
Test Document

for

TRACKit

Version 1.0

Group 5

Group Name: Chill guys

Aditya Gautam	220064	agautam22@iitk.ac.in
Aayush Singh	220024	aayushs22@iitk.ac.in
Dhruv Rai	220365	dhruvrai22@iitk.ac.in
Ved Prakash Vishwakarma	221180	vedprakash22@iitk.ac.in
Sharique Ahmad	221002	asharique22@iitk.ac.in
Dhruv Varshney	220366	vdhruv22@iitk.ac.in
Mayur Agrawal	220641	mayurag22@iitk.ac.in
Akash Verma	220097	akashv22@iitk.ac.in
Rahul Ahirwar	220856	rahula22@iitk.ac.in
Abhijeet Agarwal	210025	abhijeeta21@iitk.ac.in
Aryan Bansal	200198	aryanb20@iitk.ac.in

Course: CS253

Mentor TA: Jeswaanth Gogula

Date: 6 April 2025

CONTENTS.....	..II
REVISIONS.....	..II
1 INTRODUCTION.....	1
2 UNIT TESTING.....	2
3 INTEGRATION TESTING.....	3
4 SYSTEM TESTING.....	4
5 CONCLUSION.....	5
APPENDIX A - GROUP LOG.....	6

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
v1.00	Aditya Gautam Aayush Singh Dhruv Rai Ved Prakash Vishwakarma Sharique Ahmad Dhruv Varshney Mayur Agrawal Akash Verma Rahul Ahirwar Abhijeet Agarwal Aryan Bansal	Completed first version of the testing document	05/04/2025

1 Introduction

Testing Strategy

We used a hybrid testing strategy, as this was closest to the very high level of diversity of API requests in our App. For the functional testing of the Backend APIs and the Frontend, we used manual testing using Postman and visual inspection, while for non-functional testing related to performance, we used dedicated test scripts.

When was the Testing Conducted?

Informal testing by group members was done during every step in the project code development. Our repository, with over 290+ commits, saw the group members testing the code added in that git commit tested to the best of their abilities. We created dozens of issues in GitHub when another user spotted an issue.

Thus testing was an integral part of our development process itself.

In Addition to that, in the week preceding the submission of this document, the group engaged in vigorous and extensive formal testing of the product. Some bugs were found out, and immediately rectified/mark for rectification in the near future.

Who were the Testers?

Every group member contributed extensively to testing.

Coverage Criteria Used

We considered meeting the goals of the SRS, and the stability of the feature being tested, to be the coverage criteria.

The tool used for Testing

We used tools like Postman extensively during development and during testing. Postman is a widely used API testing tool that facilitates the development, testing, and debugging of APIs. It allows users to send requests to API endpoints, inspect responses, and automate testing workflows.

We also used Browser Rendering, and manual UI-based operations to test the various features during Integration and System Testing.

We also wrote dedicated test scripts to stress test various aspects of the app, such as creating 2500 users at once.

2 Unit Testing

Login

Unit Details:

Involves the testing of the user login functionality.

Test Owner:

Aditya Gautam, 220024

Test Date:

2 April 2025

Test Results:

Login Functionality was found to be working as expected.

```

200 OK • 145 ms • 1.5 KB | ⏺ Save Response ⏺
Body Cookies Headers (25) Test Results | ⏺
{} JSON ▾ ▶ Preview ⏺ Visualize | ⏺
1 {
2   "success": true,
3   "message": "Login successful",
4   "token": "eyJhbGciOiJIzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MiwidXNlclR5cGUiOiJmYWNlbnR5iIwiaWF0IjoxNzQzNjE1MDgxLCJleHAiOjE3NDM2MTg2ODF9.
x06u-nTPokx8sRWDPFLlvzIg3Ac7z0HW44r9FUkAS4",
5   "user": {
6     "id": 2,
7     "username": "faculty1",
8     "email": "faculty1@trackit.com",
9     "firstName": "John",
10    "lastName": "Doe",
11    "userType": "faculty",
12    "department": "Electrical Engineering",
13    "position": "Associate Professor"
14  }

```

Structural Coverage:

The unit is working well and fulfils 100% of the requirements of the SRS.

Admin : User

Unit Details:

We have tested the functionality of user management within the admin interface to ensure seamless handling of user data. This includes verifying the ability to create new users, delete existing ones, and retrieve a comprehensive list of all registered users. Additionally, we have tested the critical features of adding and removing students from courses, ensuring proper enrollment and withdrawal processes. Throughout this testing process, our focus has been on maintaining data integrity and consistency in the database.

Test Owner:

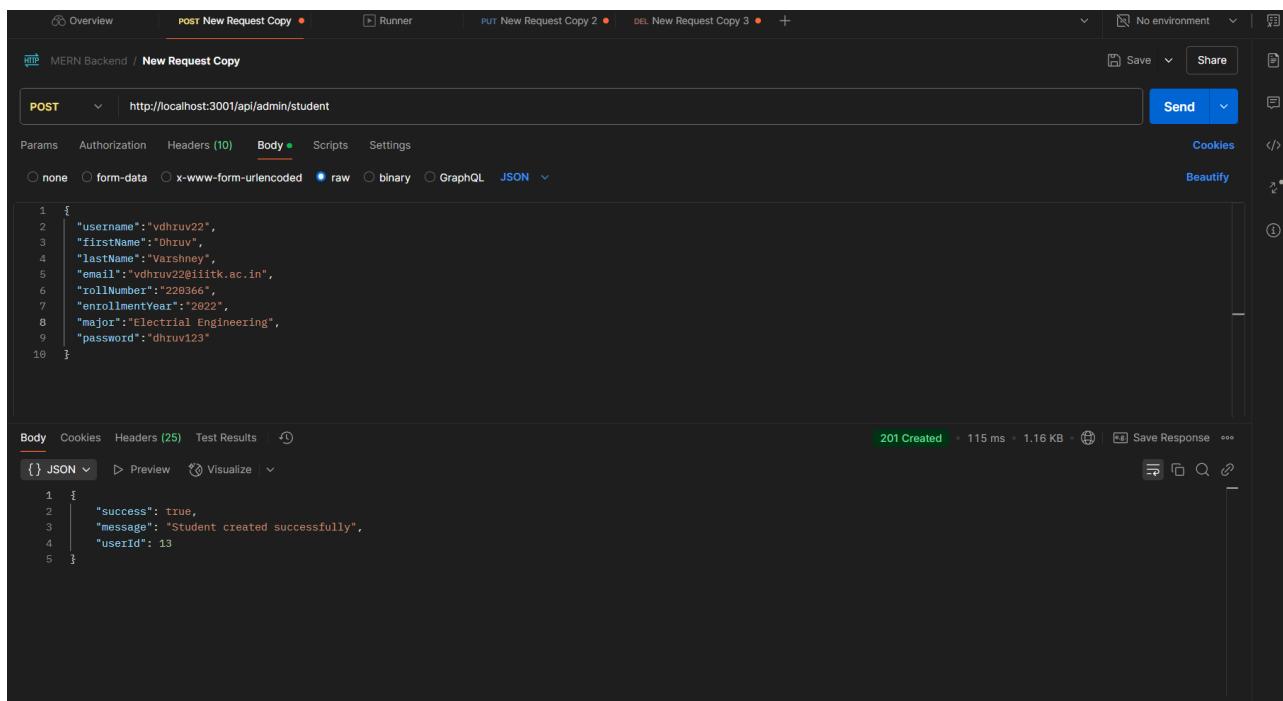
Sharique Ahmad: 221002
Dhruv Varshney: 220366

Test Date:

2 April 2025

Test Results:

API to add student on the platform:



The screenshot shows a Postman collection named "MERN Backend / New Request Copy". A POST request is being made to `http://localhost:3001/api/admin/student`. The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   "username": "vdhruv22",
3   "firstName": "Dhruv",
4   "lastName": "Varshney",
5   "email": "vdhruv22@iitk.ac.in",
6   "rollNumber": "220366",
7   "enrollmentYear": "2022",
8   "major": "Electrical Engineering",
9   "password": "dhruv123"
10 }
```

The response status is 201 Created, with a response body containing:

```
1 {
2   "success": true,
3   "message": "Student created successfully",
4   "userId": 13
5 }
```

API to add students in bulk on the platform:

POST http://localhost:3001/api/admin/bulk-students

Key	Value	Description	Bulk Edit
file	students_data.csv		
Key	Value	Description	

```

1 {
2   "success": true,
3   "message": "Successfully created 5 students",
4   "userIds": [
5     14,
6     15,
7     16,
8     17,
9     18
10   ]
11 }

```

API to add a faculty on the platform:

POST http://localhost:3001/api/admin/faculty

Key	Value	Description	Bulk Edit
username	sshaway		
firstName	Shubham		
lastName	Sahay		
email	sshay@iitk.ac.in		
department	Electrical Engineering		
position	Assistant Professor		
password	shaway123		

```

1 {
2   "username": "sshay",
3   "firstName": "Shubham",
4   "lastName": "Sahay",
5   "email": "sshay@iitk.ac.in",
6   "department": "Electrical Engineering",
7   "position": "Assistant Professor",
8   "password": "shaway123"
9 }

```

API to add faculty in bulk on the platform:

POST /api/admin/bulk-faculty

Body (10)

form-data

Key	Value	Description
file	faculty_data.csv	
Key	Value	Description

201 Created

```
{
  "success": true,
  "message": "Successfully created 5 faculty members",
  "userIds": [
    20,
    21,
    22,
    23,
    24
  ]
}
```

API to get the enrolled Courses of a User.

GET /api/users/9/courses

Body (9)

ID	Code	Name	Description	Credits	Semester	CreatedAt	UpdatedAt
3	PHI452	Philosophy of Mind	Exploration of consciousness, intentionality, mental causation, and the relationship between mind and brain.	3	Fall 2023	2025-04-02T15:58:04.539Z	2025-04-02T16:57:08.702Z
4	EE210	Digital Electronics	Introduction to digital logic, Boolean algebra, combinational and sequential circuits, and digital system design.	4	Fall 2023	2025-04-02T15:58:04.547Z	2025-04-02T15:58:04.547Z
5	EE200	Circuit Theory					

API to add a Faculty in a Course.

POST http://localhost:3001/api/courses/add-faculty

