# Design Document

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:** **15 Feb 2022**

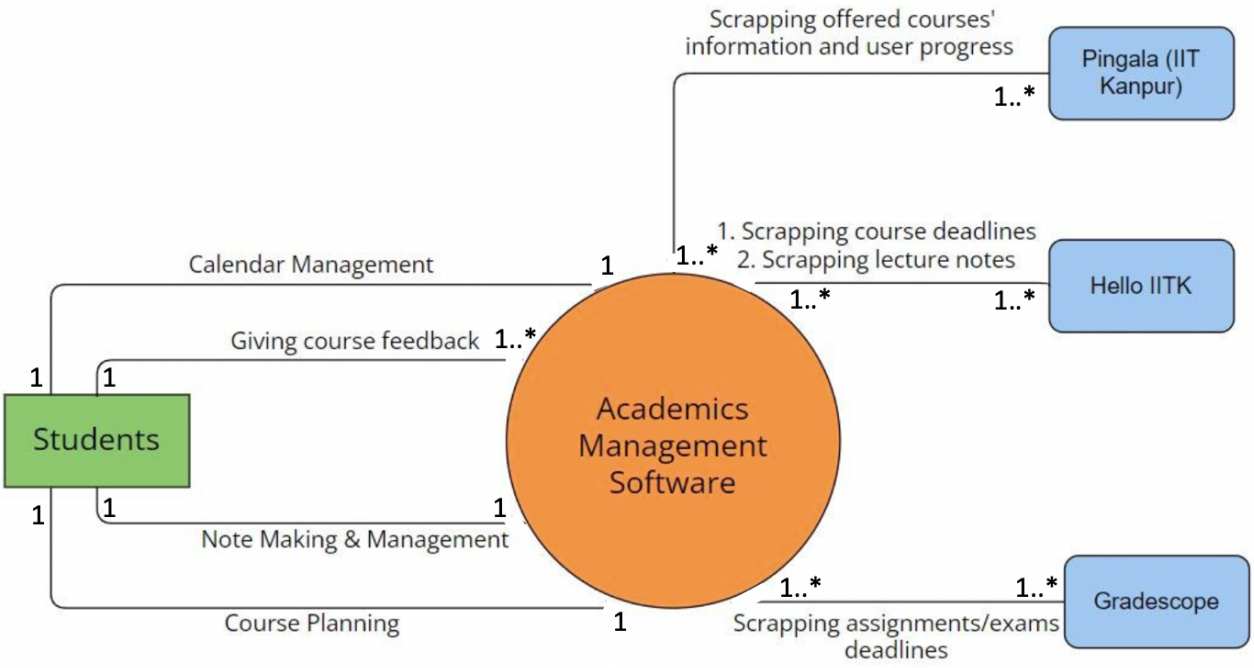
# CONTEXT

# 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. | 15/02/01 |

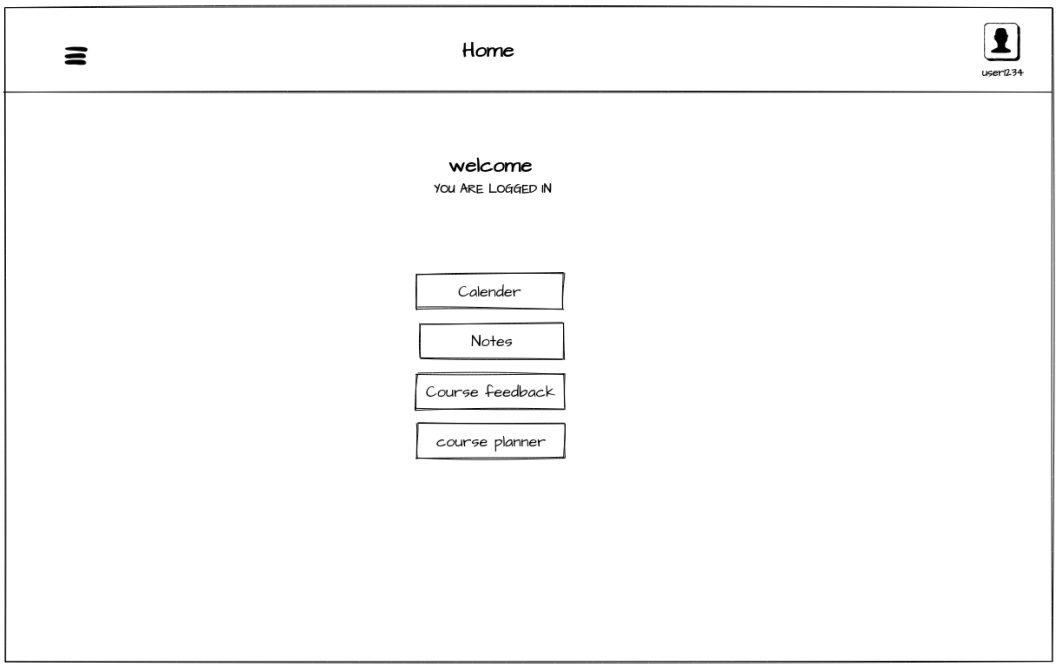
# 1 Context Design

## 1.1 Context Model

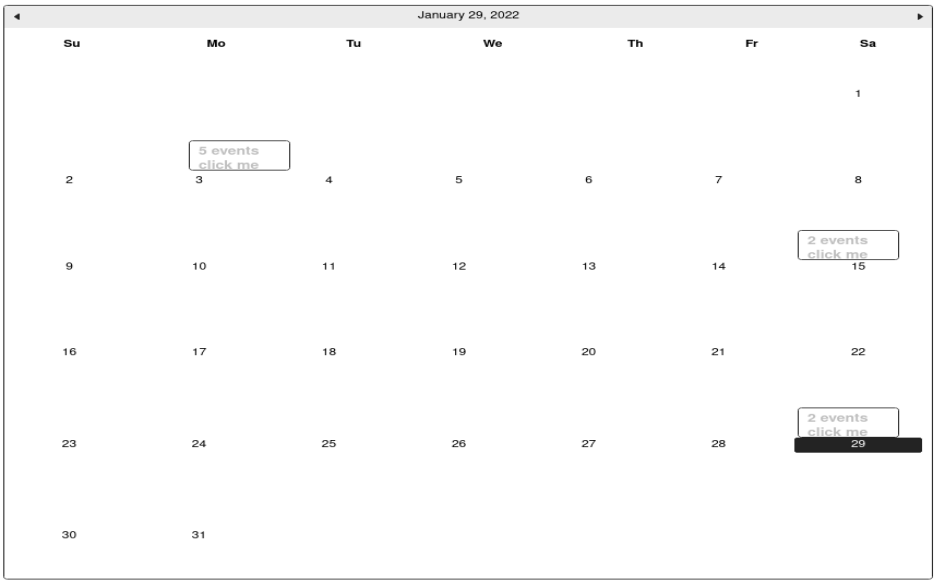



## 1.2 Human Interface Design

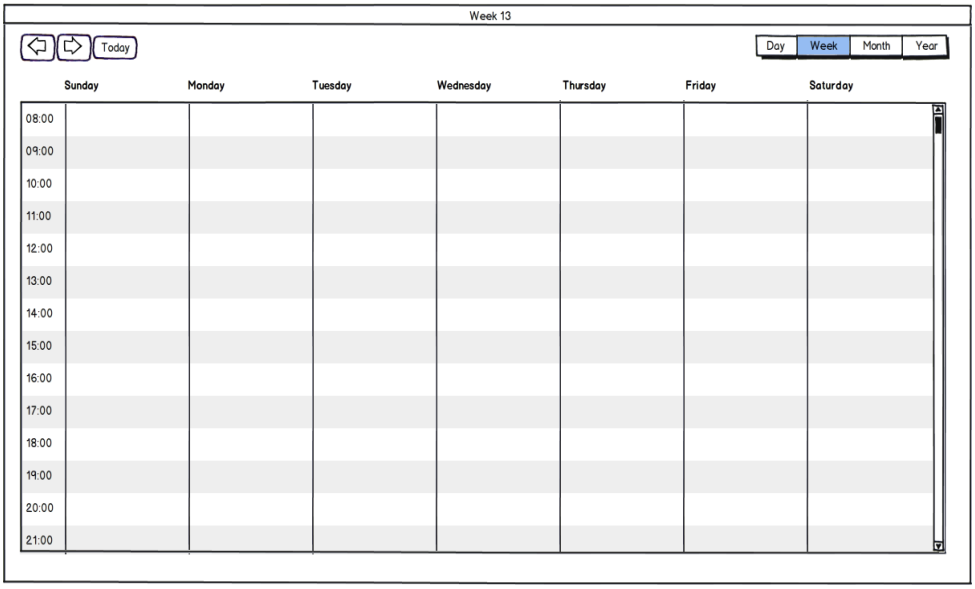
- Application Home.
  The user arrives at this page after logging in. All four services can be accessed here.



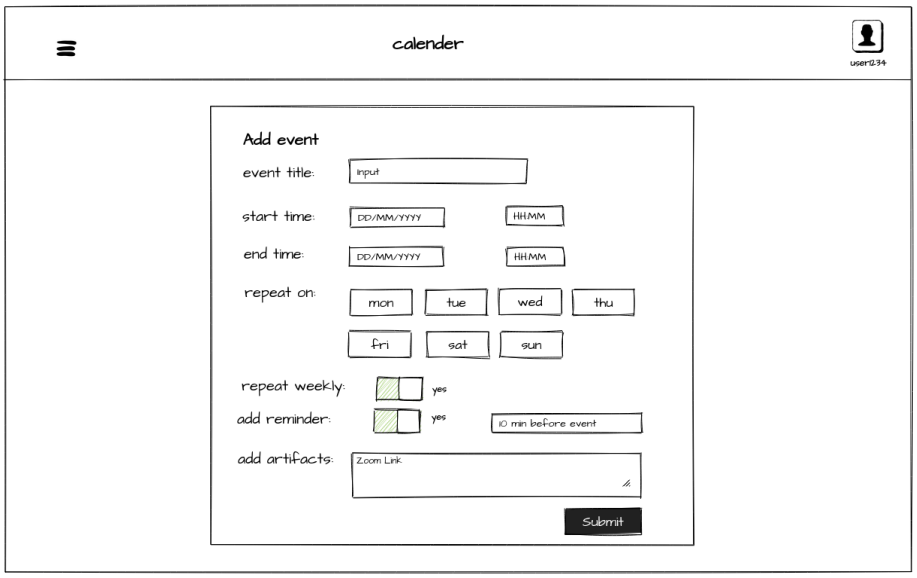
- Calendar
  - The user will have the option to switch between a weekly view and a monthly view.
