MILESTONE-6
REPORT

- KASHIFUL HAQUE
- TEJAS GAJRA
- ANANYA MODUGULA
- AARYA MOTIWALA
- MD FARHAN REZA

DECEMBER 2023

Index

Problem Statement
(Chapter 1)

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.

Software Engineering Project

Milestone 1

- Identifying Primary, Secondary and Tertiary Users
- User Stories for the requirements based on SMART guidelines

1. Identifying Primary, Secondary and Tertiary Users

- Primary Users
  - Students
  - Administrators
- Secondary Users
  - Software Developers
  - Data Scientists
- Tertiary Users
  - Third Party apps

2. Writing User stories

Students

- As a student
  - I want to be able to login/register
    - So that I can access my dashboard.
- As a student
  - I want to create a profile that includes:
    - My learning preferences (do I go with Diploma in Programming or Diploma in Data Science first, or a mix of both).
    - My past performance (GPA from my past terms).
    - My interests (preferred subjects)
      - I can choose Programming in Java first, or
      - I can choose to do DSA first, or
      - I can start with App development or DBMS, or
      - I can get the fundamentals of ML, etc.
    - My long-term career goals, like
      - Landing a job at a preferred company/role.

        - Becoming a software engineer or data scientist, etc.
- As a student
    - I want to provide feedback on courses that I have taken in the past
        - So that, the system can consider my feedback in generating recommendations for other students.
- As a student
    - I want to input my current schedule and commitments
        - So that, the system can take them into account while generating recommendations (number of courses).
- As a student
    - I want to receive a recommended learning path based on my profile
        - So that, I can make an informed decision about my course selection.
- As a student
    - I want to search the feedback of other students
        - So that, I can see what other students think about a course.
    - I also want to search the feedback withing a particular timeframe
        - So that, I can get an idea on how the course has changed over time.
- As a student
    - I want to be able to access this platform from any device
        - So that, I can get course recommendations at my convenience.
- As a student
    - I want to be able to modify my profile and preferences
        - So that, the system can adapt its recommendations to my changing needs and goals.

Administrators

- As an admin
    - I want to be able to login
        - So that, I can access my dashboard.
- As an admin
    - I want to be able to add/remove/modify courses
        - So that, the students always get the most up-to-date information about the courses.
- As an admin
    - I want to be able to manager user accounts of students like adding/removing/modifying features
        - So that, the student database is up-to-date and correct.

- As an admin
  - I want to be able to upload and manage past enrolment data
    - So that processed data can be used by the recommendation engine.
- As an admin
  - I want to be able to tag the feedback for course properly
    - So that, the feedback given by the students are organized.
- As an admin
  - I want to be able to delete a feedback if it contains any harmful or profane text
    - So that, students can work in a safe and formal environment.

Software Developers and Data Scientists

- As a software developer
  - I want to access well-documented APIs for use data retrieval and course recommendation
    - So that, I can seamless integrate the learning path recommendation system into our existing educational platform.
- As a data scientist
  - I want to access student's preference, courses and past performance data along with the course data and enrolment data from the admins
    - So that, I can a recommender system model which can generate predictions for students.

General

- As a user
  - I want the system to provide clear, precise and understandable recommendations
    - So that, an easy path can be formed out of it.
- As a user
  - I want the system to consider aggregate data (not individual) to improve recommendations
    - So that, the recommendations are more accurate and beneficial.
- As a user
  - I want the system to have a user-friendly UI
    - So that, it is easy to navigate.

Milestone-2
(Chapter 3)

Milestone 2

Storyboard: Course Recommendation system

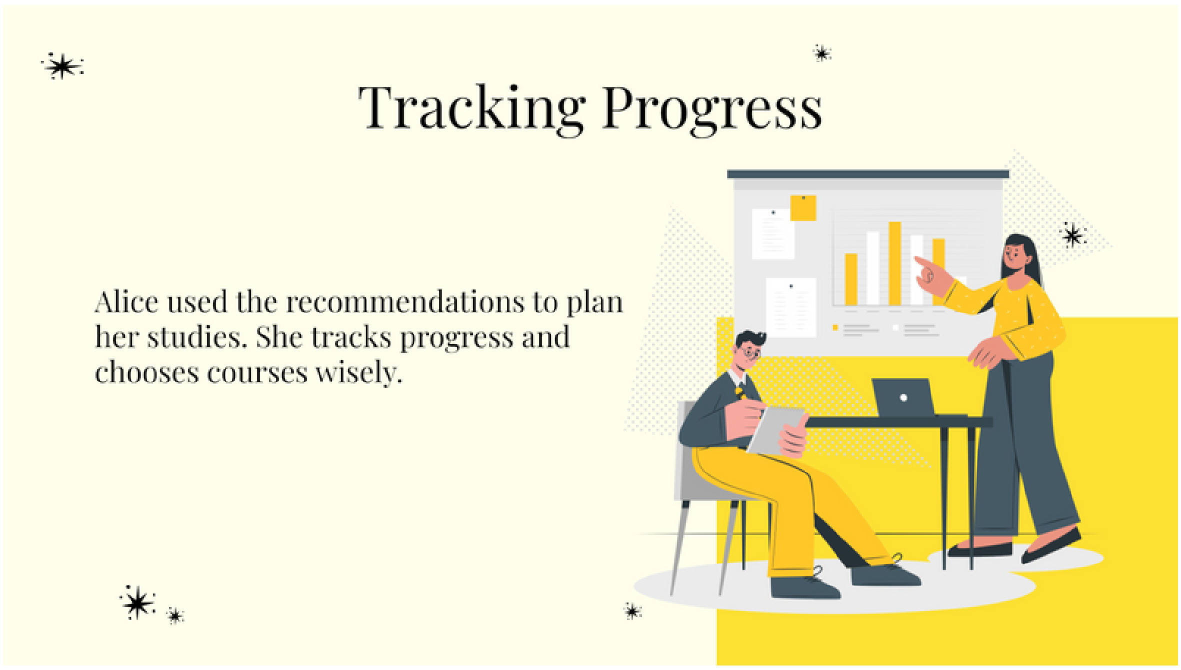
Group Number : 14

Problem statement

Once upon a time, there was a student named Alice who was struggling to choose the right courses for her degree.

Milestone-2

(Chapter 3)



Tracking Progress

Alice used the recommendations to plan her studies. She tracks progress and chooses courses wisely.

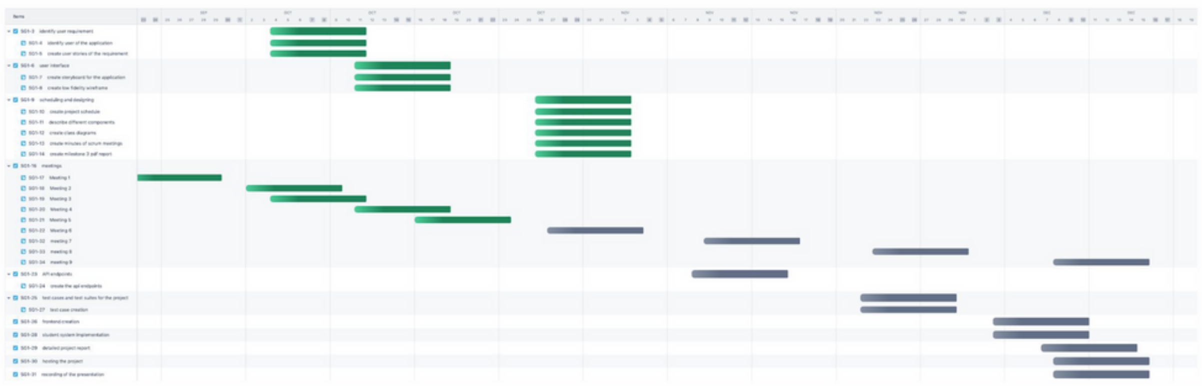
Click here to view entire story board

Milestone-3

(Chapter 4)

