2023

SOFTWARE ENGINEERING PROJECT

MILESTONE 6

FINAL SUBMISSION

PRESENTED BY TEAM : SE-SEPT-04

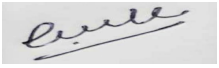
NIDHISH KUMAR (21F1003758), SHRI KRISHNA PANDEY (21F1006966), SHREYOSI SARKAR (21F1003968), NIWESH BARAJ (21F1001463), PRATHAM BHALLA (21F1003052)

Index

Honor code submission

I Shri Krishna Pandey with roll no. 21f1006966 declare that I will not use any ideas, writings, code or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody directly or indirectly or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: 

Date: 13-10-2023

I Shreyosi Sarkar with roll no.21f1003968 declare that I will not use any ideas, writings, code or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody directly or indirectly or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: 

Date: 13-10-2023

I Niwesh Baraj with roll no. 21f1001463 declare that I will not use any ideas, writings, code or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody directly or indirectly or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: 

Date: 13-10-2023

I Pratham Bhalla with roll no. 21f1003052 declare that I will not use any ideas, writings, code or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody directly or indirectly or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: 

Date: 13-10-2023

I Nidhish Kumar with roll no. 21f1003758 declare that I will not use any ideas, writings, code or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody directly or indirectly or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: 

Date: 13-10-2023

Introduction

Problem Statement

IITM BS Degree Learning Path Recommendation System:

A learning path recommendation based on both learning profile and feedback from previous term students can be a valuable tool for students. By taking into account student data from past enrollments, student performance and interests, as well as the feedback of other students who have taken similar courses, a learning path recommendation can help students identify the courses that are most likely to be beneficial to them. Such learning path recommendations can help students stay on track and make progress towards their educational goals. By providing students with a clear roadmap of the courses that they need to take, a learning path recommendation can help students pace themselves and avoid getting lost or sidetracked.

Here are some of the factors that should be considered when making a learning path recommendation:

- Enrollment data from previous terms
- The student's learning profile, including their past performance, interests, and goals.
- The feedback of other students who have taken similar courses.
- The student's schedule and other commitments.

The system should ideally have two users - an admin, and a student. An admin can load enrollment data from previous terms. The system should be able to infer patterns and provide recommendations to students. Students can also provide their feedback about past courses. The student can provide their learning profile, interests, goals, schedules and commitments, and the system should provide appropriate recommendations based on all the above inputs.
You are free to think creatively and come up with additional requirements or modify some of these requirements, as long as the overarching purpose of a learning path recommendation system is fulfilled.

Course Faculty:

Dr. Sridhar Iyer
Department of Educational Technology, IIT Bombay

Dr. Prajish Prasad
Computer Science, FLAME University

Course Support Team:

Omkar Joshi
PhD research scholar, IIT Bombay

Piyush Wairale
MTech

Shivani Varma
BTech

About Coursify App:

Coursify is a user-friendly and innovative education platform designed to empower students in their learning journey. With the integration of the Learning Path Recommendation System, Coursify goes beyond traditional course offerings. It guides students towards success by understanding their unique profiles and incorporating valuable feedback from the community. Coursify is not just an app; it's a personalized learning companion dedicated to maximizing each student's potential.

Summary of Project Milestones

The **Software Engineering Project** by **Team SE-SEPT-04** involves the development of a course recommendation system for students at an educational institution. The team comprises **Nidhish Kumar, Shri Krishna Pandey, Shreyosi Sarkar, Niwesh Baraj, and Pratham Bhalla**. The project adheres to an honor code to ensure originality and prohibits sharing work outside the group to prevent plagiarism and have done some wholesome efforts to create the app as we call **Coursify**.

Milestone 1: Identify User Requirements

The team identified primary, secondary, and tertiary users, including students, admins (POD Team), academic advisors, and alumni/seniors using **SMART** guidelines. The primary user stories were defined, focusing on students' needs for viewing recommended courses, providing academic information for personalized recommendations, accessing course details, receiving notifications, and submitting feedback.

Milestone 2: Low-Fidelity Prototype and Storyboard

During this milestone, the team created low-fidelity **prototypes** and **storyboards** to visualize the project's design. The storyboard presented a scenario where students faced confusion about course registration, sought advice from a TA, and discovered the recommendation system. The components and interactions among users, admins, advisors, and alumni were outlined.

Prototype walkthrough video link:
https://drive.google.com/file/d/1hTanlKUZCvo2_AbMIoesgtovVsSuX1Vz/view?usp=sharing

Milestone 3: Project Schedule, Component Design, Class Diagram, and DB Schema

In this milestone, the team developed a detailed project schedule using **JIRA**, assigned tasks for various **sprints**, and shared progress through a **Gantt chart** and **scrum board. Component designs** were detailed for **students, admins, and alumni/advisors**. A **class diagram** illustrated the system's structure, and a **database schema** outlined the data relationships.

Milestone 4: API Endpoints and Yaml file

Milestone 4 focuses on establishing critical API endpoints to facilitate seamless communication between the front-end and back-end of the recommendation system. The **YAML** file provides a detailed overview of endpoints addressing user authentication, data handling and profile management. The Yaml file has some dummy requests for performing all **CRUD** operations. Using **POST, GET, PUT, DELETE** methods. Successful completion sets the stage for subsequent tasks, encompassing back-end development, rigorous testing, and comprehensive system integration.

Milestone 5: API Testing and Documentation

The team conducted **thorough testing** of **API** endpoints, including user registration, login, password recovery, profile retrieval, course-related queries, and more. The **testing** involved expected inputs and outputs, ensuring the system's functionality. The **documentation** included detailed **API** endpoint descriptions, inputs, expected outputs, and actual results.

In this report -

Milestone 6: Final Submission (2023): The team submitted this final project, adhering to the honor code.

Overall Project Progress:

The project has progressed systematically, starting with user identification, prototype development, and component design. The team has followed an organized sprint-based **(Agile)** approach, ensuring timely completion of tasks. Milestone 4 API endpoints were created and then Milestone 5 demonstrates the rigorous testing of API endpoints, ensuring the reliability and effectiveness of the recommendation system. This report includes API implementation, documentation, testing that is a detailed report on work done from Milestone 1 through Milestone 5. A section describing code review, issue reporting and tracking using screenshots

This summary encapsulates the team's commitment to creating a comprehensive course recommendation system, integrating user needs, design elements, and robust API functionality.

Project presentation ppt link:

https://docs.google.com/presentation/d/1YHND0-_Cn1xOK1EDhUd8S6pO0D4R4ChD1giGeUduDKA/edit#slide=id.g2a651255b20_5_5

Project presentation video link:

https://drive.google.com/file/d/1mV59xSHgzVFdzA2m7pOWMx4hYz0H2GCt/view?usp=sharing

Identify User Requirements

Identified User

Primary Users:

Students: Student enrolled in IITM BS degree program or Direct Diploma enrolled students who want to seek guidance from recommendation system and fellow seniors.

Admins (POD Team): Admins would be Course Instructors, Teaching assistant, Support team members who have the access to the data and have to maintain the apps performance and Answer query of the students related to recommendation system.

Secondary Users:

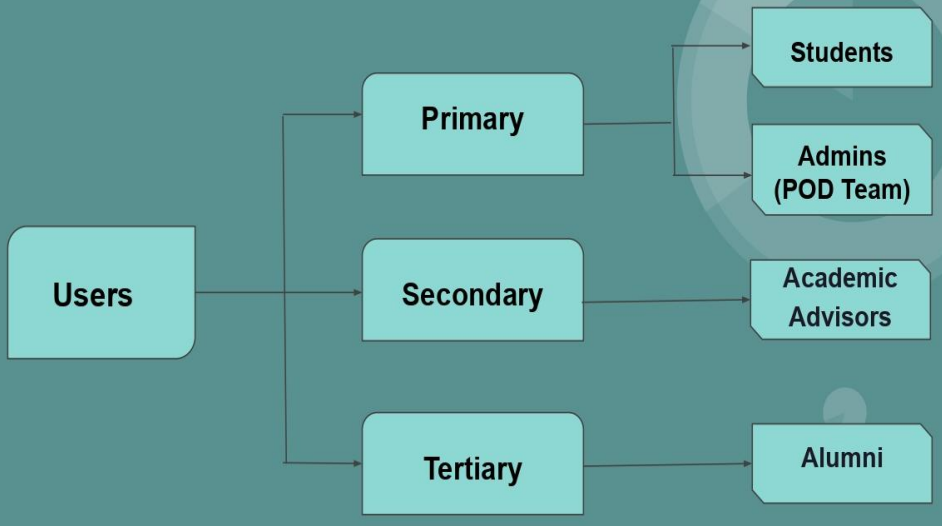
Academic Advisors: Academic advisors can be seniors, mentors and members of senate If want to give insight to the students.

Tertiary Users:

Alumni/Seniors (Optional): Alumni who have passed the BS level or have left the program can also help other students by sharing their personal experiences of the program.

