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

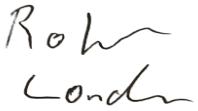
The software engineering project, titled "A Learning-Path Recommendation System for IIT(M) B.S. Degree", has been successfully completed under the supervision of instructors of Software Engineering Course, IIT Madras.

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.

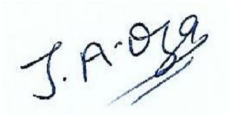
(Anhat Singh)

(Jasleen Kaur)



(Rohit Vinit Londhe)

(Anchit Mandal)



(Oza Jay Amit)

Contents

List of Figures

List of Tables

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)

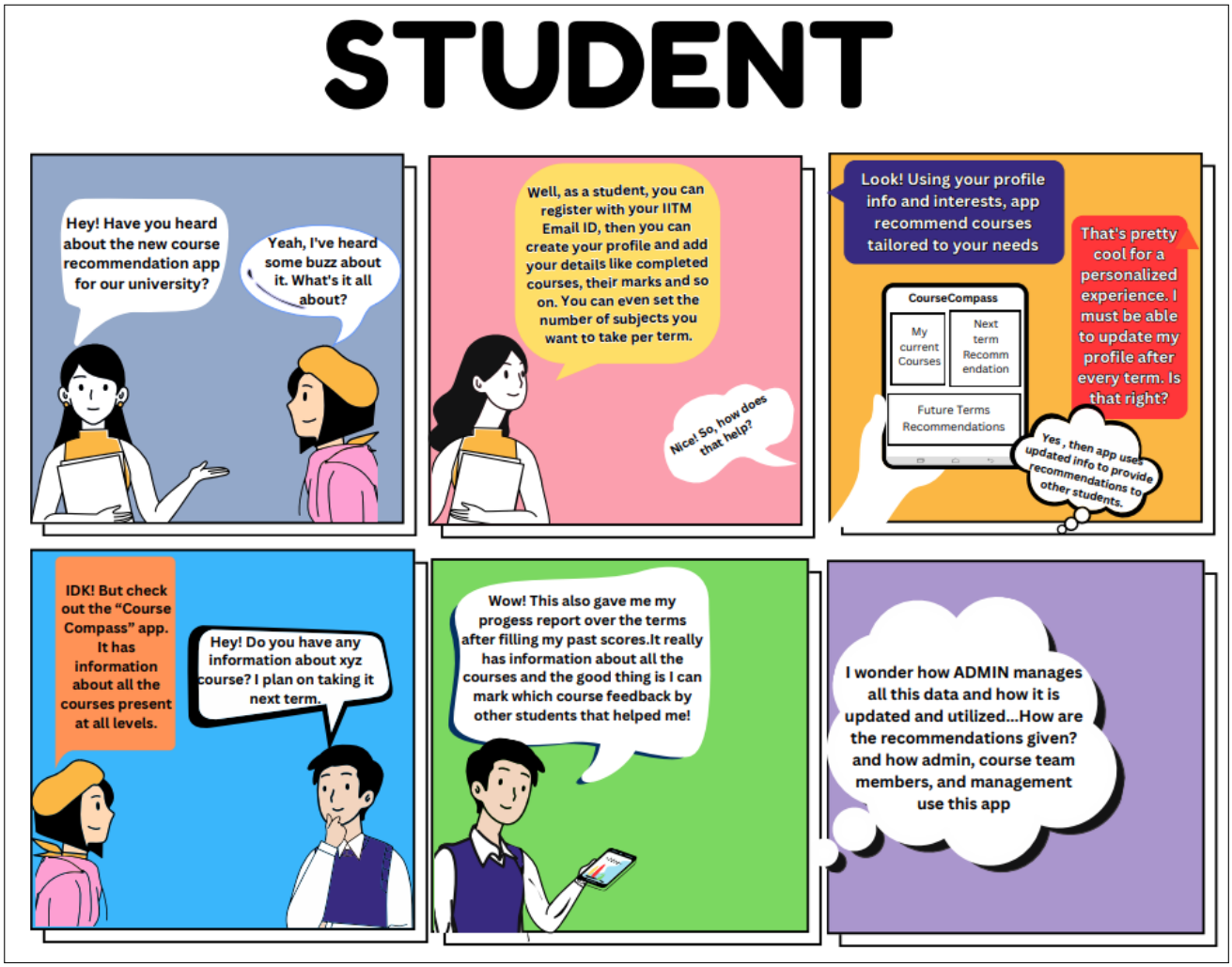


Figure 1: Student Storyboard

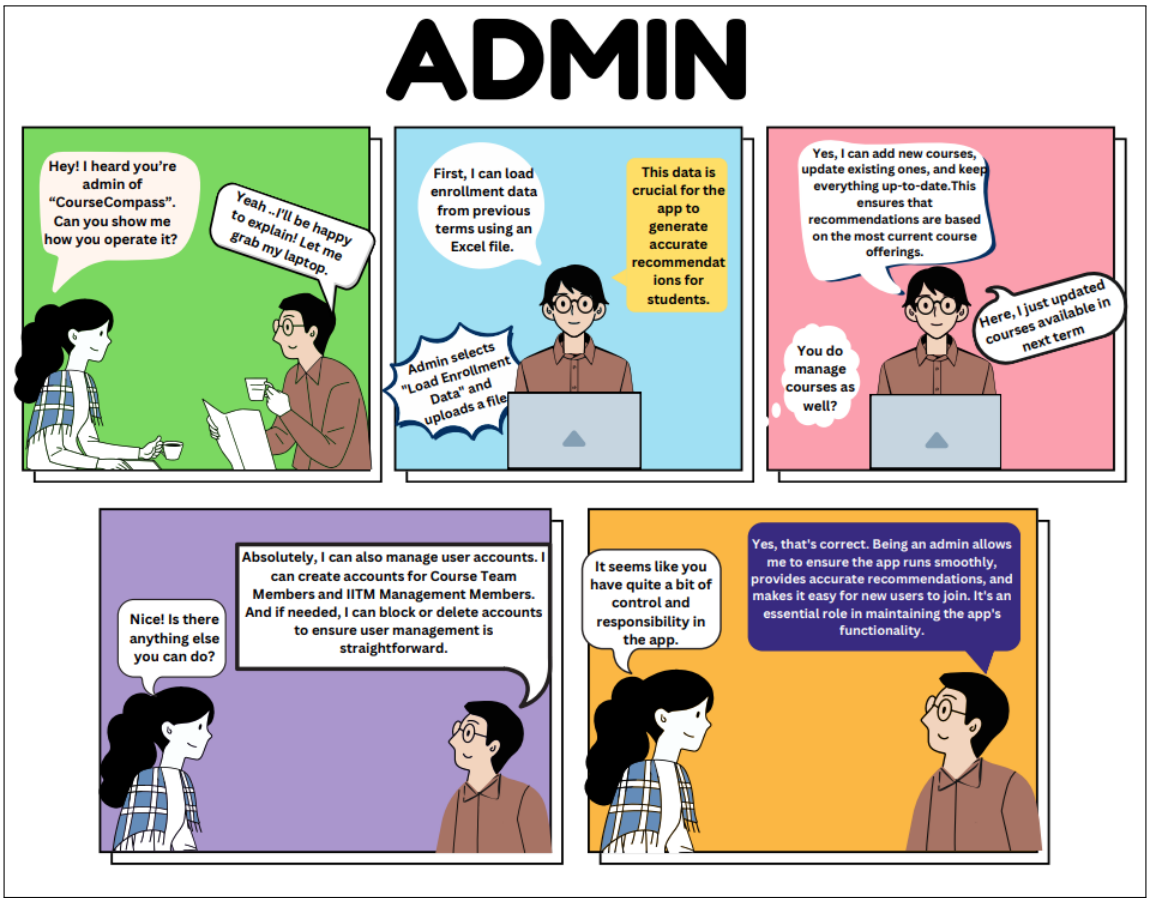


Figure 2: Admin Storyboard

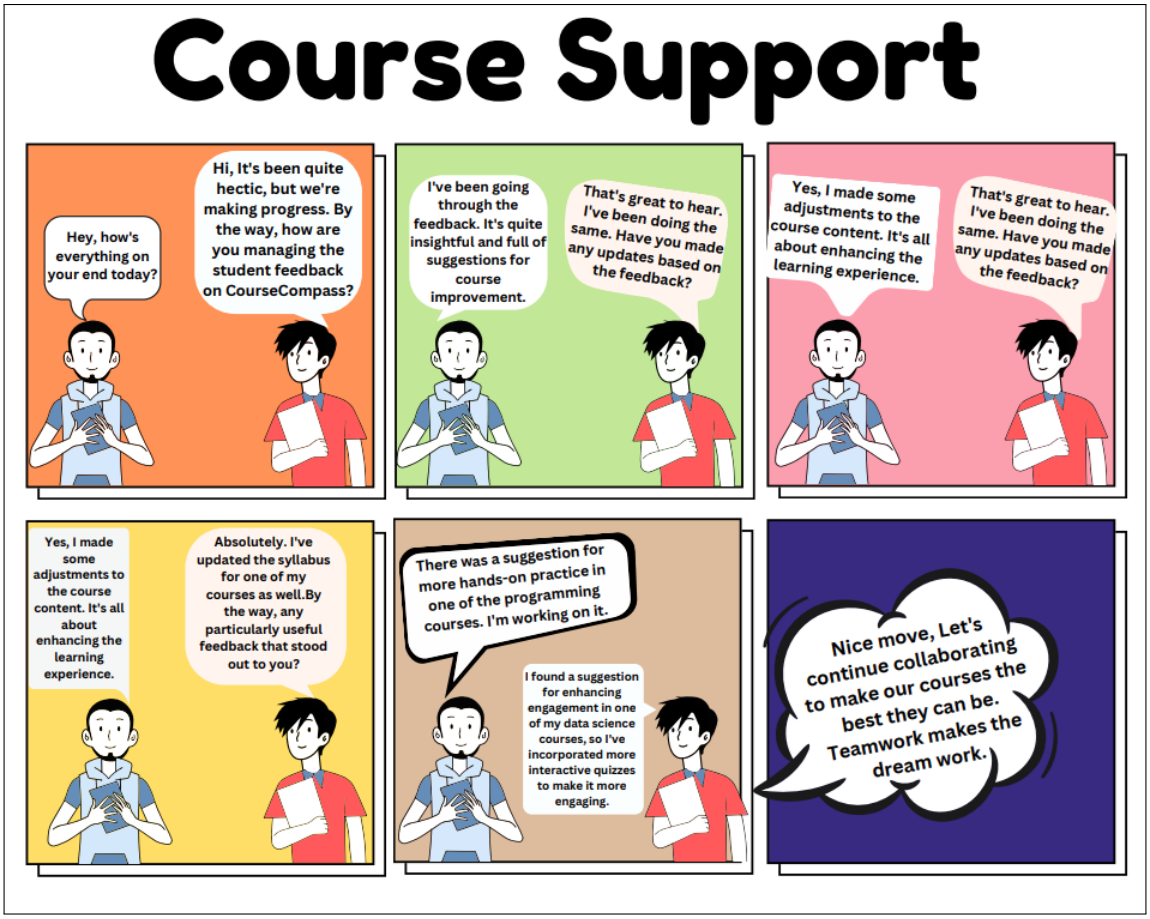


Figure 3: Course Team Member Storyboard

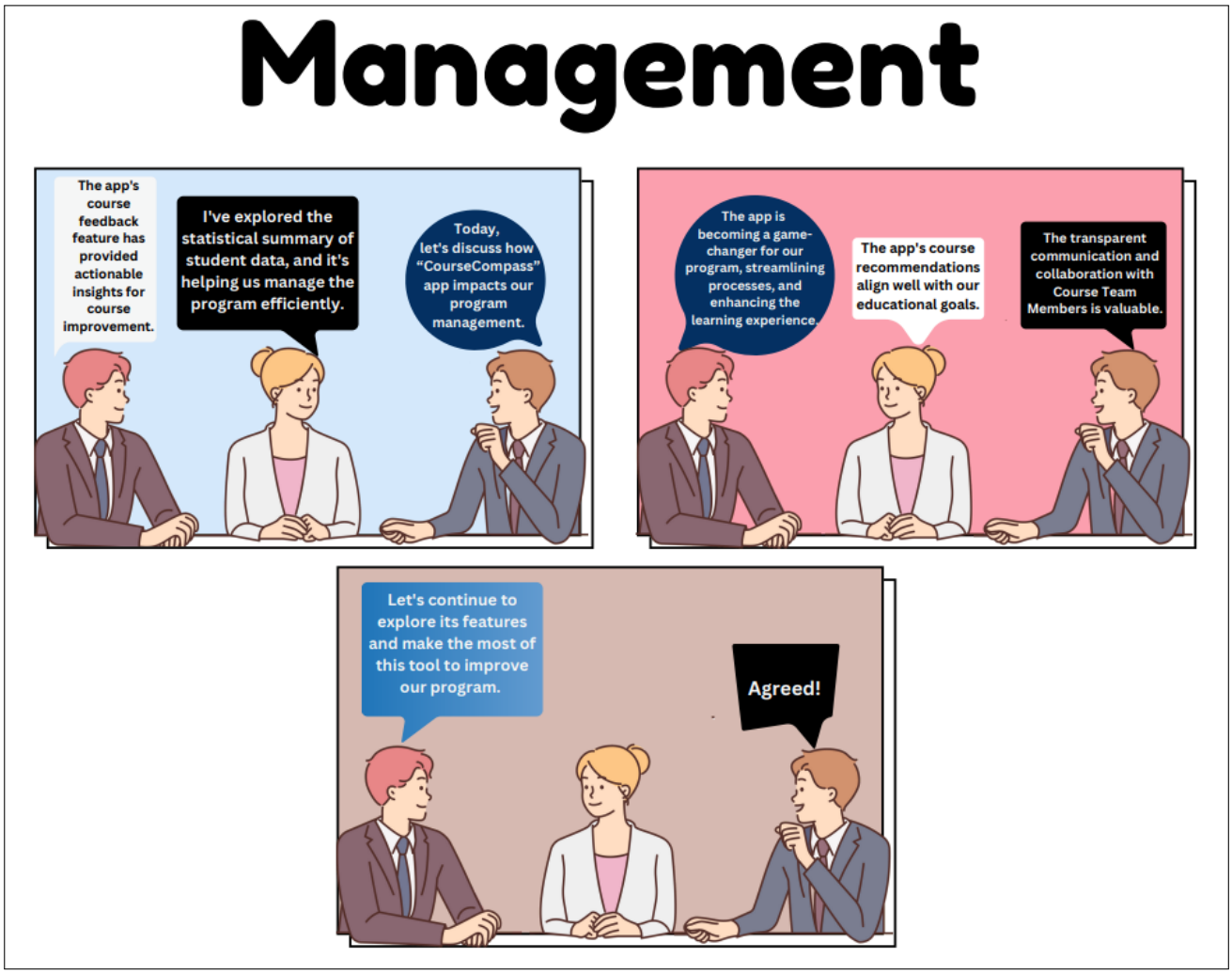


Figure 4: IITM Management Storyboard

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

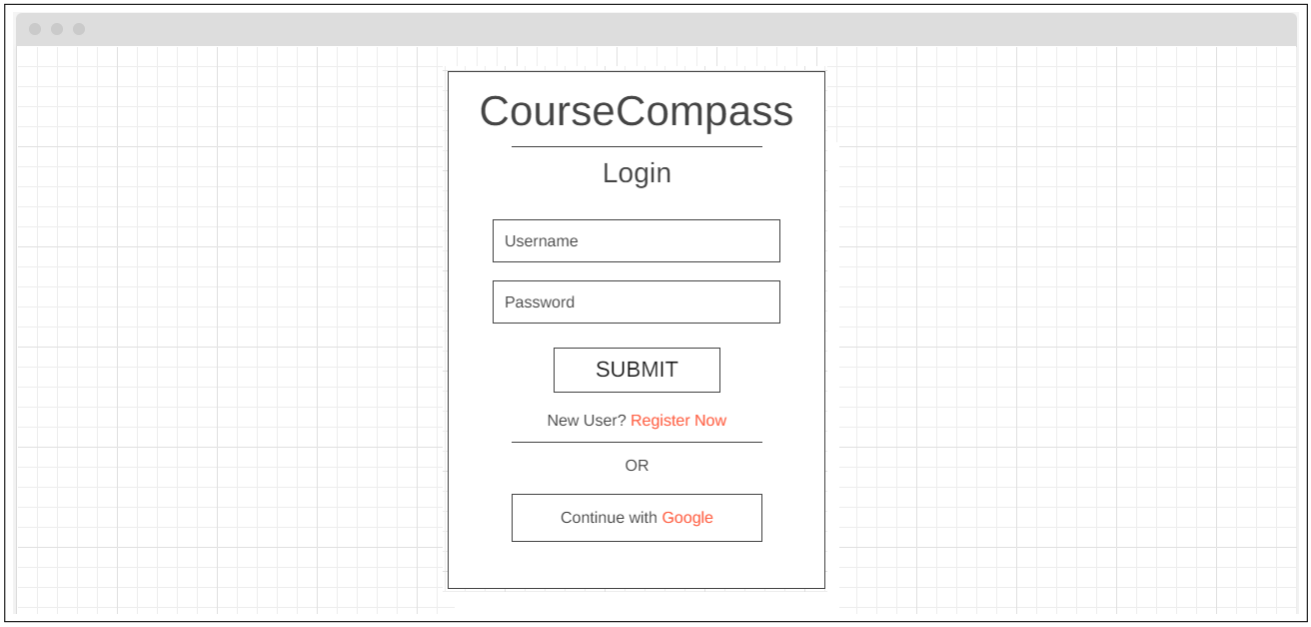


Figure 5: Login Page (Registration page will also look similar)

