5047 Personal Part Coursework

Team Organisation and project management

From the very beginning of the project, I took the initiative to create and configure the team's main workspace. At the very first meeting, I created a GitHub repository (19324937), added all participants, configured the folder structure, protected the main branch, and set up basic rules for working with Git.
Throughout the semester, the team worked at its own pace, but in the first half, there were noticeable problems with discipline and deadlines: meetings were sometimes missed, and tasks were left unfinished for weeks. About halfway through the project, I suggested introducing clear internal deadlines for each stage - after that, the situation improved significantly, and we stopped missing intermediate deadlines.
Most of the communication took place in a shared WhatsApp group. When technical problems arose, we worked as a team and looked for solutions together.
To conclude, I am happy with team work because the main scope was to complete the project to the deadline and cover all the main steps in coursework specification.

Subsystem 3:

USU Operation System: a web-based application to run on desktop or tablet computer for USU officers to manage the USU federation of student unions and realise its functions

Quality Requirements

Overview

USU Operation System: a web-based application to run on desktop or tablet computers for USU officers to manage the USU federation of student unions and realise its functions. The main function for the USU Operation System is Approve New Student Union Registration. This is the key function for USU Operation System: USU managing and approving new customers. Also it is included important factors: security(personal data), performance(number of requests in the same time), reliability and scalability (for number of universities)

Quality Requirements for Approve New Student Union Registration function.

When a university representative applies to join USU, a USU officer must have an interface that enables them to interact with the user and manage their access to the platform. This interface should be implemented as an admin panel. The functionality should include viewing applications, verifying documents and accepting or rejecting applications. Additionally, the system should have a notification feature to inform users of the successful or unsuccessful outcome of their application.

Security and Privacy Protection

The primary objective is to safeguard the personal and organisational data of the applicant ensuring that only authorised officers can make decisions regarding the application. For this purpose authentication, role-based access control and OAuth2 will be utilised. Role-based authorisation would be a reliable option. The hierarchy would be structured as follows: the administrator holds the most significant role with full platform access, enabling them to grant and revoke rights and view the history of officers' decisions. Approval or rejection of decisions must be recorded in a table with timestamps user ID and IP address the officer who made the decision and a view of the officers' activity history. Personal data is only accessible to those with the appropriate rights and this is determined by their role. This will ensure the system's reliability and protect against security breaks risks.

Performance

To ensure fast platform performance, a user-friendly UI is necessary to enable easy manipulation and management of lists. This requires the implementation of list caching pagination and asynchronous processing to allow lists and the platform to operate asynchronously and handle multiple requests simultaneously.

