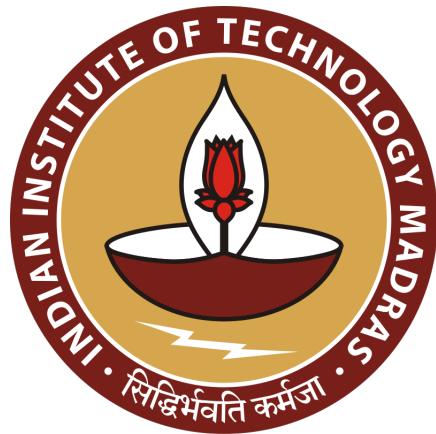


INDIAN INSTITUTE OF TECHNOLOGY, MADRAS



SOFTWARE ENGINEERING PROJECT FINAL PROJECT REPORT

A Learning-Path Recommendation System for IIT(M) B.S. Degree

Submitted by:

ANCHIT MANDAL	21F1000692
ANHAT SINGH	21F2000381
JASLEEN KAUR CHEEMA	21F1006537
OZA JAY AMIT	21F1001944
ROHIT VINIT LONDHE	21F1006995

December 20, 2023

PROJECT COMPLETION

The software engineering project, titled "A Learning-Path Recommendation System for IIT(M) B.S. Degree", has been successfully completed under the supervision of instructors of Software Engineering Course, IIT Madras.

The project aimed to develop and implement a system that can recommend personalized learning paths for students pursuing the B.S. degree at IIT Madras, based on their academic performance, interests, and goals. The project followed the agile software development model. The project was completed within the deadline, and met all the specifications and standards required as per the project statement.

The source code is available on our [GitHub repository](#). We thank instructors at IIT Madras for their support and feedback throughout the project.



(Anhat Singh)



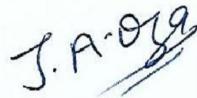
(Jasleen Kaur)



(Rohit Vinit Londhe)



(Anchit Mandal)



(Oza Jay Amit)

Contents

1 User Requirements	1
1.1 Various users of the application	1
1.1.1 Primary Users	1
1.1.2 Secondary Users	1
1.1.3 Tertiary Users	1
1.2 User Stories	1
1.2.1 As a Student	1
1.2.2 As an Admin	2
1.2.3 As a Course Team Member	2
1.2.4 As an IITM Management Member	2
2 User Interfaces	2
2.1 Storyboards	2
2.2 Wireframes	5
2.2.1 General Wireframes	6
2.2.2 Students Pages' Wireframes	6
2.2.3 Administrators Pages' Wireframes	8
2.2.4 Course Team Member Pages' Wireframes	11
2.2.5 IITM Management Pages' Wireframes	12
2.3 Usability Design Guidelines & Heuristics	12
3 Scheduling and Design	14
3.1 Project Schedule	14
3.1.1 Gantt Chart	14
3.1.2 SPRINT Schedule	16
3.2 Project Scheduling Tool	17
3.3 Various Components	17
3.3.1 User Management Component	17
3.3.2 Profile Management Component	17
3.3.3 Course Management Component	17
3.3.4 Enrolment and Analytics Component	17
3.3.5 Feedback and Rating Component	18
3.3.6 Course Catalog Component	18
3.3.7 Data Analysis Component	18
3.3.8 Recommendation Component	18
3.4 Class Diagram	18
3.5 SCRUM Meetings Schedule and Minutes	19
4 API Endpoints	22
5 Test Cases for API Endpoints	24
5.1 A pic of implemented test case	24
5.2 A test case for: http://localhost/api/v1/profile	24
5.3 A test case for: http://localhost/api/v1/profile	25
5.4 A test case for: http://localhost/api/v1/profile	26
5.5 A test case for: http://localhost/api/v1/profile	27
5.6 A test case for: http://localhost/api/v1/profile	27
5.7 A test case for: http://localhost/api/v1/profile	28
5.8 A test case for: http://localhost/api/v1/profile	29
5.9 A test case for: http://localhost/api/v1/profile	30

5.10 A test case for: http://localhost/api/v1/profile	31
5.11 A test case for: http://localhost/api/v1/profile	32
5.12 A test case for: http://localhost/api/v1/profile	33
5.13 A test case for: http://localhost/api/v1/profile	34
5.14 A test case for: http://localhost/api/v1/profile	35
5.15 A test case for: http://localhost/api/v1/profile	36
5.16 A test case for: http://localhost/api/v1/profile	37
5.17 A test case for: http://localhost/api/v1/profile	38
5.18 A test case for: http://localhost/api/v1/profile	39
5.19 A test case for: http://localhost/api/v1/profile	40
5.20 A test case for: http://localhost/api/v1/profile	41
5.21 A test case for: http://localhost/api/v1/profile	42
5.22 A test case for: http://localhost/api/v1/profile	43
5.23 A test case for: http://localhost/api/v1/admins	44
5.24 A test case for: http://localhost/api/v1/admins	45
5.25 A test case for: http://localhost/api/v1/admins	45
5.26 A test case for: http://localhost/api/v1/admins	46
5.27 A test case for: http://localhost/api/v1/admins	47
5.28 A test case for: http://localhost/api/v1/admins	48
5.29 A test case for: http://localhost/api/v1/admins	49
5.30 A test case for: http://localhost/api/v1/admins	50
5.31 A test case for: http://localhost/api/v1/admins	51
5.32 A test case for: http://localhost/api/v1/admins	52
5.33 A test case for: http://localhost/api/v1/admins	53
5.34 A test case for: http://localhost/api/v1/admins	54
5.35 A test case for: http://localhost/api/v1/admins	55
5.36 A test case for: http://localhost/api/v1/admins	56
5.37 A test case for: http://localhost/api/v1/admins	57
5.38 A test case for: http://localhost/api/v1/admins	58
5.39 A test case for: http://localhost/api/v1/admins	59
5.40 A test case for: http://localhost/api/v1/admins	60
5.41 A test case for: http://localhost/api/v1/admins	60
5.42 A test case for: http://localhost/api/v1/admins	61
5.43 A test case for: http://localhost/api/v1/admins	61
5.44 A test case for: http://localhost/api/v1/admins	62
5.45 A test case for: http://localhost/api/v1/admins	62
5.46 A test case for: http://localhost/api/v1/admins	63
5.47 A test case for: http://localhost/api/v1/admins	63
5.48 A test case for: http://localhost/api/v1/user/auth	64
5.49 A test case for: http://localhost/api/v1/user/auth	65
5.50 A test case for: http://localhost/api/v1/user/auth	66
5.51 A test case for: http://localhost/api/v1/user/auth	67
5.52 A test case for: http://localhost/api/v1/user/auth/register	68
5.53 A test case for: http://localhost/api/v1/user/auth/register	69
5.54 A test case for: http://localhost/api/v1/user/auth/register	70
5.55 A test case for: http://localhost/api/v1/user/auth/register	71
5.56 A test case for: http://localhost/api/v1/user/auth/register	72
5.57 A test case for: http://localhost/api/v1/user/auth/register	73
5.58 A test case for: http://localhost/api/v1/user/auth/register	74
5.59 A test case for: http://localhost/api/v1/user/auth/register	75
5.60 A test case for: http://localhost/api/v1/user/auth/register	76
5.61 A test case for: http://localhost/api/v1/recommendation	77

5.62 A test case for: http://localhost/api/v1/recommendation	78
5.63 A test case for: http://localhost/api/v1/recommendation	78
5.64 A test case for: http://localhost/api/v1/courses/{id}	79
5.65 A test case for: http://localhost/api/v1/courses/{id}	80
5.66 A test case for: http://localhost/api/v1/courses/{id}	80
5.67 A test case for: http://localhost/api/v1/courses/{id}	81
5.68 A test case for: http://localhost/api/v1/courses/{id}	82
5.69 A test case for: http://localhost/api/v1/courses/{id}	83
5.70 A test case for: http://localhost/api/v1/courses/{id}	84
5.71 A test case for: http://localhost/api/v1/courses/{id}	85
5.72 A test case for: http://localhost/api/v1/courses/{id}	86
5.73 A test case for: http://localhost/api/v1/courses/{id}	87
5.74 A test case for: http://localhost/api/v1/courses/{id}	87
5.75 A test case for: http://localhost/api/v1/courses/{id}	88
5.76 A test case for: http://localhost/api/v1/courses/{id}	88
5.77 A test case for: http://localhost/api/v1/courses/{id}	89
5.78 A test case for: http://localhost/api/v1/courses	90
5.79 A test case for: http://localhost/api/v1/courses	91
5.80 A test case for: http://localhost/api/v1/courses	91
5.81 A test case for: http://localhost/api/v1/courses	92
5.82 A test case for: http://localhost/api/v1/courses	93
5.83 A test case for: http://localhost/api/v1/courses	94
5.84 A test case for: http://localhost/api/v1/courses	95
5.85 A test case for: http://localhost/api/v1/courses	96
5.86 A test case for: http://localhost/api/v1/course/{id}/feedback	97
5.87 A test case for: http://localhost/api/v1/course/{id}/feedback	98
5.88 A test case for: http://localhost/api/v1/course/{id}/feedback	98
5.89 A test case for: http://localhost/api/v1/course/{id}/feedback	99
6 Project Review	100
6.1 Technologies and Tools Used	100
6.1.1 Backend	100
6.1.2 Frontend	100
6.1.3 Reports and Documentation	100
6.1.4 Other Tools Used	100
6.2 Recommendation System Algorithm	100
7 Screenshots	102
7.1 Login Page	102
7.2 Student Pages	102
7.3 Admin Pages	106
7.4 Course Team Member Pages	108
7.5 IITM Management Member	109
7.6 Github Repository and Google Drive	110
8 Issue Report	111
9 Github Repo, Video and PPT	111
List of Figures	
1 Student Storyboard	3

2	Admin Storyboard	4
3	Course Team Member Storyboard	4
4	IITM Management Storyboard	5
5	Login Page (Registration page will also look similar)	6
6	Student's homepage	6
7	Student's profile page	7
8	Page for Students to view all the courses	7
9	Student's courses and feedback page	8
10	Administrator dashboard	8
11	Administrator all courses page	9
12	Administrator add / edit courses page	9
13	Administrator Page to view course team members, IITM management members and other admins	10
14	Adding / Editing course team members, IITM management members and other admins	10
15	Course team members' Dashboard	11
16	A page for Course Team Members to view feedback given by students	11
17	A page for Course team members for adding / updating course description	12
18	IITM Management Course Feedback page	12
19	Complete Project Timeline	14
20	Milestone 1 and 2 Schedule	15
21	Milestone 3 and 4 Schedule	15
22	Milestone 5 and 6 Schedule	15
23	Sprint Board in Jira	17
24	Basic Class Diagram	19
25	API Endpoints List 1	22
26	API Endpoints List 2	23
27	API Endpoints List 3	23
28	Some implemented test cases using Python	24
29	Algorithm used for Recommendations	101
30	UI Screenshot - Login Page	102
31	UI Screenshot - Student Home Page	102
32	UI Screenshot - Student Profile Page	103
33	UI Screenshot - Student Completed Courses Page	103
34	UI Screenshot - Student All Courses Page	104
35	UI Screenshot - Student Each Course Details Page	104
36	UI Screenshot - Student Each Course Feedback	105
37	UI Screenshot - Student Each Course Give Feedback	105
38	UI Screenshot - Admin Home Page	106
39	UI Screenshot - Admin Courses Page	106
40	UI Screenshot - Admin Edit Course Info	107
41	UI Screenshot - List of other admins	107
42	UI Screenshot - Course Team Member Home Page	108
43	UI Screenshot - Course Team Member View Student Feedbacks Page	108
44	UI Screenshot - Course Team Member Edit Course Details Page	109
45	UI Screenshot - IITM Managemnt Member Page	109
46	Github Repo and the commits done by team members; 74 commits were made during the development phase	110
47	Google Drive folder with shared access	110
48	Issues Tracking in Github	111

List of Tables

1	Sprint schedule created in Jira	16
---	---	----

1 User Requirements

1.1 Various users of the application

1.1.1 Primary Users

Students enrolled in the BS Degree are the primary users of the application. They can add their course-related information, interests, and other relevant details into the app, and get course recommendations for the upcoming terms.

1.1.2 Secondary Users

Admins and the **Course Team Members** are the secondary users of the application. They can add courses into the app, update information about individual courses and ensure that the system is working well.

1.1.3 Tertiary Users

IITM Management is the tertiary user of the application. They can benefit from the knowledge base generated by the course feedback and improve the overall structure of the degree as well as the course availability based on popularity of courses.

1.2 User Stories

1.2.1 As a Student

→ **As a Student,**

I want to be able to register with my IITM Email ID,
So that I am able to access the app.

→ **As a Student,**

I want to create my profile after registration and add my completed courses, their marks, number of hours I can give per week, the maximum number of subjects I can take per term,

So that So that the app can recommend courses accustomed to my needs.

→ **As a Student,**

I want to be able to choose between Data Science, Programming or Both as my primary interests,

So that the recommendations system is able to provide my favorite subjects in earlier terms.

→ **As a Student,**

I want to update my profile after each term and add the courses I have completed with their marks and their feedbacks,

So that this data can be used by the app to provide recommendations to other students.

→ **As a Student,**

I want to view the list of courses as per the prerequisites with their feedbacks and Difficulty Ratings from other students,

So that I can see which courses are available in the upcoming terms.

→ **As a Student,**

I want to view the recommendations for the upcoming terms,

So that i can make right decisions about course selection according to my educational goal.

→ **As a Student,**

I want a Statistical summary of my Academic progress so far,
So that I can judge my performance.

→ **As a Student,**

I want to see details about individual course, the feedbacks provided by other students, and the ability to +1 the feedbacks I find helpful,
So that I get a better overview about the courses.

1.2.2 As an Admin

→ **As an Admin,**

I want to load enrollment data from previous terms using a single Excel file in 2 seconds,
So that the system can use this data to generate accurate recommendations for students.

→ **As an Admin,**

I want to manage and update the course database,
So that the recommendations are based on the most up-to-date course offerings.

→ **As an Admin,**

I want add and update the term-availability details of each course,
So that the students can get valid term-wise recommendations.

→ **As an Admin,**

I want be able to create, block and delete Course Team Member Accounts, IITM Management Accounts,
So that new users are able to join easily.

1.2.3 As a Course Team Member

→ **As a Course Team Member,**

I want to access the feedback received from the students,
So that we can make informed improvements and enhancements to the course content and structure.

→ **As a Course Team Member,**

I want to be able to add and update details about a already added course,
So that the students can get the most up to date information.

1.2.4 As an IITM Management Member

→ **As an IITM Management Member,**

I want to access the individual course feedback received from the students,
So that we can make informed improvements and enhancements to the course structure.

→ **As an IITM Management Member,**

I want a Statistical Summary of the student data,
So that we can efficiently manage and plan the program based on statistical information and performance trends.

2 User Interfaces

2.1 Storyboards

Based on the user stories, 4 boards have been identified to summarise functionality of the app:

1. Student Storyboard ([Figure 1](#))
2. Admin Storyboard ([Figure 2](#))
3. Course Team Member Storyboard ([Figure 3](#))
4. IITM Management Storyboard ([Figure 4](#))