CONTENTS

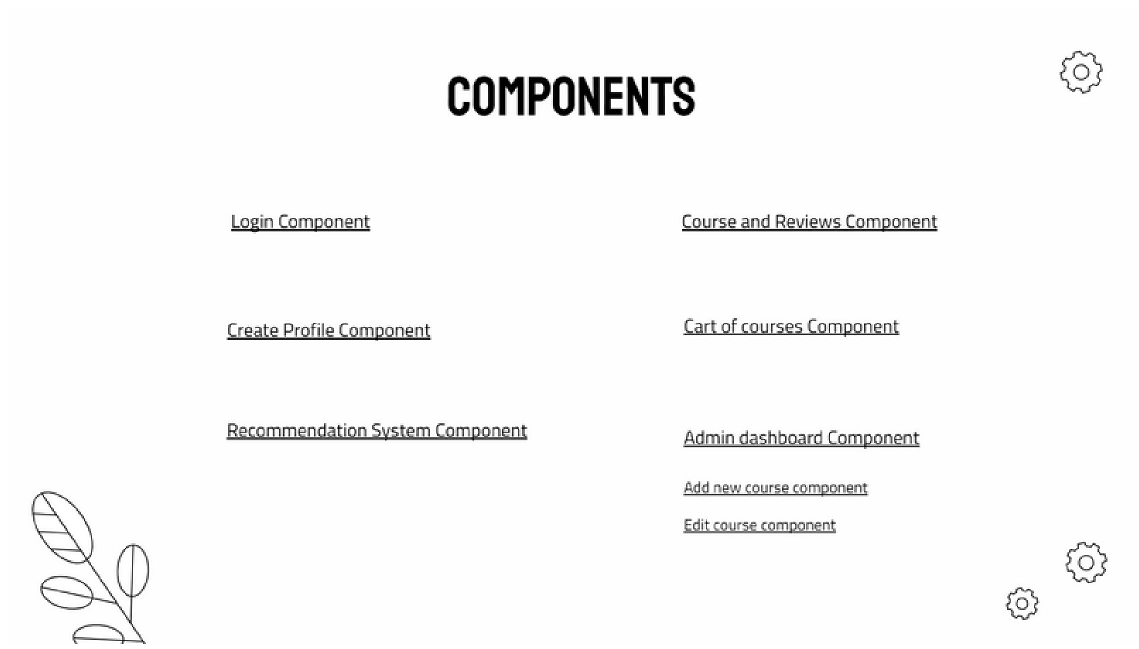
GANTT CHART



click to view full res gantt chart

Milestone-3

(Chapter 4)



COMPONENTS

Login Component

Create Profile Component

Recommendation System Component

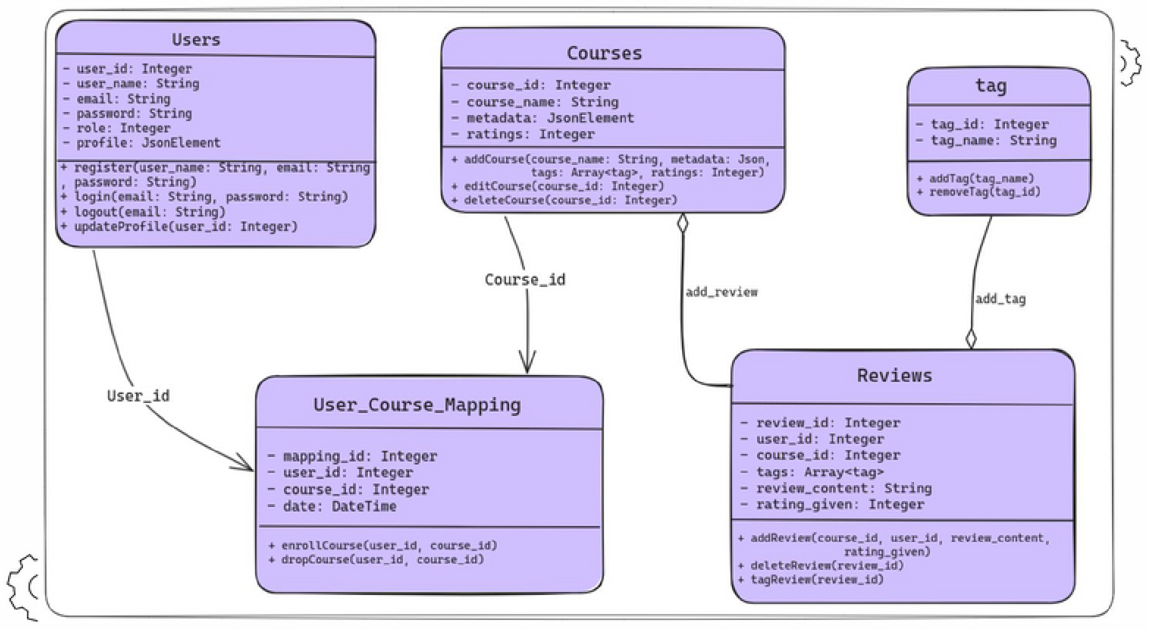
Course and Reviews Component

Cart of courses Component

Admin dashboard Component

Add new course component

Edit course component



Click here to view milestone 3

Milestone-4

(Chapter 5)

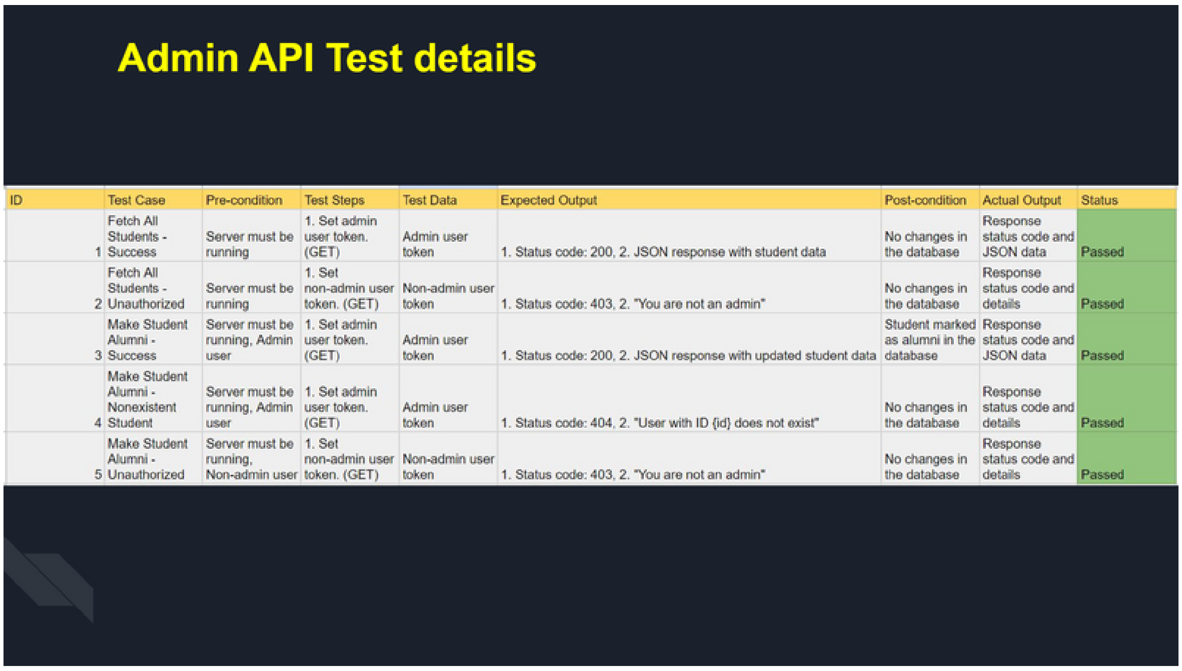
Task

- Create and Describe API endpoints as per the problem statement
- Submit all the details of the API endpoints in a YAML file.

Click here to view milestone 4

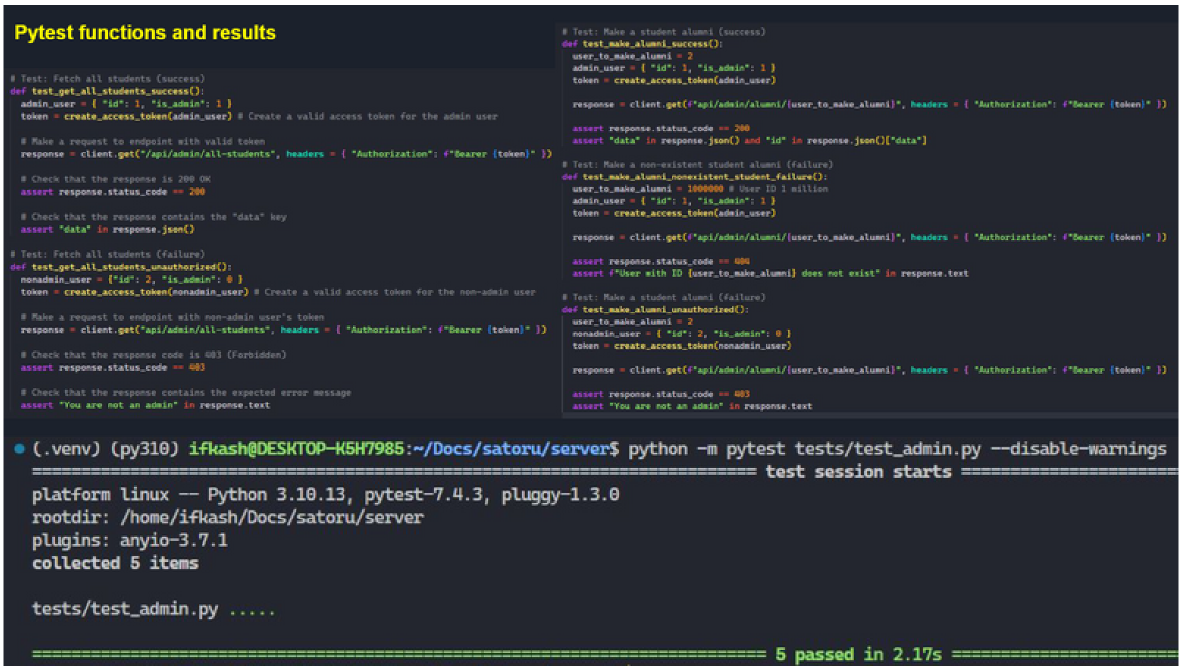
Milestone-5

(Chapter 6)



