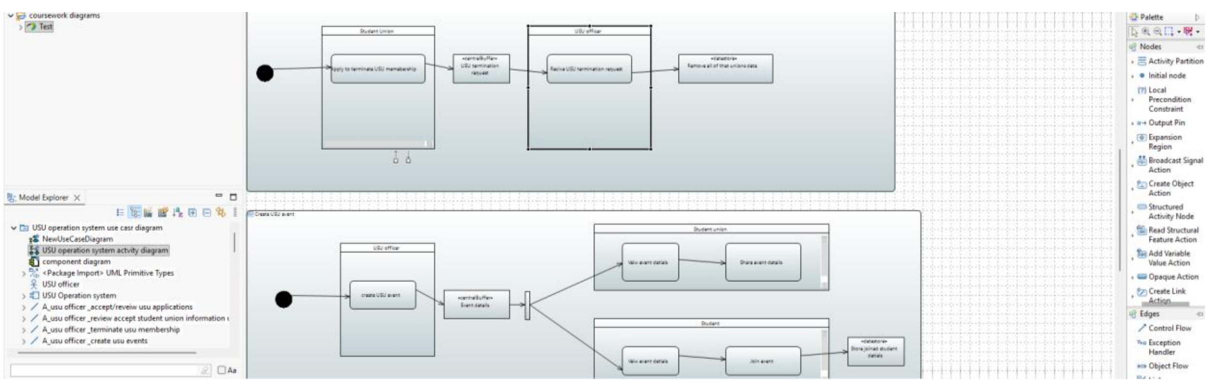
Design

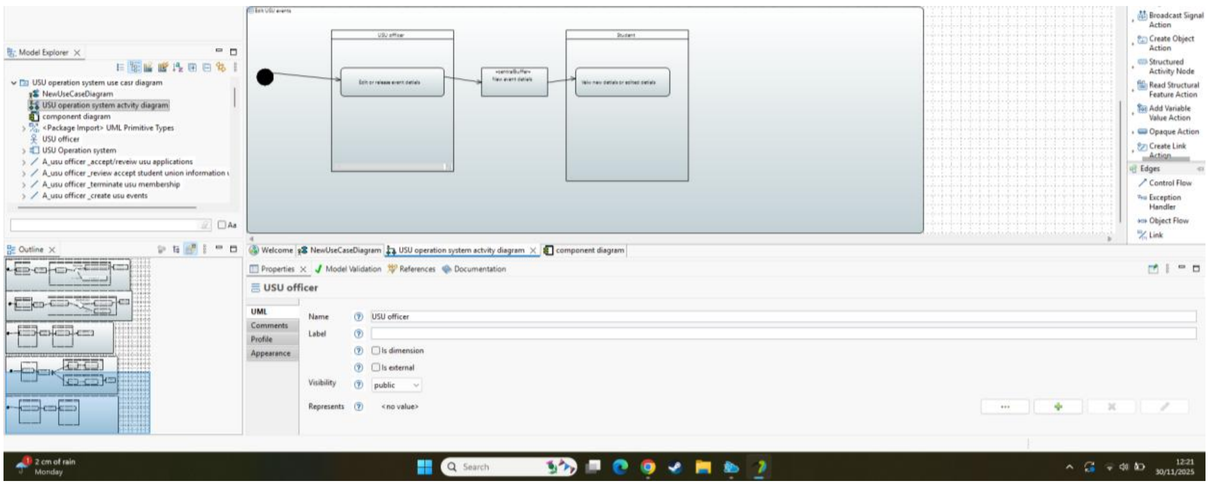
Use case and Activity diagram.

The first diagrams I constructed with my plan were my use case and my activity diagram, both of which aim to show the key features of the program, the use case diagram highlighting how all parties involved with USU will be interacting with my specific subsystem. The activity focuses on the main requirements and shows the functionality of carrying out all of these requirements.







