Faculty of Technology, Design & Environment

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 societyietal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societyietal, 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:	

STUDENT NAMES

	Student Id:	**Student Name:**	**Subsystem:**
1.			
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)*

School of Engineering, Computing & Mathematics

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

The society leader app has several features like creating events managing society information communicating with members and viewing membership details. Because the event creation process involves entering data validate again saving it through the event repository and triggering notifications through the notification service client and promotion service client the subsystem needs to follow certain quality expectations.

Security is important because only true society leaders should be able to access the event creation and management functions. The system should make sure a user is authenticated through the authentication service before they can create or publish an event. The event data that travels from the SoC leader app the event repository and the external services should be sent securely so that details like dates descriptions and locations are protected. The system should also ensure that the only person who created an event and the internal components that handle the storage or notifications can see or use that data.

The performance of the subsystem is important too. When the SoC leader enters event information the event manager should validate it quickly and store it without any delay. The confirmation message that I sent back to the user should appear promptly so that the process is efficient and user satisfactory. Notifications and promotions down both in the sequence an activity diagrams should also begin as soon as the event is saved. The subsystem should be able to handle multiple events of missions or updates at the same time without causing delay especially since the diagram shows several services working simultaneously.

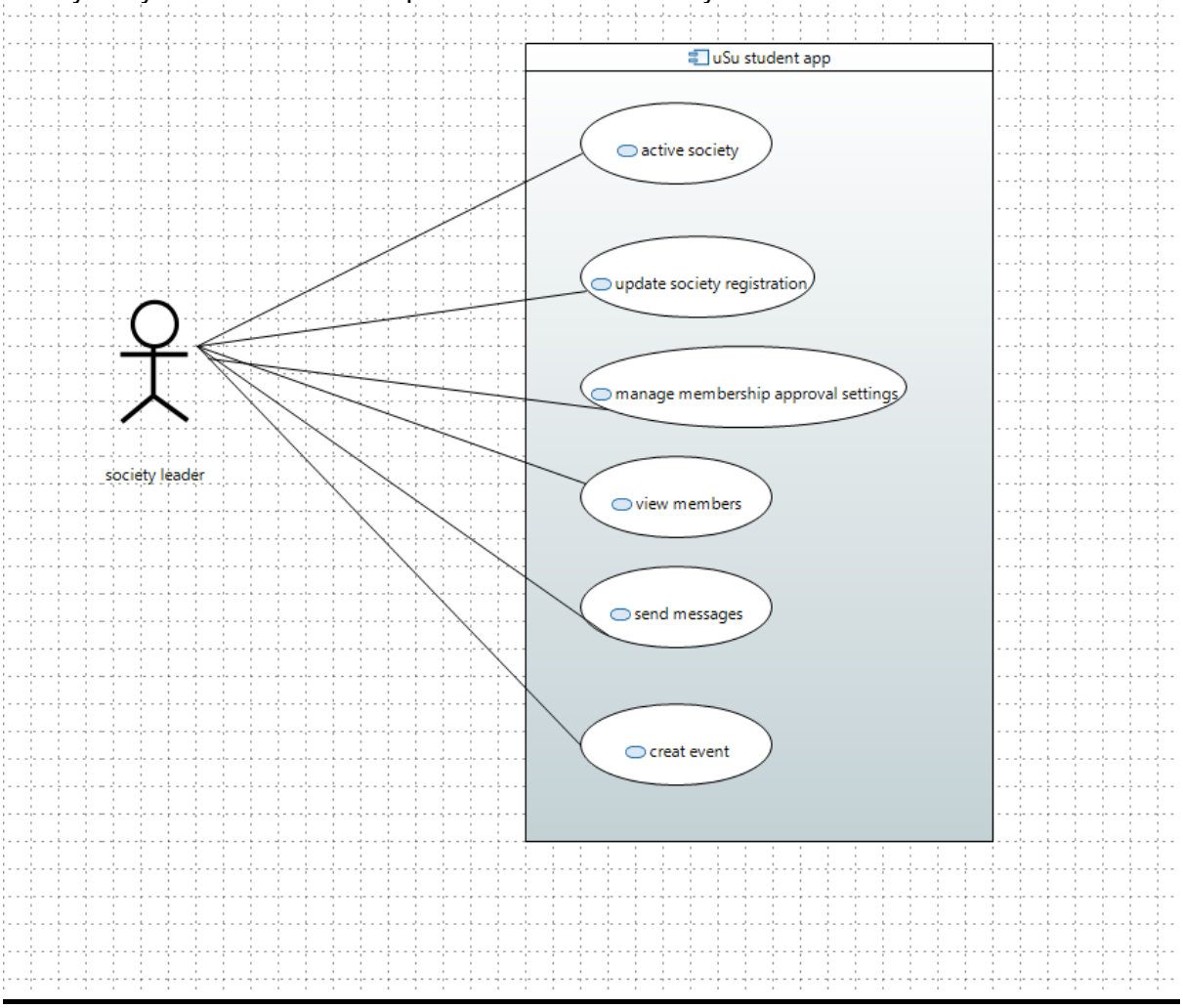
Reliability is needed because the event creation process relies on many components including the event repository and external notification or promotion services. The system should continue working even if only one part of it fails and it should not lose any crucial event information. If something goes wrong during the validation or the saving the user should be informed and the system should recover quickly without losing data. The stored event information should remain safe even if the system has to restart.