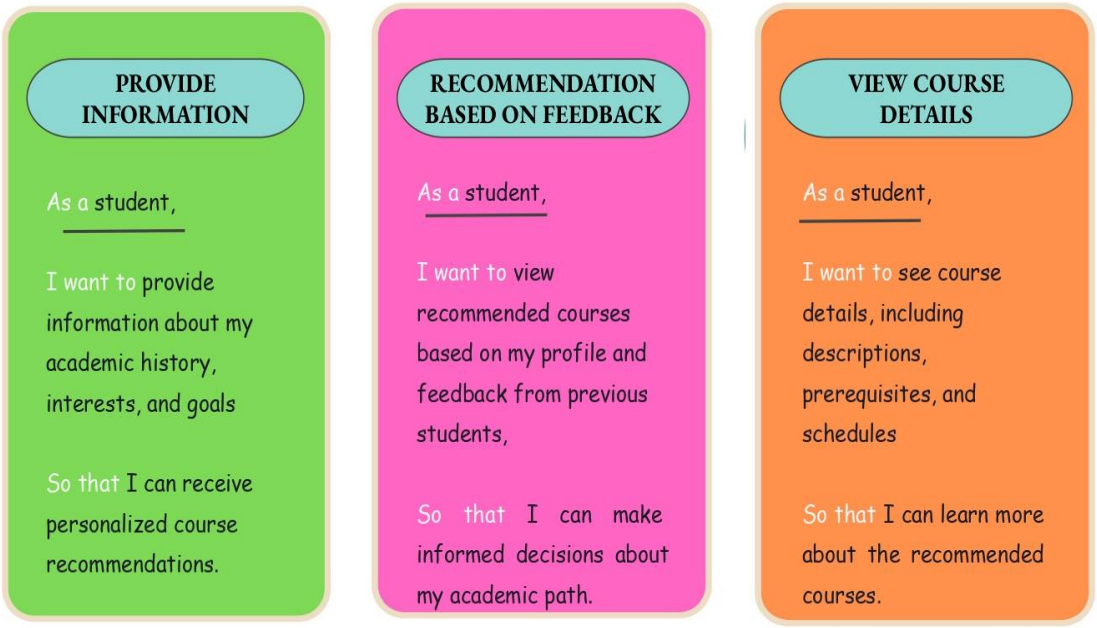
Page 1

User Identification



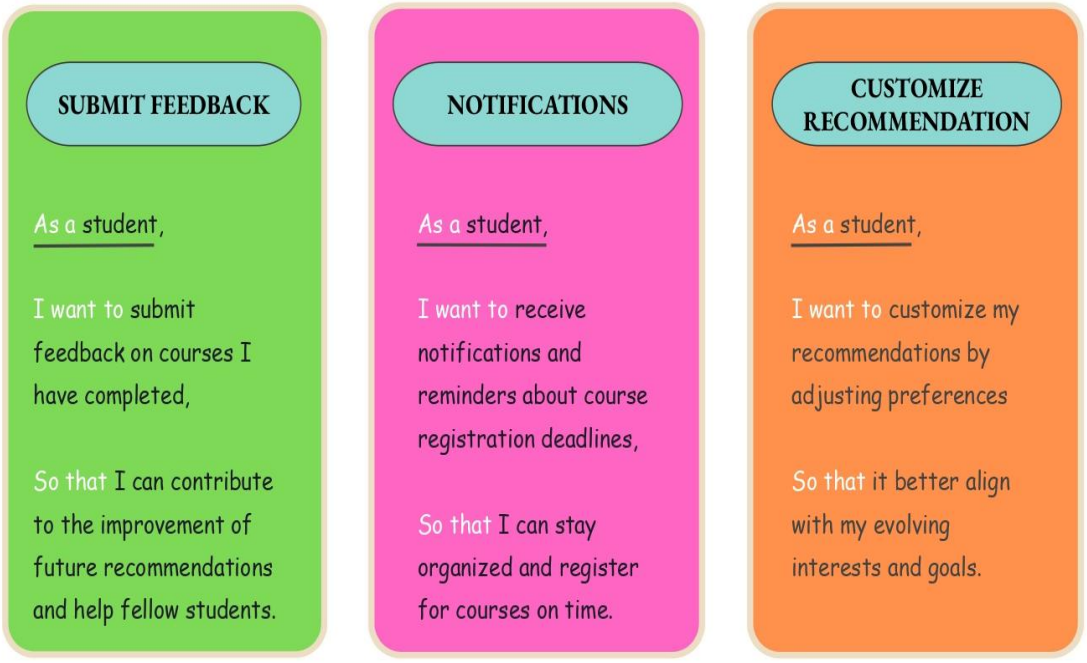
Page 2

Primary User - Student



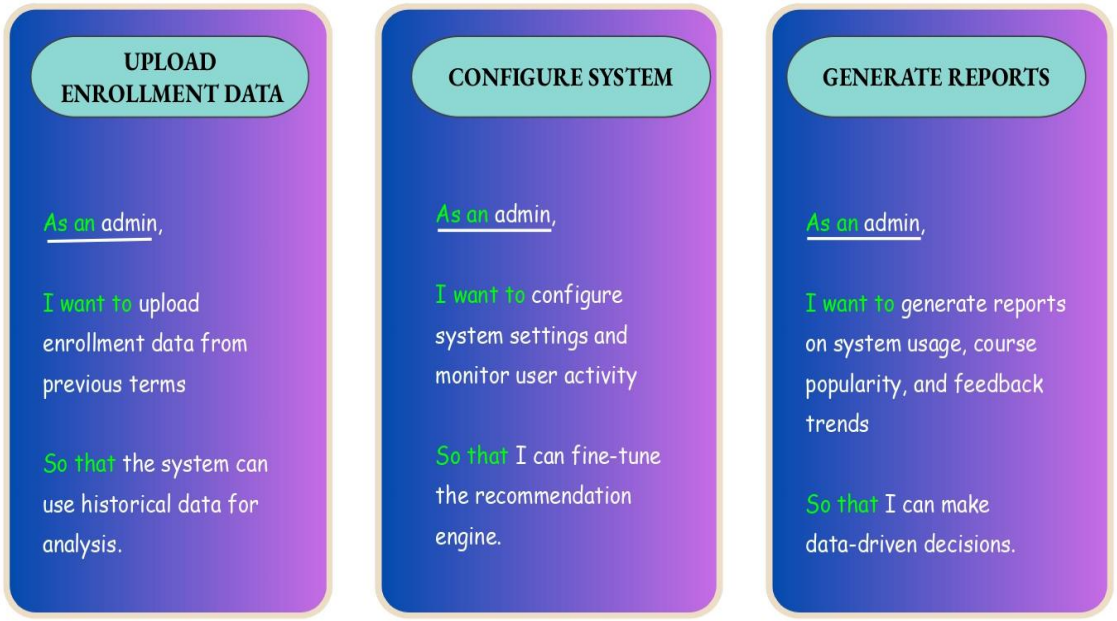
Page 3

Primary User - Student



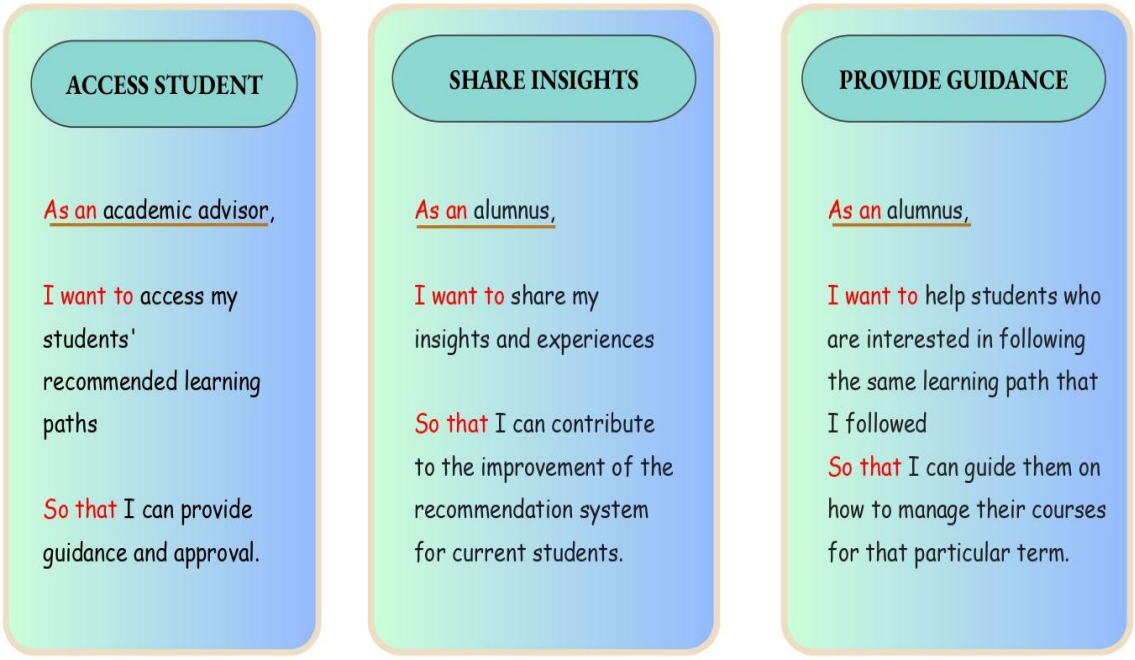
Page 4

Primary User - Admin/POD Team



Page 5

Secondary/Tertiary User- Alumni/Advisor



Page 6

Requirement Gathering Techniques

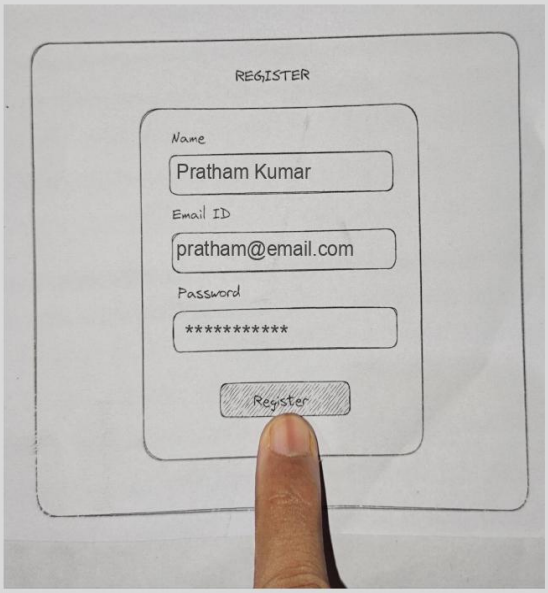
- We have first discussed among ourselves about what as a students we would require in a course recommendation system.(Focus Group)
- We attended live session and gone through the problem statement.(Naturalistic Observation)
- We conducted Informal interviews of foundational level students and Diploma level students to understand what they would like in a course recommendation system app. (Interview and Questionnaire)
- We have then tried our best to come up with Few User stories which we can work on considering **S**pecific, **M**easurable, **A**chievable, **R**elevant and **T**ime boxed.