2.2.2 Students Pages' Wireframes

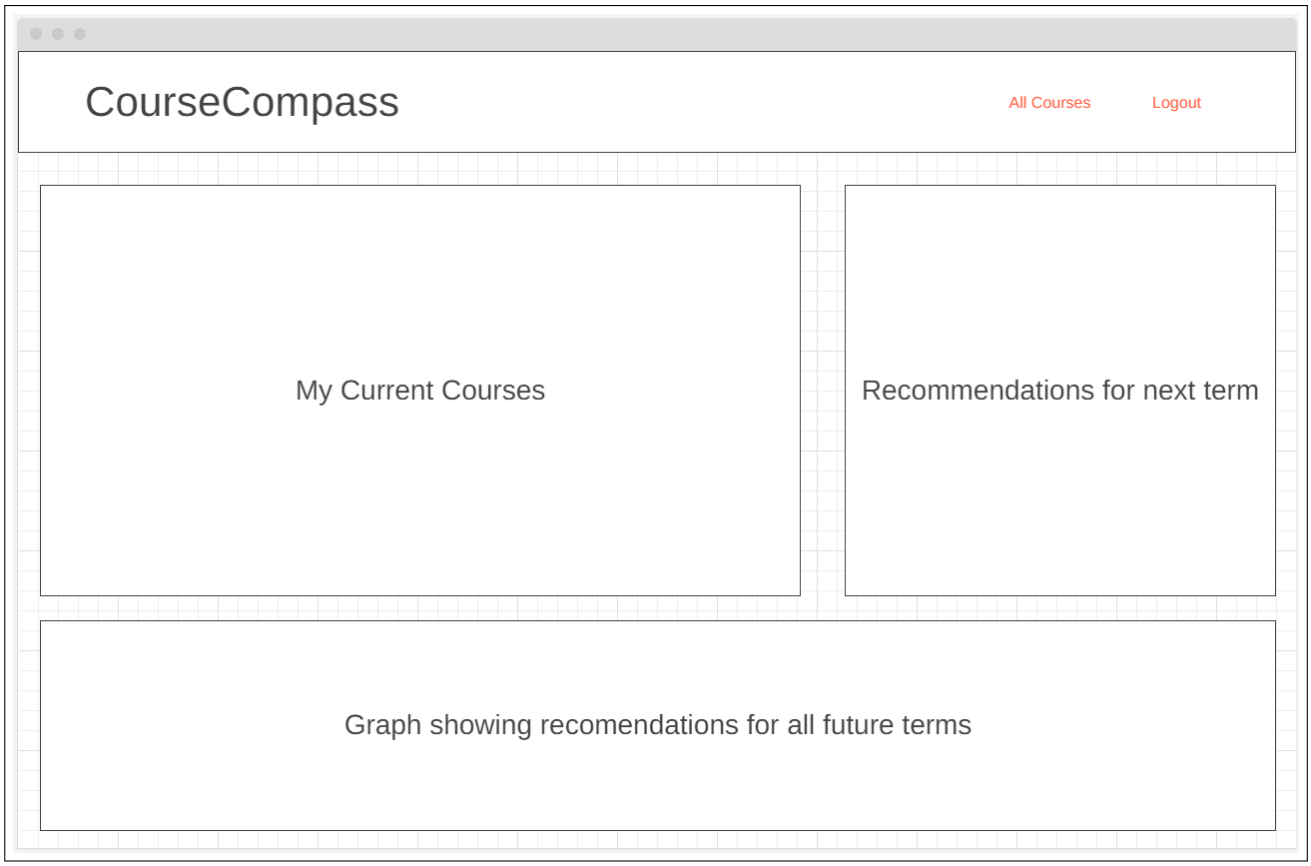


Figure 6: Student's homepage

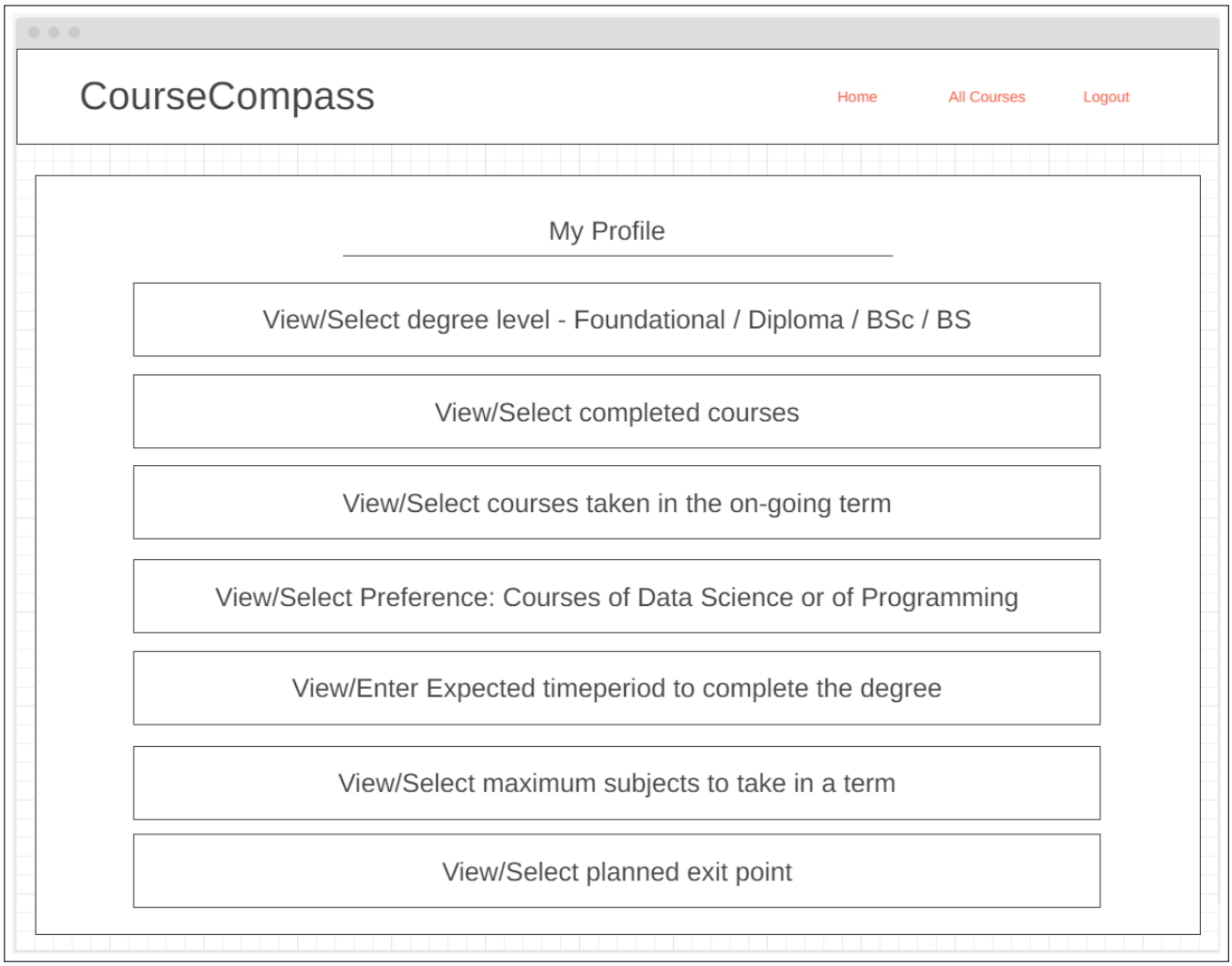


Figure 7: Student's profile page

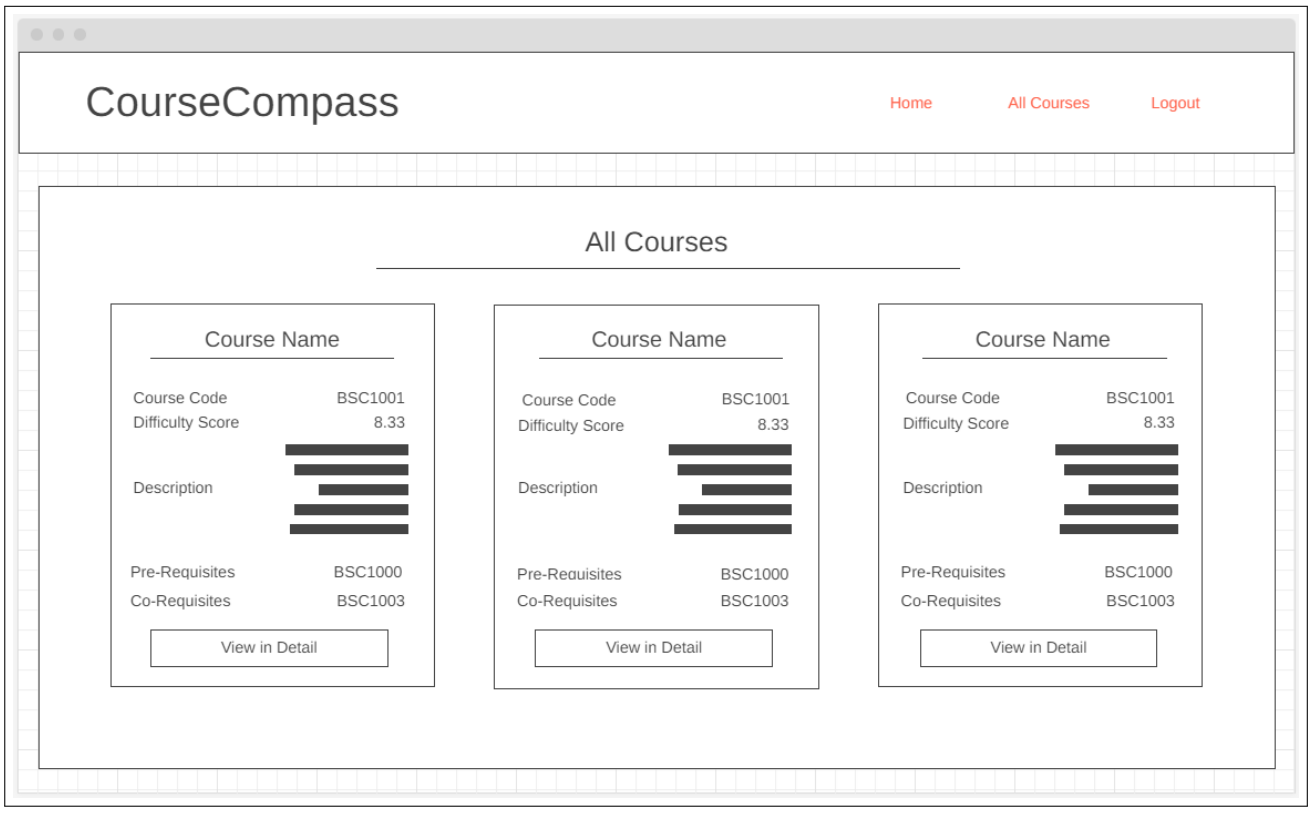


Figure 8: Page for Students to view all the courses

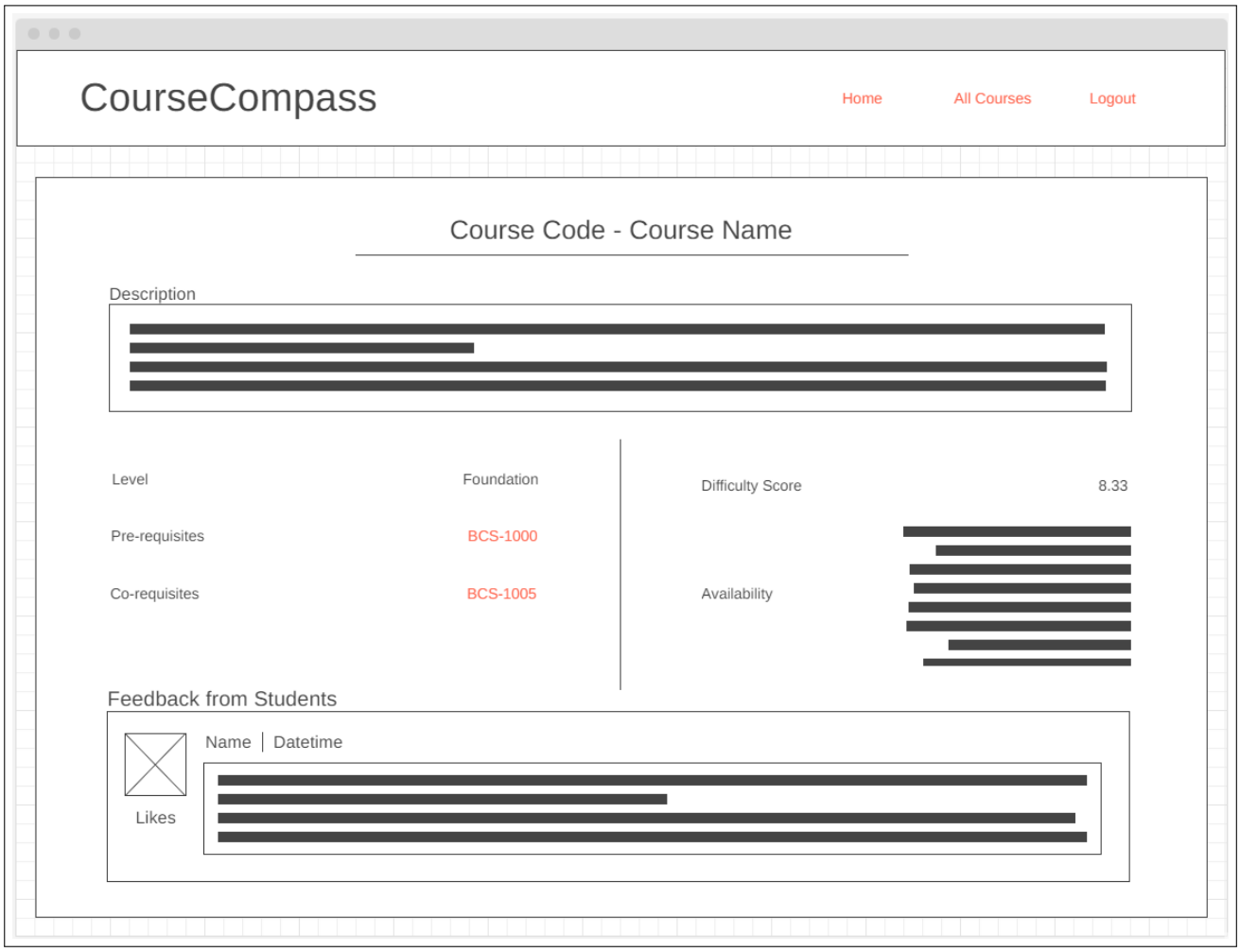


Figure 9: Student's courses and feedback page

2.2.3 Administrators Pages' Wireframes

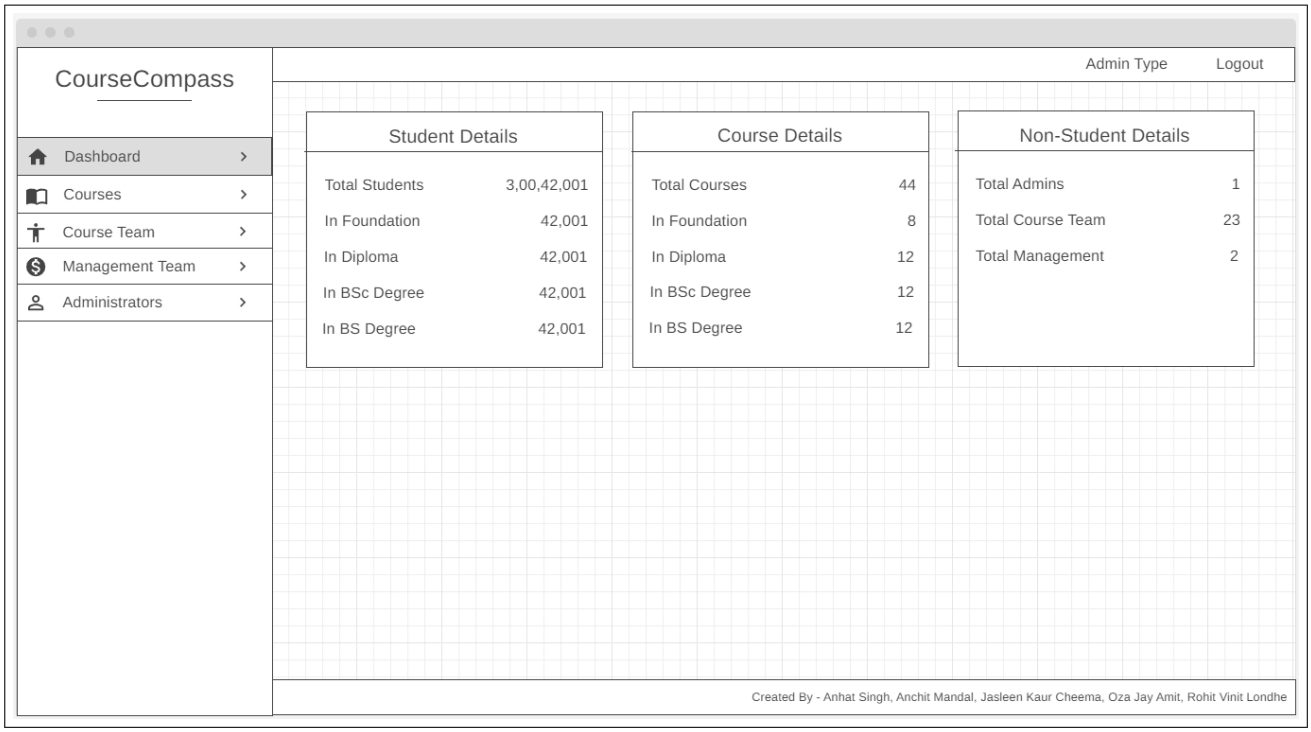


Figure 10: Administrator dashboard

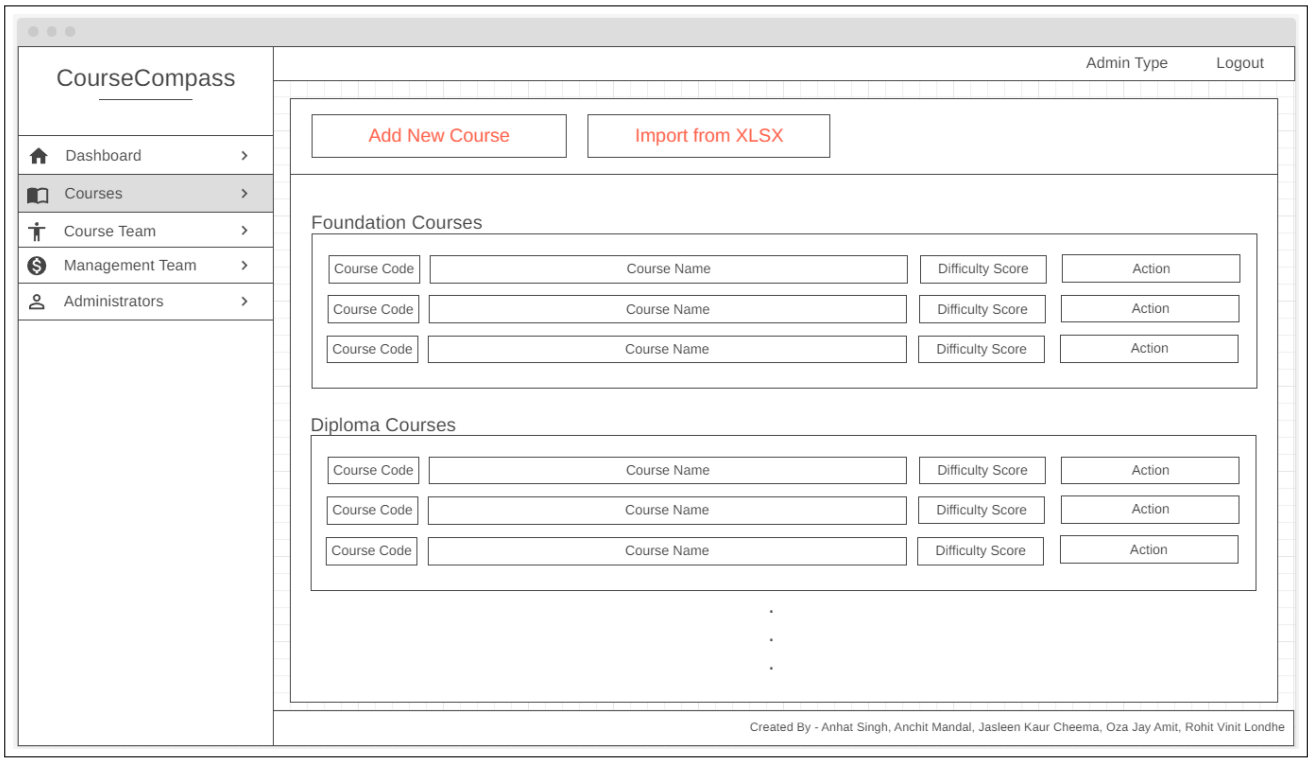


Figure 11: Administrator all courses page

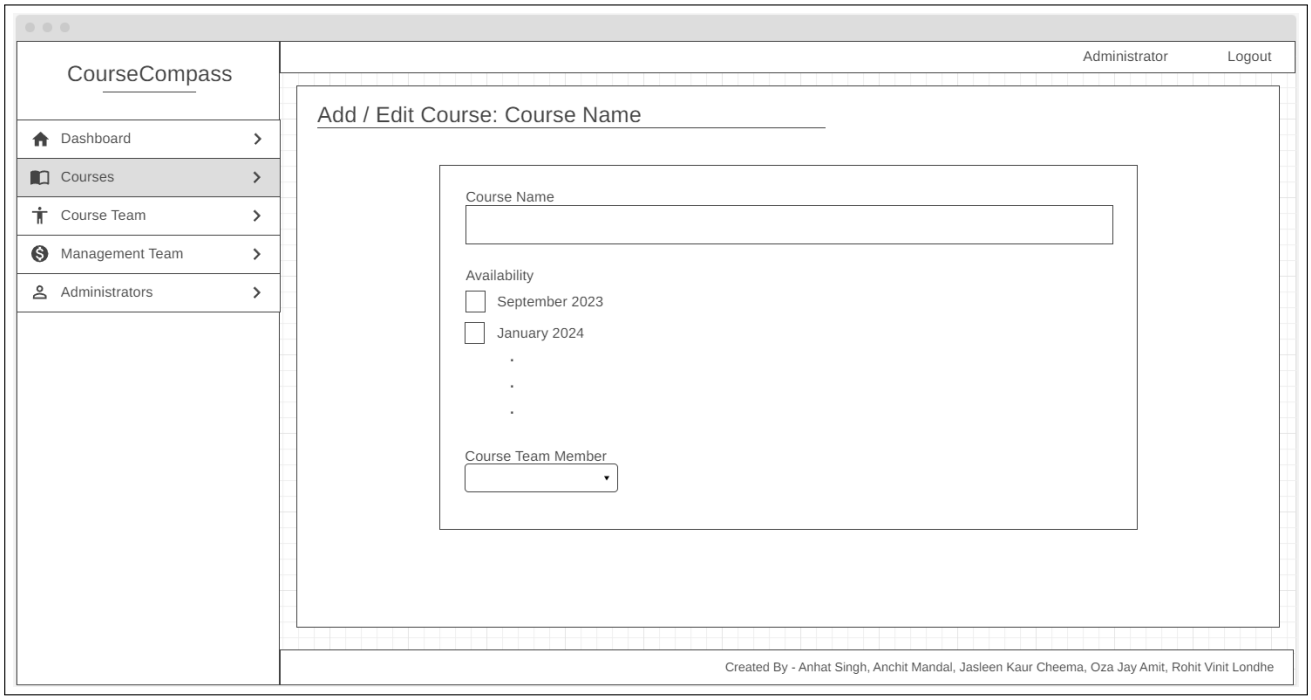


Figure 12: Administrator add / edit courses page

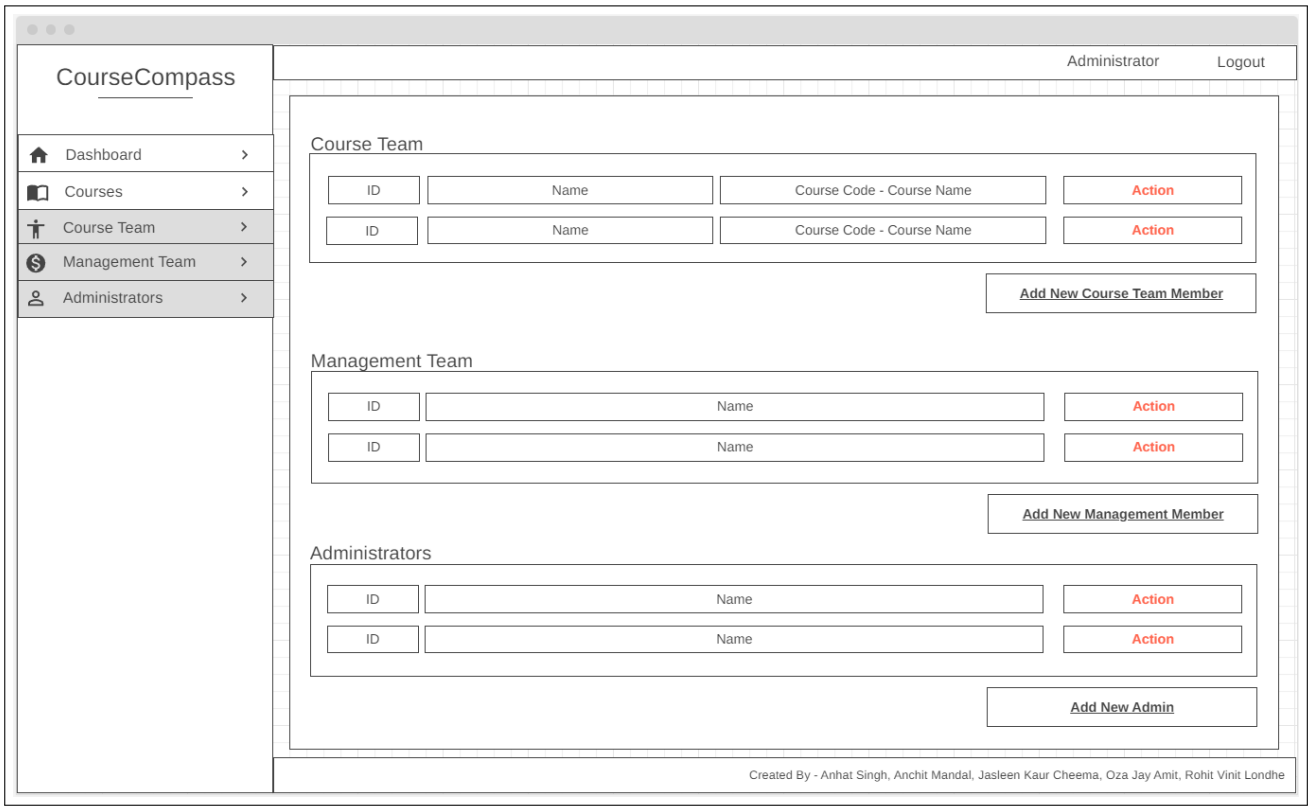


Figure 13: Administrator Page to view course team members, IITM management members and other admins