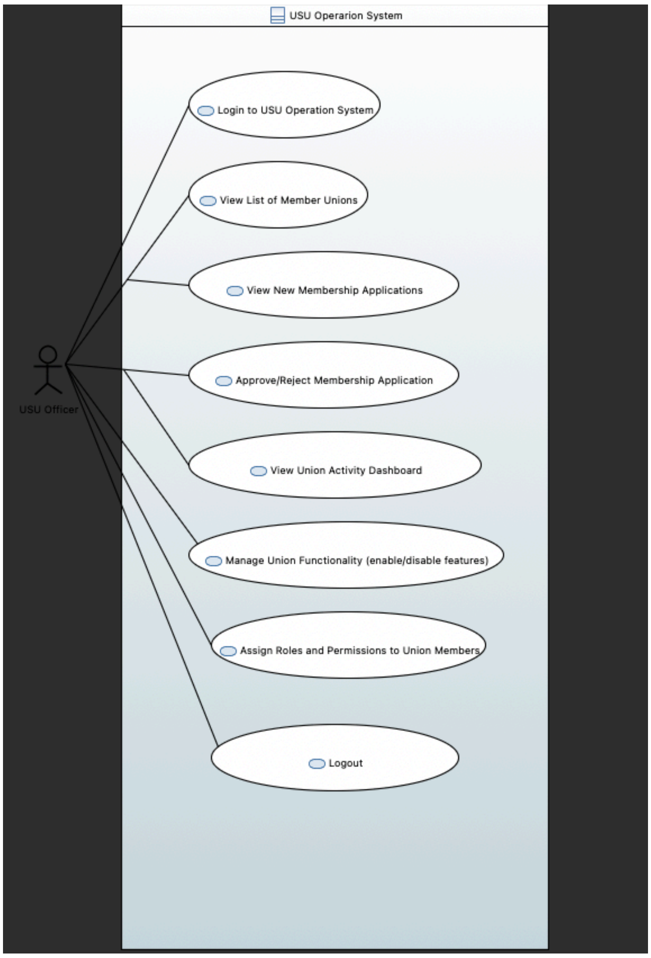
Reliability

System reliability is also a critical consideration as the system must be accessible and data integrity must be maintained in the event of failures. This necessitates the implementation of a backup storage system that will automatically activate in the event of a failure. A daily database backup will enable recovery of the last 24 hours. Error handling and validation will assist users in understanding platform functionality and a retry mechanism for background tasks will improve system resilience.