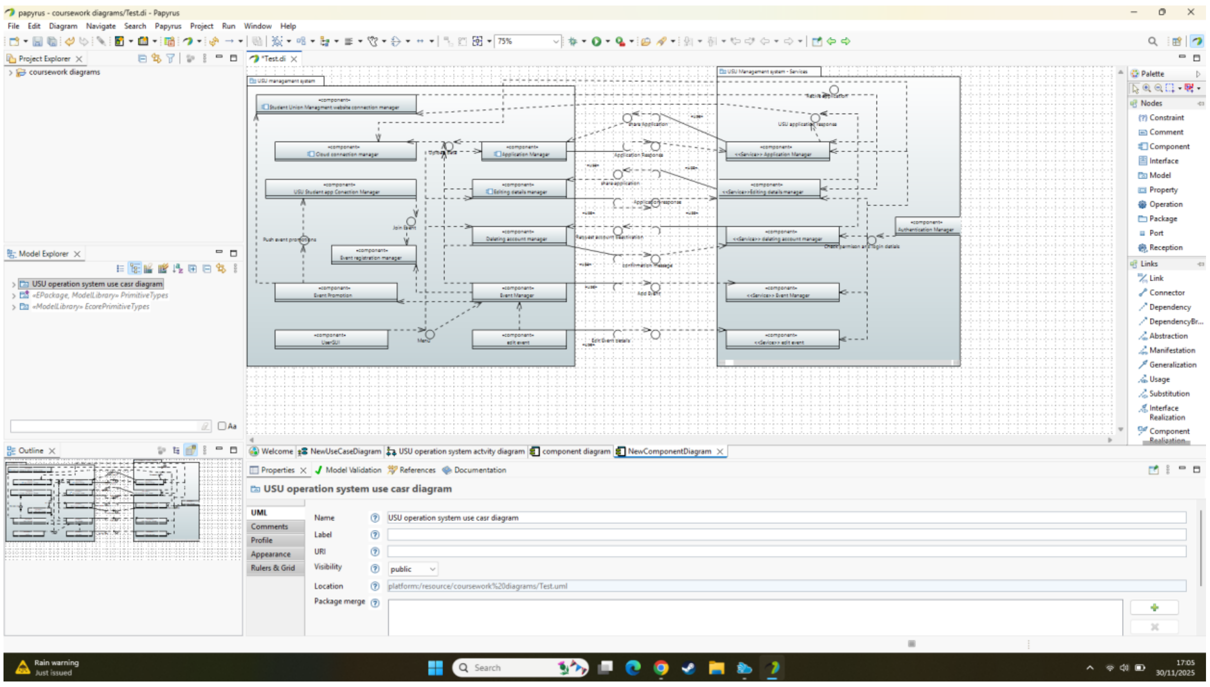

USU officer
Student
Edit or release event details
«centralBuffer» New event details
Add new details or edited details
Model Explorer
USU operation system use casr diagram
NewUseCaseDiagram
USU operation system actvity diagram
component diagram
USU officer
USU Operation system
Outline
Welcome
NewUseCaseDiagram
USU operation system actvity diagram
component diagram

USU officer

Name	USU officer
Label	
Is dimension	☐
Is external	☐
Visibility	public
Represents	<no value>

Architectural design

For my architectural design, I made a component diagram that consists of 4 main services with a couple of other features to assist with security and usage of the website.


papyrus - coursework diagrams/Test.di - Papyrus
Project Explorer
coursework diagrams
*Test.di
USU management system
USU Management system - Services
«component» Student Union Managment website connection manager
«component» Cloud connection manager
«component» Application Manager
«component» USU Student app Conection Manager
«component» Editing details manager
«component» Deleting account manager
«component» Event registration manager
«component» Event Promotion
«component» Event Manager
«component» UserGUI
«component» edit event
«component» «Service» Application Manager
«component» «Service» Editing details manager
«component» «Service» deleting account manager
«component» «Service» Event Manager
«component» «Service» edit event
«component» Authentication Manager
Share application
Application Response
Retrive application
Request account deletion
confirmation message
Add event
Edit Event details
Menu
Palette
Welcome
NewUseCaseDiagram
USU operation system actvity diagram
component diagram
NewComponentDiagram

USU operation system use casr diagram

Name	USU operation system use casr diagram
Label	
URI	
Visibility	public
Location	platform:/resource/coursework%20diagrams/Test.uml
Package merge	

Component	USU application manager
Description	This microservice allows the USU administrators to receive applications to USU and review them.
Stereo type	service
Required interface	Check permissions, Share application, menu.
Provided interface	Application response, upload data.

Component	edit account manager
Description	This microservice allows the USU administrators to receive requests to edit data and review it.
Stereo type	service
Required interface	Check permissions, share request, menu
Provided interface	Application response, upload data.