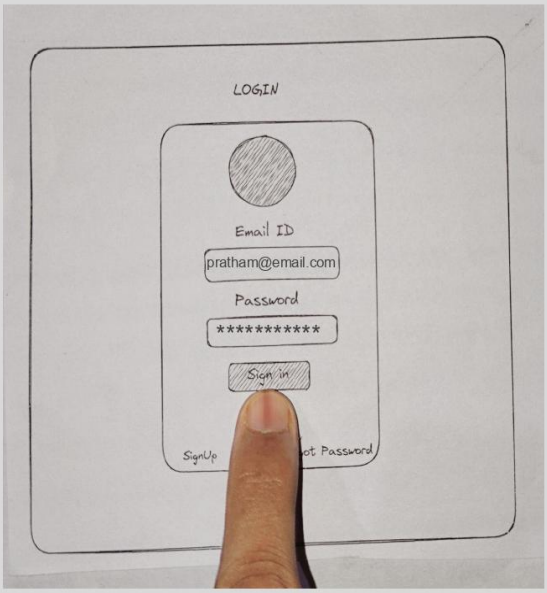
Page 7

User Interfaces Design

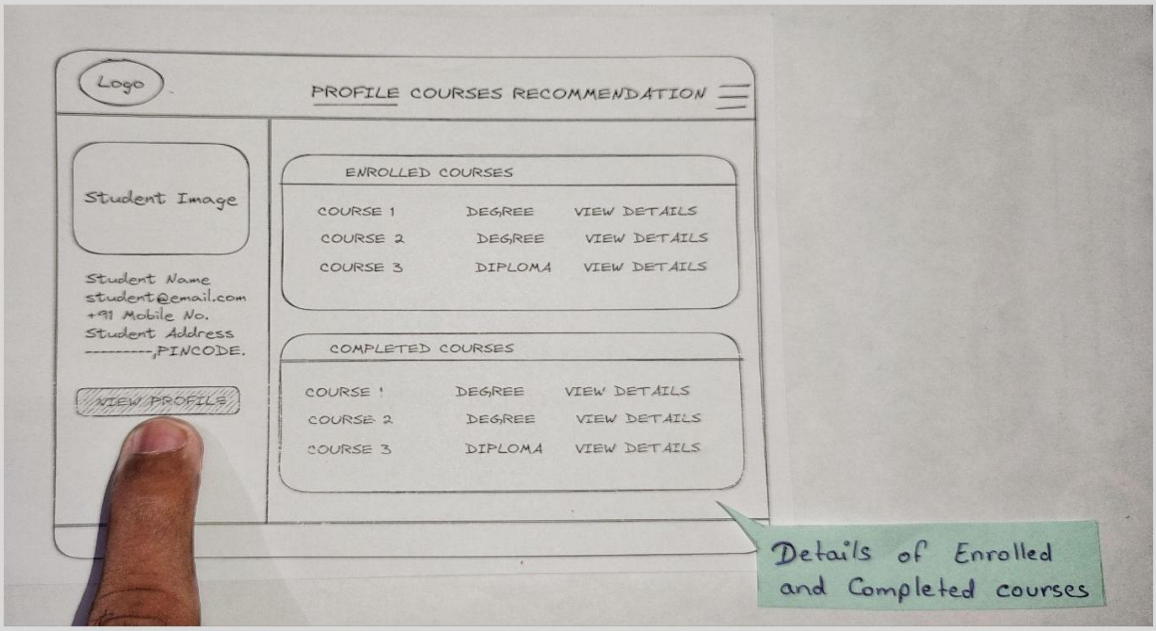
Wireframe & Paper Prototype

Common View - Registration and Login Page

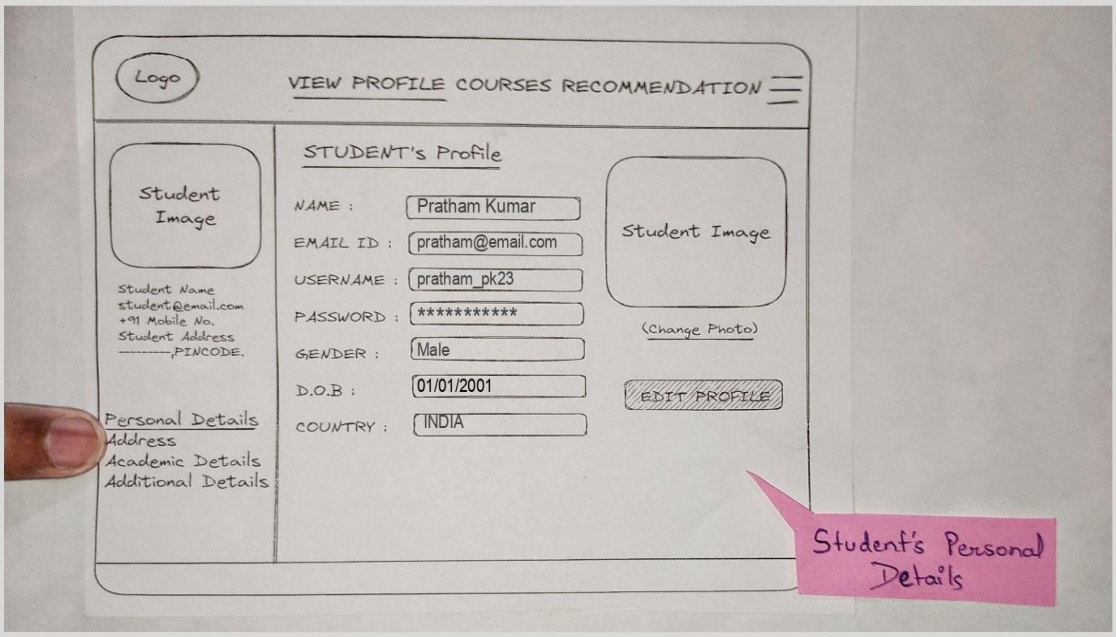




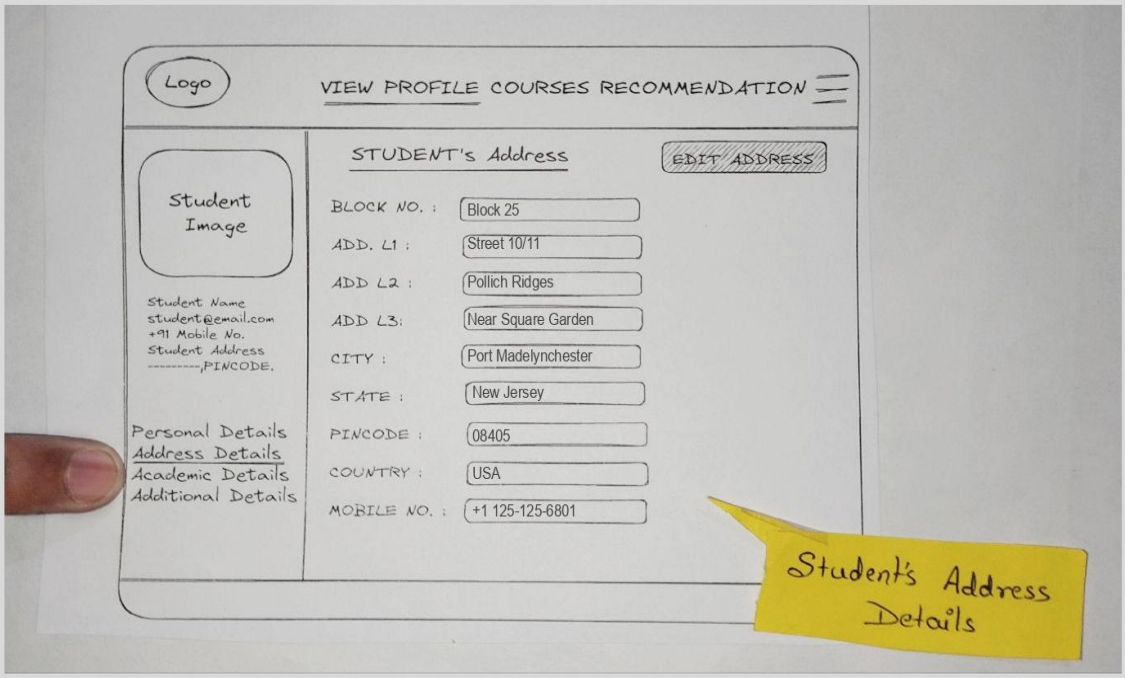
Student's View - Dashboard



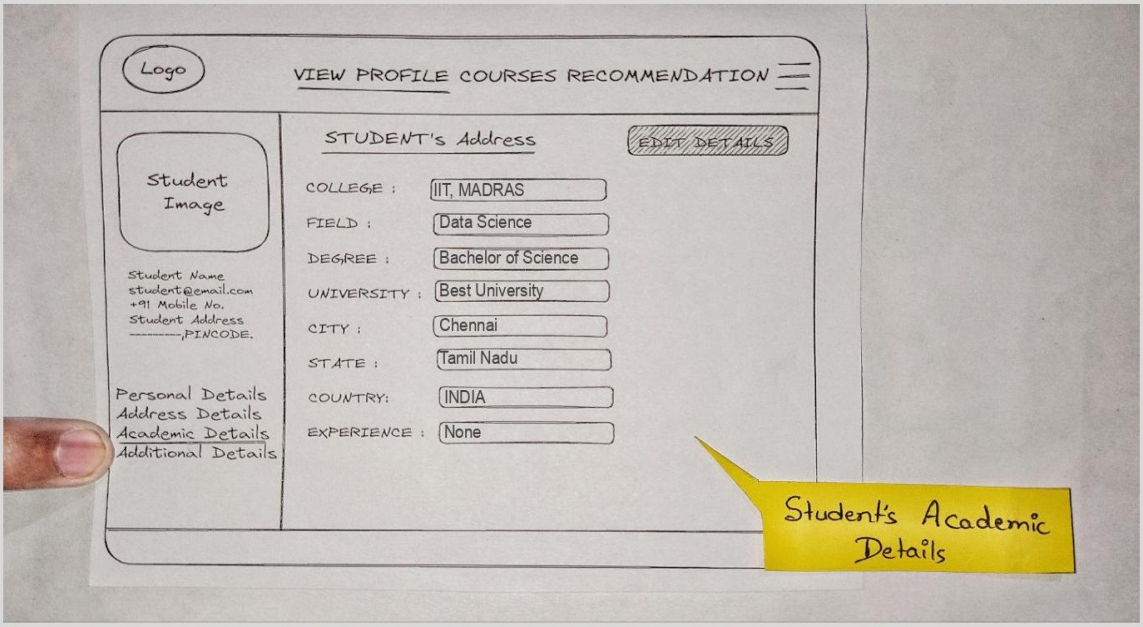
Student's View - Personal Details Page



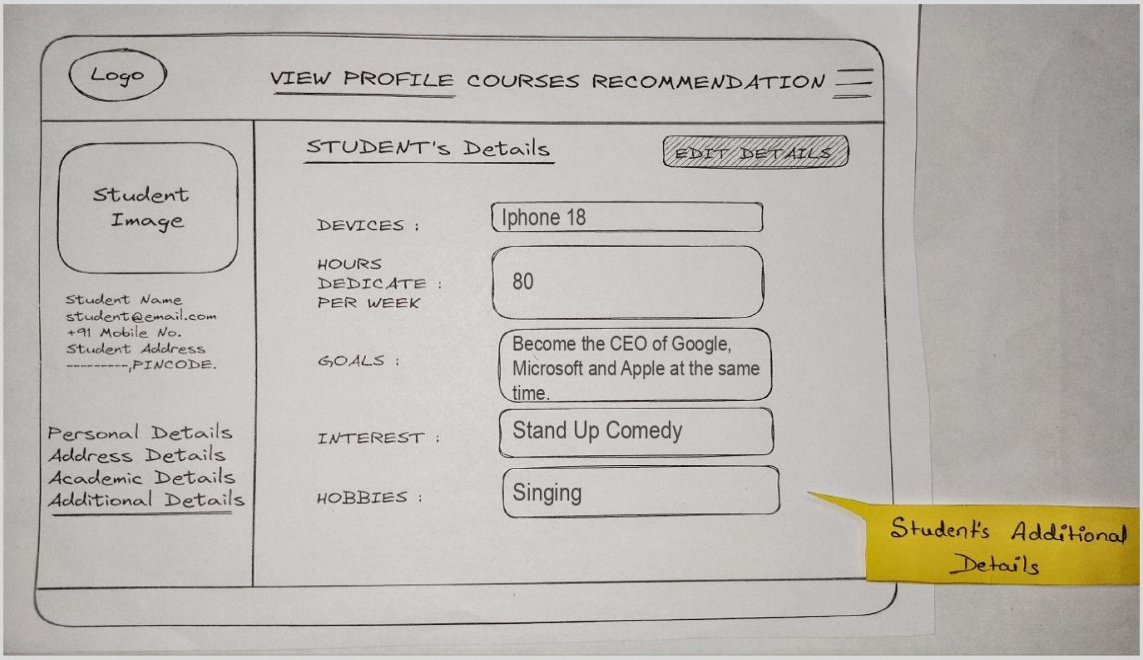
Student's View - Address Details Page



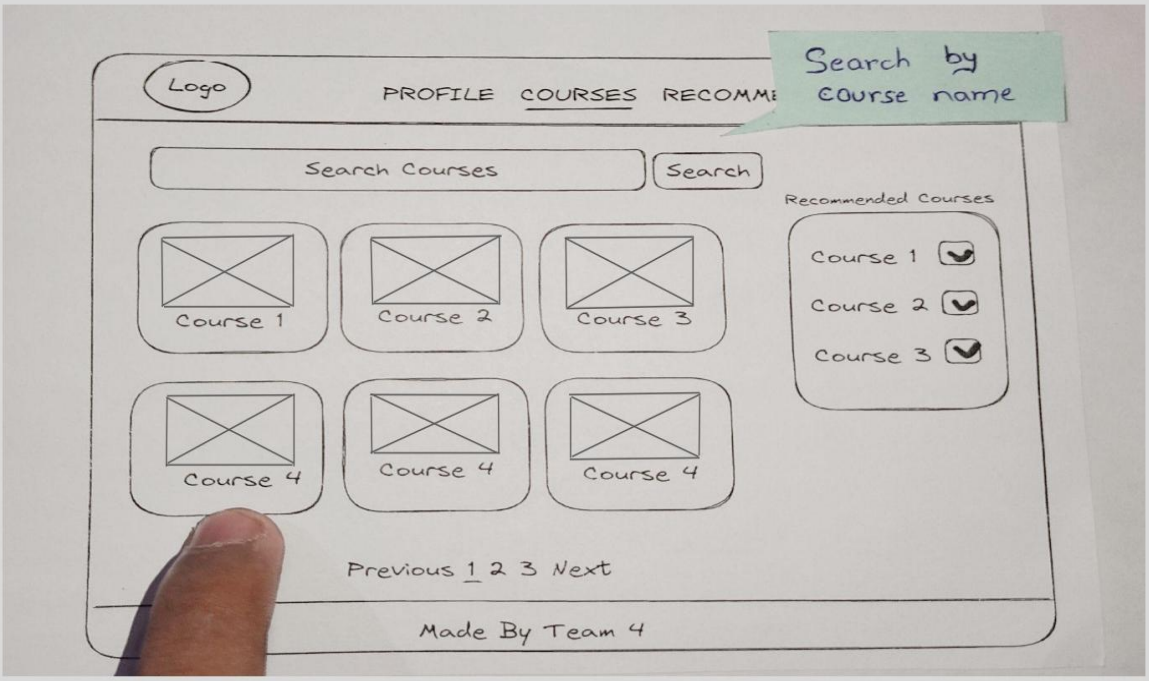
Student's View - Academic Details Page



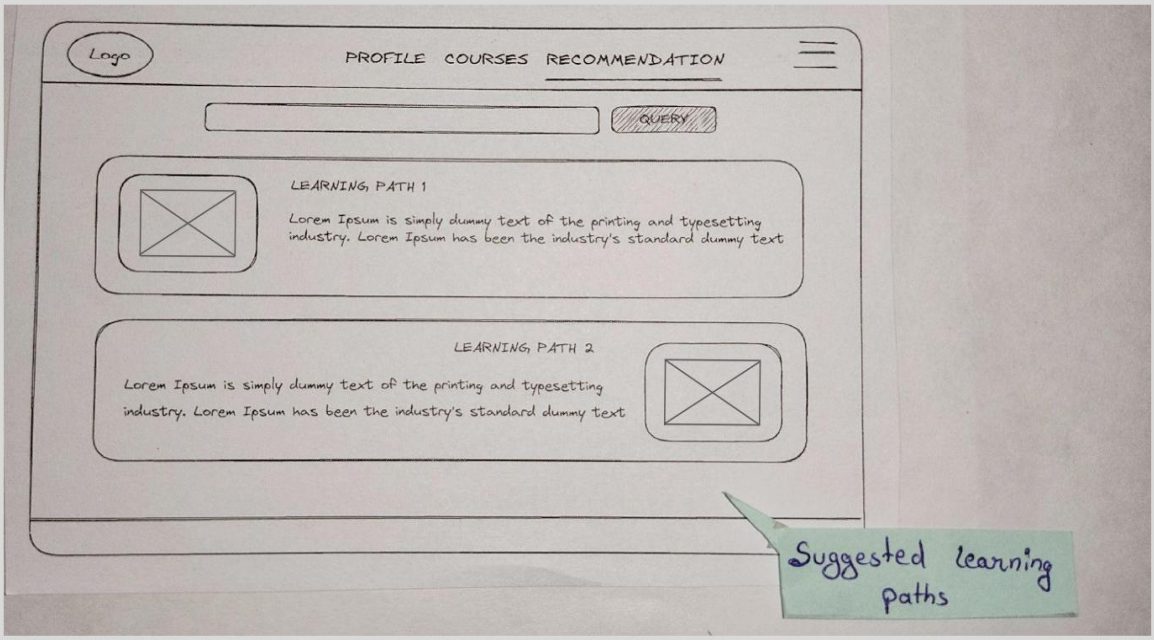
Student's View - Additional Details Page



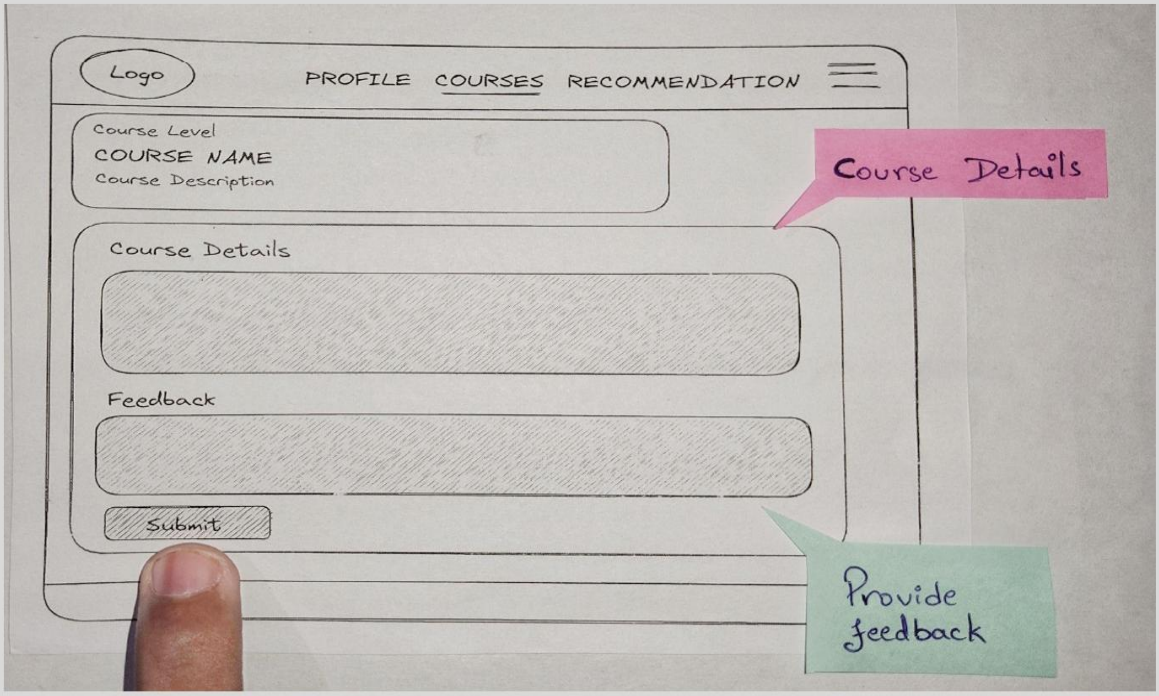
Student's View - Courses Page



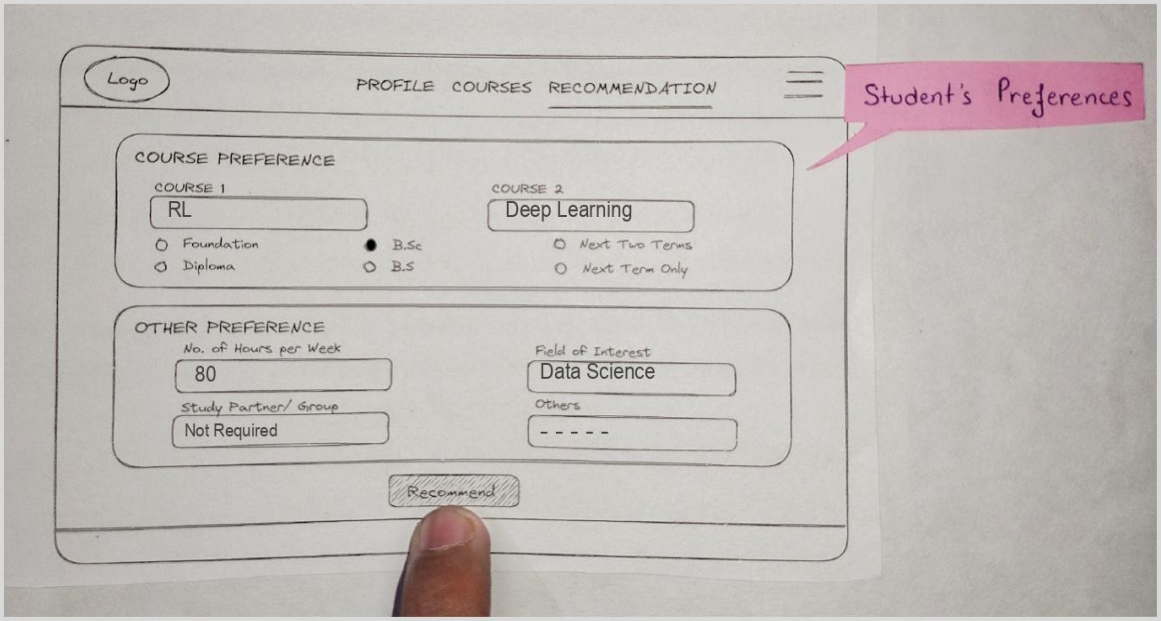
Student's View - Suggested Paths Page



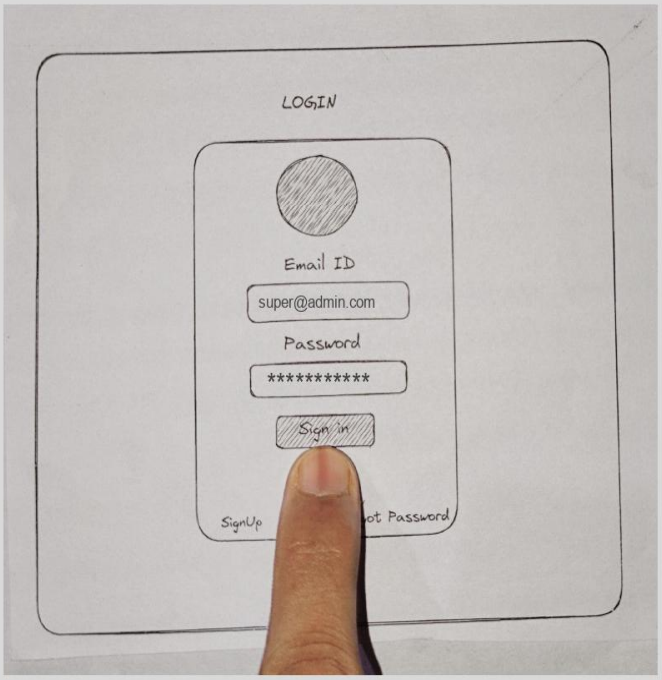
Student's Page - Course Details Page



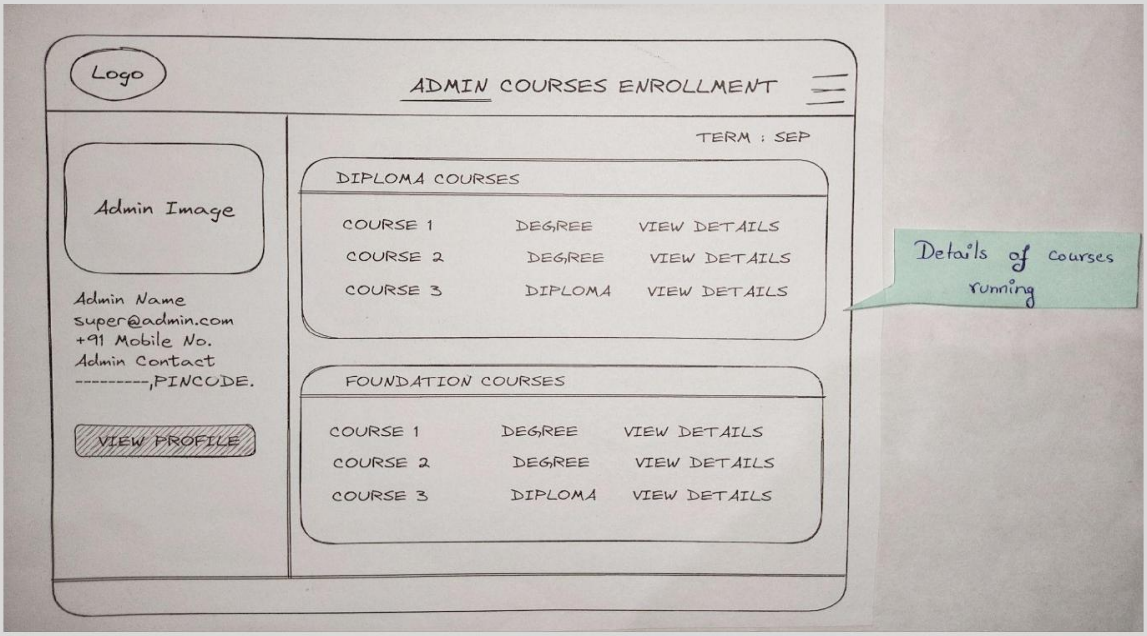
Student's View - Student's Preferences Page