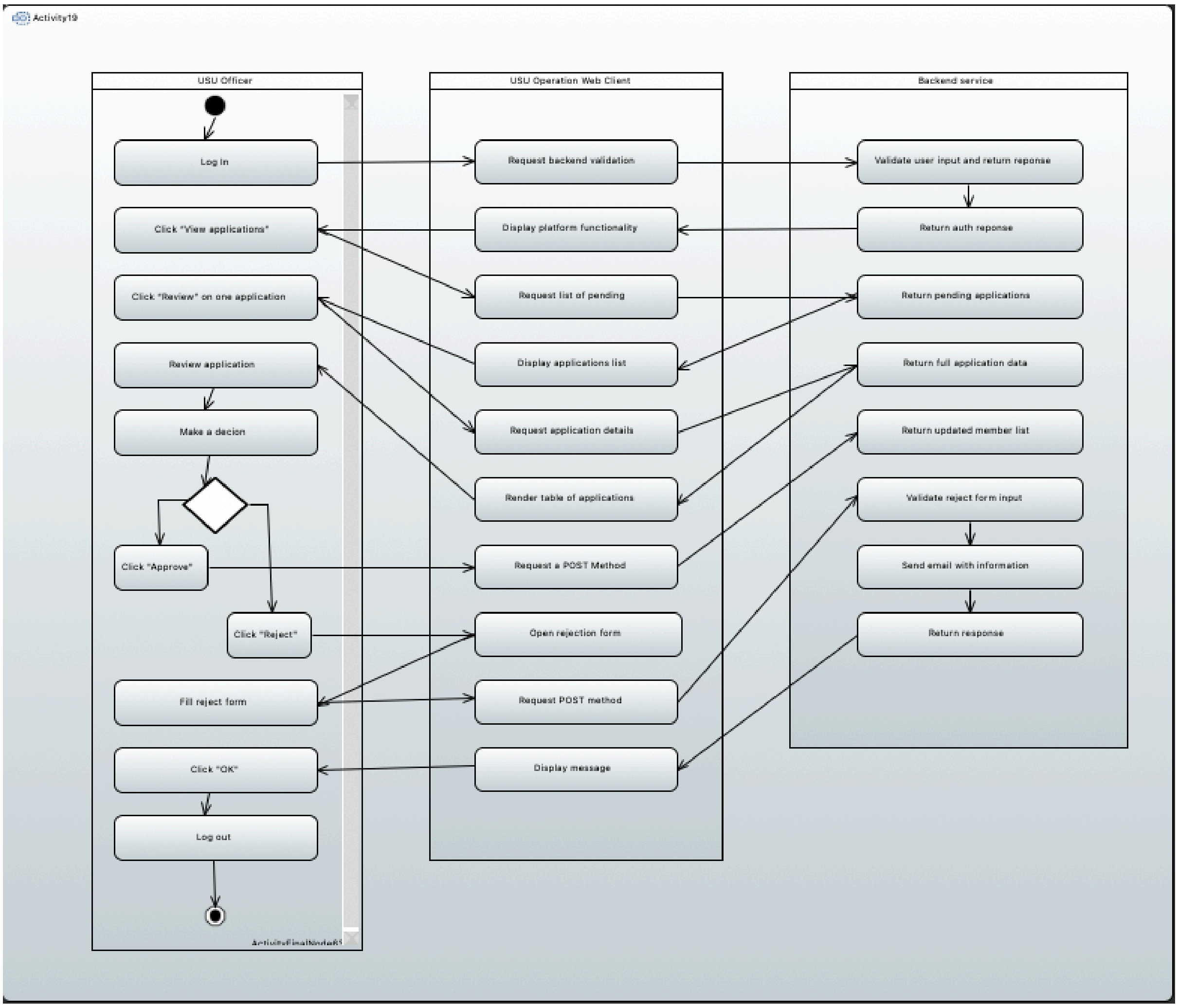
Scalability

Scalability is contingent upon the database system structure and project architecture. To ensure support and scalability, it is advisable to consider project goals in advance and construct the project according to clear criteria to ensure consistency and logical flow. This approach will facilitate a rapid and efficient system development. Key aspects of scalability include the database, its branching and load. The architecture should incorporate calculated dependencies to manage large data volumes effectively.

Use Case Diagram:


USU Operarion System
Login to USU Operation System
View List of Member Unions
View New Membership Applications
Approve/Reject Membership Application
View Union Activity Dashboard
Manage Union Functionality (enable/disable features)
Assign Roles and Permissions to Union Members
Logout
USU Officer

Activity Diagram:


Activity19
USU Officer
USU Operation Web Client
Backend service
Log In
Request backend validation
Validate user input and return reponse
Click "View applications"
Display platform functionality
Return auth reponse
Click "Review" on one application
Request list of pending
Return pending applications
Review application
Display applications list
Return full application data
Make a decion
Request application details
Return updated member list
Render table of applications
Validate reject form input
Click "Approve"
Request a POST Method
Send email with information
Click "Reject"
Open rejection form
Return response
Fill reject form
Request POST method
Click "OK"
Display message
Log out
ActivityFinalNode8

Software Architecture Design

Component	OfficerUI
Description	Web interface used by USU officers to view dashboards, manage unions.
Stereo Type	Component
Required Interface	RetrieveAdminData, UpdateAdminData
Provided Interface	

Component	AuthenticationService
Description	Central Identity provider
Stereo Type	Component
Required Interface	
Provided Interface	

Component	Union Management Service
Description	Core microservice to manage unions
Stereo Type	Component
Required Interface	ModifyUnion, RetrieveUnion
Provided Interface	

Component	Role Management Service
Description	Microservices to manage roles inside each of the union
Stereo Type	Component
Required Interface	UnionMemberList, AssignRole
Provided Interface	

Component	Activity Service
Description	Microservice to view and control union activity on the platform
Stereo Type	Component
Required Interface	UnionList

Provided Interface	

Component	Email Service
Description	Microservice to send email with information
Stereo Type	Component
Required Interface	SendEmail
Provided Interface	

Cloud System:

Component	AuthenticationManager
Description	Validates officer credentials, sessions, and permissions
Stereo Type	Component
Required Interface	Authenticate, CheckPermission
Provided Interface	

Component	UnionManagement Service
Description	Service for managing unions
Stereo Type	Component
Required Interface	ModifyUnion, RetrieveUnion
Provided Interface	CheckPermission

Component	Activity Service
Description	Service for control union activity
Stereo Type	Component
Required Interface	UnionActivity
Provided Interface	CheckPermission

Component	EmailService
Description	Sends targeted emails
Stereo Type	Component
Required Interface	SendEmailNotification
Provided Interface	

Specification of Interfaces

Name	CheckPermission	
Provider	AuthenticationManager	
Operation	Signature	verifyOfficerAccess()
	Function	Confirms officer permissions for admin tasks
Operation	Signature	decodeOfficerToken()
	Function	Extracts officer role, assigned union, and privileges

Name	Authenticate	
Provider	AuthenticationService	
Operation	Signature	officerLogin
	Function	Login function for officer
Operation	Signature	refreshToken
	Function	Keep alive session

