My chosen functional requirement: FR-UO-5 – management of union events:

1. *Student Union Management System*: a web-based application to run on desktop or tablet computer for officers of university-specific student union to manage and operate the student union.

**FR-UO-5: Management of Union’s Events**. The system should enable the union to organise events, including setting up of the events and update on the event’s progress. Each event should be registered with the following data.

* *Name*: the identifiable name of the event.
* *Theme*: a brief description of the event.
* *Connection to other event(s)*: a statement about its relation to other events, e.g. previous editions of an annual event.
* *Organisation team*: the name of the organisers and their roles.
* *Date and Time*: the date and time of the event.
* *Venue*: the location/venue of the event.
* *Plan* (Optional): a brief description of the plan of the event.
* *Participation constraints*: whether it is required to register and/or to buy a ticket in advance or on site in order to participate in the event.
* *Scope*: whether the event is open to the public, or limited to the students of nationwide, regional, or the university, or specific groups of students, such as new students at the university.
* *Promotion scope*: whether the scope that the event will be promoted to, such as nationwide, regional, university, or specific groups of students, such as students of a department or faculty, students on a campus, students of a year group, etc.
* *Promotion materials*: The promotion materials of the event, including photos, video streams, text descriptions, posters, URL to website of the event, etc.

The successful creation of an event organised by the union implies that:

* The event will be stored in the USU system and notified to the USU with the above registration data.
* The event will be announced and promoted according to the scope and promotion scope.
* A webpage for a nationwide event will be created on USU website, and promotion materials pushed to relevant users.
* If registration and/or buying a ticket is required to participate in an event, an online facility should be provided.

1. **Specification of Quality Requirements**

My chosen functional requirement: FR-UO-5 – management of union events:

**Security and privacy- how a system protects the information and ensures only authorises users are allowed access to it making sure its secure.**

**Authentication** – only an authenticated union officer is allowed to access creating and managing events. This is to be done using a secure login using email or passwords which consist of /multi factor authentication or making sure a strong password is used e.g., special characters and password length more than 12 etc. Test it by using a bypass attempt by trying to login without second factor and my access should be blocked.

**Role based access or permission:** this is to be done by officers having specific roles such as an individual who organises events like an event coordinator. They should be able to edit, view and add events based on what their roles allows within their limit. As for the union admin they should be able to do all the stuff and deleting and approving events. Role based permission will determine if the user can edit, delete or publish events.

**Data encryption:** all data should use strong encryption (AES-256-GCM) and secure communication using (TLS 1.3) protocol. This is to protect sensitive information and keeping it secure.

**activity logging –** every action should be logged, such as creating event. Editing and deleting event should be logged with timestamps, date, user id and type of action that was taken. The logs should be able to be retained within 400 days. This also helps prevent an individual not taking accountability for their action (non-repudiation).

**GDPR compliance and privacy-** Personal data of student or organiser should be kept to a minimal amount. It should only be allowed to be accessed or seen by a role that has authorisation. If asked to be erased, it should be done within 30 days upon the request.

Overall, this ensures all event data under fru05 is protected from being misused and only authorised union officers can access or manage it.

**Performance – Efficiency the system completes tasks using resources it is given, within a certain time frame.**

**Load time of page –** creating events and listing pages should load under 2 seconds. This can be tested using browser performance tools like webpage test or google chrome dev tools.

**Upload speed** – images and videos less than 50 mb should upload and process inside 5 seconds when uploading promotion materials.

**Current users-** the system should be able to handle more than 20 union officers making changes to an event at the same time. This is to be done without a delay of more than a second. This can be tested using JMeter load testing, this is supposed to be done when multiple officers can update the event at the same time and ensures that even during busy periods the system remains fast and responsive.

**Confirmation/notifications-** upon events being updated or accounts being created, an email notification should reach the intended recipient within 60 seconds.

**response time-** API requests should have an average response time of less than 500 Ms. This can be monitored and verified using performance management tools.

