Assessment cover

OXFORD BROOKES UNIVERSITY

Module No:	COMP5047	Module title:	Applied Software Engineering

Assessment title:	Software Engineering of a Modern Computer Application

Due date and time:	23:00pm, 5th Dec. 2025

Estimated total time to be spent on assignment:	84 hours per student

LEARNING OUTCOMES

On successful completion of this assignment, students will be able to achieve the module's following learning outcomes (LOs):
1. Demonstrate an understanding of the role of requirements analysis and specification in software engineering and to be able to use this knowledge to create use case models and functional models of computer applications.
2. Demonstrate an understanding of the relationship between requirements and design and to be able to apply the knowledge to create structural and behavioural models of computer applications.
3. Critically evaluate and utilise design paradigms of object-oriented analysis and design, component-based design, and service-oriented design.
4. Use software modelling language such as UML and modelling tools in the context of model-driven software engineering.
5. Work in a group to apply the knowledge and skills developed in this module

Engineering Council AHEP4 LOs assessed	
C3	Select and apply appropriate computational and analytical techniques to model complex problems, recognising the limitations of the techniques employed
C5	Design solutions for complex problems that meet a combination of societal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societal, environmental and commercial matters, codes of practice and industry standards
C6	Apply an integrated or systems approach to the solution of complex problems
C14	Discuss the role of quality management systems and continuous improvement in the context of complex problems
C16	Function effectively as an individual, and as a member or leader of a team

GROUP ID:	**08**

STUDENT NAMES

	Student Id:	**Student Name:**	**Subsystem:**
1.	**19330111**	**Babusanu Gabriel**	**Student Union Management System**
2.			
3.			
4.			

Statement of Compliance *(please tick to sign)*

☑ We declare that the work submitted is my own and that the work I submit is fully in accordance with the University regulations regarding assessments *(www.brookes.ac.uk/uniregulations/current)*

RUBRIC OR EQUIVALENT:

Marking grid and marking form are available on Moodle website of the module.

FORMATIVE FEEDBACK OPPORTUNITIES

(a) Discuss your work with your practical class tutor during practical classes; (b) Discuss your work with lecturer and/or practical class tutor in drop-in hours.

SUMMATIVE FEEDBACK DELIVERABLES

Deliverable content and standard description and criteria
A file will be in Moodle for each student to give detailed mark decomposition and additional comments as feedback on your coursework, which include:
(a) Breakdown of marks on each assessment criterion
(b) Comments on each aspect of the assessment against assessment criteria
(c) Annotations on your submitted work, if any

USU Operation System

Task 1: Introduction

The USU Operation System is a web-based application for the student union officers. The application is part of the United Student Unions (USU) platform, a cloud-based, multi-tenant system that connects university unions nationwide and is intended to replace manual administrative processes with an integrated digital workflow *(Zhu, 2025, p.1)*. The system is designed so that the officers can manage essential data such as: memberships, societies, events and communication. It's all done through an efficient central online platform. Some of the key objectives of it include enabling real-time updates and notifications for events and society changes, that way students and society leaders get updates efficiently. Another important feature is supporting cross-university services and promotions *(Zhu, 2025, p.2)*. The application is designed for desktop and smart devices and links other USU subsystems so that the actions taken by society leaders or students can give union officers notifications and ask them for approval *(Zhu, 2025, p.2)*. The system runs on a cloud-based platform that allows it to handle multiple university unions at once and take care of common tasks automatically while keeping data secure and reliable *(Zhu, 2025, p.1)*.

Task 2: Analysis and Specification for Software Quality Requirements

Introduction

This section analyses the software quality requirements for the **USU Operation System**, a web-based subsystem of the United Student Unions(USU) platform. It focuses on *Functional requirement FR-USU-1 - Management of Unions*, which defines how USU officers manage the membership of university-specific student unions within the national federation *(Zhu, 2025, p.6)*. To ensure a smooth operation, the analysis takes into consideration the most important software quality attributes (security, performance, reliability and scalability) as they are often identified by professional software-engineering sources *(Gobov and Zuieva, 2025*).

