STUDENT ACTIVITY POINTS MANAGEMENT

Software Requirements Specification

GROUP - 9

Project lead: Veda Vijayshankar

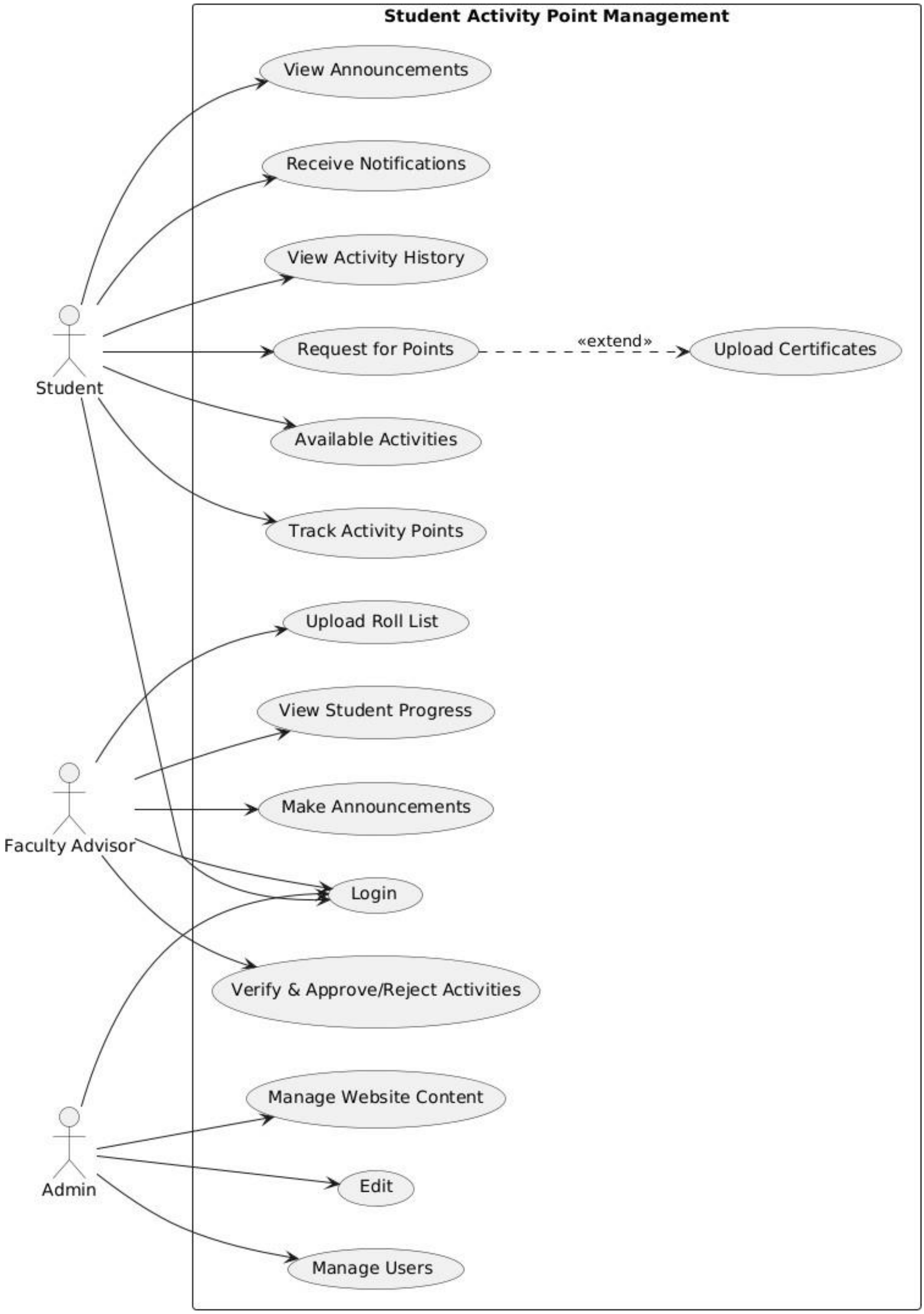
Group Members	Roll Number
Keerthana Lekshminarayanan	B220038CS
Panguluri Tharun Chowdary	B221097CS
Veda Vijayshankar	B220584CS