Component	Deleting account manager
Description	This microservice allows the USU administrators to receive requests to terminate accounts and process the requests.
Stereo type	service
Required interface	Check permissions, termination request, menu
Provided interface	Confirmation message, upload data.

Component	Event manger
Description	This microservice allows the USU administrators to create events, and it works as an interface for further microservices, such as editing or promoting events.
Stereo type	service
Required interface	Check permissions, menu
Provided interface	upload data, add event

Component	edit event (extends event manager)
Description	This microservice allows the USU administrators to update their event information.
Stereo type	service
Required interface	Check permissions, menu
Provided interface	Edit event

Component	User GUI
Description	This microservice prepares the GUI for the user so they can access all of the website's capabilities.
Stereo type	UI
Required interface	N/A
Provided interface	menu

Component	Authentication manger

Description	This microservice ensures the account accessing the software has valid login details and the correct privileges.
Stereo type	service
Required interface	N/A
Provided interface	Check privileges

Component	Event promotion (extend event manager)
Description	This microservice allows the USU administrators to create promotions for the event and share them across USU systems to things such as the USU phone app.
Stereo type	service
Required interface	N/A
Provided interface	Push event promotion

Component	Event registration manager (extend event manager)
Description	This microservice allows the USU administrators to view who has signed up for the events they are running, and it stores the data of who has signed up.
Stereo type	service
Required interface	Join event
Provided interface	N/A

Component	Student app connection manager
Description	This microservice ensures there is a connection to the USU app so data can be passed back and forth between the two systems.
Stereo type	control
Required interface	Push event promotion
Provided interface	Join event

Component	Cloud connection manager
Description	This microservice ensures there is a connection to the USU cloud storage app, so data can be stored in it.
Stereo type	control
Required interface	Upload data, check permissions.
Provided interface	N/A

Component	Student union management website connection manager
Description	This microservice ensures there is a connection to the Union management website so that data can be passed from this website to my website.

Stereo type	control
Required interface	USU response
Provided interface	Receive application

Name	Check permissions and login details.	
Provider	Authentic manger	
Operation	signature	USU login(userID(int),password(string), Ip_adress:ip_Adressscode(String)).
	Function	Check if the user has a USU account and is a USU admin. If it is unsuccessful, do not grant them any access to the program. If it is successful, an access token will allow the user to access all functions of the program.

Name	USU response	
Providers	Application manager, editing details manager, deleting account manager	
Operation	signature	USU application response (String)
	Function	passes a message back to the applying student union to inform them of their application status, whether they failed or passed the application review.
Operation	Signature	USU edit details response (String)
	Function	passes a message back to the student union, trying to edit their details to inform them of their application status, whether they failed or passed the application review.
Operation	Signature	USU account termination response (String)
	Function	Informs the union that their USU account has been terminated

Name	Share application	
Providers	Application manager, editing details manager, deleting account manager	
Operation	signature	USU application (String)
	Function	Send the application form to the application manager microservice so it can be reviewed.
Operation	Signature	USU edit details form (String)
	Function	Send the application form to the application manager microservice so it can be reviewed.
Operation	Signature	USU account termination form (String)
	Function	Informs the USU that a student union wishes to terminate their membership

Name	Upload data

Providers	Application manager, editing details manager, deleting account manager, event manager	
Operation	signature	USU application (Union name (String), Union ID (int), University of Union (string), union rep (name), Union rep id (int), USU officer name (string), USU officer id(int).
	Function	Uploads the new union data into the USU cloud system.
Operation	Signature	Unions edit information application (Union name (String), UnionID (int), University of Union (string), union rep (name), Union rep id (int), USU officer name (string), USU officer id(int).
	Function	Takes the new information the union has submitted and replaces the old information.
Operation	Signature	USU account termination token (USU id(int))
	Function	Takes the USU account ID and uses it to remove all information about the union from the system.
Operation	Signature	USU events (event name(int), event date and time (datetime), event information(string)
	Function	Takes the information of a USU event and stores it so it can be viewed by others, and people can sign up.