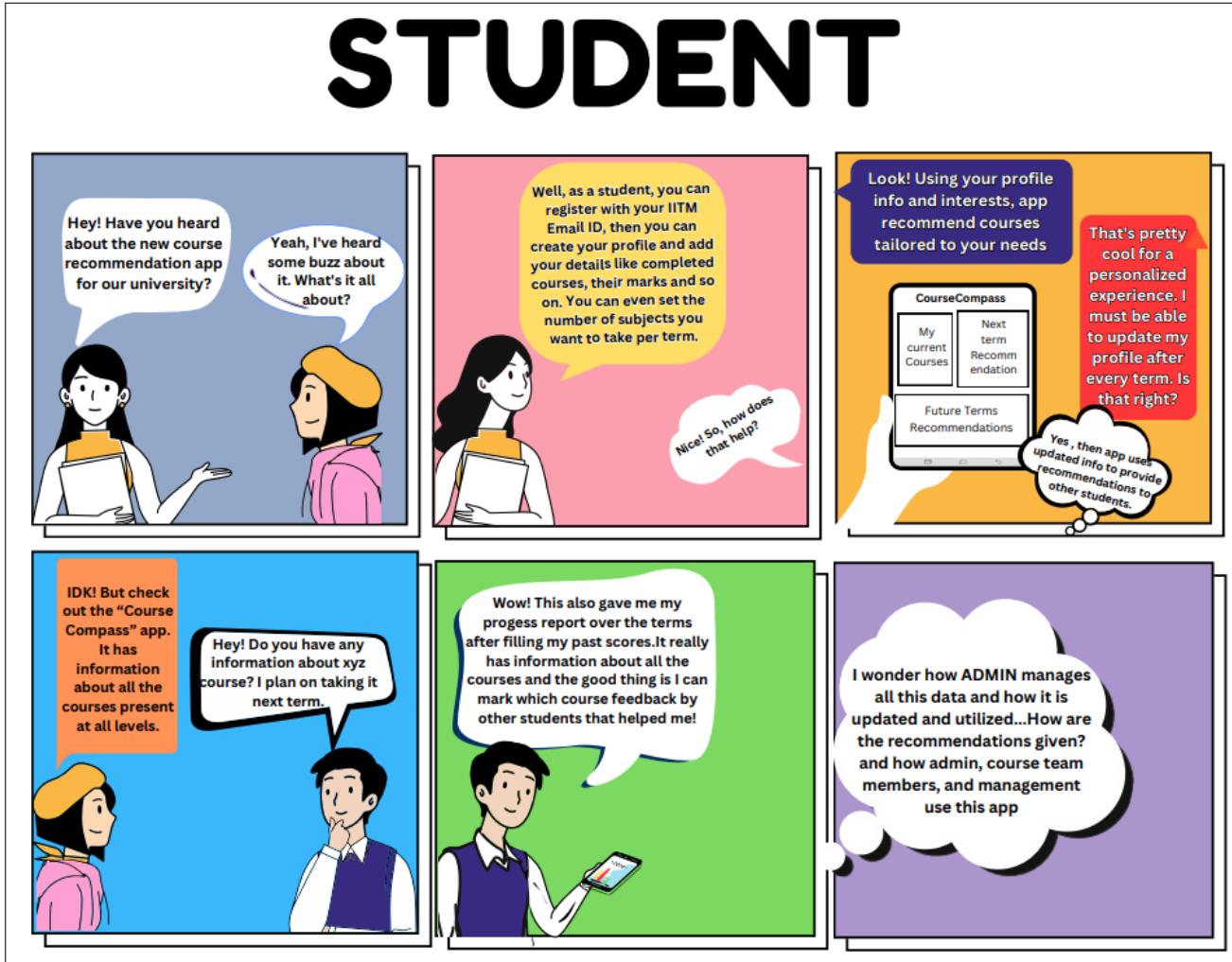


Figure 1: Student Storyboard

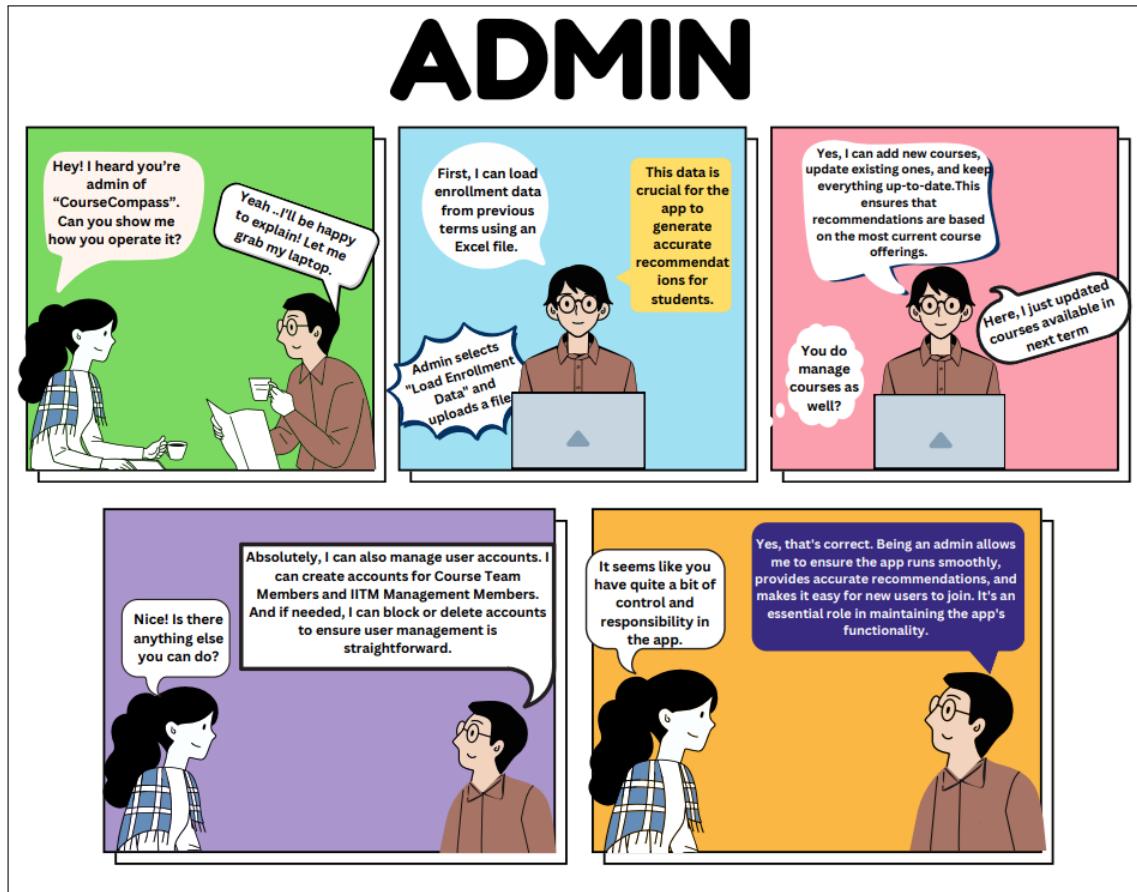


Figure 2: Admin Storyboard

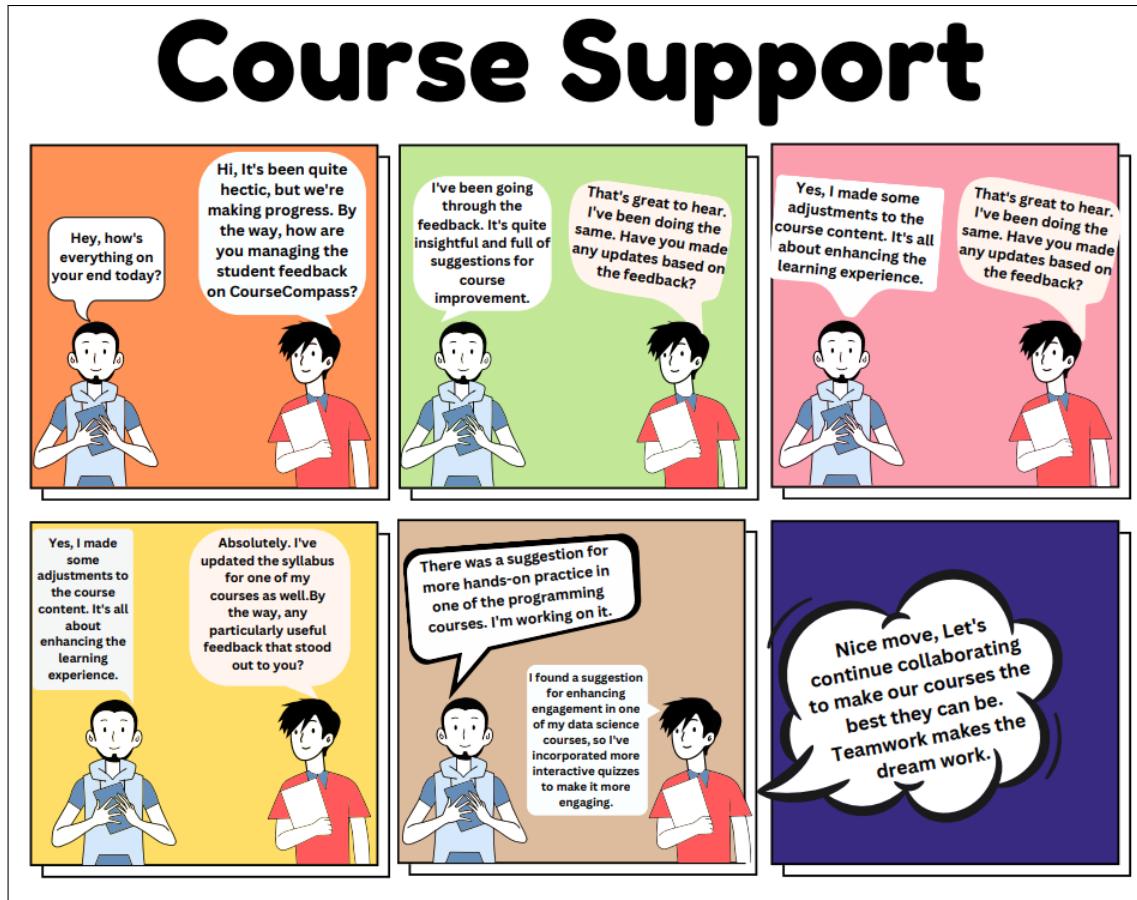


Figure 3: Course Team Member Storyboard

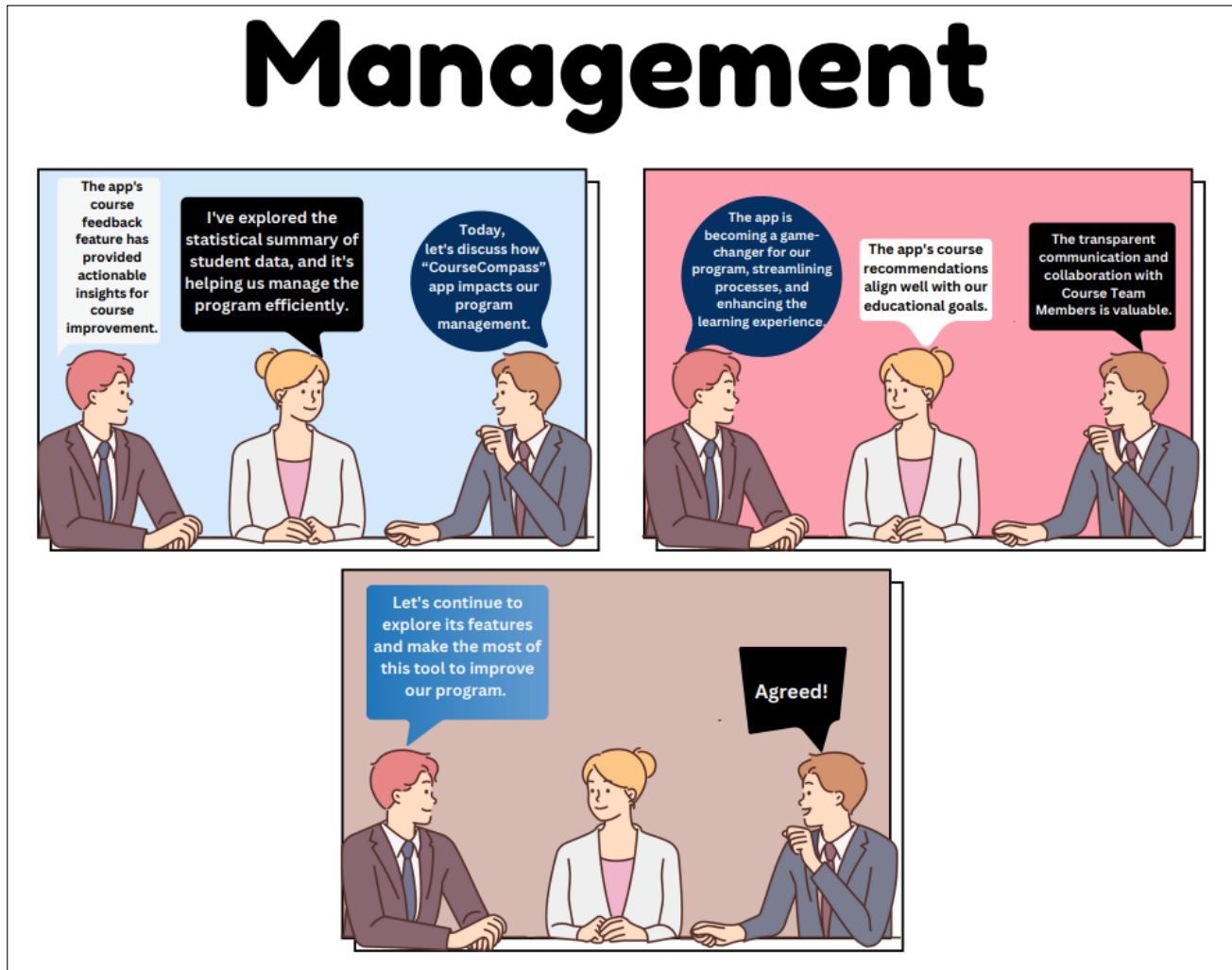


Figure 4: IITM Management Storyboard

2.2 Wireframes

Based on these storyboards and user stories, the following unique pages have been identified to be created:

1. Login/Registration Page ([Figure 5](#))
2. Student's Homepage ([Figure 6](#))
3. Student's Profile page ([Figure 7](#))
4. A Page for Students to view all the courses ([Figure 8](#))
5. A page for Student to view individual course its feedback ([Figure 9](#))
6. Administrator dashboard ([Figure 10](#))
7. A page enlisting all courses to the Administrator ([Figure 11](#))
8. A page to add/edit courses for the Administrator ([Figure 12](#))
9. A page for administrator that enlists course team members, IITM management members and other admins ([Figure 13](#))
10. A page for Adding / Editing course team members, IITM management members and other admins ([Figure 14](#))
11. Course team members' Dashboard ([Figure 15](#))
12. A page for Course Team Members to view feedback given by students ([Figure 16](#))
13. A page for Course team members for adding / updating course description ([Figure 17](#))
14. IITM Management Dashboard (similar to [Figure 10](#))
15. IITM Management Course Feedback page ([Figure 18](#))

2.2.1 General Wireframes

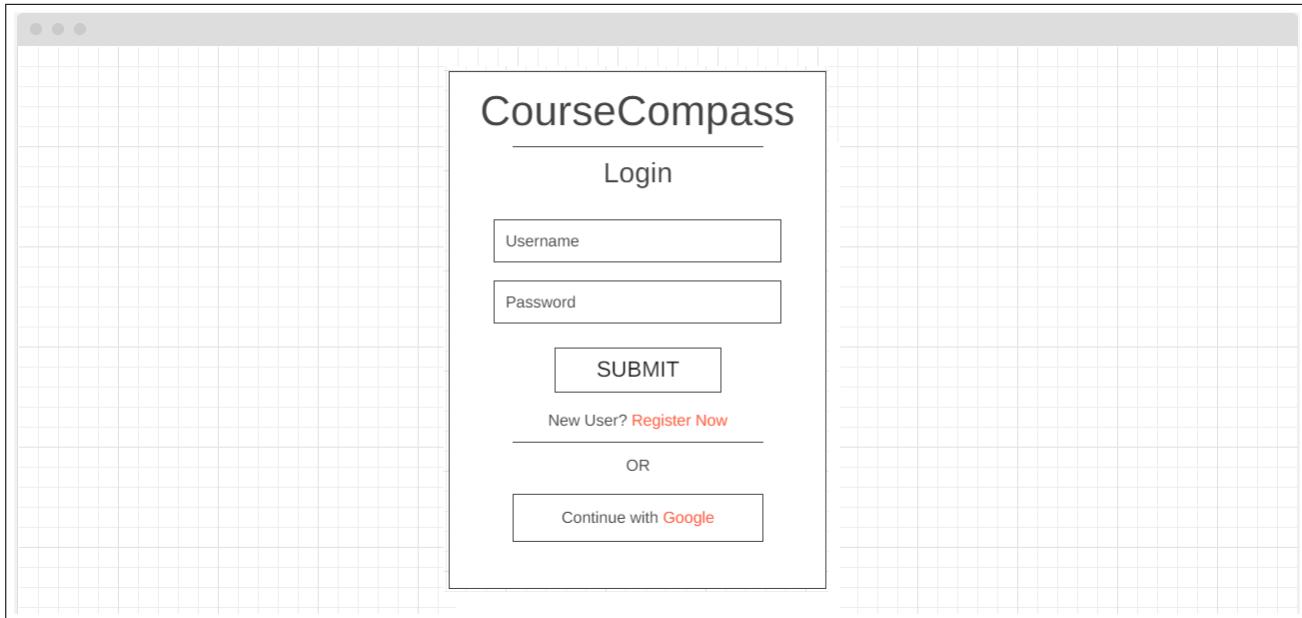


Figure 5: Login Page (Registration page will also look similar)