Admin API Test details

ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Output	Post-condition	Actual Output	Status
1	Fetch All Students - Success	Server must be running	1. Set admin user token. (GET)	Admin user token	1. Status code: 200, 2. JSON response with student data	No changes in the database	Response status code and JSON data	Passed
2	Fetch All Students - Unauthorized	Server must be running	1. Set non-admin user token. (GET)	Non-admin user token	1. Status code: 403, 2. "You are not an admin"	No changes in the database	Response status code and details	Passed
3	Make Student Alumni - Success	Server must be running, Admin user	1. Set admin user token. (GET)	Admin user token	1. Status code: 200, 2. JSON response with updated student data	Student marked as alumni in the database	Response status code and JSON data	Passed
4	Make Student Alumni - Nonexistent Student	Server must be running, Admin user	1. Set admin user token. (GET)	Admin user token	1. Status code: 404, 2. "User with ID {id} does not exist"	No changes in the database	Response status code and details	Passed
5	Make Student Alumni - Unauthorized	Server must be running, Non-admin user	1. Set non-admin user token. (GET)	Non-admin user token	1. Status code: 403, 2. "You are not an admin"	No changes in the database	Response status code and details	Passed



Pytest functions and results

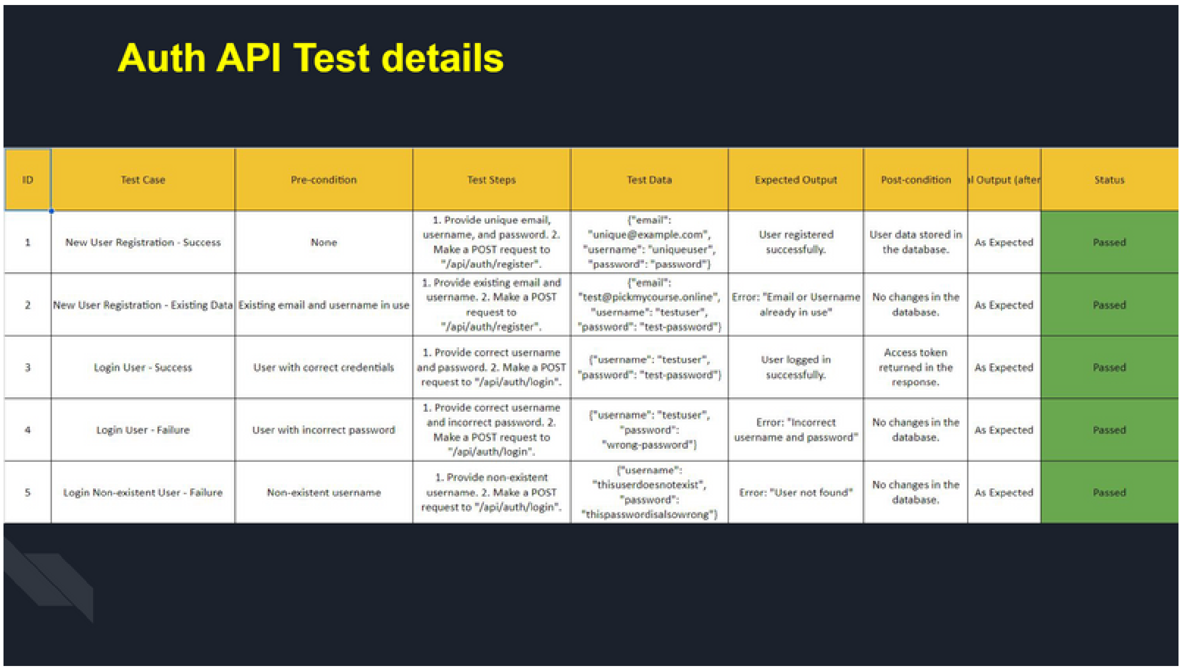
```python
# Test: Fetch all students (success)
def test_get_all_students_success():
  admin_user = { "id": 1, "is_admin": 1 }
  token = create_access_token(admin_user) # Create a valid access token for the admin user

  # Make a request to endpoint with valid token
  response = client.get("/api/admin/all-students", headers = { "Authorization": f"Bearer {token}" })

  # Check that the response is 200 OK
  assert response.status_code == 200

  # Check that the response contains the "data" key
  assert "data" in response.json()

# Test: Fetch all students (failure)
def test_get_all_students_unauthorized():
  nonadmin_user = {"id": 2, "is_admin": 0 }
  token = create_access_token(nonadmin_user) # Create a valid access token for the non-admin user

  # Make a request to endpoint with non-admin user's token
  response = client.get("api/admin/all-students", headers = { "Authorization": f"Bearer {token}" })

  # Check that the response code is 403 (Forbidden)
  assert response.status_code == 403

  # Check that the response contains the expected error message
  assert "You are not an admin" in response.text
```

```python
# Test: Make a student alumni (success)
def test_make_alumni_success():
  user_to_make_alumni = 2
  admin_user = { "id": 1, "is_admin": 1 }
  token = create_access_token(admin_user)

  response = client.get(f"api/admin/alumni/{user_to_make_alumni}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 200
  assert "data" in response.json() and "id" in response.json()["data"]

# Test: Make a non-existent student alumni (failure)
def test_make_alumni_nonexistent_student_failure():
  user_to_make_alumni = 1000000 # User ID 1 million
  admin_user = { "id": 1, "is_admin": 1 }
  token = create_access_token(admin_user)

  response = client.get(f"api/admin/alumni/{user_to_make_alumni}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 404
  assert f"User with ID {user_to_make_alumni} does not exist" in response.text

# Test: Make a student alumni (failure)
def test_make_alumni_unauthorized():
  user_to_make_alumni = 2
  nonadmin_user = { "id": 2, "is_admin": 0 }
  token = create_access_token(nonadmin_user)

  response = client.get(f"api/admin/alumni/{user_to_make_alumni}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 403
  assert "You are not an admin" in response.text
```

```
● (.venv) (py310) ifkash@DESKTOP-K5H7985:~/Docs/satoru/server$ python -m pytest tests/test_admin.py --disable-warnings
  =========================================================== test session starts ==================
  platform linux -- Python 3.10.13, pytest-7.4.3, pluggy-1.3.0
  rootdir: /home/ifkash/Docs/satoru/server
  plugins: anyio-3.7.1
  collected 5 items

  tests/test_admin.py .....

  ============================================================ 5 passed in 2.17s ===================
```

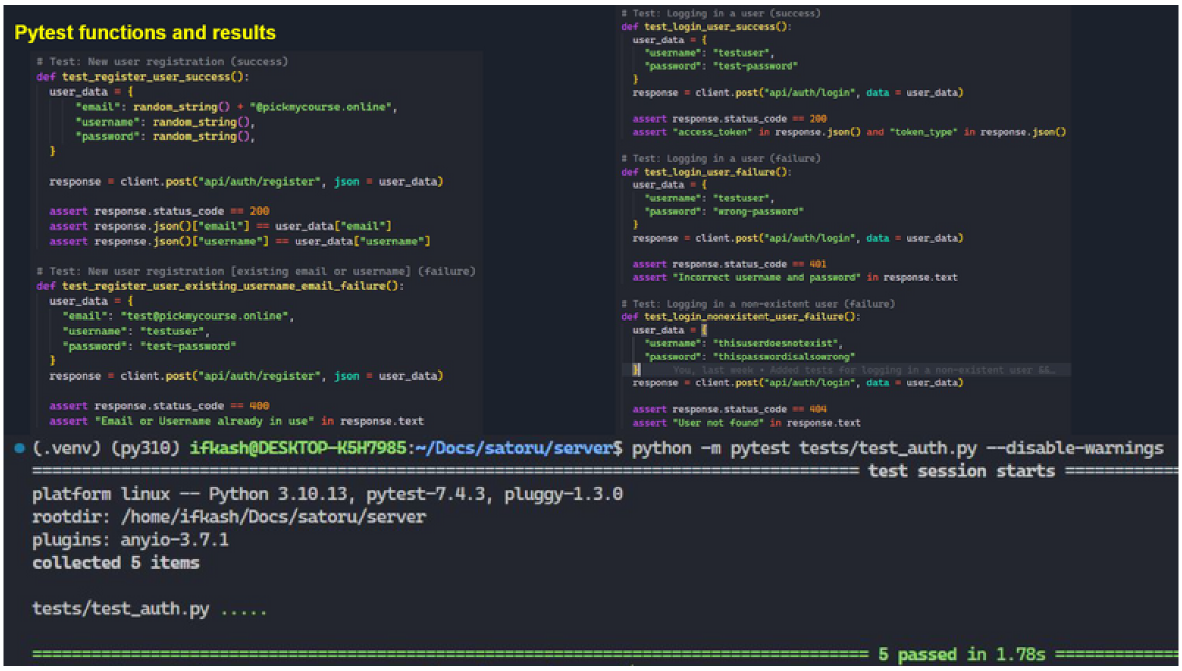
Milestone-5

(Chapter 6)