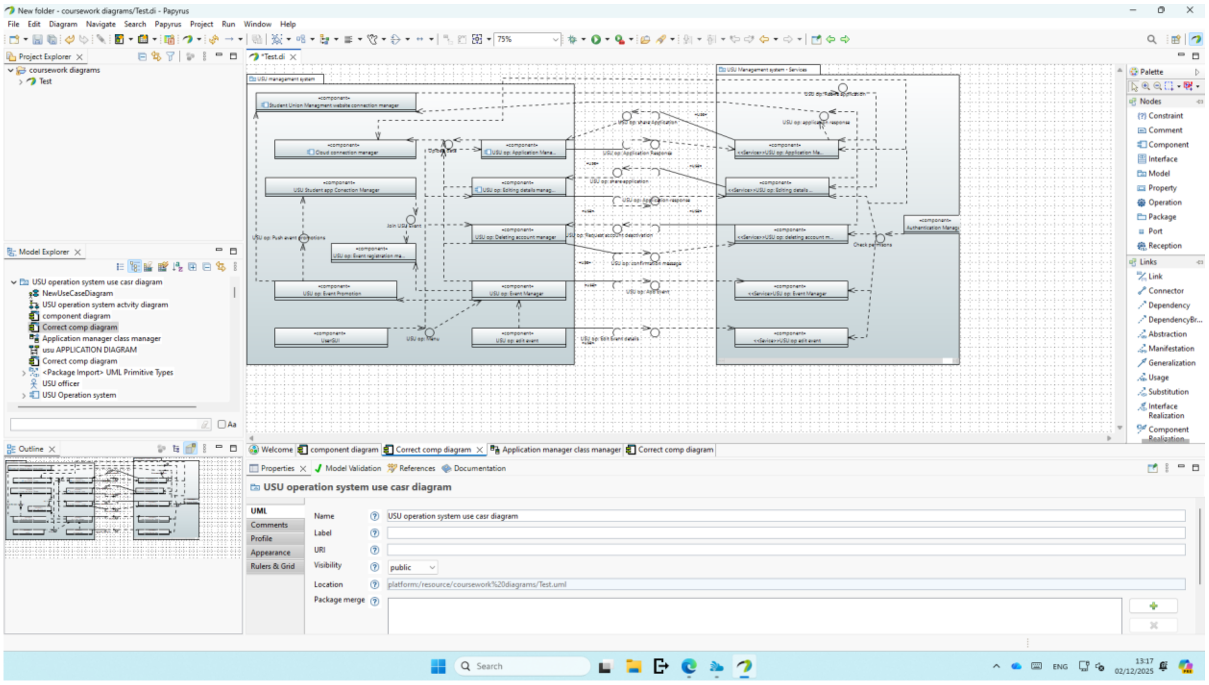
Name	menu	
Provider	UserGUI	
Operation	signature	GUI(HTML)
	Function	Loads the GUI that allows the users to access all the features of the program.

Name	Join event	
Provider	USU student app connection manager	
Operation	signature	Student registration (Student ID (int))
	Function	This takes the information of all students who have registered for an event and stores it so it can be accessed by the event manager.

Name	Push event promotion	
Provider	Event promotion	
Operation	signature	Event promotion(string)
	Function	Send a promotion for events out to student apps.

Combined architectural design

For the combined architecture of the system, my group and I decided the best approach would be to ensure all our components or interfaces have unique names, or if they are the same component or interface, they share a name, so I edited my original diagram to meet these criteria. Firstly, for components and interfaces specific to my subsystem, I added a USU op: tag in front of them, as my subsystem is the USU operating system for the other components that may be utilised across multiple subsystems. I left the names as generic names and informed my group on what generic names to use. Below is my adapted component diagram

