Individual part

**Task 2**

**FR-UO-1: Management of USU Membership**

1. **Creation of USU Membership**. The system should support university-specific student unions applying for the membership to USU. An application must provide the following data for USU examination and approval.

* Basic data about the student union, such as name and address of the university that the student union is hosted.
* URL of the student union’s website.
* Union representative personnel, include the name, contact details, student identifiers, and the roles in the union, and USU system user IDs as students.
* Delegated union officers, including their names, contact details, and user IDs in USU system as students.

A successful application of USU membership implies:

* The creation of a customised tenant account in the USU system for the university-specific student union
* The storage of the above registration data of the student union in USU computer system.
* The creation of computer access accounts for the delegated IT officers and ready for activate according to the information provided.
* If required, a website for the student union will be created and hosted on the USU portal; otherwise the existing union website will be mirrored on the USU portal.

1. **Update of USU Membership Registration Data**. The system should enable the union to update its data registered to USU, including the change of union representative personnel information and the delegated union IT officers.
2. **Termination of USU membership**. The system should also enable the union to request the termination its membership to USU. Once the termination request is approved, the data about the union will be removed from the system permanently. The union’s access to the system will be disabled.

## Task 2: Analysis and Specify Software Quality Requirements (20 Marks)

In this task, each member of the team will work as a requirements analyst to produce a document that defines the quality requirements on your subsystem. You are required to submit a document as a part of the coursework submission that defines the quality requirements on ONE functional requirement given in the case study document. The definition of quality requirements should clearly specify the requirements on the following quality attributes.

* **Security and Privacy protection**

The security and privacy are main points when creating a system, since this ensures that all the data provided by the user is kept safe and encrypted, to ensure this we should adhere to The Data protection Act (1988) & (2018).

The data protection act 1988 has eight principles which we must consider while processing the system

The Data protection Act ensures that all data used must be processed fairly and lawfully, all data should be obtained only for specified, lawful purposes, must be adequate and relevant and not excessive for the purpose this is collected, must not be kept for a longer period that necessary for the purpose stated, this must be done in accordance with the rights of data subjects, must be kept secure therefore appropriate measures must be taken to protect the data from unauthorised users, accidental loss or any other damage, as well as this can't be transferred to countries outside the EEA unless appropriate level of protection is provided.

* **Performance**

One of the main points when creating a system is ensuring the user satisfaction, therefore we want to ensure that the system response fast and processes memberships applications and updates within two seconds under normal circumstances. Furthermore, we want it to be able to handle up to 500 concurrent users efficiently therefore the database will be optimised. As well as to ensure stable operation during peak activity we should use load balance and caching mechanism, which means no delays or performance degradation.

* **Reliability**

For the system to be reliable we must achieve a high % such as 99% uptime and it recover automatically from failures within minutes, therefore users don't need to wait long periods to access the system again. To safeguard memberships, we will use regular backups, and to prevent partial updates we will use transactional consistency.

Another common fact we should look at is uninterrupted service and data integrity across all operations; to achieve this error detection, logging and recovery mechanisms will help us, maintaining dependable availability for authorised user, which in a way improves the performance too.

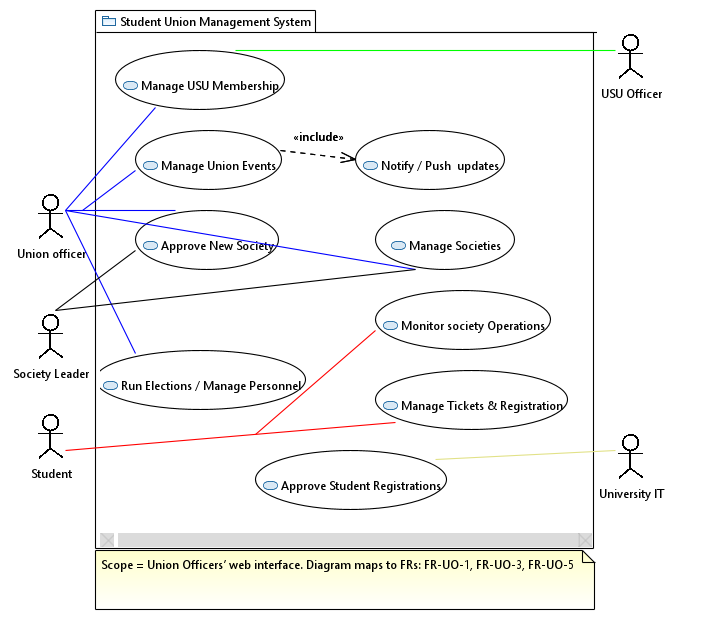
* **Scalability**

A growing number of student unions and users will be accessing this system; therefore, the system must efficiently support them without any sort of loss in the performance. To achieve this, since demand increases, we will use a modular cloud-based architecture allowing resource to be expanded. Furthermore, to manage large scale data the database will support partitioning and replication while sustaining stable performance since the number of members will rise

## **Task 3: Specification and Modelling Software Functional Requirements (20 Marks)**

In this task, you will work as a requirements analyst to produce a UML model of the software system to be developed using the software modelling tool Papyrus. The UML model should contain the following types of models.

