# Software Requirements Specification

for

# Academics Management Software

**Version 1.0**

**Prepared by**


**Group #17:** **Group Name: *same_storm***

| | | |
|---|---|---|
| **Achint Agrawal** | **180028** | achintag@iitk.ac.in |
| **Somya Lohani** | **190848** | somyaloh@iitk.ac.in |
| **Sachin** | **180639** | thakan@iitk.ac.in |
| **Udhav Gupta** | **180829** | udhavgup@iitk.ac.in |
| **Yash Gupta** | **190998** | gyash@iitk.ac.in |
| **Himanshu Sood** | **190381** | himsood@iitk.ac.in |
| **Piyush Agarwal** | **190600** | piyuagr@iitk.ac.in |
| **Aditya Ranjan** | **190068** | aranjan@iitk.ac.in |
| **Yash Gupta** | **190997** | yashbg@iitk.ac.in |
| **Utkarsh Jain** | **190928** | utkjain@iitk.ac.in |


**Course:** **CS253**

**Mentor TA:** ***Ms. Shatroopa Saxena***

**Date:** **30 Jan 2022**

# CONTENT

# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 1.0 | Entire Team of Developers | Starting from a template document, details of all the sections have been added. | 01/02/22 |

# 1 Introduction

## 1.1 Product Scope

In the duration of the semester, the UG students face a lot of challenges in terms of managing their academic courses, even more so in online semesters. There can be various notes, assignment/report deadlines, quizzes, projects etc. which students need to keep a track of for each course. Besides just managing current courses, it is common for them to plan their future semesters. This requires them to chart their eight semesters out and get realistic course feed-backs during the time of registrations. All of these processes can get complicated and may lead to critical mistakes at the student's end. Furthermore, managing these things takes time which can be used for something more productive by the students.

Our project aims to provide a platform exclusively for IIT Kanpur students to manage everything related to their academics in a single place. Complete with a calendar scheduling section, a course planner, a note making feature and a course feedback section, our product offers students the opportunity to plan their academic curriculum wisely. They can hereby use their time judiciously and reduce their stress levels to a great extent.

## 1.2 Intended Audience and Document Overview

Our software requirements specification document is intended for:

1) The developers, which is essentially our team of 10 students, who will refer to this document as a guideline for the development process over the course of the project.
2) The TA who has been assigned to us, who will be our project manager and will overlook the development process.
3) The marketing team, which will be our team of 10 and eventually the IITK administration which will be promoting the use of the software for its students. The document will be used for extracting important attributes and benefits of the software when needed.
4) The students and teaching staff of IITK who will be the end-users of our product and might need the document for referring to its features in future.
5) The testers, which will be the developers initially, followed by TAs and the course instructor. The document gives the specifications of the end product and hence will be crucial to the testing process.
6) Documentation writers, which are the developers themselves, who will use the documentation to review the software development from time to time.
7) The maintenance team, which will be the administration of IITK, which will use this document to navigate their way through the software to be able to maintain it.

The SRS has been structured into the following five sections to increase it's interpretability:

- Introduction
  As the name suggests, the introduction will be giving the reader an overview of the product description, it's objectives, it's goals and who it will cater to. It will equip the reader with

prerequisites for understanding the document in terms of providing them with the convention followed and a glossary of definitions and abbreviations used in the document. The section concludes by giving relevant references and acknowledgements, which would help walk the reader through our thought process.

- Overall Description
  The overall description gives the user a deeper insight into the product by elaborating on the system hierarchy and the relationship between the different subsystems. It goes on to explain the origin of the product and how it has evolved into its present form. It also explains the functionalities of the product and the different challenges and dependencies for the software. These include the constraints which limit the scope of the product and the external factors which might affect the operation of the software.
- Specific Requirements
  This section essentially dwells on the functional requirements of the product. It begins by outlining the user, the hardware and the software interfaces. The requirements are made more interpretable by means of UI views, use case models and These describe the interactions of the software with the
    - user (in terms of graphical views of the interface prefaced with brief description)
    - hardware devices (in terms of the different supported devices and the working of their interactions with the software)
    - other softwares on the system (in terms of the other different softwares on our system that our software will communicate with)
- Other Non-functional Requirements
  As is suggested by the name, this section will deal with emergent system properties such as:
    - Performance requirements (in terms of the expected performance from the different product functionalities under various circumstances)
    - Safety and security requirements (in terms of description of circumstances under which the software security can be breached and how it can be prevented by means of safety certifications and authentication requirements)
    - Software quality attributes (in terms of the distinctive quality characteristics such as availability, customizability etc)
- Other Requirements
  This section is the one that has evolved the most over the course of the development since it consists of the requirements which haven't been added in the above sections such as database requirements, reuse objectives etc.
  Intended sections for the different readers are:
    - For the developers, it is essential that all the five sections be gone through in the provided order so that they understand the core ideas of the product and the development is in strict accordance with both the functional and non-functional requirements.
    - Our project managers should go through all the sections to be able to understand the product along with its attributes and challenges. This would allow them to understand the project at a deeper level and assist the team.
    - The marketing team should go through the sections of Overall description and Other Non-functional requirements (specifically the Software quality attributes section)

since they need to have an overall idea of the product and its key functionalities with special emphasis on how it performs better than the other goods in the market.
- The users of the product should go through the sections of Overall description and Specific requirements to be able to understand the product and it's features.
- The testers should go through the sections of sections of specific requirements and Other non-functional requirements to understand the parameters on which the document has to be tested.
- The maintenance team should go through the sections of Specific requirements, Other non-functional requirements and Other requirements to be able to understand the dependencies and requirements of the system and maintain it.

## 1.3 Definitions, Acronyms and Abbreviations

CC: Computer Centre
SRS: Software Specification Requirement
UI: User Interface

Course types:
DC: Department Compulsory
DE: Department Elective
IC: Institute Compulsory
ESO: Engineering Science Options
OE: Open Electives

## 1.4 Document Conventions

Formatting Conventions: This document follows the IEEE formatting requirements

## 1.5 References and Acknowledgments

SRS Template: Course Instructor, CS253
UI style guide:
- pingala - UI for Course Planner
- google calendar - UI for calendar
- Mookit - UI for course feedback forum
- Leetcode - UI for course feedback forum