Security and Privacy Protection

Security ensures that only authorised officers can access or update membership records. The USU Operation System should implement secure login with multi-factor authentication (MFA) and encrypted HTTPS communication to protect data transferred between universities *(Zhu, 2025, p.3)*. Role-based access control makes sure that each user can only access and perform actions permitted by their assigned role, while audit logging records every membership change to maintain accountability *(Zhu, 2025, p.3)*. Modern studies emphasises that considering security requirements early significantly reduces vulnerabilities in web-based systems *(Gobov and Zuieva, 2025*). These controls align with the privacy principles defined in KF-5 *(Zhu, 2025, p.3).*

These security measures are important for protecting sensitive student and union data within the multi-tenant USU environment. Because the platform operates across multiple universities, any unauthorised access or data leak could affect thousands of users at once. Making sure strong authentication, encryption and access control will maintain the integrity of shared data *(Zhu, 2025, p.3)*.

Performance

High performance is extremely important to maintaining responsiveness when multiple unions submit requests at the same time. The system must provide quick loading times and near-real-time updates across all connected universities. This aligns with the *Dynamic Update of Information* feature, which needs the system to offer live data to users in real time *(Zhu, 2025, p.2)*. Using cloud services that split tasks between smaller components helps the system handle more requests at once and avoid delays, which matches the multi-tenant design described in the introduction *(Zhu, 2025, p.1)*.

Studies show that system speed has a strong impact on how satisfied users feel when using large online platforms *(Gobov and Zuieva, 2025*). Using efficient data transfer and caching strategies will further support smooth, reliable performance across all universities *(Zhu, 2025, p.2)*.

Reliability

Reliability guarantees that the USU Operation System stays functional and accurate even when technical faults or network problems occur. The system should include automatic backup and recovery tools to protect membership and event data, while redundant servers keep it running if one part fails. These ideas

correlate with the union management requirements described in the case study, which depend on the system remaining available and accurate at all times *(Zhu, 2025, p.6)*.
Effective error handling and rollback functions are important to prevent incomplete updates from corrupting stored data. Keeping the system reliable builds user trust and makes sure of a stable operation, which is especially vital for national-level organisational systems *(Schuszter and Cioca, 2024)*. These reliability measures support the consistent service required by the United Student Unions federation *(Zhu P.6)*.

Scalability

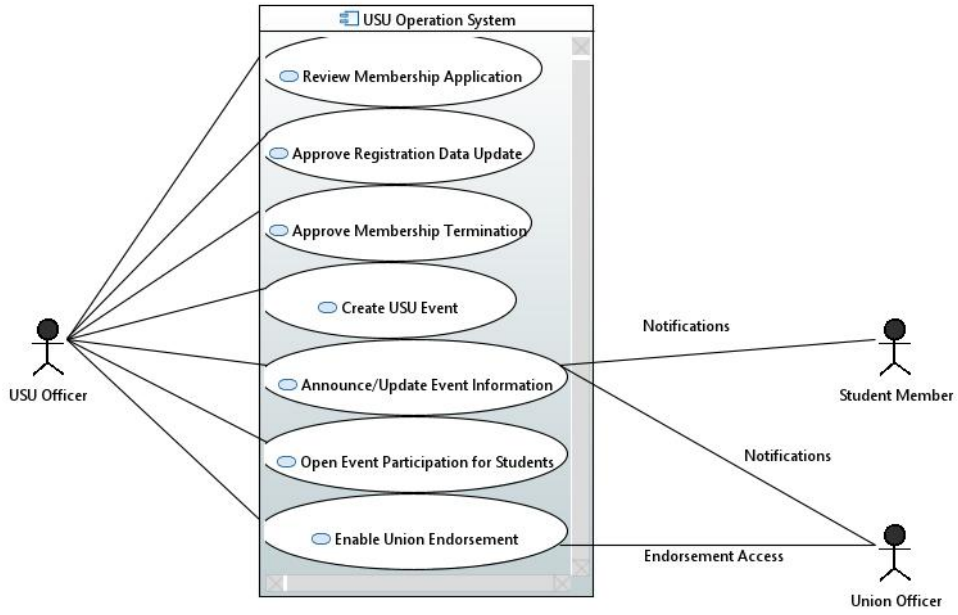
Scalability makes sure that the USU Operation System can grow as more universities, societies and students join the federation. The system should handle the increasing number of users, events and data without losing speed or reliability. This connects directly to the *Services Across University-Specific Student Unions* feature, which states that the system must support nationwide operations involving multiple universities *(Zhu, 2025, p.2)*.
To achieve this, the platform should use a distributed cloud structure that allows it to expand by adding more computing resources when demand increases. Load balancing and modular design will make sure that the system remains responsive even during busy periods *(Zhu, 2025, p.1-2)*. Research highlights that scalability and flexibility are essential attributes of sustainable modern software, allowing systems to evolve without major redesigns *(Schuszter and Cioca, 2024)*.
A scalable design also makes the USU platform future-proof. As new unions register or more students participate in events, the system can easily adapt to the growing workload. This flexibility supports the long-term goal of providing equal access and reliable performance for all member universities *(Zhu, 2025, p.2)*.