The subsystem must scale to support more SOC's events and notifications over a period of time. Since the diagrams show the separate components for managing events sending notifications and promoting events each of these should be able to run on any instance when the system gets busy when other SoC leaders are creating their event at the same time. It should be able to support a growing number of stored event proposals in the back end without slowing down searches or validation.

Overall these quality requirements match the workflows interactions shown in the diagrams they ensure the SoC leader app remains secure responsive reliable and capable of supporting more users and the events in the future.

Task 3A

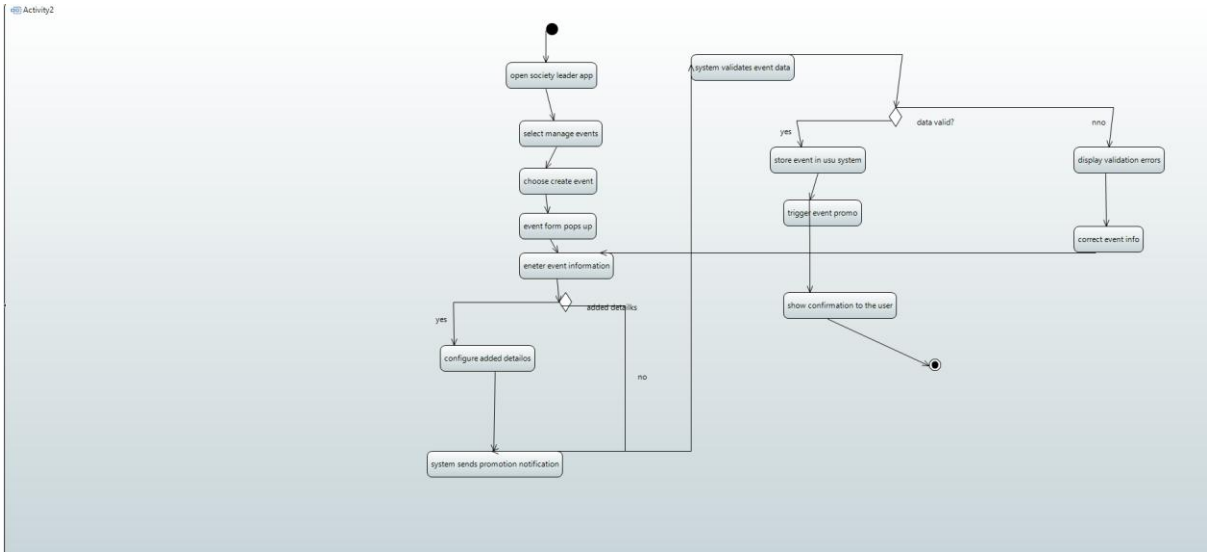
My use case diagram shows what the society leader app does inside the student union app. The Society leader is drawn outside the boundary of the system and shows that they're an external user. Within the boundaries, the diagram includes the actions that they can perform, like activating a society, updating their registration process, managing memberships, viewing their members and sending messages I'm creating events. Each use case represents a feature that my Society leader app must support. This diagram makes it fairly easy to see the main responsibilities of the subsystem and how the user will interact with it.