2.2.2 Students Pages' Wireframes

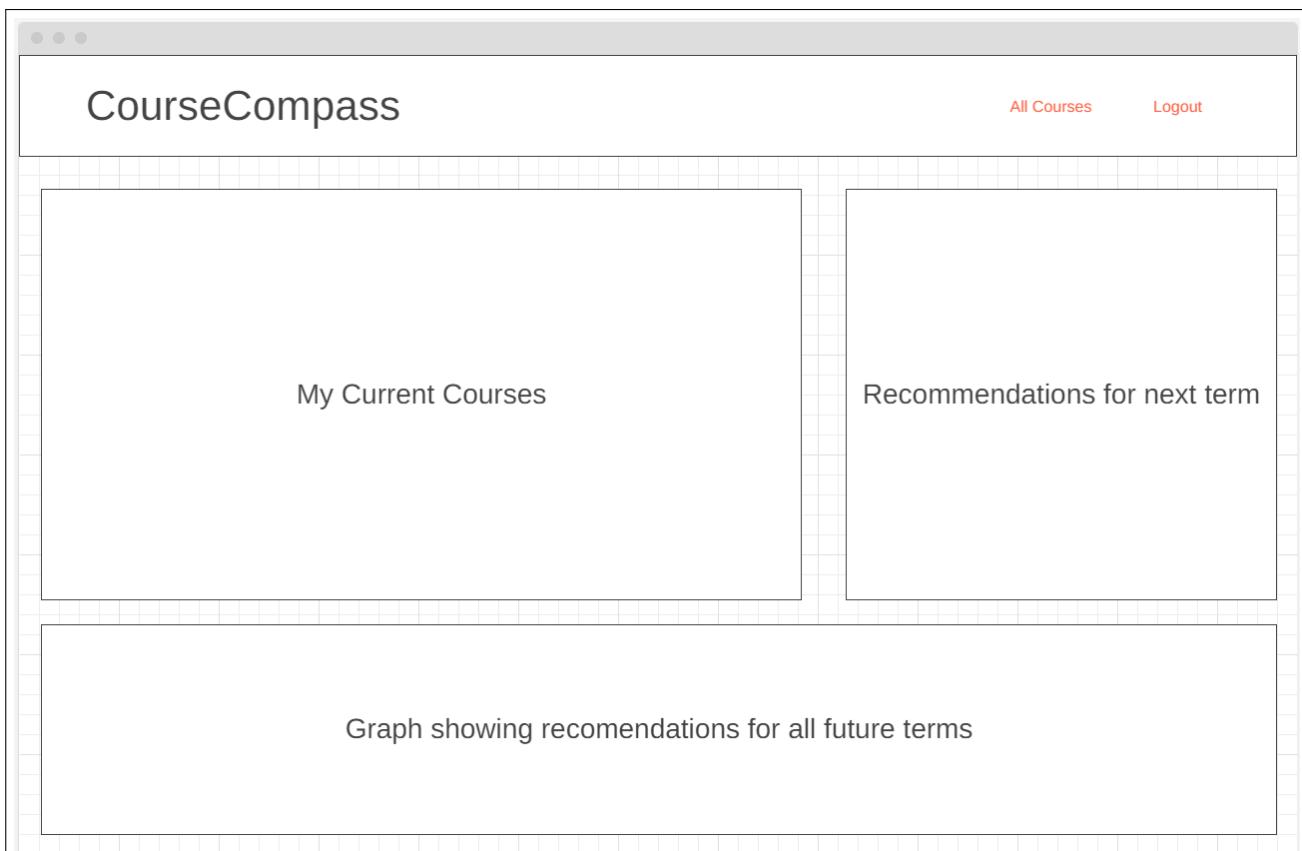


Figure 6: Student's homepage

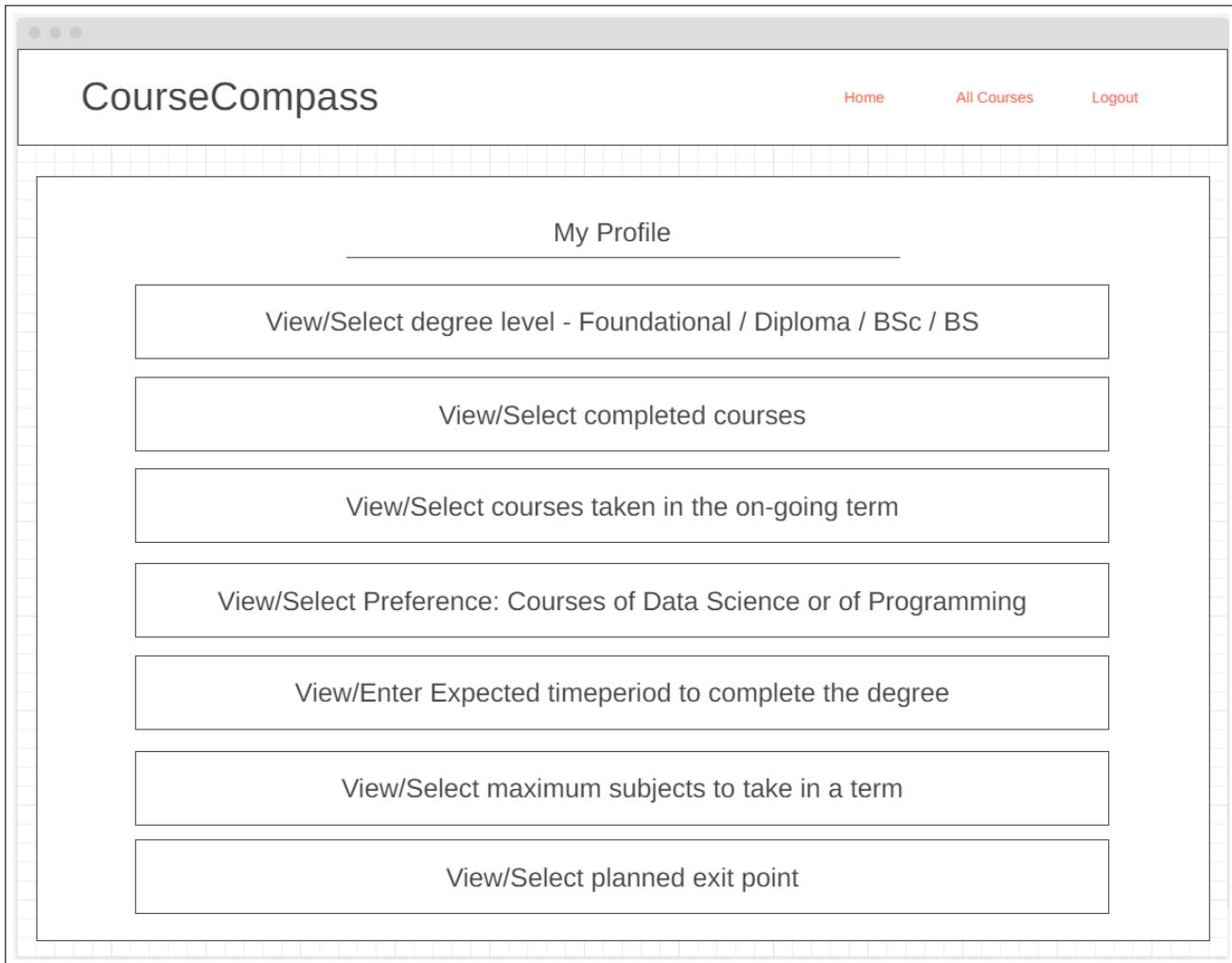


Figure 7: Student's profile page

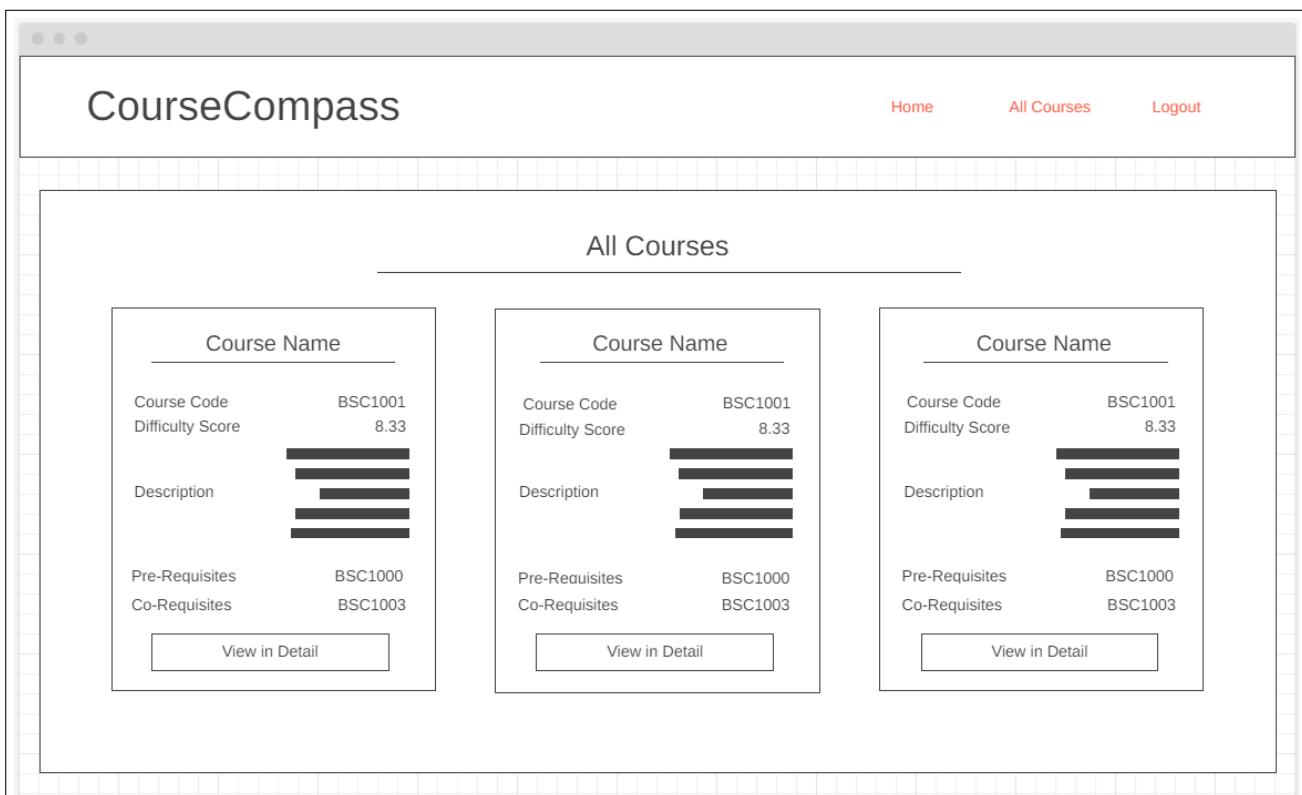


Figure 8: Page for Students to view all the courses

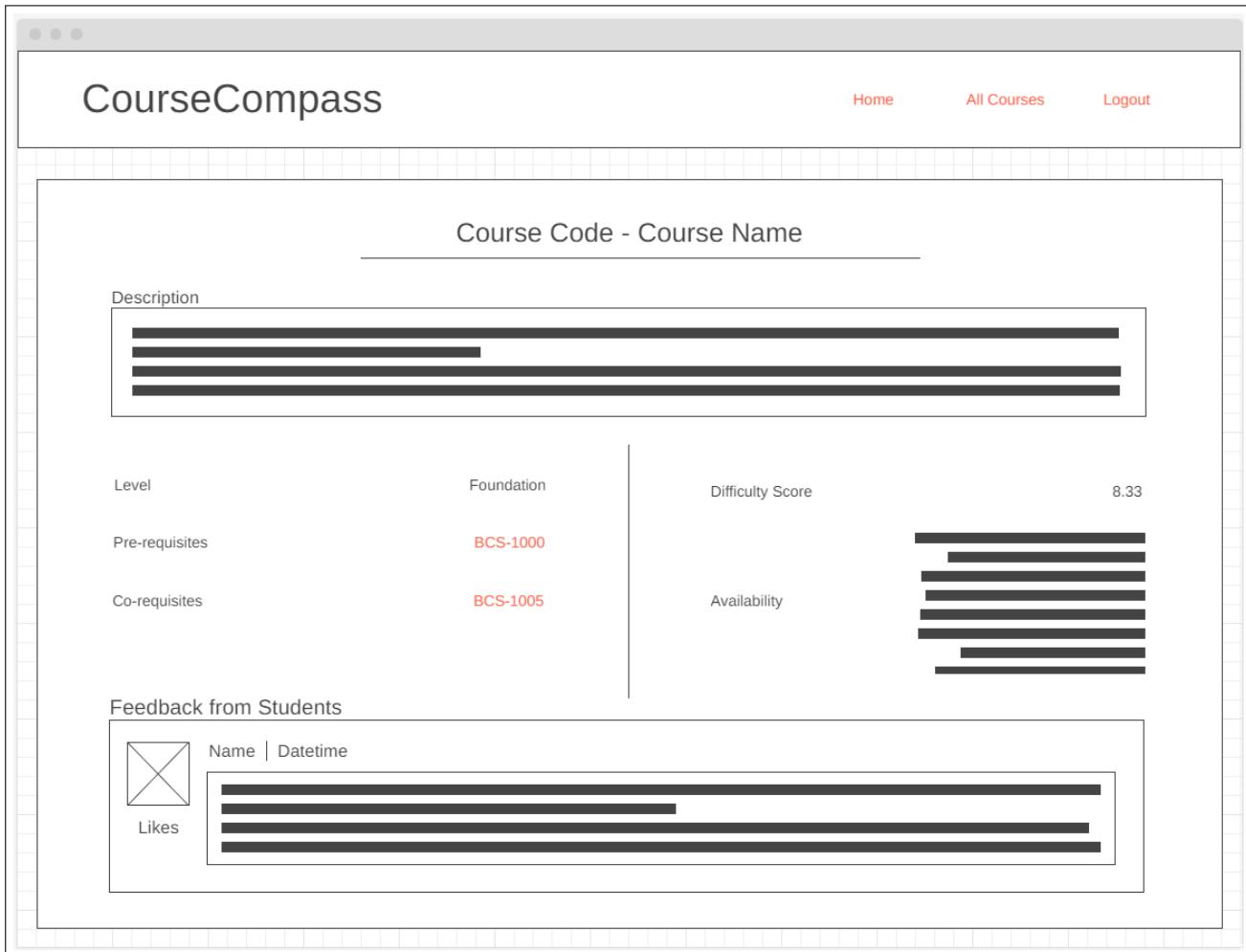


Figure 9: Student's courses and feedback page

2.2.3 Administrators Pages' Wireframes

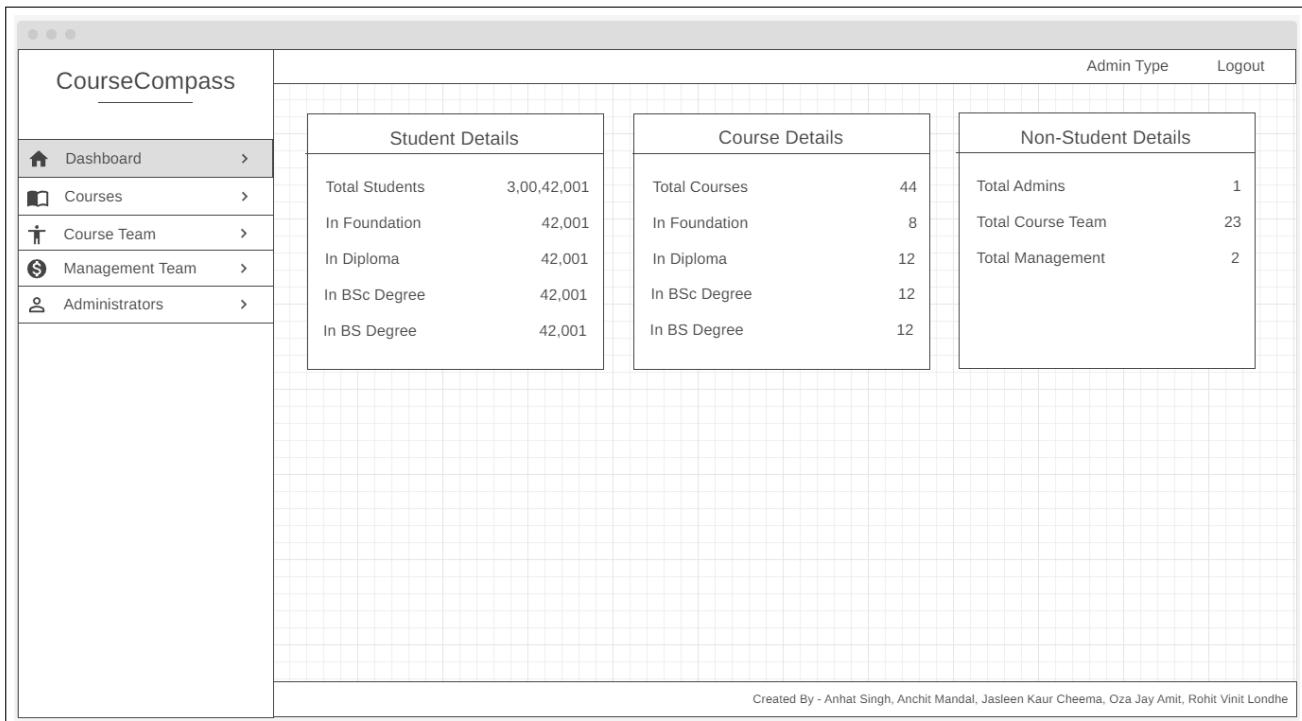


Figure 10: Administrator dashboard

The screenshot shows the 'CourseCompass' application interface for administrators. The left sidebar contains navigation links: Dashboard, Courses (selected), Course Team, Management Team, and Administrators. The main content area has tabs for 'Foundation Courses' and 'Diploma Courses'. Each tab displays a table with columns for Course Code, Course Name, Difficulty Score, and Action. Buttons for 'Add New Course' and 'Import from XLSX' are at the top. The footer credits the team: 'Created By - Anhat Singh, Anchit Mandal, Jasleen Kaur Cheema, Oza Jay Amit, Rohit Vinit Londhe'.

Figure 11: Administrator all courses page

The screenshot shows the 'CourseCompass' application interface for adding or editing a course. The left sidebar shows the 'Courses' link is selected. The main content area is titled 'Add / Edit Course: Course Name'. It includes fields for 'Course Name' (with a placeholder), 'Availability' (checkboxes for September 2023 and January 2024), and 'Course Team Member' (a dropdown menu). The footer credits the team: 'Created By - Anhat Singh, Anchit Mandal, Jasleen Kaur Cheema, Oza Jay Amit, Rohit Vinit Londhe'.

Figure 12: Administrator add / edit courses page

The screenshot shows the 'CourseCompass' application interface for administrators. The left sidebar contains navigation links: Dashboard, Courses, Course Team (selected), Management Team, and Administrators. The main content area is titled 'Course Team' and displays two rows of member information. Each row has columns for ID, Name, Course Code - Course Name, and Action (with a red link). Below this is a button 'Add New Course Team Member'. The next section is 'Management Team', showing two rows with similar columns and an 'Add New Management Member' button. The final section is 'Administrators', also showing two rows with columns for ID, Name, and Action, and an 'Add New Admin' button. The top right corner shows 'Administrator' and 'Logout'. A footer at the bottom right credits the team: 'Created By - Anhat Singh, Anchit Mandal, Jasleen Kaur Cheema, Oza Jay Amit, Rohit Vinit Londhe'.

Figure 13: Administrator Page to view course team members, IITM management members and other admins

The screenshot shows the 'CourseCompass' application interface for adding or editing administrators. The left sidebar contains navigation links: Dashboard, Courses, Course Team, Management Team, and Administrators (selected). The main content area is titled 'Add / Edit Admins' and contains fields for 'Email' and 'Type' (a dropdown menu). The top right corner shows 'Administrator' and 'Logout'. A footer at the bottom right credits the team: 'Created By - Anhat Singh, Anchit Mandal, Jasleen Kaur Cheema, Oza Jay Amit, Rohit Vinit Londhe'.

Figure 14: Adding / Editing course team members, IITM management members and other admins

2.2.4 Course Team Member Pages' Wireframes

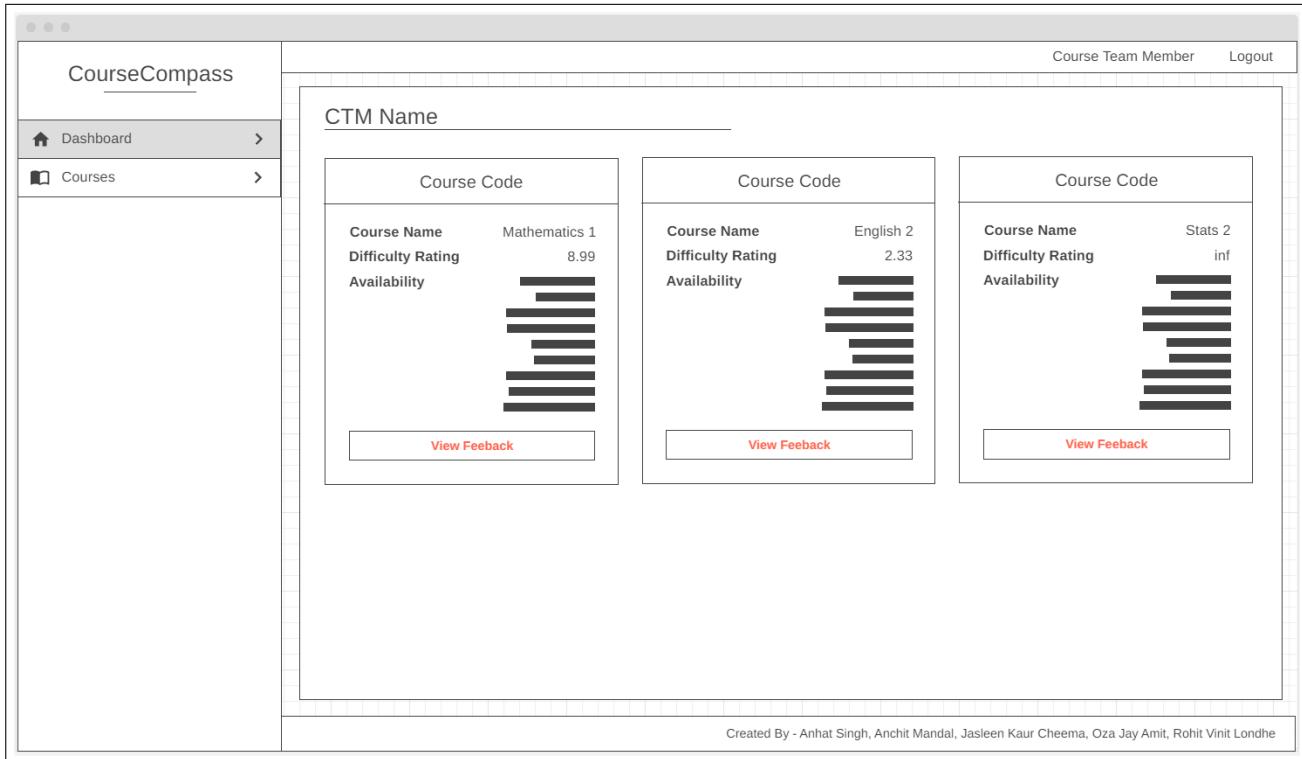


Figure 15: Course team members' Dashboard

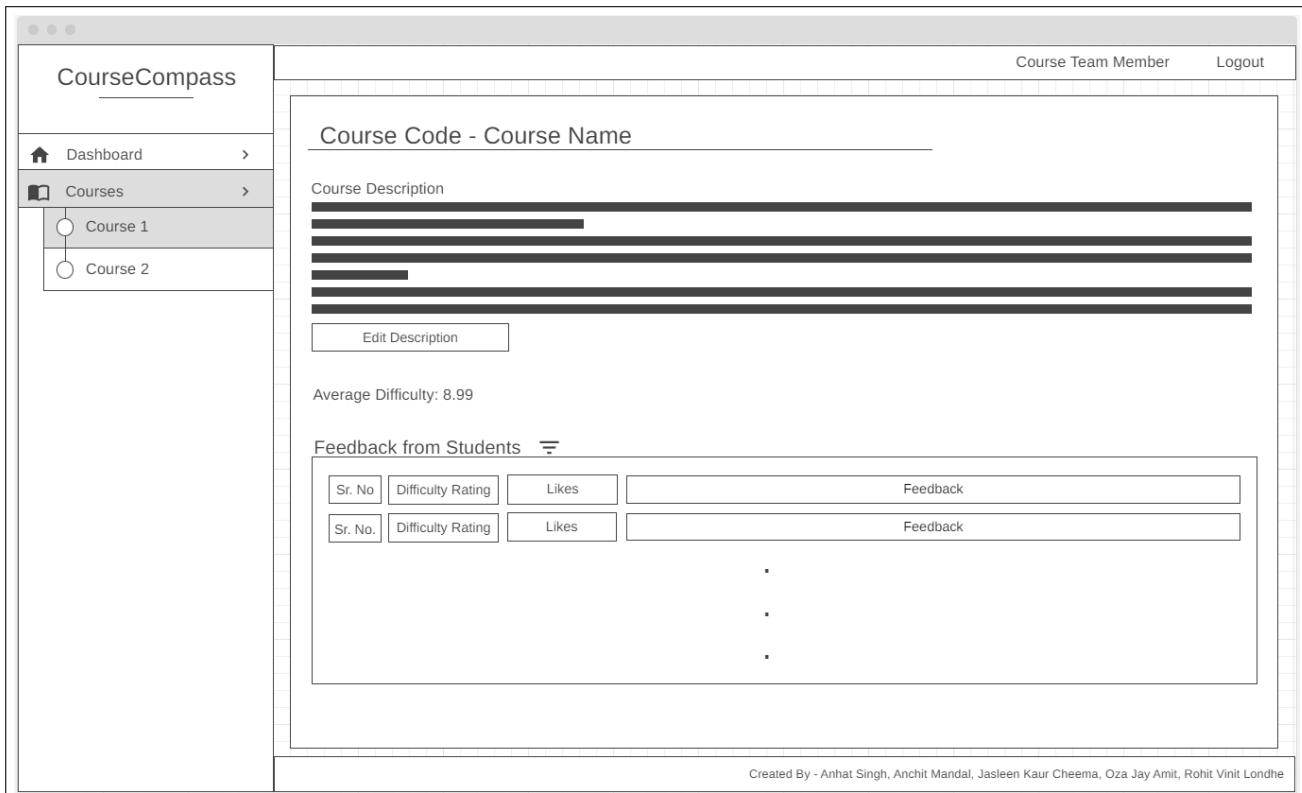


Figure 16: A page for Course Team Members to view feedback given by students

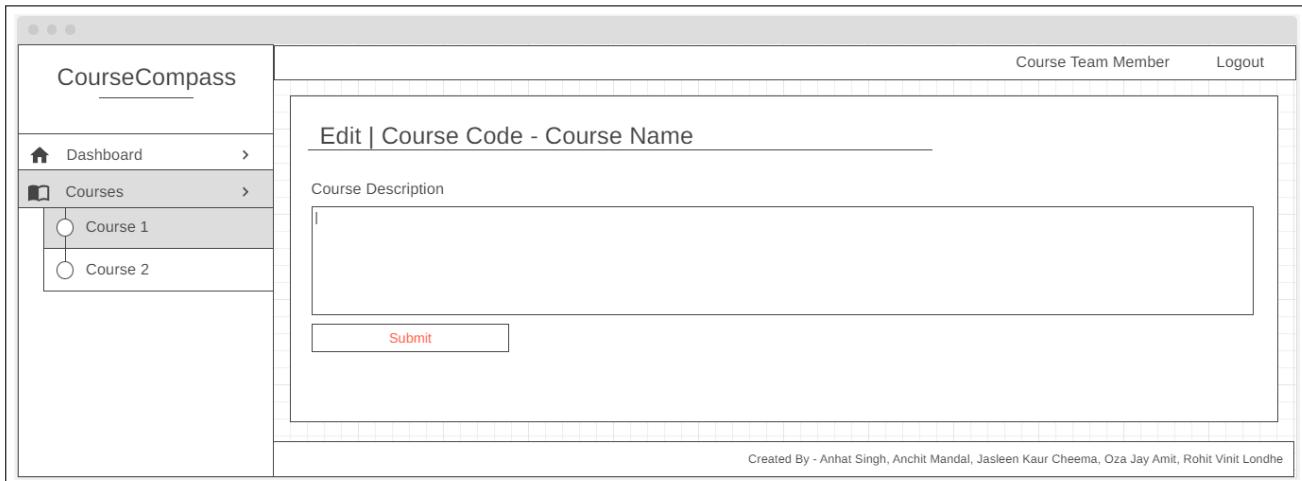


Figure 17: A page for Course team members for adding / updating course description