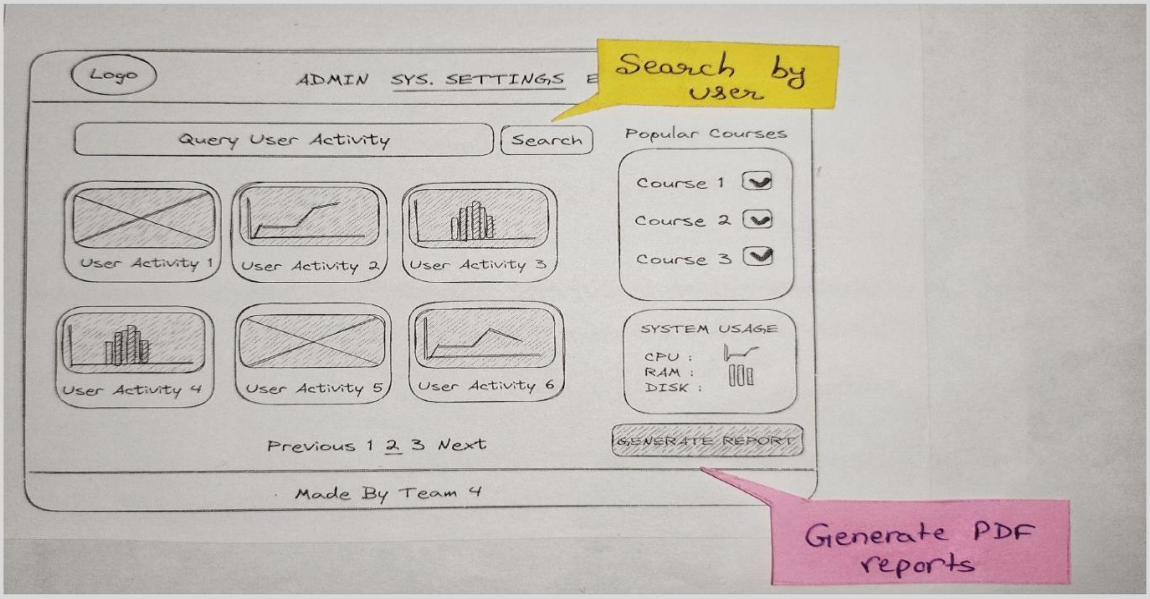
Admin's View - Login



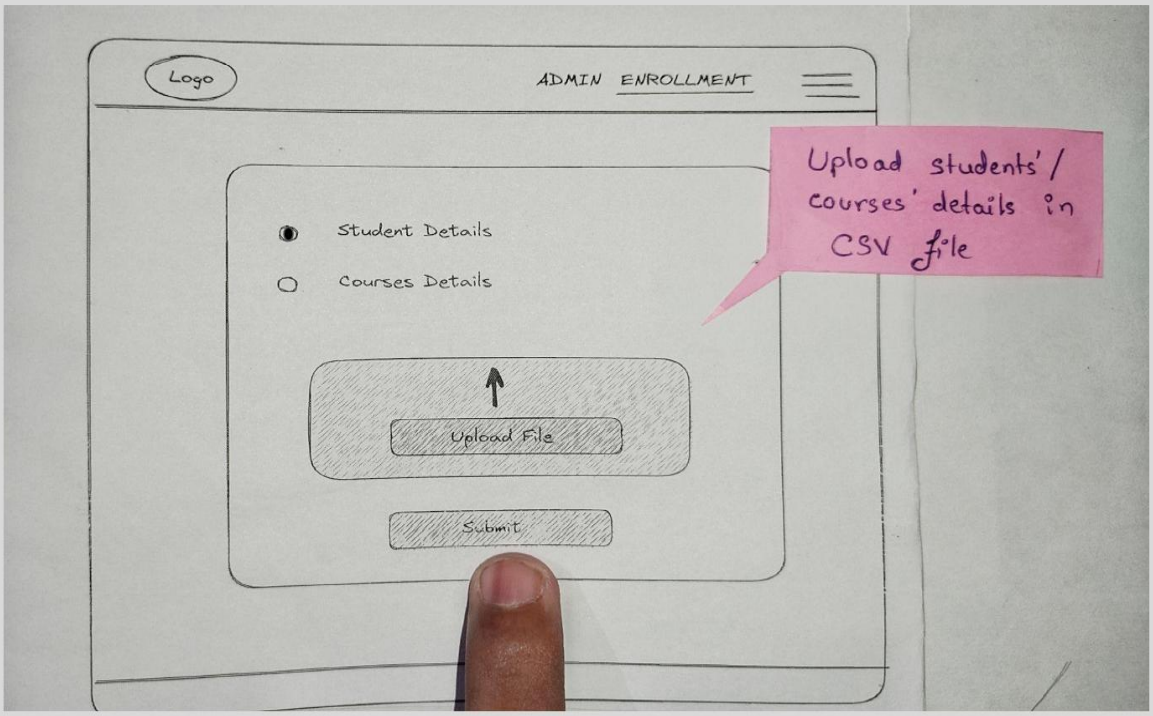
Admin's View - Dashboard



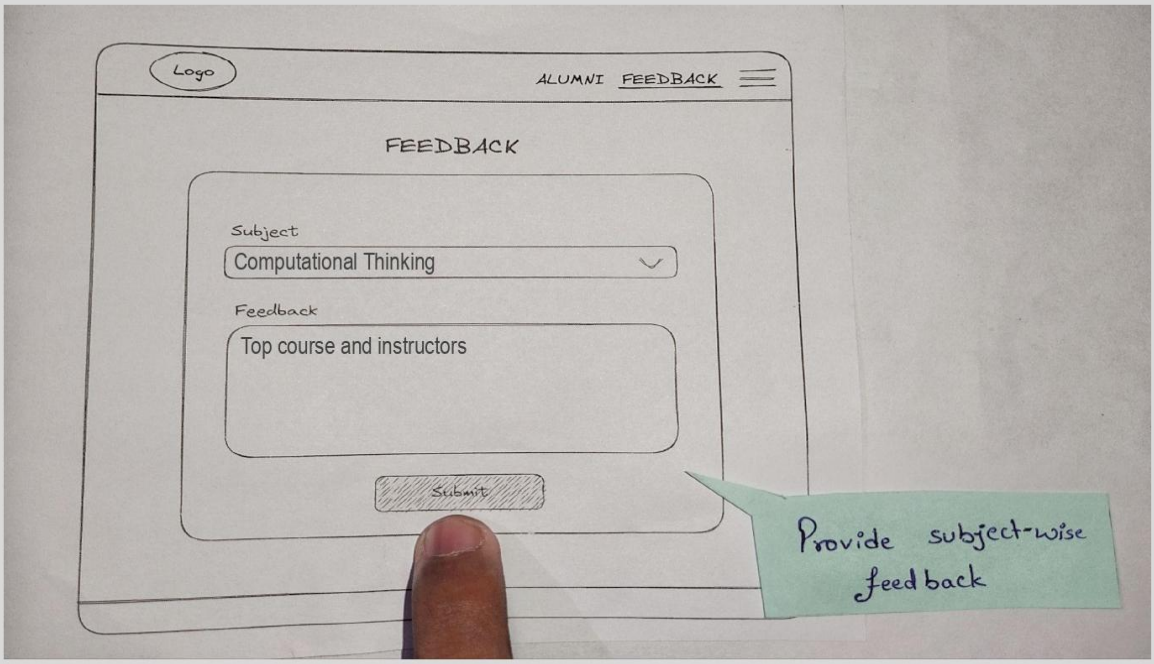
Admin's View - Statistics Page



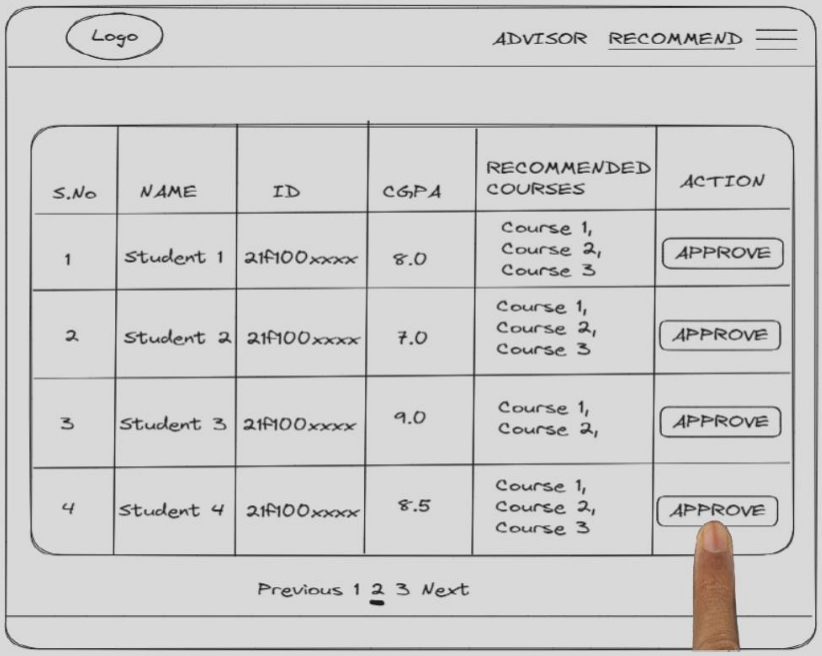
Admin's View - Update Students'/Courses' Details Page



Alumni's View - Feedback Page



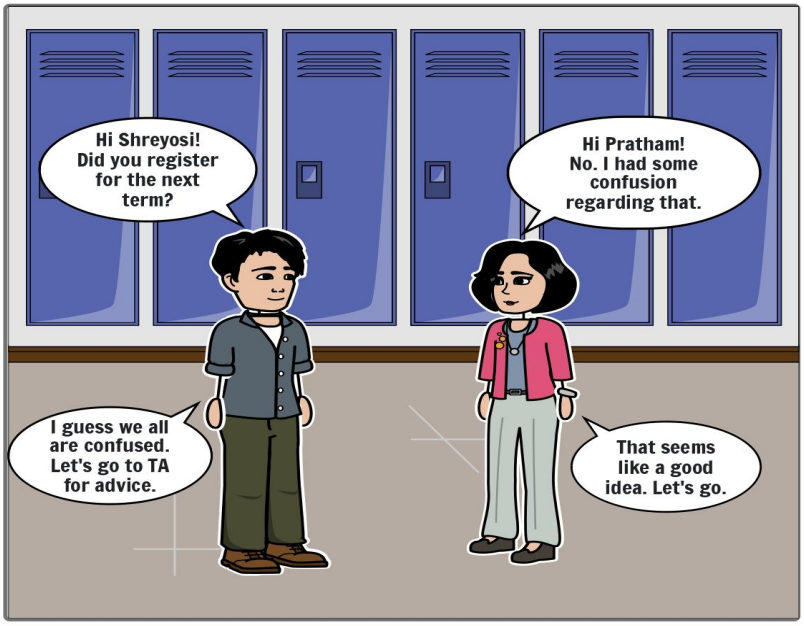
Academic Advisor's View - Recommend Page



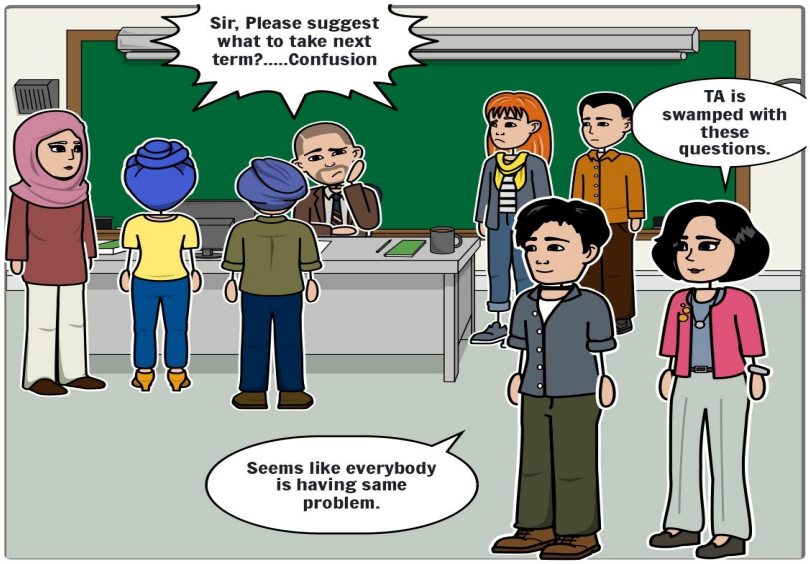
Academic Advisor can Approve or Reject Student's chosen learning path

Storyboard

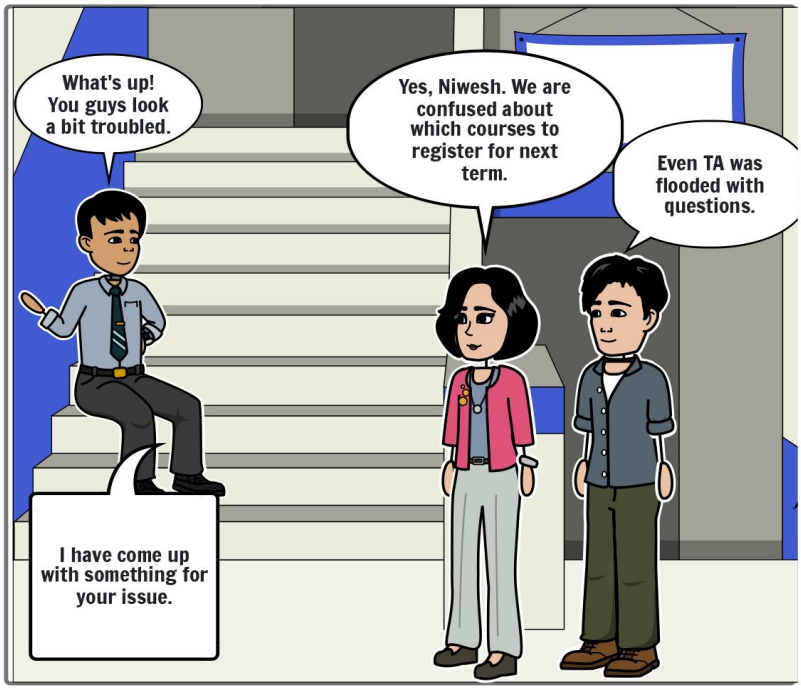
Story Begins



This story starts with two students discussing with each other for confusion in course selection.



In TA's Office



Outside TA's office

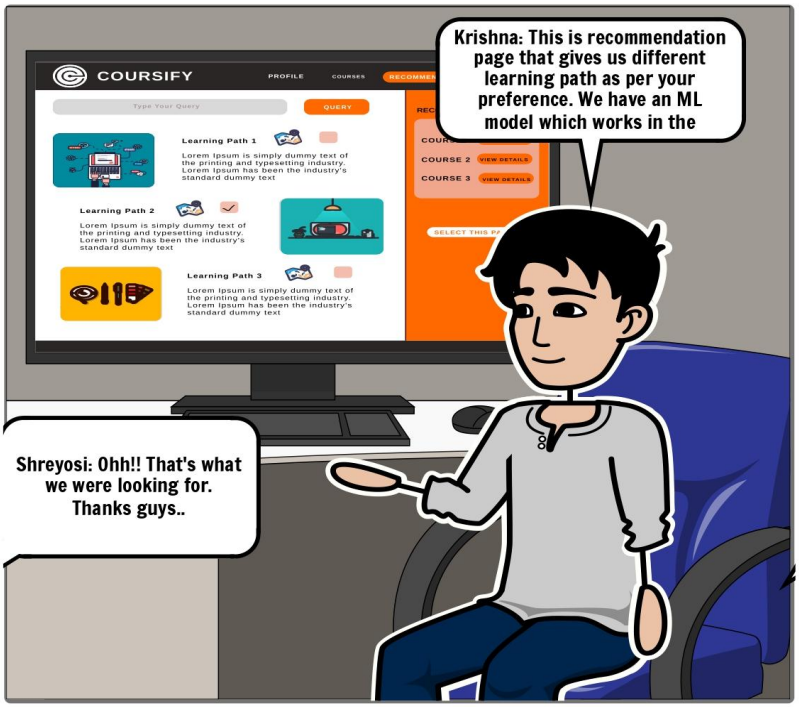
Story 1

From the perspective of a student

At Krishna's home

At Krishna's home

At Krishna's home



At Krishna's home

At Krishna's home

Story 2

From the perspective of an admin

Nidhish explaining admin's perspective to TA in TA's office

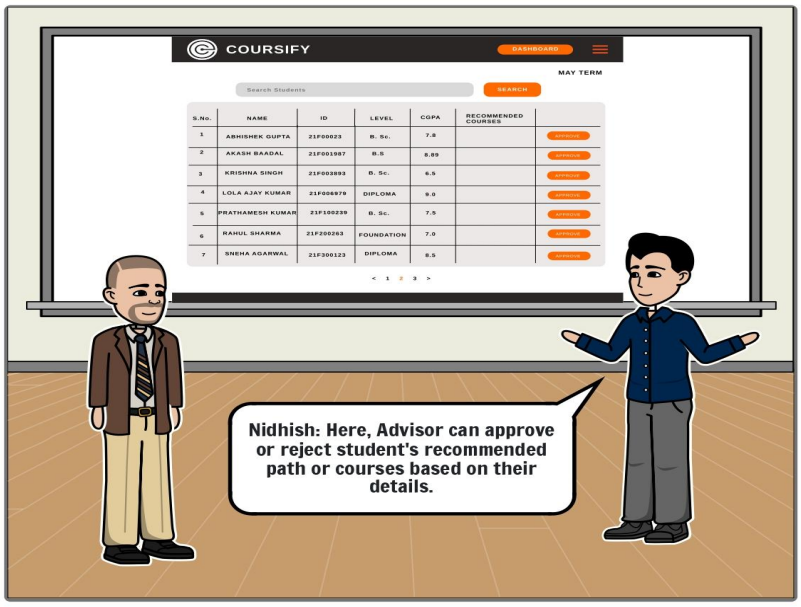
Nidhish explaining admin's perspective to TA in TA's office

Nidhish explaining admin's perspective to TA in TA's office

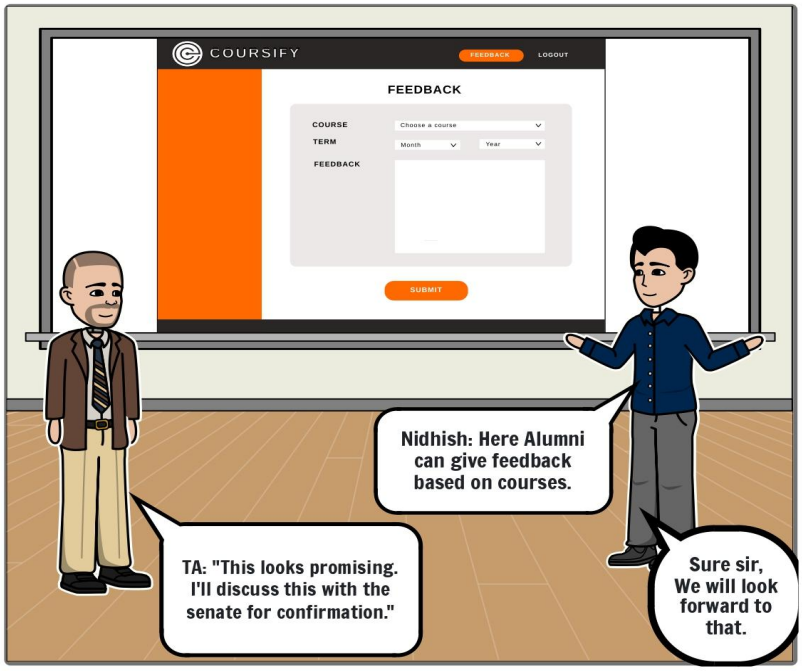
Nidhish explaining admin's perspective to TA in TA's office

Story 3

From the perspective of an advisor/alumni



Nidhish explaining advisor's perspective to TA in TA's office



Nidhish explaining alumni's perspective to TA in TA's office

TA explains about the app in senate office

TA congratulates about the approval of the app outside senate office

Project Scheduling and Component Design

1. PROJECT SCHEDULE

Majorly, JIRA has been used for scheduling, making sprint and Gantt chart timeline.

1.1 TASK DISTRIBUTION

As of November 2, 2023, we have successfully completed several sprints in our project, with each sprint contributing to the development of our recommendation system. These sprints represent distinct phases of our project, focusing on different aspects of its development. Below, we provide a summary of the key activities and accomplishments within each sprint.

- Sprint 1 (27/09/2023 - 12/10/2023) - Milestone 1 Preparation (Completed)
  In this sprint, the team prepared for Milestone 1 by identifying users, understanding the problem statement, and assigning roles.
  The key tasks included:
  - "Identify users" assigned to Pratham.
  - "User stories" assigned to Nidhish.
  - "Milestone 1 Report" assigned to Shri Krishna.