Auth API Test details

ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Output	Post-condition	l Output (after	Status
1	New User Registration - Success	None	1. Provide unique email, username, and password. 2. Make a POST request to "/api/auth/register".	{"email": "unique@example.com", "username": "uniqueuser", "password": "password"}	User registered successfully.	User data stored in the database.	As Expected	Passed
2	New User Registration - Existing Data	Existing email and username in use	1. Provide existing email and username. 2. Make a POST request to "/api/auth/register".	{"email": "test@pickmycourse.online", "username": "testuser", "password": "test-password"}	Error: "Email or Username already in use"	No changes in the database.	As Expected	Passed
3	Login User - Success	User with correct credentials	1. Provide correct username and password. 2. Make a POST request to "/api/auth/login".	{"username": "testuser", "password": "test-password"}	User logged in successfully.	Access token returned in the response.	As Expected	Passed
4	Login User - Failure	User with incorrect password	1. Provide correct username and incorrect password. 2. Make a POST request to "/api/auth/login".	{"username": "testuser", "password": "wrong-password"}	Error: "Incorrect username and password"	No changes in the database.	As Expected	Passed
5	Login Non-existent User - Failure	Non-existent username	1. Provide non-existent username. 2. Make a POST request to "/api/auth/login".	{"username": "thisuserdoesnotexist", "password": "thispasswordisalsowrong"}	Error: "User not found"	No changes in the database.	As Expected	Passed



Pytest functions and results

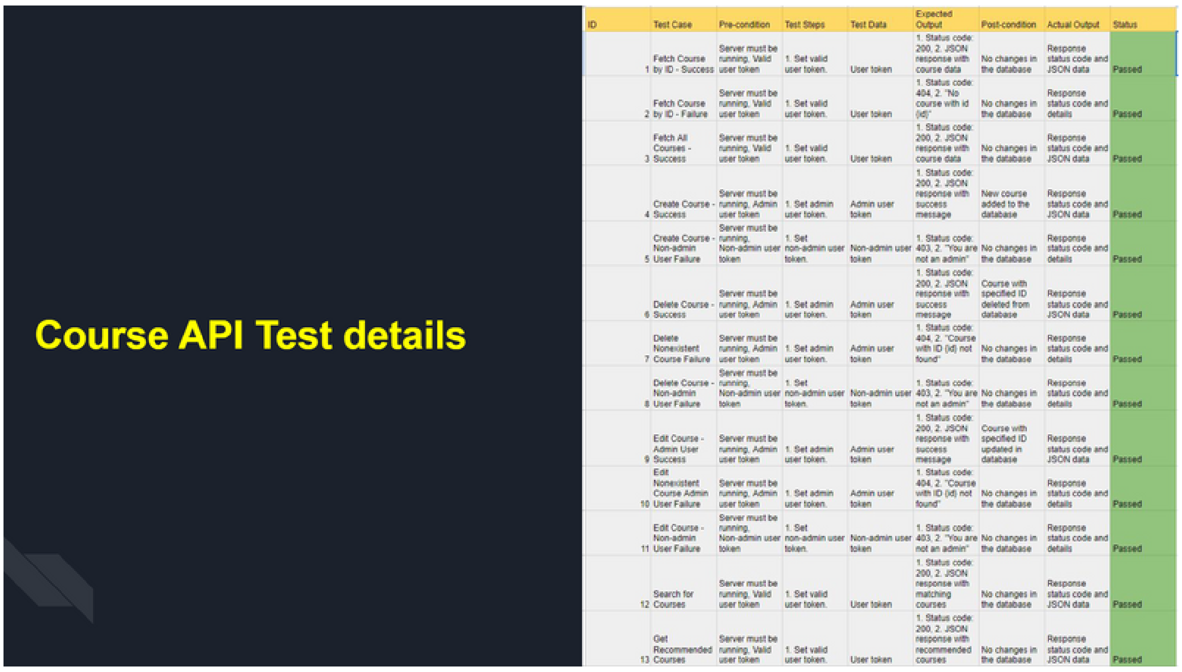
```python
# Test: New user registration (success)
def test_register_user_success():
  user_data = {
      "email": random_string() + "@pickmycourse.online",
      "username": random_string(),
      "password": random_string(),
  }

  response = client.post("api/auth/register", json = user_data)

  assert response.status_code == 200
  assert response.json()["email"] == user_data["email"]
  assert response.json()["username"] == user_data["username"]

# Test: New user registration [existing email or username] (failure)
def test_register_user_existing_username_email_failure():
  user_data = {
    "email": "test@pickmycourse.online",
    "username": "testuser",
    "password": "test-password"
  }
  response = client.post("api/auth/register", json = user_data)

  assert response.status_code == 400
  assert "Email or Username already in use" in response.text
```

```python
# Test: Logging in a user (success)
def test_login_user_success():
  user_data = {
    "username": "testuser",
    "password": "test-password"
  }
  response = client.post("api/auth/login", data = user_data)

  assert response.status_code == 200
  assert "access_token" in response.json() and "token_type" in response.json()

# Test: Logging in a user (failure)
def test_login_user_failure():
  user_data = {
    "username": "testuser",
    "password": "wrong-password"
  }
  response = client.post("api/auth/login", data = user_data)

  assert response.status_code == 401
  assert "Incorrect username and password" in response.text

# Test: Logging in a non-existent user (failure)
def test_login_nonexistent_user_failure():
  user_data = {
    "username": "thisuserdoesnotexist",
    "password": "thispasswordisalsowrong"
  }      You, last week • Added tests for logging in a non-existent user &&…
  response = client.post("api/auth/login", data = user_data)

  assert response.status_code == 404
  assert "User not found" in response.text
```

```
(.venv) (py310) ifkash@DESKTOP-K5H7985:~/Docs/satoru/server$ python -m pytest tests/test_auth.py --disable-warnings
==================================================================================== test session starts ============
platform linux -- Python 3.10.13, pytest-7.4.3, pluggy-1.3.0
rootdir: /home/ifkash/Docs/satoru/server
plugins: anyio-3.7.1
collected 5 items

tests/test_auth.py .....

===================================================================================== 5 passed in 1.78s ============
```

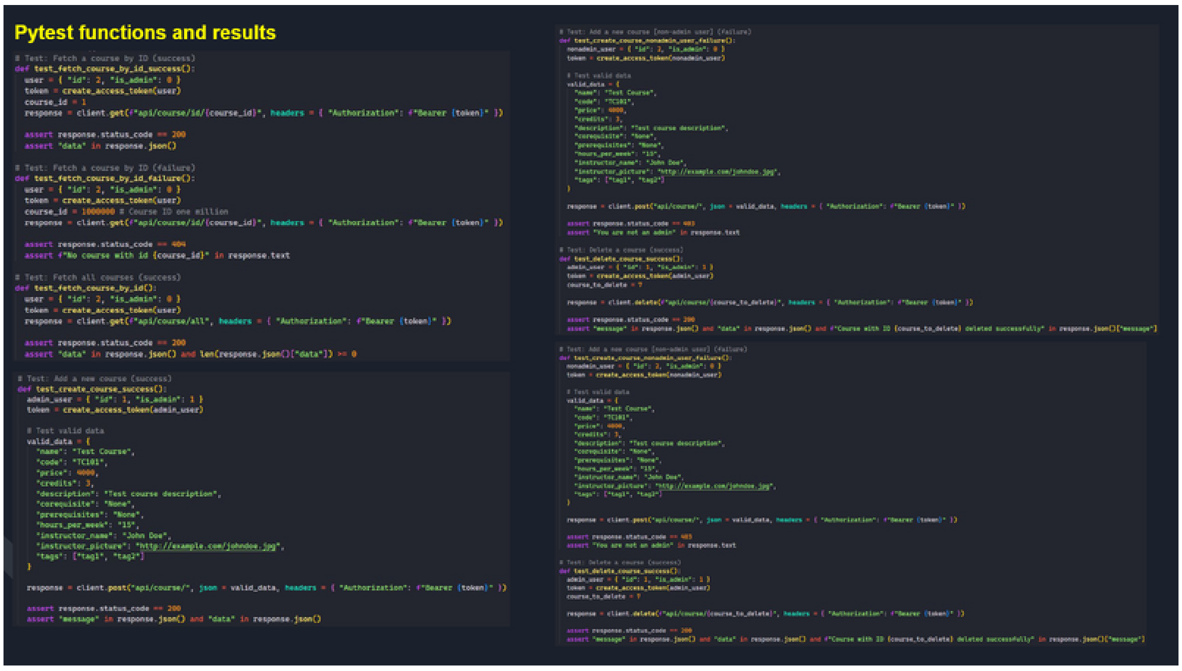
Milestone-5

(Chapter 6)