2.2.5 IITM Management Pages' Wireframes

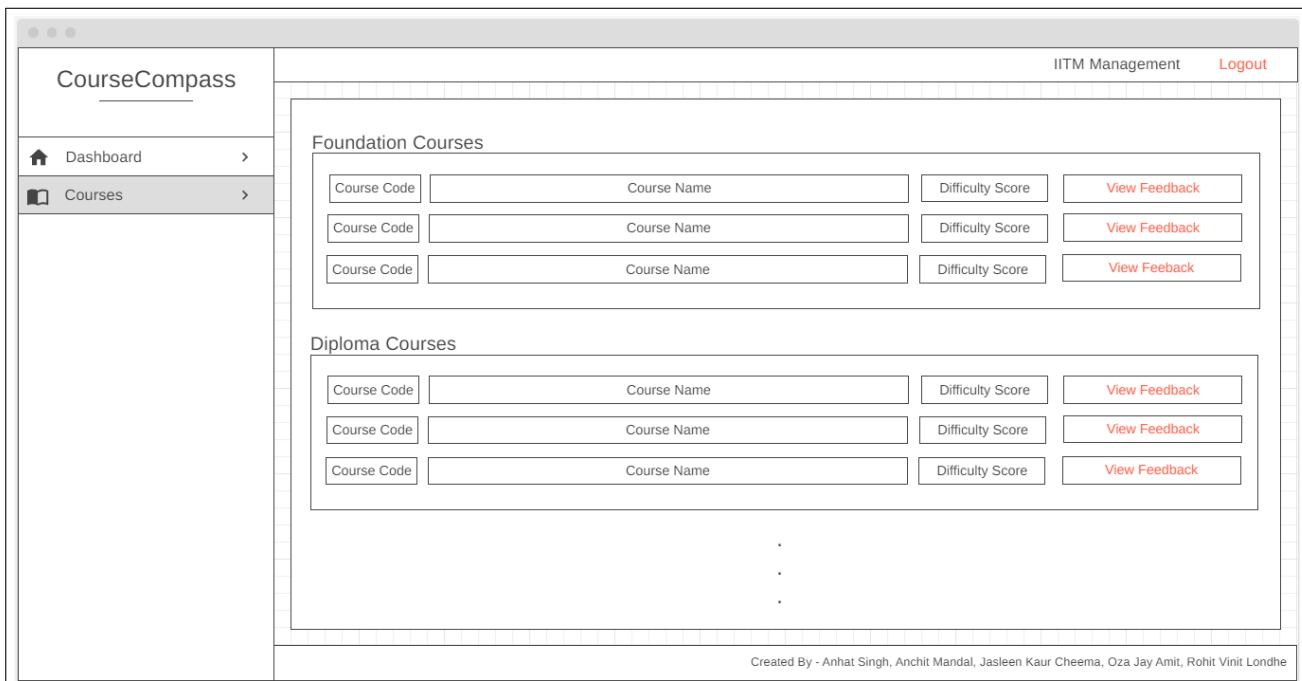


Figure 18: IITM Management Course Feedback page

2.3 Usability Design Guidelines & Heuristics

The low-fidelity wireframes have been designed with common Design guidelines (Effectiveness, Efficiency, Safety, Learnability, Memorability) ensuring that it is obvious what each page displays and easy to determine the flow of information. Here are some specific recommendations based on these principles followed while creating the wireframes:

1. Registration and Profile Creation

- **Efficiency:** Make the registration process straightforward and efficient. Use email verification for account creation.
- **Memorability:** Ensure that users can easily update their profiles after registration. Use a clear and accessible interface.

2. Interest Selection

- **Effectiveness:** Make it easy for students to select their primary interests. Provide clear options for Data Science, Programming, or Both.

- **Memorability:** Allow users to change their primary interests later if needed.

3. Recommendations

- **Effectiveness:** Display recommendations based on a student's profile clearly and prominently.

4. Course Information

- **Effectiveness:** Provide detailed course information, including prerequisites, feedback, and difficulty ratings.
- **Memorability:** Use a consistent layout for displaying course details and feedback.

5. Admin Features

- **Efficiency:** Admin features should be accessible from a centralized dashboard. Ensure quick data loading from Excel files and easy course management.
- **Safety:** Implement authentication and authorization for admin functions to prevent unauthorized access or modifications.

6. Course Team and Management Access

- **Effectiveness:** Ensure that Course Team Members and IITM Management can access student feedback and statistics efficiently.
- **Usability:** Provide user-friendly interfaces for adding, updating, and viewing course details and feedback.

7. Statistical Summaries

- **Effectiveness:** Display academic and student data summaries in a clear and informative manner for both students and management.
- **Memorability:** Use consistent visual representations for statistics.

8. Feedback Interaction

- **Efficiency:** Allow students, Course Team Members, and IITM Management to interact with feedback efficiently, such as +1 (upvoting) helpful feedback.
- **Usability:** Implement clear feedback forms and voting mechanisms.

9. Consistency

- **Usability:** Maintain a consistent design throughout the application to ensure users can easily navigate and understand the interface.

10. User Assistance and Help

- **Usability:** Provide tooltips, help sections, or a knowledge base to assist users in understanding how to use the software effectively and safely.

11. Error Handling

- **Safety:** Implement user-friendly error messages and validation checks to prevent user errors and ensure safe operation.

12. Responsive Design

- **Usability:** Ensure that the UI is responsive and works well on various devices and screen sizes to cater to a broader user base.

13. Testing and Feedback

- **Effectiveness:** Regularly test the UI with actual users and collect their feedback to make continuous improvements in line with user needs and preferences.

3 Scheduling and Design

3.1 Project Schedule

3.1.1 Gantt Chart

After a lot of discussion, a general schedule mentioned in Figure 19 has been decided. The breakdown of individual milestone in Figure 19 is given in Figure 20, Figure 21 and Figure 22. Details about each task is given in Table 1.

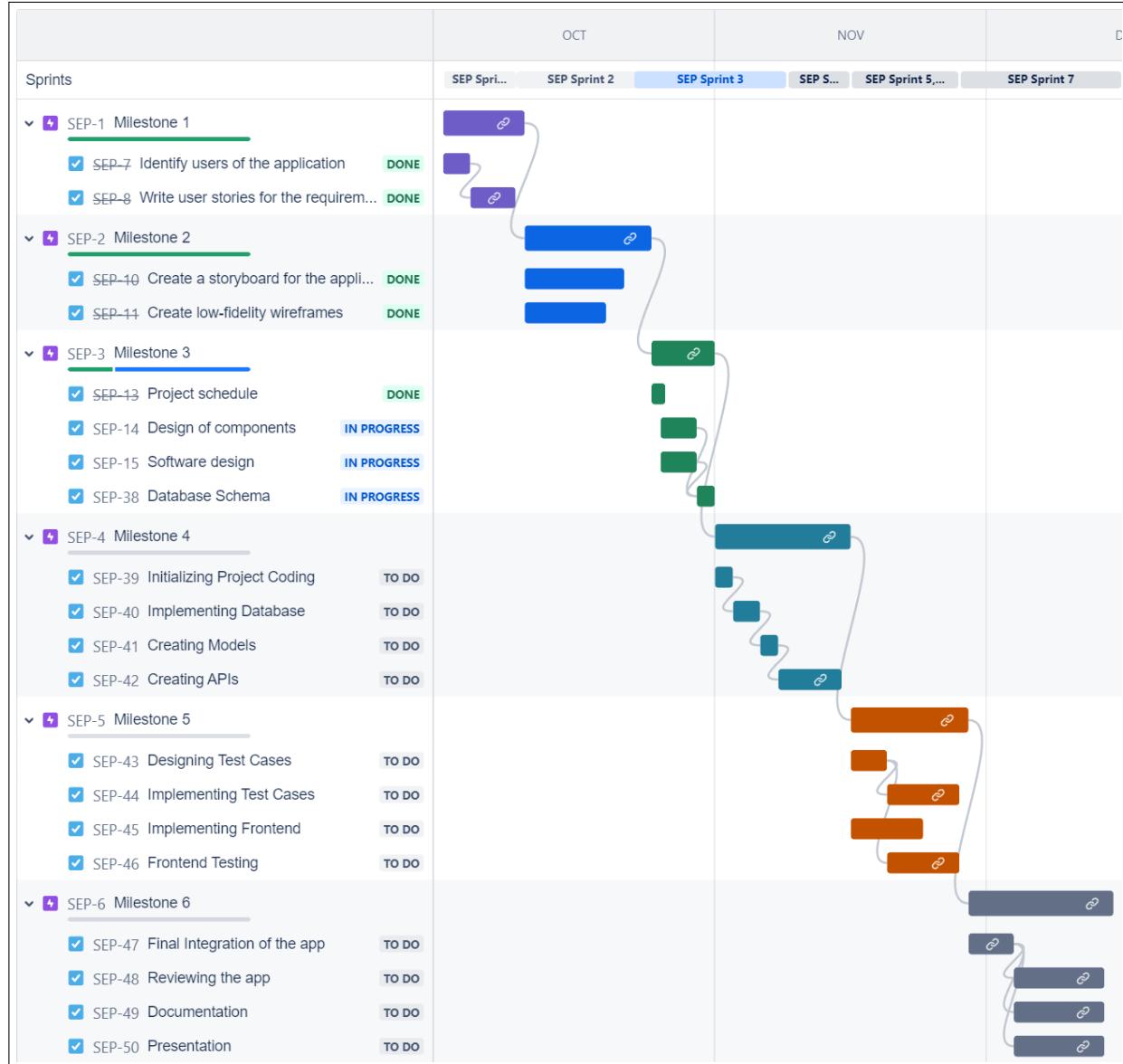


Figure 19: Complete Project Timeline

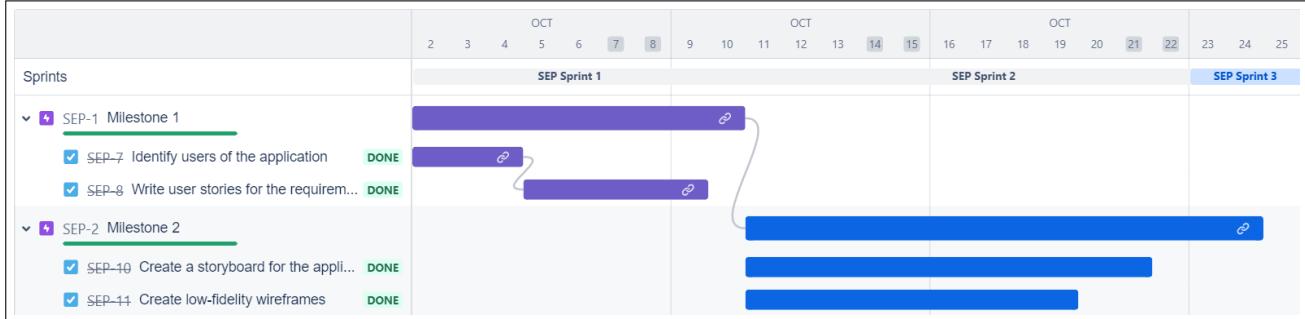


Figure 20: Milestone 1 and 2 Schedule

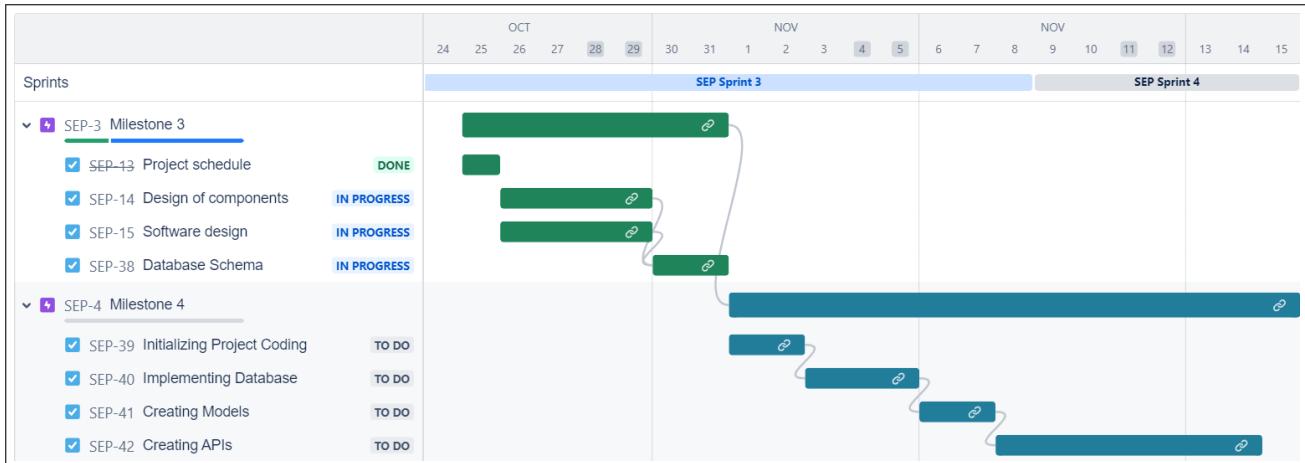


Figure 21: Milestone 3 and 4 Schedule

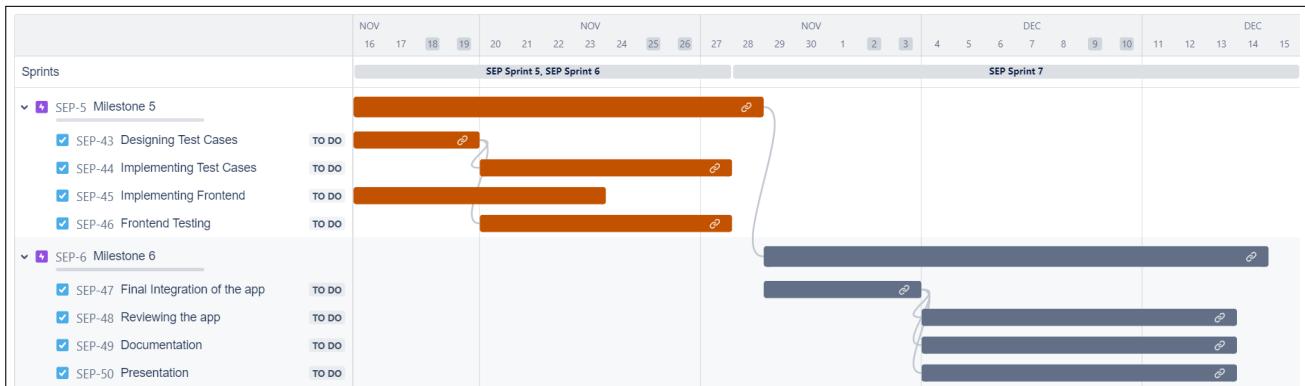


Figure 22: Milestone 5 and 6 Schedule

As evident from the figures, each task in the milestones is self-explanatory and need not be further described.

The tasks above have divided into various sprints, details of which are in [Table 1](#).

3.1.2 SPRINT Schedule

Activity	Start Date (dd/mm/yyyy)	End Date (dd/m-m/yyyy)	Duration (days)
Sprint 1			
Identify users of the application	02/10/2023	04/10/2023	3
Write user stories	05/10/2023	09/10/2023	5
Sprint 2			
Create storyboards	11/10/2023	21/10/2023	11
Create low-fidelity wireframes	11/10/2023	19/10/2023	9
Sprint 3			
Project schedule	25/10/2023	25/10/2023	1
Design of components	26/10/2023	29/10/2023	4
Software design	26/10/2023	29/10/2023	4
Database Schema	30/10/2023	31/10/2023	2
Initializing Project Coding	01/11/2023	02/11/2023	2
Implementing Database	03/11/2023	05/11/2023	3
Creating Models	06/11/2023	07/11/2023	2
Sprint 4			
Creating APIs	08/11/2023	14/11/2023	7
Sprint 5			
Designing Test Cases	16/11/2023	19/11/2023	4
Implementing Test Cases	20/11/2023	27/11/2023	8
Sprint 6			
Implementing Frontend	16/11/2023	23/11/2023	8
Frontend Testing	20/11/2023	27/11/2023	8
Sprint 7			
Final Integration of the app	29/11/2023	03/12/2023	5
Reviewing the app	04/12/2023	13/12/2023	10
Documentation	04/12/2023	13/12/2023	10
Presentation	04/12/2023	13/12/2023	10

Table 1: Sprint schedule created in Jira

Various Sprints have been created to complete the tasks. Each Sprint has Analysis, Design and Implementation as its sub-phases may it be for drawing (Storyboards) or coding (API coding). Also, each task in the sprints is self-explanatory and need not be further described.

3.2 Project Scheduling Tool

After checking out various tools, **Jira Software by Atlassian** was decided to be used based on its in-built features of sprints, epics, connection to github and task management features. A screenshot showing the usage of the app by the team is in [Figure 23](#).

The screenshot shows the Jira Software interface for a project titled "Software Engineering - Software project". The current sprint is "SEP Sprint 3". The left sidebar includes links for "Board", "Timeline", "Backlog", "Reports", "Issues", "Add view", "Code", and "Project pages". The main area displays a Kanban-style board with three columns: "TO DO 3", "IN PROGRESS 3", and "DONE 1". Each column contains tasks grouped by milestone. For example, the "TO DO 3" column has tasks like "Initializing Project Coding" and "Implementing Database", both under "MILESTONE 4". The "IN PROGRESS 3" column has tasks like "Design of components" and "Software design", both under "MILESTONE 3". The "DONE 1" column has a "Project schedule" section with a task "Database Schema" under "MILESTONE 3". The top right of the screen shows a search bar, a notification icon, and other navigation options.

Figure 23: Sprint Board in Jira

3.3 Various Components

3.3.1 User Management Component

- Handles user authentication, registration, and profile creation.
- Assigns and manages user roles based on permissions.
- Provides user access control and permission management.

3.3.2 Profile Management Component

- Stores and retrieves user profile information securely.
- Manages user preferences, interests, and contact details.
- Enables profile updates and modifications by users.
- Ensures privacy and data protection measures for user profiles.
- Supports user-initiated data deletion or modification requests.