Use case Diagrams: https://lucid.co
UI creation: http://www.mockflow.com

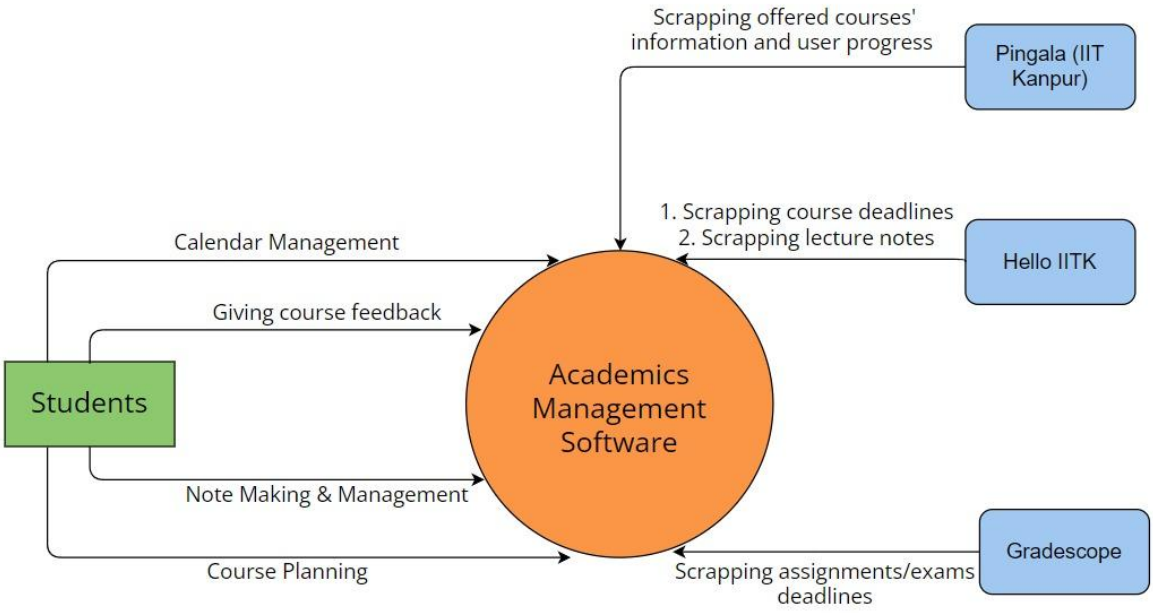
Ensuring availability (software quality attribute):
https://www.infoworld.com/article/3629416/5-steps-to-improve-your-application-availability.html

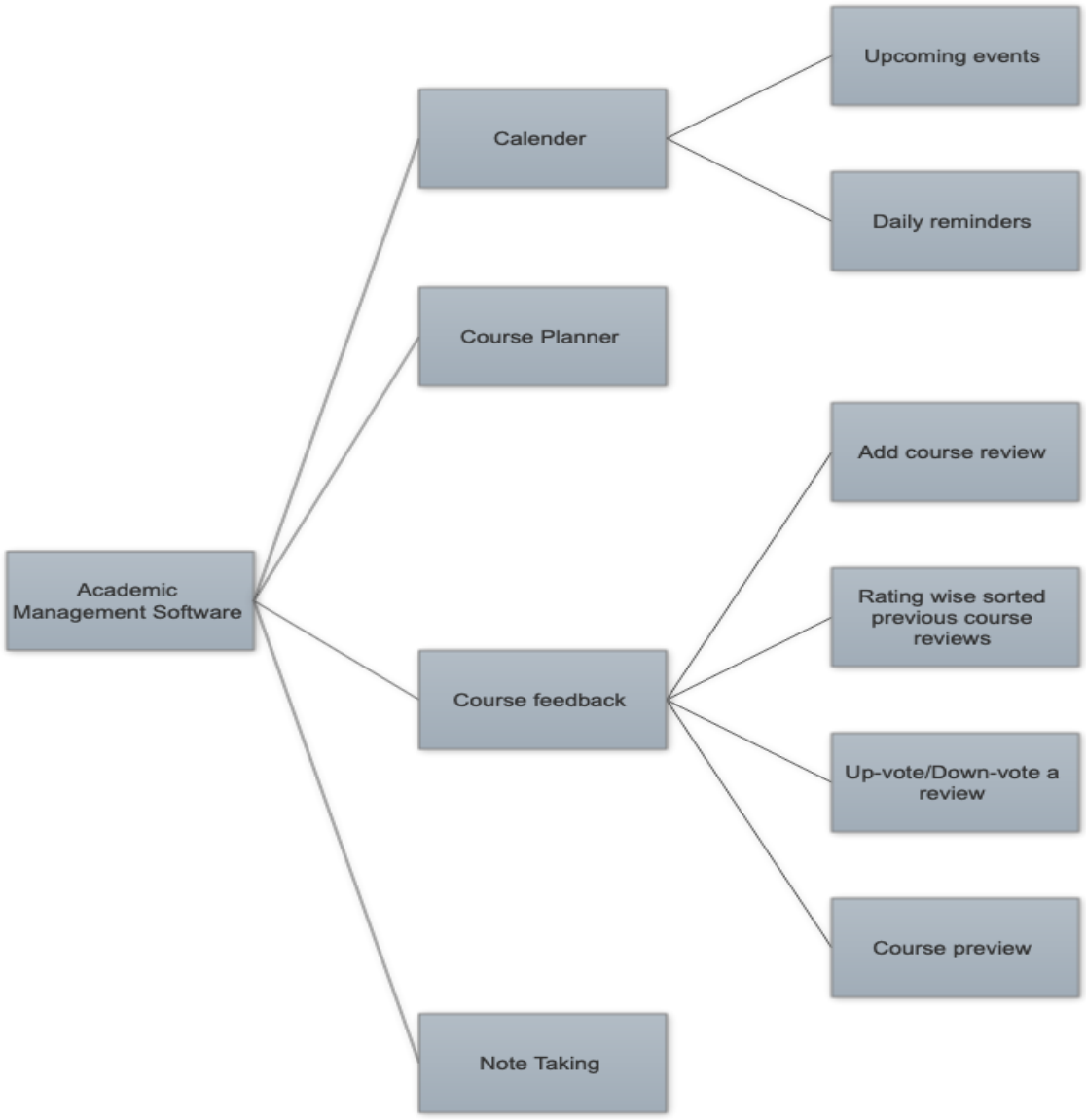
# 2 Overall Description

## 2.1 Product Overview

– The origin of the idea of our project ‘Academic Management Software’, is from a blend of functions of Google calendar, Pingala, and Mookit. We are aiming to include the most useful features from a student and academic viewpoint in our software.
Our project isn’t a follow-on or component of an existing project but a derivation and implementation of features from multiple existing softwares. All the functions and features of our software will work independently from any other software, hence it’ll be a self-contained product.

### System Context Diagram






## 2.2 Product Functionality

The app will contain the following main functionalities

1. **A calendar-like section** that can store all class schedules, quiz schedules, assignment (and other) deadlines, etc. The user can add a description for each of these “events” (for example, if the semester is online, they can add a meeting link). This will be customisable and at the same time will have an automated component.
2. **A course planner** can help the user plan their future semesters. The user can add multiple course plans (as there can be multiple future possibilities); in each plan, they can add

courses semester-wise. Appropriate warnings can be automatically given (credits not in range, prerequisite not complete, etc.).

3. **Note-taking feature**: the user can take quick digital notes for any course, which are saved for that course (for example, they can take notes during lectures and edit them later).
4. **Course feedback**: users who have done a course can give their ratings/feedback for that course, and other users can see feedback to make a possibly better decision about choosing courses (currently, one has to ask around to get feedback).

## 2.3 Design and Implementation Constraints

– Challenges in interacting with other application

- students' academic data need to be accessed frequently for the features like course planner to work. There are following two options to get this data
    - Scrape data from existing platforms like Pingala and Mookit
        - This alternative may face issues due to limitations of tools(java, c++) available. A better option like python might be easier to work with for its implementation
    - Allow students to add their own course data and maintain it in our application database itself.

– Challenges in accessing the essential data for app to work

App will assume following data to be available for proper functioning of crucial features:

- course schedule for academic calendar
- deadline date for important events like assignments(for upcoming events or reminders)
- details of courses completed by a student(for course planner)
- course content details for course preview feature(for course preview feature)
- students registered in any given course in any given semester(for validating a course review)

– Security Constraints:

- One student should not be able to access the courses and academic information of other students.
- CC credentials must be kept secure.