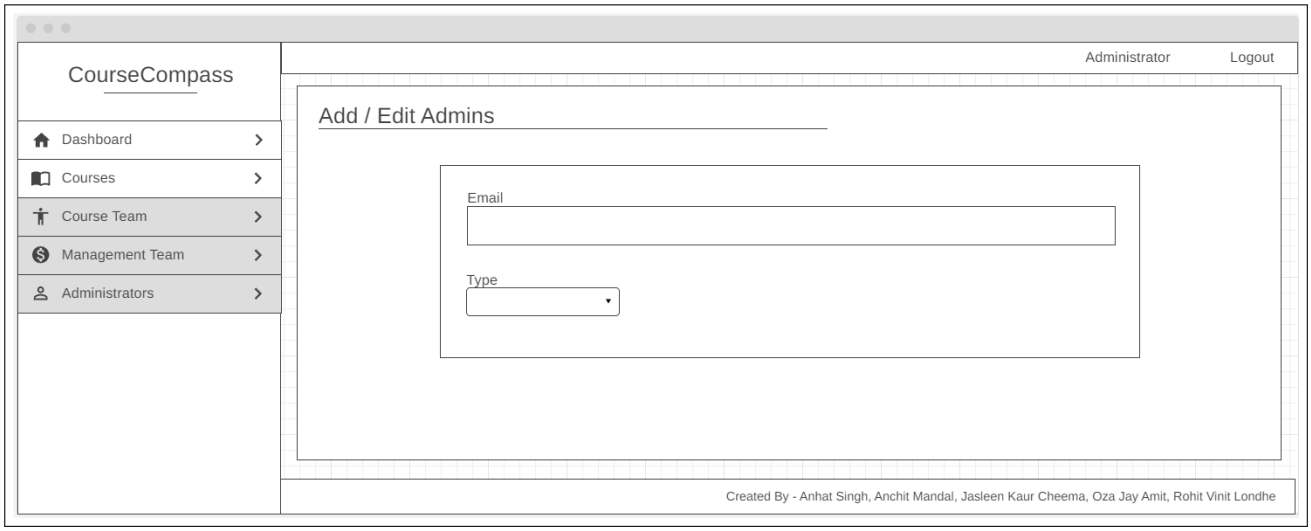


Figure 14: Adding / Editing course team members, IITM management members and other admins

2.2.4 *Course Team Member Pages' Wireframes*

Figure 15: Course team members' Dashboard

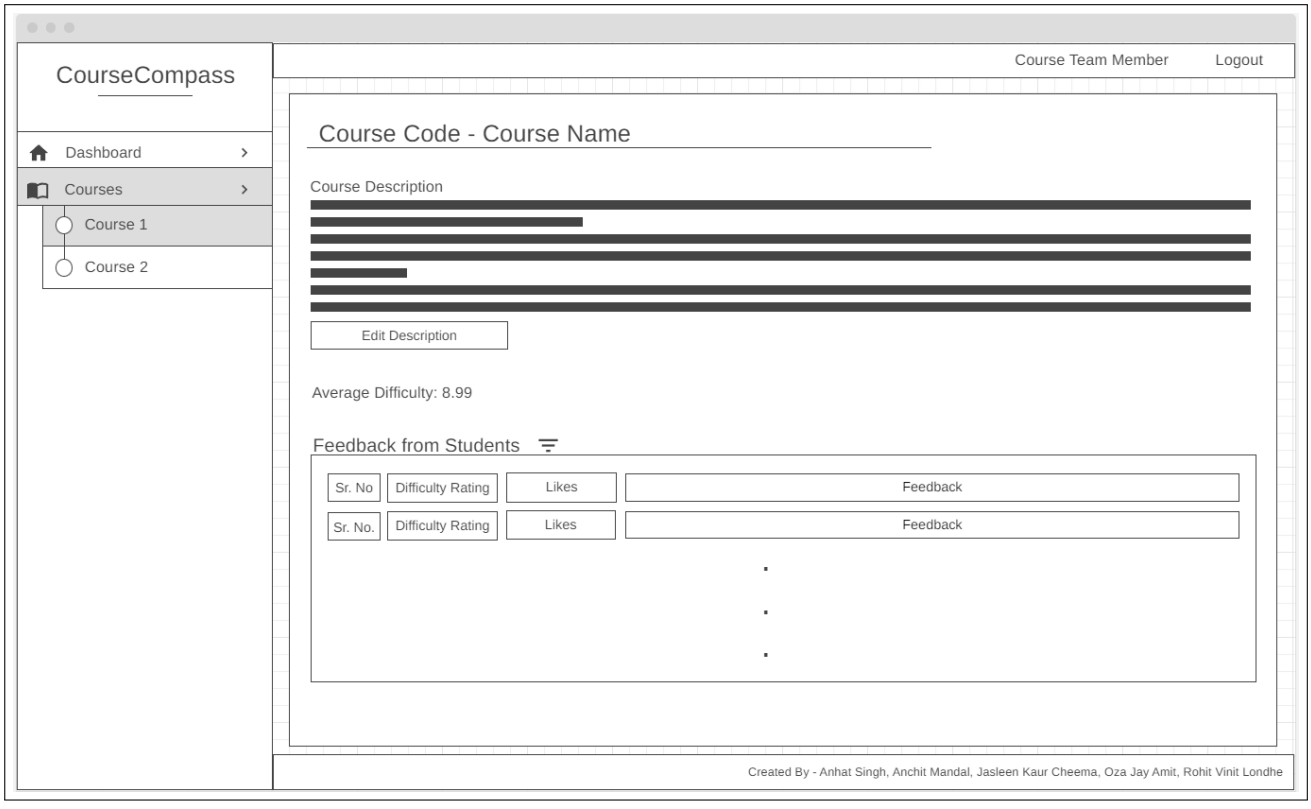


Figure 16: A page for Course Team Members to view feedback given by students

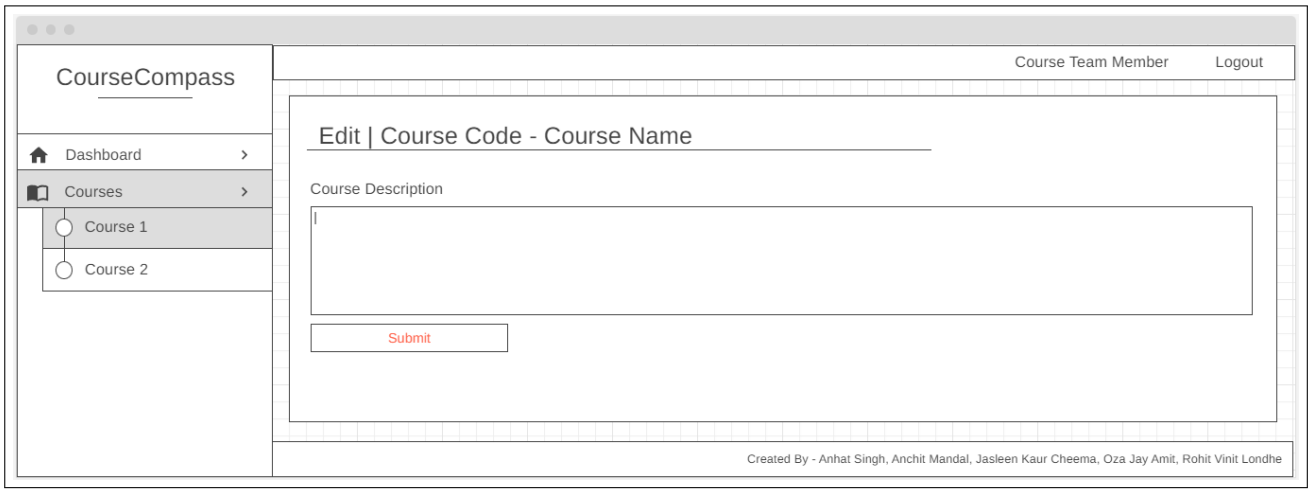


Figure 17: A page for Course team members for adding / updating course description

2.2.5 IITM Management Pages' Wireframes

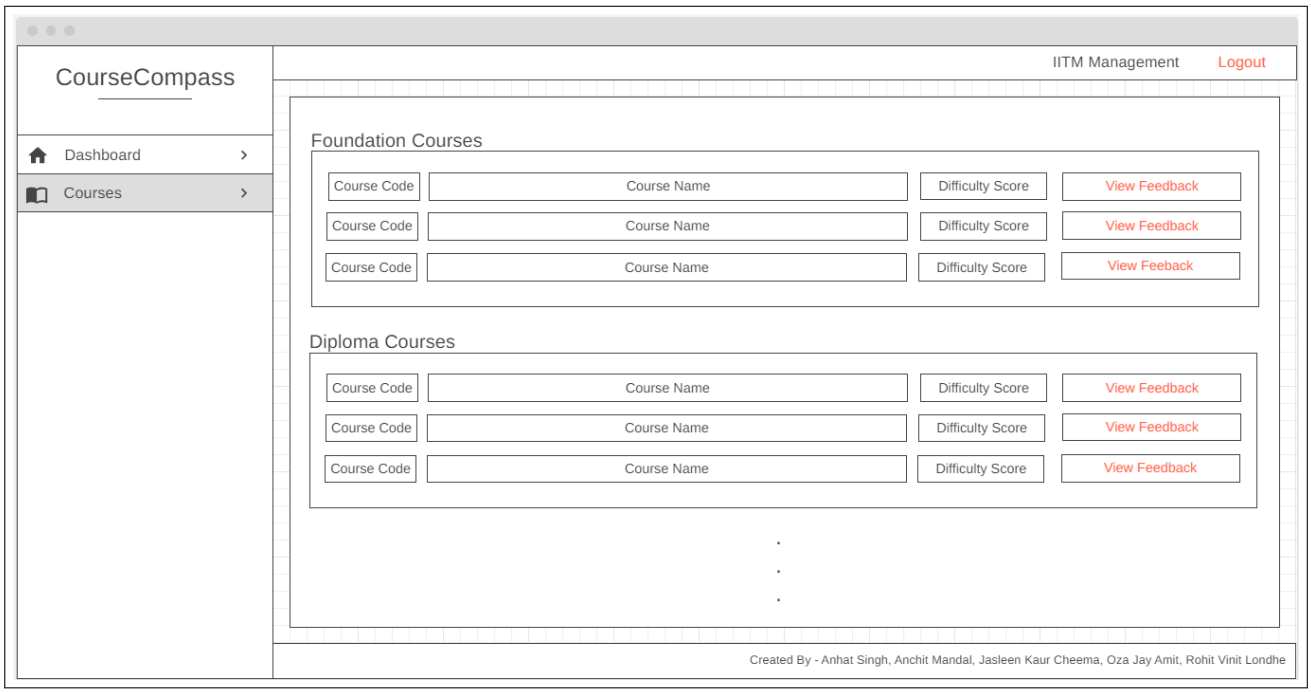


Figure 18: IITM Management Course Feedback page

2.3 Usability Design Guidelines & Heuristics

The low-fidelity wireframes have been designed with common Design guidelines (Effectivess, Effeciency, 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.