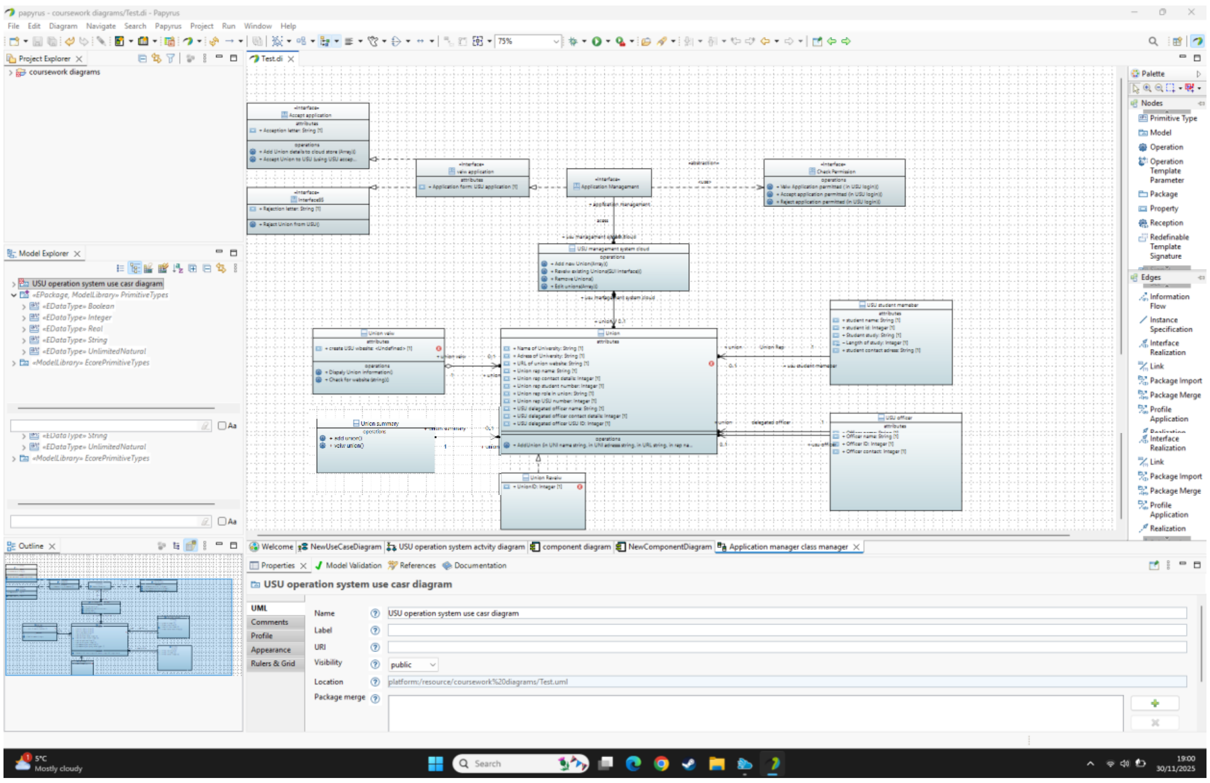

New folder - coursework diagrams/Test.di - Papyrus
File Edit Diagram Navigate Search Papyrus Project Run Window Help
75%
Project Explorer
coursework diagrams
Test
*Test.di
USU management system
«component» Student Union Managment website connection manager
«component» Cloud connection manager
«component» USU Student app Conection Manager
«component» USU op: Application Mana...
«component» USU op: Editing details manag...
«component» USU op: Deleting account manager
«component» USU op: Event registration ma...
«component» USU op: Event Promotion
«component» USU op: Event Manager
«component» UserGUI
«component» USU op: edit event
Upload data
Join USU Event
USU op: Push event promotions
USU op: Menu
USU op: share Application
USU op: Application Response
USU op: share application
USU op: Application response
USU op: Request account deactivation
USU op: confirmation message
USU op: Add Event
USU op: Edit Event details
«use»
USU Management system - Services
USU op: Receive application
USU op: application response
«component» <<Service>>USU op: Application Ma...
«component» <<Service>>USU op: Editing details ...
«component» <<Service>>USU op: deleting account m...
«component» <<Service>>USU op: Event Manager
«component» <<Service>>USU op: edit event
«component» Authentication Manag...
Check permissions
Palette
Nodes
Constraint
Comment
Component
Interface
Model
Property
Operation
Package
Port
Reception
Links
Link
Connector
Dependency
DependencyBr...
Abstraction
Manifestation
Generalization
Usage
Substitution
Interface Realization
Component Realization
Model Explorer
USU operation system use casr diagram
NewUseCaseDiagram
USU operation system actvity diagram
component diagram
Correct comp diagram
Application manager class manager
usu APPLICATION DIAGRAM
Correct comp diagram
<Package Import> UML Primitive Types
USU officer
USU Operation system
Aa
Outline
Welcome
component diagram
Correct comp diagram
Application manager class manager
Correct comp diagram
Properties
Model Validation
References
Documentation
UML
Comments
Profile
Appearance
Rulers & Grid
Search
ENG
13:17
02/12/2025

USU operation system use casr diagram

Name	USU operation system use casr diagram
Label	
URI	
Visibility	public
Location	platform:/resource/coursework%20diagrams/Test.uml
Package merge	

Software detailed design

The first thing I made in my detailed software design was a structural diagram of one of my microservices. The microservice I selected to look at in more detail is the application manager. I decided that this was the best microservice to look at, as it was one of the 4 main features requested of my system, and it was very similar to the edit details microservice. As well as this, it also worked with a lot of data, so taking a closer look at it was important.