Revision History

Version	Date	Description of Changes
1.0	02-03-2025	FINAL SRS DOCUMENT

CONTENTS

USE CASE DIAGRAM



FUNCTIONAL REQUIREMENTS

2.1 USER REQUIREMENTS

F1: The system must support role-based access, allowing students and faculty advisors to log in via Google Authentication.

F2: Students should be able to request activity points by uploading certificates as proof of completed activities.

- Students must be able to enter activity details, including name, type, date, and organizer to request activity points.
- Duplicate activity submissions should be prevented.

F3: The system must display a student's activity history, showing all previously submitted activities along with their status (Pending, Approved, or Rejected).

F4: Automated notifications must be sent to students regarding approvals, rejections and new announcements.

F5: Students should be able to view announcements posted by faculty advisors.

F6: Students must first seek approval from the concerning faculty member before their activity point request is finalized by the faculty advisor.

F7: A list of available activities that students can participate in order to earn points must be accessible.

F8: Faculty advisors should be able to verify, approve, or reject student activity submissions.

F9: They must be able to track student progress, including earned points, pending activities, and history.

F10: Faculty advisors should be allowed to publish announcements related to activities, deadlines, or updates.

- FA must provide title and body of announcements.
- Both FA and student can view announcements.

F11: Automated notifications must be sent to faculty advisors regarding new activity submissions or pending approvals.

F12: Admins must have exclusive access to manage user accounts of students and faculty advisors.

F13: Admin must be able to add or remove students as well as faculty advisors from the database when they join or leave the institution.

F14: The admin should have the authority to create, update, and modify platform guidelines.

F15: The admin should be responsible for uploading and managing the roll list for faculty advisors.

2.2 SYSTEM REQUIREMENTS

F16: A secure authentication mechanism must be in place for student, faculty, and admin logins.

F17: Reports on student progress and activity participation should be generated by the system.

F18: The system should validate activity submissions and allow students to upload proof in PDF format.

F19: Student records and uploaded certificates must be stored securely in the database.

F20: The platform must display a list of available activities that students can participate in which can be modified, added or deleted by Admins.

F21: A categorized list of upcoming activities (mandatory or optional) should be displayed.
F22: The system should display total activity points along with a category-wise breakdown of each student.
F23: Student details, including Name, Roll Number, and Department, should be maintained in the system.
F24: Faculty information, such as Name, Email, and Department, must be stored securely.

NON FUNCTIONAL REQUIREMENTS

3.1 PERFORMANCE REQUIREMENTS

- The system should handle at least 50 concurrent users smoothly.
- The dashboard should load within 3-5 seconds under normal conditions. Overall, the performance of the system will be fast and accurate.