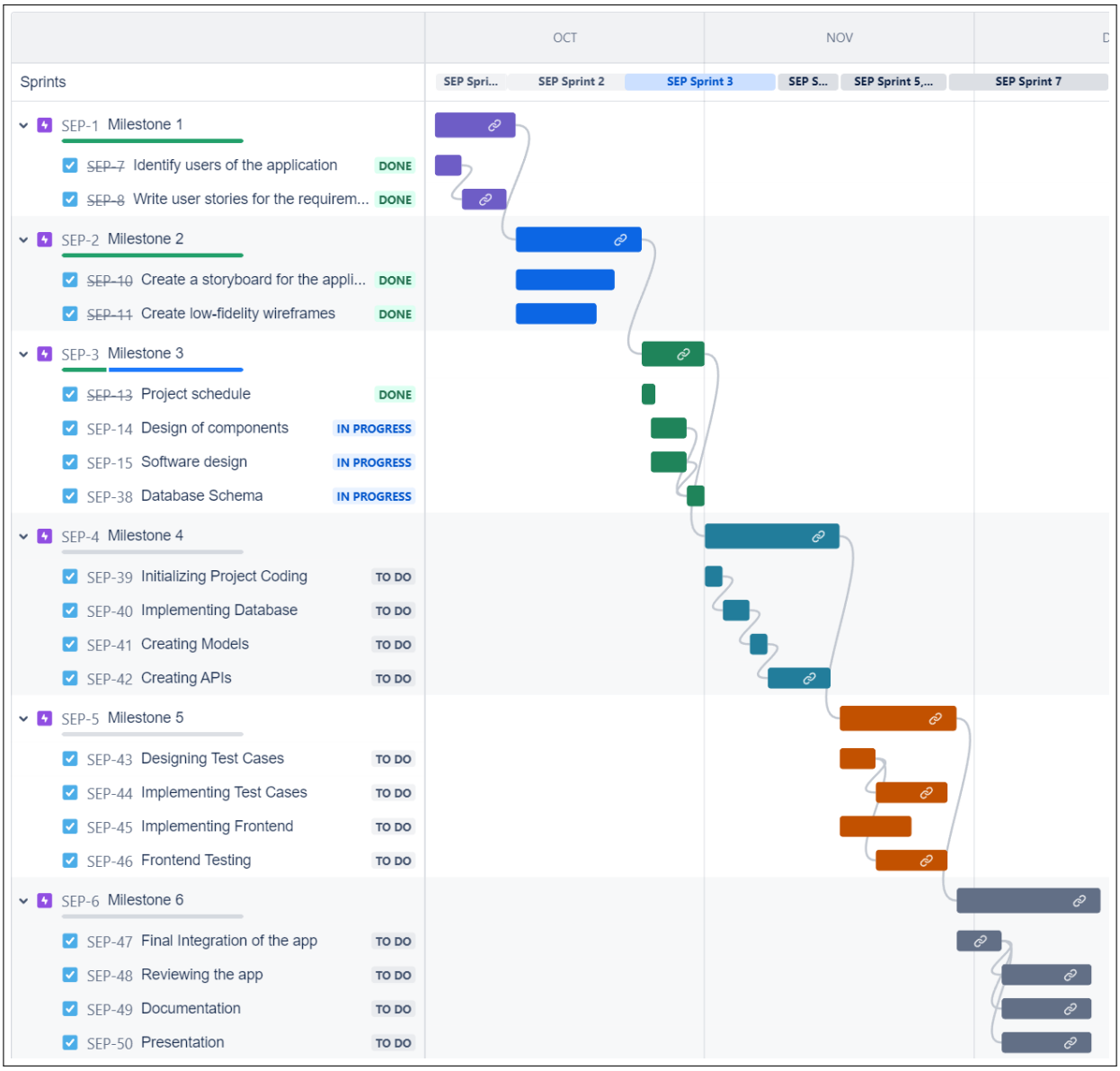


Figure 19: Complete Project Timeline

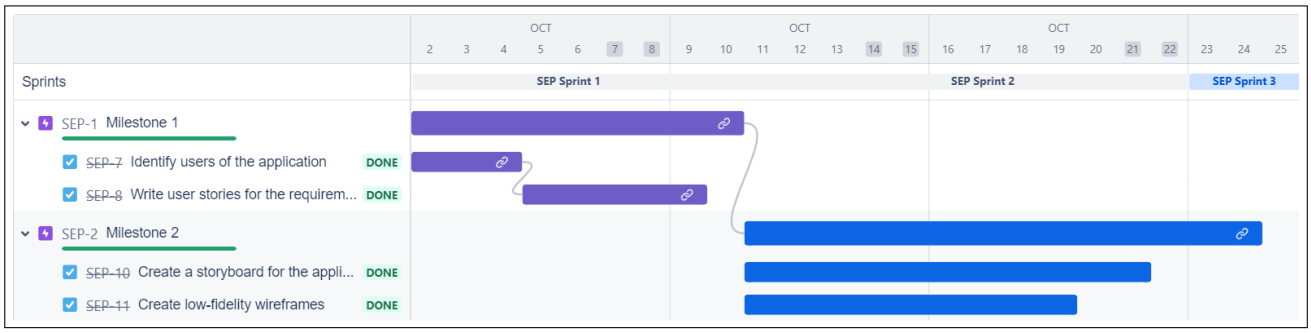


Figure 20: Milestone 1 and 2 Schedule

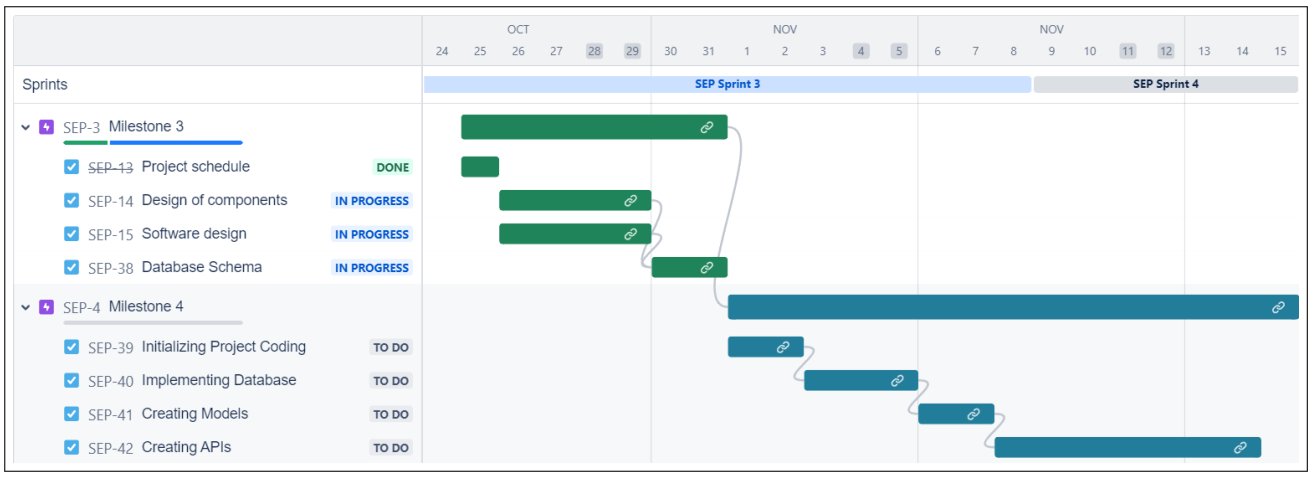


Figure 21: Milestone 3 and 4 Schedule

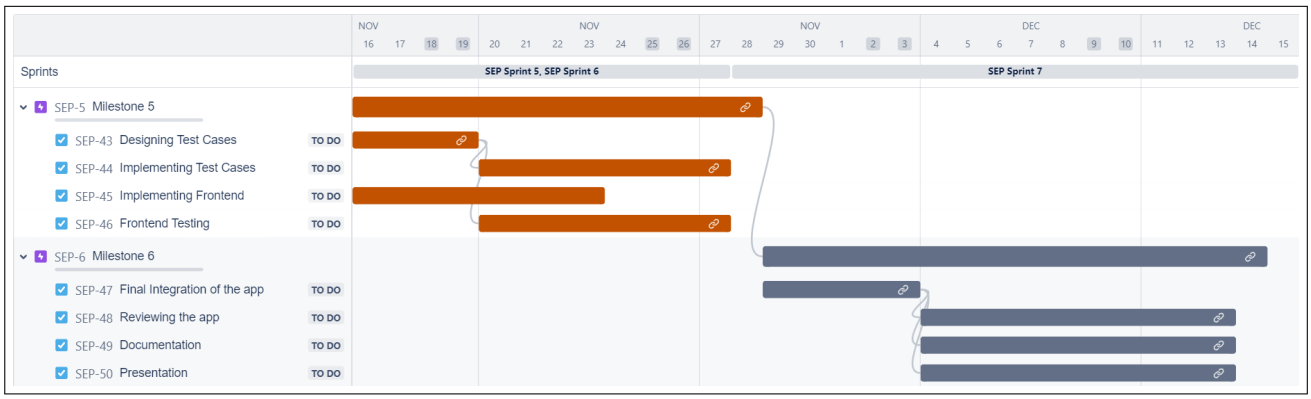


Figure 22: Milestone 5 and 6 Schedule

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/yyy)	End Date (dd/mm/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.

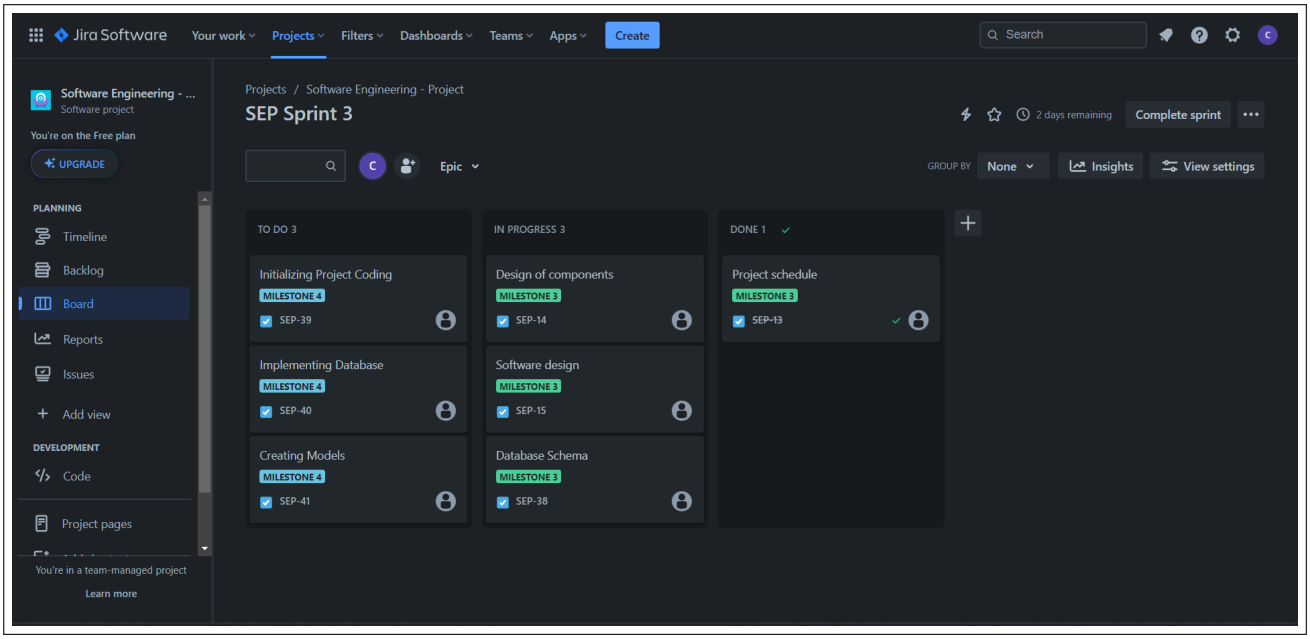


Figure 23: Sprint Board in Jira

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 retuen 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.

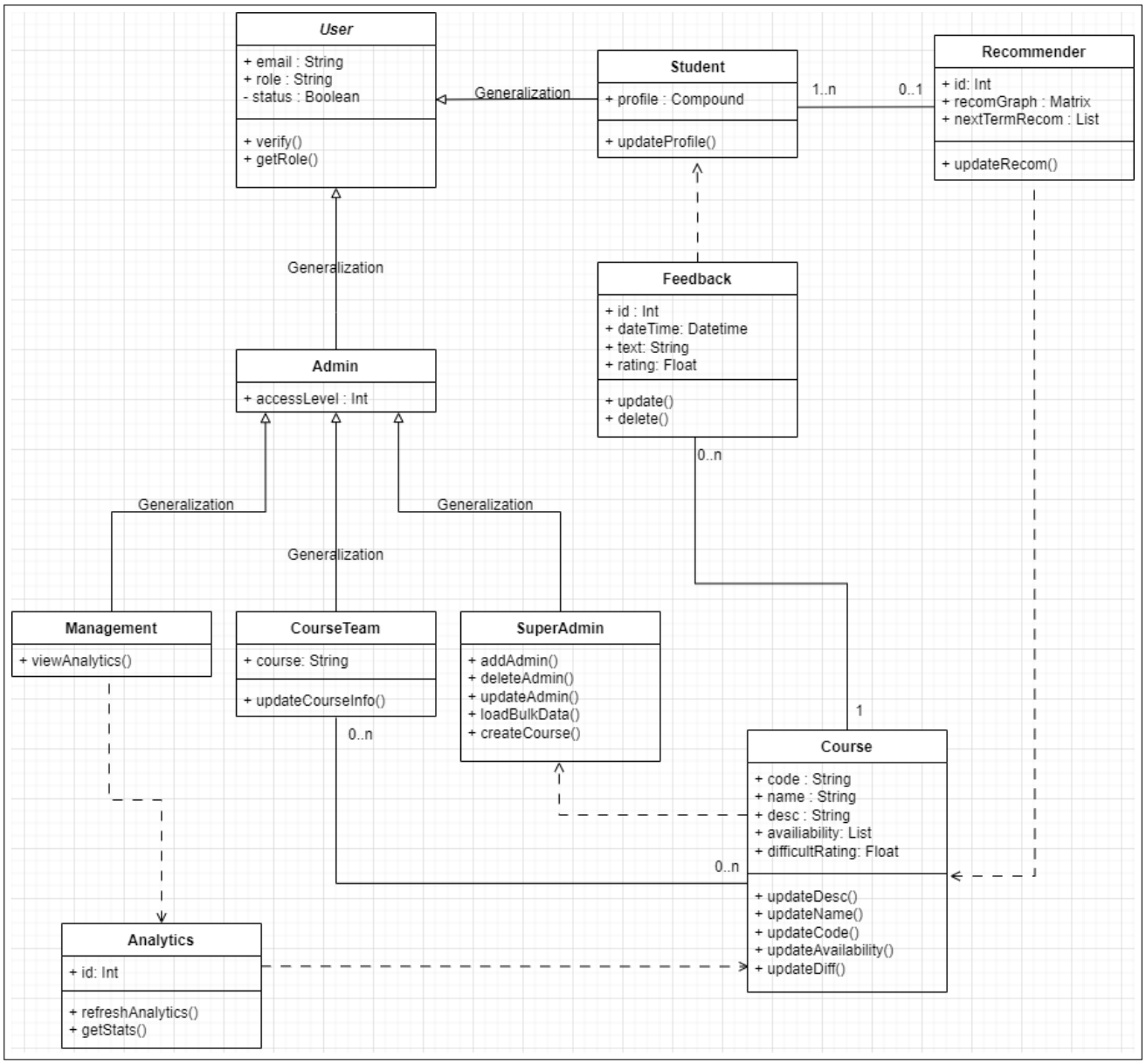


Figure 24: Basic Class Diagram

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**: 32 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.