  - Monthly view.
    **Description :** Clicking on the event-button on any date will display a list of events for that day, with their description. No button will be visible for dates with no events added.



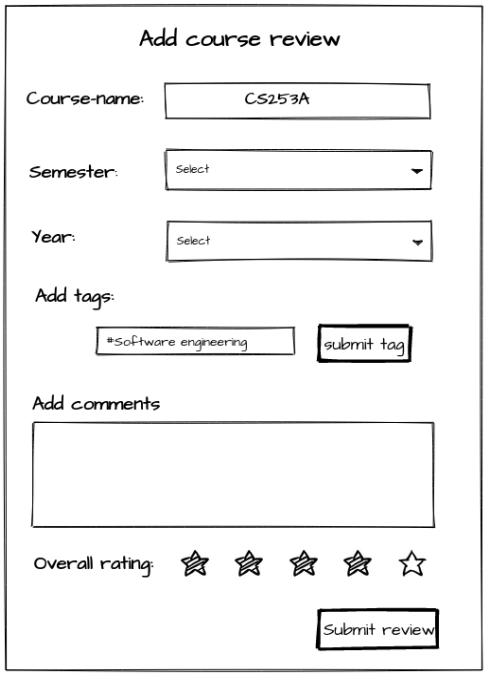

  - Weekly view
    **Description:** Clicking on the event-button in the time-slot of an event under any day will show a description of the event.




  - 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.

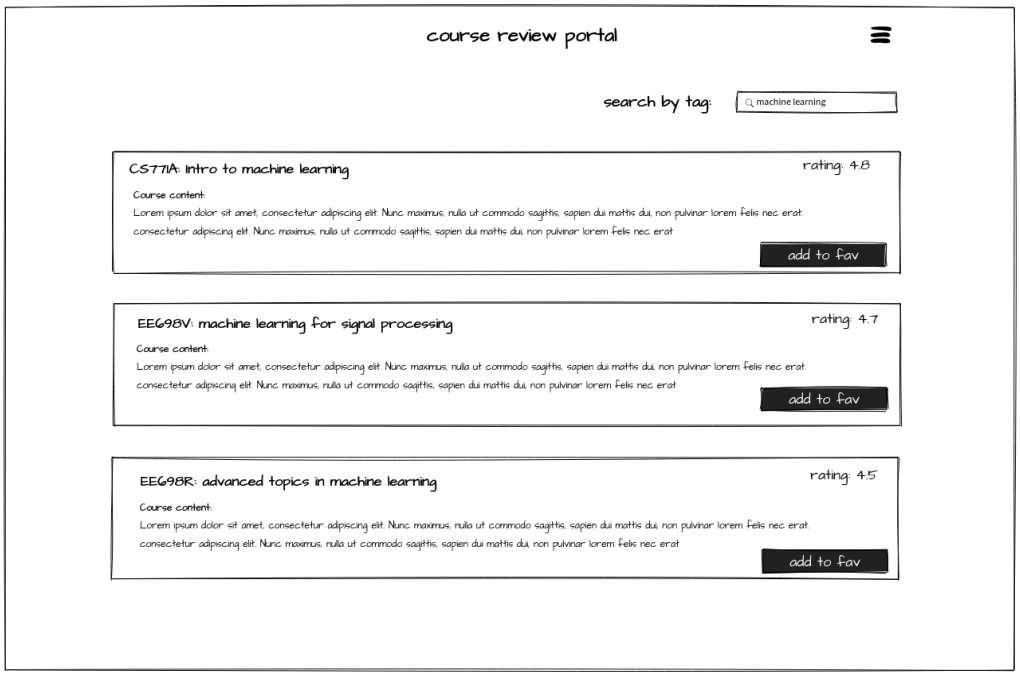



- Course Feedback
    - Add course review
      **Description:** Users will be presented above form in order to add a new course review or edit an existing course review.