Course API Test details

ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Output	Post-condition	Actual Output	Status
1	Fetch Course by ID - Success	Server must be running, Valid user token	1. Set valid user token.	User token	1. Status code: 200, 2. JSON response with course data	No changes in the database	Response status code and JSON data	Passed
2	Fetch Course by ID - Failure	Server must be running, Valid user token	1. Set valid user token.	User token	1. Status code: 404, 2. "No course with id {id}"	No changes in the database	Response status code and details	Passed
3	Fetch All Courses - Success	Server must be running, Valid user token	1. Set valid user token.	User token	1. Status code: 200, 2. JSON response with course data	No changes in the database	Response status code and JSON data	Passed
4	Create Course - Success	Server must be running, Admin user token	1. Set admin user token.	Admin user token	1. Status code: 200, 2. JSON response with success message	New course added to the database	Response status code and JSON data	Passed
5	Create Course - Non-admin User Failure	Server must be running, Non-admin user token	1. Set non-admin user token.	Non-admin user token	1. Status code: 403, 2. "You are not an admin"	No changes in the database	Response status code and details	Passed
6	Delete Course - Success	Server must be running, Admin user token	1. Set admin user token.	Admin user token	1. Status code: 200, 2. JSON response with success message	Course with specified ID deleted from database	Response status code and JSON data	Passed
7	Delete Nonexistent Course Failure	Server must be running, Admin user token	1. Set admin user token.	Admin user token	1. Status code: 404, 2. "Course with ID {id} not found"	No changes in the database	Response status code and details	Passed
8	Delete Course - Non-admin User Failure	Server must be running, Non-admin user token	1. Set non-admin user token.	Non-admin user token	1. Status code: 403, 2. "You are not an admin"	No changes in the database	Response status code and details	Passed
9	Edit Course - Admin User Success	Server must be running, Admin user token	1. Set admin user token.	Admin user token	1. Status code: 200, 2. JSON response with success message	Course with specified ID updated in database	Response status code and JSON data	Passed
10	Edit Nonexistent Course Admin User Failure	Server must be running, Admin user token	1. Set admin user token.	Admin user token	1. Status code: 404, 2. "Course with ID {id} not found"	No changes in the database	Response status code and details	Passed
11	Edit Course - Non-admin User Failure	Server must be running, Non-admin user token	1. Set non-admin user token.	Non-admin user token	1. Status code: 403, 2. "You are not an admin"	No changes in the database	Response status code and details	Passed
12	Search for Courses	Server must be running, Valid user token	1. Set valid user token.	User token	1. Status code: 200, 2. JSON response with matching courses	No changes in the database	Response status code and JSON data	Passed
13	Get Recommended Courses	Server must be running, Valid user token	1. Set valid user token.	User token	1. Status code: 200, 2. JSON response with recommended courses	No changes in the database	Response status code and JSON data	Passed



Pytest functions and results

```python
# Test: Fetch a course by ID (success)
def test_fetch_course_by_id_success():
  user = { "id": 2, "is_admin": 0 }
  token = create_access_token(user)
  course_id = 1
  response = client.get(f"api/course/id/{course_id}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 200
  assert "data" in response.json()

# Test: Fetch a course by ID (failure)
def test_fetch_course_by_id_failure():
  user = { "id": 2, "is_admin": 0 }
  token = create_access_token(user)
  course_id = 1000000 # Course ID one million
  response = client.get(f"api/course/id/{course_id}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 404
  assert f"No course with id {course_id}" in response.text

# Test: Fetch all courses (success)
def test_fetch_course_by_id():
  user = { "id": 2, "is_admin": 0 }
  token = create_access_token(user)
  response = client.get(f"api/course/all", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 200
  assert "data" in response.json() and len(response.json()["data"]) >= 0
```

```python
# Test: Add a new course (success)
def test_create_course_success():
  admin_user = { "id": 1, "is_admin": 1 }
  token = create_access_token(admin_user)

  # Test valid data
  valid_data = {
    "name": "Test Course",
    "code": "TC101",
    "price": 4000,
    "credits": 3,
    "description": "Test course description",
    "corequisite": "None",
    "prerequisites": "None",
    "hours_per_week": "15",
    "instructor_name": "John Doe",
    "instructor_picture": "http://example.com/johndoe.jpg",
    "tags": ["tag1", "tag2"]
  }

  response = client.post("api/course/", json = valid_data, headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 200
  assert "message" in response.json() and "data" in response.json()
```

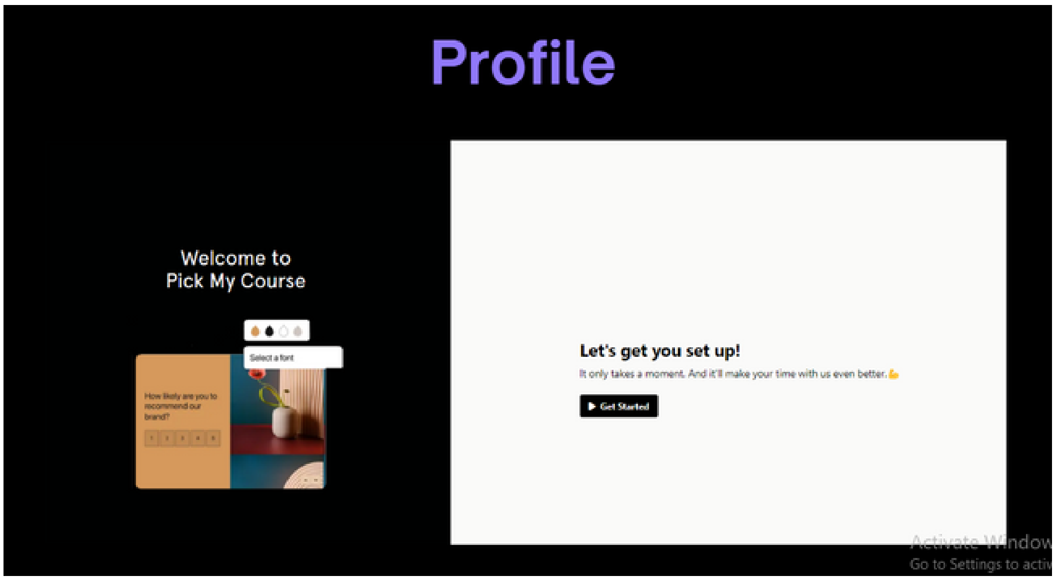
```python
# Test: Add a new course [non-admin user] (failure)
def test_create_course_nonadmin_user_failure():
  nonadmin_user = { "id": 2, "is_admin": 0 }
  token = create_access_token(nonadmin_user)

  # Test valid data
  valid_data = {
    "name": "Test Course",
    "code": "TC101",
    "price": 4000,
    "credits": 3,
    "description": "Test course description",
    "corequisite": "None",
    "prerequisites": "None",
    "hours_per_week": "15",
    "instructor_name": "John Doe",
    "instructor_picture": "http://example.com/johndoe.jpg",
    "tags": ["tag1", "tag2"]
  }

  response = client.post("api/course/", json = valid_data, headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 403
  assert "You are not an admin" in response.text

# Test: Delete a course (success)
def test_delete_course_success():
  admin_user = { "id": 1, "is_admin": 1 }
  token = create_access_token(admin_user)
  course_to_delete = 7

  response = client.delete(f"api/course/{course_to_delete}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 200
  assert "message" in response.json() and "data" in response.json() and f"Course with ID {course_to_delete} deleted successfully" in response.json()["message"]
```

```python
# Test: Add a new course [non-admin user] (failure)
def test_create_course_nonadmin_user_failure():
  nonadmin_user = { "id": 2, "is_admin": 0 }
  token = create_access_token(nonadmin_user)

  # Test valid data
  valid_data = {
    "name": "Test Course",
    "code": "TC101",
    "price": 4000,
    "credits": 3,
    "description": "Test course description",
    "corequisite": "None",
    "prerequisites": "None",
    "hours_per_week": "15",
    "instructor_name": "John Doe",
    "instructor_picture": "http://example.com/johndoe.jpg",
    "tags": ["tag1", "tag2"]
  }

  response = client.post("api/course/", json = valid_data, headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 403
  assert "You are not an admin" in response.text

# Test: Delete a course (success)
def test_delete_course_success():
  admin_user = { "id": 1, "is_admin": 1 }
  token = create_access_token(admin_user)
  course_to_delete = 7

  response = client.delete(f"api/course/{course_to_delete}", headers = { "Authorization": f"Bearer {token}" })

  assert response.status_code == 200
  assert "message" in response.json() and "data" in response.json() and f"Course with ID {course_to_delete} deleted successfully" in response.json()["message"]
```