#

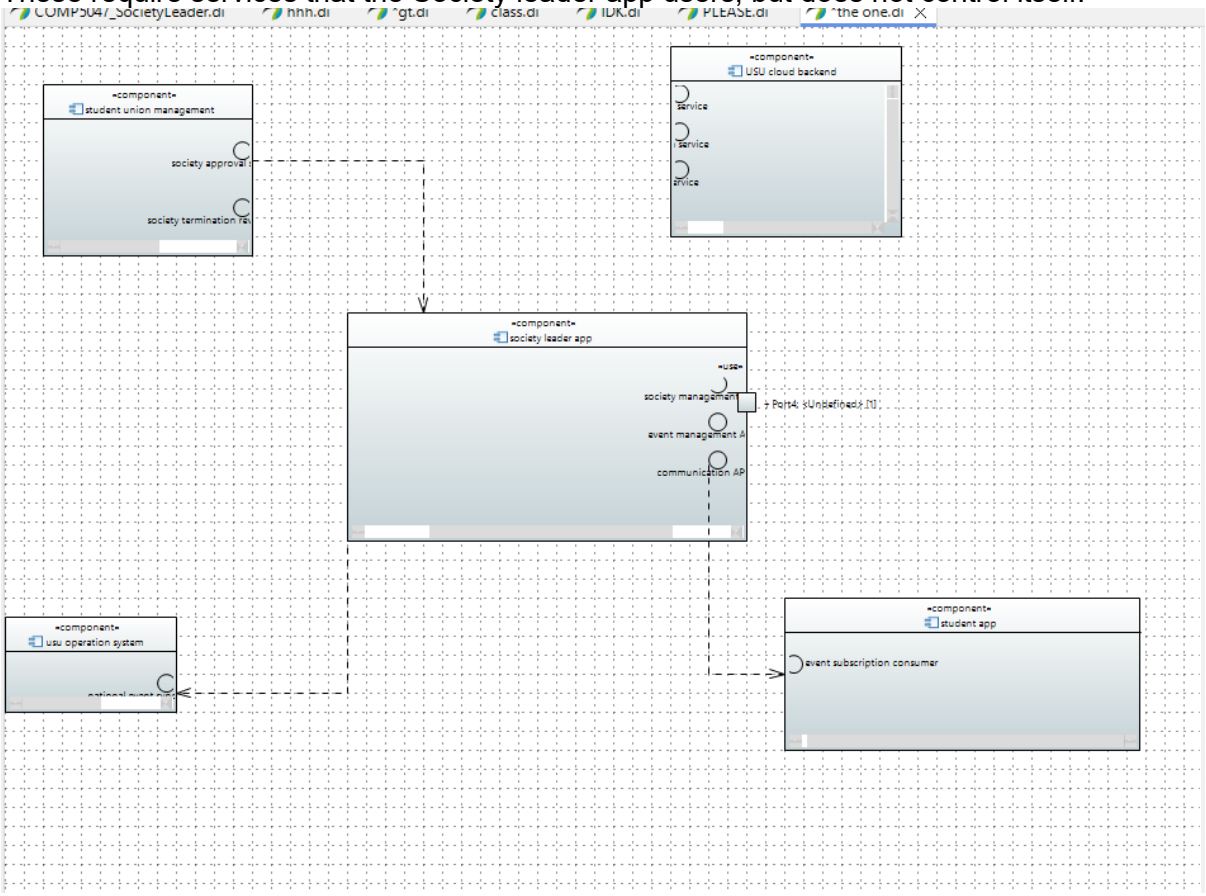
Task 3b

The activity diagram shows the steps that society leader will follow when creating a new event within the app. The process starts with the user opening the society leader app, going to the events section and choosing to create an event. An event form will appear, and the user will then enter the event details. The system will validate the event data. If the information isn't valid, the system shows errors and the user will have to fix them. If the information is valid, the system saves the event in the student union system and may send a promotional notification so that other people know about it. Then the system will show a message to the user to confirm it. This diagram shows each step clearly and explains how the user and the system would work together. It will also use a decision point to show the process can change depending on whether data is valid or not.