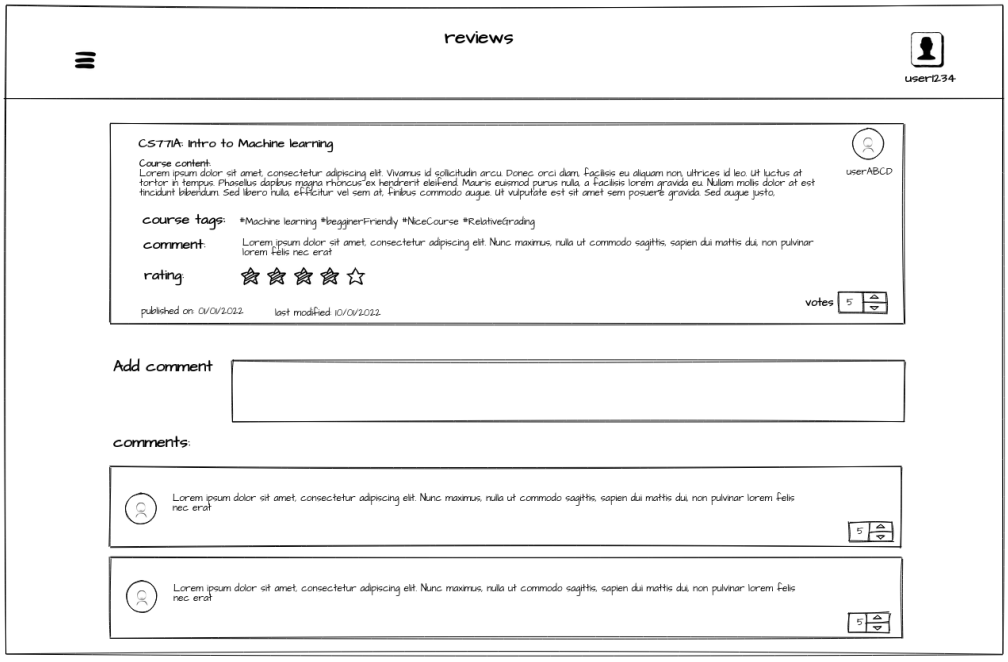



    - Search for courses
      **Description:**

- 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

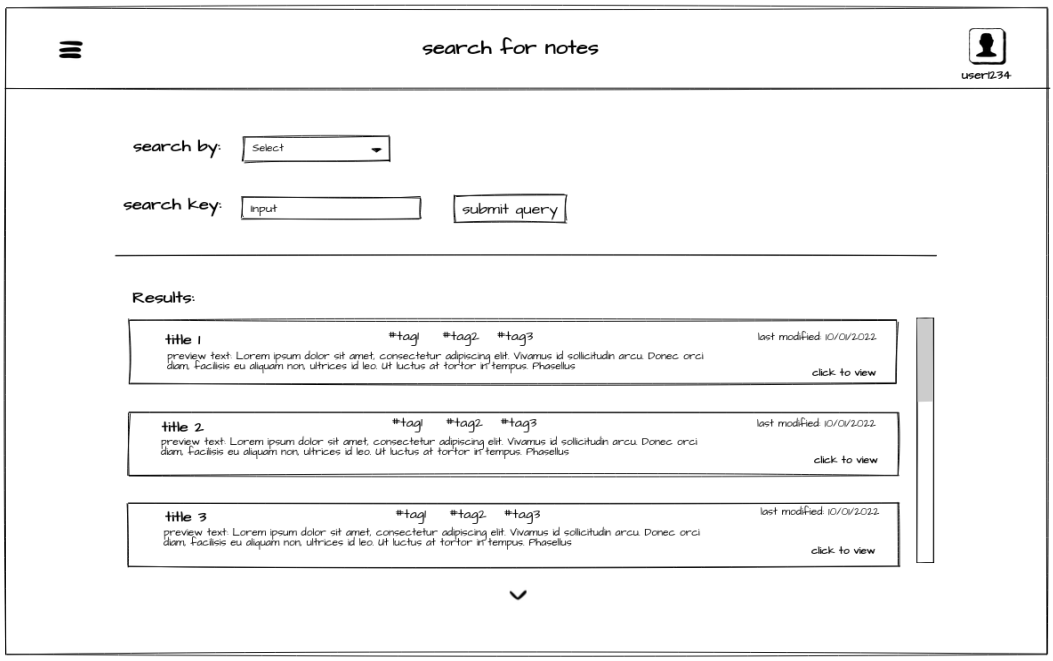
■ user can upvote/downvote a particular review
■ user can provide comments regarding a given review and also provide his/her reaction on other's comments