## 2.4 Assumptions and Dependencies

- The list of courses being offered in the current semester are known and students can only pick courses from this list.
- If we scrape data from Pingala, then any changes made to pingala might affect the functionality of our application.
- One of the challenges in the course review feature will be to ensure the reliability of the course reviews. We are assuming that the student giving the review has already completed the course and these course reviews are honest and nobody will manipulate this feature to serve their own purpose.

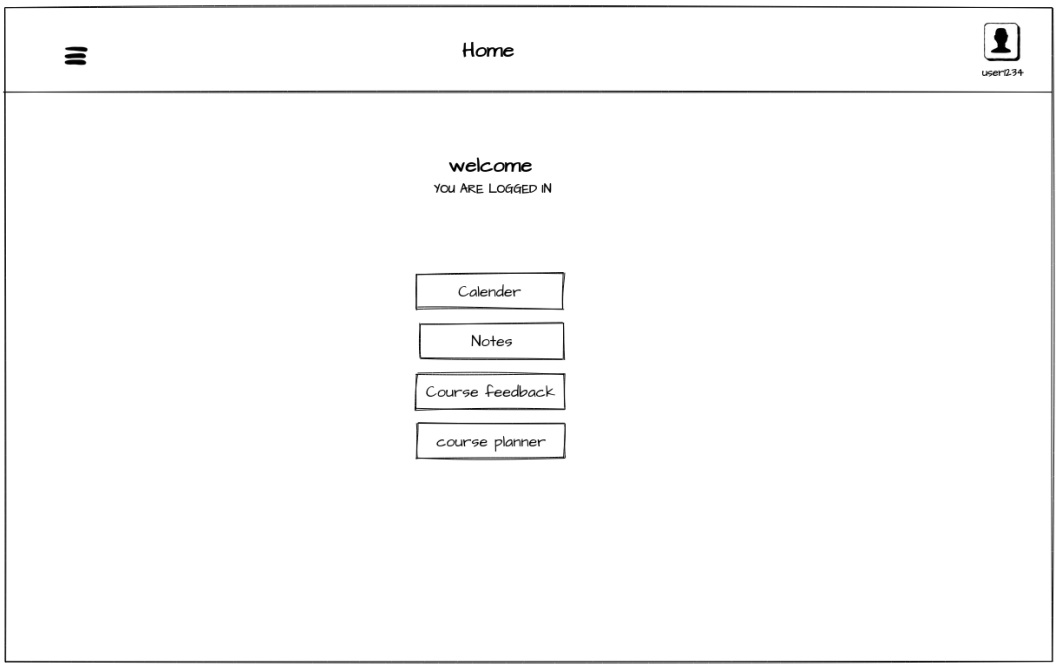
# 3 Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

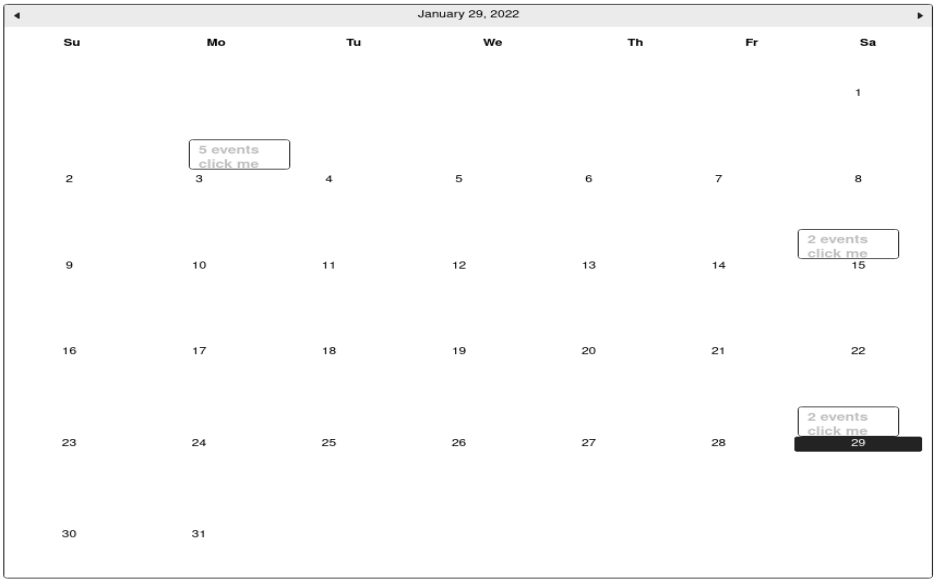
Main components that will feature in each of the sections:

- Application Home



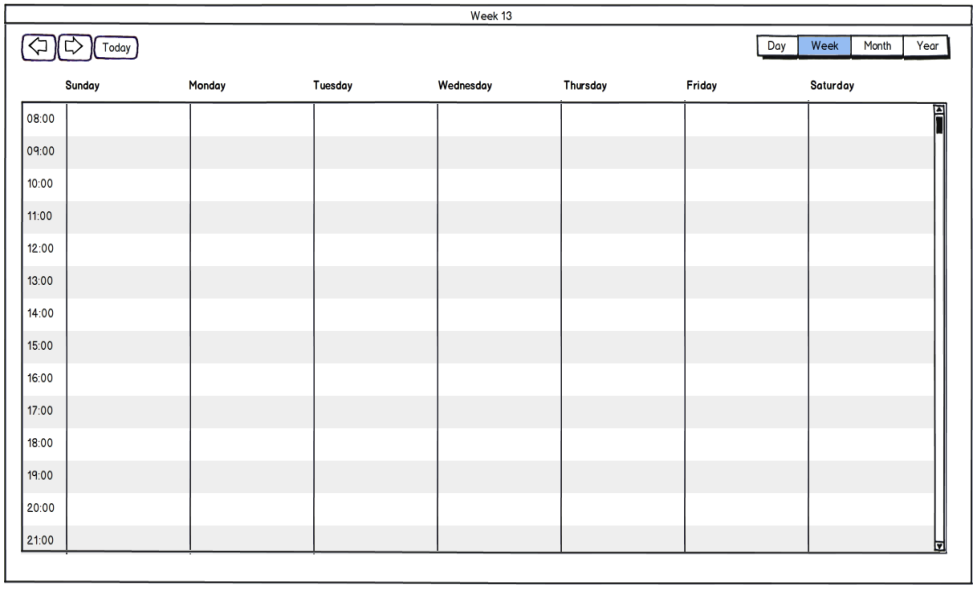

**Description:** After successfully logging in, users can navigate to any of four services from the homepage itself.

- Calendar
    - Views of the calendar
        - Monthly




**Description:** click me event-button will open a daily view or a list of events and their details like timing, subject, etc.