papyrus - coursework diagrams/Test.di - Papyrus
File Edit Diagram Navigate Search Papyrus Project Run Window Help
Project Explorer
coursework diagrams
Test.di
«Interface» Accept application
«Interface» new application
«Interface» Application Management
«Interface» Check Permission
USU management system cloud
Union data
Union
USU student member
Union summary
USU officer
Union Review
Model Explorer
USU operation system use casr diagram
Outline
Palette
Nodes
Edges

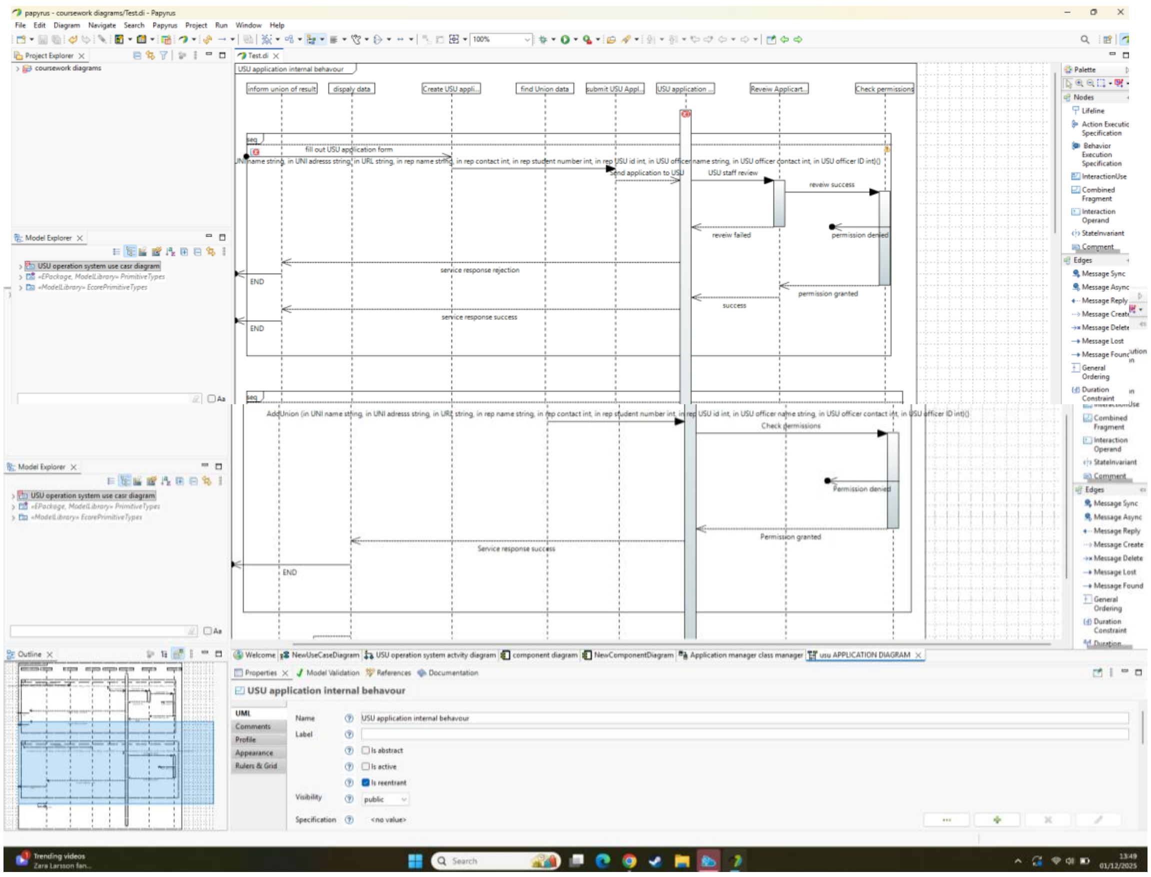
Welcome | NewUseCaseDiagram | USU operation system actvity diagram | component diagram | NewComponentDiagram | Application manager class manager