- Note-taking
  - Add new note

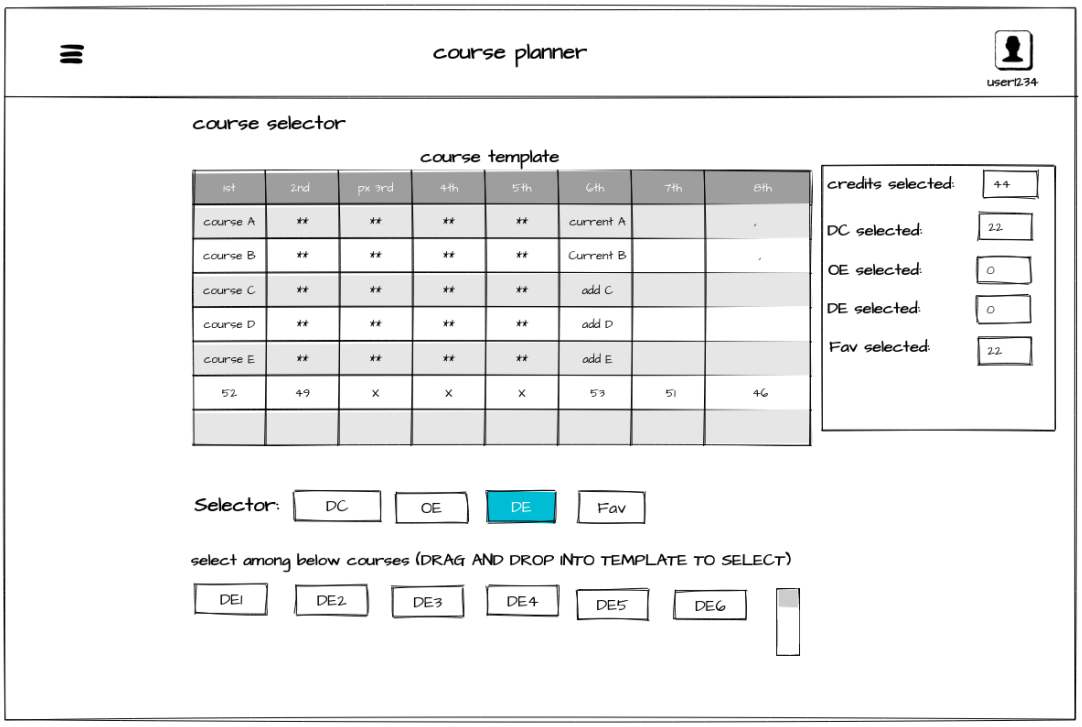
**Description:**

- 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

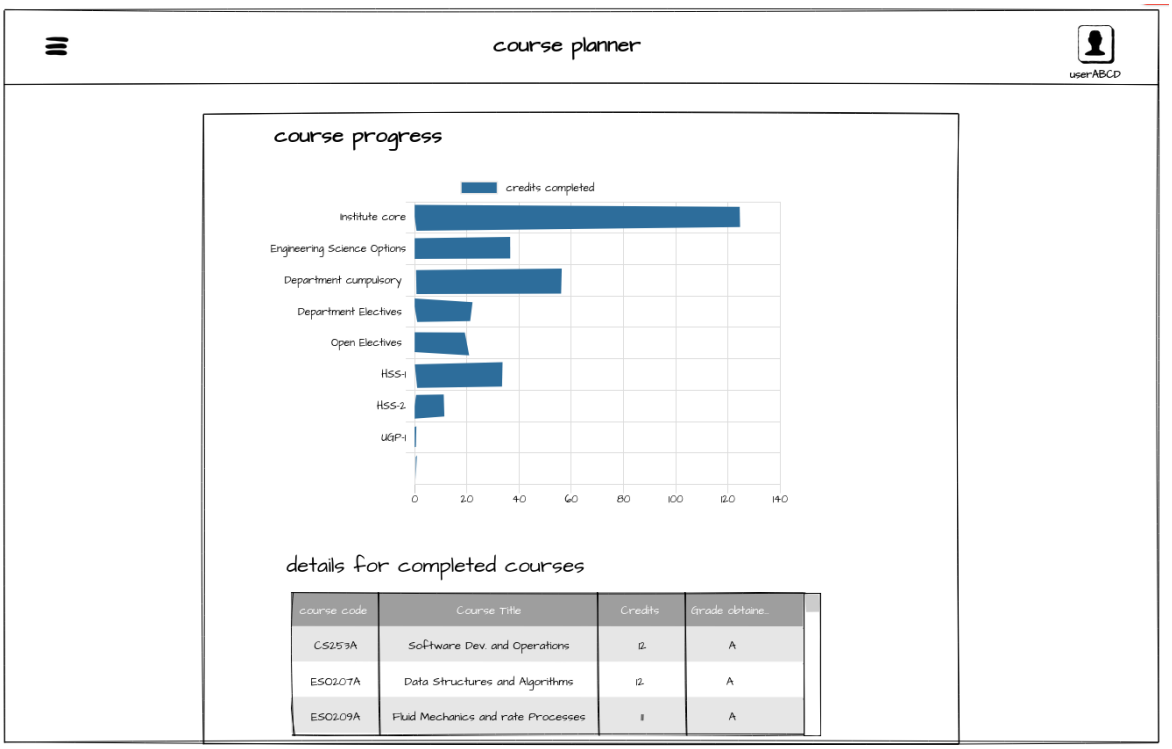
and related courses will be fetched to display. These displayed courses can be dragged and dropped into templates.

- 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

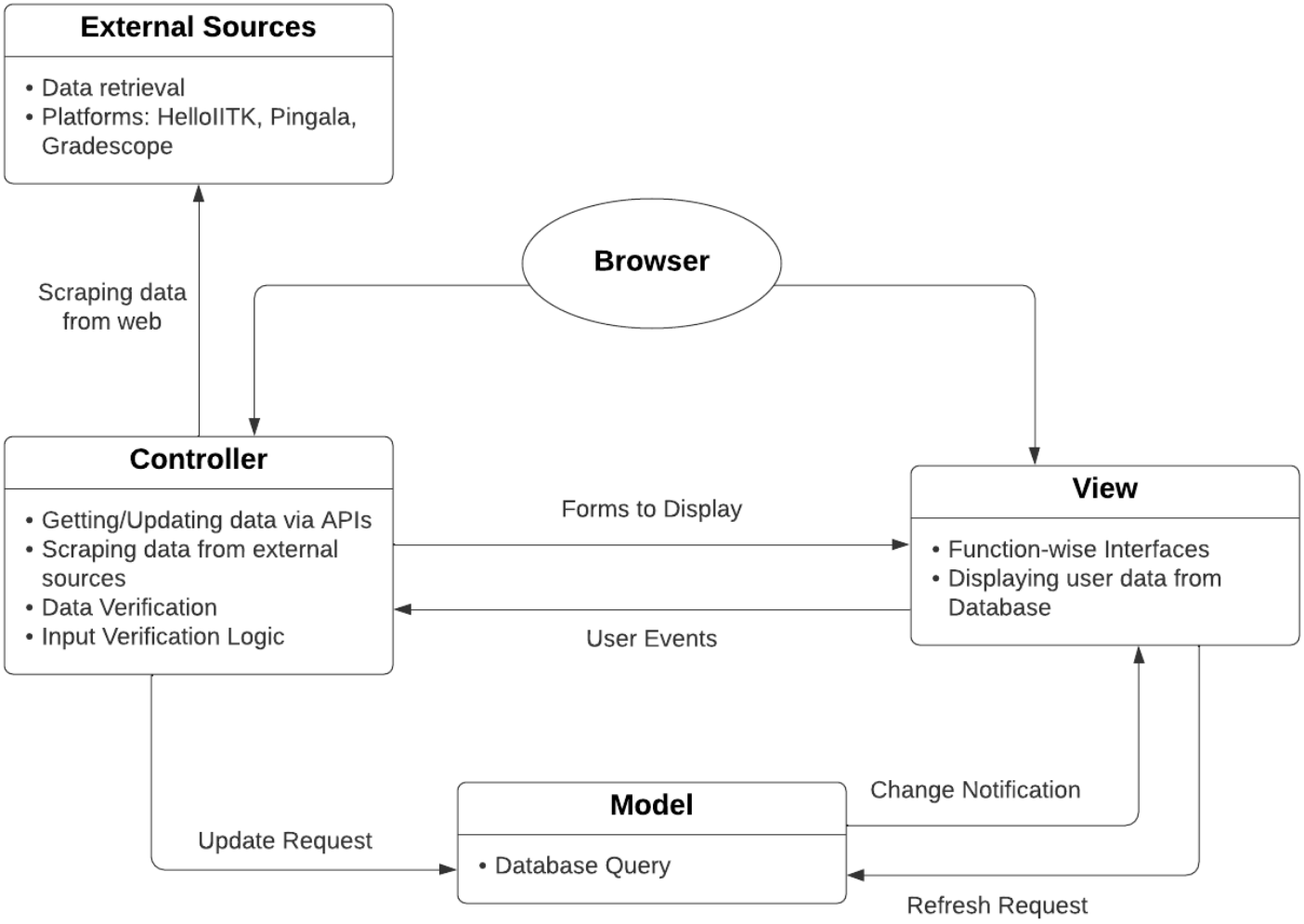
  **Description:**

  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.



# 2 Architecture Design

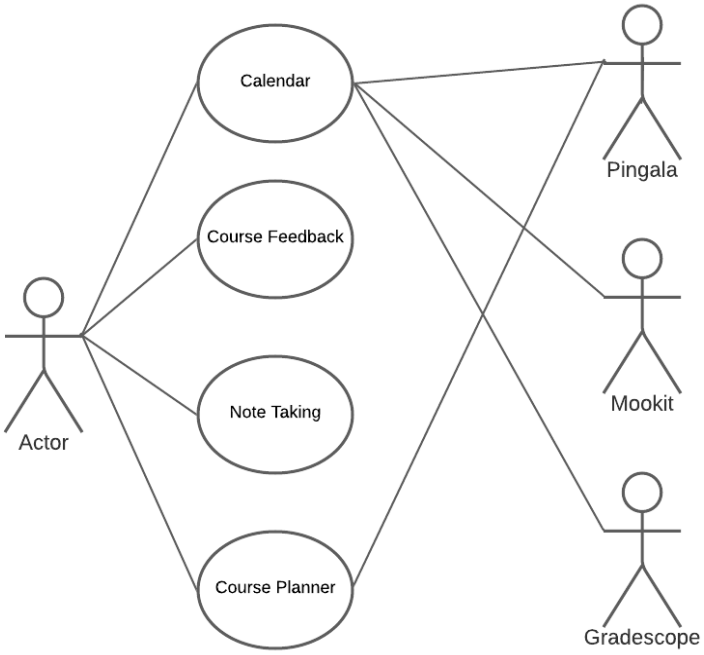
The architecture is mainly a Model-View-Controller pattern architecture, with an extra block for external sources (i.e. HelloIITK, Pingala, Gradescope platforms).