Overall, this ensures fast response time guarantees union officers can manage events smoothly and efficiently.

**Reliability –** this is how reliable a system is meaning if a system can perform the tasks required in conditions which it is said to have been able to handle for a specified time it had stated. This to be done without slowing down or failing.

**System uptime/downtime –** system must maintain an uptime of at least 99.5% a month not including scheduled maintenance. Once again this can be measures using monitoring tools. E.g. uptime records. When system is downtime, it should be on read only mode if database is unavailable. The system shouldn’t be out of service for any more than 100 hour per year. Downtime will affect users’ ability to access events and edit events. This is very important because officer must always be able to have access to the data, especially when updating or handling changes to venues, ticketing or promotion materials.

**Auto save**- System must be able to save the last piece of changes made without being told to. In case of crashing and to prevent data loss so should make saves every 30 seconds ensuring its reliable. This can be tested though UI tests and applies to creating and editing our events.

**Error handling** – in the event the system fails to make saves it should keep trying at least 3 times before outputting an error message. System should be alerting the user to make manual saves and warning them data loss could happen. You should run tests by failing on purpose to make sure system outputs message. This applies to updating events and promotion uploads. Aldo Event data should also all have no missing or incorrect fields inputted, if not appropriate error messages should be outputted.

**Failure rate threshold** – This is how many times a system is allowed to fail per year. It should be less than 5 times a year, this can be tested by checking incident reports. It should also consider the mean time between failures (MTBF) of a maximum of 1000 hours. This is required so that event creation or updates are not interrupted as students could receive wrong or outdated information.

**System recovery** – maximum downtime would be before the system is restored after going down and repaired (MTTR) is 12 hours.

Overall, reliable operations will prevent the loss of the event details and make sure updates reach the USU system consistently.

**Scalability –** Ability of a system to be able to handle extreme loads. For example, more data and events added. This is to be done without having to slow down or lose out on performance and reliability.

**User load handling** – the system should be an able to support at least 100 active users making changes adding or deleting events at the same time or updating events at the same time. We can test this by using load testing tools to run a simulation of having multiple active users at once and increase them slowly. This will be making sure its scalable.

**Data handling** – system should be able to handle the data base having up to 100000 events stored at once without the system slowing down. It should also ensure this is done without API response time decreasing and effecting user usage. So many events should store within the system.

**Resource adjustment** - system should adjust resources like CPU memory and storage when loads increase more than 80% so the system remains smooth without slowing down. The System should use vertical scalability e.g. It should increase allocated CPU and memory resources if this does happen by 20%, this could be needed when large files could be uploaded to a union. Horizontal scalability should also be used when too many officers add events at the same time system can create extra servers to share workload.

Overall, this ensures the system can handle growth in the number of events, users and uploaded media as the union will become bigger.

**2. Software Functional Modelling**

**2.1. Use case model**

**A screenshot of a diagram

AI-generated content may be incorrect.**

**2.2. Activity model**

**A screenshot of a computer

AI-generated content may be incorrect.**

**3. Software architectural Design**

3.1. Architecture of the subsystem

A screenshot of a computer

AI-generated content may be incorrect.

**3.2. Specification of the components and interfaces**

**Interfaces**

|  |  |
| --- | --- |
| **Component** | EventGuiManagement |
| **Description** | This is a microservice running on the cloud to process the event creation. |
| **Stereo Type** | Service |
| **Required Interface** | IAuthorization, Istorage |
| **Provided Interface** | IEventManagement |

|  |  |
| --- | --- |
| **Component** | EventPromotionManagement |
| **Description** | This is a microservice running on the cloud to process the event promotion material uploads. |
| **Stereo Type** | Service |
| **Required Interface** | Istorage |
| **Provided Interface** | PromotionUpload |

|  |  |
| --- | --- |
| **Component** | **EvevntRegistrationManagement** |
| **Description** | This is a microservice running on the cloud to process and configure the ticketing/registration |
| **Stereo Type** | Service |
| **Required Interface** | IAuthorisation |
| **Provided Interface** | RegistrationService |

|  |  |
| --- | --- |
| **Component** | **NotificationManagement** |
| **Description** | This is a microservice running on the cloud to process and notify the creation of events or changes |
| **Stereo Type** | Service |
| **Required Interface** | Inotification |
| **Provided Interface** | ValidationService |