Properties | Model Validation | References | Documentation

USU operation system use casr diagram

UML | Comments | Profile | Appearance | Rulers & Grid

Name	USU operation system use casr diagram
Label	
URI	
Visibility	public
Location	platform:/resource/coursework%20diagrams/Test.uml
Package merge	


papyrus - coursework diagrams/Test.di - Papyrus
USU application internal behaviour
inform union of result
dispaly data
Create USU appli...
find Union data
submit USU Appl...
USU application ...
Reveiw Applicart...
Check permissions
seq
fill out USU application form
UNI name string, in UNI adresss string, in URL string, in rep name string, in rep contact int, in rep student number int, in rep USU id int, in USU officer name string, in USU officer contact int, in USU officer ID int)()
send application to USU
USU staff review
reveiw success
reveiw failed
permission denied
service response rejection
END
permission granted
success
service response success
END
seq
AddUnion (in UNI name string, in UNI adresss string, in URL string, in rep name string, in rep contact int, in rep student number int, in rep USU id int, in USU officer name string, in USU officer contact int, in USU officer ID int)()
Check permissions
Permission denied
Permission granted
Service response success
END
Palette
Nodes
Edges

Welcome | NewUseCaseDiagram | USU operation system actvity diagram | component diagram | NewComponentDiagram | Application manager class manager | usu APPLICATION DIAGRAM

Properties | Model Validation | References | Documentation

USU application internal behaviour

UML | Comments | Profile | Appearance | Rulers & Grid

Name	USU application internal behaviour
Label	
	☐ Is abstract
	☐ Is active
	☑ Is reentrant
Visibility	public
Specification	<no value>

Team performance report

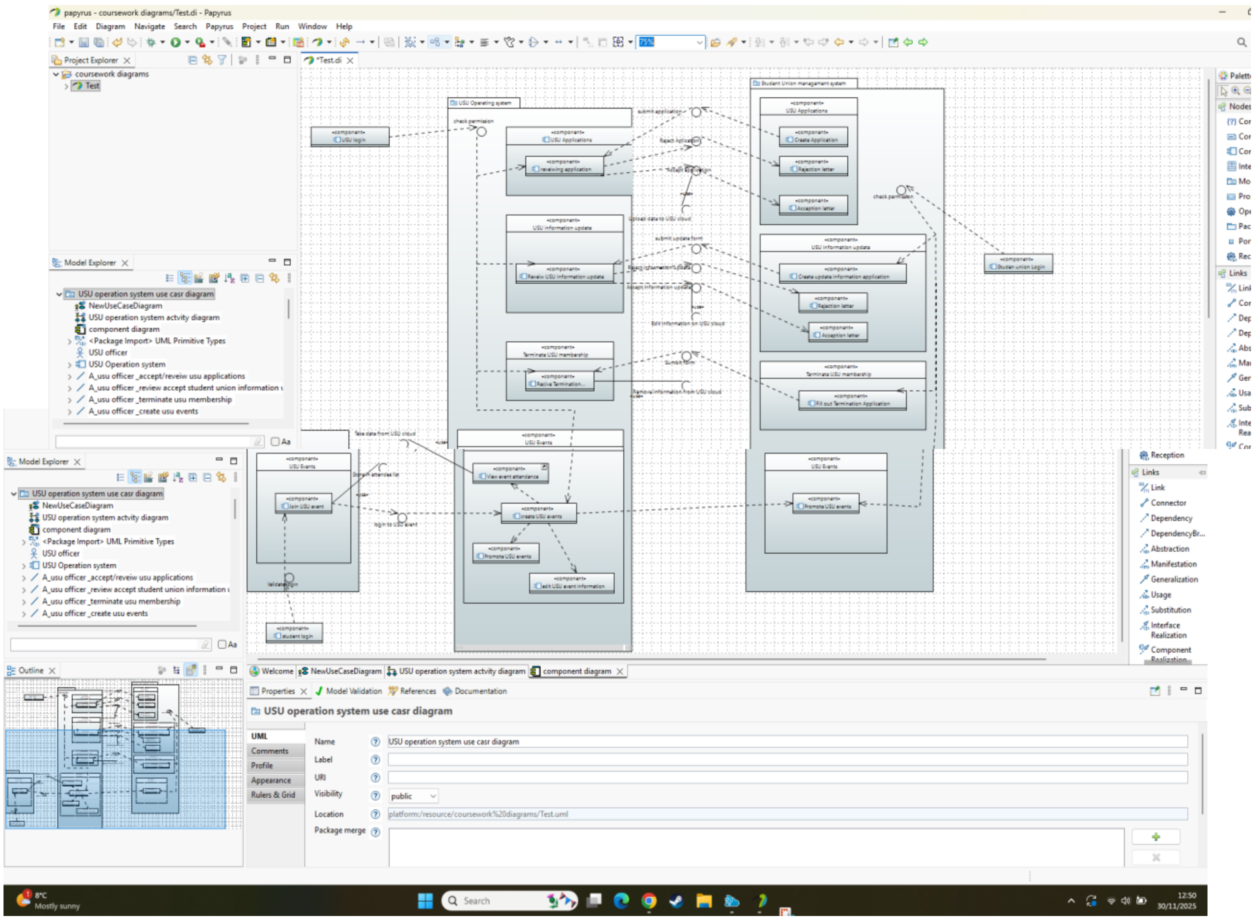
Overall as a group I feel we worked very well me and tania where constantly in attendance at all the weekly meeting which allowed for the agile programming approach to work really well for the two of us as we were able to share ideas and correct each other's incorrect work meanwhile Lucas was always active online responding to feedback we shared and sharing his own feedback on our work. However, Malachai stopped responding to the group messages after week 2, so communicating with him to get his part 4b was difficult, as, despite attempts from the group, we were unable to get his input for a lot of the work.