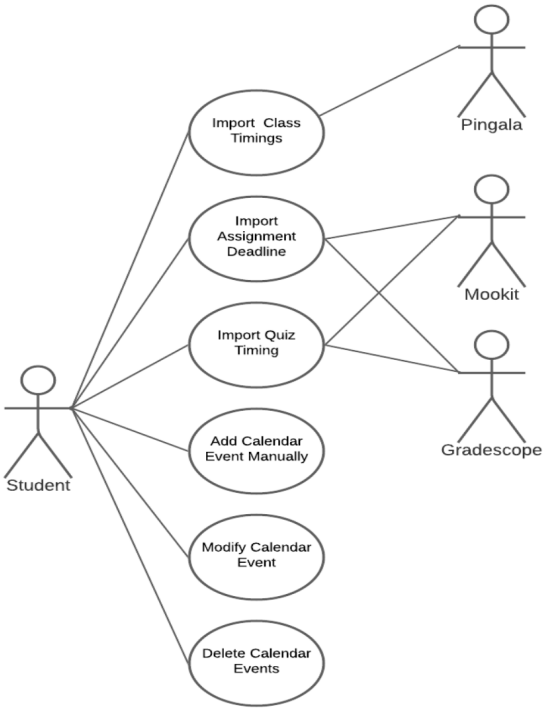
# 3 Object Oriented Design

## 3.1 Use Case Diagrams

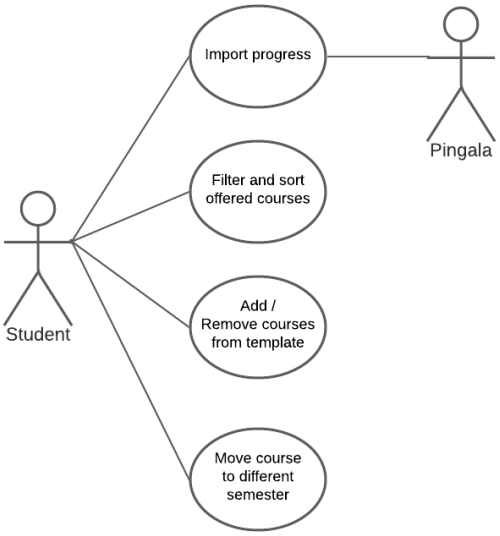
Overall



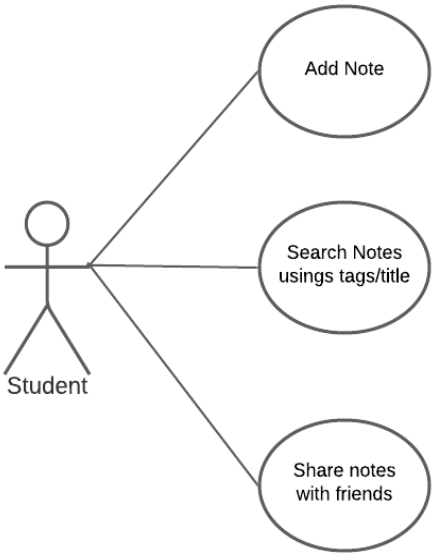

### 3.1.1 Calendar Management



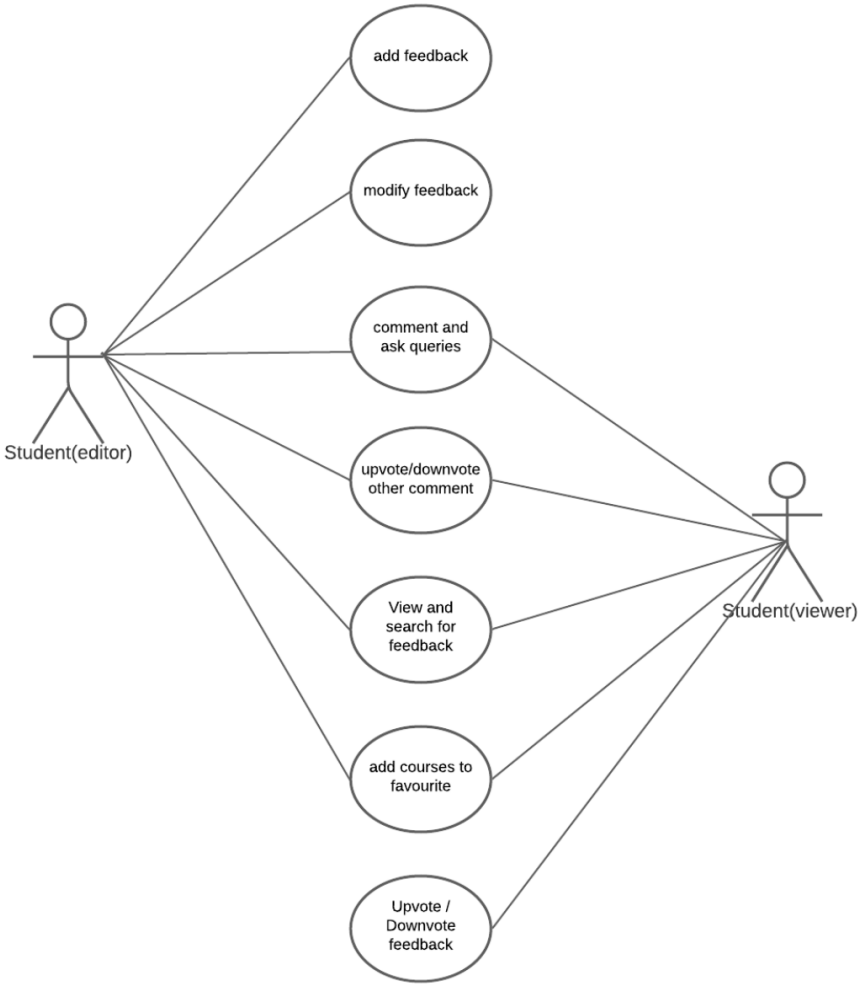
### 3.1.2 Course Planner



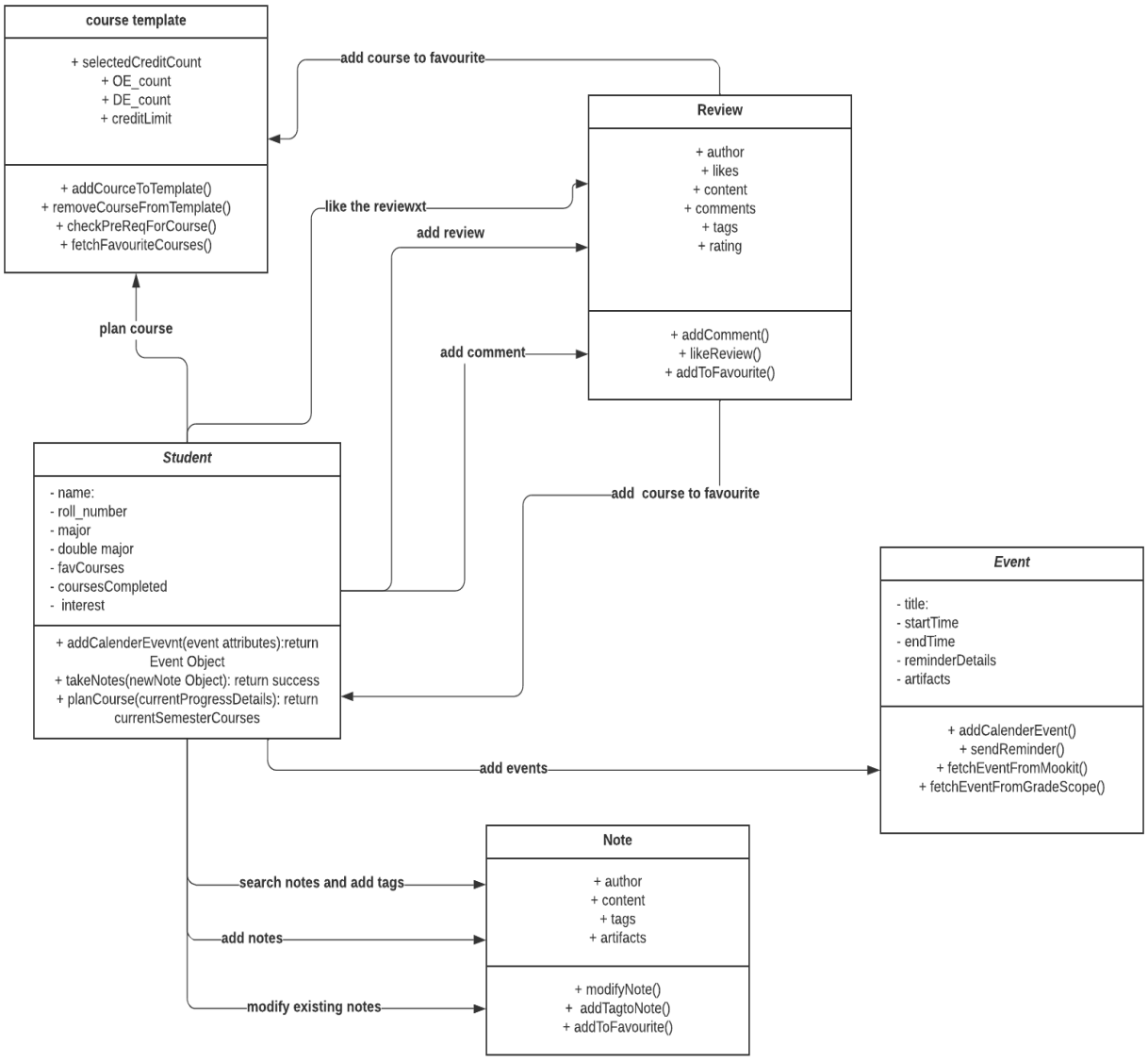

### 3.1.3 Note Taking

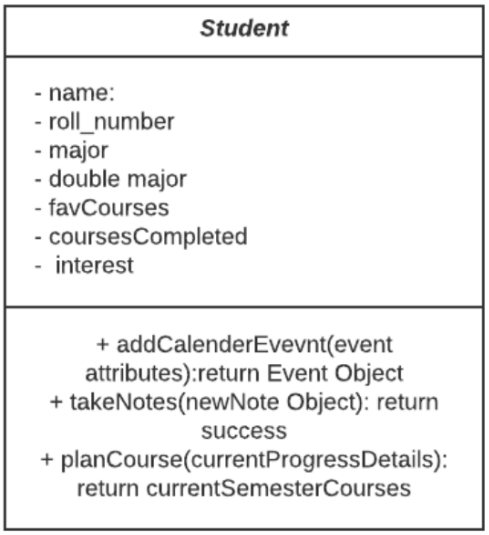


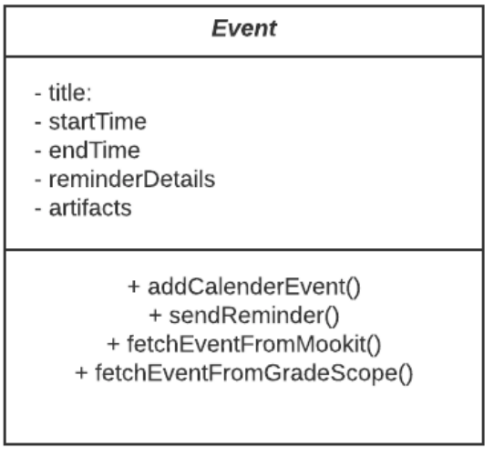
### 3.1.4 Course Feedback




## 3.2 Class Diagrams

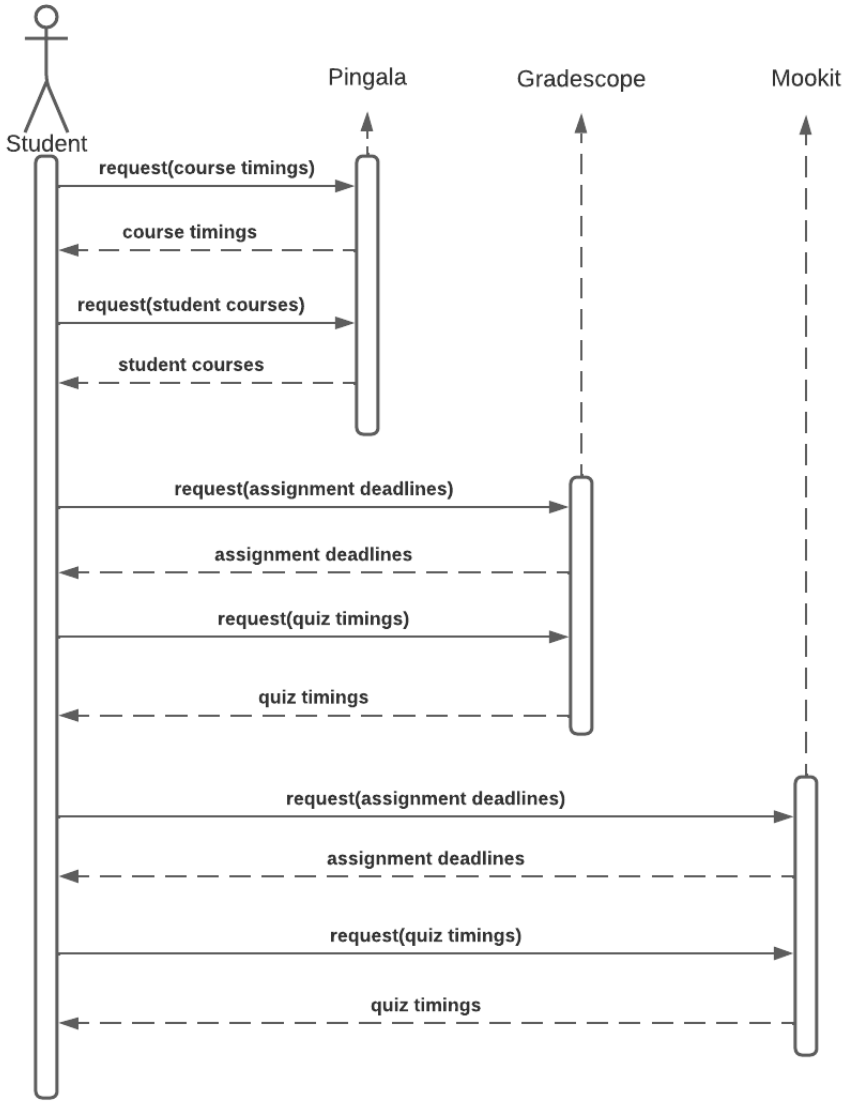

course template
+ selectedCreditCount
+ OE_count
+ DE_count
+ creditLimit
+ addCourceToTemplate()
+ removeCourseFromTemplate()
+ checkPreReqForCourse()
+ fetchFavouriteCourses()
add course to favourite
Review
+ author
+ likes
+ content
+ comments
+ tags
+ rating
+ addComment()
+ likeReview()
+ addToFavourite()
like the reviewxt
add review
plan course
add comment
Student
- name:
- roll_number
- major
- double major
- favCourses
- coursesCompleted
- interest
+ addCalenderEvevnt(event attributes):return Event Object
+ takeNotes(newNote Object): return success
+ planCourse(currentProgressDetails): return currentSemesterCourses
add course to favourite
Event
- title:
- startTime
- endTime
- reminderDetails
- artifacts
+ addCalenderEvent()
+ sendReminder()
+ fetchEventFromMookit()
+ fetchEventFromGradeScope()
add events
Note
+ author
+ content
+ tags
+ artifacts
+ modifyNote()
+ addTagtoNote()
+ addToFavourite()
search notes and add tags
add notes
modify existing notes