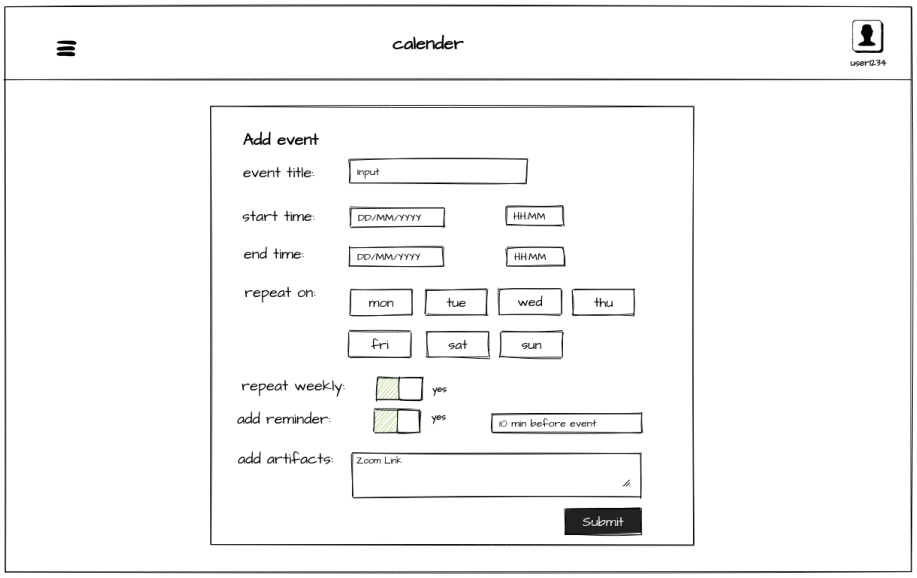
        - Weekly




**Descriptions**:

- user will be given an option to switch between weekly view and monthly view as toggle switch demonstrate in sketch above
- Monthly view will contain the counts of events on each day and a button to show details of these events on click

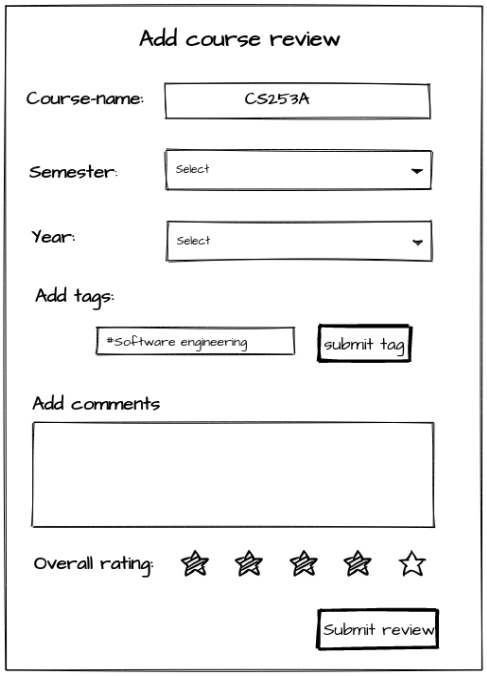
- Add an event manually




**Description:** Apart from scrapping events from hello.iitk, we also let users manually add events using the given UI. We can also add artifacts related to that event like zoom link, description, location etc. We can also add reminders.

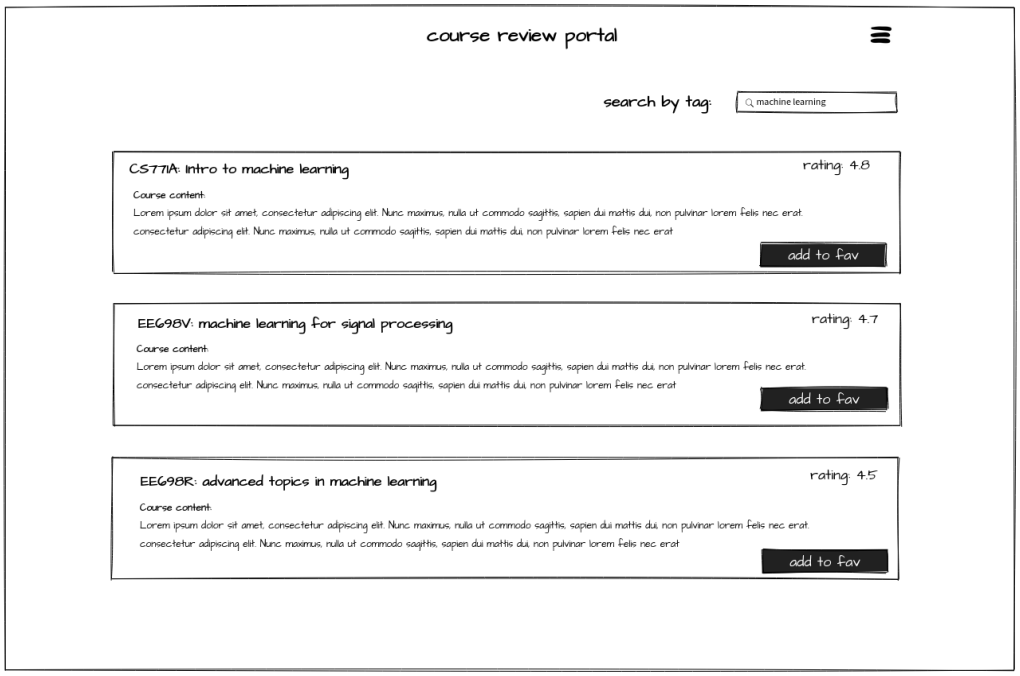
- Course Feedback

  - Add course review(also anonymous)




**Description:** Users will be presented above form in order to add a new course review or edit an existing course review.

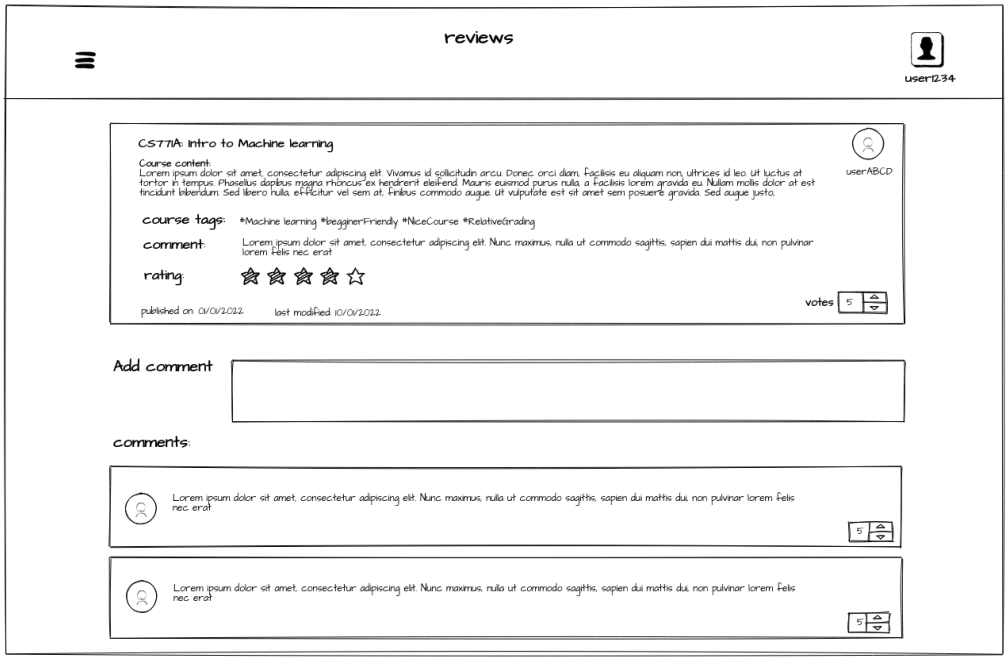
- Search for courses




**Description:**

- The search page displays all courses relevant to the provided tag. It displays a preview of the courses.
- On clicking this preview user will be directed to a list of reviews associated with the given course. These reviews will themselves be in preview form.