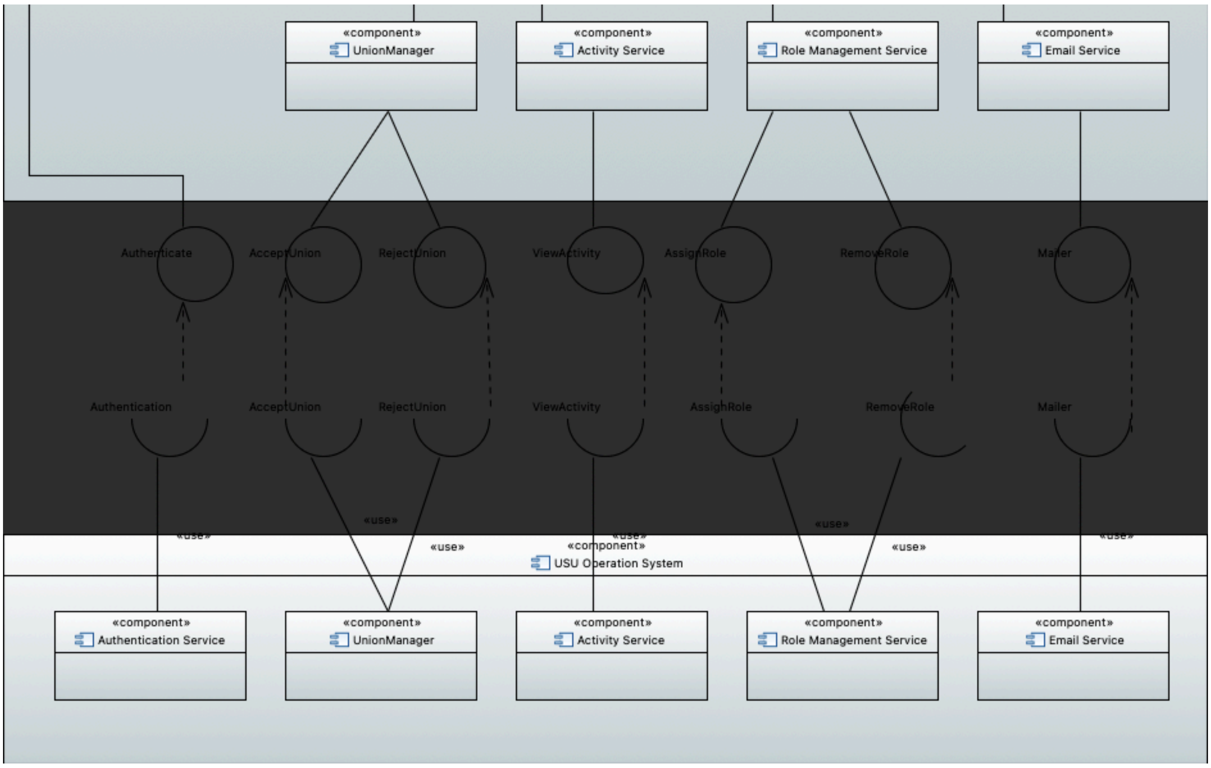
Name	CheckPermission	
Provider	AuthenticationService	
Operation	Signature	verifyOfficerRole
	Function	Check for assigned role to the customer

Name	ModifyUnion	
Provider	UnionManagementService	
Operation	Signature	createUnion()
	Function	Function to create new union
Operation	Signature	updateUnionInfo()
	Function	Function to update union
	Signature	deleteUnion()
	Function	Function to delete union