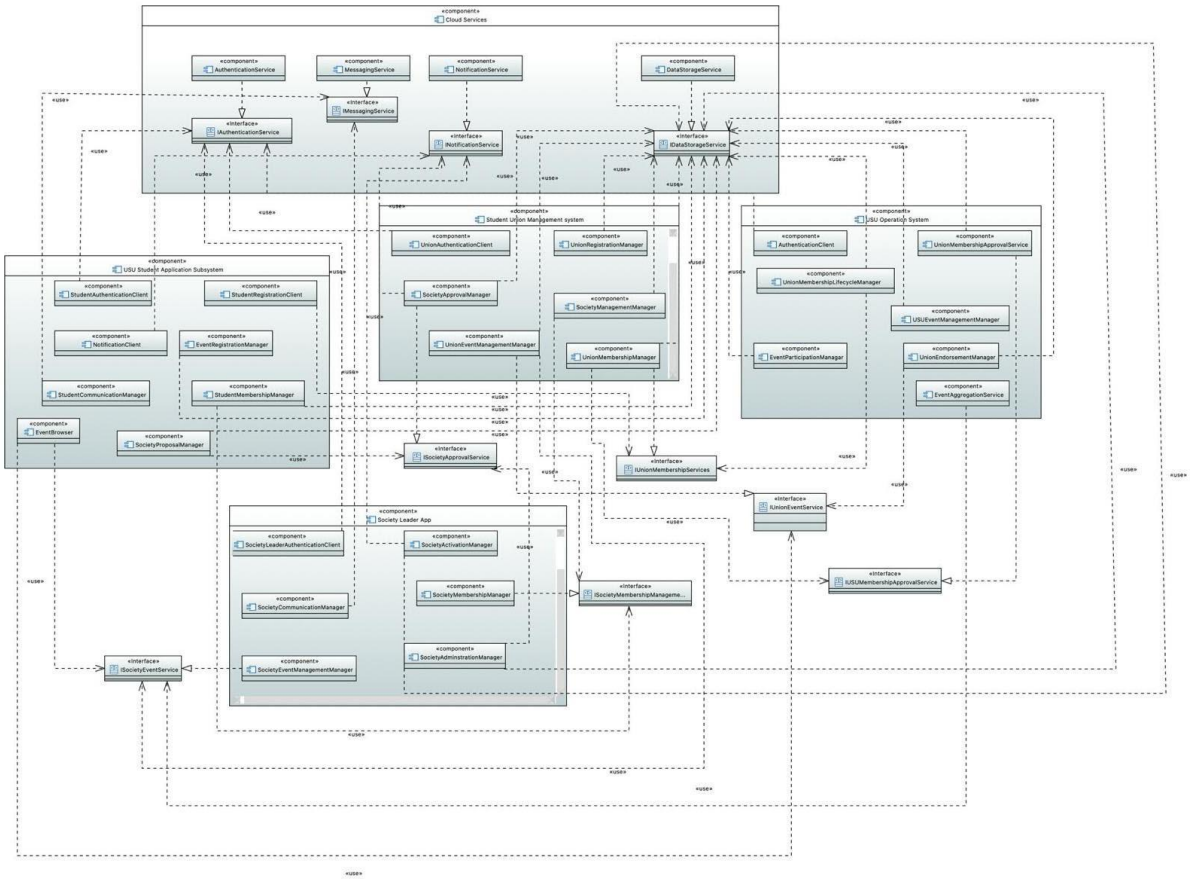


Task4A

My component diagram shows how the Society leader app is organized within the overall student union system. The subsystem is split into smaller compartments, each with its own job. For example, there are components for managing events, managing member communication and society registration. Each component has its own interface which will show the operations that it has to offer. The diagram shows that the subsystem depends on external services like authentication, notifications, messaging and data storage. These require services that the Society leader app users, but does not control itself.