- Detailed course review



**Description:**

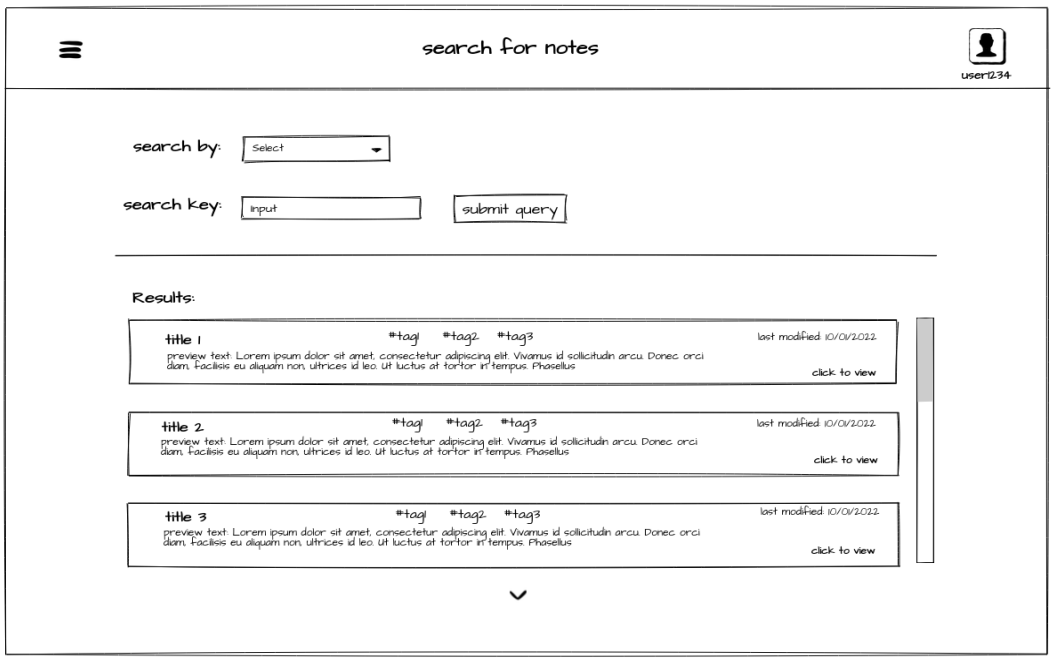
- user can peek into details of a review
  - details of the course review contains course description
  - also tags associated with the course
  - tags mentioned will be links which on clicking may direct user to course search page and search for that particular tag by its own
- user can upvote/downvote a particular review
- user can provide comments regarding a given review and also provide his/her reaction on other's comments

- Note-taking
  - Add new note


**Description:**

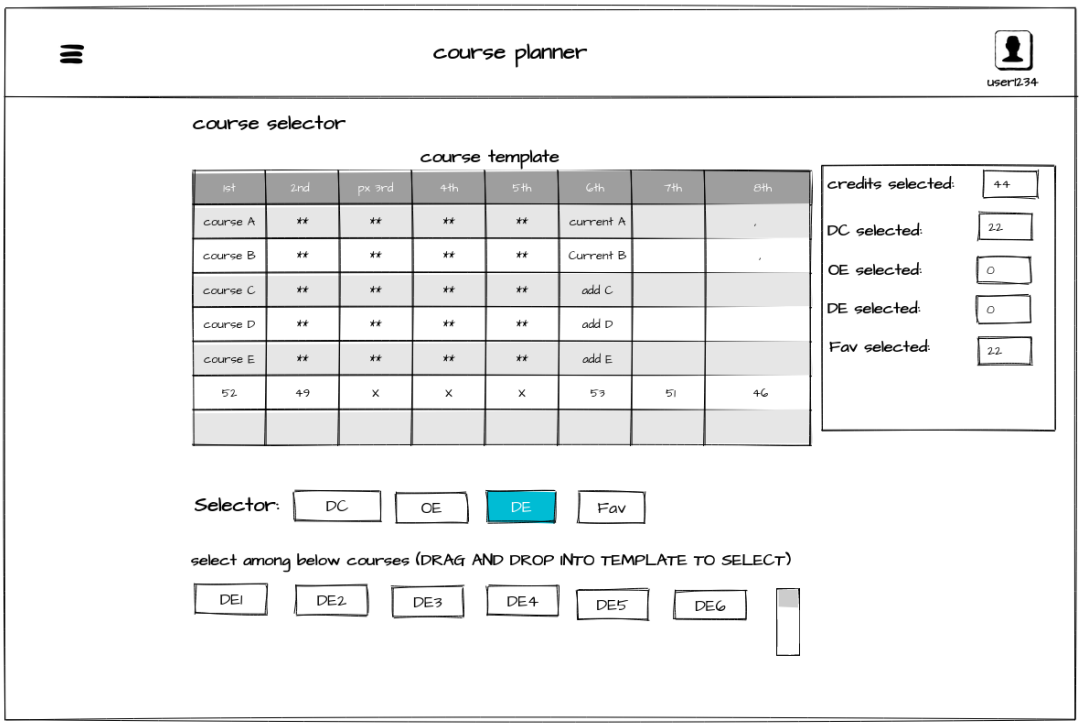
- The main aim of the quick notes feature is to let users add and manage their notes with minimal effort. That is why following details are crucial:
- user will always be provided with a list of tags to choose from, to tag their documents and make their retrieval easy in future
- added notes can be edited multiple times
- user will only need to add document title and main text in the notes
- timestamp of last changes will be added automatically to the document

- List all notes associated with some course/general tag




**Description:** Search feature gives users options to filter notes by their titles or by tags associated with the documents.