Conclusion

In summary, the quality requirements of the USU Operation System makes sure that the platform can operate securely, efficiently and dependably across all participating universities. A strong security helps protect sensitive union and student data, while high performance supports real-time updates and smooth day-to-day operations. Reliability measures offer a stable service even during unfortunate events and scalability allows the system to grow as more unions and students join the federation. All-in-all, these attributes support the goals outlined in the case study and create a solid foundation for managing nationwide student-union activities within the United Student Unions platform.

Task 3: Use Case Model

The following use case diagram illustrates the most important functions of the USU Operation System based on the functional requirements FR-USU-1 and FR-USU-2. It shows the interactions between the USU Officer, Student Member and Union Officer with the main system features.

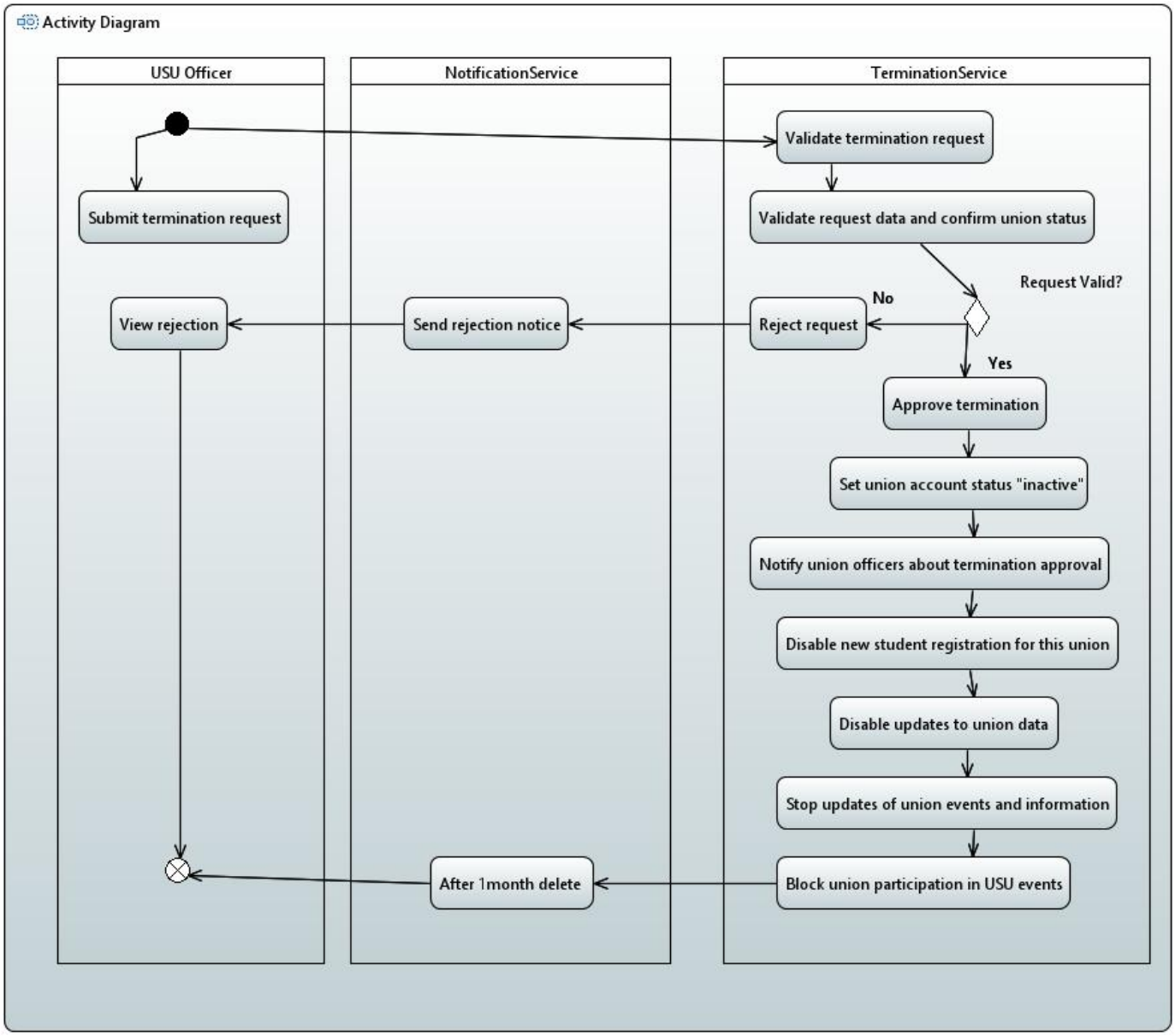


Use Case Description

The USU Operation system enables USU Officers to manage national-level union activities by performing several key actions. They review and approve new membership applications submitted by university-specific student unions, assess updates to a union's registration data, and process termination requests when a union leaves the federation *(Zhu, 2025, p. 6)*. The system also allows USU Officers to create nationwide USU events by entering details such as the theme, time of the event, venue, promotion scope and to publish announcements or updates that are automatically sent to students and union officers *(Zhu, 2025, p.6-7)*. Additionally, officers can enable participation features, allowing student members to subscribe to or register for events. They can enable union officers to officially endorse events, thereby increasing visibility and engagement across the platform *(Zhu, 2025, p. 7)*. Together, these use cases define how the USU Operation System supports the core administrative and event-management functions of the United Student Union federation.

Activity Diagram

This activity diagram models the internal workflow of the use case *"Approve Membership Termination"*, showing the validation, approval/rejection, notifications, restrictions, and data removal defined in FR-USU-1(c).