```

1 {
2   "courseId": "1",
3   "userId": "7"
4 }

```

Body Cookies Headers (25) Test Results

```

1 {
2   "success": true,
3   "message": "Faculty added to course successfully"
4 }

```

200 OK 43 ms 1.15 KB

API to get all the users.

GET http://localhost:3001/api/admin/users

```

1 {
2   "success": true,
3   "data": [
4     {
5       "id": 1,
6       "username": "admin",
7       "email": "admin@trackit.com",
8       "firstName": "Admin",
9       "lastName": "User",
10      "userType": "admin",
11      "createdAt": "2025-04-02T15:50:04.496Z",
12      "updatedAt": "2025-04-02T15:50:04.496Z",
13      "student": null,
14      "faculty": null
15    },
16    {
17      "id": 2,
18      "username": "faculty1",
19      "email": "faculty1@trackit.com",
20      "firstName": "John",
21      "lastName": "Doe",
22      "userType": "faculty",
23      "createdAt": "2025-04-02T15:50:04.587Z",
24      "updatedAt": "2025-04-02T15:57:06.779Z",
25      "student": null,
26      "faculty": {
27        "department": "Electrical Engineering"
28      }
29    }
30  ]
31 }

```

200 OK 36 ms 4.82 KB

API for Removing a Student from a Course

The screenshot shows a Postman interface with the following details:

- URL:** `http://localhost:3001/api/courses/remove-student/1/13`
- Method:** `DELETE`
- Headers:** `(9)`
- Body:** `raw` (selected), `JSON` (viewed)
- Response:**
 - Status:** `200 OK`
 - Time:** `49 ms`
 - Size:** `1.15 KB`
 - Content:**

```
1 {  
2   "success": true,  
3   "message": "Student removed from course successfully"  
4 }
```

API to get UserID from Roll Number.

The screenshot shows a Postman interface with the following details:

- URL:** `http://localhost:3001/api/student/rollNumber/CS20B002`
- Method:** `GET`
- Headers:** `(6)`
- Body:** `raw` (selected), `JSON` (viewed)
- Response:**
 - Status:** `200 OK`
 - Time:** `33 ms`
 - Size:** `1.11 KB`
 - Content:**

```
1 {  
2   "success": true,  
3   "userId": 9  
4 }
```

API to add Student in a Course

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** <http://localhost:3001/api/courses/add-student>
- Body (JSON):**

```
1 {
2   "courseId": "1",
3   "userId": "13"
4 }
```
- Response Status:** 200 OK
- Response Time:** 63 ms
- Response Size:** 1.15 KB
- Response Body:**

```
1 {
2   "success": true,
3   "message": "Student added to course successfully"
4 }
```

API to remove a Faculty from a Course.

The screenshot shows a Postman interface with the following details:

- Method:** DELETE
- URL:** <http://localhost:3001/api/courses/remove-faculty/1/2>
- Body (JSON):**

```
1 {}
```
- Response Status:** 200 OK
- Response Time:** 72 ms
- Response Size:** 1.15 KB
- Response Body:**

```
1 {
2   "success": true,
3   "message": "Faculty removed from course successfully"
4 }
```

API to update the user details.

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** <http://localhost:3001/api/admin/user/13>
- Body:** Raw JSON (selected)
- JSON Body:**

```

1 {
2   "id": "13",
3   "username": "asharique22",
4   "firstName": "Sharique3",
5   "lastName": "Doe",
6   "email": "john.doe@example.com",
7   "userType": "student",
8   "rollNumber": "2021001",
9   "major": "Computer Science",
10  "enrollmentYear": "2021",
11  "faculty": null,
12  "student": {
13    "rollNumber": "2021001",
14    "major": "Computer Science",
15    "enrollmentYear": "2021"
16  }
17 }
```
- Response:** 200 OK | 65 ms | 1.14 KB

API to delete a User

The screenshot shows a POSTMAN interface with the following details:

- Method:** DELETE
- URL:** <http://localhost:3001/api/admin/user/13>
- Body:** Raw JSON (selected)
- JSON Body:**

```

1 {
2   "success": true,
3   "message": "User deleted successfully"
4 }
```
- Response:** 200 OK | 26 ms | 1.14 KB

Structural Coverage:

The unit is working well and fulfils 100% of the requirements of the SRS

Admin: Courses

Unit Details:

This includes testing endpoints for creating, retrieving, updating, and deleting courses. The tests ensured proper validation of input data, appropriate error handling, and correct responses for both successful and failure scenarios.

Test Owner:

Sharique Ahmad: 221002

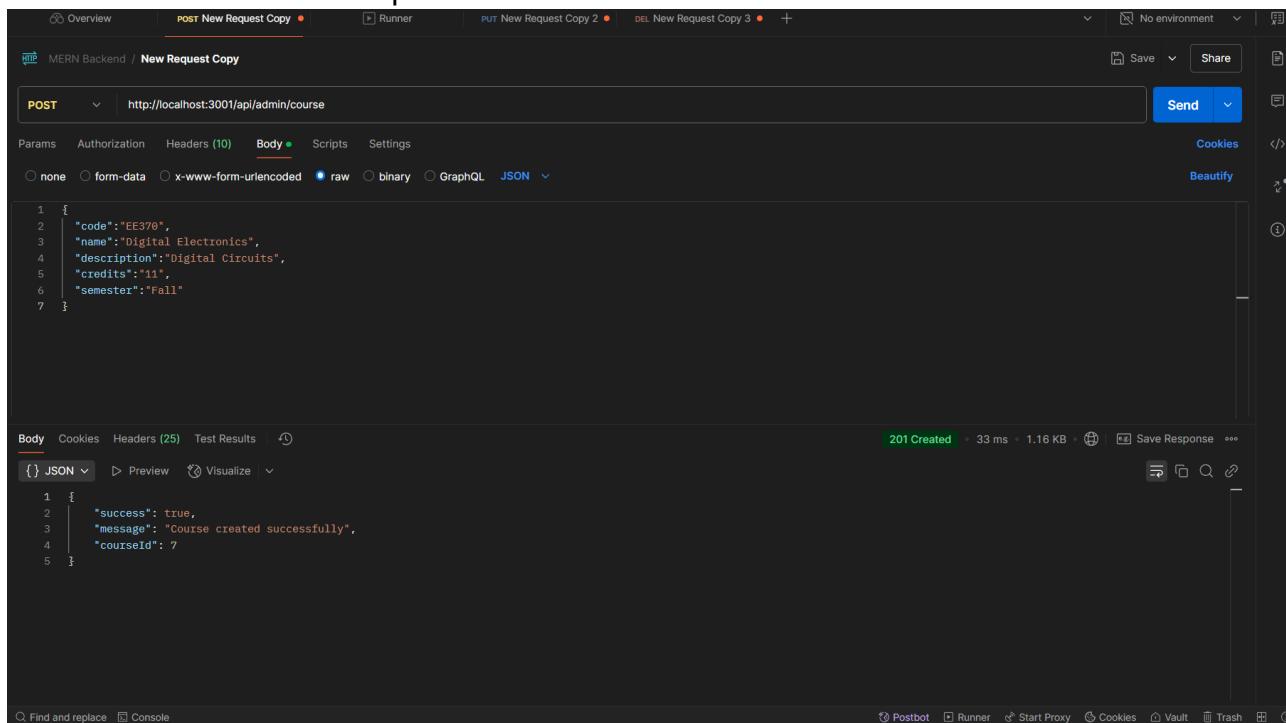
Dhruv Varshney: 220366

Test Date:

3 April 2025

Test Results:

API to add a course on the platform:



The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:3001/api/admin/course`. The request body is a JSON object representing a course:

```

1  {
2   "code": "EE370",
3   "name": "Digital Electronics",
4   "description": "Digital Circuits",
5   "credits": "11",
6   "semester": "Fall"
7 }

```

The response status is `201 Created`, and the response body is:

```

1  {
2   "success": true,
3   "message": "Course created successfully",
4   "courseId": 7
5 }

```

API to add courses in bulk on the platform:

POST <http://localhost:3001/api/admin/bulk-courses>

Body

Key	Value	Description
file	course_details.csv	

Body **Cookies** **Headers (25)** **Test Results**

```
{
  "success": true,
  "message": "Successfully created 2 courses",
  "courses": [
    {
      "id": 8,
      "code": "EE320"
    },
    {
      "id": 9,
      "code": "ESC201"
    }
  ]
}
```

201 Created | 141 ms | 1.21 KB | Save Response

API to delete a Course.

DELETE <http://localhost:3001/api/courses/2>

Body

```
{
  "success": true,
  "message": "Course deleted successfully"
}
```

200 OK | 84 ms | 1.14 KB

API to update the Course details.

PUT http://localhost:3001/api/courses/1

Body

```

1 {
2   "id": "1",
3   "code": "EE320",
4   "name": "Introduction to Communication Systems",
5   "description": "Introduction to signals and systems, discrete-time signals, linear time-invariant systems, z-transforms, Fourier analysis, and other aspects of communication systems.",
6   "credits": 9,
7   "semester": "Fall 2023",
8   "Students": [],
9   "Faculty": []
10 }

```

Response

```

1 {
2   "success": true,
3   "message": "Course updated successfully"
4 }

```

API to get course details.

GET http://localhost:3001/api/courses/1

Body

```

1 {
2   "success": true,
3   "data": {
4     "id": 1,
5     "code": "EE321",
6     "name": "Communication Systems",
7     "description": "Introduction to signals and systems, discrete-time signals, linear time-invariant systems, z-transforms, Fourier analysis, and other aspects of communication systems.",
8     "credits": 4,
9     "semester": "Fall 2023",
10    "createdAt": "2025-04-02T15:50:04.527Z",
11    "updatedAt": "2025-04-02T17:06:38.948Z",
12    "faculty": [
13      {
14        "userId": 2,
15        "department": "Electrical Engineering",
16        "position": "Associate Professor",
17        "createdAt": "2025-04-02T15:50:04.595Z",
18        "updatedAt": "2025-04-02T15:50:04.595Z",
19        "user": {
20          "id": 2,
21          "username": "faculty1",
22          "email": "faculty1@trackit.com",
23          "firstName": "John",
24          "lastName": "Doe"
25        }
26      }
27    ]
28  }
29 }

```

API to get all courses.