**Components**

|  |  |
| --- | --- |
| **Component** | **Database** |
| **Description** | This is a microservice running on the cloud to store all notifications, event details/updates, ticketing, promotion etc. |
| **Stereo Type** | Service |
| **Required Interface** | Inotification |
| **Provided Interface** | Istorage, IAuthorization |

|  |  |  |
| --- | --- | --- |
| **Name** | **IAuthorization** | |
| **Provider** | **Database** | |
| **Operation** | **Signature** | CheckData(userid: string,action: String) |
| **Function** | Checks if the officer has entered the correct data to  be stored into the database and gives authorization,  if not returns a failure. |
| **Operation** | **Signature** | CheckTicketing(TicketRequired: Boolean,  registrationRequired: Boolean, onlineTicketing: Boolean) |
| **Function** | Check if the ticketing registration is needed or not and  Gives authorization. |

|  |  |  |
| --- | --- | --- |
| **Name** | Istorage | |
| **Provider** | Database | |
| **Operation** | **Signature** | ChecksStorage(saveEvent in (event:Event Event):void,  loadEventIinid: String) Event,  deleteEvent(in id: String): Boolean |
| **Function** | Interface to send the correct promotion and data before  sending to the database |

|  |  |  |
| --- | --- | --- |
| **Name** | Inotification | |
| **Provider** | NotificationManagement | |
| **Operation** | **Signature** | Notifies(successful:Boolean) |
| **Function** | Interface to send notification if its true or false and if  event been created and stores to send to officer. |

|  |  |  |
| --- | --- | --- |
| **Name** | IEventManagement | |
| **Provider** | EventGUIManagement | |
| **Operation** | **Signature** | createEvent(in data:EventData):Event,updateEvent(in id  : string,in id: EventData)Event), cancelEvent(in id: String):  Boolean, getEvent(in id:string):Event  listEvent(in Filter:EventFilter):Event. |
| **Function** | Interface To send the Event updating, creating details to  Cloud component. |

|  |  |  |
| --- | --- | --- |
| **Name** | RegistrationService | |
| **Provider** | EventRegistrationManagment | |
| **Operation** | **Signature** | RegistrationRequired:Boolean |
| **Function** | Interface To send the Event ticketing details to  Cloud component. |

|  |  |  |
| --- | --- | --- |
| **Name** | notificationService | |
| **Provider** | NotificationManagment | |
| **Operation** | **Signature** | Send Notifications(successful:Boolean) |
| **Function** | Interface To send the Event notification to the cloud |

|  |  |  |
| --- | --- | --- |
| **Name** | PromotionService | |
| **Provider** | EventPromotionManagement | |
| **Operation** | **Signature** | Material: String[1], type:MaterialType[1],url:Boolean[1],  Description:String[0..1] |
| **Function** | Interface To send the event promotion details to the  Cloud. |

|  |  |  |
| --- | --- | --- |
| **Name** | ValidationService | |
| **Provider** | Database | |
| **Operation** | **Signature** | ValidateForCreating(in data:Event), validateForUpdate(  Ine event: Event) |
| **Function** | Interface to send validation to the cloud to access event detail for monitoring |

|  |  |  |
| --- | --- | --- |
| **Name** | EventDetailAccess | |
| **Provider** | Database | |
| **Operation** | **Signature** | name: String  theme: String  dateTime: DateTime  connectionInfo: String EventId: String  Venue: String plan: String scope: EventScope  promotionScope: PromotionScope  participation: ParticipationConstraints  materials: PromotionMaterial  status: EventStatus |
| **Function** | Interface To allow searching of the events connected  to events stored in the database. |

**4. Detailed Design**

**4.1. Structural model**

A screenshot of a computer

AI-generated content may be incorrect.