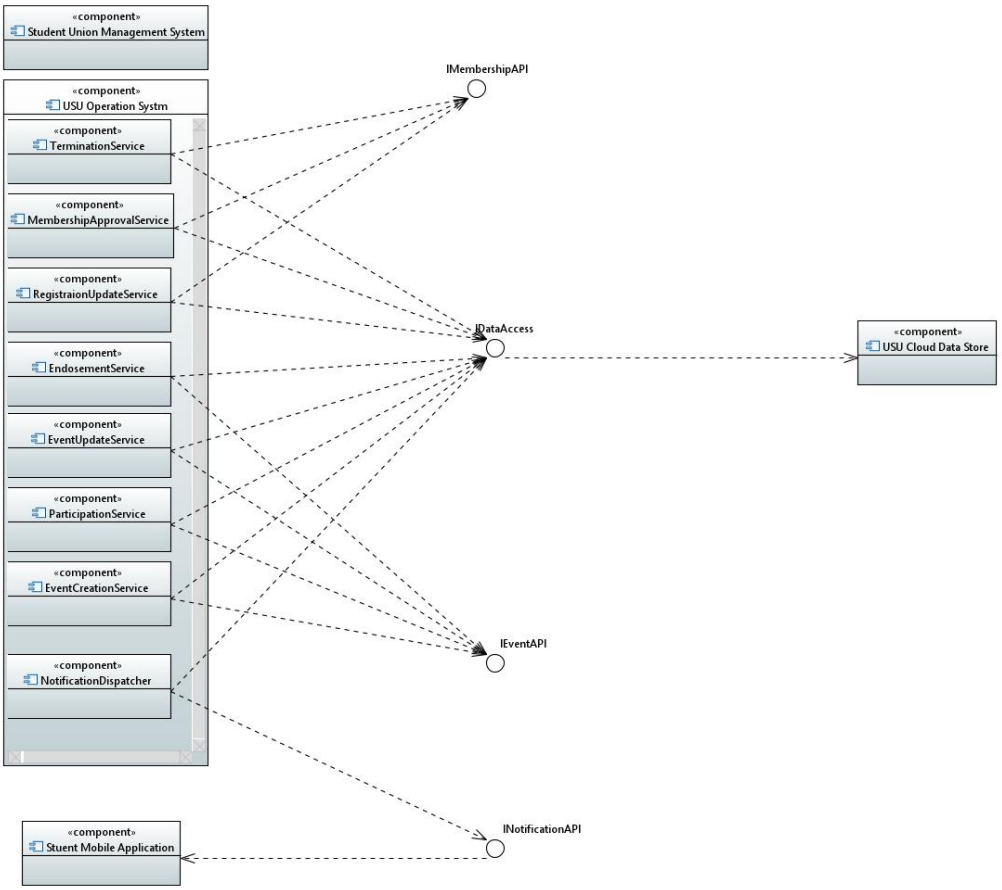
Subsystem Interaction Explanation

The USU Operation System interacts with several other subsystems within the United Student Unions platform to support nationwide student-union activities. It receives membership applications, registration updates and endorsement actions from the Student Union Management System, which is operated by university-specific union officers. It also communicates with the student mobile application by sending event announcements, updates and participation notifications to student members across all universities *(Zhu, 2025, p.2-3)*. The subsystem relies on the shared cloud backend described in the case study, which allows for real-time data sharing and synchronisation between all users *(Zhu, 2025, p.1-2)*. Through these interactions, the USU Operation System acts as the main decision-making and approval layer of the USU federation by coordinating membership, events and cross-union communication.