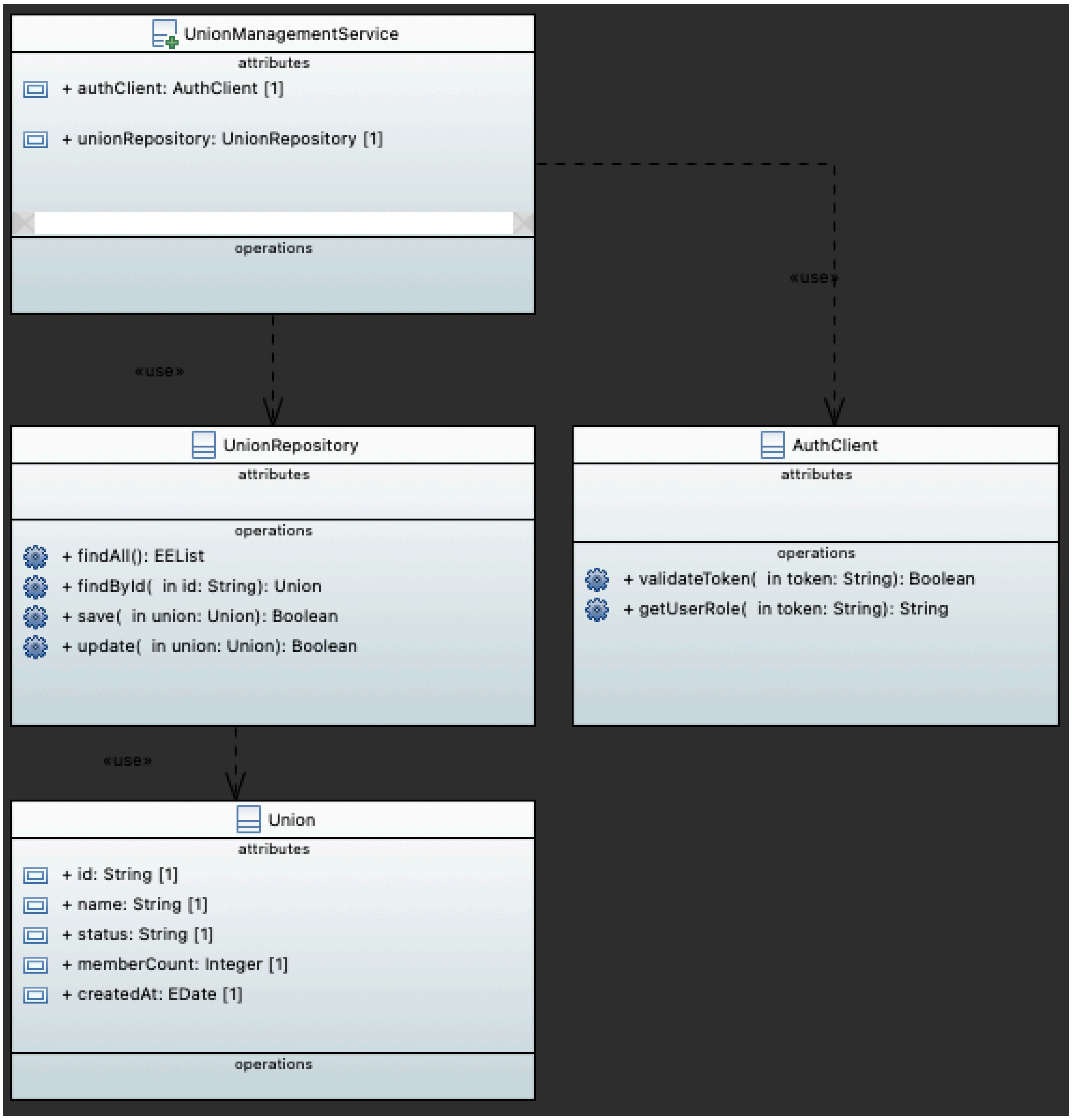
Name	UnionManagement	
Provider	UnionManagementService	
Operation	Signature	AcceptUnion
	Function	Function to accept applied union
Operation	Signature	RejectUnion
	Function	Function to reject applied union

Name	UnionActivity	
Provider	ActivityService	
Operation	Signature	viewActivity
	Function	Function to view and check union activity