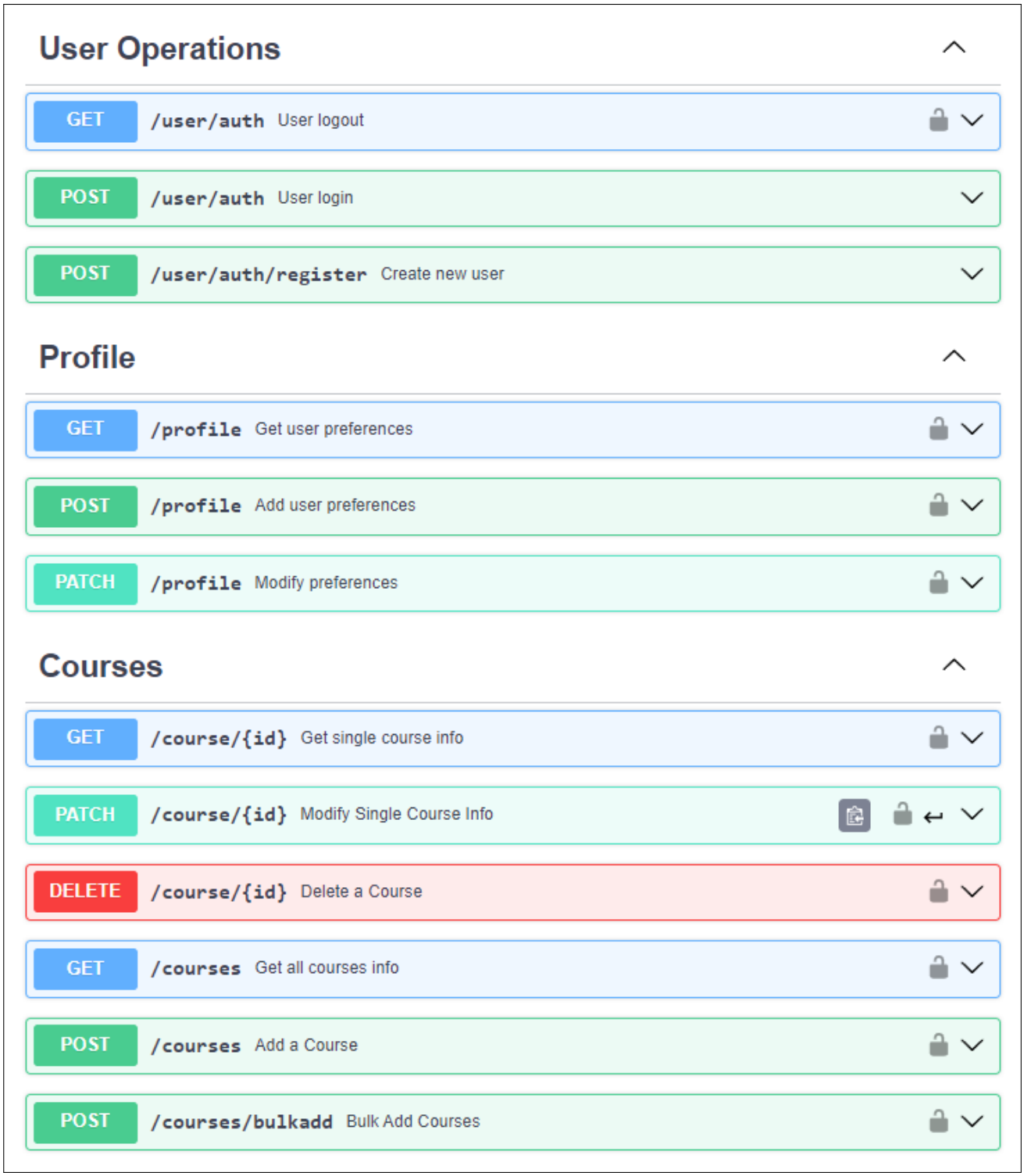


Figure 25: API Endpoints List 1

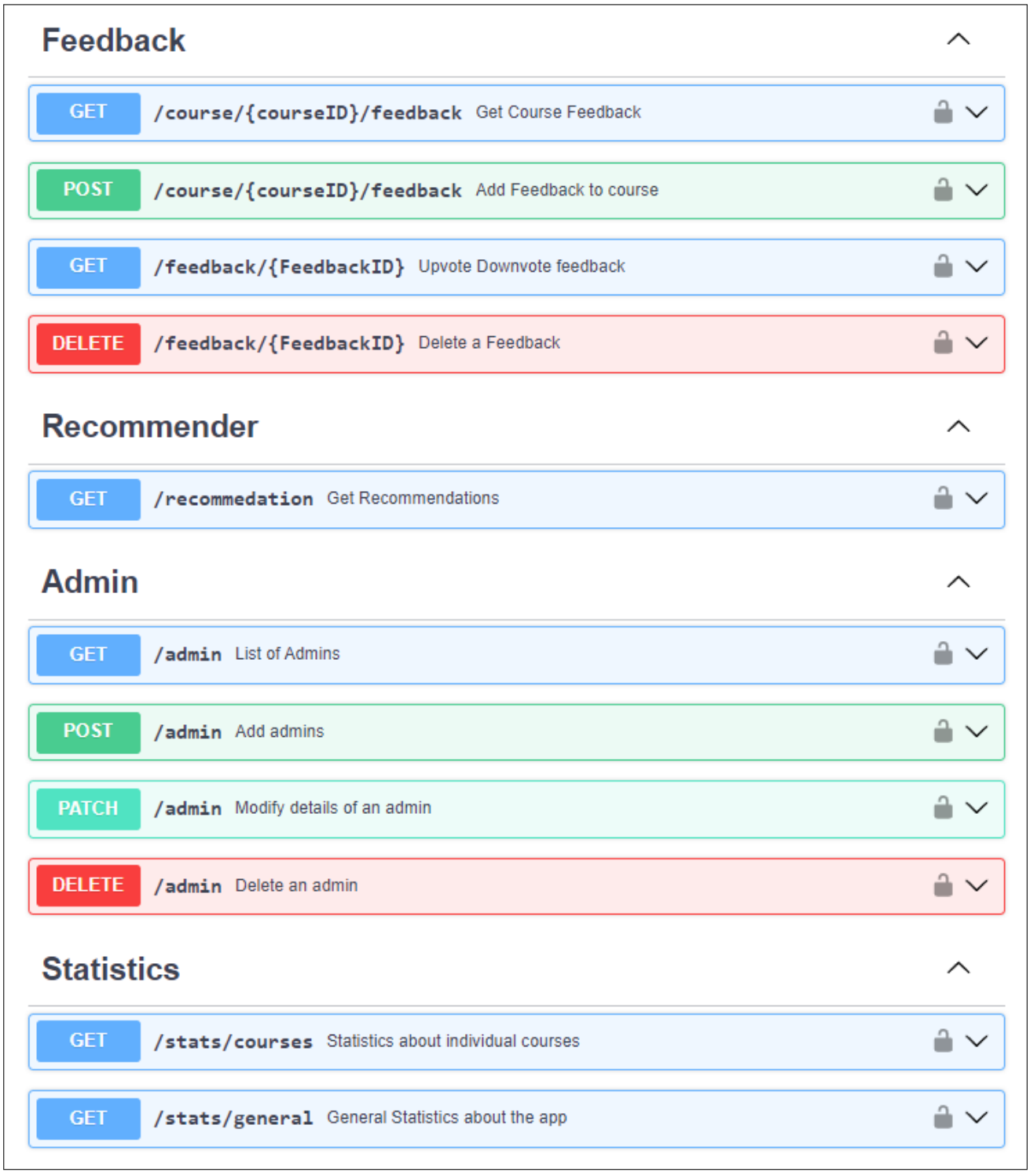


Figure 26: API Endpoints List 2

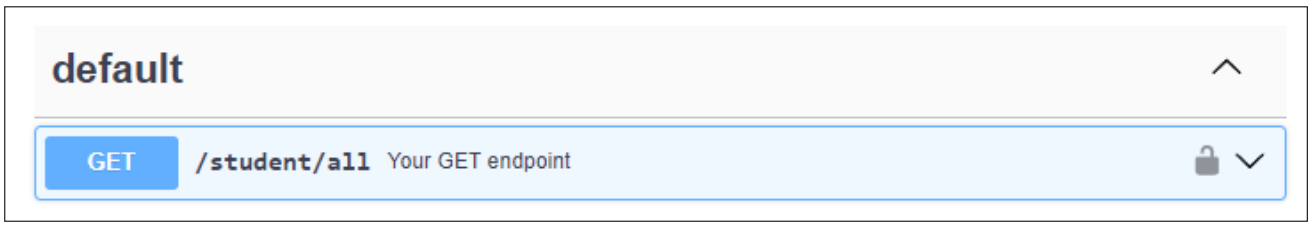


Figure 27: API Endpoints List 3

5 Test Cases for API Endpoints

5.1 A pic of implemented test case

```python
import pytest
import requests, json

BASE_URL = "http://localhost/api/v1"
token = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4f

# Test Case 1
def test_profile_endpoint_correct_info_present():
    response = requests.get(f"{BASE_URL}/profile", headers={'Accept': 'application/json', 'Authorization': f"Bearer {token}"})
    assert response.status_code == 200

# Test Case 2
def test_profile_endpoint_auth_token_missing():
    response = requests.get(f"{BASE_URL}/profile", headers={'Accept': 'application/json'})
    assert response.status_code == 401

# Test Case 3
def test_profile_endpoint_all_details_correct_and_present():
    input_dict={"pic": "ABCD.png","maximum_courses_in_a_term": 3,"current_degree_level": "diploma","dp_or_ds": "both","current_courses": ["B
    data = json.dumps(input_dict)
    response = requests.post(f"{BASE_URL}/profile", data=data, headers={'Accept': 'application/json', 'Authorization': f"Bearer {token}"})
    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

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

Output Response: application/json

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

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

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

Output Response: application/json


```
{
   "email": "abc@xyz.com",
   "role": "student",
   "bio": {
     "maximum_courses_in_a_term": 3,
     "current_degree_level": "foundation",
     "dp_or_ds": "ds",
     "current_courses": [
       "BSHS1001"
     ],
     "completed_courses": [
       {
         "id": "        BSMA1001",
         "marks": 70
       }
     ]
   }
}
```

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 87.9
    }
  ]
}'
```

Output Response: application/json

```
{
  "message": "created",
  "status": "success"
}
```

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

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

Output Response: application/json

```
404
{
  "message": "The requested resource was not found on this server",
  "status": "error"
}
```

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

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

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCDE.png",
  "maximum_courses_in_a_term": 4,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 92
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 4,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 90
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCDE.png",
  "maximum_courses_in_a_term": 2,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 90
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCDE.png",
  "maximum_courses_in_a_term": 2,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1004"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 90
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2006",
      "marks": 87.9
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 1,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 87.9
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 5,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 87.9
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "graduated",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 87.9
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "diploma",
  "dp_or_ds": "dk",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 87.9
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": -9
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 130
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "pic": "ABCD.png",
  "maximum_courses_in_a_term": 3,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001"
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "maximum_courses_in_a_term": 4,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 99
    }
  ]
}'
```

Output Response: application/json

```
{
  "message": "accepted",
  "status": "success"
}
```

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --data '{
  "maximum_courses_in_a_term": 4,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 99
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "maximum_courses_in_a_term": 4,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 99
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/profile \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "maximum_courses_in_a_term": 4,
  "current_degree_level": "diploma",
  "dp_or_ds": "both",
  "current_courses": [
    "BSHS1001"
  ],
  "completed_courses": [
    {
      "id": "BSC2001",
      "marks": 99
    }
  ]
}'
```

Output Response: application/json

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

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

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

Output Response: application/json


```
[
  {
    "name": "ABCD",
    "email": "ABCD@EFGH.com",
    "role": "admin",
    "created_at": "2012-04-23T18:25:43.511Z"
  },
  {
    "name": "IJKL",
    "email": "IJKL@EFGH.com",
    "role": "im",
    "created_at": "2012-04-23T18:25:43.511Z"
  }
]
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

```
{
  "message": "created",
  "status": "success"
}
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

Test Case 28

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

Input Request

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/admins \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "email": "a@b",
  "role": "im"
}'
```

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json


```
{
  "message": "accepted",
  "status": "success"
}
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

```
curl --request PATCH \
  --url http://localhost/api/v1/admins \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "email": "IJKL@EFGH.com",
}'
```

Output Response: application/json

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

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

```
{
  "auth": {
    "message": "Login successful",
    "authToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwi
  },
  "profile": {
    "name": "A Student",
    "email": "abc@xyz.com",
    "role": "student"
  }
}
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

Test Case 50

Endpoint	http://localhost/api/v1/user/auth/register
Category	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

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

Output Response: application/json

```
{
  "message": "created",
  "status": "success"
}
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json


```
{
  "no_courses": 3,
  "upcoming_term": [
    "BSC2001",
    "BSC2002",
    "BSC2003"
  ],
  "matrix_order": [
    "BSC2001",
    "BSC2002",
    "BSC2003",
    "BSC2004",
    "BSC2005",
    "BSC2006",
    "BSC2007",
    "BSC2008"
  ],
  "recommendation_matrix": [
    [
      1,
      0,
      1,
      0,
      0,
      0,
      1,
      0
    ],
    [
      1,
      0,
      1,
      0,
      0,
```

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

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 \
2   --url http://localhost/api/v1/recommedation \
3   --header 'Accept: application/json' \
4   --header 'Authorization: Bearer ***'
```

Output Response: application/json

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

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

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

Output Response: application/json


```
{
  "id": "BSC2001",
  "name": "Course 1",
  "description": "This is course 1",
  "difficulty_rating": 8.92,
  "level": "bsc",
  "dp_or_ds": "ds",
  "credits": 4,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json


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

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

```
curl --request PATCH \
  --url http://localhost/api/v1/course/BSC2002 \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 3,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

```
{
  "message": "accepted",
  "status": "success"
}
```

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

```
curl --request PATCH \
  --url http://localhost/api/v1/course/BSC2002 \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "na": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 3,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request PATCH \
  --url http://localhost/api/v1/course/BSC2002 \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --data '{
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 3,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request PATCH \
  --url http://localhost/api/v1/course/BSC2002 \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 3,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request PATCH \
  --url http://localhost/api/v1/course/BSC20 \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 3,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

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

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

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

Output Response: application/json

```

```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json


```
{
  "courses": [
    {
      "id": "BSC2001",
      "name": "Course 1",
      "description": "This is course 1",
      "difficulty_rating": 8.92,
      "level": "Bsc",
      "credits": 4,
      "dp_or_ds": "dp",
      "pre_req": [
        "BSC2001",
        "BSC2002"
      ],
      "co_req": [
        "BSC2001",
        "BSC2002"
      ],
      "availability": [
        "Sept2022",
        "March2002"
      ],
      "instructors": [
        {
          "name": "ABCD",
          "email": "ABCD@efgh.com"
        },
        {
          "name": "EFGH",
          "email": "ijkl@efgh.com"
        }
      ]
    }
  ]
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/courses \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "code": "BSC2001",
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 4,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

```
{
  "message": "created",
  "status": "success"
```

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

```
curl --request POST \
  --url http://localhost/api/v1/courses \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --data '{
  "code": "BSC2001",
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 4,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

```
401
{
  "message": "Bad data sent",
  "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

```
curl --request POST \
  --url http://localhost/api/v1/courses \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "code": "BSC2001",
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 4,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

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

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

```
curl --request POST \
  --url http://localhost/api/v1/courses \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer ***' \
  --header 'Content-Type: application/json' \
  --data '{
  "coe": "BSC2001",
  "name": "Course 1",
  "description": "This is course 1",
  "level": "Degree",
  "dp_or_ds": "dp",
  "credits": 4,
  "pre_req": [
    "BSC2001",
    "BSC2002"
  ],
  "co_req": [
    "BSC2001",
    "BSC2002"
  ],
  "availability": [
    "Sept2022",
    "March2002"
  ],
  "instructors": [
    {
      "name": "ABCD",
      "email": "ABCD@efgh.com"
    },
    {
      "name": "EFGH",
      "email": "ijkl@efgh.com"
    }
  ]
}'
```

Output Response: application/json

```
{
  "message": "Bad Request sent",
  "status": "error"
```

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

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

Output Response: application/json


```
{
  "feedbacks": [
    {
      "id": 1,
      "poster": "21f2000381@ds.study.iitm.ac.in",
      "pic": "myprofilepic.png",
      "rating": 5.6,
      "description": "This is a good course",
      "time": "2012-04-23T18:25:43.511Z"
    }
  ]
}
```

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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

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

Output Response: application/json

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

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.