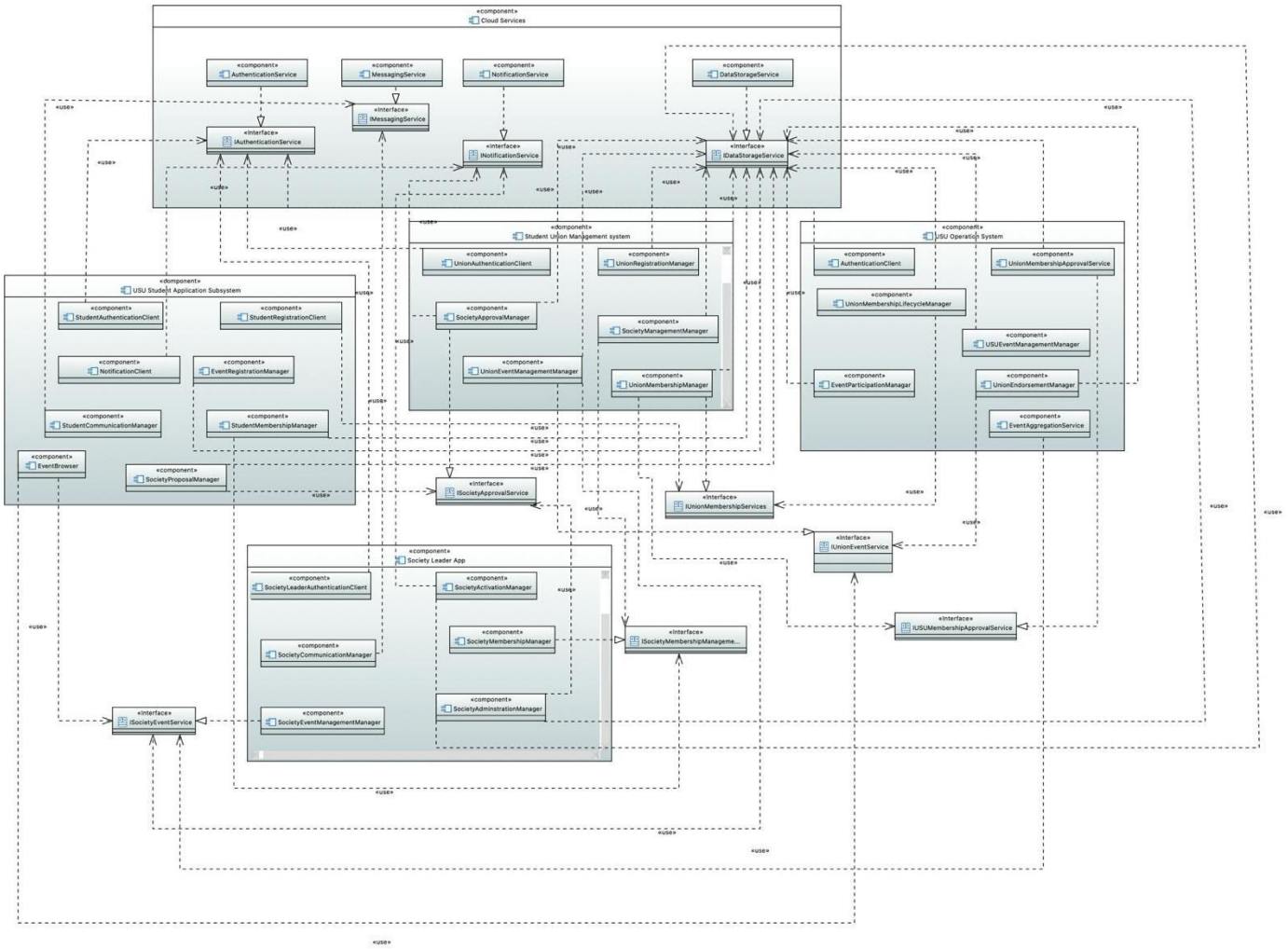
Task 4: Architectural Design

The architectural design of the USU Operation System follows a microservices-based structure, where each functional requirement is implemented as an independent component. The subsystem includes eight microservices that support the membership, event, participation, and endorsement functionality described in FR-USU-1 and FR-USU-2 *(Zhu,2025, p. 6-7)*. The MembershipApprovalService, RegistrationUpdateService and TerminationService handle union membership approval, data updates, and membership termination. EventCreationService, EventUpdateService, ParticipationService and EndorsementService manage nationwide USU event creation, updates, student participation, and union endorsements. The NotificationDispatcher microservice sends real-time announcements and updates to students and union officers.

These microservices interact with external subsystems through well-defined interfaces based on the functional requirements FR-USU-1 and FR-USU-2 in the case study *(Zhu, 2025, p.6-7)*. The Student Union Management System connects via the membership and event APIs to submit applications, registration changes and endorsements. The Student Mobile Application receives