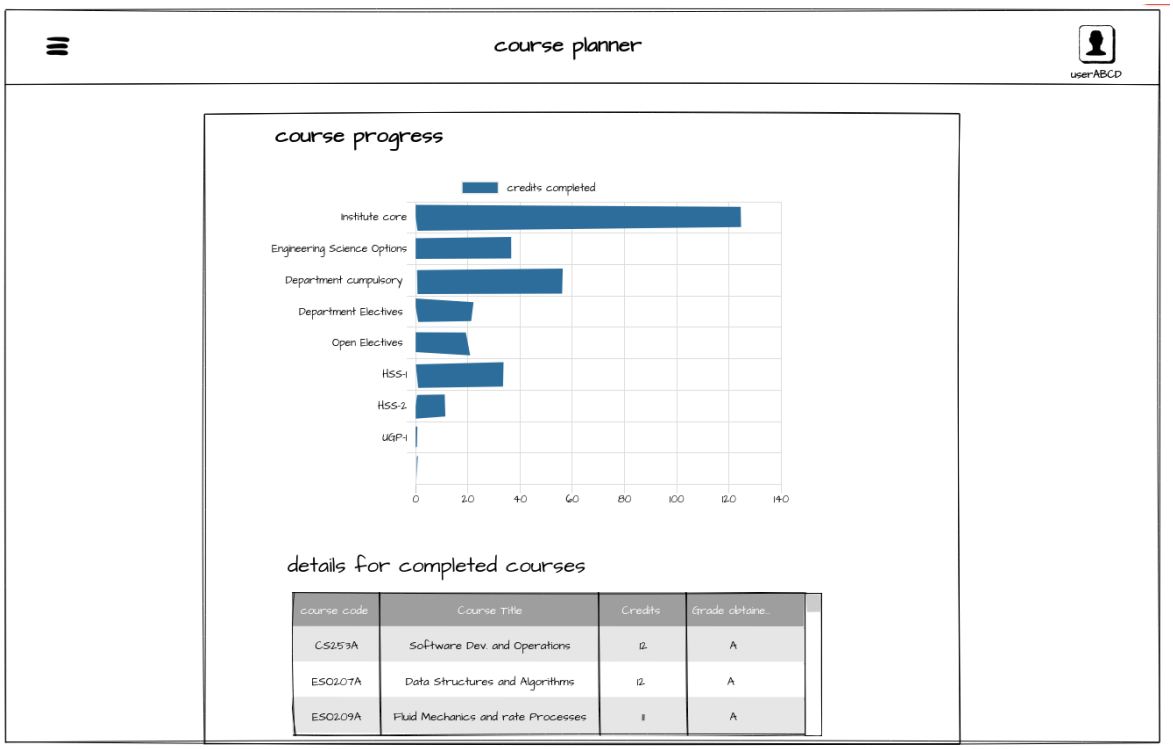
- **Course planner:**
  - Course selector



**Description:**

- This feature will provide the user an interactive interface to plan his/her current semester. Users can toggle between multiple categories of courses and related courses will be fetched to display. These displayed courses can be dragged and dropped into templates.
- As a user changes its template adding some course, backend logic will tell the user feasibility of that course being done(satisfying prereq., credits under limit etc.)
- One section will show the user his/her selected courses and their categories.

- **Show course progress**




**Description:**

This section shows the student's course progress. It is a kind of dashboard that will show credits completed so far, courses completed and their details(credit count, grades obtained etc.) and other information that the user might need in order to plan his/her courses.

### 3.1.2 Hardware Interfaces

- Physical devices like mobiles, tablets, laptops, desktops, etc. will be needed to run the application on the web.

### 3.1.3 Software Interfaces

- We will need an API for fetching the available course information from Pingala and then make it available on the screen. One other API will be needed to fetch user-specfic information like the completed courses
- We will need APIs for scrapping deadlines like assignment submission deadlines, quizzes, etc. from mooKIT, gradescope

## 3.2 Functional Requirements

### 3.2.1 F1: Calendar Management

The application must enable the user to create, modify and delete events that depict his/her engagement. This functionality should mimic Google Calendar along with allowing the user to drag and drop various available courses into it, which are needed to be manually added in Google Calendar.
In the scheduled events, the user must have the freedom to add tags, meeting links, description and other related information. The user can choose the courses that he wants to appear in the calendar. With this, discussion hours/tutorial/labs will be automatically added to the calendar in the form of event cards. The user can also choose to import the quizzes/exams and assignments deadlines from mooKIT, Gradescope, etc. Since these events may not be present on mooKIT in an organized way, the user will still have the freedom to add the deadlines by himself/herself.

### 3.2.2 F2: Course Planner

It is very important for a student to play with the template carefully and wisely so as to get the maximum benefit from the IITK academics. For this, a student needs to consider a lot of criteria before taking up a course. This includes pre-requisites, clashes with DCs or other important courses, credit limits, course feedback, professor feedback, grading etc. Students then have to make use of various platforms and contact various seniors to know about all these things.
Course Planner functionality of this application will try to bring everything together. A student can plan the forthcoming semesters at any time (*but the available courses may differ). In the course planner, the student will get the DCs for that semester already added in the calendar (*he may choose to drop some or all of them). The available courses list will be scrapped from a public database in which the courses with no pre-requisites or the pre-requisites of whom are already done by him/her will be highlighted (*prerequisites keep changing, these will be the one available from the previous year list). The student can choose the courses from this list which will then appear temporarily in the week-view calendar (similar to pingala) where he/she can avoid clashes.
The user can more than one plan for the upcoming semesters without destroying the other plans (this is similar to creating various versions of semester course plans without destroying the previous ones). One out of these plans is declared as the primary plan. The user can compare these plans to decide on the most suitable one for them.

### 3.2.3 F3: Course Feedback

During the pre-registration and add-drop period, it is important for a student to get valuable feedback from seniors about the course and professor. He/she then needs to approach many other students for this, which takes a lot of time. The course feedback functionality will allow people to rate the courses on various parameters like grading scheme, course content, professor's teaching style. Students can also add their own review for the courses which the other users can upvote or downvote.

The courses can be sorted wrt various parameters to give the most suitable courses for the user. Based on this, the student can then make an informed decision of whether to take up that course or not.

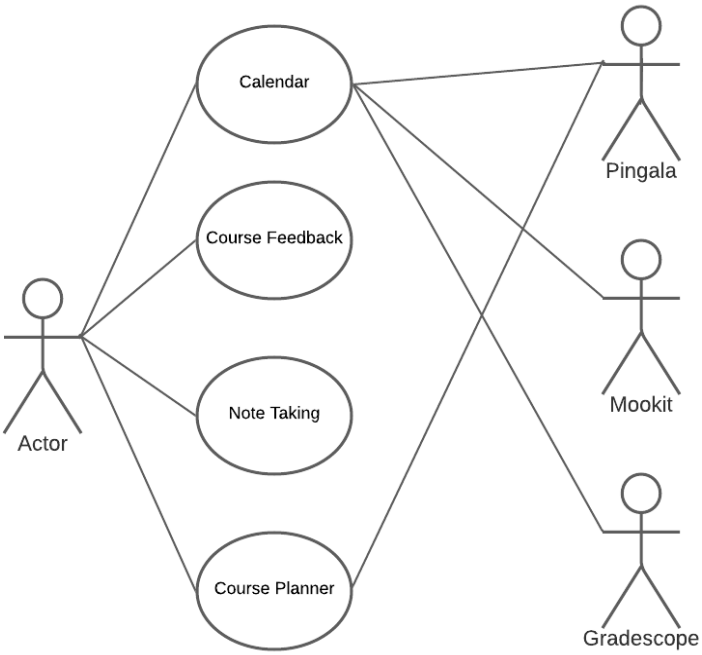
(* Need to decide whether the course review will be anonymous or not)

### 3.2.4 F4: Note-taking Feature

In online lectures, many people like to take note of what the professor teaches. Other software available does provide scribble ability but is not that feasible to be used during classes. In this, the user can take notes in the whiteboard/doc available in the app or can upload the images of the hand-written notes. Each course will have the notes stored in an organized way which then can be accessed when connected to the internet.

## 3.3 Use Case Model

**Overall UML Diagram**



### 3.3.1 Calendar Management- U1

**Author –** Achint Agrawal(180028)

**Purpose** - Students can add important events like class timings, assignment deadlines, etc. to their personal calendar either manually or by importing them from pingala, mookit or gradescope.