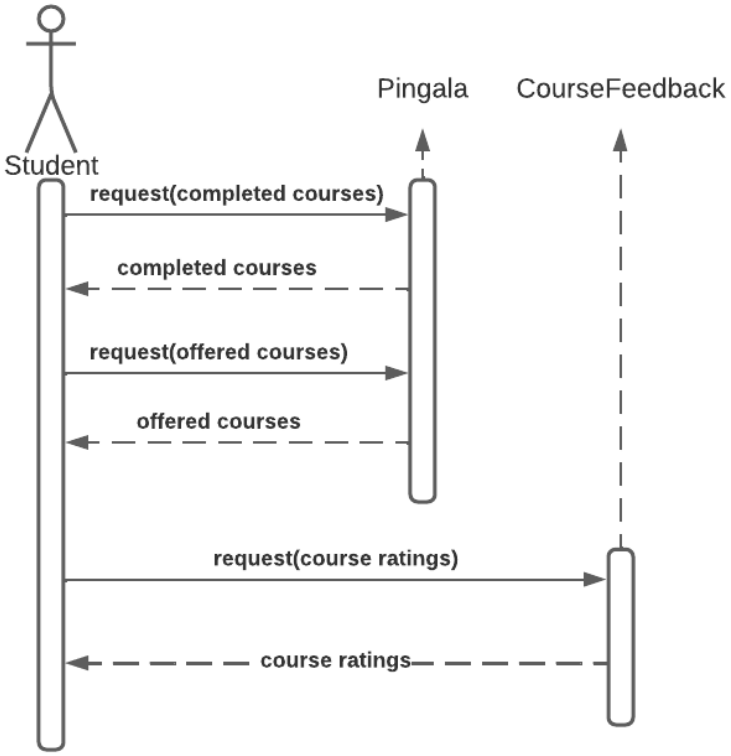




## 3.3 Sequence Diagrams

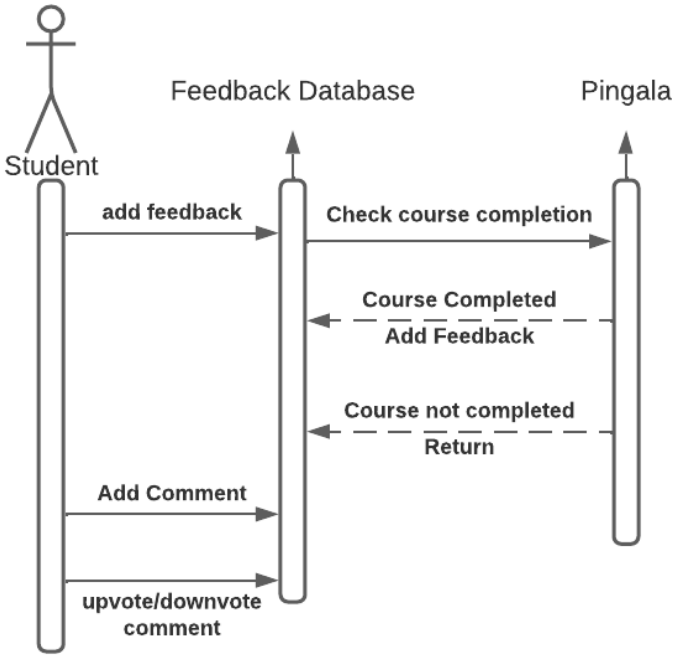
### 3.3.1 Calendar



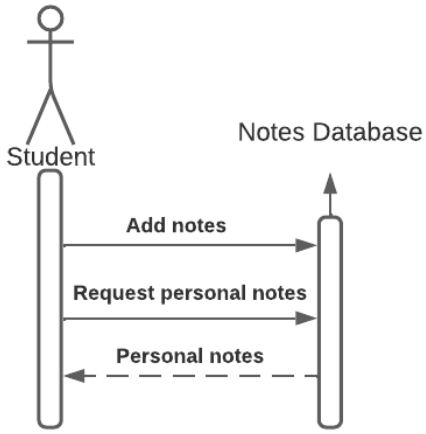

### 3.3.2 Course Planner



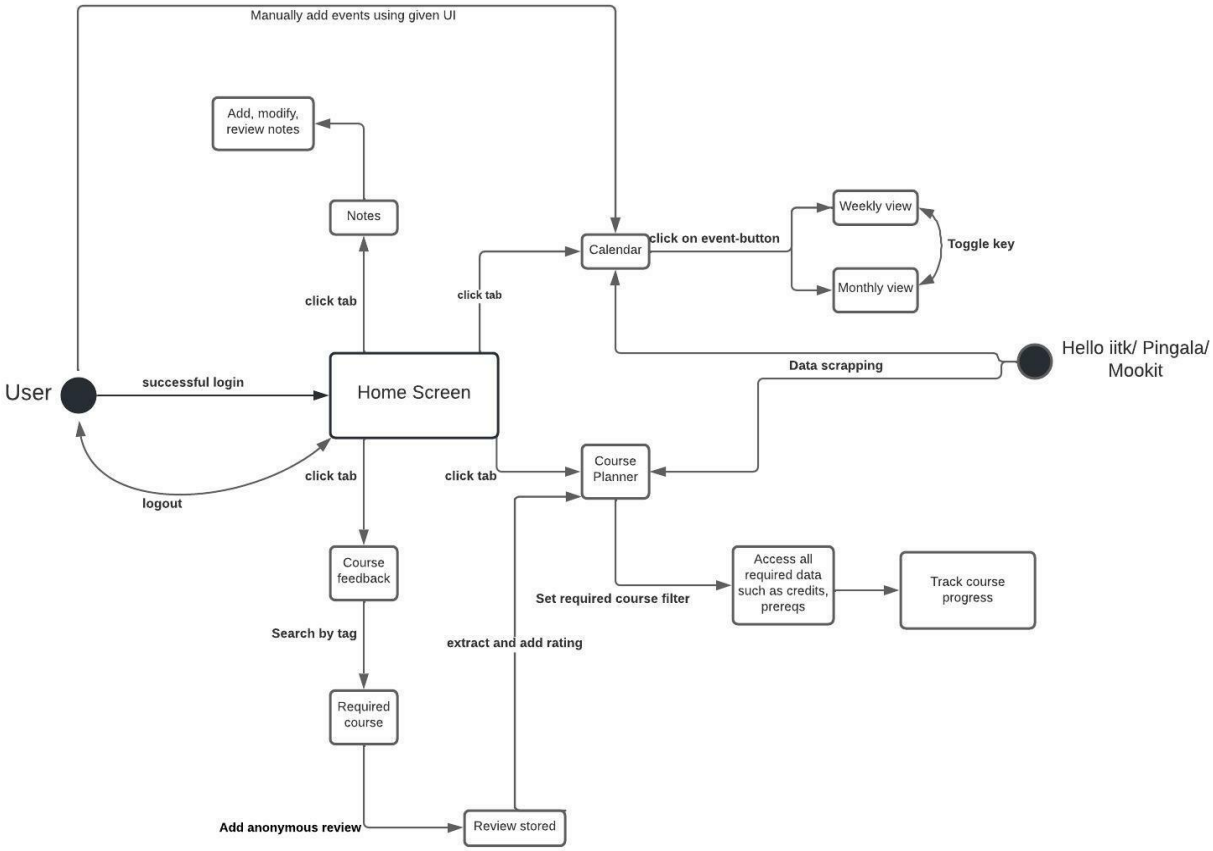

### 3.3.3 Course Feedback



### 3.3.4 Note Taking




## 3.4 State Diagrams



# 4 Project Plan

*<Provide a detailed plan for the implementation and testing of the software. Identify the major tasks and their dependencies. Capture the plan in a Gantt Chart. Also, for each task, identify the team member who will be responsible for completing the task.>*