- Sprint 2 (13/10/2023 - 19/10/2023) - Low-Fidelity Prototype and Storyboard (Completed)
  Sprint 2 focused on creating low-fidelity prototypes and storyboards for the project, allowing us to visualize the project's design.
  - "Low fidelity prototype" assigned to Niwesh
  - "Paper prototype" assigned to Nidhish
  - "Storyboard - write story" assigned to Shri Krishna

- Sprint 3 (20/10/2023 - 27/10/2023) - Digital Prototype and Milestone 2 Report (Completed) Sprint 3 involved creating a digital prototype and further developing the project's design. Additionally, we worked on the Milestone 2 Report to document our progress.
    - "Digital prototype" assigned to Niwesh
    - "Storyboard - create storyboard" assigned to Shreyosi
    - "Milestone 2 - Report" assigned to Pratham

- Sprint 4 (30/10/2023 - 03/11/2023) - Project Schedule and Component Design (Ongoing)

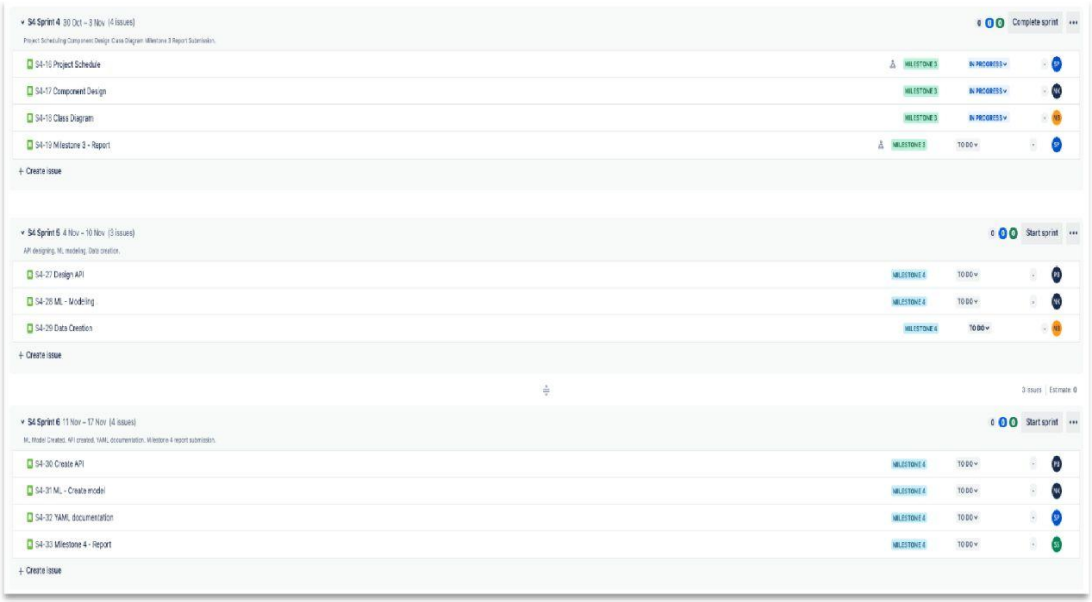
In the current sprint, we are developing the project schedule to establish a structured timeline.

We are also delving into component design and creating class diagrams. The key tasks include:

- "Project schedule" assigned to Shri Krishna
- "Component design" assigned to Nidhish
- "Class diagram" assigned to Niwesh
- "Milestone 3 Report" assigned to Shri Krishna

- Sprint 5 (04/11/2023 - 10/11/2023) - API Design and Machine Learning Modeling (Upcoming) In Sprint 5, we will focus on API design and machine learning modeling to establish the technical infrastructure of the project. The team will also work on data creation.
    - "API Design" assigned to Pratham
    - "Machine Learning Modeling" assigned to Nidhish
    - "Data Generation" assigned to Niwesh

- Sprint 6 (11/11/2023 - 17/11/2023) - API Implementation and Documentation (Upcoming)
  In Sprint 6, the team will concentrate on implementing the API, creating machine learning models, and documenting the project through YAML documentation.
  - "Create API" assigned to Pratham
  - "Create ML model" assigned to Nidhish
  - "YAML documentation" assigned to Shri Krishna
  - "Milestone 4 Report" assigned to Shreyosi



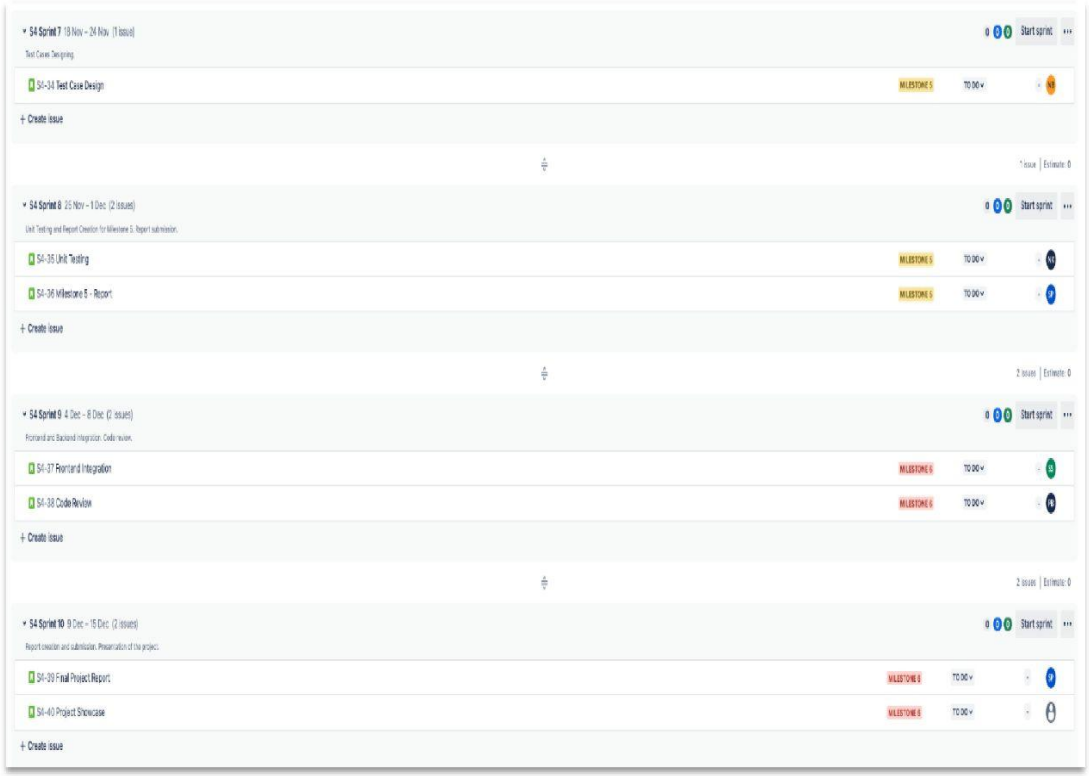
- Sprint 7 (18/11/2023 - 24/11/2023) - Test Case Design (Upcoming)
  Sprint 7 will be dedicated to designing test cases to ensure the project's functionality and reliability.
- Sprint 8 (25/11/2023 - 01/12/2023) - Unit Testing and Milestone 5 Report (Upcoming)
  In Sprint 8, we will perform unit testing to validate the components of the project. We will also work on the Milestone 5 Report.

- Sprint 9 (04/12/2023 - 08/12/2023) - Frontend Integration and Code Review (Upcoming)

  In Sprint 9, our group will focus on integrating the front-end components of the project. This will involve combining the user interface elements and ensuring a cohesive user experience. Additionally, we will conduct a code review to assess the quality and consistency of our codebase.

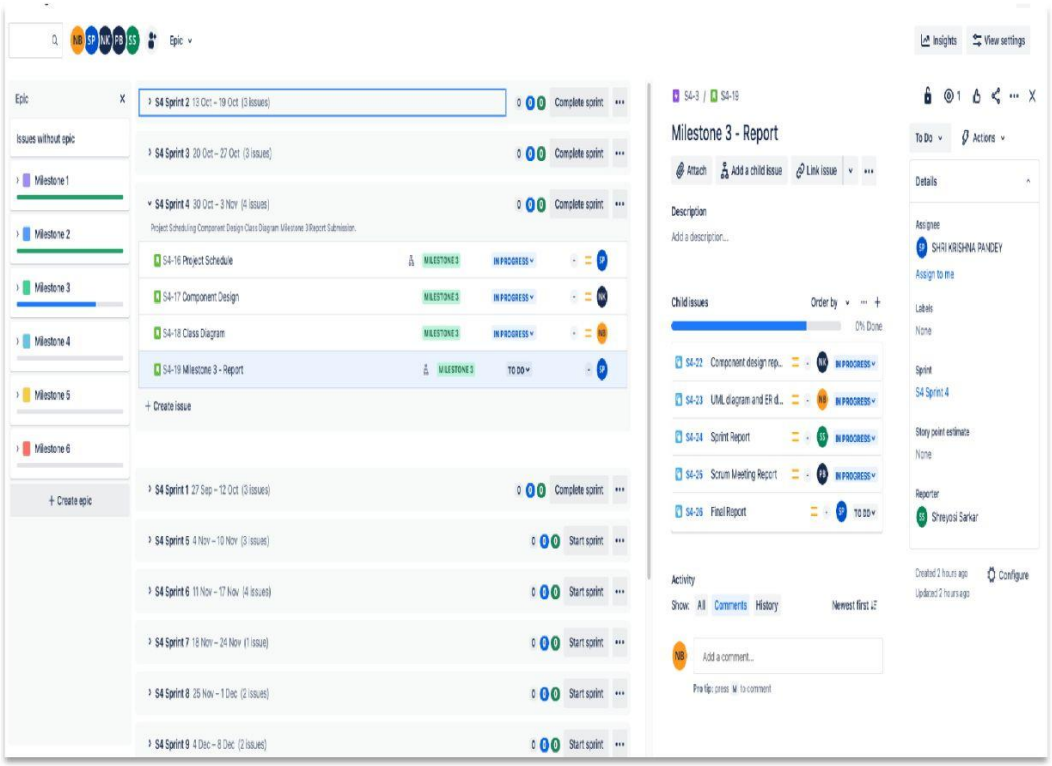
- Sprint 10 (09/12/2023 - 15/12/2023) - Final Project Report and Project Showcase (Upcoming)

  In the final sprint of our project, we will compile the final project report, summarizing the entire development process, design decisions, and outcomes. This report will serve as a comprehensive documentation of our project. We will also prepare for the Project Showcase, where we will present our project to the instructors and peers, demonstrating the key features and functionality of our recommendation system.



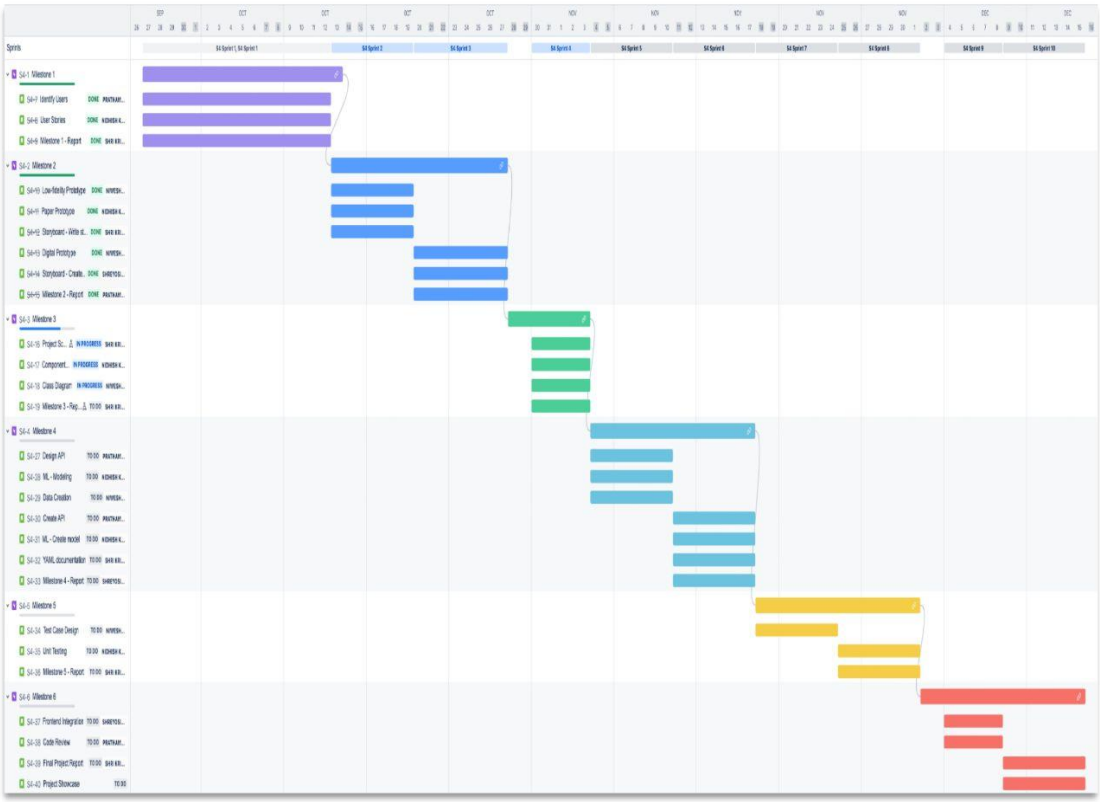
1.1.a. Dividing Sub-Tasks

Here we have also sub divided a few child issues among ourselves.



1.2 GANTT CHART

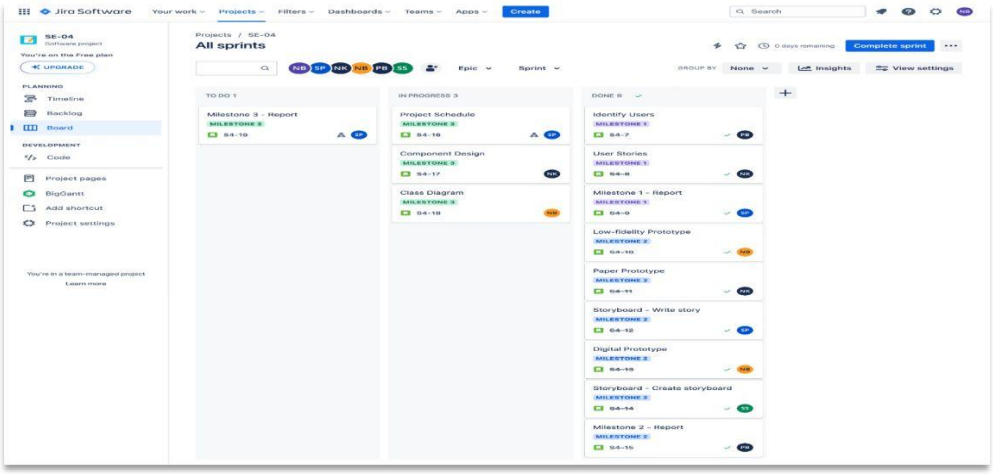
The Gantt-Chart for the project schedule is shown below. For the high-resolution image of the chart, please click here Gantt Chart Timeline



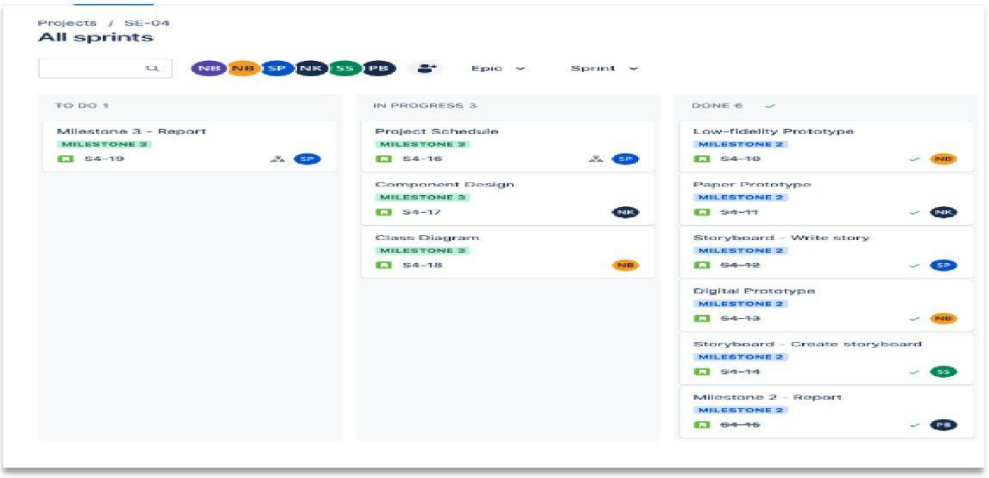
1.3 SCRUM BOARD

The scrum board consists of 'To Do', 'In Progress' and 'Done' for the active sprint. The scrum board for the milestones till now (completed and in progress) is shown below.