3.3.3 Course Management Component

- Adds, edits, and deletes courses from the catalog database.
- Manages course details such as name, description, instructors, and prerequisites.
- Controls the availability of courses for different terms or semesters.
- Handles updates and modifications to course information.
- Ensures data integrity and consistency within the course database.

3.3.4 Enrolment and Analytics Component

- Imports and exports enrolment data for analysis and recommendations.
- Generates statistical reports and insights for academic trends and performance.
- Tracks course popularity and student enrolment statistics.

3.3.5 Feedback and Rating Component

- Stores and manages student feedback and ratings for courses.
- Allows students to provide, edit or delete their feedback.
- Enables upvoting of helpful feedback from other students.
- Monitors and maintains the accuracy and relevance of feedback data.
- Facilitates analysis of feedback for course improvements.

3.3.6 Course Catalog Component

- Manages detailed information about available courses.
- Allows for sorting and filtering capabilities for users.
- Tracks and displays course availability and scheduling information.
- Maintains a comprehensive database of instructor details and feedback for each course.

3.3.7 Data Analysis Component

- Collects and processes statistical data for student performance.
- Conducts trend analysis for course popularity and academic patterns.
- Generates graphical representations and reports for data visualisation.
- Offers insights into academic progress and program assessment.

3.3.8 Recommendation Component

- Generates personalised course recommendations based on user profiles, completed courses, and interests.
- Utilises machine learning algorithms or rule-based systems for suggestion generation.
- Analyses user behaviour and preferences to suggest relevant courses.
- Collaborates with Enrolment and Analytics components to enhance recommendation accuracy.

3.4 Class Diagram

A basic class diagram describing the application is given in [Figure 24](#). The diagram follows standard UML notations. The return type and parameters have been left empty for the classes, for the sake of readability.

There are 5 major Classes:

1. User
2. Recommender
3. Feedback
4. Course
5. Analytics

with their specializations as per given in the diagram. The dotted notation represents dependency of one component on another. This diagram shall be further modified to be in-line with flask framework to facilitate easy creation of APIs.

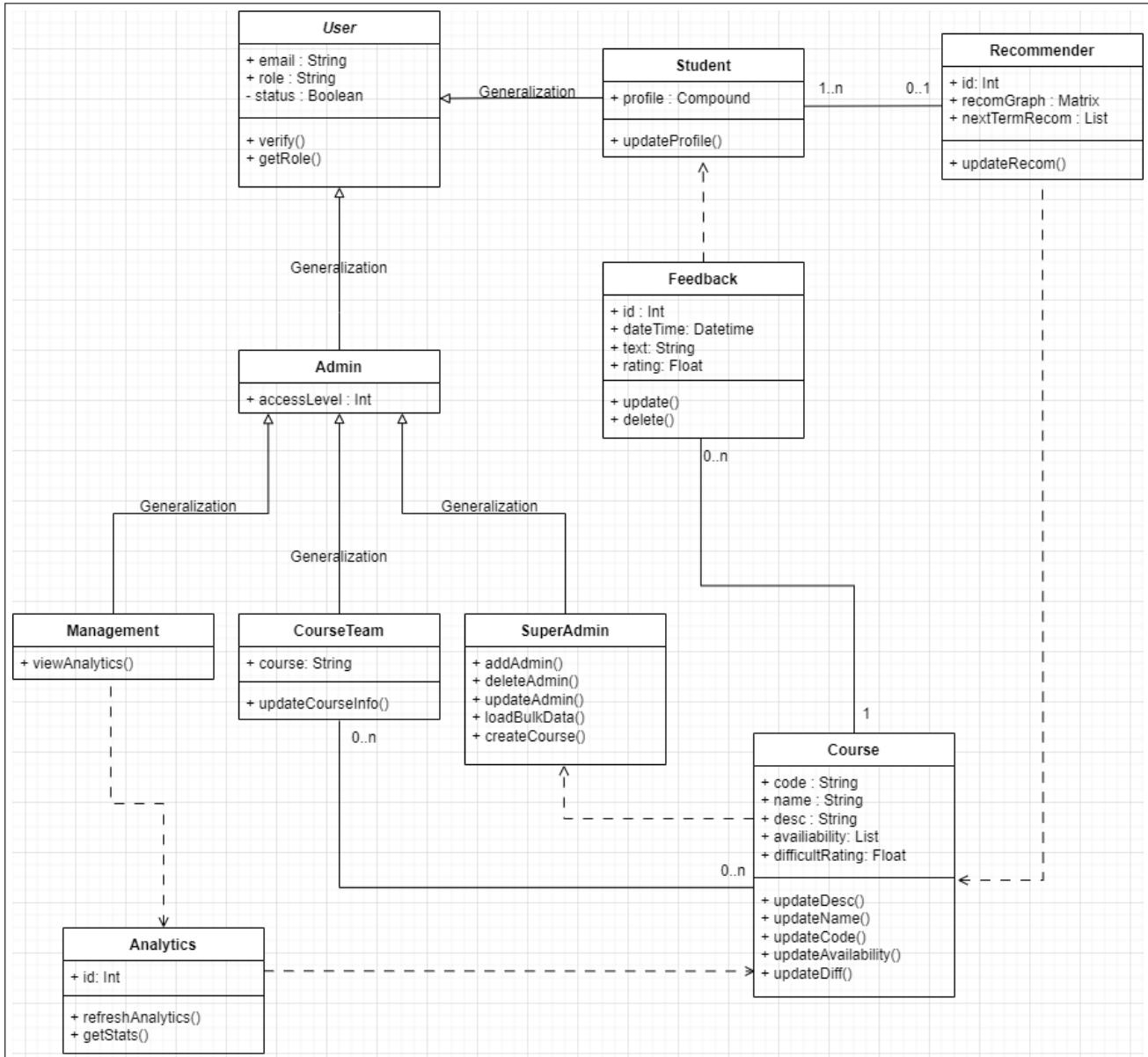


Figure 24: Basic Class Diagram

3.5 SCRUM Meetings Schedule and Minutes

Till date, 6 official meetings have been conducted details of which are in subsequent boxes. A lot of discussion also held via chat through WhatsApp. The major work completed, issues discussed and agenda is combined into a single heading "Minutes of the Meeting".

Meeting: Based on availability and need, after every 6 days starting from 2nd October.

2nd October, 2023

- **Time:** 17:00 to 17:50 Hours
- **Duration:** 50 minutes
- **Who Attended:** Anchit, Anhat, Jasleen, Jay, Rohit
- **Agenda:** Introduction
- **Minutes of the Meeting:** The main goal of the meeting was to introduce ourselves and to discuss basic approach to solve the given Problem Statement.
 1. Introduced Ourselves
 2. Understood which tech-stack we are comfortable
 3. Identified main components in the projects:
 - (a) Backend and Frontend
 - (b) API
 - (c) UI
 - (d) Recommendation Algorithms
 4. Understood and managed the timelines on Jira

8th October, 2023

- **Time:** 19:00 to 19:29 Hours
- **Duration:** 29 minutes
- **Who Attended:** Anchit, Anhat, Jasleen, Jay, Rohit
- **Agenda:** Complete Sprint 1
- **Minutes of the Meeting:**
 1. Identified Primary, Seconday and Tertiary Users
 2. Understood SMART Guidelines
 3. Understood and wrote User Stories

15th October, 2023

- **Time:** 16:45 to 17:15 Hours
- **Duration:** 30 minutes
- **Who Attended:** Anchit, Anhat, Jasleen, Jay, Rohit
- **Agenda:** Initialize discussion on Sprint 2
- **Minutes of the Meeting:**
 1. Wrote the important requirements for creating userboards
 2. Learnt and discussed different ways to create wireframes

18th October, 2023

- **Time:** 20:35 to 21:09 Hours
- **Duration:** 24 minutes
- **Who Attended:** Anchit, Anhat, Jasleen, Jay, Rohit
- **Agenda:** Ensure uniformity in storyboard and wireframe design by each team member.
- **Minutes of the Meeting:**
 1. Initiated the work on creating a storyboard
 2. Created and discussed the essential wireframes

26th October, 2023

- **Time:** 22:15 to 22:47 Hours
- **Duration:** 32 minutes
- **Who Attended:** Anchit, Anhat, Jasleen, Jay, Rohit
- **Agenda:** Complete Sprint 2
- **Minutes of the Meeting:**
 1. Completed all the wireframes
 2. Completed storyboards illustrating all the important features

1st November, 2023

- **Time:** 16:05 to 16:27 Hours
- **Duration:** 22 minutes
- **Who Attended:** Anchit, Anhat, Jasleen, Jay, Rohit
- **Agenda:** Discuss Sprint 3
- **Minutes of the Meeting:**
 1. Created/Rectified Sprints and timings in JIRA
 2. Designed all the components diagrams
 3. Created a basic required class diagram

4 API Endpoints

The API endpoints have been listed in the [Figure 25](#), [Figure 26](#) and [Figure 27](#). API yaml file can be found in the [others](#) folder in the [Github Repository](#).

User Operations		
GET	/user/auth	User logout
POST	/user/auth	User login
POST	/user/auth/register	Create new user
Profile		
GET	/profile	Get user preferences
POST	/profile	Add user preferences
PATCH	/profile	Modify preferences
Courses		
GET	/course/{id}	Get single course info
PATCH	/course/{id}	Modify Single Course Info
DELETE	/course/{id}	Delete a Course
GET	/courses	Get all courses info
POST	/courses	Add a Course
POST	/courses/bulkadd	Bulk Add Courses

Figure 25: API Endpoints List 1

Feedback		
GET	/course/{courseID}/feedback	Get Course Feedback
POST	/course/{courseID}/feedback	Add Feedback to course
GET	/feedback/{FeedbackID}	Upvote Downvote feedback
DELETE	/feedback/{FeedbackID}	Delete a Feedback
Recommender		
GET	/recommendation	Get Recommendations
Admin		
GET	/admin	List of Admins
POST	/admin	Add admins
PATCH	/admin	Modify details of an admin
DELETE	/admin	Delete an admin
Statistics		
GET	/stats/courses	Statistics about individual courses
GET	/stats/general	General Statistics about the app

Figure 26: API Endpoints List 2

default		
GET	/student/all	Your GET endpoint

Figure 27: API Endpoints List 3

5 Test Cases for API Endpoints

5.1 A pic of implemented test case

```

1 import pytest
2 import requests, json
3
4 BASE_URL = "http://localhost/api/v1"
5 token = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiWF0IjoxNTE2MjMSMDIyfQ.SflKxwRJSMeKKF2QT4f
6
7 # Test Case 1
8 def test_profile_endpoint_correct_info_present():
9     response = requests.get(f"{BASE_URL}/profile", headers={'Accept': 'application/json', 'Authorization': f"Bearer {token}"})
10    assert response.status_code == 200
11
12 # Test Case 2
13 def test_profile_endpoint_auth_token_missing():
14     response = requests.get(f"{BASE_URL}/profile", headers={'Accept': 'application/json'})
15     assert response.status_code == 401
16
17 # Test Case 3
18 def test_profile_endpoint_all_details_correct_and_present():
19     input_dict={"pic": "ABCD.png", "maximum_courses_in_a_term": 3, "current_degree_level": "diploma", "dp_or_ds": "both", "current_courses": ["B
20     data = json.dumps(input_dict)
21     response = requests.post(f"{BASE_URL}/profile", data=data, headers={'Accept': 'application/json', 'Authorization': f"Bearer {token}"})
22     assert response.status_code == 201

```

Figure 28: Some implemented test cases using Python

Test Case 1

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Auth token missing

Request Type	GET
Expected Response Code	401
Actual Response Code	401

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json'

```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }

```

Test Case 2

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	All information required correct and present
Request Type	GET
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {
2   "email": "abc@xyz.com",
3   "role": "student",
4   "bio": {
5     "maximum_courses_in_a_term": 3,
6     "current_degree_level": "foundation",
7     "dp_or_ds": "ds",
8     "current_courses": [
9       "BSHS1001"
10    ],
11    "completed_courses": [
12      {
13        "id": "BSMA1001",
14        "marks": 70
15      }
16    ]
17  }
18}
```

Test Case 3

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	All details required match and present
Request Type	POST
Expected Response Code	201
Actual Response Code	201
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSC2001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 87.9
18      }
19    ]
20  }'
```

Output Response: application/json

```
1 {
2   "message": "created",
3   "status": "success"
4 }
```

Test Case 4

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	User details do not exist
Request Type	GET
Expected Response Code	404
Actual Response Code	404
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 404
2 {
3   "message": "The requested resource was not found on this server",
4   "status": "error"
5 }
```

Test Case 5

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Wrong type of user logged in

Request Type	GET
Expected Response Code	403
Actual Response Code	403

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 6

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Auth token missing
Request Type	POST
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "pic": "ABCDE.png",
7     "maximum_courses_in_a_term": 4,
8     "current_degree_level": "diploma",
9     "dp_or_ds": "both",
10    "current_courses": [
11      "BShS1001"
12    ],
13    "completed_courses": [
14      {
15        "id": "BSC2001",
16        "marks": 92
17      }
18    ]
19  }'
```

Output Response: application/json

```
1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 7

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Wrong type of user logged in
Request Type	POST
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 4,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSHS1001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 90
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }

```

Test Case 8

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	User Auth does not exist
Request Type	POST
Expected Response Code	404
Actual Response Code	404
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCDE.png",
8     "maximum_courses_in_a_term": 2,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BShS1001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 90
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "The requested resource was not found on this server",
3   "status": "error"
4 }

```

Test Case 9

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Course code uploaded in the 'current_course' field does not exist in database
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCDE.png",
8     "maximum_courses_in_a_term": 2,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSC2001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 90
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 10

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Course code uploaded in the 'completed_course' field does not exist in database
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSC2006"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2006",
17        "marks": 87.9
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 11

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	value in the 'maximum_courses_in_a_term' field less than 2
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 1,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BShS1001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 87.9
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 12

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	value in the 'maximum_courses_in_a_term' field greater than 4
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 5,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BShS1001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 87.9
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 13

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Invalid value in 'current_degree_level'
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "graduated",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSHS1001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 87.9
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 14

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Invalid value in 'dp_or_ds'
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "dk",
11    "current_courses": [
12      "BShS1001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 87.9
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 15

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Course uploaded in the 'completed_course' field has marks less than 0
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSC2001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": -9
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 16

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Course uploaded in the 'completed_course' field has marks more than 100
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSC2001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001",
17        "marks": 130
18      }
19    ]
20 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 17

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Course uploaded in the 'completed_course' field has marks missing
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "pic": "ABCD.png",
8     "maximum_courses_in_a_term": 3,
9     "current_degree_level": "diploma",
10    "dp_or_ds": "both",
11    "current_courses": [
12      "BSC2001"
13    ],
14    "completed_courses": [
15      {
16        "id": "BSC2001"
17      }
18    ]
19 }'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 18

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	All details required match and present
Request Type	PATCH
Expected Response Code	202
Actual Response Code	202
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "maximum_courses_in_a_term": 4,
8     "current_degree_level": "diploma",
9     "dp_or_ds": "both",
10    "current_courses": [
11      "BShS1001"
12    ],
13    "completed_courses": [
14      {
15        "id": "BSC2001",
16        "marks": 99
17      }
18    ]
19  }'
```

Output Response: application/json

```
1 {
2   "message": "accepted",
3   "status": "success"
4 }
```

Test Case 19

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Auth token missing
Request Type	PATCH
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "maximum_courses_in_a_term": 4,
7     "current_degree_level": "diploma",
8     "dp_or_ds": "both",
9     "current_courses": [
10       "BSC2001"
11     ],
12     "completed_courses": [
13       {
14         "id": "BSC2001",
15         "marks": 99
16       }
17     ]
18 }'
```

Output Response: application/json

```
1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 20

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	Wrong type of user logged in
Request Type	PATCH
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "maximum_courses_in_a_term": 4,
8     "current_degree_level": "diploma",
9     "dp_or_ds": "both",
10    "current_courses": [
11      "BSSH1001"
12    ],
13    "completed_courses": [
14      {
15        "id": "BSC2001",
16        "marks": 99
17      }
18    ]
19 }'

```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }

```

Test Case 21

Endpoint	http://localhost/api/v1/profile
Category	Profile
Test Case Description	User Auth does not exist
Request Type	PATCH
Expected Response Code	404
Actual Response Code	404
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/profile \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "maximum_courses_in_a_term": 4,
8     "current_degree_level": "diploma",
9     "dp_or_ds": "both",
10    "current_courses": [
11      "BSC2001"
12    ],
13    "completed_courses": [
14      {
15        "id": "BSC2001",
16        "marks": 99
17      }
18    ]
19 }'

```

Output Response: application/json

```

1 {
2   "message": "The requested resource was not found on this server",
3   "status": "error"
4 }

```

Test Case 22

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	All required information correct and present
Request Type	GET
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 [
2 {
3   "name": "ABCD",
4   "email": "ABCD@EFGH.com",
5   "role": "admin",
6   "created_at": "2012-04-23T18:25:43.511Z"
7 },
8 {
9   "name": "IJKL",
10  "email": "IJKL@EFGH.com",
11  "role": "im",
12  "created_at": "2012-04-23T18:25:43.511Z"
13 }
14 ]
```

Test Case 23

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token absent or invalid
Request Type	GET
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json'
```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 24

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token for wrong user type used

Request Type	GET
Expected Response Code	403
Actual Response Code	403

| Output Status | As Expected |
| Result | PASSED |
Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 25

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	All information present and correct
Request Type	POST
Expected Response Code	201
Actual Response Code	201
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "created",
3   "status": "success"
4 }
```

Test Case 26

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Syntax error in request
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "emal": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 27

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token missing or invalid
Request Type	POST
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "email": "IJKL@EFGH.com",
7     "role": "im"
8   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 28

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token for wrong user used
Request Type	POST
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 29

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Email in request not in database

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "abc@def.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 30

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Email in request smaller than 3 characters

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "a@b",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 31

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	role in request invalid

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8     "role": "stu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 32

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	role or email field in request missing

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com"
8   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 32

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	All fields present and correct
Request Type	PATCH
Expected Response Code	202
Actual Response Code	202
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "accepted",
3   "status": "success"
4 }
```

Test Case 33

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Syntax error in request
Request Type	PATCH
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "eml": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 34

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token missing or invalid
Request Type	PATCH
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "email": "IJKL@EFGH.com",
7     "role": "im"
8   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 35

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token for wrong type of user used
Request Type	PATCH
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 36

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	email for which data is being patched does not exist
Request Type	PATCH
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "abc@def.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 37

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	role in request invalid

Request Type	PATCH
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8     "role": "im"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 38

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	role or email field in request missing
Request Type	PATCH
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request PATCH \
2   --url http://localhost/api/v1/admins \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "email": "IJKL@EFGH.com",
8   }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 39

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	All fields present and correct
Request Type	DELETE
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins?email=IJKL%40EFGH.com' \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'

```

Test Case 40

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Syntax error in request
Request Type	DELETE
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins?email=IJKL%40EFGH.com' \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 41

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token missing or invalid
Request Type	DELETE
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins?email=IJKL%40EFGH.com' \
3   --header 'Accept: application/json'
```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 42

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	Auth token for wrong type of user used
Request Type	DELETE
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins?email=IJKL%40EFGH.com' \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 43

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	email for which data is being deleted does not exist
Request Type	DELETE
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins?email=abc%40def.com' \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 44

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	email field in request missing

Request Type	DELETE
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins' \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 45

Endpoint	http://localhost/api/v1/admins
Category	Admin
Test Case Description	email field in request has less than 3 characters

Request Type	DELETE
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url 'http://localhost/api/v1/admins?email=IJK' \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 46

Endpoint	http://localhost/api/v1/user/auth
Category	Users
Test Case Description	All required inputs present and valid
Request Type	POST
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json'\
5   --data '{
6     "email": "abc@xyz.com",
7     "password": "abcdef",
8   }'
```

Output Response: application/json

```
1 {
2   "auth": {
3     "message": "Login successful",
4     "authToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwii
5   },
6   "profile": {
7     "name": "A Student",
8     "email": "abc@xyz.com",
9     "role": "student"
10  }
11 }
```