**4.2. Behavioural model**

**5. Reflection on Teamwork**

As an active member of the group, I contributed significantly to the organisation, coordination and overall management of the team throughout the coursework. I attended all meetings and chaired most of them (3.6,7,8,9) as shown in the team report. I prepared meeting agendas, guided discussions, recorded minutes, and ensured the group stayed on track with deadlines.

A major part of my roles was:

Organising the meetings and leading them and assigning the group with the task to create GitHub folders on time for us all to upload the tasks.

Uploaded all my work early to complete it especially 5a and 4a whilst setting a consistent pace.

I reviewed and maintained the repository structure, ensuring everyone’s files were stored correctly and consistently.

Led creation of the team report, as I took initiative to get everything done on time and uploaded the main draft for everyone to review and make relevant changes to as seen in the team meeting sand the git repository.

I also acted as a point of stability for the group reminding them all the deadlines and helping our workflow stay consistent. During the later weeks I took even more initiative setting the standard and organising the final meet ups and meeting along with doing the team report mainly and even had most of the members telling me do be the group leader.

Across the project I took responsibility for chairing meetings and keeping records, organising who creates the folders so I can upload mine work I had done early on to keep everyone involved, ensuring team members upload files on time and correctly, reviewing team progress and helping resolve misunderstandings due dates etc. preparing templates for the final report, doing the final team report, asking members to review it and uploading 5a and 4a early for team members to get a understanding.

In meeting 3 I studied the requirements for the student union management system and began planning it. I helped clarify the abstraction everyone needed in their use case and supported the team in how to structure theirs as leader of the week’s meeting.

I lead week 6 meeting and set the agenda to start reviewing 4a and encouraged everyone to cross check each other’s work in week 3 as stated and reviewed all the members work.

I lead week 7 and told everyone to start looking at 4b so we can start working on it soon, I told everyone to review task 4 so we all reviewed our task 4. I also made the changes from 3b which was advised to. I had uploaded my 4a and was waiting on the rest to do theirs so completed my task on time.

In meeting 8 I lead again; I refined my architecture design and re uploaded all my works after refining them along with my 5a and began to do the team report and organised we all met to do the 4b teamwork.

In the final meeting which I lead again I completed 5a first and showed my team, so they know and we showed all the 5b which we completed on time where we scheduled it a day before submission.

I reviewed hamza 5a and 5b and then all his other work for proof reading. I read the final submission before josh submitted to make sure everything correct. I had completed all the final report and made sure we didn’t miss anything.

Overall, I participated in all the meetings, lead most of them, organised who will create the folders, organised the meet up for 4b, created most of the team report, always done my work on time, and gave appropriate feedback when needed and reviewed all the work as shown in team meetings in GitHub. This helped us meet consistency and all the coursework specifications and working well as a group.

11/12/2025

13:00-13:30

meeting chairman: Mohammad

Attendances:

Mohammad

Hamza

Josh

Luke

Absentee:

All present

Agenda:

For Everyone:

-review 5a and b

-review all work

-Mohammad finished team report and upload it for rest to look at again after last meeting where he created most of it to confirm its all correct.

Minutes:

-Mohammad reviewed all of Hamza work on 5a and 5b and then proofread all his tasks and gave feedback

-Hamza read all of Mohammad work on 5a and 5b and then proofread all his tasks and gave feedback

-Luke read all of josh Mohammad work on 5a and 5b and then proofread all his tasks and gave feedback

-josh read all of Luke work on 5a and 5b and then proofread all his tasks and gave feedback

Actions to be taken:

For everyone:

* Submit all our work