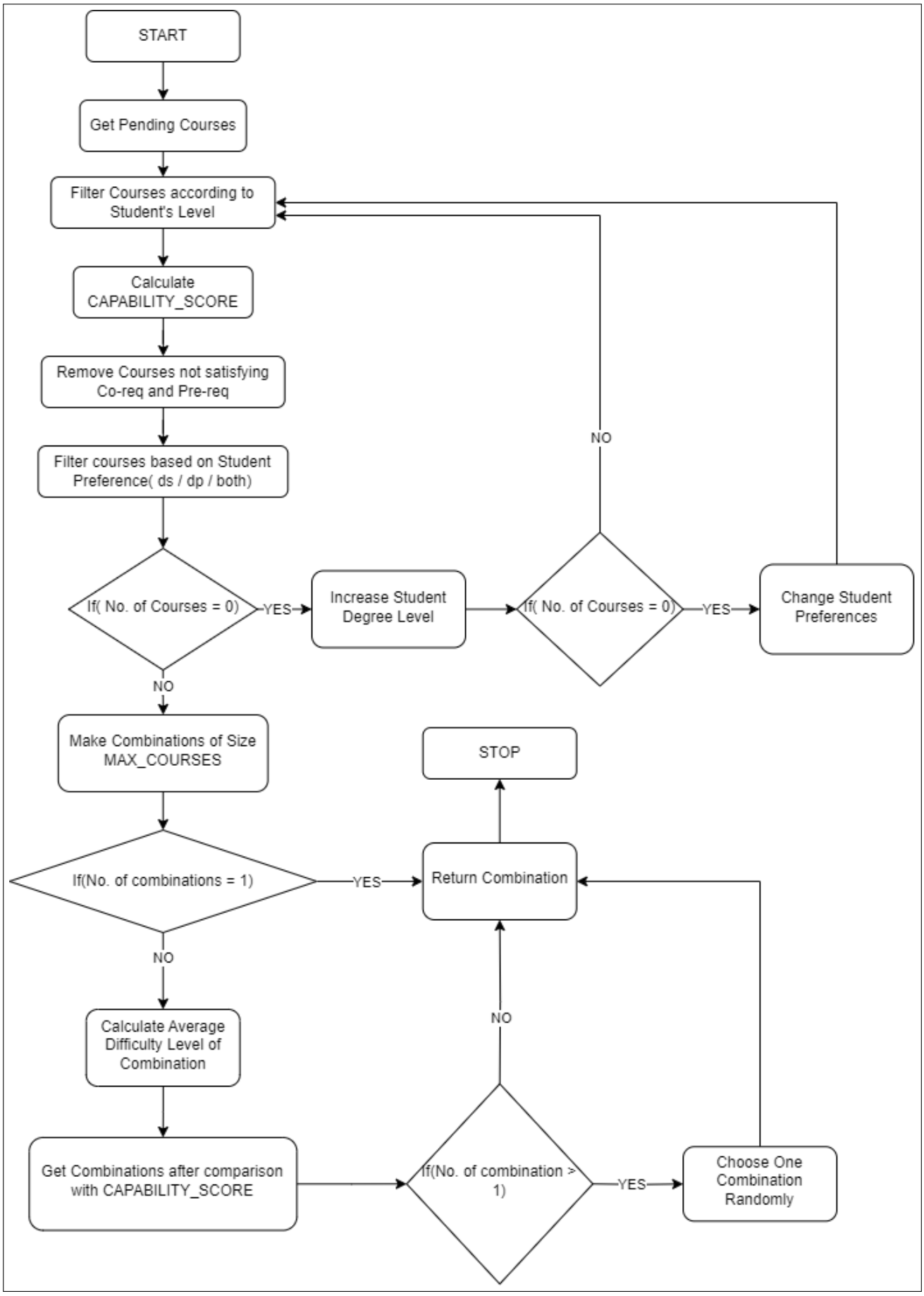


Figure 29: Algorithm used for Recommendations

7 Screenshots

7.1 Login Page

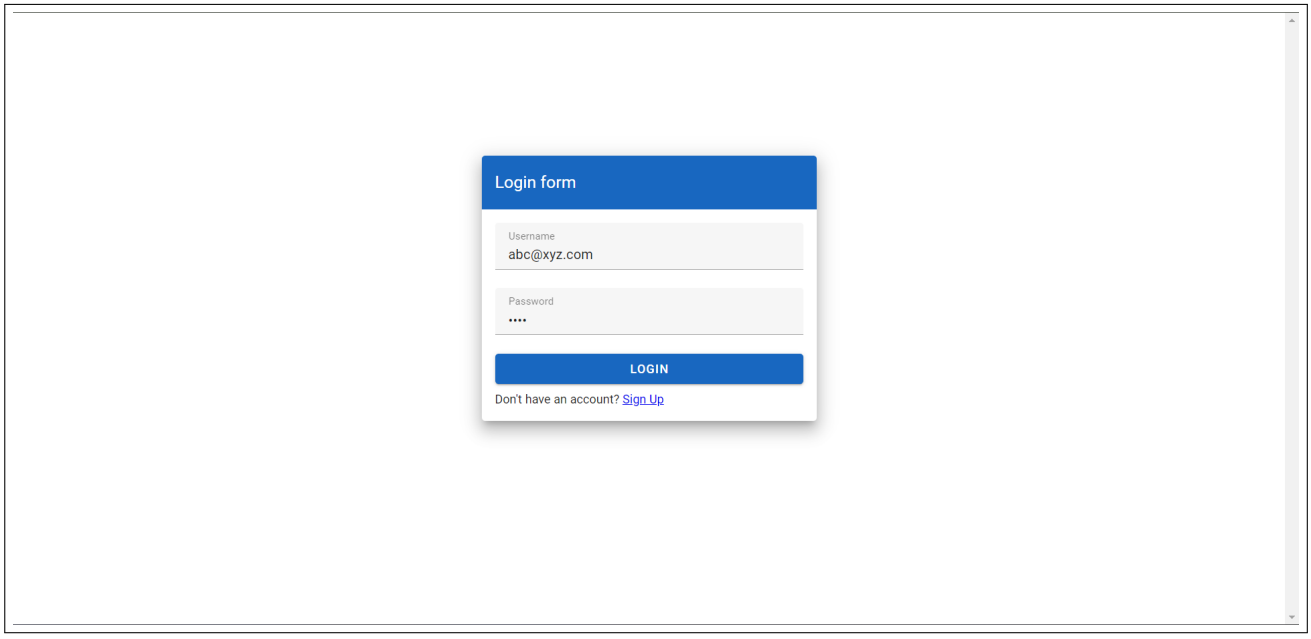


Figure 30: UI Screenshot - Login Page

7.2 Student Pages

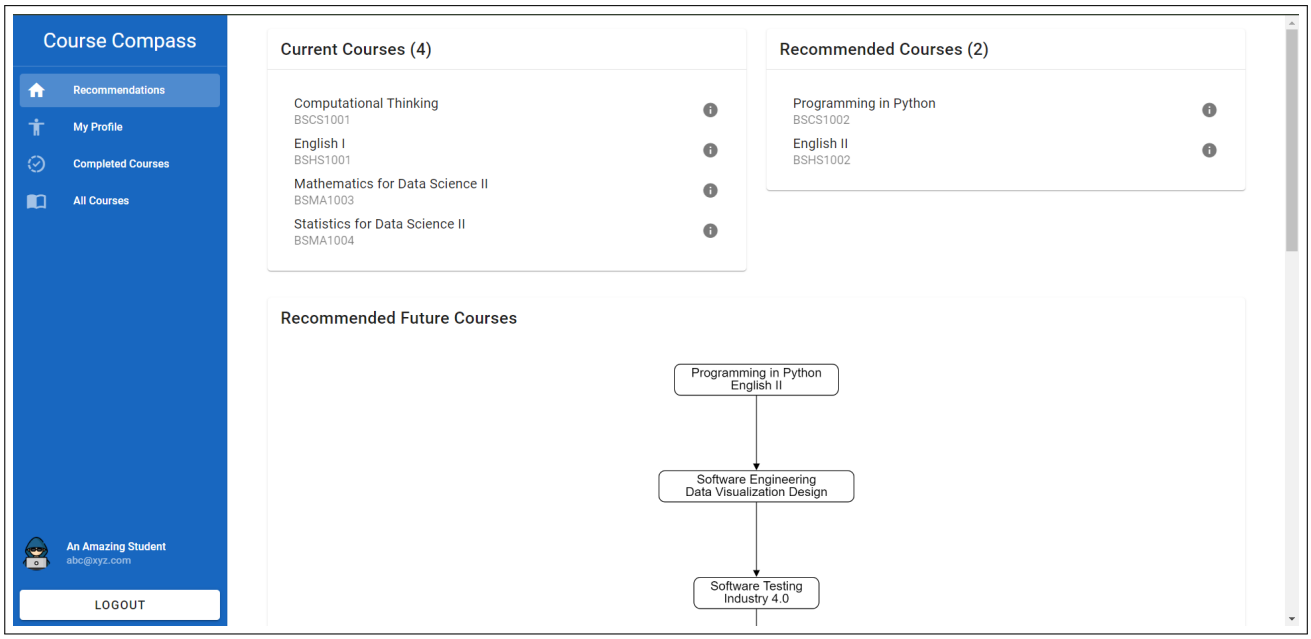


Figure 31: UI Screenshot - Student Home Page

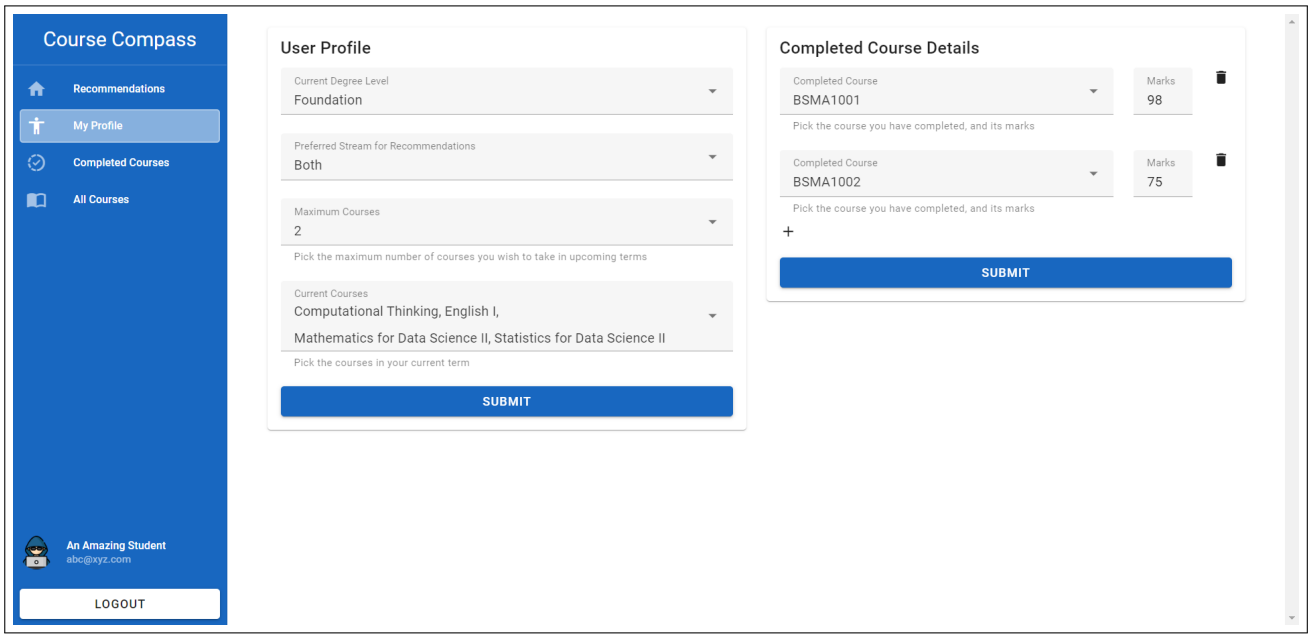


Figure 32: UI Screenshot - Student Profile Page

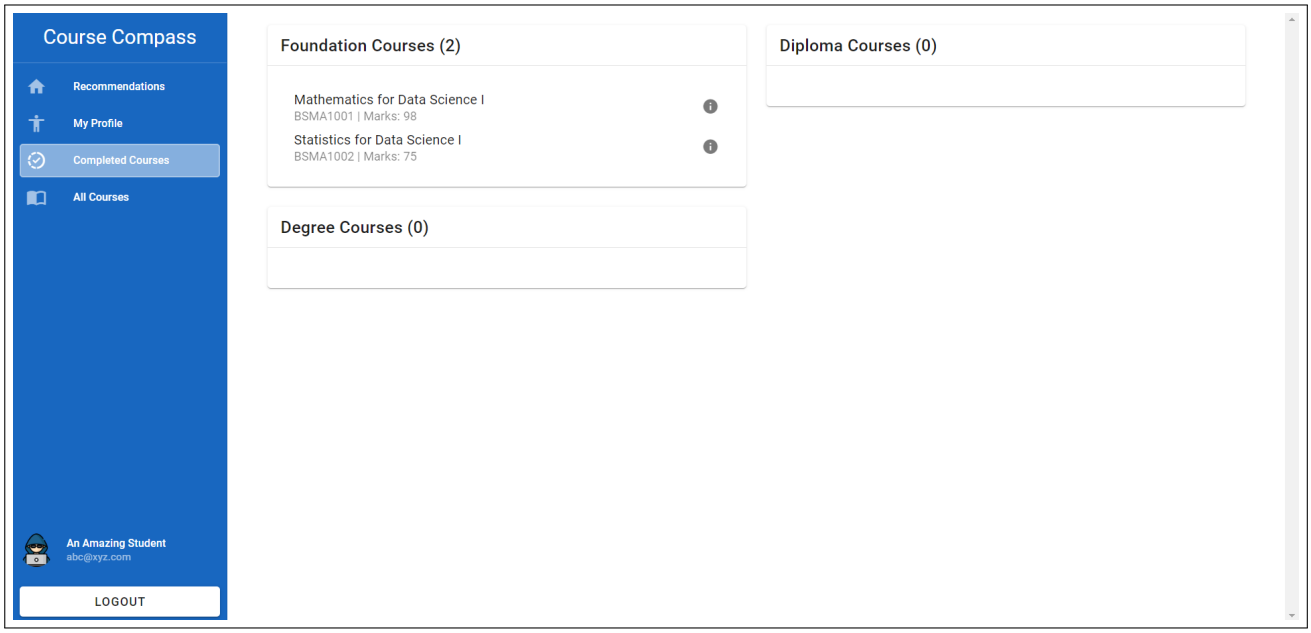


Figure 33: UI Screenshot - Student Completed Courses Page

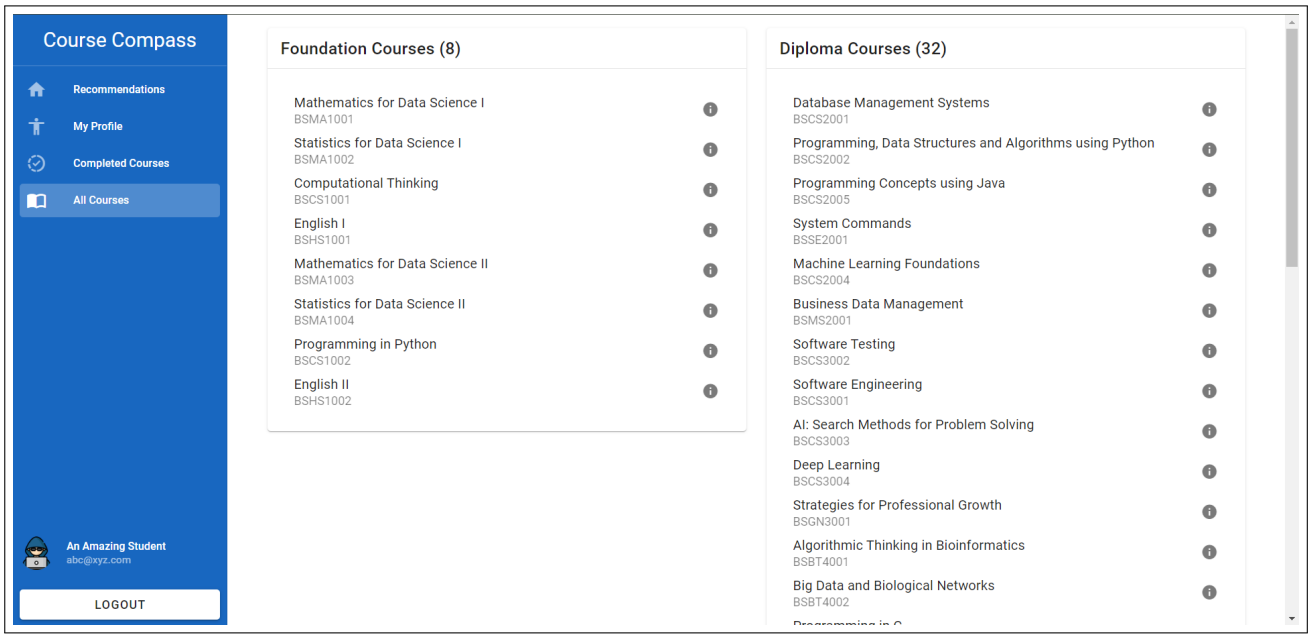


Figure 34: UI Screenshot - Student All Courses Page

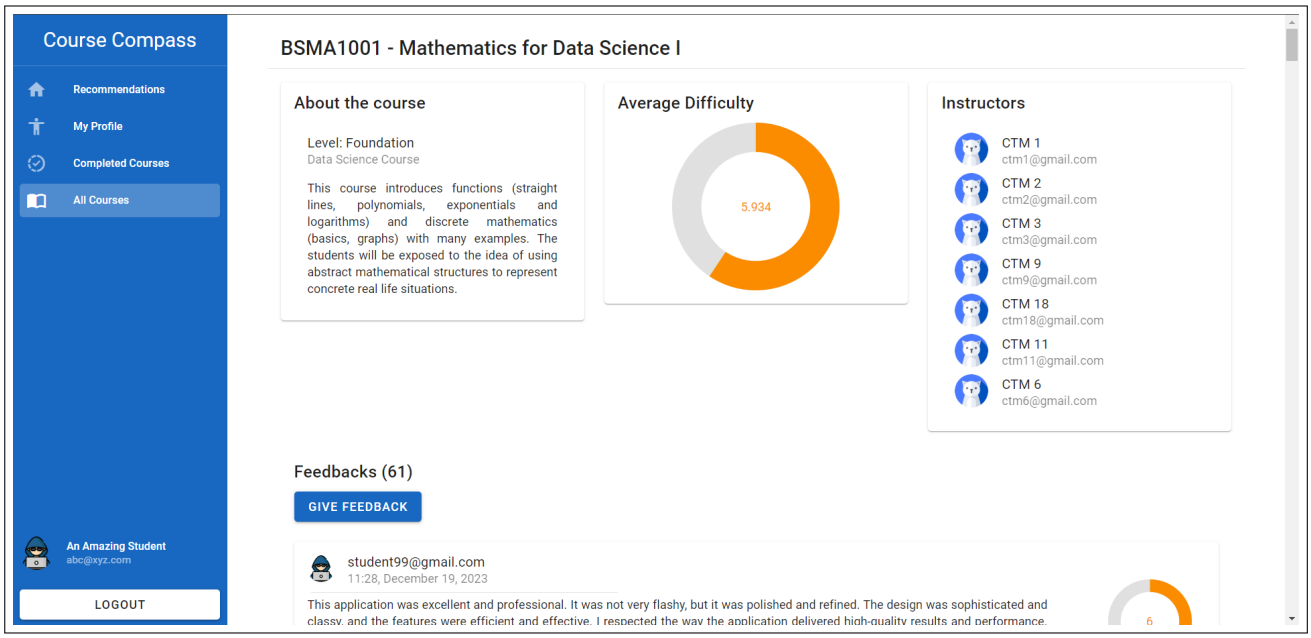


Figure 35: UI Screenshot - Student Each Course Details Page

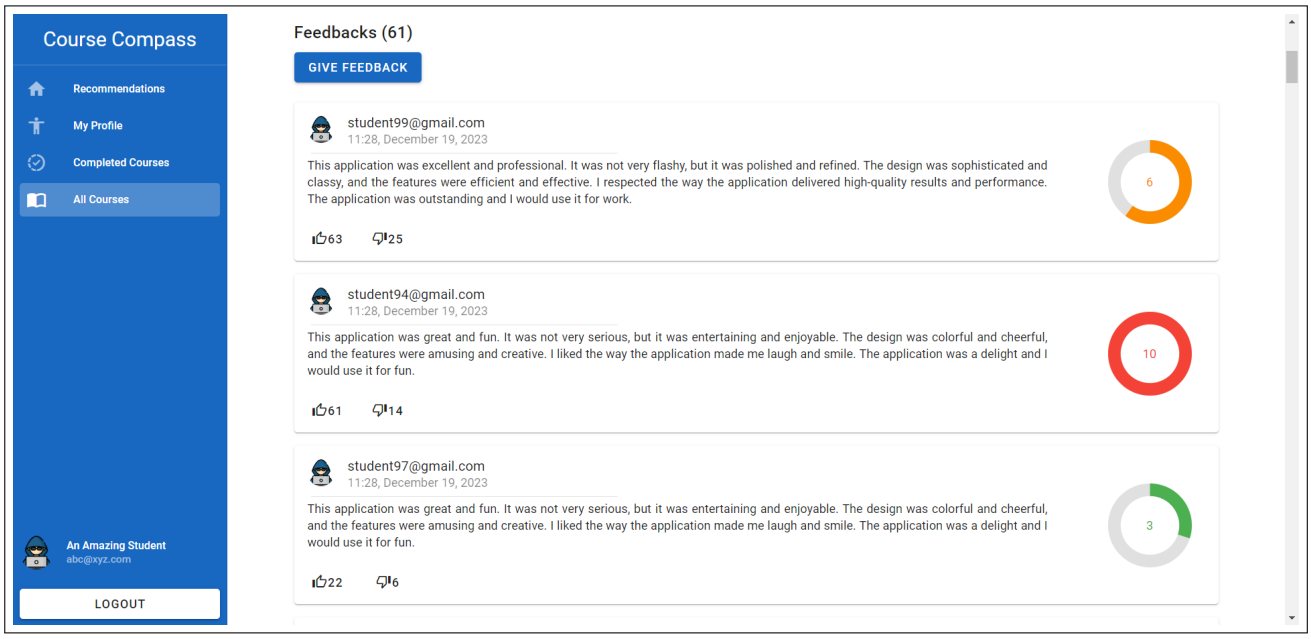