3.2 SAFETY AND SECURITY REQUIREMENTS

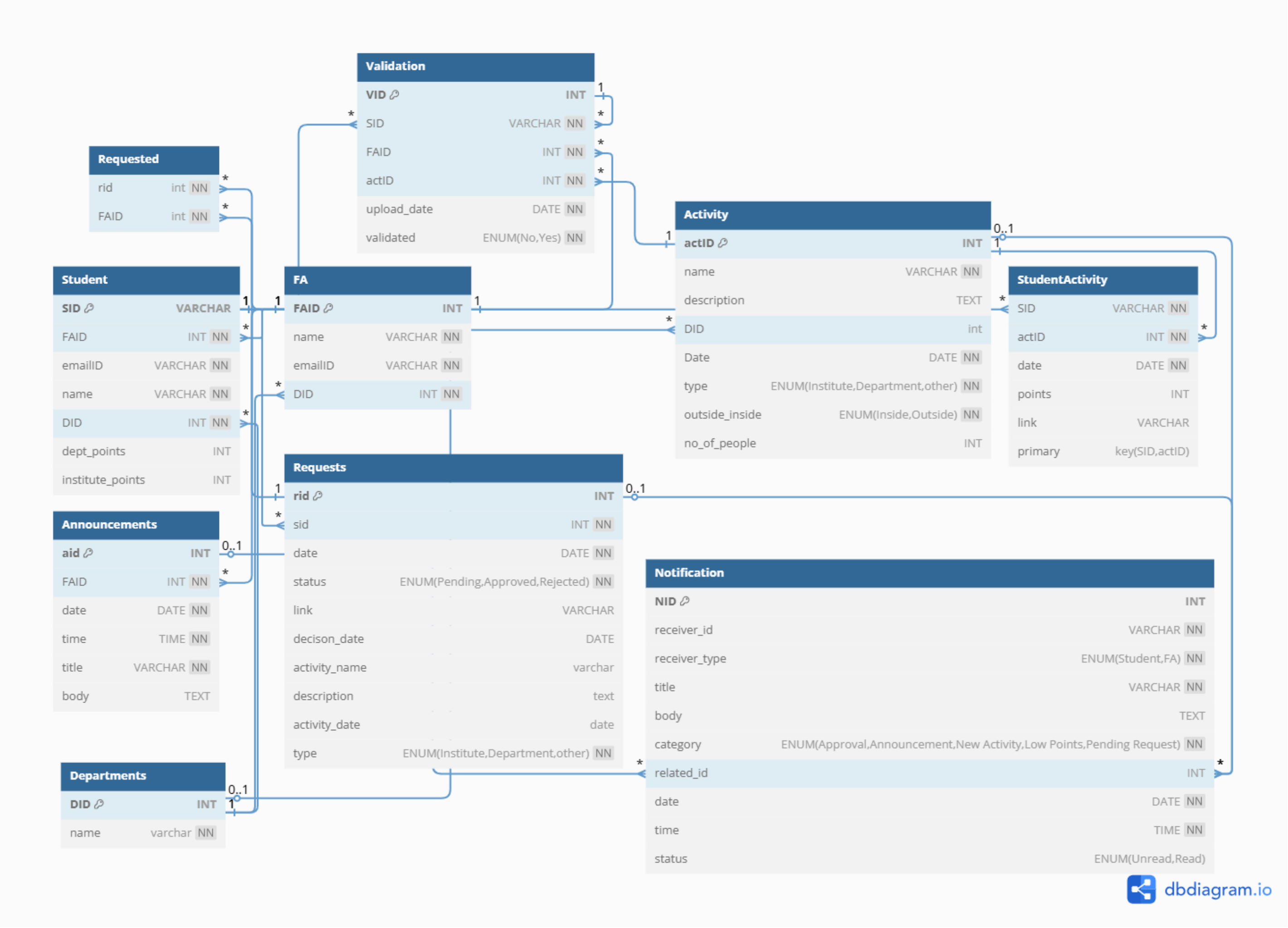
1. User Identity Authentication: The system should require authentication using Spring Security with unique usernames and passwords for each user, preventing unauthorized access to sensitive data.
2. Data Privacy: Data should be accessible only to authorized users.
3. Data Encryption: All sensitive data should be encrypted. For eg:- using BCrypt for password hashing.

- These security measures ensure that the system meets industry standards for privacy, data protection, and secure access.