Test Case 47

Endpoint	http://localhost/api/v1/user/auth
Category	Users
Test Case Description	Invalid email
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json'\
5   --data '{
6     "email": "abc@def.com",
7     "password": "abcdef",
8   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 48

Endpoint	http://localhost/api/v1/user/auth
Category	Users
Test Case Description	Wrong password
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json'\
5   --data '{
6     "email": "abc@xyz.com",
7     "password": "wrong",
8   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 49

Endpoint	http://localhost/api/v1/user/auth
Category	Users
Test Case Description	Wrong request syntax

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json'\
5   --data '{
6     "ema": "abc@xyz.com",
7     "password": "abcdef",
8   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 50

Endpoint Category	http://localhost/api/v1/user/auth/register	Users
Test Case Description	All fields present and correct	
Request Type	POST	
Expected Response Code	201	
Actual Response Code	201	
Output Status	As Expected	
Result		PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New Student",
7     "email": "new@stu.com",
8     "password": "newstu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "created",
3   "status": "success"
4 }
```

Test Case 51

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Invalid request syntax
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "n": "New Student",
7     "email": "new@stu.com",
8     "password": "newstu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 52

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Email already exists in database
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New Student",
7     "email": "abc@xyz.com",
8     "password": "newstu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 53

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Name smaller than 5 letters in request
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New",
7     "email": "new@stu.com",
8     "password": "newstu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 54

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Name larger than 55 letters in request
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "{56_length_name}",
7     "email": "new@stu.com",
8     "password": "newstu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 55

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Email larger than 55 letters in request

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New Student",
7     "email": "{56_length_mail}",
8     "password": "newstu"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 56

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Password smaller than 5 letters in request

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New Student",
7     "email": "abc@xyz.com",
8     "password": "n"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 56

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Password larger than 100 letters in request

Request Type	POST
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New Student",
7     "email": "abc@xyz.com",
8     "password": "{pass_longer_than_101}"
9   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 56

Endpoint	http://localhost/api/v1/user/auth/register
Category	Users
Test Case Description	Any field missing in request
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request POST \
2   --url http://localhost/api/v1/user/auth/register \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "New Student"
7   }'
```

Output Response: application/json

```
1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 57

Endpoint	http://localhost/api/v1/recommendation
Category	Recommender
Test Case Description	All inputs present and valid
Request Type	GET
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/recommendation \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {
2   "no_courses": 3,
3   "upcoming_term": [
4     "BSC2001",
5     "BSC2002",
6     "BSC2003"
7   ],
8   "matrix_order": [
9     "BSC2001",
10    "BSC2002",
11    "BSC2003",
12    "BSC2004",
13    "BSC2005",
14    "BSC2006",
15    "BSC2007",
16    "BSC2008"
17   ],
18   "recommendation_matrix": [
19     [
20       1,
21       0,
22       1,
23       0,
24       0,
25       1,
26       0
27     ],
28     [
29       1,
30       0,
31       1,
32       0,
33       0,
34       0
35     ]
36   ]
37 }
```

Test Case 58

Endpoint	http://localhost/api/v1/recommendation
Category	Recommender
Test Case Description	Invalid or missing auth token

Request Type	GET
Expected Response Code	401
Actual Response Code	401

Output Status	As Expected
Result	PASSED

Input Request

1

Output Response: application/json

1

```
{
  "message": "Bad data sent",
  "status": "error"
}
```

Test Case 59

Endpoint	http://localhost/api/v1/recommendation
Category	Recommender
Test Case Description	Wrong type of user auth sent

Request Type	GET
Expected Response Code	403
Actual Response Code	403

Output Status	As Expected
Result	PASSED

Input Request

1

```
curl --request GET \
  --url http://localhost/api/v1/recommendation \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***'
```

Output Response: application/json

1

```
{
  "message": "Access Denied",
  "status": "error"
}
```

Test Case 60

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	All required fields present and correct

Request Type	GET
Expected Response Code	200
Actual Response Code	200

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC2001 \
3   --header 'Accept: application/json, application/xml, multipart/form-data' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "id": "BSC2001",
3   "name": "Course 1",
4   "description": "This is course 1",
5   "difficulty_rating": 8.92,
6   "level": "bsc",
7   "dp_or_ds": "ds",
8   "credits": 4,
9   "pre_req": [
10     "BSC2001",
11     "BSC2002"
12   ],
13   "co_req": [
14     "BSC2001",
15     "BSC2002"
16   ],
17   "availability": [
18     "Sept2022",
19     "March2002"
20   ],
21   "instructors": [
22     {
23       "name": "ABCD",
24       "email": "ABCD@efgh.com"
25     },
26     {
27       "name": "EFGH",
28       "email": "ijkl@efgh.com"
29     }
30   ]
31 }
```

Test Case 61

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Syntax error in request

Request Type	GET
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/courses/BSC2001 \
3   --header 'Accept: application/json, application/xml, multipart/form-data' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 62

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Auth token missing or invalid

Request Type	GET
Expected Response Code	401
Actual Response Code	401

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC2001 \
3   --header 'Accept: application/json, application/xml, multipart/form-data'
```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 63

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Id in request does not exist

Request Type	GET
Expected Response Code	404
Actual Response Code	404

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC20 \
3   --header 'Accept: application/json, application/xml, multipart/form-data' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {
2   "message": "The requested resource was not found on this server",
3   "status": "error"
4 }
```

Test Case 64

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	All required fields present and correct
Request Type	PATCH
Expected Response Code	202
Actual Response Code	202
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request PATCH \
2   --url http://localhost/api/v1/course/BSC2002 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "name": "Course 1",
8     "description": "This is course 1",
9     "level": "Degree",
10    "dp_or_ds": "dp",
11    "credits": 3,
12    "pre_req": [
13      "BSC2001",
14      "BSC2002"
15    ],
16    "co_req": [
17      "BSC2001",
18      "BSC2002"
19    ],
20    "availability": [
21      "Sept2022",
22      "March2002"
23    ],
24    "instructors": [
25      {
26        "name": "ABCD",
27        "email": "ABCD@efgh.com"
28      },
29      {
30        "name": "EFGH",
31        "email": "ijkl@efgh.com"
32      }
33    ]
34 }'

```

Output Response: application/json

```

1 {
2   "message": "accepted",
3   "status": "success"
4 }

```

Test Case 65

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Syntax error in request
Request Type	PATCH
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request PATCH \
2   --url http://localhost/api/v1/course/BSC2002 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "na": "Course 1",
8     "description": "This is course 1",
9     "level": "Degree",
10    "dp_or_ds": "dp",
11    "credits": 3,
12    "pre_req": [
13      "BSC2001",
14      "BSC2002"
15    ],
16    "co_req": [
17      "BSC2001",
18      "BSC2002"
19    ],
20    "availability": [
21      "Sept2022",
22      "March2002"
23    ],
24    "instructors": [
25      {
26        "name": "ABCD",
27        "email": "ABCD@efgh.com"
28      },
29      {
30        "name": "EFGH",
31        "email": "ijkl@efgh.com"
32      }
33    ]
34 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }

```

Test Case 66

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Auth token missing or invalid
Request Type	PATCH
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request PATCH \
2   --url http://localhost/api/v1/course/BSC2002 \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "name": "Course 1",
7     "description": "This is course 1",
8     "level": "Degree",
9     "dp_or_ds": "dp",
10    "credits": 3,
11    "pre_req": [
12      "BSC2001",
13      "BSC2002"
14    ],
15    "co_req": [
16      "BSC2001",
17      "BSC2002"
18    ],
19    "availability": [
20      "Sept2022",
21      "March2002"
22    ],
23    "instructors": [
24      {
25        "name": "ABCD",
26        "email": "ABCD@efgh.com"
27      },
28      {
29        "name": "EFGH",
30        "email": "ijkl@efgh.com"
31      }
32    ]
33 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }

```

Test Case 67

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Sent Auth of wrong user type
Request Type	PATCH
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request PATCH \
2   --url http://localhost/api/v1/course/BSC2002 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "name": "Course 1",
8     "description": "This is course 1",
9     "level": "Degree",
10    "dp_or_ds": "dp",
11    "credits": 3,
12    "pre_req": [
13      "BSC2001",
14      "BSC2002"
15    ],
16    "co_req": [
17      "BSC2001",
18      "BSC2002"
19    ],
20    "availability": [
21      "Sept2022",
22      "March2002"
23    ],
24    "instructors": [
25      {
26        "name": "ABCD",
27        "email": "ABCD@efgh.com"
28      },
29      {
30        "name": "EFGH",
31        "email": "ijkl@efgh.com"
32      }
33    ]
34 }'

```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }

```

Test Case 68

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Course Id does not exist in system
Request Type	PATCH
Expected Response Code	404
Actual Response Code	404
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request PATCH \
2   --url http://localhost/api/v1/course/BSC20 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "name": "Course 1",
8     "description": "This is course 1",
9     "level": "Degree",
10    "dp_or_ds": "dp",
11    "credits": 3,
12    "pre_req": [
13      "BSC2001",
14      "BSC2002"
15    ],
16    "co_req": [
17      "BSC2001",
18      "BSC2002"
19    ],
20    "availability": [
21      "Sept2022",
22      "March2002"
23    ],
24    "instructors": [
25      {
26        "name": "ABCD",
27        "email": "ABCD@efgh.com"
28      },
29      {
30        "name": "EFGH",
31        "email": "ijkl@efgh.com"
32      }
33    ]
34 }'

```

Output Response: application/json

```

1 {
2   "message": "The requested resource was not found on this server",
3   "status": "error"
4 }

```

Test Case 69

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	All required fields present and correct

Request Type	DELETE
Expected Response Code	200
Actual Response Code	200

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url http://localhost/api/v1/course/BSC2001 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1
```

Test Case 70

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Syntax error in request

Request Type	DELETE
Expected Response Code	400
Actual Response Code	400

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url http://localhost/api/v1/coure/BSC2001 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4 }
```

Test Case 71

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Auth token missing or invalid

Request Type	DELETE
Expected Response Code	401
Actual Response Code	401

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url http://localhost/api/v1/course/BSC2001 \
3   --header 'Accept: application/json'
```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 72

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Sent Auth of wrong user type

Request Type	DELETE
Expected Response Code	403
Actual Response Code	403

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request DELETE \
2   --url http://localhost/api/v1/course/BSC2001 \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 73

Endpoint	http://localhost/api/v1/courses/{id}
Category	Courses
Test Case Description	Course Id does not exist in system

Request Type	DELETE
Expected Response Code	404
Actual Response Code	404

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request DELETE \  
2   --url http://localhost/api/v1/course/BSC20 \  
3   --header 'Accept: application/json' \  
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {  
2   "message": "The requested resource was not found on this server",  
3   "status": "error"  
4 }
```

Test Case 74

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	All required fields present and correct
Request Type	GET
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {
2   "courses": [
3     {
4       "id": "BSC2001",
5       "name": "Course 1",
6       "description": "This is course 1",
7       "difficulty_rating": 8.92,
8       "level": "Bsc",
9       "credits": 4,
10      "dp_or_ds": "dp",
11      "pre_req": [
12        "BSC2001",
13        "BSC2002"
14      ],
15      "co_req": [
16        "BSC2001",
17        "BSC2002"
18      ],
19      "availability": [
20        "Sept2022",
21        "March2002"
22      ],
23      "instructors": [
24        {
25          "name": "ABCD",
26          "email": "ABCD@efgh.com"
27        },
28        {
29          "name": "EFGH",
30          "email": "ijkl@efgh.com"
31        }
32      ]
33    }
34 }
```

Test Case 75

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	Auth token missing or invalid

Request Type	GET
Expected Response Code	401
Actual Response Code	401

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json'
```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 76

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	Sent Auth of wrong user type

Request Type	GET
Expected Response Code	403
Actual Response Code	403

Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 77

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	No courses in system

Request Type	GET
Expected Response Code	404
Actual Response Code	404

Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

```
1 {
2   "message": "The requested resource was not found on this server",
3   "status": "error"
4 }
```

Test Case 78

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	All required fields present and correct
Request Type	POST
Expected Response Code	201
Actual Response Code	201
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "code": "BSC2001",
8     "name": "Course 1",
9     "description": "This is course 1",
10    "level": "Degree",
11    "dp_or_ds": "dp",
12    "credits": 4,
13    "pre_req": [
14      "BSC2001",
15      "BSC2002"
16    ],
17    "co_req": [
18      "BSC2001",
19      "BSC2002"
20    ],
21    "availability": [
22      "Sept2022",
23      "March2002"
24    ],
25    "instructors": [
26      {
27        "name": "ABCD",
28        "email": "ABCD@efgh.com"
29      },
30      {
31        "name": "EFGH",
32        "email": "ijkl@efgh.com"
33      }
34    ]
35 }'

```

Output Response: application/json

```

1 {
2   "message": "created",
3   "status": "success"
4

```

Test Case 79

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	Auth token missing or invalid
Request Type	POST
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Content-Type: application/json' \
5   --data '{
6     "code": "BSC2001",
7     "name": "Course 1",
8     "description": "This is course 1",
9     "level": "Degree",
10    "dp_or_ds": "dp",
11    "credits": 4,
12    "pre_req": [
13      "BSC2001",
14      "BSC2002"
15    ],
16    "co_req": [
17      "BSC2001",
18      "BSC2002"
19    ],
20    "availability": [
21      "Sept2022",
22      "March2002"
23    ],
24    "instructors": [
25      {
26        "name": "ABCD",
27        "email": "ABCD@efgh.com"
28      },
29      {
30        "name": "EFGH",
31        "email": "ijkl@efgh.com"
32      }
33    ]
34 }'

```

Output Response: application/json

```

1 401
2 {
3   "message": "Bad data sent",
4   "status": "error"
}

```

Test Case 80

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	Sent Auth of wrong user type
Request Type	POST
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "code": "BSC2001",
8     "name": "Course 1",
9     "description": "This is course 1",
10    "level": "Degree",
11    "dp_or_ds": "dp",
12    "credits": 4,
13    "pre_req": [
14      "BSC2001",
15      "BSC2002"
16    ],
17    "co_req": [
18      "BSC2001",
19      "BSC2002"
20    ],
21    "availability": [
22      "Sept2022",
23      "March2002"
24    ],
25    "instructors": [
26      {
27        "name": "ABCD",
28        "email": "ABCD@efgh.com"
29      },
30      {
31        "name": "EFGH",
32        "email": "ijkl@efgh.com"
33      }
34    ]
35 }'

```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4

```

Test Case 81

Endpoint	http://localhost/api/v1/courses
Category	Courses
Test Case Description	Syntax error in request
Request Type	POST
Expected Response Code	400
Actual Response Code	400
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request POST \
2   --url http://localhost/api/v1/courses \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***' \
5   --header 'Content-Type: application/json' \
6   --data '{
7     "coe": "BSC2001",
8     "name": "Course 1",
9     "description": "This is course 1",
10    "level": "Degree",
11    "dp_or_ds": "dp",
12    "credits": 4,
13    "pre_req": [
14      "BSC2001",
15      "BSC2002"
16    ],
17    "co_req": [
18      "BSC2001",
19      "BSC2002"
20    ],
21    "availability": [
22      "Sept2022",
23      "March2002"
24    ],
25    "instructors": [
26      {
27        "name": "ABCD",
28        "email": "ABCD@efgh.com"
29      },
30      {
31        "name": "EFGH",
32        "email": "ijkl@efgh.com"
33      }
34    ]
35 }'

```

Output Response: application/json

```

1 {
2   "message": "Bad Request sent",
3   "status": "error"
4

```

Test Case 82

Endpoint	http://localhost/api/v1/course/{id}/feedback
Category	Feedback
Test Case Description	All required fields present and correct
Request Type	GET
Expected Response Code	200
Actual Response Code	200
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC2001/feedback \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer **'
```

Output Response: application/json

```
1 {
2   "feedbacks": [
3     {
4       "id": 1,
5       "poster": "21f2000381@ds.study.iitm.ac.in",
6       "pic": "myprofilepic.png",
7       "rating": 5.6,
8       "description": "This is a good course",
9       "time": "2012-04-23T18:25:43.511Z"
10    }
11  ]
12 }
```

Test Case 83

Endpoint	http://localhost/api/v1/course/{id}/feedback
Category	Courses
Test Case Description	Sent Auth token of wrong user type
Request Type	GET
Expected Response Code	403
Actual Response Code	403
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC2001/feedback \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer **'
```

Output Response: application/json

```

1 {
2   "message": "Access Denied",
3   "status": "error"
4 }
```

Test Case 84

Endpoint	http://localhost/api/v1/course/{id}/feedback
Category	Courses
Test Case Description	Auth token missing or invalid
Request Type	GET
Expected Response Code	401
Actual Response Code	401
Output Status	As Expected
Result	PASSED

Input Request

```

1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC2001/feedback \
3   --header 'Accept: application/json'
```

Output Response: application/json

```

1 {
2   "message": "Bad data sent",
3   "status": "error"
4 }
```

Test Case 85

Endpoint	http://localhost/api/v1/course/{id}/feedback
Category	Courses
Test Case Description	Id in request does not exist
Request Type	GET
Expected Response Code	404
Actual Response Code	404
Output Status	As Expected
Result	PASSED

Input Request

```
1 curl --request GET \
2   --url http://localhost/api/v1/course/BSC2006/feedback \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer **'
```

Output Response: application/json

```
1 {
2   "message": "The requested resource was not found on this server",
3   "status": "error"
4 }
```

6 Project Review

6.1 Technologies and Tools Used

6.1.1 *Backend*

- Language - Python
- Flask
- Flask-RESTful
- Flask-SQLAlchemy
- Flask-JWT-Extended
- flask_expects_json
- Pandas

6.1.2 *Frontend*

- Vue JS 3
- Vuetify 3
- Vue Router 4
- Vue ChartJS
- Cytoscape
- Cytoscape-dagre

6.1.3 *Reports and Documentation*

- Report Making - LATEX
- Presentations - Google Slides
- API Documentation - Stoplight

6.1.4 *Other Tools Used*

- Github (for development environment)
- Jira (for schedule tracking)
- Google Drive (for documents collaboration)
- Google Meet (for collaborative development – pair programming)
- Whatsapp (for general communication)

6.2 Recommendation System Algorithm

The algorithm mentioned in [Figure 29](#) is used to provide recommendation to the students. Once enough data is collected, this algorithm can be replaced with appropriate ML/AI models.