| Task | Start Date | Expected Duration | Contributing Members | Progress |
|---|---|---|---|---|
| Project Ideation | | | All | 100% |
| Conceptualisation with Mentors | | | All | 100% |
| Development of SRS | | | All | 100% |
| Development of design template | | | Somya, Sachin, Aditya, Achint, Yash (190998), Udhav | 100% |
| Logic flow generation | 16-02-2022 | 2 days | Achint, Sachin, Himanshu | 75% |
| Backend framework generation of different modules and classes | 18-02-2022 | 2 days | Piyush, Somya, Udhav | 40% |
| Login - authorization scheme generation | 20-02-2022 | 3 days | Yash, Yash, Utkarsh | 40% |
| Development of web-scraping mechanisms | 20-02-2022 | 1 week | Aditya, Utkarsh, Somya, Sachin | 30% |
| Database Generation | 24-02-2022 | 1 week | Yash, Yash, Piyush | 0% |
| Database Integration with backend framework | 28-02-2022 | 1 week | Yash (190997), Aditya, Sachin | 0% |
| Frontend prototype generation | 28-02-2022 | 3 days | Piyush, Achint, Aditya | 60% |
| Login page front-end implementation | 03-03-2022 | 2 days | Somya, Udhav, Himanshu | 0% |
| Calendar page front-end implementation | 03-03-2022 | 3 day | Piyush, Utkarsh, Sachin | 0% |

| | | | | |
|---|---|---|---|---|
| Notes page front-end implementation | 05-03-2022 | 2 days | Somya, Aditya, Achint, Udhav | 0% |
| Course review front-end implementation | 07-03-2022 | 2 days | Utkarsh, Udhav, Yash (190998) | 0% |
| Course planner front-end implementation | 09-03-2022 | 2 days | Aditya, Achint, Himanshu | 0% |
| Resolution and integration of the back end with the front-end | 11-03-2022 | 2 days | Piyush, Sachin, Somya | 0% |
| Deployment | 13-03-2022 | 2 days | Udhav, Utkarsh | 0% |
| Alpha Release | 15-03-2022 | 2 days | Aditya, Yash, Yash, Somya | 0% |
| Beta Release | 20-03-2022 | 5 days | Piyush, Himanshu, Utkarsh, Udhav | 0% |

## Frontend

1. Welcome page - This will act as the entry page for the website. It will contain a brief introduction of the website as well as links to sign up and log in
2. Login page - Will ask the user for authentication details (username and password) to enter the website. Will match credentials with the database
3. Sign up page - Will allow users of the institute to sign up.
4. Add Course review - Users can add course reviews
5. View Course reviews - Will allow users to view course reviews filtered by course name, view detailed course reviews, like and add comments on them.
6. Calendar - The calendar application should allow users to add tasks, with the option to repeat them on a weekly basis. Three views will be provided, a daily view, a weekly view, and a monthly view.
7. Course Planner - Users can add semesters, and the courses they have done in each semester. Will ensure that each course can be added only once.
8. Profile - A basic profile page containing username, roll number, department and degree of the student.

## Backend

- **Database Designing** -
  Before starting the implementation of the backend APIs, the team worked on deciding database structure for storing user and course details.

- User details: Stored user details like login credentials and other user-specific details like name, hashed-password, rollNo, department, semesters completed, favorite courses, course plan and notes. (Primary Key - username)
- Course details: Stored all course related details in one single collection. The details include course name, course ID, credits, class schedule, pre-requisites, etc. (Primary Key - course ID)
- User Notes: Notes created by the users are stored in a separate collection. A document contains the following fields - username, title, courseID, content, timestamp. (Primary Key - _id - default ID provided by mongo DB for every document)
- User Calendars: Calendars associated with each user are stored in a separate collection of the database. The collection contains sufficient information about the events scheduled by the user in the calendar. (Primary Key - username)
- Course reviews: Course reviews given to the courses are stored in a different collection to maintain consistency in data management. For easy navigation through the reviews, a separate id - reviewID field has been created that recognizes a review associated with a course and its author uniquely. A single collection contains - reiviewID, review, a list of comments given to that review. (Primary Key - reviewID)
- Semester Plan - User plans his/her semester by choosing the courses he/she wants to take in the next semester. There is a separate collection for storing these plans as well. (Primary Key - _id)

- **Backend Logic Design** -
  Implemented the core logic for the various functionalities used in the backend
  - registerUser() - Registers a new user into the application using username, rollNo, name, department and a valid password.
  - loginUser() - Log a new user into the application and then generate a JWT and store it into the cookie of the browser
  - logout() - Logout the user
  - addReview() - add a review to the course and assign a unique review ID to the review
  - addComment() - add a comment to a course review with a unique comment ID
  - addCalendarEvent() - add a calendar event with the start/end time/date, title, description.
  - addNote() - Add a note with note title, and the note content.
  - addSemPlan() - Add a new semester with the desired courses from a list of available courses in that semester.

## Product Testing

*Development Testing*

1. Initial testing of the backend to be carried out using Postman API calls
2. Once the frontend is ready, it will be tested for usability, ease of navigation and functionality manually
3. We will carry out unit tests using Jest

4. After integrating the backend and frontend, integration tests will be conducted to check if application is functioning per expectations, and any found bugs will be fixed.

*Alpha testing*

1. To be carried out by members of the team
2. Will do a thorough check for bugs and evaluate the overall usability
3. Any bugs found at this stage will be fixed.

*Beta Testing*

1. To be carried out by members of another team
2. Will receive a test document and work on fixing the bugs identified, prioritizing them by impact on user experience and application usability

# 5 Other Requirements

<*This section is Optional. Please provide any other details that are suitable for being included in the design document .*>

# Appendix A - Group Log

<*Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document*>

| DATE | MEMBERS PRESENT | TOPIC OF DISCUSSION |
|---|---|---|
| 02- 03- 2022 | All members were present | A selection of the language, library frameworks based on advantageous features and team members' knowledge. |
| 06- 03- 2022 | All members were present | A more precise division of tasks was done and the broad structure of the codebase was decided. |
| 10- 03- 2022 | All members were present | Progress on the tasks assigned was gauged and allocation of new tasks was done. |
| 13- 03- 2022 | All members were present | Implementation document was brought up and how to approach it was discussed, besides progress check. |
| 17- 03- 2022 | All members were present | Updates were done in favour of better features and the code completed was discussed. |
| 19- 03- 2022 | All members were present | The code completed was discussed and changes to be made were discussed. |