real-time event updates through the notification API. All microservices use a shared data-access interface to communicate with the USU Cloud Data Store, ensuring synchronisation and consistency across the federation *(Zhu, 2025, p. 1-2)*. This microservices structure supports scalability, reliability and maintainability by promoting loose coupling between services, which is a recognised advantage of microservice architectures *(Newman, 2021)*.



Task 4b: Architecture ot the whole System



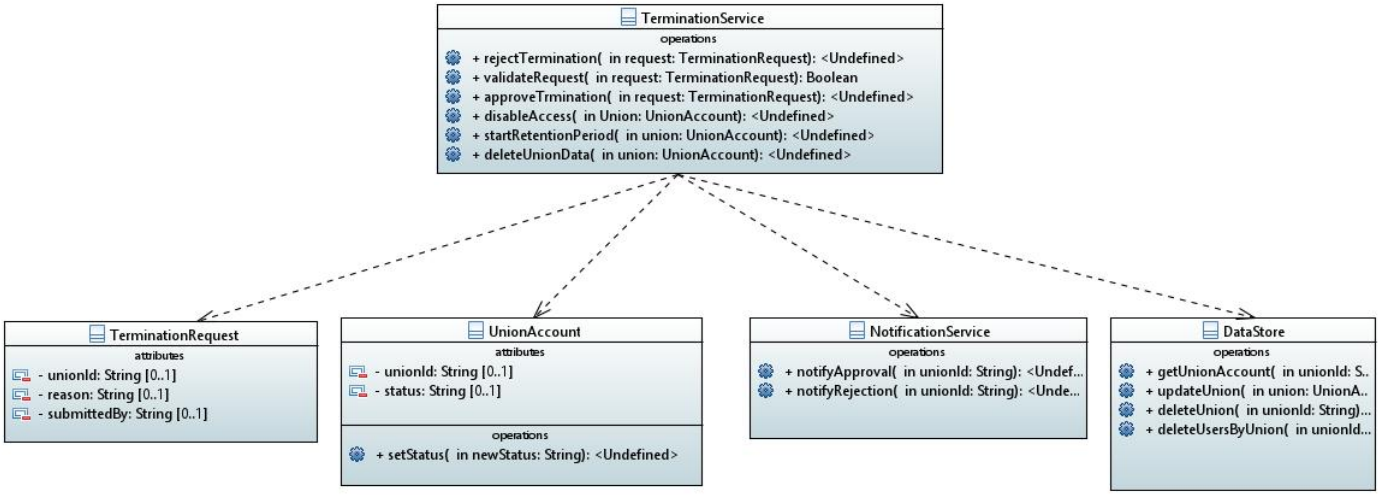
Task 5: Software Detailed Design

Structural Design – Class Diagram

The diagram presented below is the class diagram for the **TerminationService** microservice in the USU Operation System. This microservice implements the "**Approve Membership Termination**" use case described in **FR-USU-1(c)**. It first validates the termination request and then either approves or rejects it based on the conditions defined in the case study. Once the union is marked for termination the accounts in the system are then marked as *inactive*, which then initiates the one-month deactivation period. After the one-month period the union is deleted permanently and every data associated with it *(Zhu, 2025, p.7*).