Our assigned group leader, Tania, did a good job of setting realistic deadlines, which both Lucas and I were able to meet, allowing for meeting to consist of progress summaries and planning for future progress and meetings.

So overall, I feel as a group we worked well, and I played my role well, helping add to meeting notes as Tania and I were the only members in attendance for a majority of the meetings, and I was able to successfully meet most of the group deadlines set.

Individual performance report

Overall, I was pleased with my individual performance, as I mentioned, I was able to keep up with deadlines set by the group, but I was also really proud of the quality of my work. The area I feel I excelled the most in was finding and analysing the requirements of the system. I wanted to highlight this area in particular as I feel like my report for this area was thorough, covering all the functions they required with good detail and my thought process in prioritising the non-functional requirements aligned well with the brief. Meanwhile, I found areas involving papyrus slightly more difficult, liked my activity diagram as I feel I was able to show how all related parties can have an impact on my subsystem via the actions they take on their subsystem, and I feel my activity diagram showed all functionality of my program. However, creating the component diagram proved more difficult. I believe the original structure diagram I created was completely incorrect; however, I kept it as it shows how my subsystem interacts with other subsystems well, and I have put my incorrect component diagram below.


papyrus - coursework diagrams/Test.di - Papyrus
File Edit Diagram Navigate Search Papyrus Project Run Window Help
Project Explorer
coursework diagrams
Test
Model Explorer
USU operation system use casr diagram
NewUseCaseDiagram
USU operation system actvity diagram
component diagram
<Package Import> UML Primitive Types
USU officer
USU Operation system
A_usu officer _accept/reveiw usu applications
A_usu officer _review accept student union information
A_usu officer _terminate usu membership
A_usu officer _create usu events
Outline
*Test.di
Welcome
NewUseCaseDiagram
USU operation system actvity diagram
component diagram
Properties
Model Validation
References
Documentation
UML
Comments
Profile
Appearance
Rulers & Grid
Mostly sunny
Search
12:50
30/11/2025

USU operation system use casr diagram

Name: USU operation system use casr diagram

Label:

URI:

Visibility: public

Location: platform:/resource/coursework%20diagrams/Test.uml

Package merge:

As is clear in the image, when making this diagram, I was unsure of how to properly make a competent diagram, which is another issue I faced for the structure diagram and behaviour diagrams; however, I feel the diagrams I made after making this mistake with the component diagrams are correct and a lot more insightful as to how my system should properly work. Along with trying to learn how the diagrams should look and what components they needed to contain, the task of making diagrams was made even more difficult by my lack of knowledge of papyrus, so at the same time, I had to learn how to use the software. This made some of the tasks very difficult. I found making the behaviour diagrams particularly time-consuming and fiddly; however, I eventually managed to make them, and I am proud of my final products.

Reference

ISO/IEC 9126-1:2001, Software engineering — Product quality — Part 1: Quality model, International Organisation for Standardisation, 2001.

Niri, M. (2025). Secure Code Design[Lecture], COMP5020 Foundations of Security.Oxford Brookes, 1st October.

Zhu, H. (2025). Software Quality [Lecture], COMP5047 Applied Software Engineering.Oxford Brookes, 30th September.