```

1 {
2   "success": true,
3   "data": [
4     {
5       "id": 1,
6       "code": "EE321",
7       "name": "Communication Systems",
8       "description": "Introduction to signals and systems, discrete-time signals, linear time-invariant systems, z-transforms, Fourier analysis, and other aspects of communication systems.",
9       "credits": 4,
10      "semester": "Fall 2023",
11      "createdAt": "2025-04-02T15:50:04.527Z",
12      "updatedAt": "2025-04-02T17:06:38.948Z"
13    },
14    {
15      "id": 2,
16      "code": "CS253",
17      "name": "Software Development",
18      "description": "Software development methodologies, design patterns, testing strategies, version control systems, and project management.",
19      "credits": 3,
20      "semester": "Fall 2023",
21      "createdAt": "2025-04-02T15:50:04.533Z",
22      "updatedAt": "2025-04-02T15:50:04.533Z"
23    },
24    {
25      "id": 3,
26      "code": "PHI452",
27    }
28  ]
29 }
30 
```

Events

Unit Details:

Involves the testing of the Events Database, APIs and Static Frontend.

Test Owner:

Aayush Singh, 220024

Test Date:

2 April 2025

Test Results:

First, Testing the APIs required in Events.

API to fetch all the courses a user is enrolled in:

It will only work when we provide token(authorization that a user is signed in)

The screenshot shows the Postman interface with a history of API calls on the left and a detailed view of a specific request on the right.

History:

- GET http://172.27.16.252:3001/api/users/profile
- POST http://172.27.16.252:3001/api/auth/login
- March 23
 - GET http://14.139.38.159/api/users/profile
- March 16
 - GET http://localhost:3000/api/users/profile
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/profile
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/id/courses
 - POST http://localhost:3000/api/auth/login
 - GET http://localhost:3000/api/users/profile
 - GET http://localhost:3000/api/users/courses
 - POST http://localhost:3001/api/auth/login
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/courses

Request Details: GET http://localhost:3001/api/users/8/courses ...

Headers:

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1NilsInR5cCl6lkpXVCJ9eyJpZC16OCwidXNlclR5c...
Key	Value

Body:

```

1   "success": true,
2   "data": [
3     {
4       "id": 1,
5       "code": "EE321",
6       "name": "Communication Systems",
7       "description": "Introduction to signals and systems, discrete-time signals, linear time-invariant systems, z-transforms, Fourier analysis, and other aspects of communication systems.",
8       "credits": 4,
9       "semester": "Fall 2023",
10      "createdAt": "2025-04-02T15:30:31.080Z",
11      "updatedAt": "2025-04-02T15:30:31.080Z"
12    },
13    {
14      "id": 2,
15      "code": "CS253",
16    }
  
```

API to get events for a particular course:
This will also work with authorization

The screenshot shows the Postman interface with a history of API calls on the left and a detailed view of a specific request on the right.

History:

- GET http://172.27.16.252:3001/api/users/profile
- POST http://172.27.16.252:3001/api/auth/login
- March 23
 - GET http://14.139.38.159/api/users/profile
- March 16
 - GET http://localhost:3000/api/users/profile
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/profile
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/id/courses
 - POST http://localhost:3000/api/auth/login
 - GET http://localhost:3000/api/users/profile
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/courses
 - GET http://localhost:3000/api/users/courses

Request Details: GET http://localhost:3001/api/events/course/1

Headers:

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1NilsInR5cCl6lkpXVCJ9eyJpZC16OCwidXNlclR5c...
Key	Value

Body:

```

1   "success": true,
2   "data": [
3     {
4       "id": "d06651a4-8eee-4374-88e7-4f8c311ef093",
5       "title": "Midterm Exam",
6       "description": "Covers chapters 1-5",
7       "start": "2023-11-10T08:00:00.000Z",
8       "end": "2023-11-10T10:00:00.000Z",
9       "createdAt": "2025-04-02T17:47:13.916Z",
10      "updatedAt": "2025-04-02T17:47:13.916Z",
11      "courseId": 1,
12      "createdBy": 8
13    },
14    {
15      "id": "f030daeb-79e2-4e35-b7eb-0af03e30372d",
16      "title": "Endterm Exam",
17      "description": "Carry Calculator with you",
18    }
  
```

API to add events for a course by a faculty:

It has an authorization check for faculty of that particular course, also we would provide the data in the body of the API call in JSON format.

The screenshot shows the Postman interface with a successful POST request to `http://localhost:3001/api/events/course/1`. The request body contains the following JSON:

```

1
2   "title": "Midterm Exam",
3   "description": "Covers chapters 1-5",
4   "start": "2023-11-10T08:00:00Z",
5   "end": "2023-11-10T10:00:00Z"
6

```

The response status is `201 Created`, and the response body is:

```

1
2   "success": true,
3   "message": "Event created successfully",
4   "data": {
5     "id": "d06651a4-8eee-4374-88e7-4f8c311ef093",
6     "title": "Midterm Exam",
7     "description": "Covers chapters 1-5",
8     "start": "2023-11-10T08:00:00.000Z",
9     "end": "2023-11-10T10:00:00.000Z",
10    "courseId": "1",
11    "createdBy": 8,
12    "updatedAt": "2025-04-02T17:47:13.910Z",
13    "createdAt": "2025-04-02T17:47:13.910Z"
14
15

```

API for updating an event: It has an authorization check for faculty of that particular course. Also we would provide the data in the body of the API call in JSON format.

The screenshot shows the Postman interface with a successful PUT request to `http://localhost:3001/api/events/d06651a4-8eee-4374-88e7-4f8c311ef093`. The request body contains the following JSON:

```

1
2   "title": "Updated Midterm Exam",
3   "description": "Now covers chapters 1-6"
4

```

The response status is `200 OK`, and the response body is:

```

1
2   "success": true,
3   "message": "Event updated successfully",
4   "data": {
5     "id": "d06651a4-8eee-4374-88e7-4f8c311ef093",
6     "title": "Updated Midterm Exam",
7     "description": "Now covers chapters 1-6",
8     "start": "2023-11-10T08:00:00.000Z",
9     "end": "2023-11-10T10:00:00.000Z",
10    "courseId": "1",
11    "createdBy": 8,
12    "updatedAt": "2025-04-02T17:47:13.910Z",
13    "createdAt": "2025-04-02T17:47:13.910Z"
14
15

```

API for deleting an event: It has an authorization check for faculty of that particular course.

The screenshot shows the Postman interface with a DELETE request to `http://localhost:3001/api/events/d06651a4-8eee-4374-88e7-4f8c311ef093`. The Headers tab includes an `Authorization` header with a Bearer token. The response body is:

```

1
2 "success": true,
3 "message": "Event deleted successfully"
4

```

Deleted Event doesn't exist now: Works as intended.

The screenshot shows the Postman interface with a GET request to `http://localhost:3001/api/events/course/1`. The Headers tab includes an `Authorization` header with a Bearer token. The response body is:

```

1
2 "success": true,
3 "data": [
4   {}
5 ]
6

```

API for getting all events of a particular user: This API call is useful in the main dashboard calendar.

The screenshot shows the Postman interface with a successful API call to `http://localhost:3001/api/events/user/1`. The response body is a JSON object:

```

1
2   "success": true,
3   "data": [
4     {
5       "id": "f030daeb-79e2-4e35-b7eb-0af03e30372d",
6       "title": "endterm Exam",
7       "description": "Carry Calculator with you",
8       "start": "2025-11-10T08:00:00.000Z",
9       "end": "2025-11-10T10:00:00.000Z",
10      "createdAt": "2025-04-02T18:00:49.425Z",
11      "updatedAt": "2025-04-02T18:00:49.425Z",
12      "courseId": 1,
13      "createdBy": 8
14    }
15  ]
16

```

All the APIs are giving desired output. Now checking the frontend UI of events.
Course home:

The screenshot shows the CS253 course home page. The sidebar includes links for Course Home, Lectures, Announcements, Calendar, Result, Forum, and Logout. The main content area displays a calendar for April 2025 with the following events:

Day	Sun	Mon	Tue	Wed	Thu	Fri	Sat
	30	31	01	02	03 Quiz 1	04	05
	06	07	08	09	10	11	12
	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28	29	30	01	02	03

Your Attendance
95%
You have attended:
19/20 classes

Main Dashboard:

TRACKit

- [My Courses](#)
- [Performance](#)
- [Profile](#)
- [Contact Us](#)

[Logout](#)

The dashboard features a sidebar with 'My Courses' (highlighted), 'Performance', 'Profile', 'Contact Us', and a 'Logout' button. Below the sidebar is a monthly calendar for April 2025. The calendar includes course tiles for CS253, EE210, EE200, ECO111, PHI452, and EE321, along with assignment submission and project submission dates.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	01	02 Quiz EE221	03 Quiz 1 CS252	04	05
06	07	08	09	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
Assignment Submission CS252				PROJECT SUBMISS CS252		
27	28	29	30	01	02	03

TRACKit

- [My Courses](#)
- [Performance](#)
- [Profile](#)
- [Contact Us](#)

[Logout](#)

The dashboard features a sidebar with 'My Courses' (highlighted), 'Performance', 'Profile', 'Contact Us', and a 'Logout' button. Below the sidebar is a weekly timeline for the period from March 30 to April 4, 2025. The timeline shows various time slots from 1:00 PM to 8:00 PM, with specific events like '8:18 PM - 8:21 PM' highlighted in blue.

30 Sun	31 Mon	01 Tue	02 Wed	03 Thu	04 Fri	05 Sat
1:00 PM						
2:00 PM						
3:00 PM						
4:00 PM						
5:00 PM						
6:00 PM						
7:00 PM						
8:00 PM			8:18 PM - 8:21 PM			

The screenshot shows a student's calendar page for the EE321 course. The left sidebar includes links for My Courses, Performance, Profile, Contact Us, and Logout. The main area displays a daily agenda for Wednesday, April 02, 2025, from 12:00 PM to 8:00 PM. A blue bar at the bottom indicates a scheduled event from 8:18 PM to 8:21 PM. The top navigation bar lists other courses: Software Devl..., Digital Electr..., Circuit Theory, Principles of ..., and Philosophy of ...

Student Side Calendar Page:

The screenshot shows a faculty member's calendar page for the EE321 course. The left sidebar includes links for Course Home, Lectures, Announcements, Calendar (which is highlighted), Result, Forum, and Logout. The main area displays a monthly calendar for April 2025. A dark blue box highlights the date April 03, labeled 'quiz'. The top navigation bar lists other courses: Software Devl..., Digital Electr..., Circuit Theory, Principles of ..., and Philosophy of ...

Faculty Side Calendar Page:

The screenshot shows the CS253 course calendar interface. On the left, there's a sidebar with links: Course Home, Lectures, Announcements, Calendar (which is highlighted), Result, and Forum. On the right, the main area displays a monthly calendar for Fall 2023. A modal window titled "Add New Event" is open in the center. It contains fields for "Title", "Start Date" (set to "yyyy-mm-ddT--::--"), and "End Date" (also set to "yyyy-mm-ddT--::--"). There's also a "Description" text area and two buttons at the bottom: "Add Event" (green) and "Cancel" (red). The calendar background shows dates from 30 to 03, with a dark blue box highlighting the 01st. A tooltip "PROJECT SUBMISS..." is visible near the 01st.

This screenshot is similar to the one above, but the "Add New Event" dialog box is larger and more detailed. It includes a "Title" field and a "Start Date" field with a placeholder "yyyy-mm-ddT--::--". Below these is a date picker for April 2025. The calendar shows days from 30 to 03. The 03rd is highlighted with a blue border. The date picker interface includes a month dropdown ("April, 2025"), navigation arrows, and a grid of days. Buttons for "Clear" and "Today" are at the bottom of the date picker. The rest of the interface is identical to the first screenshot, including the sidebar and the "PROJECT SUBMISS..." tooltip.

Structural Coverage:

The unit is working well and fulfills 100% of the requirements of the SRS. We tested backend and frontend separately in this unit testing, and would cover its integration in integration testing part.

Announcement

Unit Details:

Involves the testing of the Announcements Database and APIs. The announcement database is unique to each course, where an announcement object has a heading along with a body, along with the corresponding frontend. Only Faculties associated with that course can create announcements in that course, but they can be viewed by all students and other Faculty members.

Test Owner:

Aditya Gautam, 220064

Test Date:

2 April 2025

Test Results:

Getting all Announcements: Test Successful

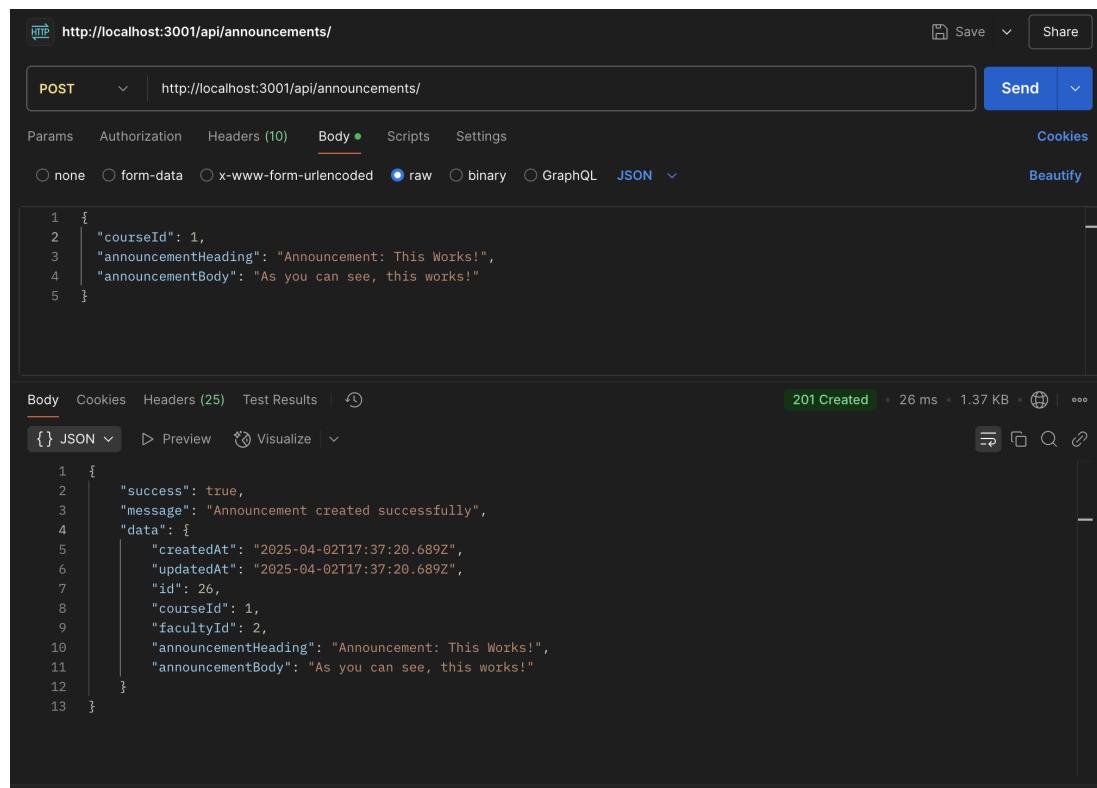
The screenshot shows a Postman API client interface. The URL is `http://localhost:3001/api/announcements/course/1`. The method is `GET`. The Headers tab shows two checked items: `Content-Type: application/json` and `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...`. The Body tab shows a JSON response with the following content:

```

1  {
2   "success": true,
3   "data": [
4     {
5       "id": 4,
6       "courseId": 1,
7       "facultyId": 2,
8       "announcementHeading": "Announcement 4 for EE321",
9       "announcementBody": "This is the 4th announcement for the course EE321. The tutorial session has been rescheduled due to a faculty meeting.",
10      "createdAt": "2025-04-02T14:50:34.811Z",
11      "updatedAt": "2025-04-02T14:50:34.811Z",
12      "faculty": {
13        "userId": 2,
14        "department": "Electrical Engineering",
15        "position": "Associate Professor",
16      }
17    }
18  ]
19}

```

The response status is `200 OK`, with a response time of `83 ms` and a size of `3.06 KB`.

Creating new announcement (Faculty only): Test Successful


The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:3001/api/announcements/
- Body:** JSON (selected)
- Body Content:**

```

1  {
2    "courseId": 1,
3    "announcementHeading": "Announcement: This Works!",
4    "announcementBody": "As you can see, this works!"
5  }

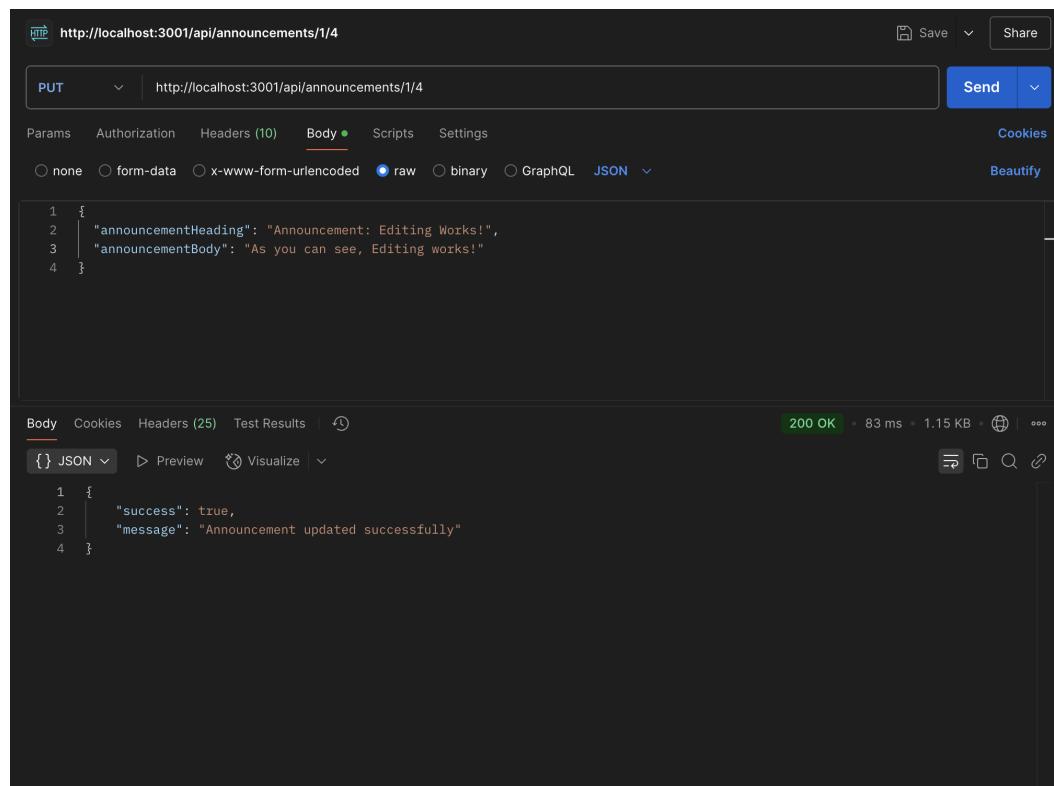
```

- Response:** 201 Created (26 ms, 1.37 KB)
- Response Body:**

```

1  {
2    "success": true,
3    "message": "Announcement created successfully",
4    "data": {
5      "createdAt": "2025-04-02T17:37:20.689Z",
6      "updatedAt": "2025-04-02T17:37:20.689Z",
7      "id": 26,
8      "courseId": 1,
9      "facultyId": 2,
10     "announcementHeading": "Announcement: This Works!",
11     "announcementBody": "As you can see, this works!"
12   }
13 }

```


Editing Existing Announcement (Faculty only): Test Successful


The screenshot shows a Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:3001/api/announcements/1/4
- Body:** JSON (selected)
- Body Content:**