Figure 36: UI Screenshot - Student Each Course Feedback

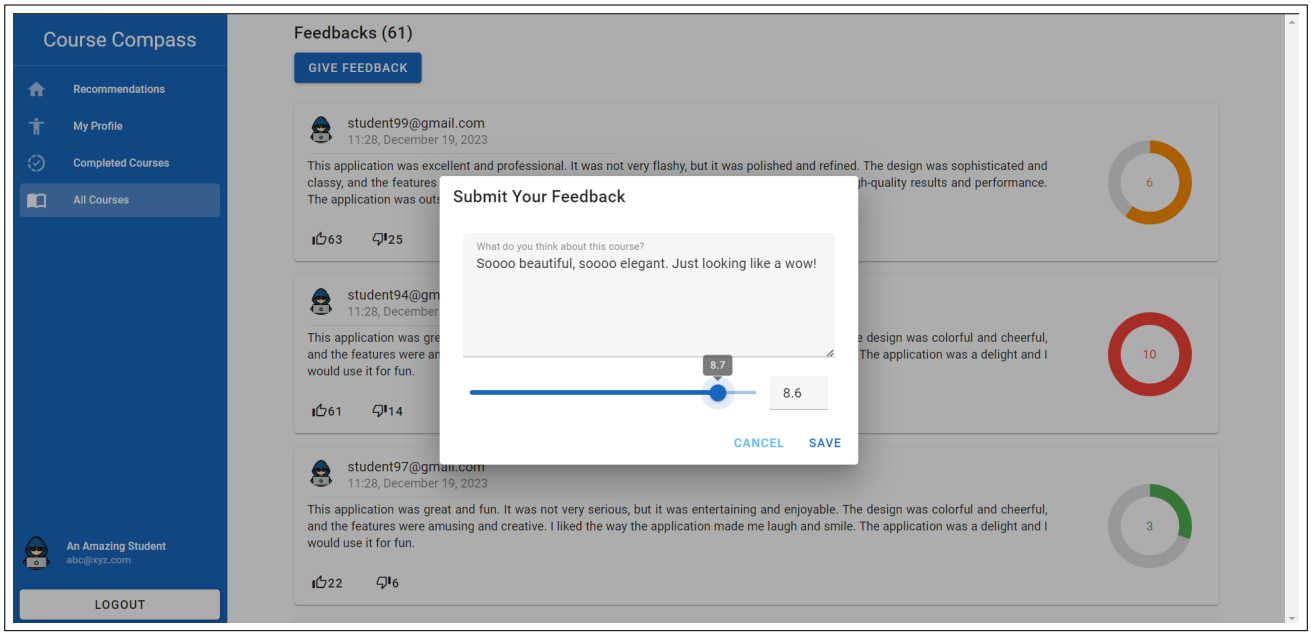


Figure 37: UI Screenshot - Student Each Course Give Feedback

7.3 Admin Pages

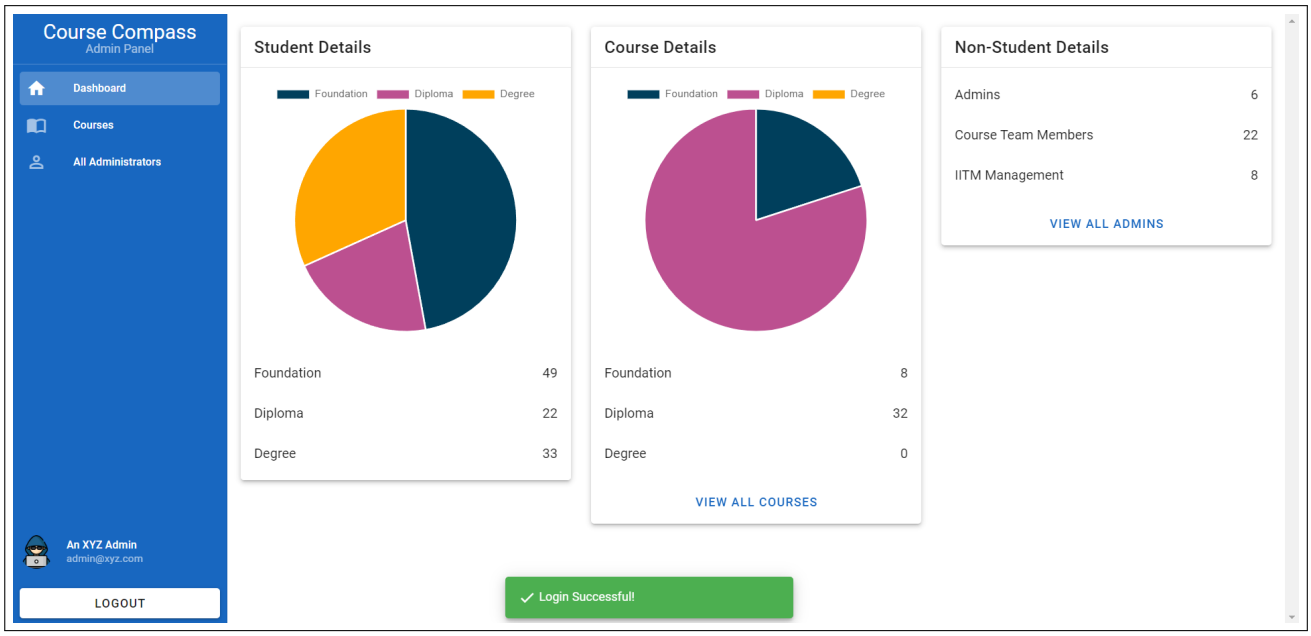


Figure 38: UI Screenshot - Admin Home Page

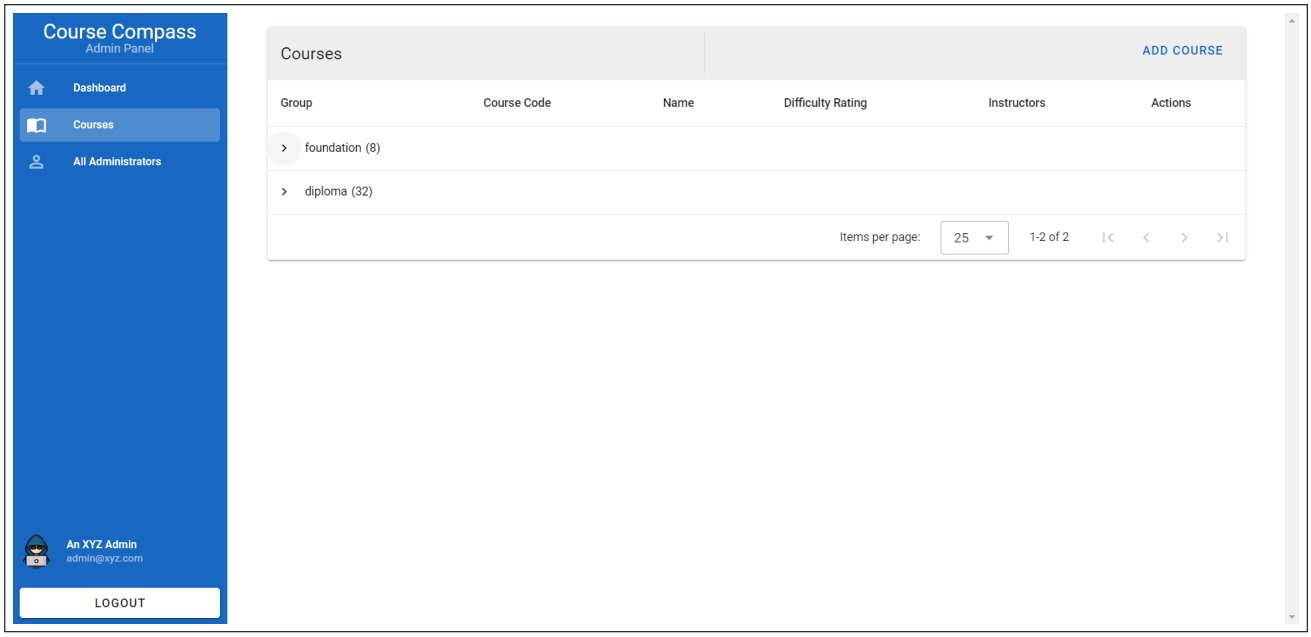


Figure 39: UI Screenshot - Admin Courses Page

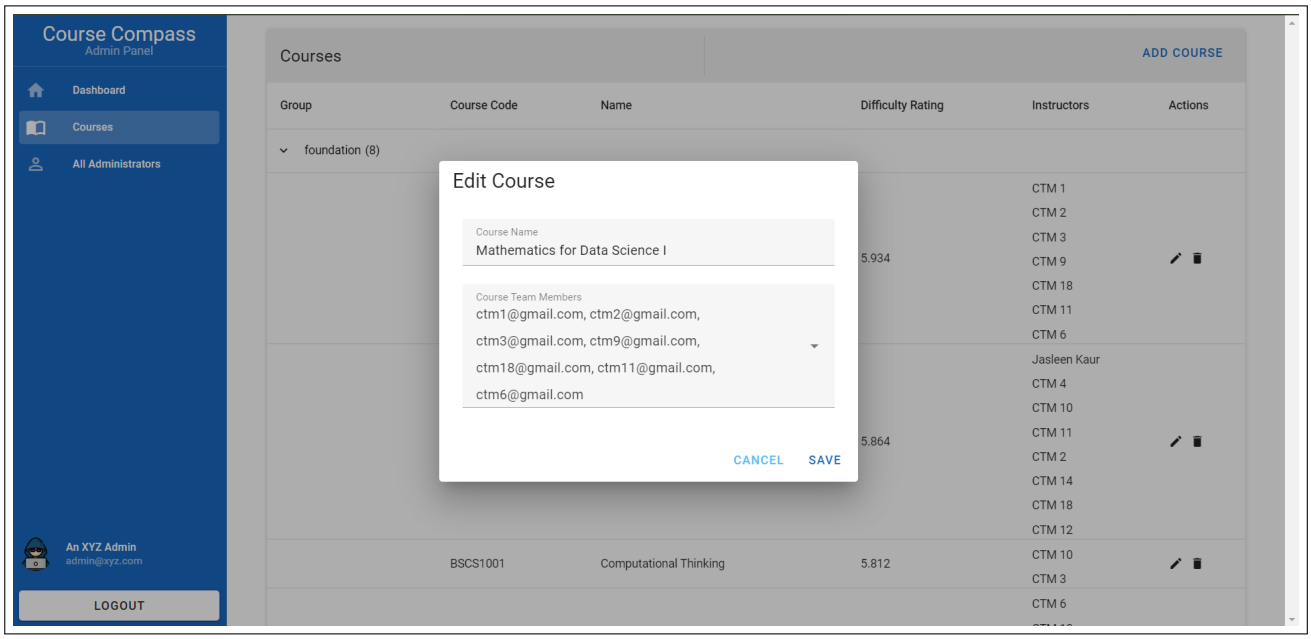


Figure 40: UI Screenshot - Admin Edit Course Info

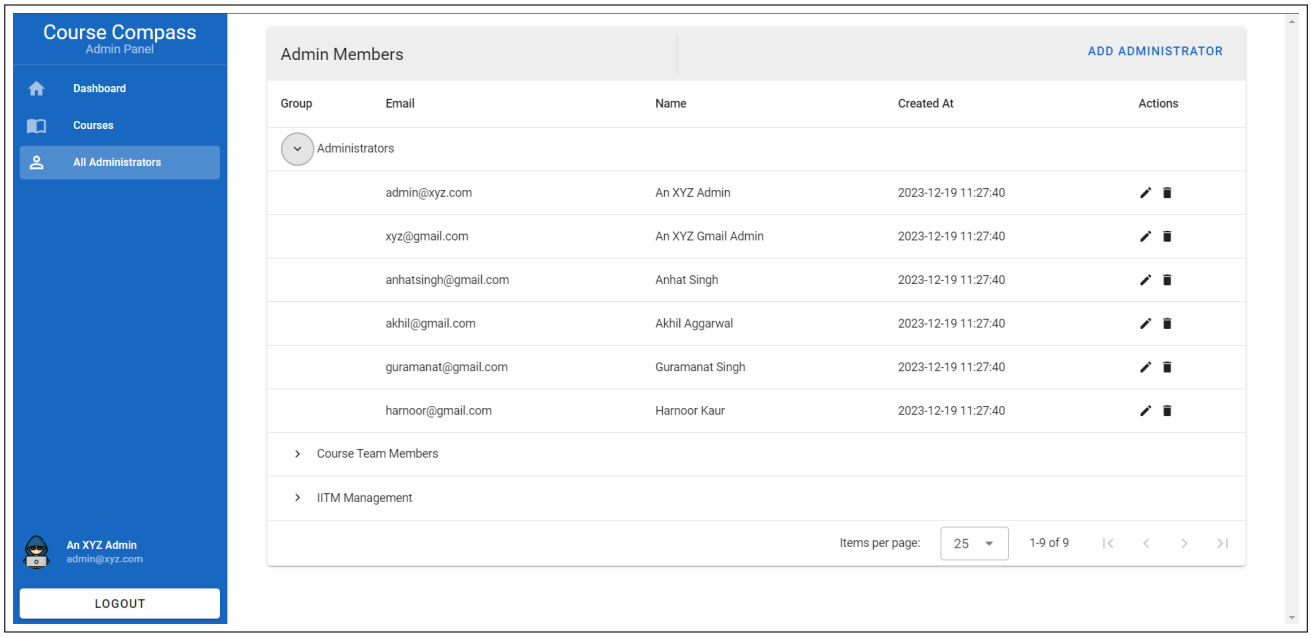


Figure 41: UI Screenshot - List of other admins

7.4 Course Team Member Pages

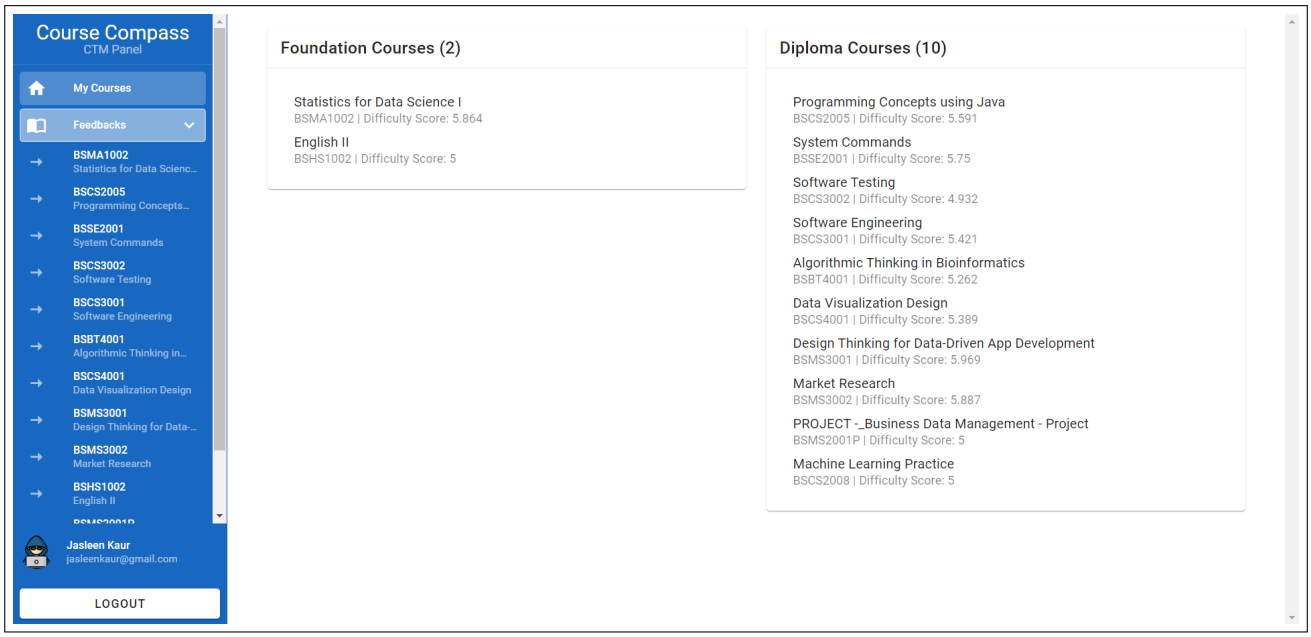


Figure 42: UI Screenshot - Course Team Member Home Page

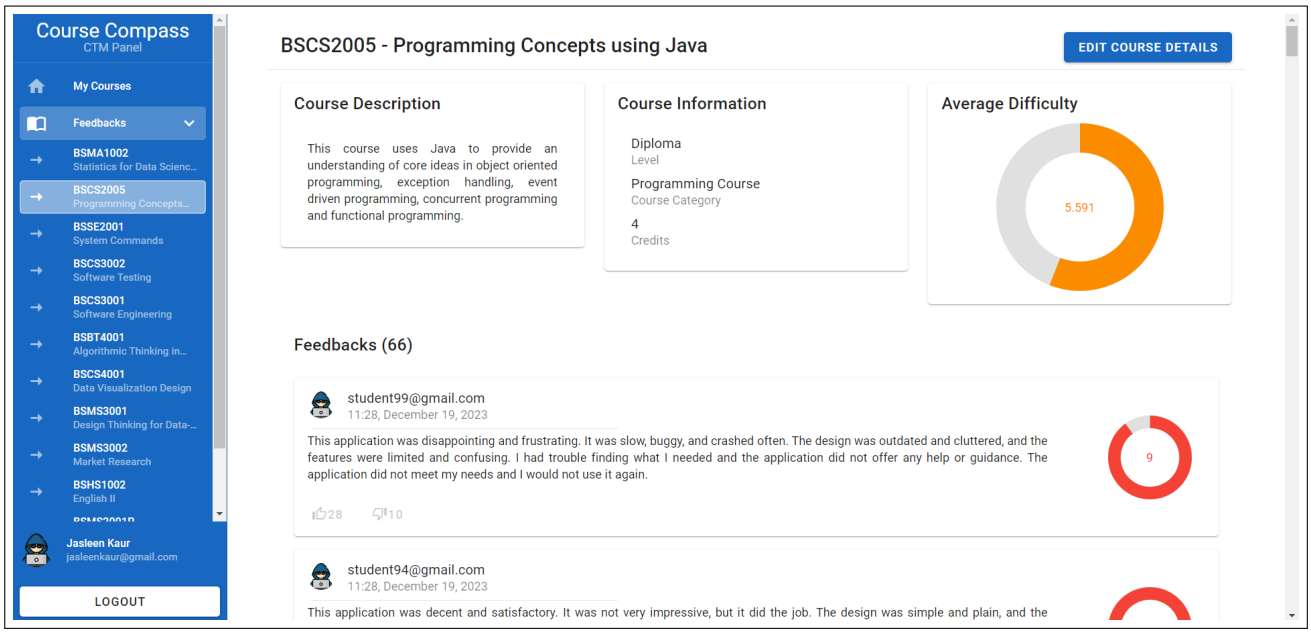


Figure 43: UI Screenshot - Course Team Member View Student Feedbacks Page