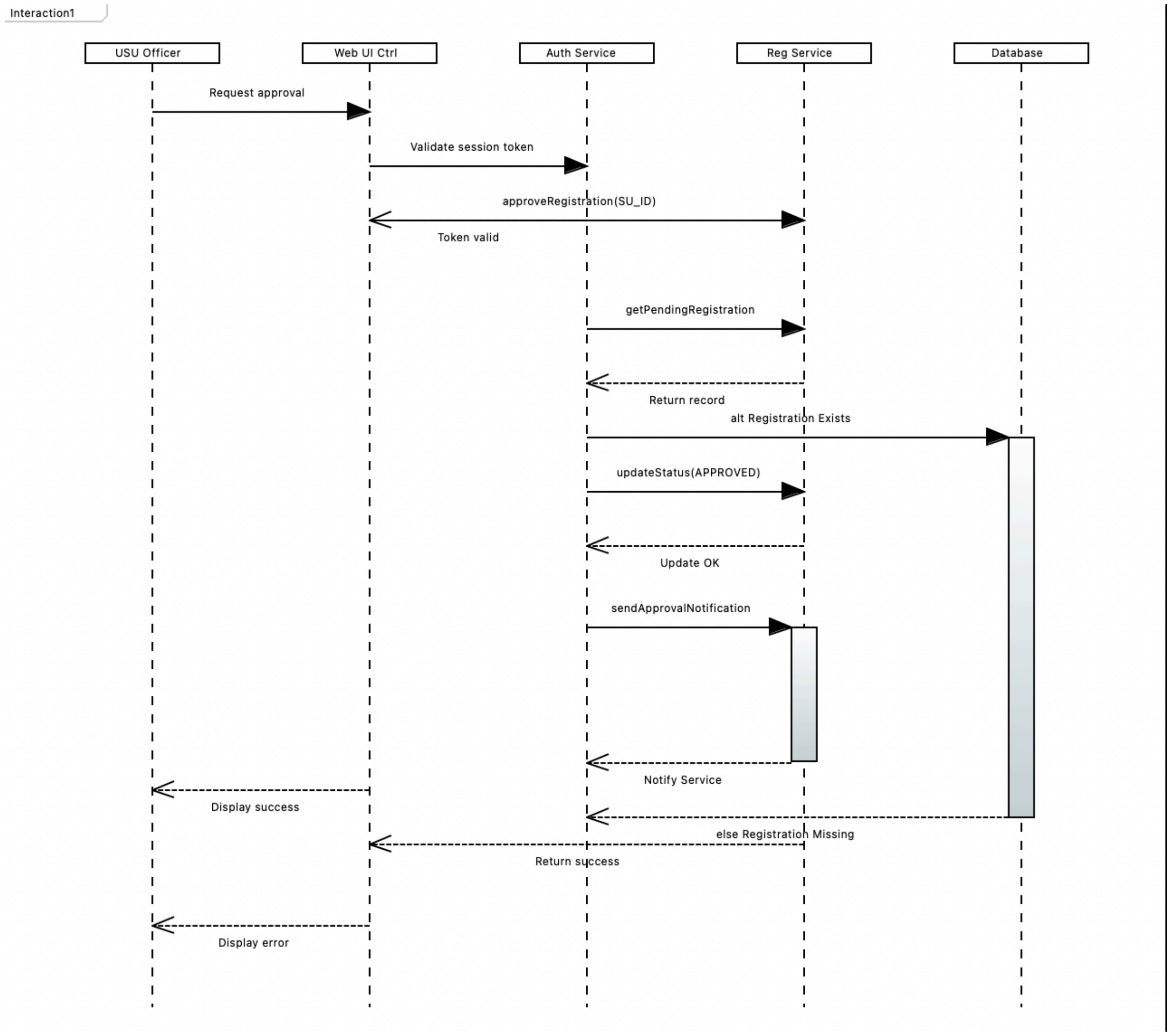
Component Diagram



ClassDiagram


UnionManagementService
attributes
+ authClient: AuthClient [1]
+ unionRepository: UnionRepository [1]
operations
«use»
«use»
UnionRepository
attributes
operations
+ findAll(): EEList
+ findById(in id: String): Union
+ save(in union: Union): Boolean
+ update(in union: Union): Boolean
AuthClient
attributes
operations
+ validateToken(in token: String): Boolean
+ getUserRole(in token: String): String
«use»
Union
attributes
+ id: String [1]
+ name: String [1]
+ status: String [1]
+ memberCount: Integer [1]
+ createdAt: EDate [1]
operations

SequenceDiagram


Interaction1
USU Officer
Web UI Ctrl
Auth Service
Reg Service
Database
Request approval
Validate session token
approveRegistration(SU_ID)
Token valid
getPendingRegistration
Return record
alt Registration Exists
updateStatus(APPROVED)
Update OK
sendApprovalNotification
Notify Service
Display success
else Registration Missing
Return success
Display error