The central **TerminationService** acts as the control class and collaborates with the following supporting components:

- **TerminationRequest**, which stores the termination details submitted by the university-specific student union, including the union identifier, the reason for termination and the submitting officer.
- **UnionAccount**, which represents the USU-level account of the union and whose *status* attribute is updated during the approval or rejection of termination request.
- **NotificationService**, this is responsible for sending approval or rejection notifications to the relevant union officers, as required by the membership-termination workflow in **FR-USU-1(c).**
- **DataStore** provides the operations needed to retrieve and update union records, disable union access, and permanently delete union data after the one-month inactive period *(Zhu, 2025, p.7)*.

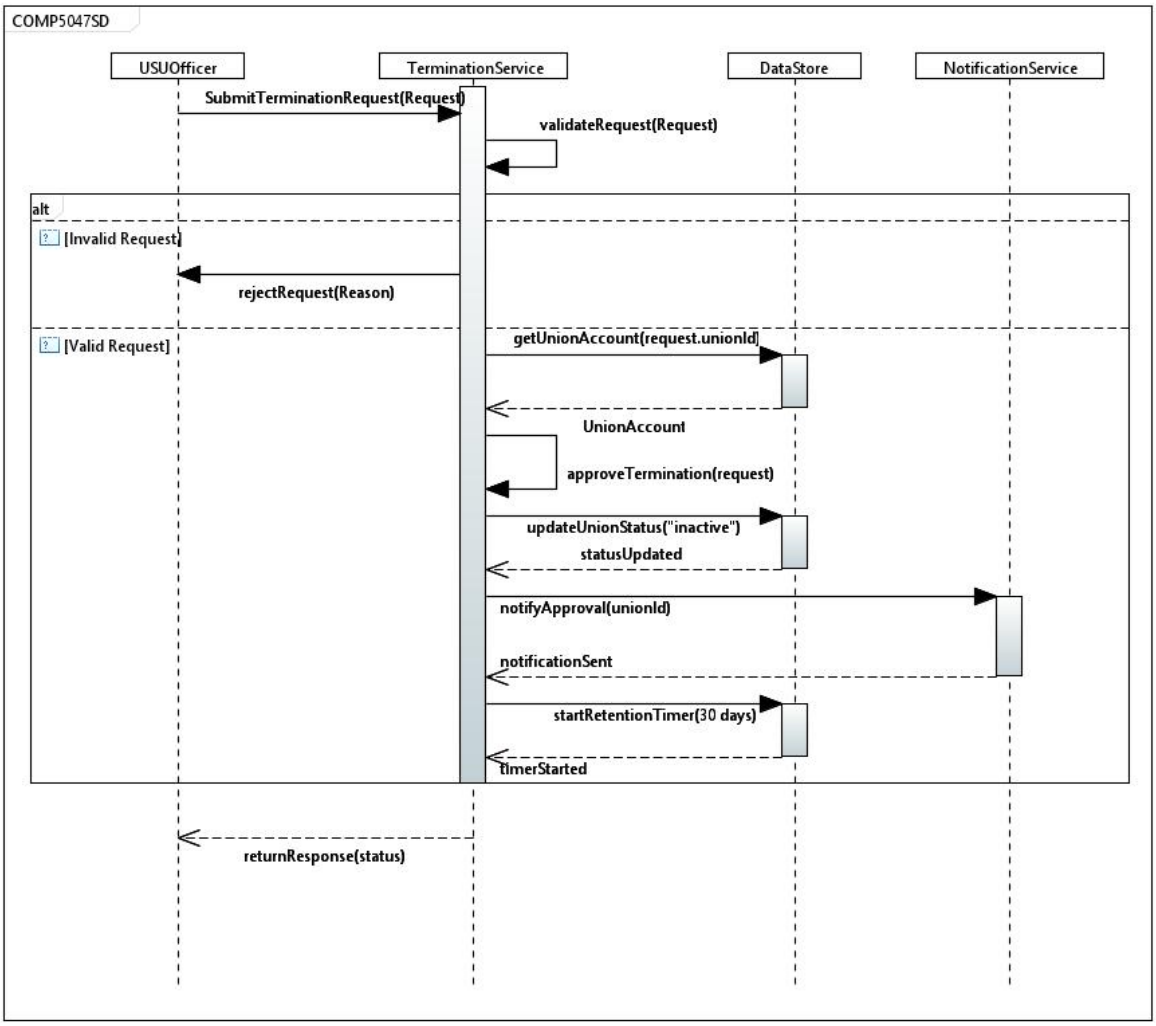


Behavioural Design – Sequence Diagram

The sequence diagram below illustrates the internal behaviour of the **TerminationService** microservice for the "**Approve Membership Termination**" use case described in FR-USU-1(c). The service begins by retrieving the relevant **TerminationRequest** and **UnionAccount** details from the **DataStore** to ensure that the request and union status are up to date *(Zhu, 2025, p. 7)*.

An **alt** frame captures the two possible outcomes: **approval** and **rejection**. In the approval branch, the service validates the request, updates the **UnionAccount** to inactive through the **DataStore**, and triggers the required one-month deactivation period. The service then invokes the **NotificationService** to inform union officers of the approved termination. After the one-month delay, *TerminationService* instructs **DataStore** to permanently delete the union and all associated data, as required by the case study *(Zhu, 2025, p. 7)*.

In the rejection branch, the service records that the request has been rejected, leaves the **UnionAccount** active, and sends a rejection notification through the **NotificationService**. This ensures that both paths defined in FR-USU-1(c) are fully represented. Overall, the diagram is consistent with the class diagram and shows how the service coordinates the **TerminationRequest**, **UnionAccount**, **NotificationService**, and **DataStore** objects to complete the termination workflow.



Individual Teamwork Report