**Requirements Traceability –** Calendar Management

**Priority** - High

**Preconditions** - User Authentication

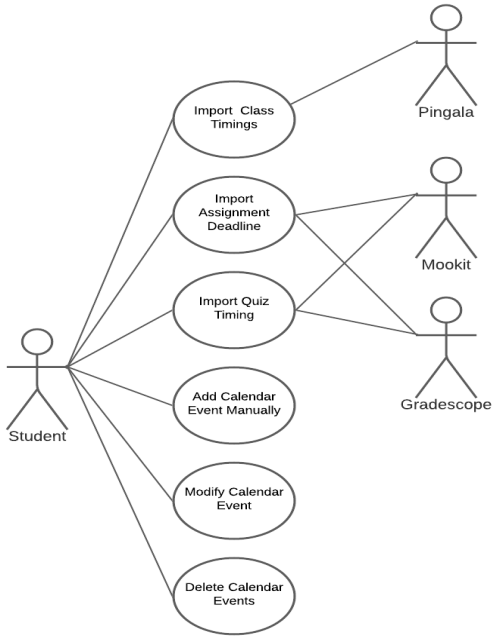
**Post conditions** - Any modifications made by the user must be updated and displayed correctly in the calendar.

**Actors** – Student, Pingala, Gradescope

**Exceptions** - Login failure to other platforms, invalid event details like end time/ begin time.

**Includes** None

**Notes/Issues** - None




**Description –** The student is provided with multiple options to add events to the calendar. Class timings can be imported from pingala.iitk.ac.in for the courses that the student is taking. Assignment deadlines and quiz timings can be imported from hello.iitk.ac.in or gradescope.com depending on the course. This ensures correctness of the calendar and automates the tedious task of adding each event manually. Additionally, students can also add/delete/modify calendar events manually.

### 3.3.2 Course Planner (U2)

**Author –** Achint Agrawal (180028)

**Purpose** - Students would be able to easily find suitable courses to add during add/drop period.

**Requirements Traceability –** Course Planner

**Priority** - High

**Preconditions** - Courses taken by students in previous semesters should be known.

**Post conditions** -

- Total credits being taken in any semester should lie within the minimum and maximum credit requirement.

- Prerequisites of any course being added to a current semester should have been completed in a semester before that.

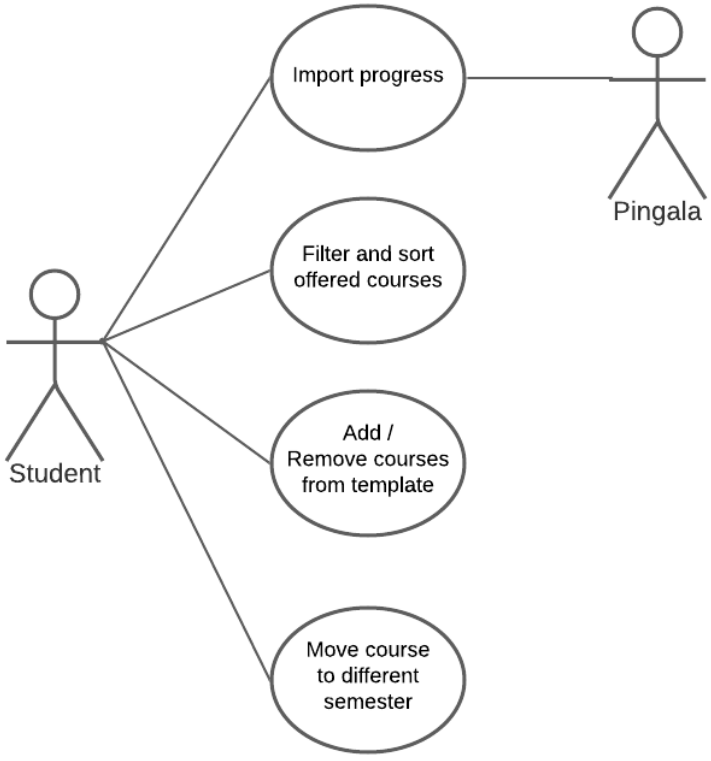
Actors – Student, Pingala

**Exceptions** -

- Some students might get a prerequisite waiver allowing them to take courses they are not eligible for.
- Some students might get an overload/underload waiver allowing them to take credits outside the usual limits.

**Includes** None

**Notes/Issues** - None




**Description:** The course planner provides and easy interface for planning the courses that a student wants to take in the upcoming semesters. A student can import his/her course progress from pingala directly. All courses being offered in the upcoming semester can be filtered and sorted according to course type, credits, department, professor, timing and

rating. These courses can be dragged and dropped in and out of the template which would update the credits in the semester accordingly and a warning would be raised if the credits don't lie in the valid range.

### 3.3.3 Course feedback (U3)

**Author –** Sachin(180639)

**Purpose** - user wants to add review for the course he/she recently finished doing.

**Requirements Traceability –** Identify all requirements traced to this use case

1. availability of course feedback(which is main feature) on the portal will be fulfilled

**Priority** - High priority

**Preconditions** - Any condition that must be satisfied before the use case begins

Before letting a user add feedback for a course, we must verify whether he/she has actually completed the course or at least registered for the course. Only users satisfying this condition will be allowed to post their feedback.

**Post conditions** - The conditions that will be satisfied after the use case successfully completes

User authentication.

**Actors** – Actors (human, system, devices, etc.) that trigger the use case to execute or provide input to the use case

- Student(editor)
- Student(those who have knowledge of the course and can judge the feedback provided by the editor)
- Student(viewers, those who will use the provided feedback in order to make a choice about their curriculum)

**Exceptions** - Exceptions that may happen during the execution of the use case

Even an eligible user might provide some false feedback about the course(for whatever possible reasons). This can add to the unreliability of the information on the feedback portal.

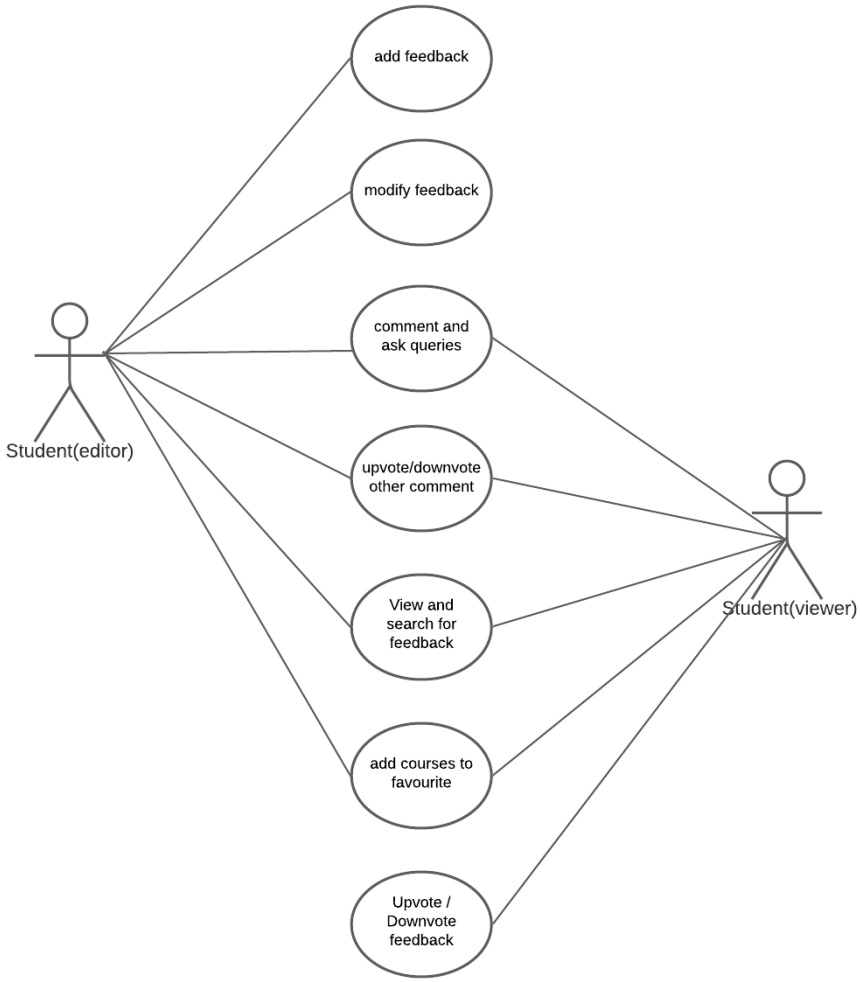
**Includes** (other use case IDs)