Click here to view milestone 5

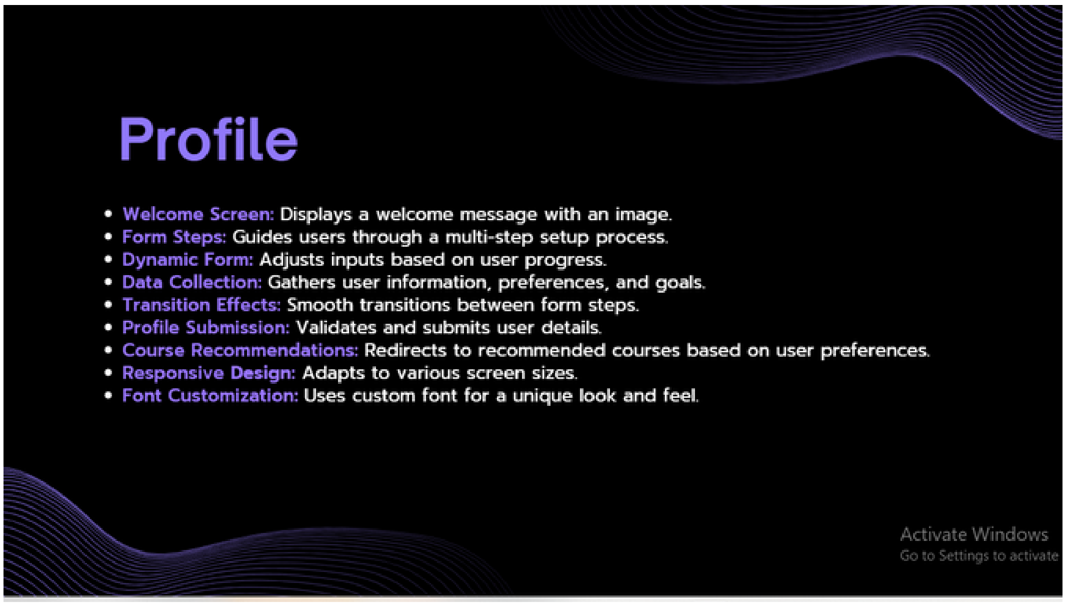
Milestone-6

(Chapter 7)



Let's get you set up!

It only takes a moment. And it'll make your time with us even better.



Profile

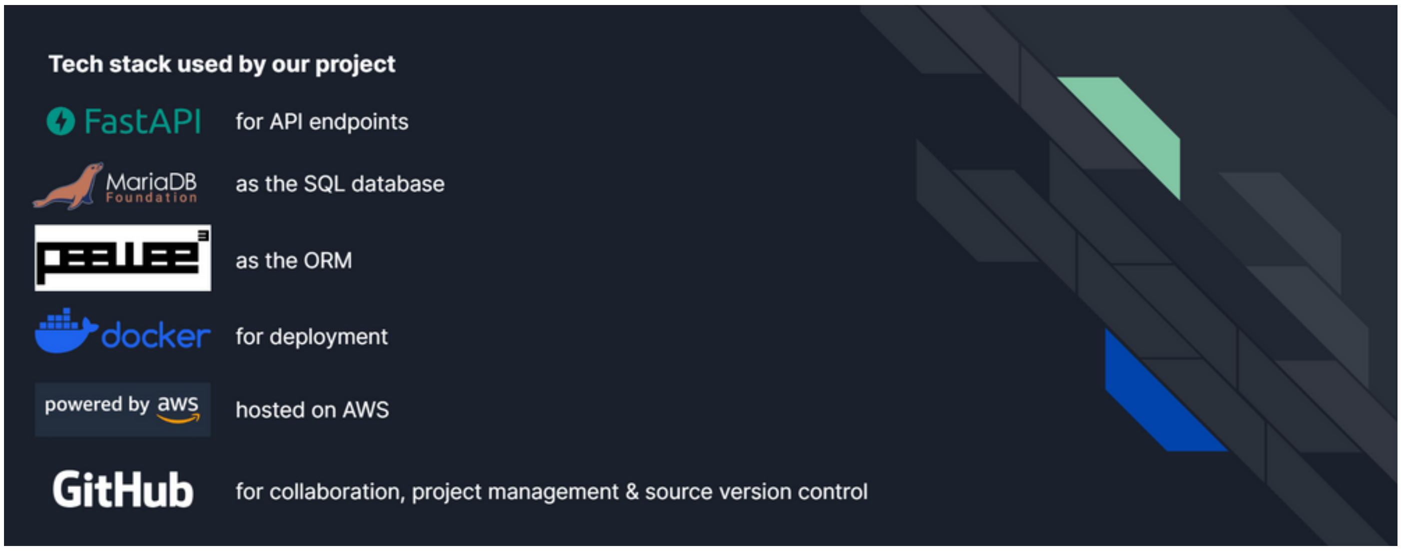
- **Welcome Screen:** Displays a welcome message with an image.
- **Form Steps:** Guides users through a multi-step setup process.
- **Dynamic Form:** Adjusts inputs based on user progress.
- **Data Collection:** Gathers user information, preferences, and goals.
- **Transition Effects:** Smooth transitions between form steps.
- **Profile Submission:** Validates and submits user details.
- **Course Recommendations:** Redirects to recommended courses based on user preferences.
- **Responsive Design:** Adapts to various screen sizes.
- **Font Customization:** Uses custom font for a unique look and feel.

Click here to view milestone 6

Implementation details
(Chapter 8)

Technologies and Tools used:

- VS Code as IDE for both API and frontend development
- Git and Github for collaborating on backend and frontend of the project
- JIRA for Project Management
- Vue 3 and Pinia for development of frontend


Tech stack used by our project

FastAPI	for API endpoints
MariaDB Foundation	as the SQL database
peewee	as the ORM
docker	for deployment
powered by aws	hosted on AWS
GitHub	for collaboration, project management & source version control

🚧 Software Engineering Project (Group 14)

🖧 API details

🌱 Prod Environment	🔗 URL
🏭 Base URL	api.pickmycourse.online
🖥️ Swagger Doc	api.pickmycourse.online/docs

💻 Web app details

🌱 Environment	🔗 URL
📱 Vue frontend	pickmycourse.online

⛏️ Check Projects board

💫 To deploy the API using Docker

> Well it goes without saying, make sure you have Docker installed on your system

🛢️ Run a MariaDB instance

> To quickly get a MariaDB instance running, run the following 👇

```
docker run --name mariadb-dev \
  -v /path/on/your/system:/var/lib/mysql:Z \
  -e MARIADB_DATABASE=some-db-name \
  -e MARIADB_ROOT_PASSWORD=strong-root-password \
  -p 3306:3306 \
  -d mariadb:latest
```

Also need to migrate the peewee DB models to MariaDB, look into the `peewee-migrate` tool

👉 Then, follow the steps to deploy the ⚡ FastAPI server

- `cd server`
- `cp .env.example .env`
- Make sure to edit the `.env` file with proper details
- `sh deploy.sh`

🏃 To run this locally on your machine

> You'll need MariaDB for this API to work.

> Follow the steps mentioned above to quickly spin up a MariaDB instance using docker, either on your local machine or some remote machine. If you don't want to use Docker, follow the MariaDB documentation then.
>
> You'll also need to make a copy of `.env` file with proper details (`.env.example` is given). Follow the below steps next:

- Use Git Bash on Windows (avoid using `cmd` or `powershell`) Better if you use WSL
  - `cd server`
- Create & activate python virtual environment
  - Linux 👉 `python3 -m venv .venv`
  - Windows or `conda` 👉 `python -m venv .venv`
  - Linux 👉 `source .venv/bin/activate`
  - Windows 👉 `source .venv/Scripts/activate`
- Install the requirements
  - `pip install -r requirements.txt`
- Run it using the shell script
  - `sh run.sh`
- To run the client
  - `cd client`
  - `npm install`
  - `npm run dev`