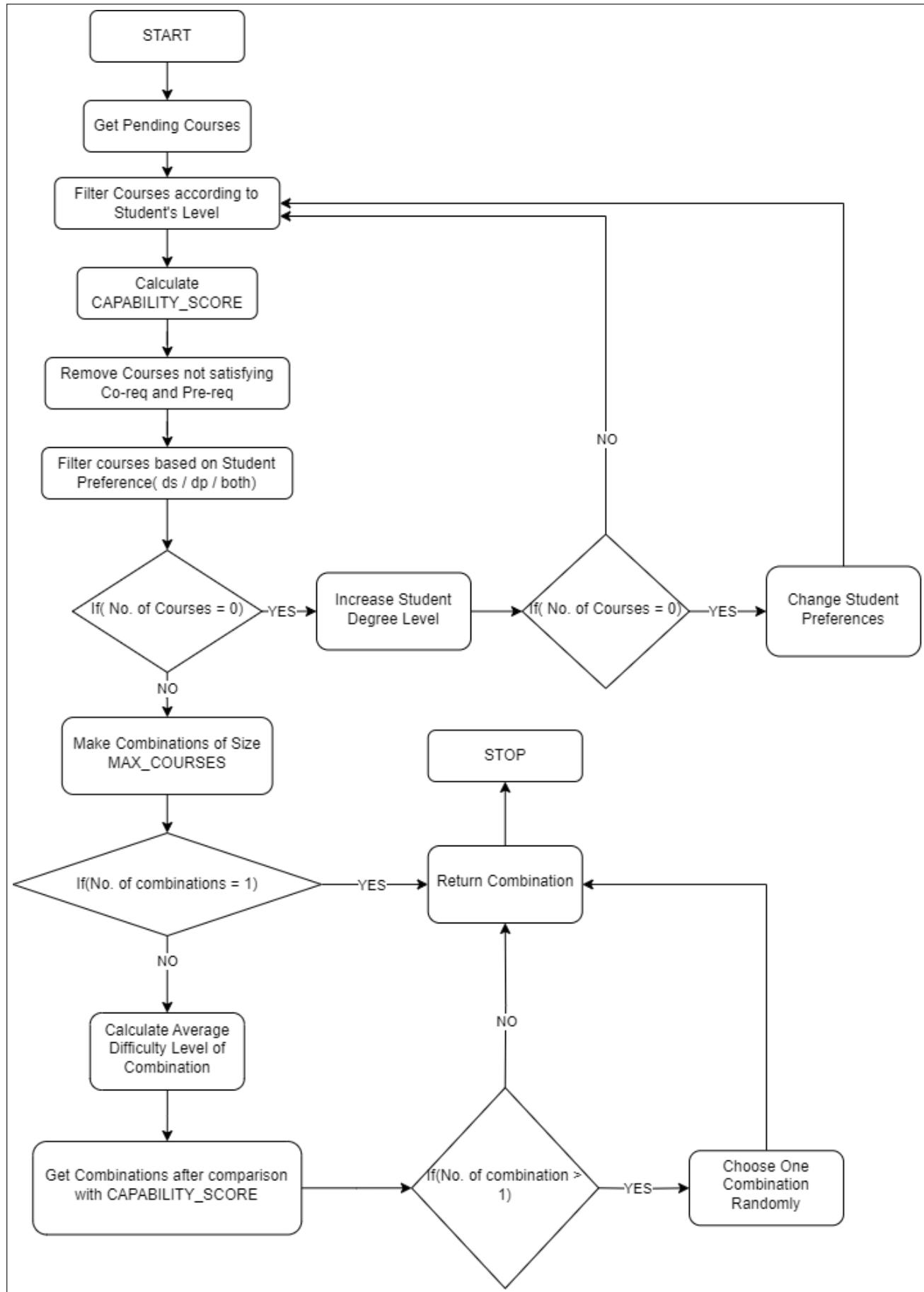


Figure 29: Algorithm used for Recommendations

7 Screenshots

7.1 Login Page

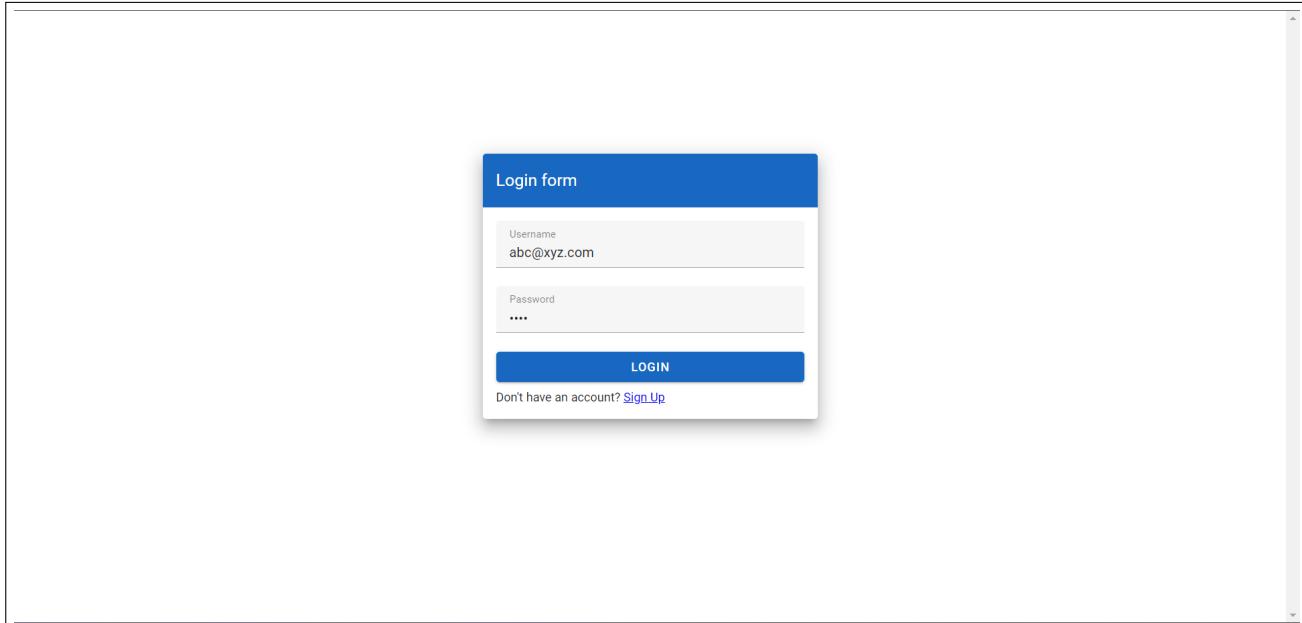


Figure 30: UI Screenshot - Login Page

7.2 Student Pages

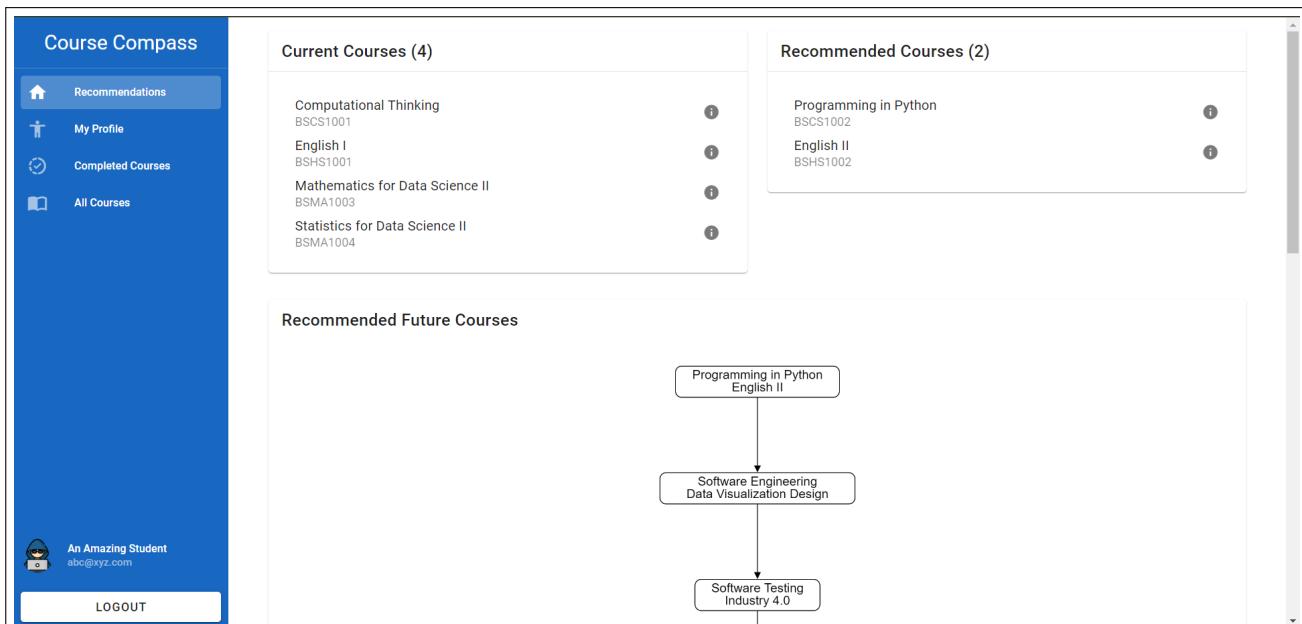


Figure 31: UI Screenshot - Student Home Page

The screenshot shows the 'Course Compass' student profile page. On the left sidebar, 'Completed Courses' is selected. The main area is divided into two sections: 'User Profile' and 'Completed Course Details'.
User Profile:

- Current Degree Level: Foundation
- Preferred Stream for Recommendations: Both
- Maximum Courses: 2
- Current Courses: Computational Thinking, English I, Mathematics for Data Science II, Statistics for Data Science II

Completed Course Details:

- Completed Course: BSMA1001, Marks: 98
- Completed Course: BSMA1002, Marks: 75

At the bottom right is a large blue 'SUBMIT' button.

Figure 32: UI Screenshot - Student Profile Page

The screenshot shows the 'Course Compass' student completed courses page. On the left sidebar, 'Completed Courses' is selected. The main area displays completed courses under three categories:
Foundation Courses (2):

- Mathematics for Data Science I
BSMA1001 | Marks: 98
- Statistics for Data Science I
BSMA1002 | Marks: 75

Diploma Courses (0):
Degree Courses (0):

Figure 33: UI Screenshot - Student Completed Courses Page

Course Compass

- Recommendations
- My Profile
- Completed Courses
- All Courses

An Amazing Student
abc@xyz.com

Logout

Foundation Courses (8)

- Mathematics for Data Science I
BSMA1001
- Statistics for Data Science I
BSMA1002
- Computational Thinking
BSCS1001
- English I
BSHS1001
- Mathematics for Data Science II
BSMA1003
- Statistics for Data Science II
BSMA1004
- Programming in Python
BSCS1002
- English II
BSHS1002

Diploma Courses (32)

- Database Management Systems
BSCS2001
- Programming, Data Structures and Algorithms using Python
BSCS2002
- Programming Concepts using Java
BSCS2005
- System Commands
BSE2001
- Machine Learning Foundations
BSCS2004
- Business Data Management
BSMS2001
- Software Testing
BSCS3002
- Software Engineering
BSCS3001
- AI: Search Methods for Problem Solving
BSCS3003
- Deep Learning
BSCS3004
- Strategies for Professional Growth
BSGN3001
- Algorithmic Thinking in Bioinformatics
BSBT4001
- Big Data and Biological Networks
BSBT4002
- Programming in C

Figure 34: UI Screenshot - Student All Courses Page

Course Compass

- Recommendations
- My Profile
- Completed Courses
- All Courses

An Amazing Student
abc@xyz.com

Logout

BSMA1001 - Mathematics for Data Science I

About the course

Level: Foundation
Data Science Course

This course introduces functions (straight lines, polynomials, exponentials and logarithms) and discrete mathematics (basics, graphs) with many examples. The students will be exposed to the idea of using abstract mathematical structures to represent concrete real life situations.

Average Difficulty

5.934

Instructors

CTM 1 ctm1@gmail.com
CTM 2 ctm2@gmail.com
CTM 3 ctm3@gmail.com
CTM 9 ctm9@gmail.com
CTM 18 ctm18@gmail.com
CTM 11 ctm11@gmail.com
CTM 6 ctm6@gmail.com

Feedbacks (61)

GIVE FEEDBACK

student99@gmail.com
11:28, December 19, 2023

This application was excellent and professional. It was not very flashy, but it was polished and refined. The design was sophisticated and classy and the features were efficient and effective. I respected the way the application delivered high-quality results and performance.

Rating: 6

Figure 35: UI Screenshot - Student Each Course Details Page

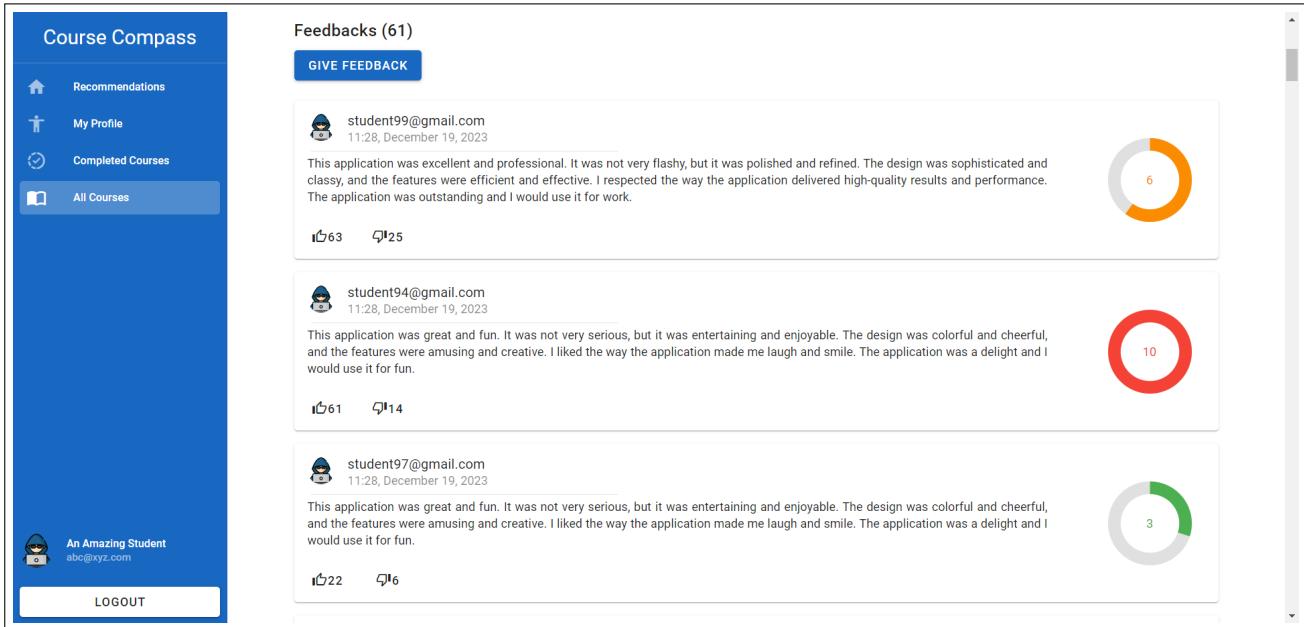


Figure 36: UI Screenshot - Student Each Course Feedback

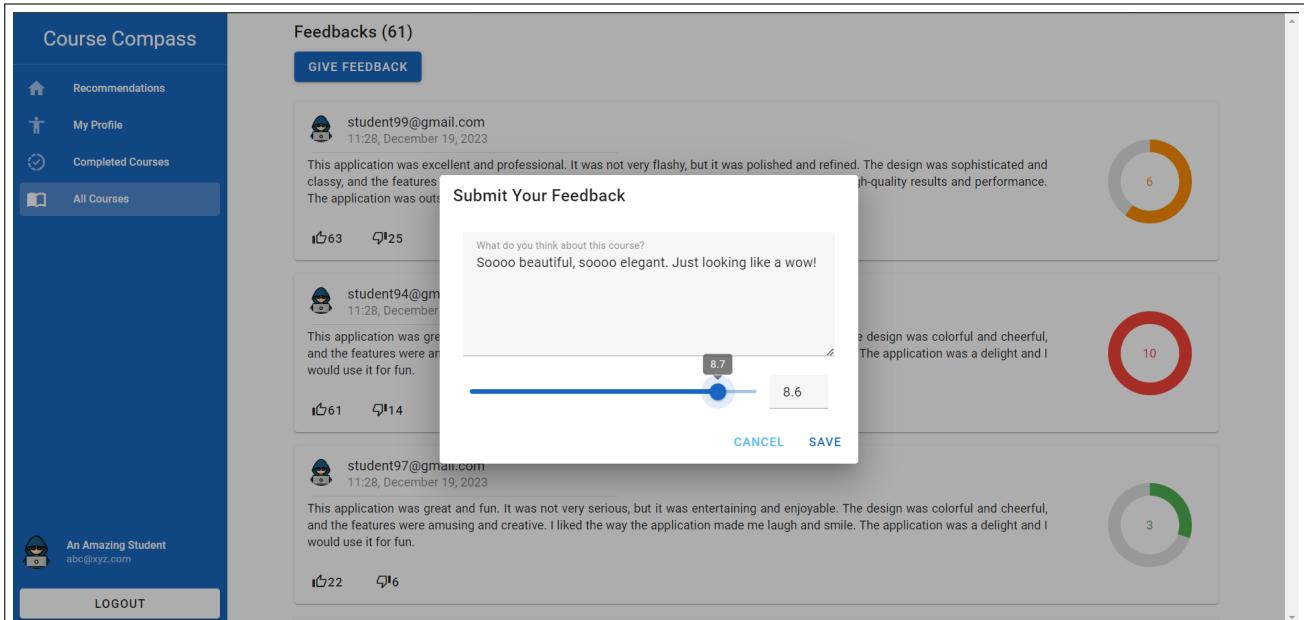


Figure 37: UI Screenshot - Student Each Course Give Feedback

7.3 Admin Pages

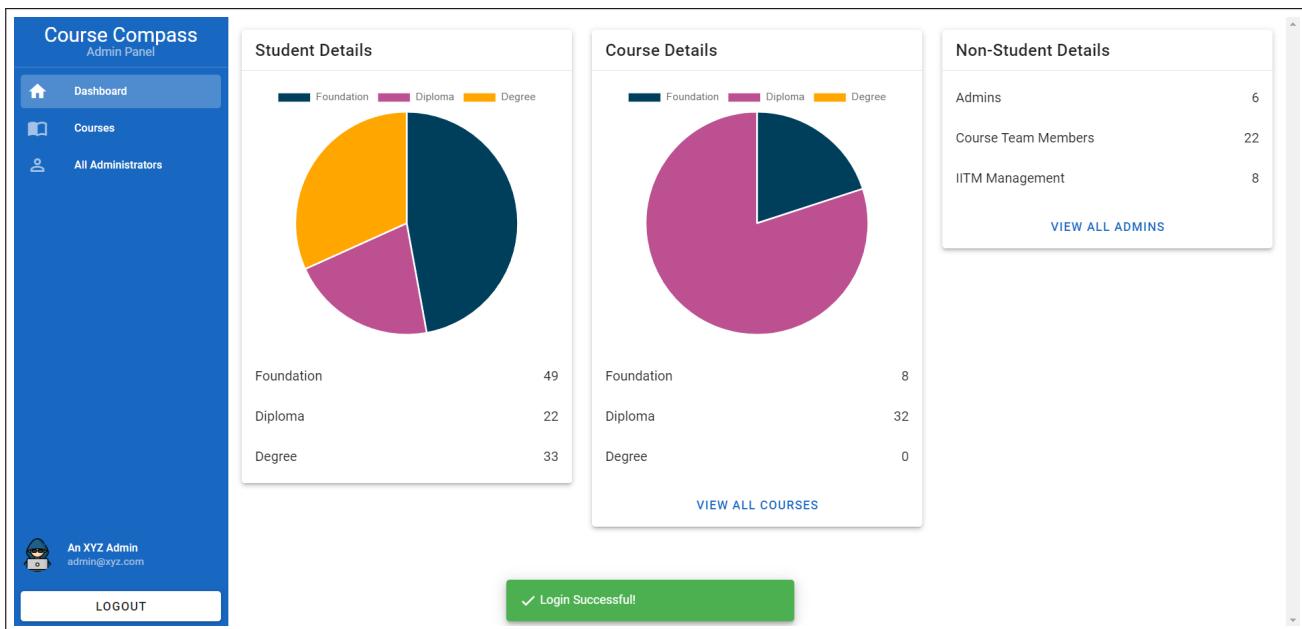


Figure 38: UI Screenshot - Admin Home Page

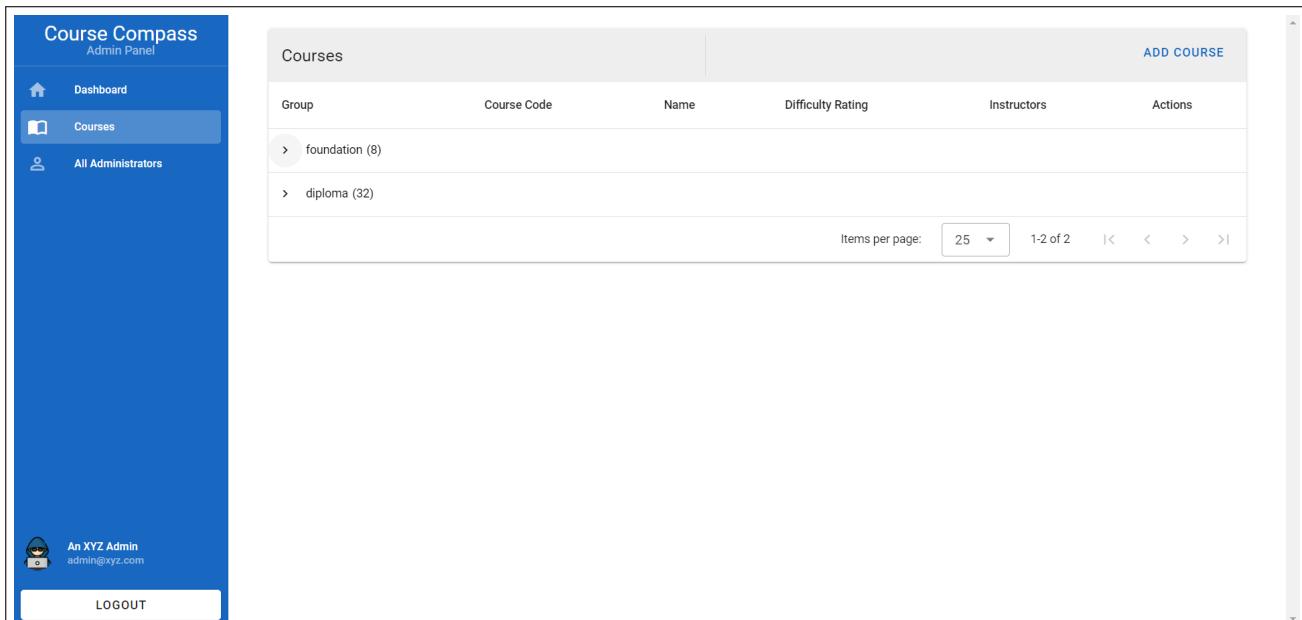


Figure 39: UI Screenshot - Admin Courses Page

The screenshot shows the Course Compass Admin Panel. On the left, there's a sidebar with a dark blue background. It has a user icon, the text 'An XYZ Admin admin@xyz.com', and a 'LOGOUT' button. The main area is titled 'Courses' and contains a table with columns: Group, Course Code, Name, Difficulty Rating, Instructors, and Actions. A modal window titled 'Edit Course' is open in the center. It has fields for 'Course Name' (Mathematics for Data Science I) and 'Course Team Members' (a dropdown menu listing several email addresses). At the bottom of the modal are 'CANCEL' and 'SAVE' buttons.