3.3 SOFTWARE QUALITY ATTRIBUTES

1. Reliability
   - The system will not lag and will provide instant and accurate results to all users.
2. Adaptability
   - The system is initially designed to support up to 200 users but should allow future upgrades to accommodate growth.
3. Maintainability
   - The backend code should follow a modular architecture, with separate controllers, services, and repositories for better maintainability.
   - Proper comments and documentation should be included to facilitate future modifications and debugging.
4. Portability
   - The system can be deployed on any machine, ensuring flexibility across different environments.
5. Compliance
   - The system should adhere to basic data privacy regulations, ensuring that sensitive data is not publicly exposed.
6. Usability
   - The UI should be fully responsive, ensuring optimal usability across laptops, tablets, and mobile devices.
   - Basic help documentation should be provided to assist users in navigating the system effectively.

DATABASE DESIGN



Validation		
VID	INT	
SID	VARCHAR	NN
FAID	INT	NN
actID	INT	NN
upload_date	DATE	NN
validated	ENUM(No,Yes)	NN

Requested		
rid	int	NN
FAID	int	NN

Student		
SID	VARCHAR	
FAID	INT	NN
emailID	VARCHAR	NN
name	VARCHAR	NN
DID	INT	NN
dept_points	INT	
institute_points	INT	

FA		
FAID	INT	
name	VARCHAR	NN
emailID	VARCHAR	NN
DID	INT	NN

Activity		
actID	INT	
name	VARCHAR	NN
description	TEXT	
DID	int	
Date	DATE	NN
type	ENUM(Institute,Department,other)	NN
outside_inside	ENUM(Inside,Outside)	NN
no_of_people	INT	

StudentActivity		
SID	VARCHAR	NN
actID	INT	NN
date	DATE	NN
points	INT	
link	VARCHAR	
primary	key(SID,actID)	

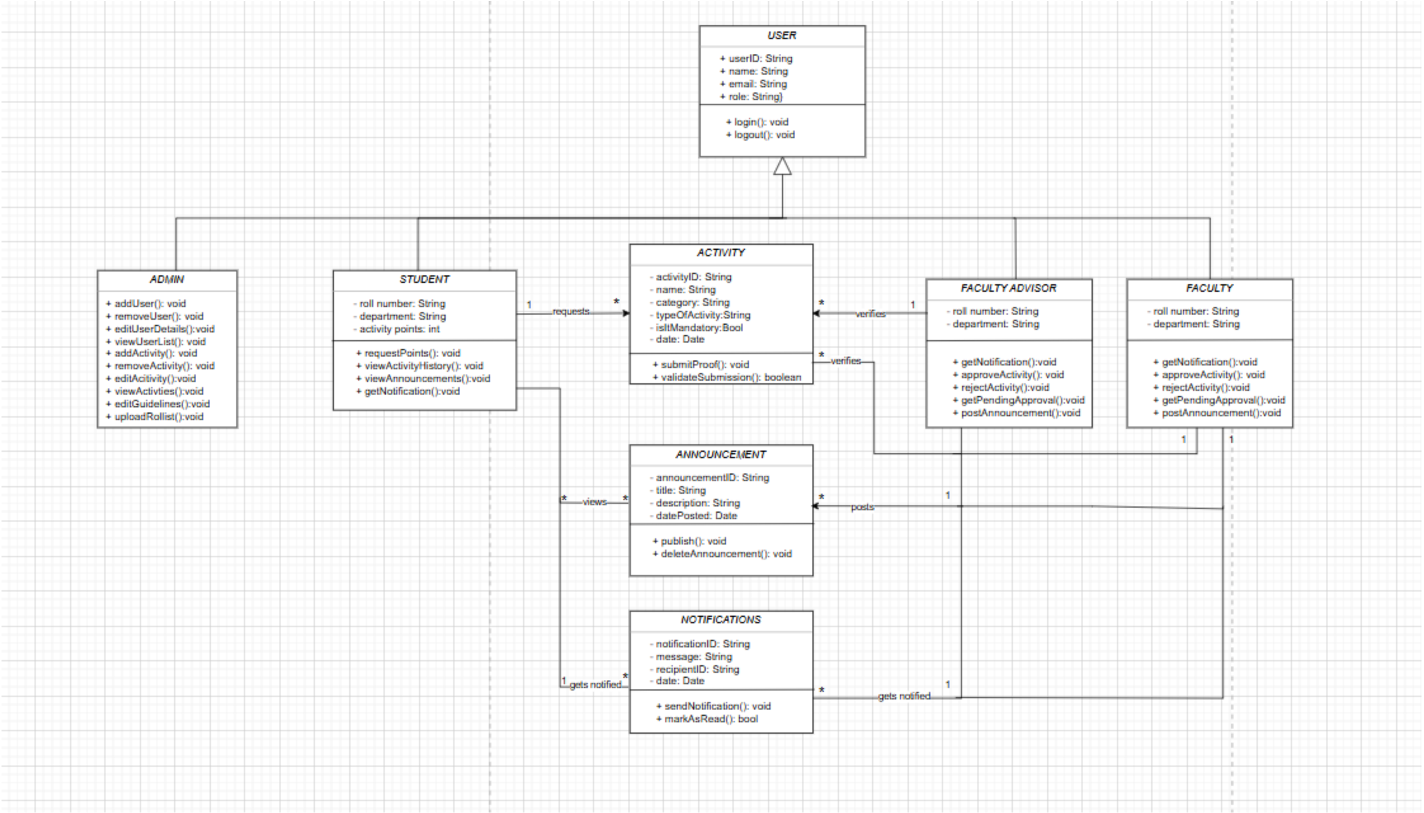
Requests		
rid	INT	
sid	INT	NN
date	DATE	NN
status	ENUM(Pending,Approved,Rejected)	NN
link	VARCHAR	
decison_date	DATE	
activity_name	varchar	
description	text	
activity_date	date	
type	ENUM(Institute,Department,other)	NN