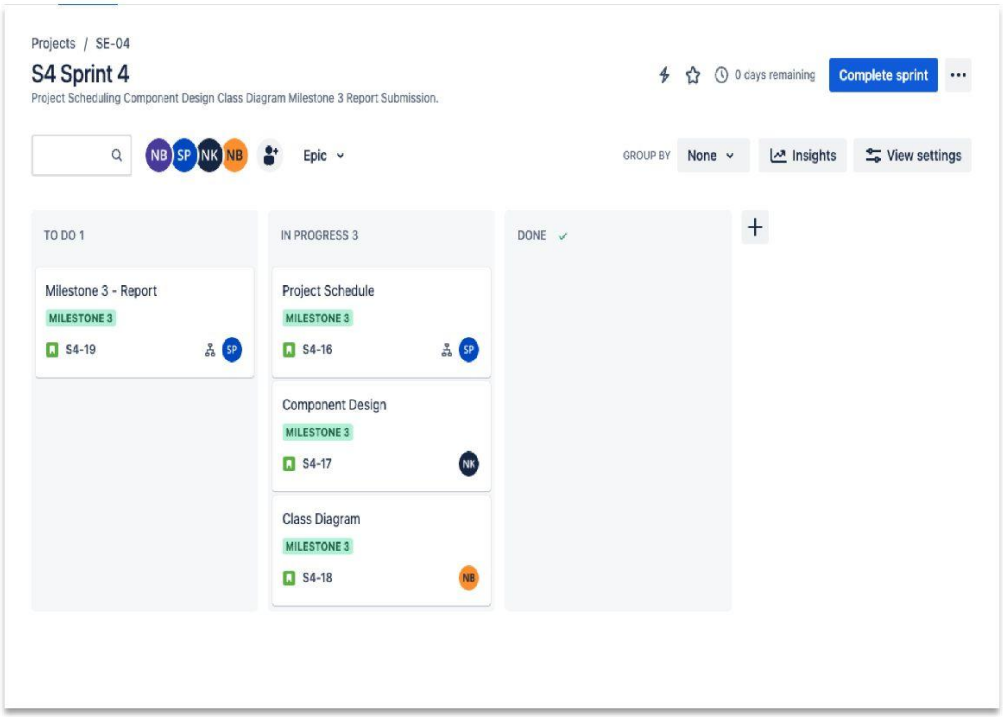
- This photo is after completing milestone 1 and sprint 1 of Scrum board.



- This photo is after completing milestone 2 and sprint 2 & 3.



- This photo is before milestone 3 completion.



1.4 SCRUM MEETING

Few Scrum meeting details are given below:

Meet 1

Date: 27-09-2023, Day: Wednesday, Time: 6 pm

Attendees

- Niwesh
- Pratham
- Nidhish
- Shri Krishna
- Shreyosi

Agenda

- Understanding problem statement
- Introduction with role assignment

Discussion Results

- Introduction
- Discussion about User Stories Requirements
- Discussion on Milestone 1, Assigned Tasks
- WhatsApp Group Creation for General Discussion
- Sheet Creation for Centralized Project Details

Work Assignment

- Niwesh - Assigned to the frontend development of the app
- Pratham - Assigned to the backend development and API of the app
- Nidhish - Assigned to the backend development and API of the app
- Shri Krishna - Assigned to collect students' requirements in a recommendation system
- Shreyosi - Assigned to the frontend development of the app

Minutes Covered: 30 minutes.

Meet 5

Date: 18-10-2023, Day: Wednesday, Time: 6 pm

Attendees

- Niwesh
- Pratham
- Nidhish
- Shri Krishna
- Shreyosi

Agenda

- Milestone 2 - Work Review session
- Future Work Assignment

Discussion Results

- Prototypes Review: The team reviewed incomplete prototypes, discussing their progress and identifying areas that require further development.
- Storyboard Development: The team discussed the creation of a storyboard, focusing on the script and the tools for building it. The goal was to define the narrative structure and select appropriate tools for this task.
- Initiation of Storyboard Development:
  - Niwesh, Shri Krishna and Pratham began brainstorming ideas for the story that will be depicted in the storyboard.
  - Niwesh and Shreyosi took the initiative to start developing the storyboard using storyboardthat.com.

Work Assignment

- Niwesh - Create storyboard
- Pratham - Write script for the storyboard
- Nidhish - Complete previous work assignment
- Shri Krishna - Write script for the storyboard
- Shreyosi - Create storyboard

Minutes Covered: 25 minutes.

Sprint 2

Meet 6

Date: 19-10-2023, Day: Thursday, Time: 7 pm

Attendees

- Niwesh
- Pratham
- Nidhish
- Shri Krishna
- Shreyosi

Agenda

- Milestone 2 - Work Review session
- Create Storyboard

Discussion Results

- Storyboard Development:
  - Shri Krishna and Pratham framed the story by writing the script for the story and collaborating in the development of the storyboard.
- Wireframe and PDF Front Page:
  - Niwesh was responsible for designing the cover page of the PDF containing the Storyboard and Wireframe. Additionally, Niwesh and Shreyosi worked on the Wireframe design.
- All the tasks of Milestone 2 were reviewed.
- Paper Prototype:
  - Nidhish and Niwesh collaborated in the creation of the paper prototype, a crucial step in the development process.

Work Assignment

- Niwesh - Continue working on storyboard and wireframe
- Pratham - Export storyboard images and create PDF using Xournal
- Nidhish - Print and take pictures for paper prototype and submit the milestone 2
- Shri Krishna - Export storyboard images and create PDF using Xournal
- Shreyosi - Continue working on storyboard and wireframe

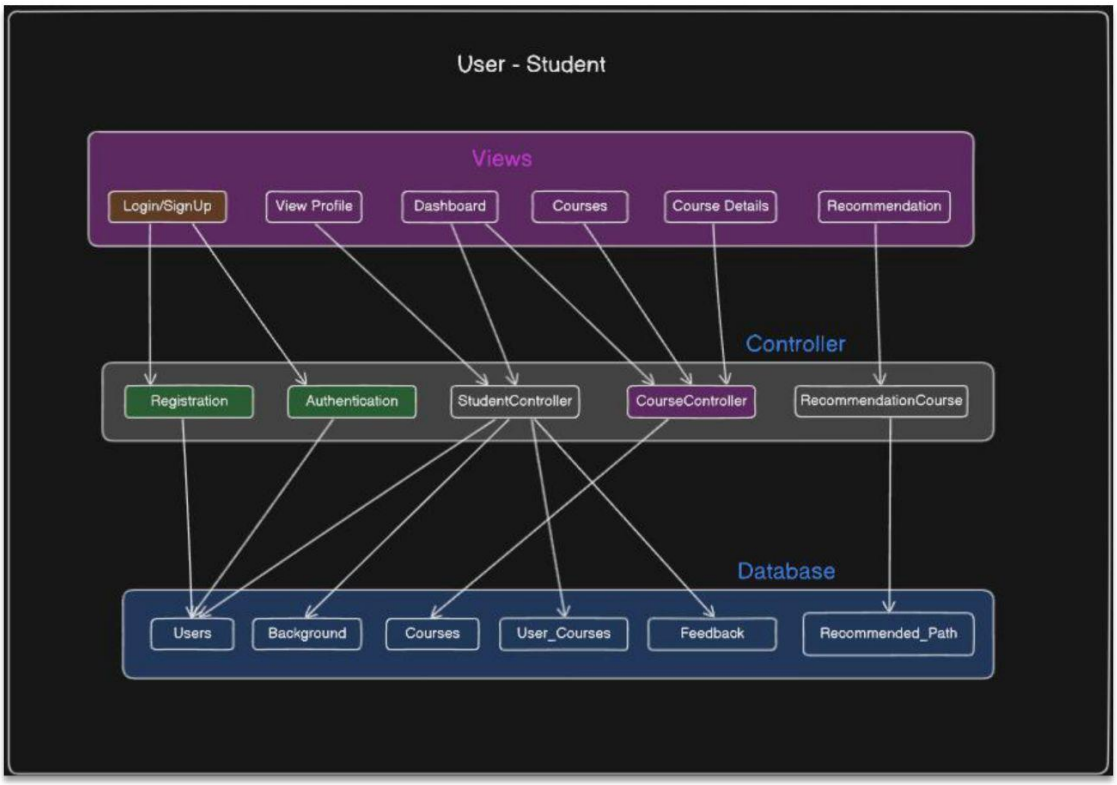
Minutes Covered: 30 minutes.

Sprint 2

All Scrum meeting details till 2/Nov/2023 are in the given link: Scrum Meeting Details

2. COMPONENT DESIGN

Components as per User- Student

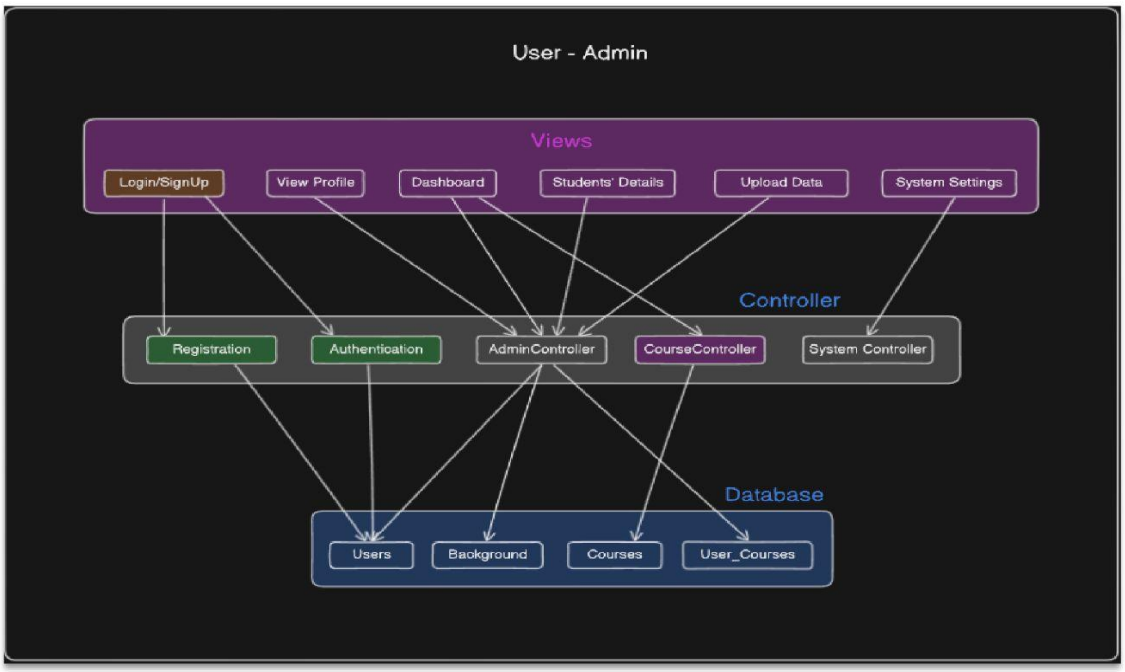


Student Component:

- Login/Signup: This view component uses the Registration and Authentication controller, which interacts with the Users database.
- Dashboard: The Dashboard view is managed by the StudentController, which accesses data from the Users, Background, User_courses, and Feedback databases.
- Courses: The Courses view is supported by the CourseController, connecting to the Courses database.
- View Profile: This component relies on the StudentController, which utilizes data from the Users, Background, User_courses, and Feedback databases.
- Course Details: The Course Details view is powered by the CourseController, fetching information from the Courses database.

- Recommendation: The Recommendation view is influenced by the RecommendationCourse controller, which interacts with data from the Users, Courses, Feedback, and Recommendation_path databases.

Components as per User- Admin

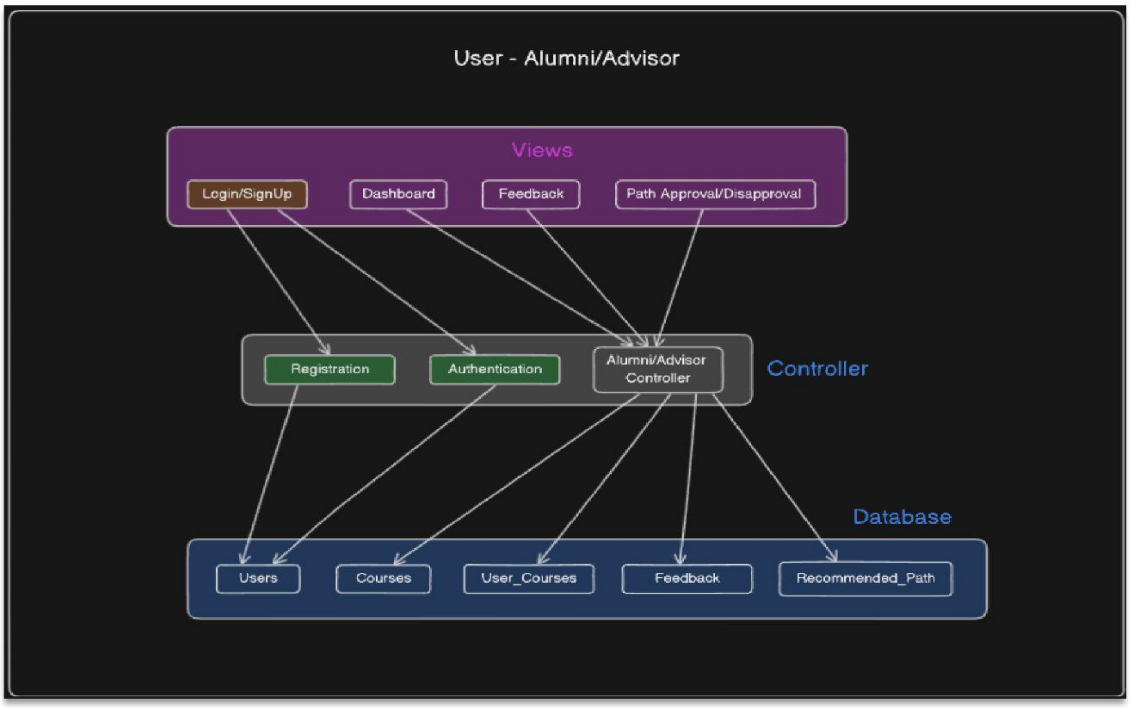


Admin Component:

- Login/Signup: This view component is responsible for user authentication and registration, utilizing the Registration and Authentication controllers. It accesses user account data stored in the Users database.
- View Profile: This component, managed by the AdminController, allows administrators to access and update user profiles, incorporating data from the Users, Background, and User_courses databases.
- Dashboard: The Dashboard view, overseen by the AdminController and CourseController, provides administrators with an organized overview of the system. It fetches data from the Users, Background, User_courses, and Courses databases.

- Students' Details: This view component, managed by the AdminController, focuses on providing administrators with detailed information about students. It accesses data from the Users, Background, and User_courses databases.
- Upload Data: This component, under the management of the AdminController, allows administrators to upload and manage user-related data. It interacts with the Users, Background, and User_courses databases.
- System Settings: The System Settings view, under the jurisdiction of the SystemController, enables administrators to configure system parameters and settings, enhancing system performance and functionality. It will call various system related API calls.