Group	Course Code	Name	Difficulty Rating	Instructors	Actions
foundation (8)			5.934	CTM 1 CTM 2 CTM 3 CTM 9 CTM 18 CTM 11 CTM 6 Jasleen Kaur CTM 4 CTM 10 CTM 11 CTM 2 CTM 14 CTM 18 CTM 12 CTM 10 CTM 3 CTM 6	edit trash
	BSCS1001	Computational Thinking	5.864		edit trash

Figure 40: UI Screenshot - Admin Edit Course Info

The screenshot shows the Admin Members section of the Course Compass Admin Panel. The sidebar on the left is identical to Figure 40. The main area is titled 'Admin Members' and contains a table with columns: Group, Email, Name, Created At, and Actions. There are two sections: 'Administrators' (containing 6 entries) and 'Course Team Members' (containing 1 entry). At the bottom, there are pagination controls for 'Items per page:' (set to 25), '1-9 of 9', and navigation arrows.

Group	Email	Name	Created At	Actions
Administrators	admin@xyz.com	An XYZ Admin	2023-12-19 11:27:40	edit trash
	xyz@gmail.com	An XYZ Gmail Admin	2023-12-19 11:27:40	edit trash
	anhatsingh@gmail.com	Anhat Singh	2023-12-19 11:27:40	edit trash
	akhil@gmail.com	Akhil Aggarwal	2023-12-19 11:27:40	edit trash
	guramanat@gmail.com	Guramanat Singh	2023-12-19 11:27:40	edit trash
	harnoor@gmail.com	Harnoor Kaur	2023-12-19 11:27:40	edit trash
Course Team Members				
IITM Management				

Figure 41: UI Screenshot - List of other admins

7.4 Course Team Member Pages

The screenshot shows the Course Compass CTM Panel. On the left is a sidebar with a user profile for 'Jasleen Kaur' and an email 'jasleenkaur@gmail.com'. The main area is divided into two sections: 'Foundation Courses (2)' and 'Diploma Courses (10)'. Under 'Foundation Courses (2)', there are two entries: 'Statistics for Data Science I' (BSMA1002) with a difficulty score of 5.864 and 'English II' (BSHS1002) with a difficulty score of 5. Under 'Diploma Courses (10)', there are ten entries: 'Programming Concepts using Java' (BSCS2005), 'System Commands' (BSSE2001), 'Software Testing' (BSCS3002), 'Software Engineering' (BSCS3001), 'Algorithmic Thinking in Bioinformatics' (BSBT4001), 'Data Visualization Design' (BSCS4001), 'Design Thinking for Data Science' (BSMS3002), 'Market Research' (BSHS1002), 'Machine Learning Practice' (BSCS3003), and 'PROJECT - Business Data Management - Project' (BSMS2001P).

Figure 42: UI Screenshot - Course Team Member Home Page

This screenshot shows the Course Compass CTM Panel for the course 'BSCS2005 - Programming Concepts using Java'. The sidebar shows the same user profile and course list as Figure 42. The main content area includes a 'Course Description' section with a detailed description of the course objectives, a 'Course Information' section with details like 'Diploma Level', 'Programming Course', 'Course Category', and '4 Credits', and a 'Average Difficulty' section featuring a donut chart with a value of 5.591. Below this, a 'Feedbacks (66)' section displays two student reviews. The first review by 'student99@gmail.com' on December 19, 2023, states: 'This application was disappointing and frustrating. It was slow, buggy, and crashed often. The design was outdated and cluttered, and the features were limited and confusing. I had trouble finding what I needed and the application did not offer any help or guidance. The application did not meet my needs and I would not use it again.' The second review by 'student94@gmail.com' on December 19, 2023, states: 'This application was decent and satisfactory. It was not very impressive, but it did the job. The design was simple and plain, and the features were functional but basic.' Below each review are upvote and downvote counts.

Figure 43: UI Screenshot - Course Team Member View Student Feedbacks Page

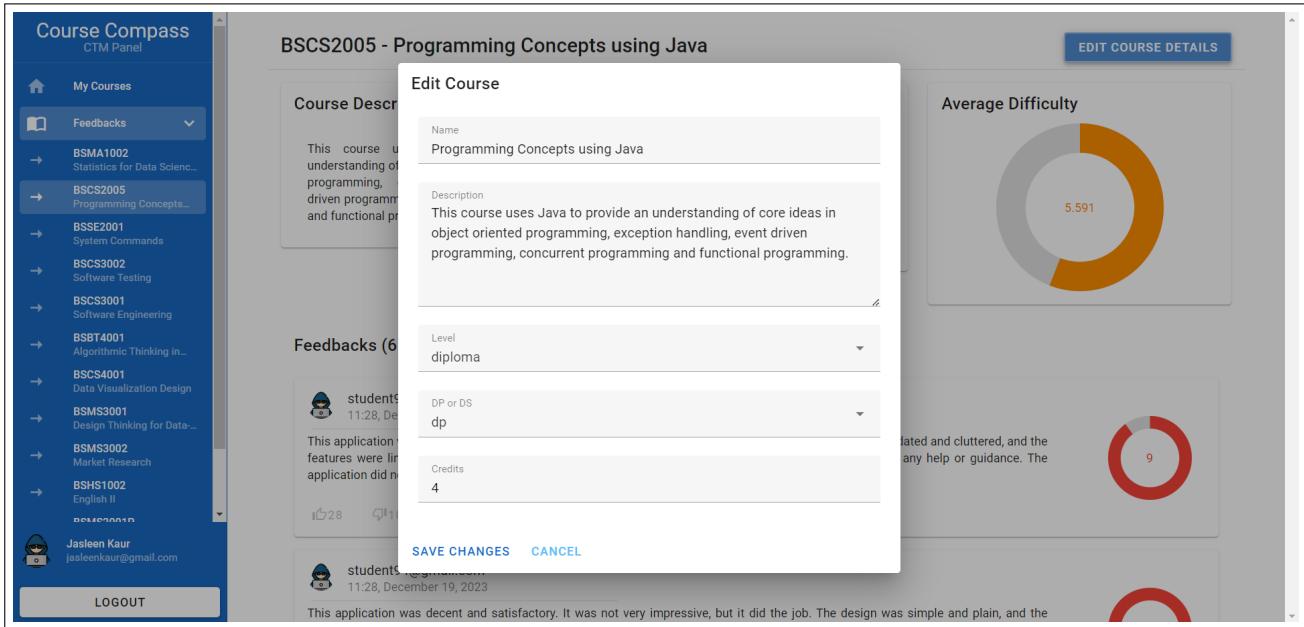


Figure 44: UI Screenshot - Course Team Member Edit Course Details Page

7.5 IITM Management Member

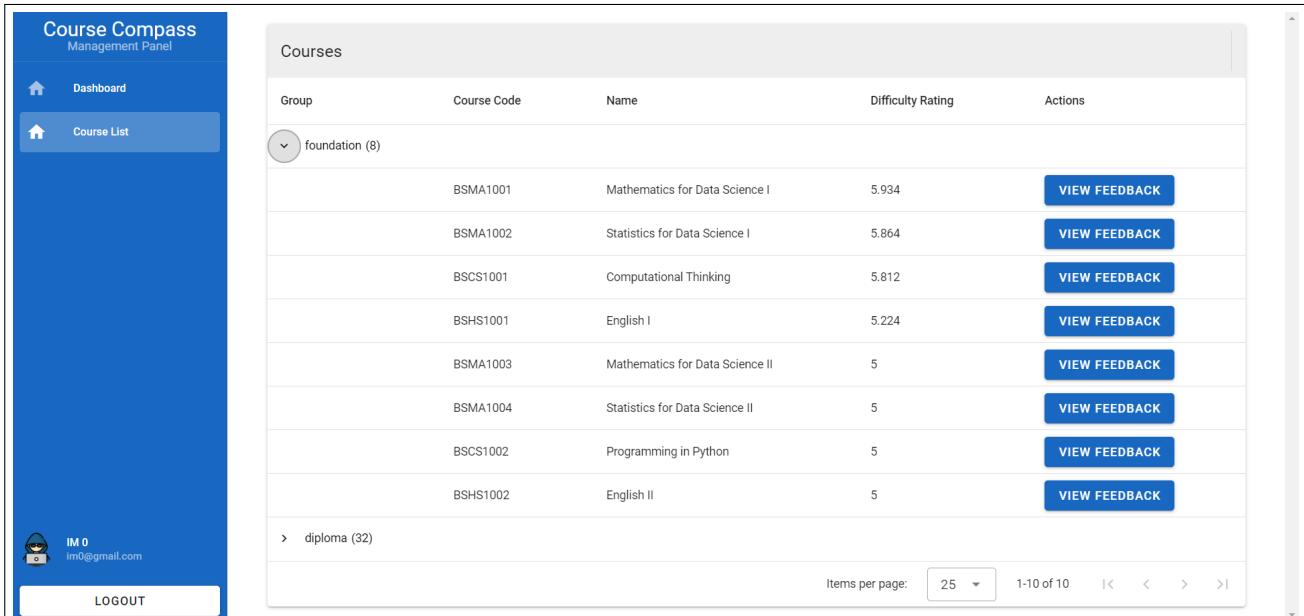


Figure 45: UI Screenshot - IITM Management Member Page

7.6 Github Repository and Google Drive

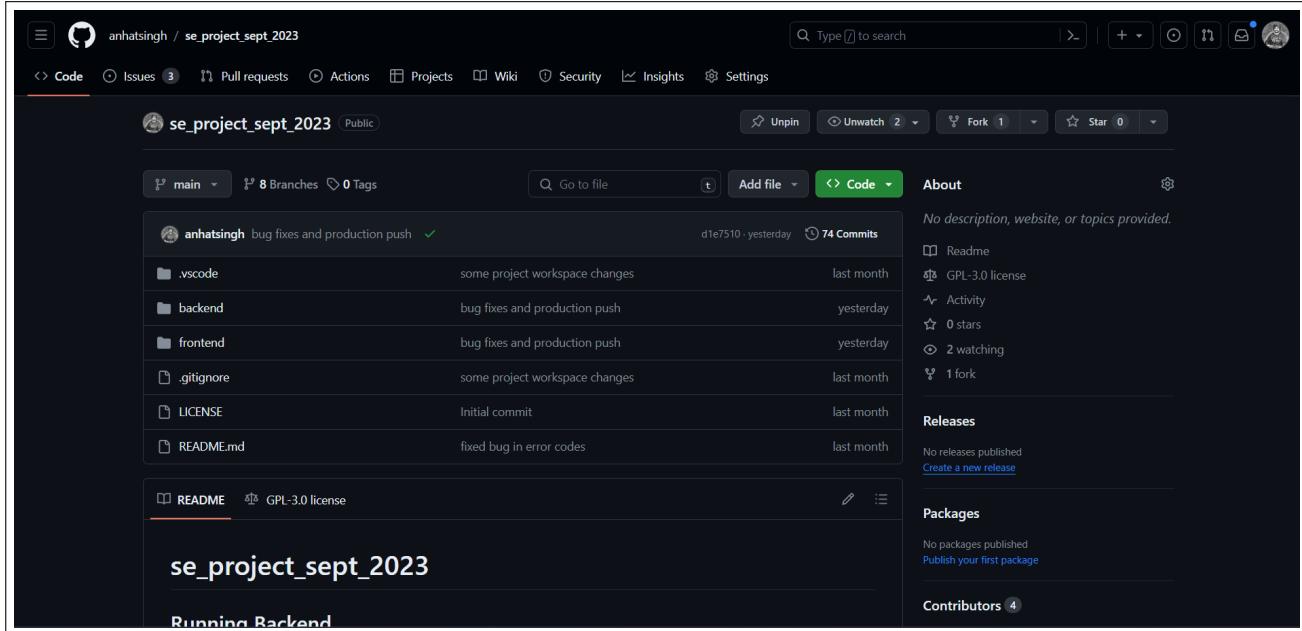


Figure 46: Github Repo and the commits done by team members; 74 commits were made during the development phase

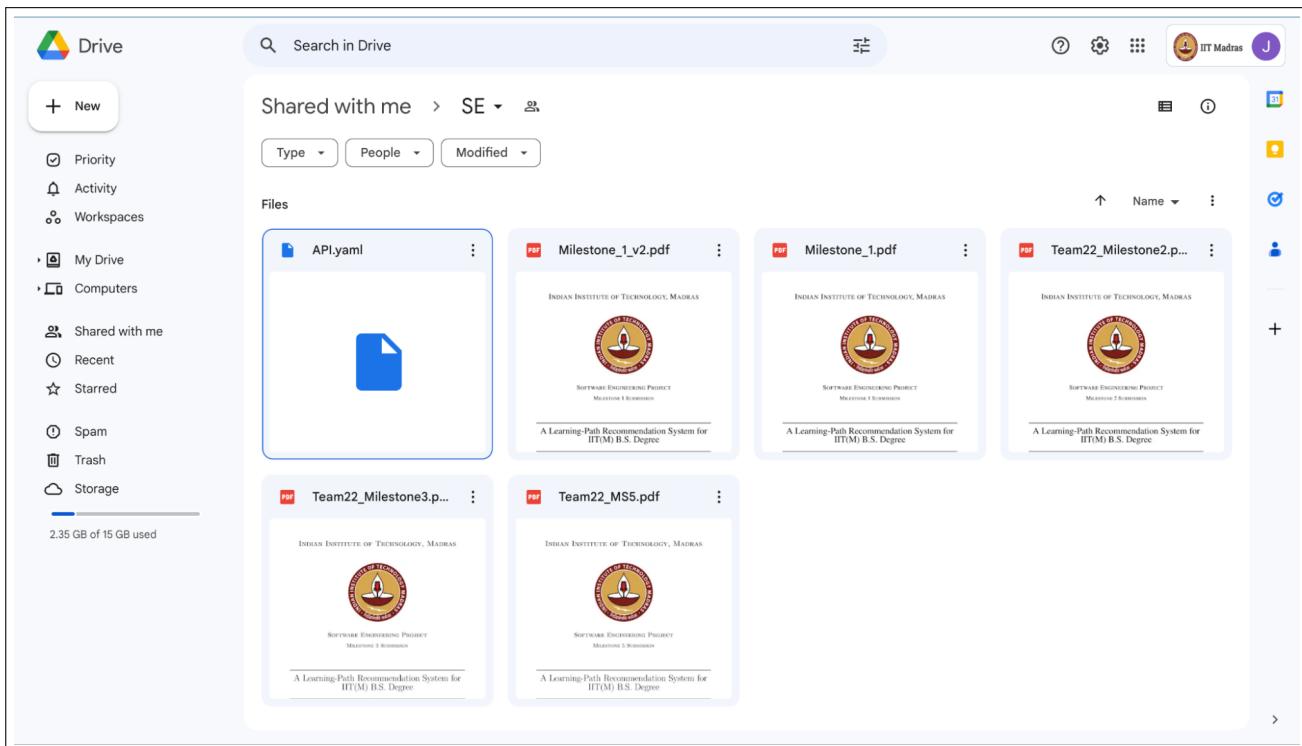


Figure 47: Google Drive folder with shared access

8 Issue Report

For issue reporting, we have used various platforms like:

1. Github Issues
2. Jira
3. Whatsapp
4. Google Docs

Figure 48 shows screenshots of some of the issues.

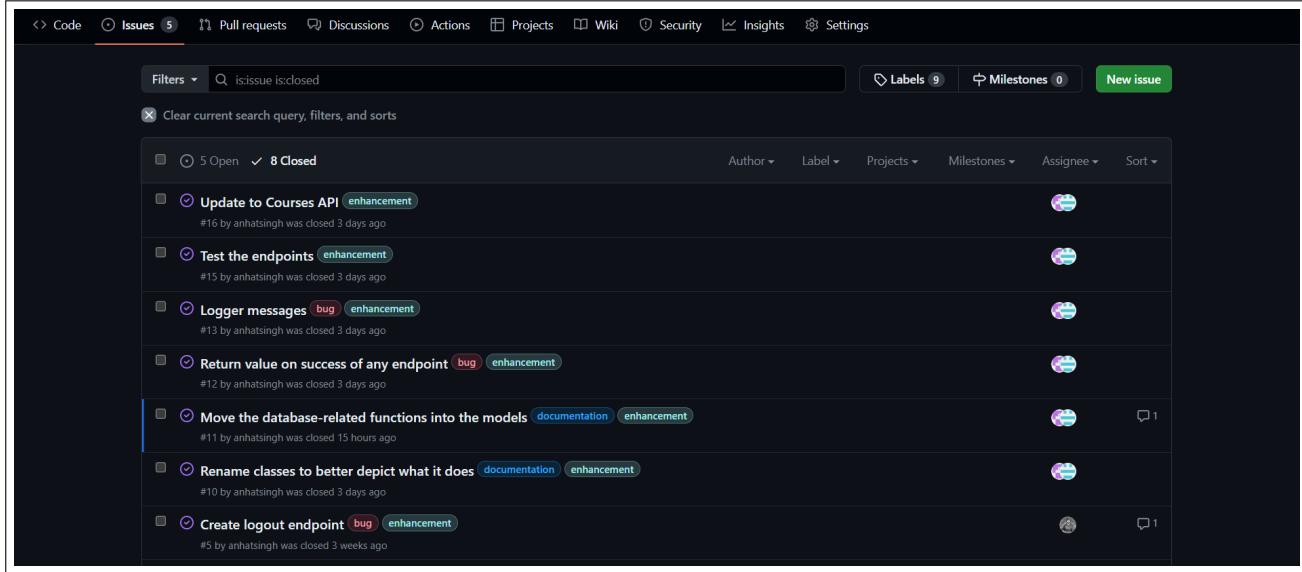


Figure 48: Issues Tracking in Github

9 Github Repo, Video and PPT

- Github Repository: <https://github.com/anhatsingh/Software-Engineering-Project-IITM-Sept2023>
- Video Link: [Google Drive](#)
- PPT Link: [Google Drive](#)