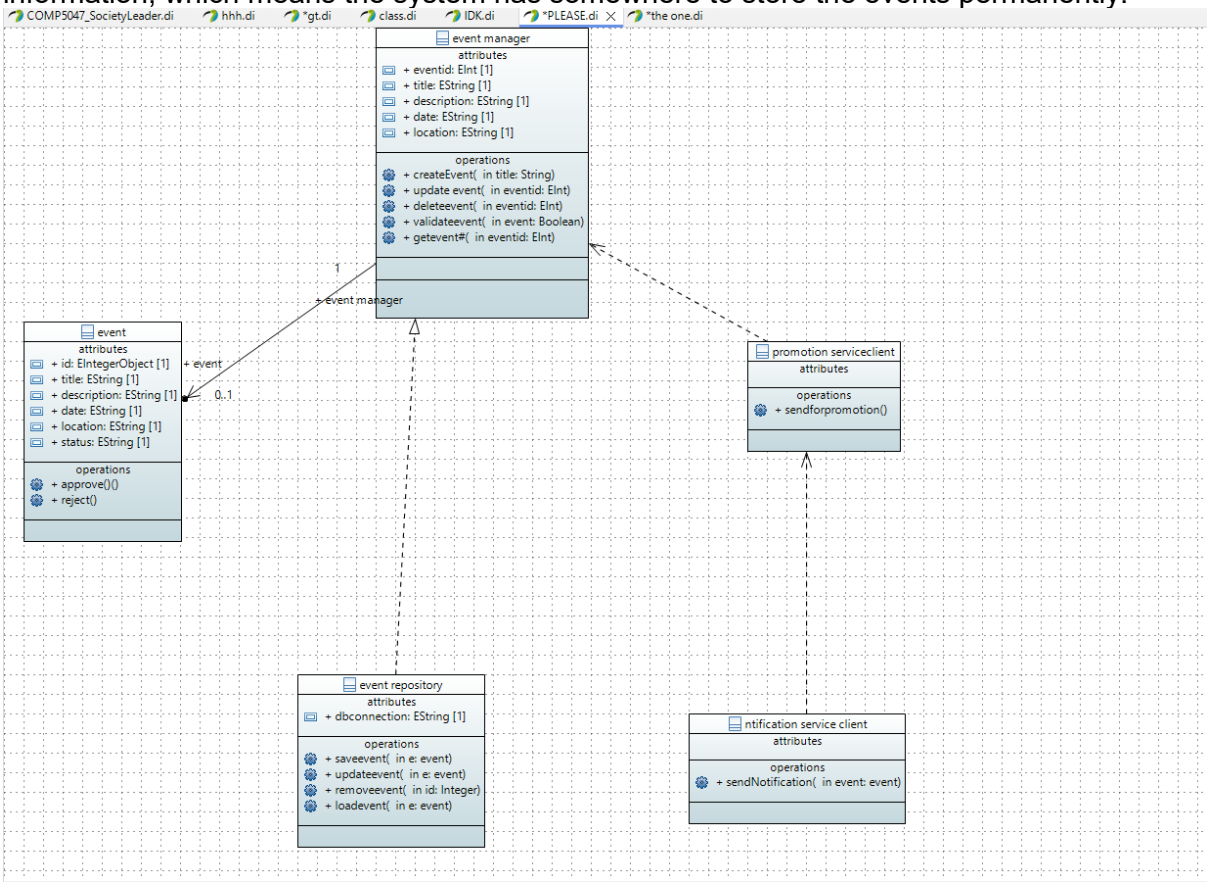


TASK 4B


«component»
Cloud Services
«component»
AuthenticationService
«component»
MessagingService
«component»
NotificationService
«component»
DataStorageService
«Interface»
IMessagingService
«Interface»
IAuthenticationService
«Interface»
INotificationService
«Interface»
IDataStorageService
«use»
«componeht»
Student Union Management system
«component»
UnionAuthenticationClient
«component»
UnionRegistrationManager
«component»
SocietyApprovalManager
«component»
SocietyManagementManager
«component»
UnionEventManagementManager
«component»
UnionMembershipManager
«component»
USU Operation System
«component»
AuthenticationClient
«component»
UnionMembershipApprovalService
«component»
UnionMembershipLifecycleManager
«component»
USUEventManagementManager
«component»
EventPartcipationManagar
«component»
UnionEndorsementManager
«component»
EventAggregationService
«component»
USU Student Application Subsystem
«component»
StudentAuthenticationClient
«component»
StudentRegistrationClient
«component»
NotificationClient
«component»
EventRegistrationManager
«component»
StudentCommunicationManager
«component»
StudentMembershipManager
«component»
EventBrowser
«component»
SocietyProposalManager
«Interface»
ISocietyApprovalService
«Interface»
IUnionMembershipServices
«Interface»
IUnionEventService
«Interface»
IUSUMembershipApprovalService
«component»
Society Leader App
«component»
SocietyLeaderAuthenticationClient
«component»
SocietyActivationManager
«component»
SocietyCommunicationManager
«component»
SocietyMembershipManager
«Interface»
ISocietyMembershipManageme...
«Interface»
ISocietyEventService
«component»
SocietyEventManagementManager
«component»
SocietyAdminstrationManager

TASK 5A

The class diagram shows how the main class is inside the Society's event manager component work together so that they can support the event-creating process. The key classes event proposal which stores all the information about an event into the society leader app to publish it. Includes details like the name, the date, the time, the description and the status of the event. The society leader class represents the user who creates the event and this class is linked to the event proposal to show that the society leaders are able to make multiple event proposals. The event repository class is there to save and load event information, which means the system has somewhere to store the events permanently.



TASK5B

The sequence diagram shows how the app handles the event creation process. It begins with the user sending event details to the event manager. The event manager then talks to the event repository which stores the event within the system. After saving the event, the event manager will send a message to the notification or the promotion service in order for the Society leader or members to be informed. When the event is created, a message is sent back to the user to confirm what has been created.