Announcements		
aid	INT	
FAID	INT	NN
date	DATE	NN
time	TIME	NN
title	VARCHAR	NN
body	TEXT	

Departments		
DID	INT	
name	varchar	NN

Notification		
NID	INT	
receiver_id	VARCHAR	NN
receiver_type	ENUM(Student,FA)	NN
title	VARCHAR	NN
body	TEXT	
category	ENUM(Approval,Announcement,New Activity,Low Points,Pending Request)	NN
related_id	INT	
date	DATE	NN
time	TIME	NN
status	ENUM(Unread,Read)	

CLASS DIAGRAM


USER
+ userID: String
+ name: String
+ email: String
+ role: String)
+ login(): void
+ logout(): void
ADMIN
+ addUser(): void
+ removeUser(): void
+ editUserDetails():void
+ viewUserList(): void
+ addActivity(): void
+ removeActivity(): void
+ editAcitivity():void
+ viewActivties():void
+ editGuidelines():void
+ uploadRollist():void
STUDENT
- roll number: String
- department: String
- activity points: int
+ requestPoints(): void
+ viewActivityHistory(): void
+ viewAnnouncements():void
+ getNotification():void
1
requests
*
ACTIVITY
- activityID: String
- name: String
- category: String
- typeOfActivity:String
- isItMandatory:Bool
- date: Date
+ submitProof(): void
+ validateSubmission(): boolean
*
verifies
1
*
verifies
FACULTY ADVISOR
- roll number: String
- department: String
+ getNotification():void
+ approveActivity(): void
+ rejectActivity():void
+ getPendingApproval():void
+ postAnnouncement():void
FACULTY
- roll number: String
- department: String
+ getNotification():void
+ approveActivity(): void
+ rejectActivity():void
+ getPendingApproval():void
+ postAnnouncement():void
1
1
*
views
*
ANNOUNCEMENT
- announcementID: String
- title: String
- description: String
- datePosted: Date
+ publish(): void
+ deleteAnnouncement(): void
*
posts
1
NOTIFICATIONS
- notificationID: String
- message: String
- recipientID: String
- date: Date
+ sendNotification(): void
+ markAsRead(): bool
1
gets notified
*
*
gets notified
1