🩺 To test the API endpoints

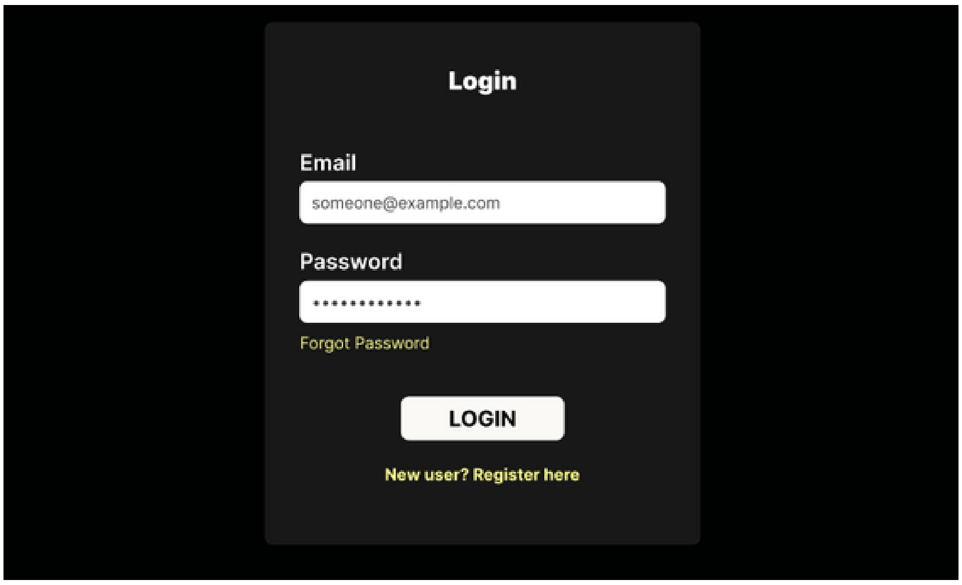
- Create and activate the environment as described above, install requirements
- Run the `pytest.sh` script
  - `sh pytest.sh`

💉 For writing tests

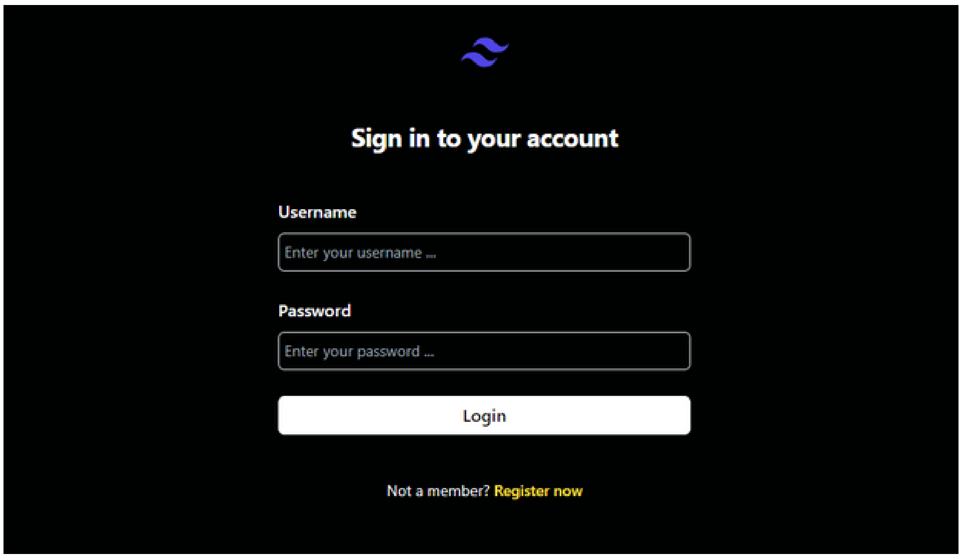
- The directory name has to be `tests`
- The filename must start with `test_`
  - Example: Use name like `test_auth.py` to make tests for `auth.py` endpoints

Wireframe vs Actual
(Chapter 9)

Wireframe



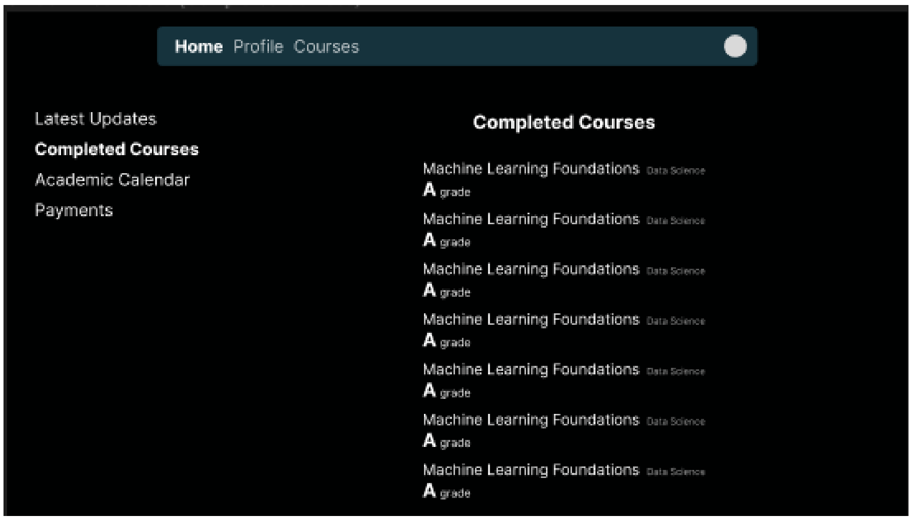
Actual



Wireframe vs Actual

(Chapter 9)

Wireframe

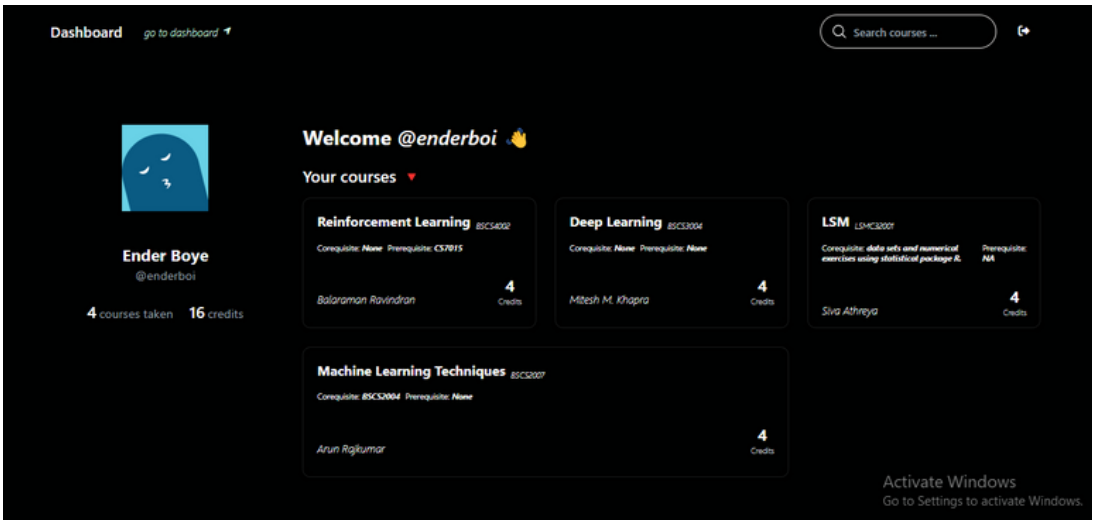


Latest Updates
Completed Courses
Academic Calendar
Payments

Completed Courses

Machine Learning Foundations Data Science
A grade
Machine Learning Foundations Data Science
A grade
Machine Learning Foundations Data Science
A grade
Machine Learning Foundations Data Science
A grade
Machine Learning Foundations Data Science
A grade
Machine Learning Foundations Data Science
A grade
Machine Learning Foundations Data Science
A grade

Actual



Ender Boye
@enderboi
4 courses taken **16** credits

Welcome *@enderboi* 👋

Your courses

Reinforcement Learning *BSCS4002*
Corequisite: ***None*** Prerequisite: ***CS7015***
Balaraman Ravindran **4** Credits

Deep Learning *BSCS3004*
Corequisite: ***None*** Prerequisite: ***None***
Mitesh M. Khapra **4** Credits

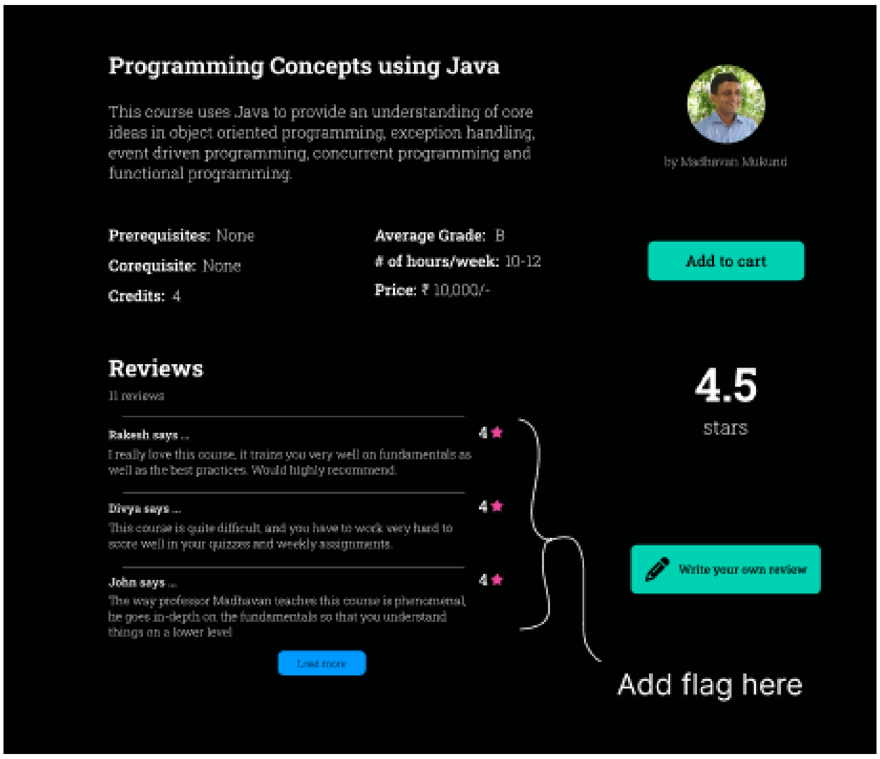
LSM *LSMC3001*
Corequisite: ***data sets and numerical exercises using statistical package R.*** Prerequisite: ***NA***
Siva Athreya **4** Credits

Machine Learning Techniques *BSCS2007*
Corequisite: ***BSCS2004*** Prerequisite: ***None***
Arun Rajkumar **4** Credits

Activate Windows
Go to Settings to activate Windows.