Our teamwork throughout the coursework was overall effective, although at times it was difficult to coordinate and stay fully connected as a group. Communication improved toward the end of the project, which helped us complete the shared architectural tasks more smoothly.

I believe I contributed a fair amount to both the technical work and the collaborative process. I completed all responsibilities for my assigned subsystem and supported group mates whenever they needed clarification or assistance. I also participated in discussions on interfaces and architectural decisions for the group's integrated model.

One challenge I faced was interpreting the **USU Operation System** subsystem correctly. I initially understood it as the system that enables USU Officers to perform federation-level tasks such as membership approvals and terminations, which is accurate according to the case study. However, this caused some confusion within the group when integrating subsystem responsibilities in Task 4(b), as different members had different assumptions. By coming to an understanding we managed to surpass this challenge and continue. I personally still am confused if my interpretation is correct and so are we as a group but there was not much we could do to actually figure it out further.

Overall, despite early coordination issues, I engaged in the project, contributed meaningfully to both individual and group tasks, and worked constructively with the team to resolve misunderstandings and complete the coursework successfully.

References

Denys Gobov, and Oleksandra Zuieva. “Software Quality Attributes in Requirements Engineering.” *International Journal of Information Technology and Computer Science*, vol. 17, no. 4, 8 Aug. 2025, pp. 38–48, www.researchgate.net/publication/394404279_Software_Quality_Attributes_in_Requirements_Engineering, https://doi.org/10.5815/ijitcs.2025.04.04. Accessed 11 Nov. 2025.

Newman, Sam. “Building Microservices.” *O’REILLEY*, O’Reilly Media, Inc, 2021, learning.oreilly.com/library/view/building-microservices-2nd/9781492034018/preface01.html. Accessed 18 Nov. 2025.

Schuszter, Ioan Cristian, and Marius Cioca. “Increasing the Reliability of Software Systems Using a Large-Language-Model-Based Solution for Onboarding.” *Inventions*, vol. 9, no. 4, 15 July 2024, p. 79, www.mdpi.com/2411-5134/9/4/79, https://doi.org/10.3390/inventions9040079. Accessed 12 Nov. 2025.

Zhu, Prof. Hong. *Case Study for Coursework USU*. Oxford Brookes University, 2025.