None

**Notes/Issues** - Any relevant notes or issues that need to be resolved

None

**UML Diagram:**




**Description** – The course feedback section is used by the students to share course feedback for different courses. All students can view and search for course feedback and add favourite courses accordingly. The author of the feedback can perform actions like modify

the original post, comment under the post and upvote/downvote any comments under this post. However, he/she cannot upvote/downvote his own post. Any other student who views the port can perform all these actions except modifying the original post or any comment that is not added by him/her.

### 3.3.4 Note Taking (U4)

**Author –** Sachin(180639)

**Purpose** - To fulfill the main functionality of taking and managing digital notes on the portal. It will help group similar notes(based on their tags) together and make their retrieval easier. Also students will be able to add handwritten notes in the form of PDFs to the same portal.

**Requirements Traceability –** note-taking and note-management

**Priority** - High

**Preconditions** - User authentication.

**Post conditions** - The conditions that will be satisfied after the use case successfully completes

1. Any changes made to notes should be successfully saved to the database.

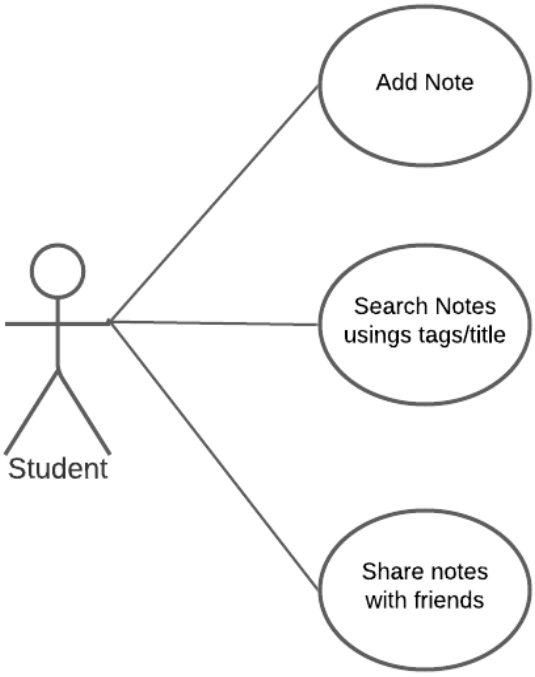
**Actors** – Student

**Exceptions** - Internet failure or any such disturbance should not erase already made changes to a document(if it was not saved). There must be some means to restore those changes.

**Includes** - None

**Notes/Issues** - None

**UML diagram:**




**Description:**

This feature will provide mainly two functionalities.

- Add note: will let the user add notes with title and tags associated with the note. Users can add raw text or images into those notes. Tags will help store similar notes together
- Search notes: will let the user search among a bunch of notes. Users can search using the title of note or tag associated with the note etc.
- Share notes: will let users share his/her notes with the friends. We will enable sharing through our platform itself.

# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

No particular performance requirements.

## 4.2 Safety and Security Requirements

- Security for Application Login: the account contains academic information such as their courses, future course planning, course deadlines. It might also contain personal information if the user uses the calendar for personal events.
- Security for HelloIITK/Pingala login (same credentials as CC ID and password): Automating the addition of assignment deadlines, classes, and academic information requires scraping data from HelloIITK and Pingala, and the software will ask for those credentials. The credentials for those platforms are the same as the IITK CC credentials which can be used to access institute emails, CC services, personal files on the CC computers, HelloIITK, personal data from Pingala, etc. Hence, ensuring that these credentials remain secure is of huge importance.
- Security for Anonymity in course feedbacks: The feedbacks for courses can be given anonymously, and hence the anonymity of the user must be maintained.

## 4.3 Software Quality Attributes

### 4.3.1 Usability

The application is intended to be a companion to the students to assist them in their academics and boost their productivity. Hance, usability, i.e. ease of using/learning to use the software is an important aspect.

- Quick tour on the first time the user logs into the application
- User manual will be provided to the users which details the various steps needed to use the application
- Proper feedback and error messages/pop-ups will be shown on the completion of various tasks

### 4.3.2 Availability

As the application will be needed throughout the semester, it is important that it is available as much as possible.

- The application will be tested regularly and extensively to ensure the removal of major bugs early on.

- [1] Various issues and possible points of failures to be noted with their severity as well as likelihood of happening (risk matrix)
- Proper automatic error handling to be implemented in the code so that failures are notified as soon as possible

# Appendix A – Data Dictionary

**Actors:**

| Actor | Description |
|---|---|
| Student | The student currently using the Academic Management Software |
| Pingala | Service that imports student's course progress, course prerequisites, time table and student details from pingala.iitk.ac.in |
| Mookit | Service that imports assignment deadlines and quiz timings from hello.iitk.ac.in |
| Gradescope | Service that imports assignment deadlines and quiz timings from gradescope.com |

**Functional requirements:**

| Functional Requirement | Description |
|---|---|
| F1 | calender management service, include calender view, daily reminders, add event feature |
| F2 | course planner feature, to plan current semester. also include student progress view to track course completed and their details |
| F3 | Course feedback include features like add course review, browse through reviews, upvote or downvote a review, add course to favourite courses etc. |
| F4 | note-taking feature to add quick notes, and browse/search through added notes, share notes with friends etc. |

**Use Cases:**

| Functional Requirement | Description |
|---|---|
| U1 | Calendar Management |
| U2 | Course Planner |
| U3 | Course feedback |
| U4 | Note Taking |

# Appendix B - Group Log

| *Date* | *Members Present* | *Topic of Discussion* |
|---|---|---|
| 26th January, 2022 | All members were present | Analysis of the sections in the SRS and work distribution for its first draft |
| 28th January, 2022 | All members were present | Review of the work done on the SRS till then, and discussion on the use cases. |
| 28th January, 2022 | All members were present | Discussion with the project mentor, and reviewing of the SRS by the mentor |
| 30th January, 2022 | All members were present | Additions to be made to the existing document were thoroughly discussed |
| 1st February, 2022 | All members were present | The document was reviewed again by the mentor and the suggestions were worked towards. |