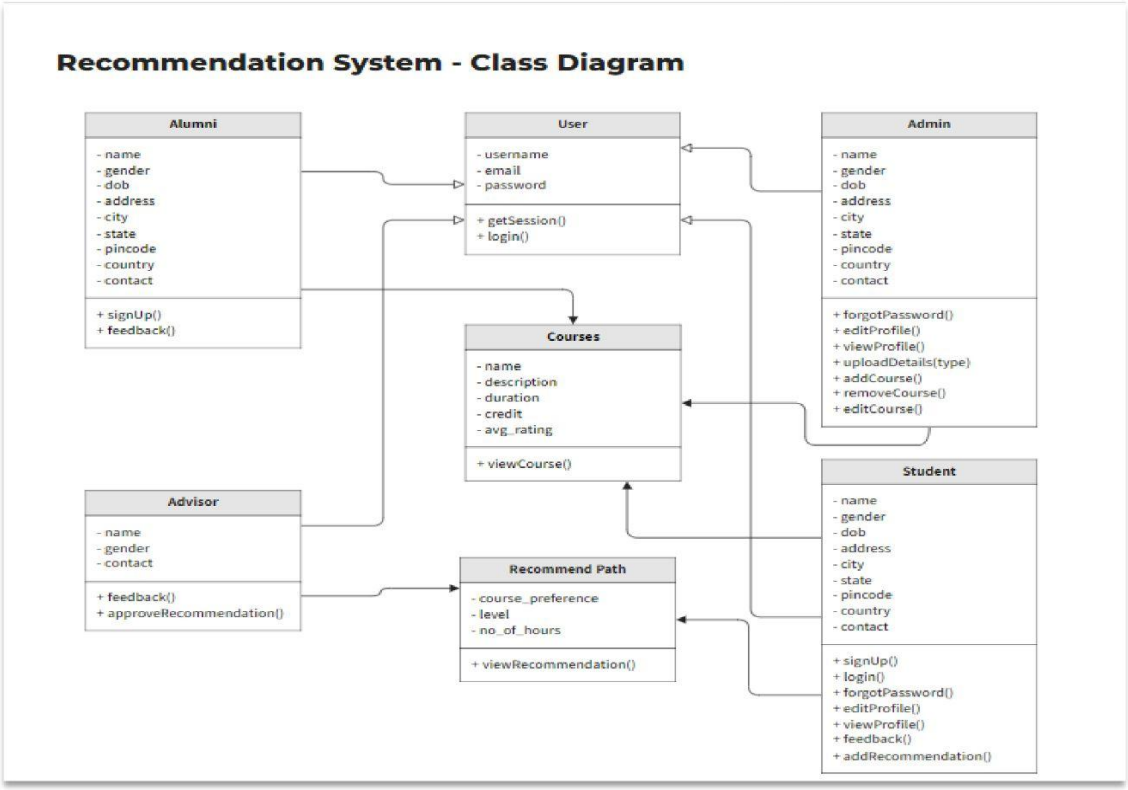
Components as per User- Alumni/Advisor



Alumni/Advisor Component:

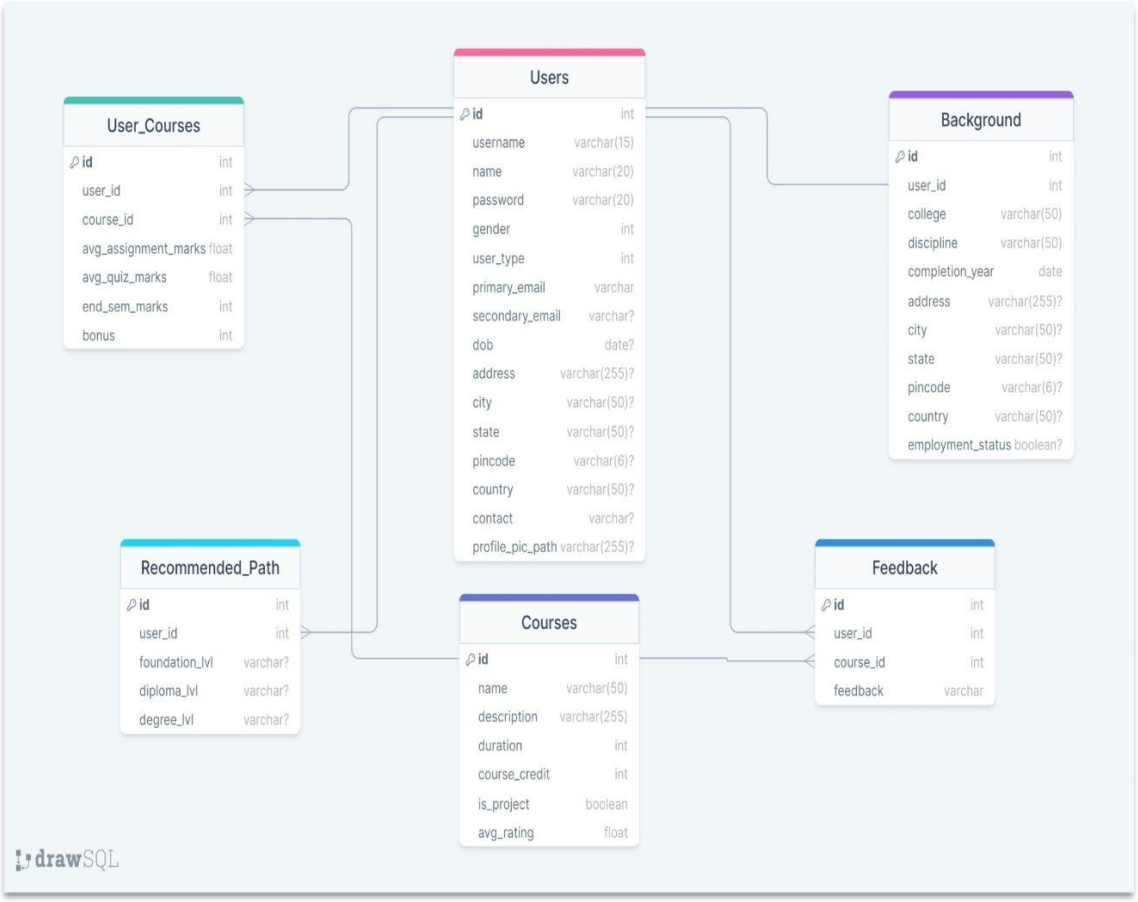
- Login / Signup: This view component will use the Registration and Authentication controller which is going to use the Users table in the database.
- Dashboard, Feedback and Path Approval / Disapproval: These view components will use the functionalities of Admin/Advisor Controller which will use Courses, User_Courses, Feedback and Recommended_Path table in the database.

3. CLASS DIAGRAM



- The class diagram is shown above. For the high-resolution image of the chart click here: **CLASS DIAGRAM LINK**

4. DB SCHEMA



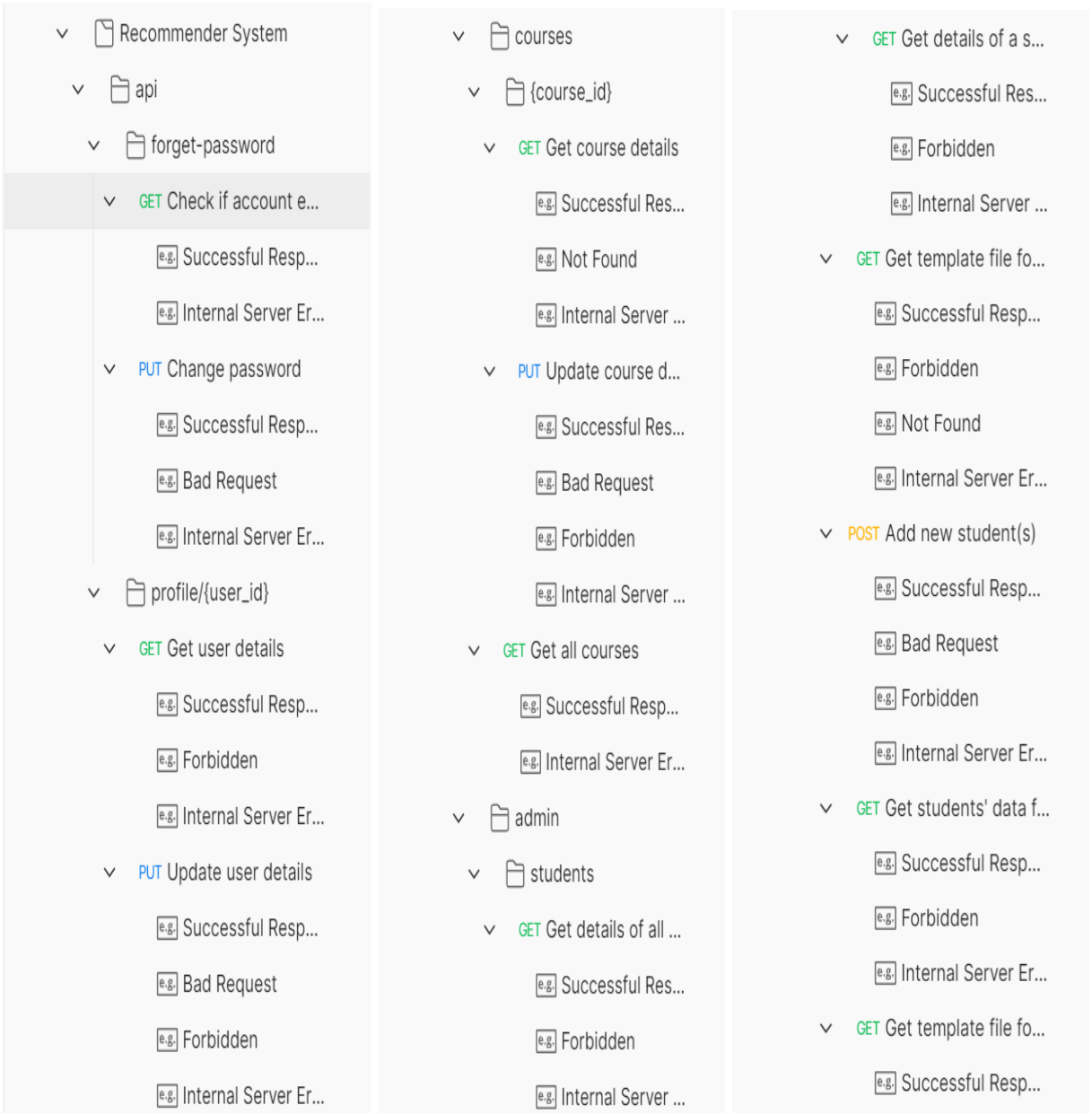
➢ For the high-resolution image of the DB Schema click here: [DB SCHEMA LINK]

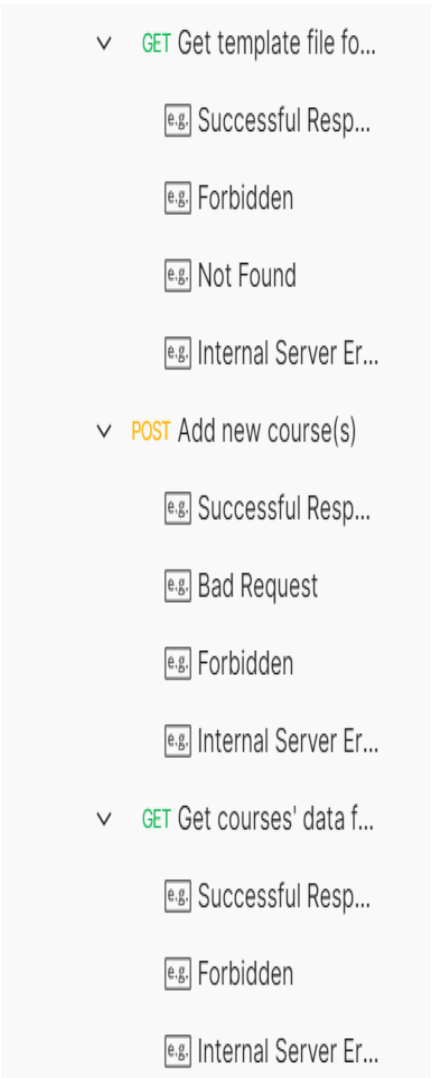
App Development

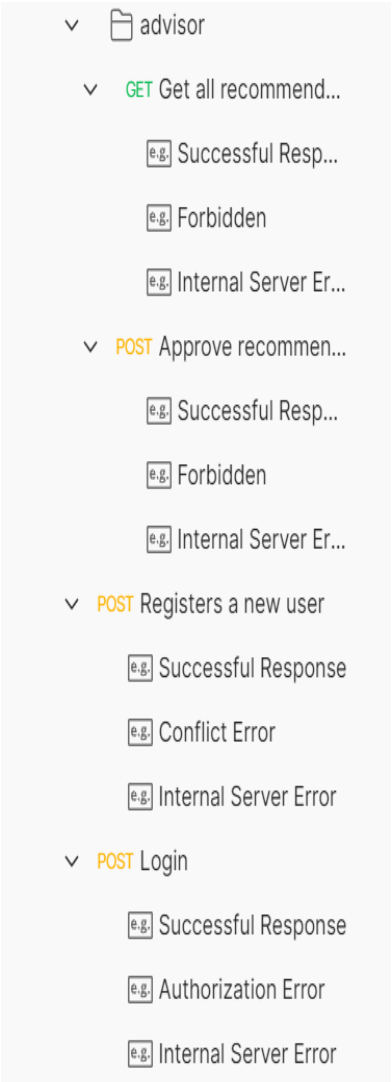
API Endpoints

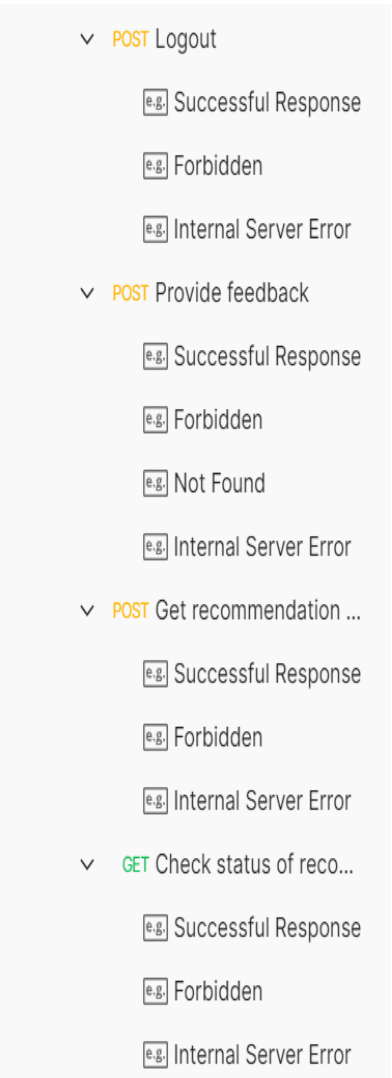
Below are the screenshots of api endpoints which we have used in the backend part of the code link for the same is provided below for better understanding in the yaml file.

Api Yaml:
https://github.com/bsc-iitm/soft-engg-project-sept-2023-se-sept-04/blob/main/Milestone-4/Milestone%204.yaml




GET Get template file fo...
Successful Resp...
Forbidden
Not Found
Internal Server Er...
POST Add new course(s)
Successful Resp...
Bad Request
Forbidden
Internal Server Er...
GET Get courses' data f...
Successful Resp...
Forbidden
Internal Server Er...


advisor
GET Get all recommend...
Successful Resp...
Forbidden
Internal Server Er...
POST Approve recommen...
Successful Resp...
Forbidden
Internal Server Er...
POST Registers a new user
Successful Response
Conflict Error
Internal Server Error
POST Login
Successful Response
Authorization Error
Internal Server Error


POST Logout
Successful Response
Forbidden
Internal Server Error
POST Provide feedback
Successful Response
Forbidden
Not Found
Internal Server Error
POST Get recommendation ...
Successful Response
Forbidden
Internal Server Error
GET Check status of reco...
Successful Response
Forbidden
Internal Server Error

API Endpoint being tested: http://127.0.0.1:8000/api/register

```
def test_user_registration_valid_data():
    input_dict = {
                    "username": "string",
                    "name": "string",
                    "password": "string",
                    "gender": 0,
                    "primary_email": "string"
                }
    data = json.dumps(input_dict)
    header = {"Content-Type": "application/json"}
    response = client.post(user_registration_endpoint, data=data, headers=header)
    assert response.status_code == 201
    assert response.json() == "Successfully Registered!"
```

Inputs:

- Request Method: POST
- JSON: {“username”: “string”, “name”: “string”, “password”: “string”, “gender”: 0, “primary_email”: “string”}

Expected Output:

- HTTP Status Code: 201
- JSON: {“Successfully Registered!”}

Actual Output:

- HTTP Status Code: 201
- JSON: {“Successfully Registered!”}

Result: Success

1

API Endpoint being tested: http://127.0.0.1:8000/api/login

```
def test_user_login_valid_data():
    input_dict = {
                    "email": "string",
                    "password": "string"
                }
    data = json.dumps(input_dict)
    header = {"Content-Type": "application/json"}
    response = client.post(user_login_endpoint, data=data, headers=header)
    assert response.status_code == 200
    assert response.json()['id'] is not None
    assert response.json()['token'] is not None
```

Inputs:

- Request Method: POST
- JSON: {“email”: “dummy@gmail.com”, “password”: “string”}

Expected Output:

- HTTP Status Code: 200
- JSON: {“id”: 1, “token”: “some string”}

Actual Output:

- HTTP Status Code: 200
- JSON: {“id”: 1, “token”: “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IjIxZjEwMDM3NThAZHMuc3R1ZHkuaWl0bS5hYy5pbiIsImV4cGlyZXMiOjE3MDE4ODUxNTUuMjE3NzMwNX0._Ht0U-TXm1t1qGE2ICwnuAUOfodbKx00n5HPUHhCGBA“}

Result: Success

2

API Endpoint being tested: http://127.0.0.1:8000/api/forget-password

```python
def test_user_forget_password_get_valid_data():

    endpoint = f"{user_forget_password_endpoint}?email={valid_email}"
    header = {"Content-Type": "application/json"}
    response = client.get(endpoint, headers=header)
    assert response.status_code == 200
    assert response.json()
```

Inputs:

- Request Method: GET
- JSON: {“email”: “dummy@gmail.com”}

Expected Output:

- HTTP Status Code: 200
- JSON: {true}

Actual Output:

- HTTP Status Code: 200
- JSON: {true}

Result: Success

3

API Endpoint being tested: http://127.0.0.1:8000/api/forget-password

```python
def test_user_forget_password_put_valid_data():

    input_dict = {
                    "email": valid_email,
                    "new_password": "string"
                }

    data = json.dumps(input_dict)
    header = {"Content-Type": "application/json"}
    response = client.put(user_forget_password_endpoint, data=data, headers=header)
    assert response.status_code == 202
    assert response.json() == "Successfully Changed!"
```

Inputs:

- Request Method: PUT
- JSON: {"email": "dummy@gmail.com", "new_password": "random_password"}

Expected Output:

- HTTP Status Code: 202
- JSON: {"`Successfully Changed!`"}

Actual Output:

- HTTP Status Code: 202
- JSON: {"`Successfully Changed!`"}

Result: Success

Pytest Outputs for the above endpoints

```
$ python -m pytest -v tests/test_user_apis.py
============================= test session starts =============================
platform win32 -- Python 3.10.11, pytest-7.2.1, pluggy-1.0.0 -- C:\Users\HP\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: D:\GitHub\Academic\Recommendation-System
plugins: anyio-3.7.0, shutil-1.7.0, typeguard-2.13.3
collected 4 items

tests/test_user_apis.py::test_user_registration_valid_data PASSED        [ 25%]
tests/test_user_apis.py::test_user_login_valid_data PASSED               [ 50%]
tests/test_user_apis.py::test_user_forget_password_get_valid_data PASSED [ 75%]
tests/test_user_apis.py::test_user_forget_password_put_valid_data PASSED [100%]

============================== warnings summary ===============================
tests/test_user_apis.py::test_user_registration_valid_data
tests/test_user_apis.py::test_user_login_valid_data
tests/test_user_apis.py::test_user_forget_password_put_valid_data
  C:\Users\HP\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\httpx\_content.py:204: DeprecationWarning: Use `co
ntent=<...>` to upload raw bytes/text content.
    warnings.warn(message, DeprecationWarning)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================== 4 passed, 3 warnings in 1.17s ========================
```

API Endpoint being tested: http://127.0.0.1:8000/api/profile/{user_id}

Request Method: GET

```
def test_get_valid_user_data():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(f"{user_profile_endpoint}/{valid_user_id}", headers=header)
    assert response.status_code == 200
    assert response.json()['username'] is not None
    assert response.json()['name'] is not None
    assert response.json()['primary_email'] is not None
    assert response.json()['profile_pic'] is not None
```

API Endpoint being tested: http://127.0.0.1:8000/api/logout

Request Method: POST

```
def test_valid_user_logout():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(user_logout_endpoint, headers=header)
    assert response.status_code == 200
    assert response.json() == "Successfully logged out!"
```

API Endpoint being tested: http://127.0.0.1:8000/api/courses

Request Method: GET

```
def test_get_all_courses():

    header = {"Content-Type": "application/json"}
    response = client.get(courses_endpoint, headers=header)
    assert response.status_code == 200
    assert response.json() is not []
```

5

API Endpoint being tested:
http://127.0.0.1:8000/api/courses/{course_id}

Request Method: GET

```
def test_get_valid_course():

    header = {"Content-Type": "application/json"}
    response = client.get(f"{courses_endpoint}/{valid_course_id}", headers=header)
    assert response.status_code == 200
    assert response.json()['name'] != ""
    assert response.json()['course_id'] != ""
```

API Endpoint being tested:
http://127.0.0.1:8000/api/courses/{course_id}

Request Method: PUT

```
def test_update_valid_course():

    input_data = {
                    "course_id": "string",
                    "name": "string",
                    "description": "string",
                    "duration": 0,
                    "course_credit": 0,
                    "is_project": True,
                    "status": True,
                    "level": 0
                }

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.put(f"{courses_endpoint}/{valid_course_id}", headers=header)
    assert response.status_code == 202
    assert response.json() == "Course details updated successfully!"
```

API Endpoint being tested: http://127.0.0.1:8000/api/admin/students

Request Method: GET

```
def test_get_all_students_details():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(all_students_endpoint, headers=header)
    assert response.status_code == 200
    assert response.json() != []
```

API Endpoint being tested:
http://127.0.0.1:8000/api/admin/students/{roll}

Request Method: GET

```
def test_get_student_details():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(specific_student_endpoint, headers=header)
    assert response.status_code == 200
    assert response.json()['name'] != ""
    assert response.json()['roll'] != ""
    assert response.json()['level'] != ""
    assert response.json()['status'] is not None
```

API Endpoint being tested:
http://127.0.0.1:8000/api/admin/students-graph-data

Request Method: GET

```
def test_get_students_graph_data():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(students_graph_endpoint, headers=header)
    assert response.status_code == 200
```

API Endpoint being tested:
http://127.0.0.1:8000/api/admin/course-graph-data

Request Method: GET

```
def test_get_courses_graph_data():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(courses_graph_endpoint, headers=header)
    assert response.status_code == 200
```

API Endpoint being tested:
http://127.0.0.1:8000/api/advisor/recommendation-paths

Request Method: GET

```
def test_get_all_recommendation_paths():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(all_recommendation_paths_endpoint, headers=header)
    assert response.status_code == 200
```

API Endpoint being tested:
http://127.0.0.1:8000/api/advisor/recommendation-path/{path_id}

Request Method: POST

```
def test_approve_recommendation_path():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.post(f"{recommendation_path_endpoint}/{valid_path_id}", headers=header)
    assert response.status_code == 204
    assert response.json() in ["Recommended path approved successfully!",
                               "Recommended path disapproved successfully!"]
```

API Endpoint being tested:
http://127.0.0.1:8000/api/recommendation-path

Request Method: POST

```
def test_get_recommendation_paths():

    input_dict = {
                    "n_hours": "string"
                }

    data = json.dumps(input_dict)
    header = {"token": get_token(), "Content-Type": "application/json", data: data}
    response = client.post(recommendation_path_endpoint, headers=header)
    assert response.status_code == 200
```

8

API Endpoint being tested:
http://127.0.0.1:8000/api/recommendation-paths/status

Request Method: GET

```python
def test_get_recommendation_paths_status():

    header = {"token": get_token(), "Content-Type": "application/json"}
    response = client.get(recommendation_path_status_endpoint, headers=header)
    assert response.status_code == 200
```

API Endpoint being tested:
http://127.0.0.1:8000/api/feedback/{course_id}

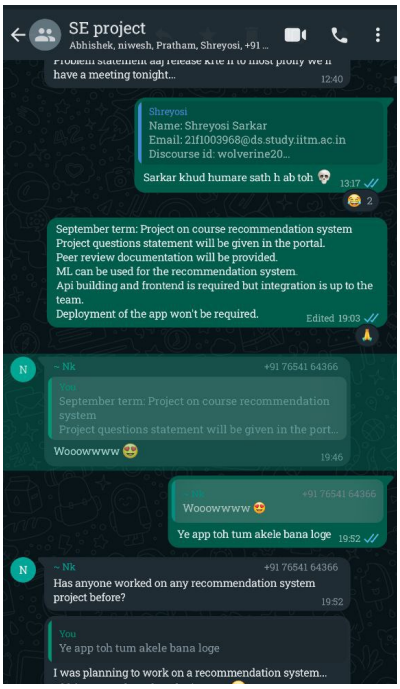
Request Method: POST

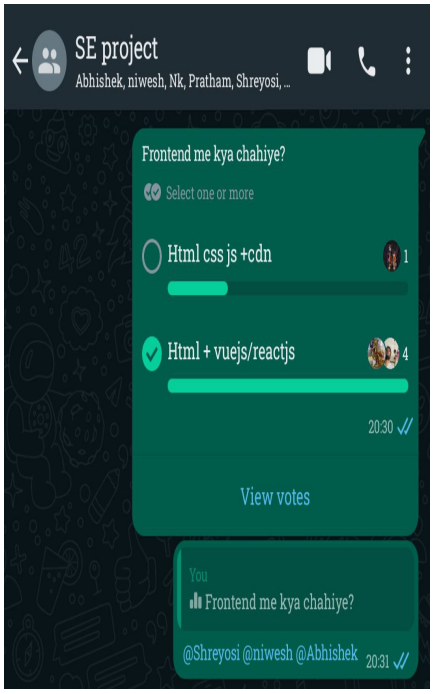
```python
def test_valid_course_feedback():

    input_dict = {
                    "feedback": "string"
                }

    data = json.dumps(input_dict)
    header = {"token": get_token(), "Content-Type": "application/json", data: data}
    response = client.post(f"{course_feedback_endpoint}/{valid_course_id}", headers=header)
    Explain Code | Ask Sourcery
    assert response.status_code == 200
    assert response.json() == "Feedback provided successfully!"
```

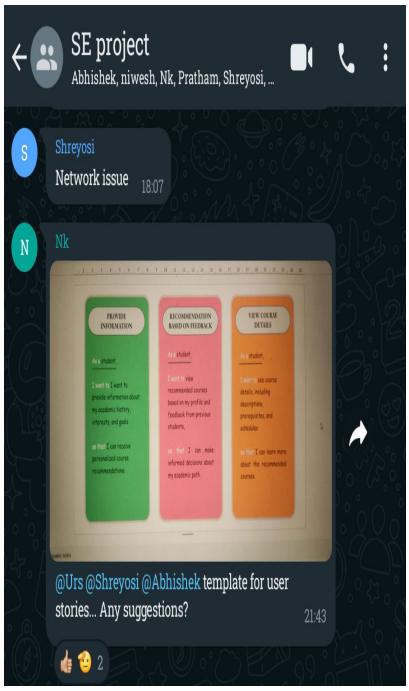
Issue Tracking and Reporting

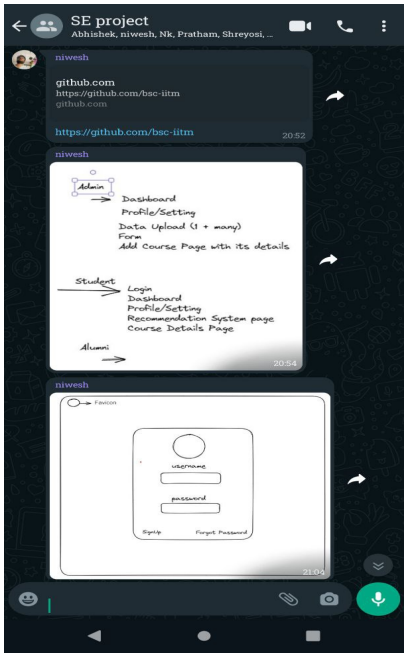
We have created a whatsapp group for constant communication and easy access to review or code. Below are a few screenshots attached to view. We also have attached github screenshots which we have used during building our app.

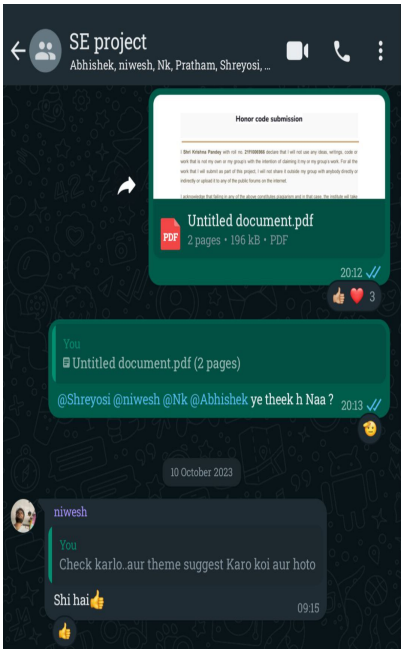
Whatsapp Screenshots:

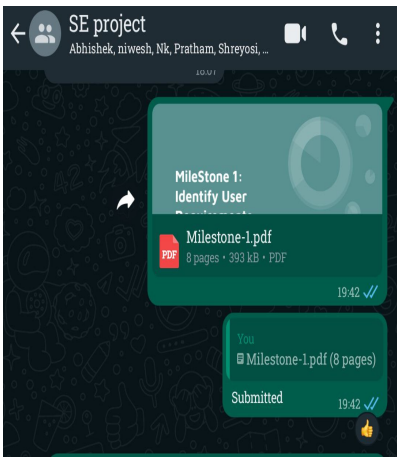


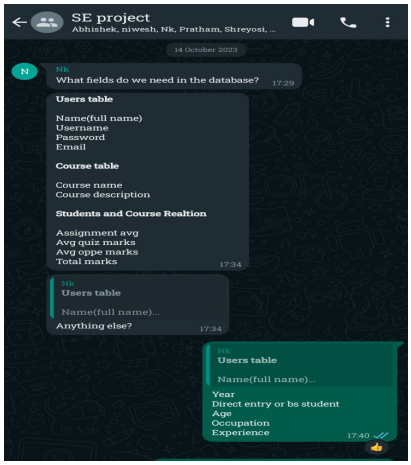


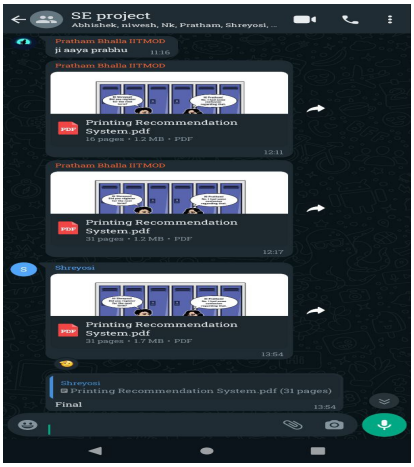


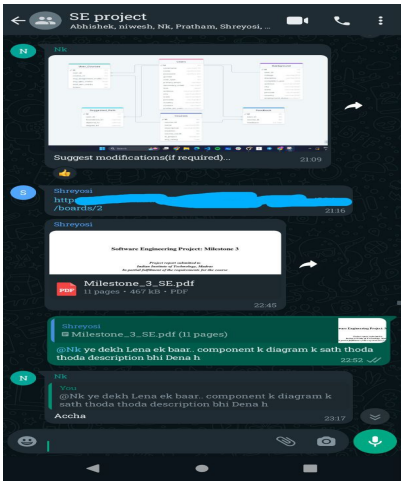


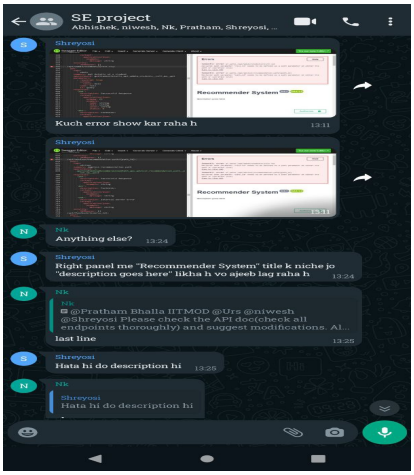




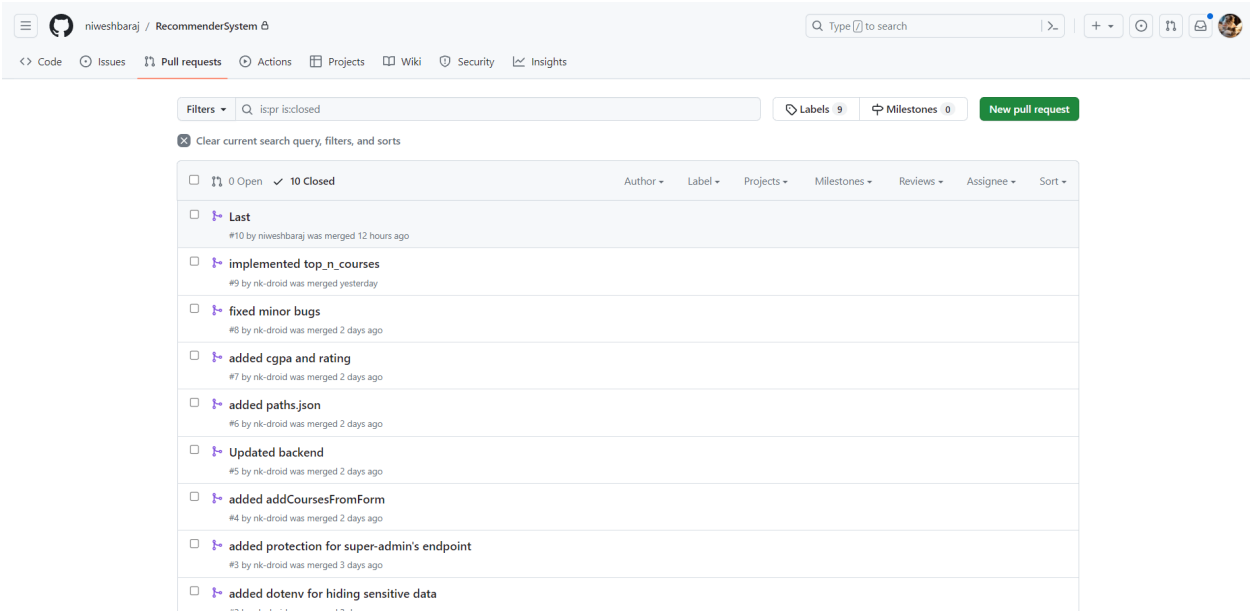








Github Screenshot:



Tech Stack Used

- **GitHub** for collaboration
- **VsCode** for build app locally
- **Whatsapp** for communication
- **G-Doc, G-Sheet, G-Slides, G-Meet** for creating report and storing meetings details
- **Jira** for work assignment and creating deadlines

Tech Stack used

- Frontend
  - React
- Backend (including API doc)
  - Flask
- Design
  - Figma/Canva
- Documentation(Contents)
  - Google Doc
- Progress Tracking
  - Jira - All
  - GitHub - All

Ends.