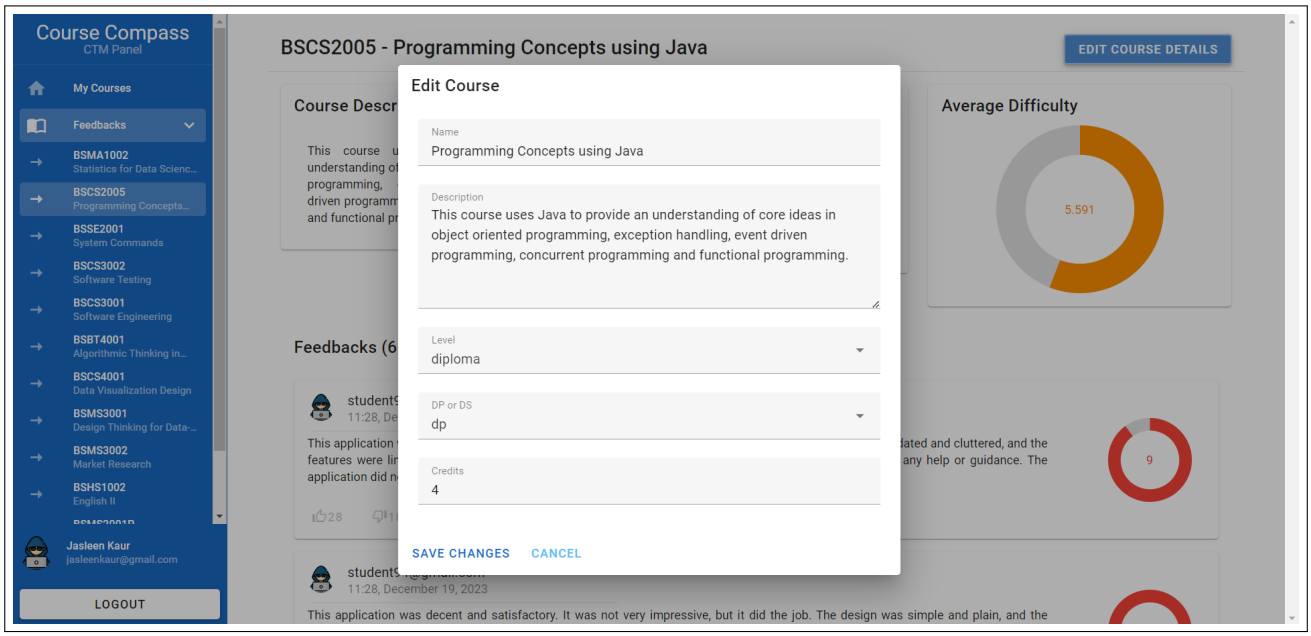


Figure 44: UI Screenshot - Course Team Member Edit Course Details Page

7.5 IITM Management Member

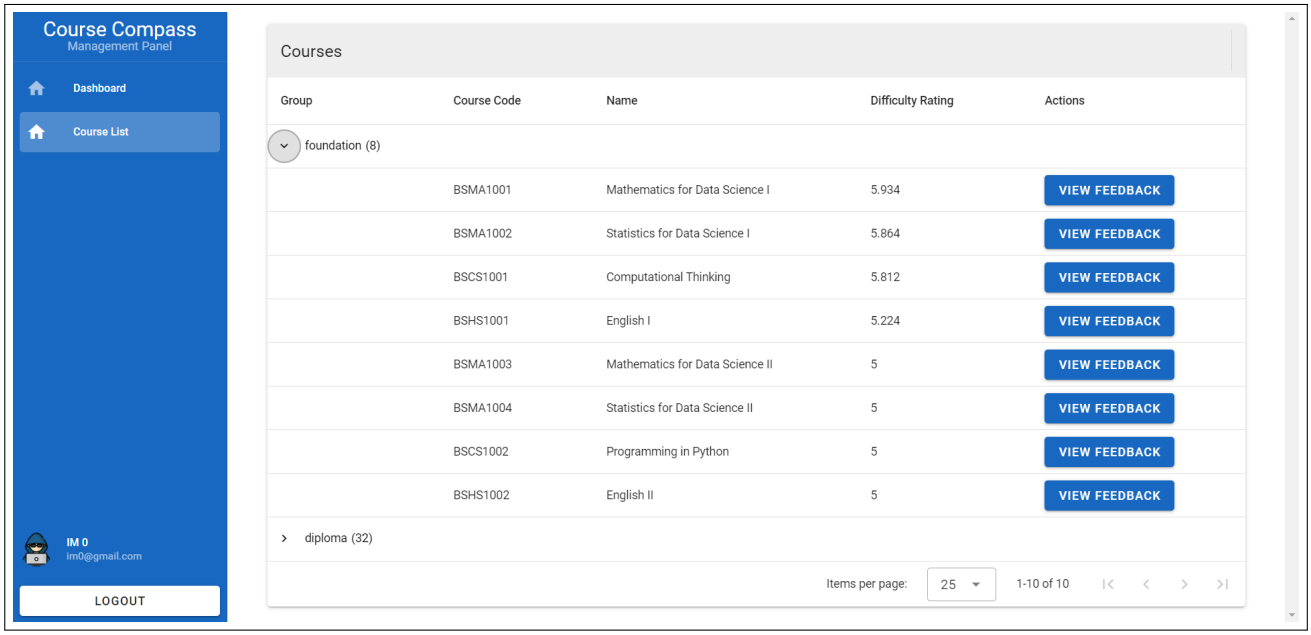


Figure 45: UI Screenshot - IITM Managemnt Member Page

7.6 Github Repository and Google Drive

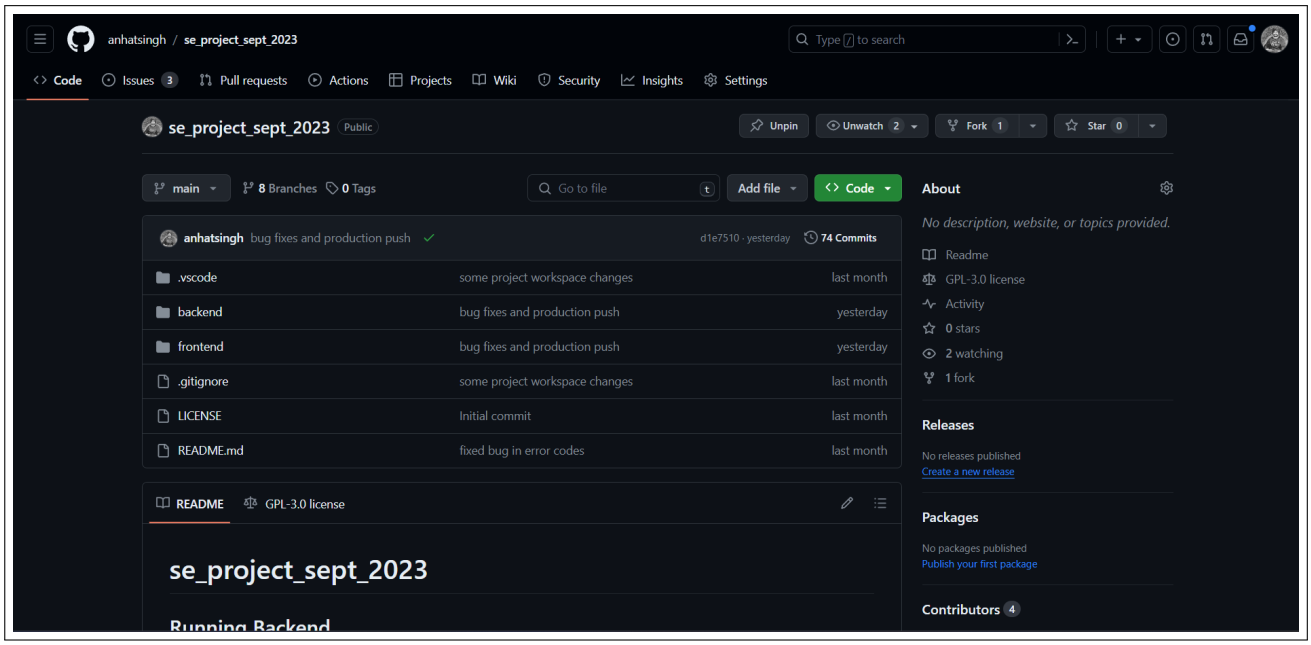


Figure 46: Github Repo and the commits done by team members; 74 commits were made during the development phase

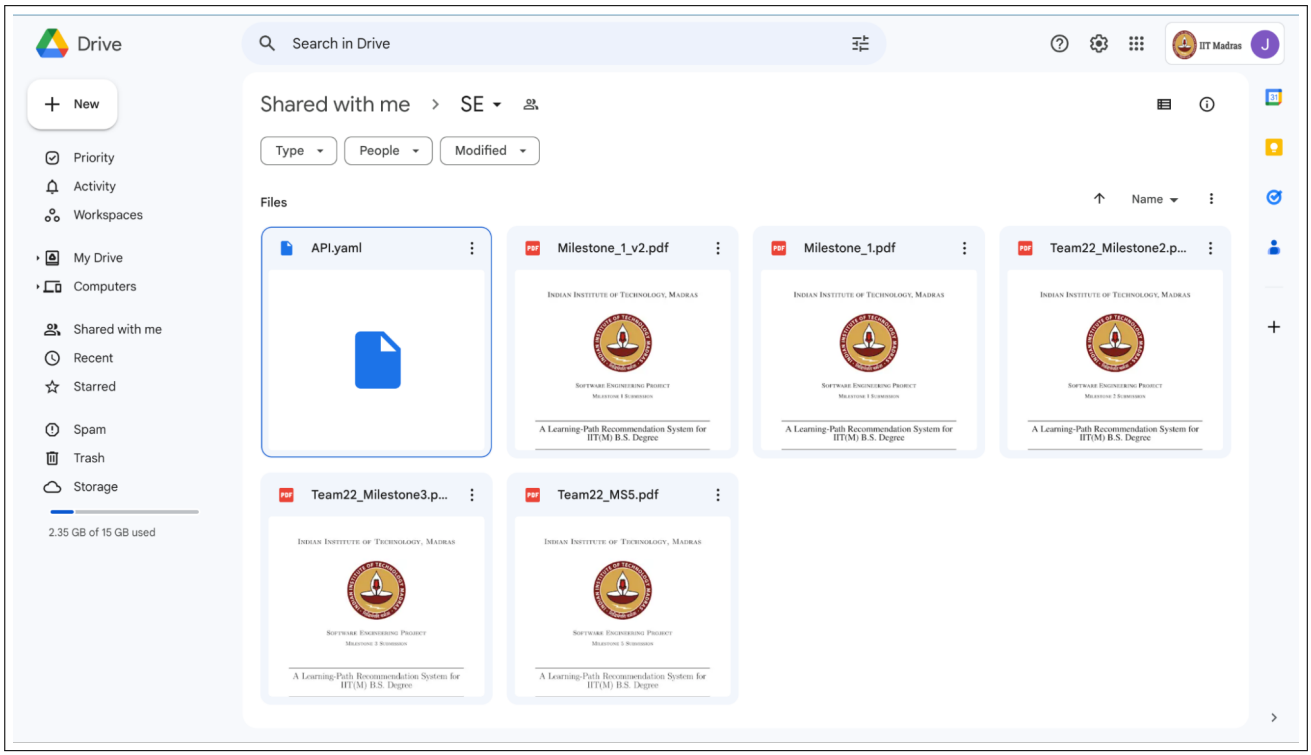


Figure 47: Google Drive folder with shared access

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.

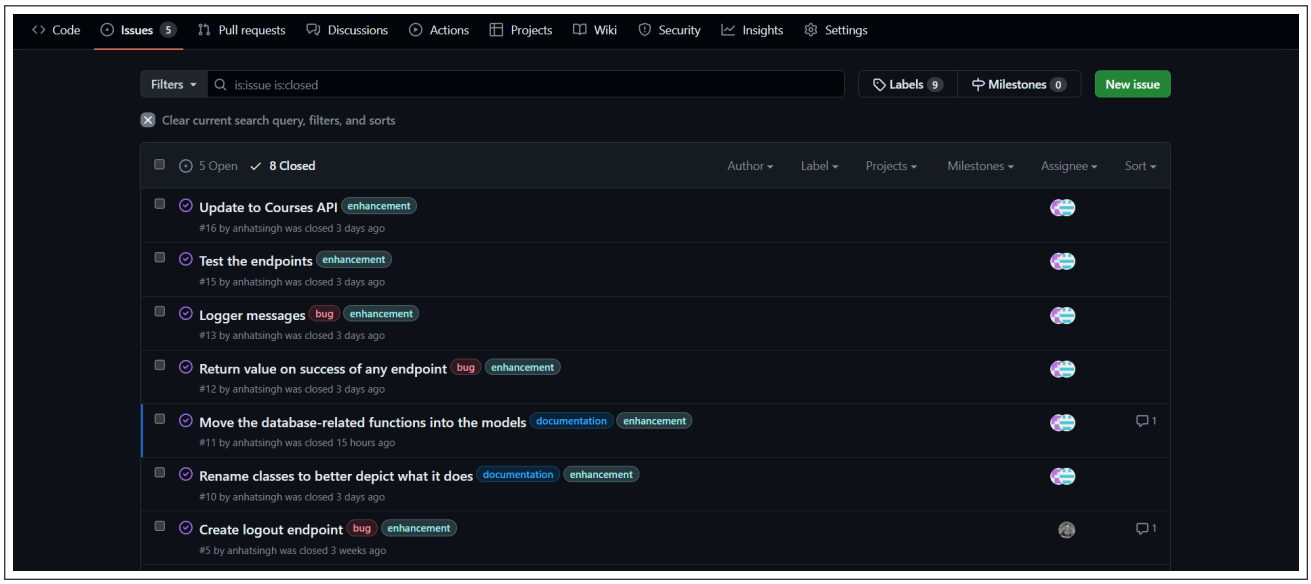


Figure 48: Issues Tracking in Github

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