Wireframe vs Actual
(Chapter 9)

Wireframe



Programming Concepts using Java

This course uses Java to provide an understanding of core ideas in object oriented programming, exception handling, event driven programming, concurrent programming and functional programming.

by Madhavan Mukund

Prerequisites: None
Corequisite: None
Credits: 4

Average Grade: B
# of hours/week: 10-12
Price: ₹ 10,000/-

Add to cart

Reviews
11 reviews

Rakesh says ... 4★
I really love this course, it trains you very well on fundamentals as well as the best practices. Would highly recommend.

Divya says ... 4★
This course is quite difficult, and you have to work very hard to score well in your quizzes and weekly assignments.

John says ... 4★
The way professor Madhavan teaches this course is phenomenal, he goes in-depth on the fundamentals so that you understand things on a lower level

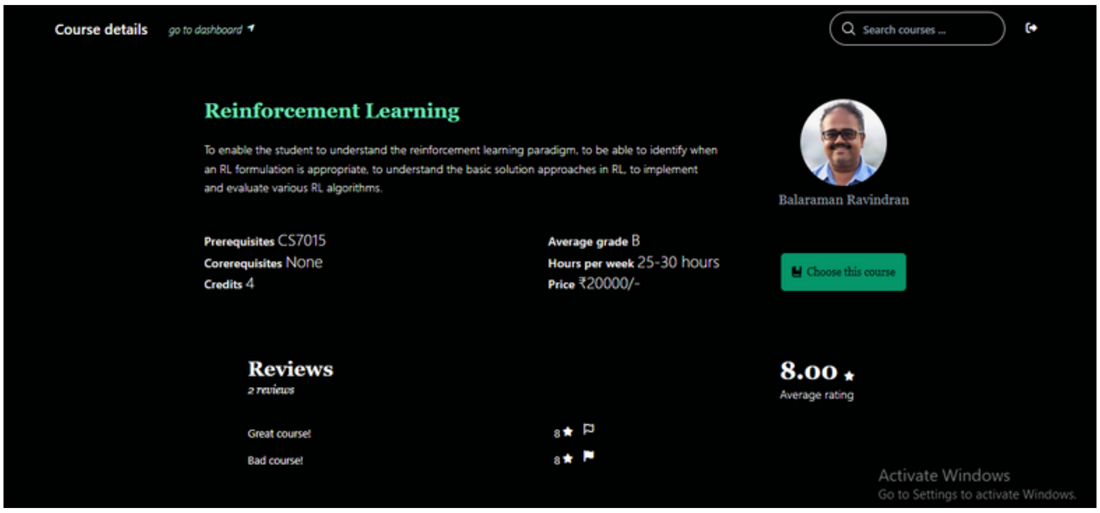
Load more

4.5
stars

Write your own review

Add flag here

Actual



Course details *go to dashboard*

Search courses ...

Reinforcement Learning

To enable the student to understand the reinforcement learning paradigm, to be able to identify when an RL formulation is appropriate, to understand the basic solution approaches in RL, to implement and evaluate various RL algorithms.

Balaraman Ravindran

Prerequisites CS7015
Corerequisites None
Credits 4

Average grade B
Hours per week 25-30 hours
Price ₹20000/-

Choose this course

Reviews
2 reviews

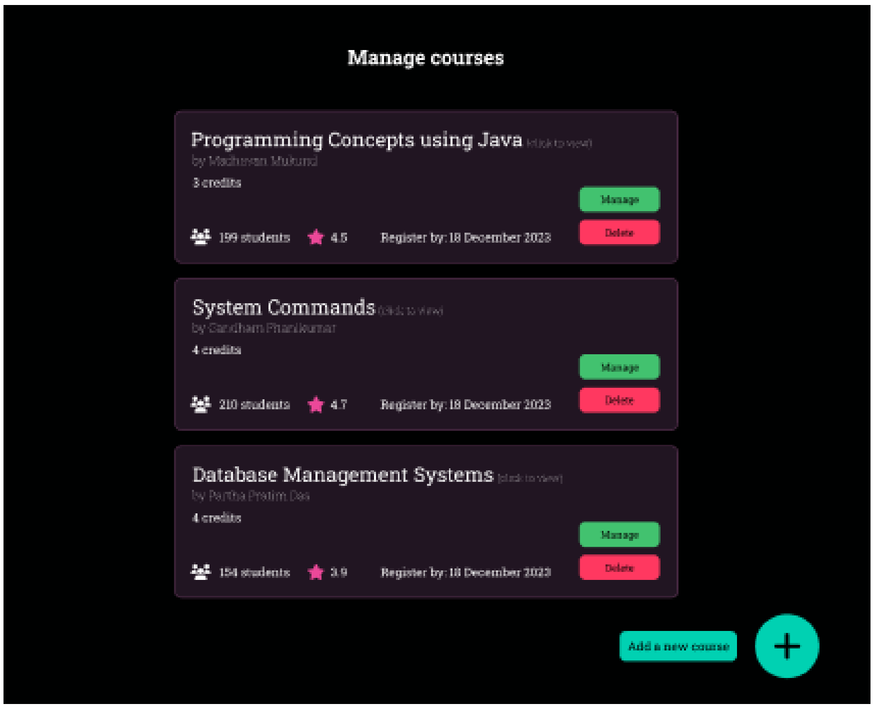
Great course! 8★
Bad course! 8★

8.00★
Average rating

Activate Windows
Go to Settings to activate Windows.

Wireframe vs Actual
(Chapter 9)

Wireframe: Click here to view wireframe



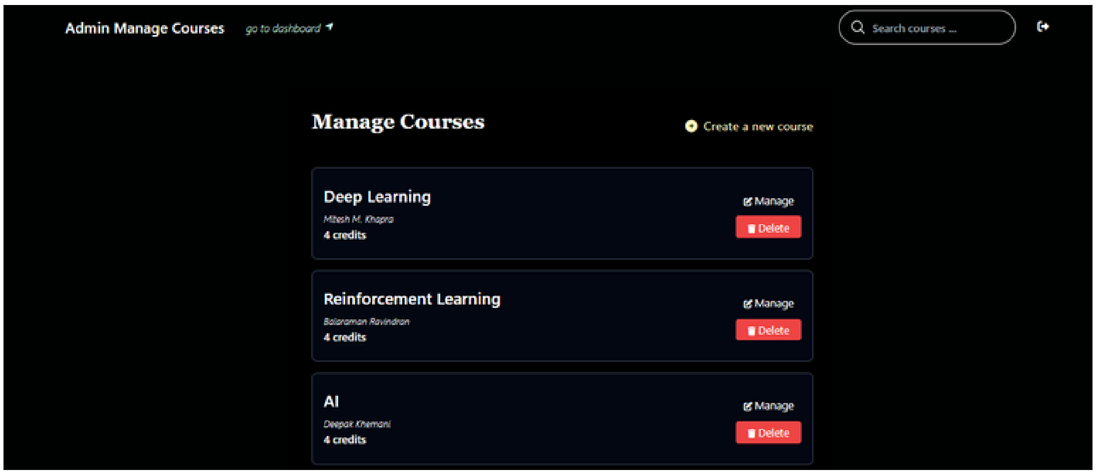
Manage courses

Programming Concepts using Java (click to view)
by Madhavan Mukund
3 credits
199 students 4.5 Register by: 18 December 2023

System Commands (click to view)
by Gandham Phanikumar
4 credits
210 students 4.7 Register by: 18 December 2023

Database Management Systems (click to view)
by Partha Pratim Das
4 credits
154 students 3.9 Register by: 18 December 2023

Actual: Click here to view the website



Admin Manage Courses

Manage Courses

Deep Learning
Mitesh M. Khapra
4 credits

Reinforcement Learning
Balaraman Ravindran
4 credits

AI
Deepak Khemani
4 credits

Group 14 Presentation PPT

Group 14 Presentation Video