1. Use Case Model (10 Marks, Individual effort): Each member of the team should develop one Use Case Diagram to define the use cases of the subsystem to specify the scope of the software engineering project.



**Use case**: Approve New Society

**ID**: APPnSOC

**Primary Acto**r: Union Officer (Blue association)

**Secondary actors**: The one in being charge as applicant is the society leader, furthermore the student union system, University IT will be taking in part for student identity validation.

**Initiating Event:** The one in charge to start the wake the system up will be a registered student, in this case the student leader, who will be submiting a new society application.

**Preconditions**: The applicant must be a valid student, therefore registered and with a student ID. Application form is submitted.

**Postconditions**: In the case this is approved: a society record will be created, allowing the creation of a new society and it being published on the union website and notifying users. If rejected: a notification will be sent regarding the application being rejected and the reason.

**Successful Scenario**:

1. The society leader fills the application form and submits it.
2. Application is stored by the system outputting a message saying: Pending, and notifies the Union Officers.
3. Application in review by the Union Officer
4. Application is approved by the Union Officer.
5. Notification is sent back to the applicant regarding the form.
6. System created a society record, published the society page, opens subscriptions and notifies the society leader about once again.

**Alternative Scenario**:

* In case of the application being incomplete or missing information a notification is sent to the applicant. Society leader updates and resubmits and back to step 2.
* Previously in step 4 the application was accepted, but if this is rejected the system notifies the leader and the flow ends, and this is marked as rejected.

**Frequency**: Few times a month (occasional)

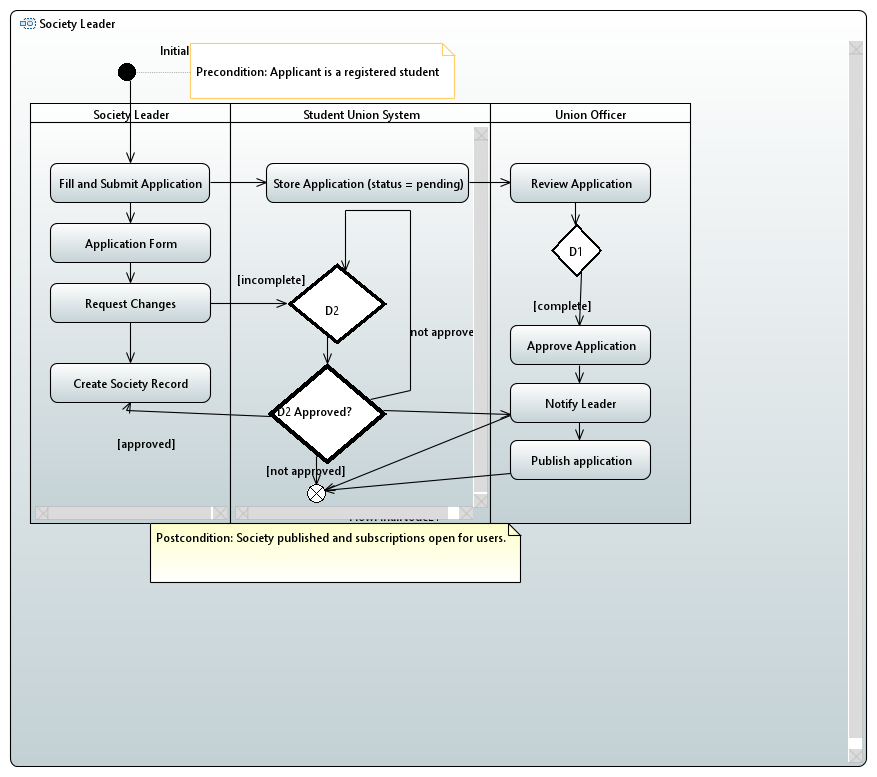
**Related FRs**: FR-ST-5, FR-UO-3

1. Activity Model (10 Marks, Individual effort): Each team member should select one use case of your subsystem to produce one Activity Diagram for the selected use case to specify the interactions between a user and the subsystem.

This UML shows the process behind accepting or rejecting a society, as described in FR-UO-3.

This involves three main participants which are, the society leader, the student union system, and the union officer. The society leader oversees applying, which is stored by the system as pending status. If this is approveed, the system will be published and the society leader notified, if this wasn't the case and it gets rejected, then the system would notify the leader regarding the status, and why.

The process starts when a registered and valid student submits the application, and this ends when its approved, published and open for subscriptions.



## **Task 4: Software Architectural Design (20 Marks)**

In this task, you will work as a software architect to produce an Architectural Design of the system in the microservices architectural style. The design should be at two different levels as follows.

1. Architecture of the subsystem (10 Marks, Individual effort): Each member of the team should produce an architectural design of your subsystem with focus on the microservices your subsystem provides and the microservices that your subsystem requests and other subsystems provide.

1. Architecture of the whole system (10 Marks, Team effort): The team should produce an architectural design of the whole system through integrating the subsystems together.

Note:

1. You should specify the architectural designs in the UML component diagrams with a set of component nodes that represent microservices and a set of interfaces to represent the APIs that are the connectors between them.
2. The components and connectors in architectural designs, including their methods and parameters, should be specified in a textual documentation to define their functionalities and meanings.
3. In this part of the design, you are not required to give the internal structure of the components, which will be the work of the task on detailed design.