```

1  {
2    "announcementHeading": "Announcement: Editing Works!",
3    "announcementBody": "As you can see, Editing works!"
4  }

```

- Response:** 200 OK (83 ms, 1.15 KB)
- Response Body:**

```

1  {
2    "success": true,
3    "message": "Announcement updated successfully"
4  }

```

Deleting Existing Announcement (Faculty only): Test Successful

The screenshot shows a Postman interface with the following details:

- Request URL:** `http://localhost:3001/api/announcements/1/`
- Method:** `DELETE`
- Headers:**
 - `Content-Type: application/json`
 - `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...`
- Response:**
 - Status: `200 OK`
 - Time: `73 ms`
 - Size: `1.15 KB`
 - Body (JSON):

```

1  {
2    "success": true,
3    "message": "Announcement deleted successfully"
4 }
```

Testing the Static Frontend: Test Successful, all components rendering flawlessly

The screenshot shows the EE321 course frontend with the following components:

- Left Sidebar:**
 - EE321** (Course Home)
 - Lectures
 - Announcements** (highlighted)
 - Calendar
 - Result
 - Forum
- Top Bar:**
 - EE321 • 4 Credits • Fall 2023
 - + Add Announcement**
 - User icon
- ANNOUNCEMENTS Section:**
 - Announcement 3 for EE321**: This is the 3rd announcement for the course EE321. The upcoming lecture will cover important topics for the mid-term exam.
Posted by: John Doe (faculty1)
Created: 4/6/2025, 1:02:59 AM
 - Announcement 4 for EE321**
 - Announcement 2 for EE321**
 - Announcement 1 for EE321**
- Bottom Navigation:** Logout

Structural Coverage:

The unit is working well and fulfils 100% of the requirements of the SRS.

Course Description Entries

Unit Details:

Involves the testing of the Course Description Entry Database and APIs. The Course Description Entry database is unique to each course, where an Course Description Entry object has a heading along with a body, along with the corresponding frontend. Only Faculties associated with that course can create Course Description Entries in that course, but they can be viewed by all students and other Faculty members.

Test Owner:

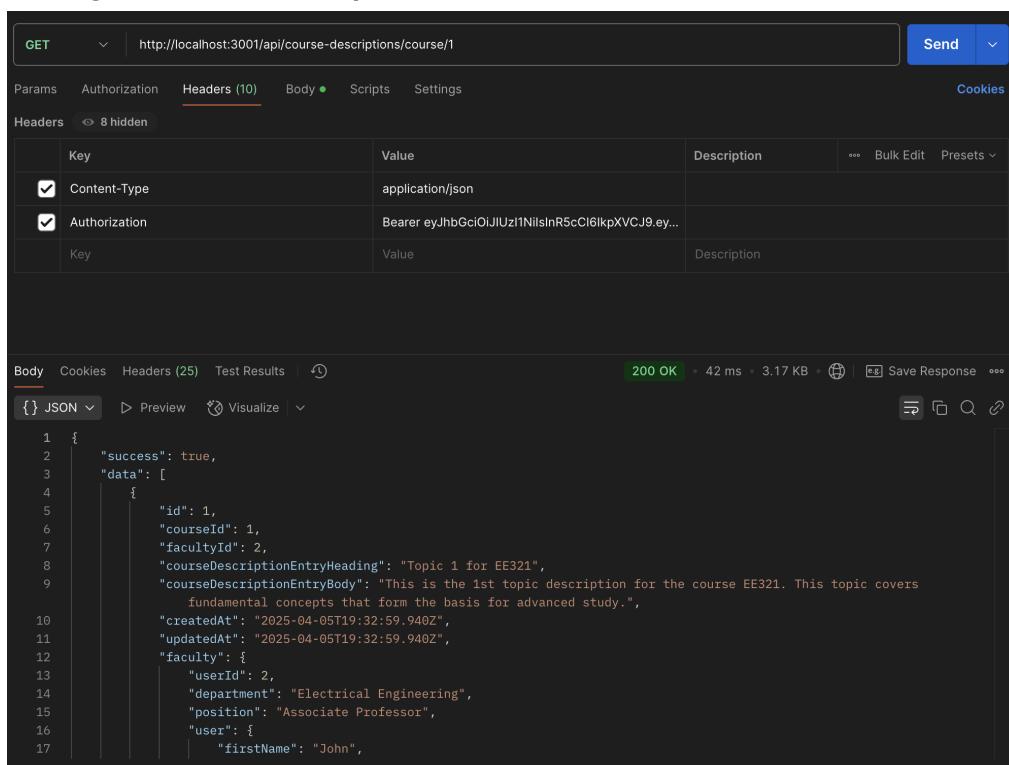
Aditya Gautam, 220064

Test Date:

2 April 2025

Test Results:

Getting all Course Description Entries: Test Successful



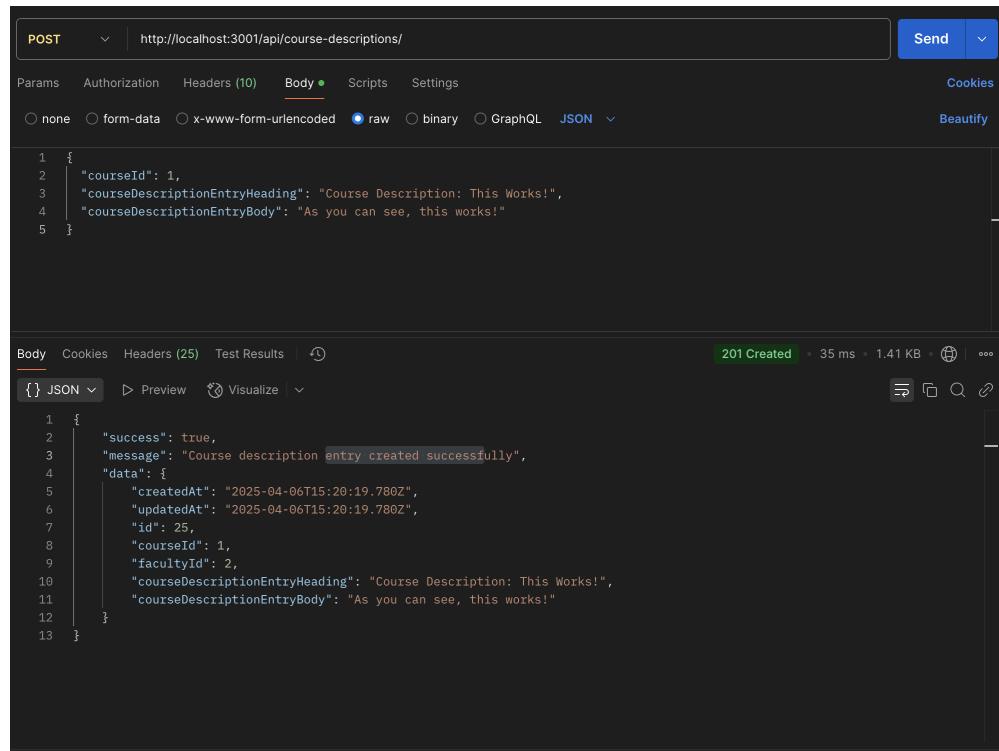
The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** http://localhost:3001/api/course-descriptions/course/1
- Headers:**
 - Content-Type: application/json
 - Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...
- Body:** (Empty JSON object)
- Response Status:** 200 OK
- Response Time:** 42 ms
- Response Size:** 3.17 KB
- Response Content:**

```

1 {
2   "success": true,
3   "data": [
4     {
5       "id": 1,
6       "courseId": 1,
7       "facultyId": 2,
8       "courseDescriptionEntryHeading": "Topic 1 for EE321",
9       "courseDescriptionEntryBody": "This is the 1st topic description for the course EE321. This topic covers fundamental concepts that form the basis for advanced study.",
10      "createdAt": "2025-04-05T19:32:59.940Z",
11      "updatedAt": "2025-04-05T19:32:59.940Z",
12      "faculty": {
13        "userId": 2,
14        "department": "Electrical Engineering",
15        "position": "Associate Professor",
16        "user": {
17          "firstName": "John",

```

Creating new Course Description Entry (Faculty only): Test Successful


The screenshot shows a Postman request to `http://localhost:3001/api/course-descriptions/` using the `POST` method. The `Body` tab is selected, showing the raw JSON payload:

```

1 {
2   "courseId": 1,
3   "courseDescriptionEntryHeading": "Course Description: This Works!",
4   "courseDescriptionEntryBody": "As you can see, this works!"
5 }

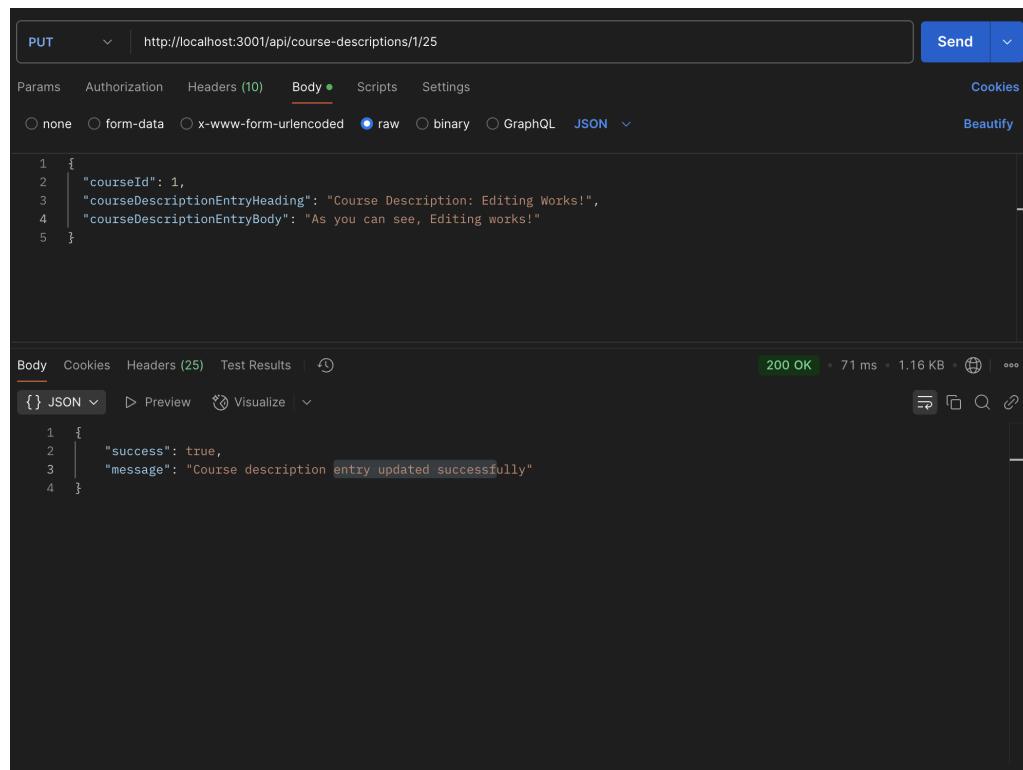
```

The response status is `201 Created`, with a response time of `35 ms` and a size of `1.41 KB`. The response body is:

```

1 {
2   "success": true,
3   "message": "Course description entry created successfully",
4   "data": {
5     "createdAt": "2025-04-06T15:20:19.780Z",
6     "updatedAt": "2025-04-06T15:20:19.780Z",
7     "id": 25,
8     "courseId": 1,
9     "facultyId": 2,
10    "courseDescriptionEntryHeading": "Course Description: This Works!",
11    "courseDescriptionEntryBody": "As you can see, this works!"
12  }
13 }

```

Editing Existing Course Description Entry (Faculty only): Test Successful


The screenshot shows a Postman request to `http://localhost:3001/api/course-descriptions/1/25` using the `PUT` method. The `Body` tab is selected, showing the raw JSON payload:

```

1 {
2   "courseId": 1,
3   "courseDescriptionEntryHeading": "Course Description: Editing Works!",
4   "courseDescriptionEntryBody": "As you can see, Editing works!"
5 }

```

The response status is `200 OK`, with a response time of `71 ms` and a size of `1.16 KB`. The response body is:

```

1 {
2   "success": true,
3   "message": "Course description entry updated successfully"
4 }

```

Deleting Existing Course Description Entry (Faculty only): Test Successful

DELETE <http://localhost:3001/api/course-descriptions/1/25> Send

Params Authorization Headers (10) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautiful

```
1 {  
2   "courseId": 1,  
3   "courseDescriptionEntryHeading": "Course Description: Editing Works!",  
4   "courseDescriptionEntryBody": "As you can see, Editing works!"  
5 }
```

Body Cookies Headers (25) Test Results ⚡

200 OK 74 ms 1.16 KB

Body Cookies Headers (25) Test Results ⚡

{ } JSON Preview Visualize

200 OK 74 ms 1.16 KB

```
1 {  
2   "success": true,  
3   "message": "Course description entry deleted successfully"  
4 }
```

Testing the Static Frontend: Test Successful, all components rendering flawlessly

EE321

COMMUNICATION SYSTEMS

EE321 • 4 Credits • Fall 2023 • faculty

	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28	29	30	01	02	03

Course Home

Lectures

Announcements

Calendar

Result

Forum

+ Add Section

Topic 1 for EE321

This is the 1st topic description for the course EE321. This topic covers fundamental concepts that form the basis for advanced study.

Topic 2 for EE321

Topic 3 for EE321

Topic 4 for EE321

Logout

Structural Coverage:

The unit is working well and fulfils 100% of the requirements of the SRS.

Forum

Unit Details:

Involves the testing of the Forum Database and APIs.

Test Owner:

Rahul Ahirwar, 220856

Test Date:

2 April 2025

Test Results: Testing the APIs required in Forum.

- API to get forum posts for a course.

The screenshot shows the Postman application interface. In the top navigation bar, there are links for Home, Workspaces, and API Network. The search bar contains the placeholder "Search Postman". On the right side of the header, there are buttons for Invite, Settings, and Upgrade. Below the header, the left sidebar has sections for Collections, Environments, Flows, and History. The main workspace shows a collection named "http" with a single item: "GET http://localhost:3001/api/forum/course/1". The request details show a GET method, URL "http://localhost:3001/api/forum/course/1", and various tabs for Params, Authorization, Headers (10), Body, Scripts, and Settings. The Headers tab is currently selected. The Body tab shows a JSON response with the following content:

```
1  {
2      "success": true,
3      "data": [
4          {
5              "id": "e80a86e1-bc75-4bbf-a3c7-66279040be6f",
6              "query": "hi",
7              "courseId": 1,
8              "userId": 9,
9              "createdAt": "2025-04-02T17:38:59.875Z",
10             "updatedAt": "2025-04-02T17:38:59.875Z",
11             "deletedAt": null,
12             "user": {
13                 "id": 9,
14                 "username": "student2",
15                 "firstName": "Bob",
16                 "lastName": "Williams",
17                 "userType": "student"
18             },
19             "replies": [

```

The response status is 200 OK, with a duration of 101 ms and a size of 1.69 KB. There are buttons for Save, Share, and Send.

- API to create a new post.

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3001/api/forum/course/1`. The response status is `201 Created` with a response time of 53 ms and a size of 1.46 KB. The JSON response body is:

```

1 {
2   "success": true,
3   "message": "Post created successfully",
4   "data": {
5     "id": "64112ee2-d281-4fa6-a3a7-4b1ea24ae17f",
6     "query": "This is a test forum post for unit testing",
7     "courseId": 1,
8     "userId": 10,
9     "createdAt": "2025-04-02T17:55:38.048Z",
10    "updatedAt": "2025-04-02T17:55:38.048Z",
11    "deletedAt": null,
12    "user": {
13      "id": 10,
14      "name": "John Doe"
15    }
16  }
17}

```

- API to add a reply to a post

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3001/api/forum/post/64112ee2-d281-4fa6-a3a7-4b1ea24ae17f/reply`. The response status is `201 Created` with a response time of 39 ms and a size of 1.48 KB. The JSON response body is:

```

1 {
2   "success": true,
3   "message": "Reply created successfully",
4   "data": {
5     "id": "c8a5a9c2-8636-49b0-b1c4-1db664dad288",
6     "content": "This is a test reply for unit testing",
7     "postId": "64112ee2-d281-4fa6-a3a7-4b1ea24ae17f",
8     "userId": 10,
9     "createdAt": "2025-04-02T18:00:51.797Z",
10    "updatedAt": "2025-04-02T18:00:51.797Z",
11    "user": {
12      "id": 10,
13      "name": "John Doe"
14    }
15  }
16}

```

- API to delete a reply.

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3001/api/forum/reply/7d417243-4765-4466-b8a8-948948fa2661`. The method is set to `DELETE`. The response status is `200 OK` with a response time of 46 ms and a size of 1.14 KB. The response body is:

```

1  {
2   |   "content": "This is to test delete reply"
3  }

```

- API to delete a post

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3001/api/forum/post/64112ee2-d281-4fa6-a3a7-4b1ea24ae17f`. The method is set to `DELETE`. The response status is `200 OK` with a response time of 41 ms and a size of 1.14 KB. The response body is:

```

1  {
2   |   "success": true,
3   |   "message": "Post deleted successfully"
4  }

```

Students and faculty are able to create new queries over Forum.

The screenshot shows the EE321 course forum interface. On the left, there's a sidebar with links for Course Home, Lectures, Announcements, Calendar, Result, and Forum. The 'Forum' link is highlighted with a black bar. The main area is titled 'FORUM' and shows two posts from user 'student2'. The first post, dated 4/3/2025, 12:07:51 AM, says 'Second Query Created' and has a 'Reply' button. The second post, dated 4/3/2025, 12:07:27 AM, says 'First Query Created' and also has a 'Reply' button. There are delete icons next to each post.

Instructor and other students of the course are able to see and reply to the query raised.

This screenshot shows the same EE321 forum interface. It displays a post from 'student2' and a reply from 'faculty1 Instructor'. The reply text reads 'I as a fulty able to see the query raised.' A 'Reply' button is visible at the bottom of the reply section.

Structural Coverage:

The unit is working well and fulfils 100% of the requirements of the SRS.

Results

Unit Details:

Classes: Result class, Exam class and its interaction with User class and Course class.

Functions: There are various functions for performing read, write, delete and modifying tasks on the result and exam data, description of each of them is given within the test results section.

Test Owner:

Dhruv Rai, 220365

Test Date:

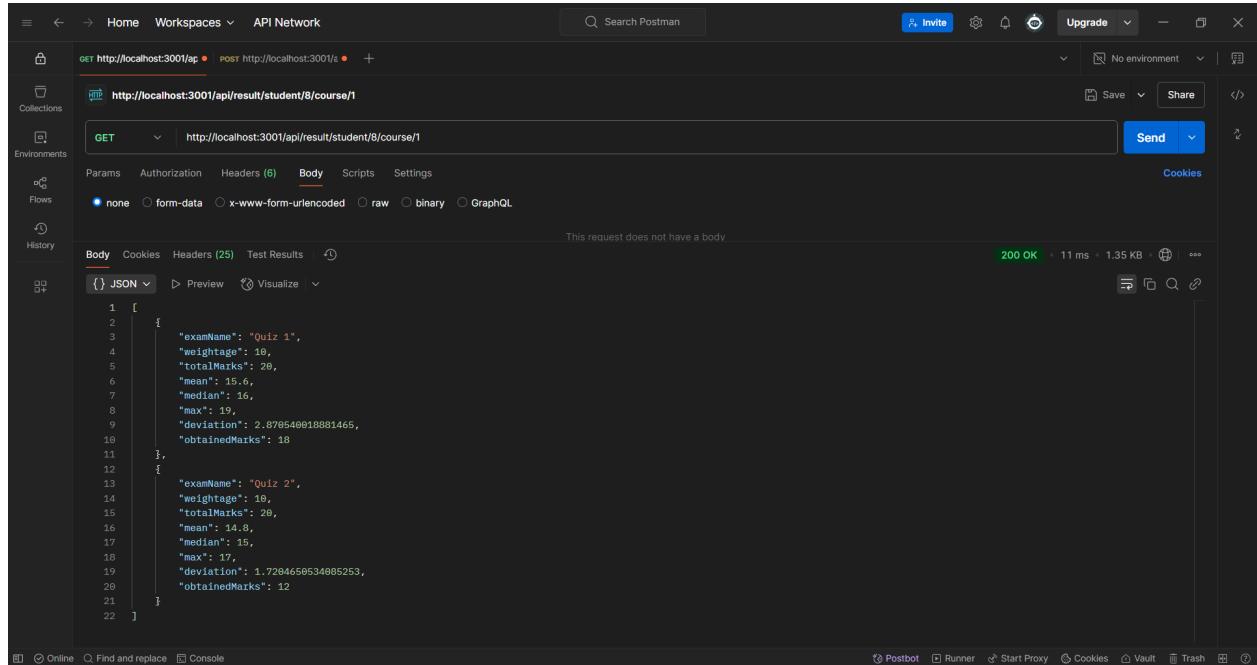
26/03/25-02/04/25

Test Results:

First, I presented the tests for the backend and then for the frontend part. The details of the tests are as follows-

Api to fetch result for a student-

I have tested this api and it is able to get data every time when it is present in the database. I have also tested it for the edge cases when the course id is wrong or the user id is wrong or the database is not present and in every scenario it worked as expected.

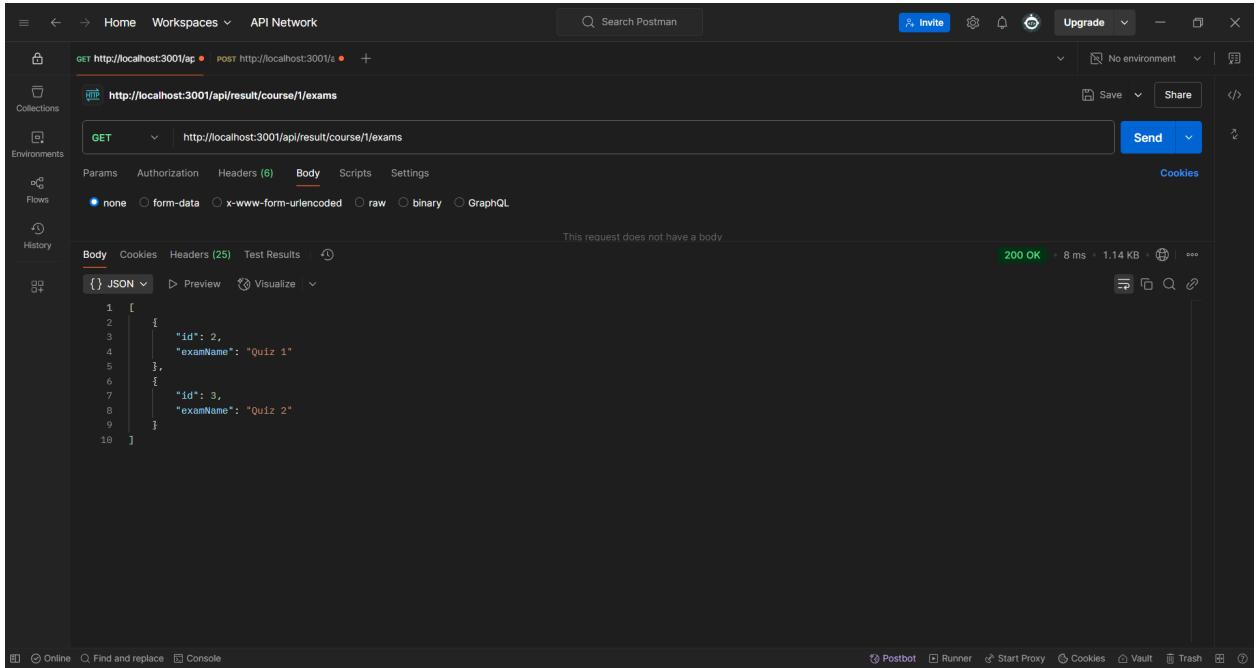


```

[{"id": 1, "studentId": 8, "courseId": 1, "examName": "Quiz 1", "weightage": 10, "totalMarks": 20, "mean": 15.6, "median": 16, "max": 19, "deviation": 2.870540018881465, "obtainedMarks": 18}, {"id": 2, "studentId": 8, "courseId": 1, "examName": "Quiz 2", "weightage": 10, "totalMarks": 20, "mean": 14.8, "median": 15, "max": 17, "deviation": 1.7204650534085253, "obtainedMarks": 12}]
  
```

Api to fetch exam names and id for a course-

It takes course id as input and gives the name and id of all the exams present in the database for that particular course. It gives a major bug in initial testing in which this api is giving all the exams rather than the exams related to a particular course. I have successfully resolved this bug and now this is working fine in most of the scenarios.



Api to fetch exams and details of them for a course-

It takes course id as input and gives all the details for all the exams present in the database for that particular course. This api worked fine in normal cases and also for the edge cases.

```

[{"id": 2, "examName": "Quiz 1", "totalMarks": 26, "weightage": 10, "mean": 15.6, "median": 16, "max": 19, "deviation": 2.870540018881465}, {"id": 3, "examName": "Quiz 2", "totalMarks": 26, "weightage": 10, "mean": 14.8, "median": 15, "max": 17, "deviation": 1.7204650534005253}]
  
```

Api to fetch results for all the students for an exam-

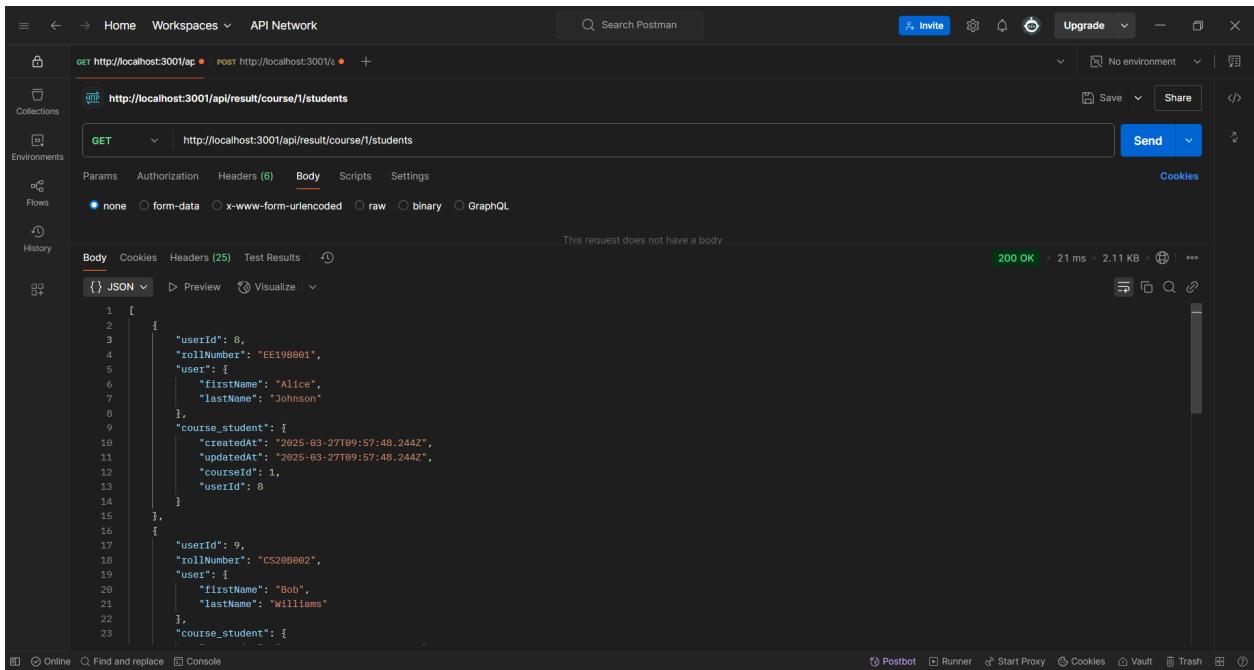
It takes course id and exam id as input and gives results of all the students for that exam. It will be used by faculty to view results for a particular exam. This api worked fine in normal cases but has minor bug when obtained marks are not present for a student. I have successfully fixed this bug and now in absence of obtained marks it returns null for that user without giving any error. Now this unit is working fine.

```

{
  "examName": "Quiz 2",
  "totalMarks": 26,
  "weightage": 10,
  "mean": 14.8,
  "median": 15,
  "max": 17,
  "deviation": 1.7204650534005253,
  "results": [
    {
      "userId": 8,
      "rollNumber": "EE19B001",
      "name": "Alice Johnson",
      "obtainedMarks": 12
    },
    {
      "userId": 9,
      "rollNumber": "CS20B002",
      "name": "Bob Williams",
      "obtainedMarks": 14
    },
    {
      "userId": 10,
      "rollNumber": null,
      "name": null,
      "obtainedMarks": null
    }
  ]
}
  
```

Api to fetch all the enrolled students in a course-

This unit will be used to get details of all the students enrolled in a course. This will be called by frontend when the faculty wants to add results for an exam. This unit passed all the tests successfully for most of the scenarios.



The screenshot shows the Postman application interface. A GET request is made to `http://localhost:3001/api/result/course/1/students`. The response is a 200 OK status with a response time of 21 ms and a size of 2.11 KB. The JSON response body contains two student records:

```
[{"id": 1, "userId": 8, "rollNumber": "EE19B001", "user": {"firstName": "Alice", "lastName": "Johnson"}, "course_student": {"createdAt": "2025-03-27T09:57:48.244Z", "updatedAt": "2025-03-27T09:57:48.244Z", "courseId": 1, "userId": 8}, "course": {"courseId": 1, "courseName": "Introduction to Computer Science", "description": "This course covers the basics of computer science, including algorithms, data structures, and programming concepts."}}, {"id": 2, "userId": 9, "rollNumber": "CS20B002", "user": {"firstName": "Bob", "lastName": "Williams"}, "course_student": {"createdAt": "2025-03-27T09:57:48.244Z", "updatedAt": "2025-03-27T09:57:48.244Z", "courseId": 1, "userId": 9}, "course": {"courseId": 1, "courseName": "Introduction to Computer Science", "description": "This course covers the basics of computer science, including algorithms, data structures, and programming concepts."}}]
```

Api to publish result for a new exam-

This unit will be used to publish results for a particular exam, this will be called by the faculty side frontend. One requirement for this api is that the data should only contain marks for the students enrolled in a course. If the data contain more or less students it can cause errors. Although it affects robustness but with a smart frontend this can be ensured that the data has proper format. Once data is in the correct format it works fine in all the scenarios.

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3001/api/result/course/1/publish`. The response status is `201 Created` with a response time of `95 ms` and a size of `1.14 KB`. The response body is:

```

1 {
2   "examName": "Midterm Exam",
3   "weightage": 20,
4   "totalMarks": 100,
5   "results": [
6     {
7       "userId": 8,
8       "obtainedMarks": 85
9     },
10    {
11      "userId": 9,
12      "obtainedMarks": 90
13    },
14    {
15      "userId": 10,
16      "obtainedMarks": 78
17    },
18    {
19      "userId": 11,
20      "obtainedMarks": 88
21  }
22]
23}
24
25   "message": "Exam and results published successfully"
26
27

```

Api to delete an exam and all its related results from the database-

This unit works fine for all the cases. When there is an exam with the given exam id it performs as expected and shows the completion message as output.

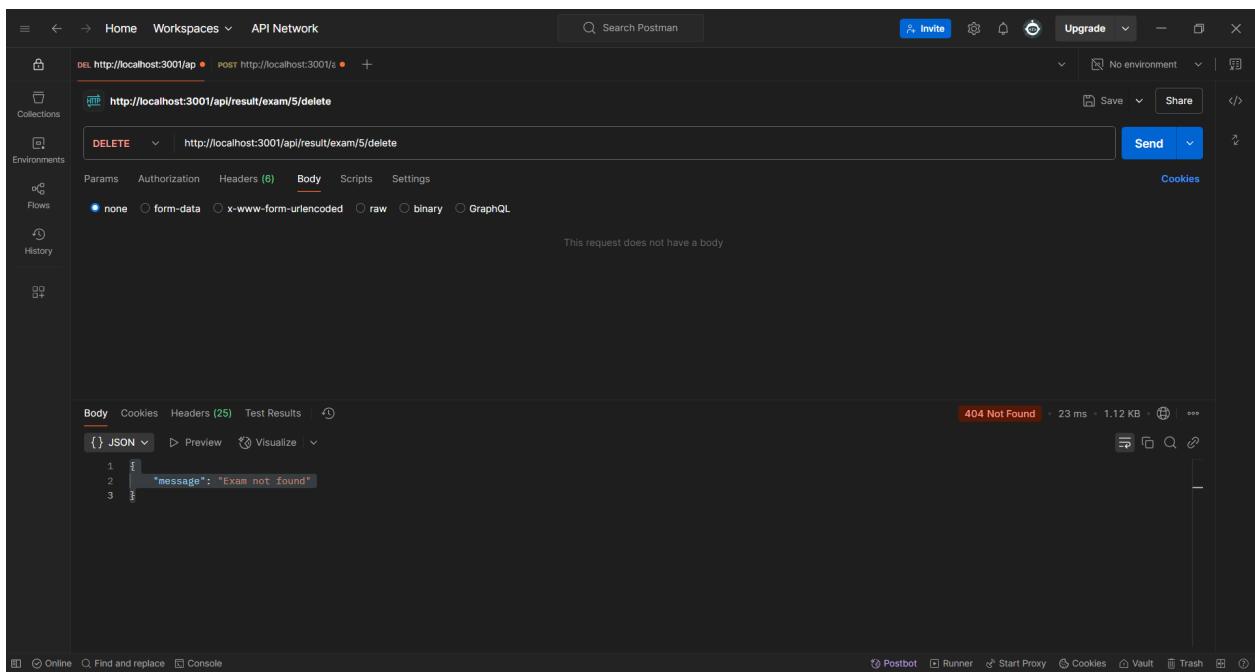
The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3001/api/result/exam/4/delete`. The response status is `200 OK` with a response time of `44 ms` and a size of `1.15 KB`. The response body is:

```

1 {
2   "message": "Exam and associated results deleted successfully"
3 }

```

But when no exam with a given exam id is present in the database. It correctly presents an error message which says “Exam not found”. In all other cases it also performs well.



Api to modify exam result-

This unit performs as expected for all the normal cases as well as for the edge cases like cases when wrong exam id is given or data is not present in the database.

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', 'Search Postman', 'Invite', 'Upgrade', and environment settings. On the left, there are sections for 'Collections', 'Environments', 'Flows', and 'History'. The main workspace displays a sequence of three requests: 'DEL http://localhost:3001/api/exam', 'Post http://localhost:3001/api/exam', and 'PUT http://localhost:3001/api/exam'. The current request is a 'PUT' to 'http://localhost:3001/api/result/exam/2/modify'. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {  
2     "examName": "Quiz 1 - Updated",  
3     "weightage": 25,  
4     "totalMarks": 100,  
5     "results": [  
6         {  
7             "userId": 8,  
8             "obtainedMarks": 88  
9         },  
10        {  
11            "userId": 9,  
12            "obtainedMarks": 92  
13        },  
14        {  
15            "userId": 10,  
16            "obtainedMarks": 80  
17        },  
18        {  
19            "userId": 11,  
20            "obtainedMarks": 85  
21        }  
22    ]  
23}
```

The 'Body' tab also contains tabs for 'Cookies', 'Headers (25)', 'Test Results', and a preview section showing the response body:

```
1 {  
2     "message": "Exam and results modified successfully"  
3 }
```

The bottom status bar shows the response: '200 OK' with a duration of '86 ms' and a size of '1.14 KB'. Navigation icons for Postbot, Runner, Start Proxy, Cookies, Vault, and Trash are also present.

Now I will present the test results for the frontend units which will serve as the user interface for the result page in the system. For now all data is static as the backend is not connected it is tested only to ensure that frontend units are working fine.

The faculty result page-

This page renders successfully in all the cases.

The screenshot shows the 'RESULTS' section of the EE321 course page. At the top, it displays 'EE321 • 4 Credits • Fall 2023'. Below this, a message says 'Select Exam to view details:' followed by a dropdown menu with the option '-- Select an exam --'. A blue button labeled 'Add Result' is located to the right. A table titled 'All results' lists exam names, total marks, weightage, mean, median, max, and std dev. The table contains two rows: 'Quiz 1 - Updated' (100 total marks, 25% weightage) and 'Quiz 2' (20 total marks, 10% weightage). On the left sidebar, there are links for Course Home, Lectures, Announcements, Calendar, Result (which is highlighted), and Forum. At the bottom left is a 'Logout' link.

Add result form-

This page works successfully and shows the list of students enrolled in the course.

The screenshot shows the 'Add New Exam Result' form. It has fields for 'Exam Name' (e.g., Midterm Exam), 'Total Marks' (e.g., 100), and 'Weightage (%)' (e.g., 20). Below this is a table titled 'Student Marks' with columns for 'Roll Number', 'Name', and 'Marks'. The table lists five students: Alice Johnson, Bob Williams, Charlie Brown, Diana Miller, and Ethan Davis, each with a mark of 0. At the bottom right are 'Cancel' and 'Publish Results' buttons. The left sidebar is identical to the previous screenshot, featuring links for Course Home, Lectures, Announcements, Calendar, Result (highlighted), and Forum, along with a 'Logout' link.

It also ensures that the obtained marks for any student does not exceed above the total marks available for the exam.

The screenshot shows a user interface for adding exam results. On the left, a sidebar menu for 'EE321' includes 'Course Home', 'Lectures', 'Announcements', 'Calendar', 'Result' (which is selected), and 'Forum'. Below the sidebar is a 'Logout' link. The main content area is titled 'Add New Exam Result'. It has fields for 'Exam Name' (quiz 3), 'Total Marks' (40), and 'Weightage (%)' (10). A table titled 'Student Marks' lists five students with their roll numbers, names, and marks. The mark for Alice Johnson is highlighted with a red border and the text 'Max is 40'. At the bottom right are 'Cancel' and 'Publish Results' buttons.

Roll Number	Name	Marks
EE19B001	Alice Johnson	45 Max is 40
CS20B002	Bob Williams	35
EE21B003	Charlie Brown	34
CS19B004	Diana Miller	23
EE20B005	Ethan Davis	28

It also makes sure that faculty enter marks for every student enrolled in the exam which is consistent with our backend units used for publishing the exam result.

This screenshot shows the same 'Add New Exam Result' page as the previous one, but with a different state. The mark for Alice Johnson is now empty and highlighted with a red border, accompanied by the error message 'Mark is required'. All other fields and data remain the same as in the first screenshot.

Roll Number	Name	Marks
EE19B001	Alice Johnson	0 Mark is required
CS20B002	Bob Williams	35
EE21B003	Charlie Brown	34
CS19B004	Diana Miller	23
EE20B005	Ethan Davis	28

It also checks that marks obtained by students are not negative.

Roll Number	Name	Marks
EE19B001	Alice Johnson	-12 Cannot be negative
CS20B002	Bob Williams	14
EE21B003	Charlie Brown	17
CS19B004	Diana Miller	15
EE20B005	Ethan Davis	11

If faculty enter correct marks for every student the 'publish result' button works fine.

Detailed result view for an exam-

The faculty can view the details of an exam for which the results have been published by selecting the exam from the dropdown menu or from the table.

Quiz	Mean	Std Dev	Median	Maximum
Quiz 1 - Updated	100	4.2	87.0	92
Quiz 2	20	1.7	14.8	15
Quiz 3	40	4.6	31.4	36

This unit also works fine and when faculty click on any exam the frontend displays the details of the exam correctly. At the top it displays basic details of the exam like exam name, total marks, weightage, mean, median, maximum, standard deviation.

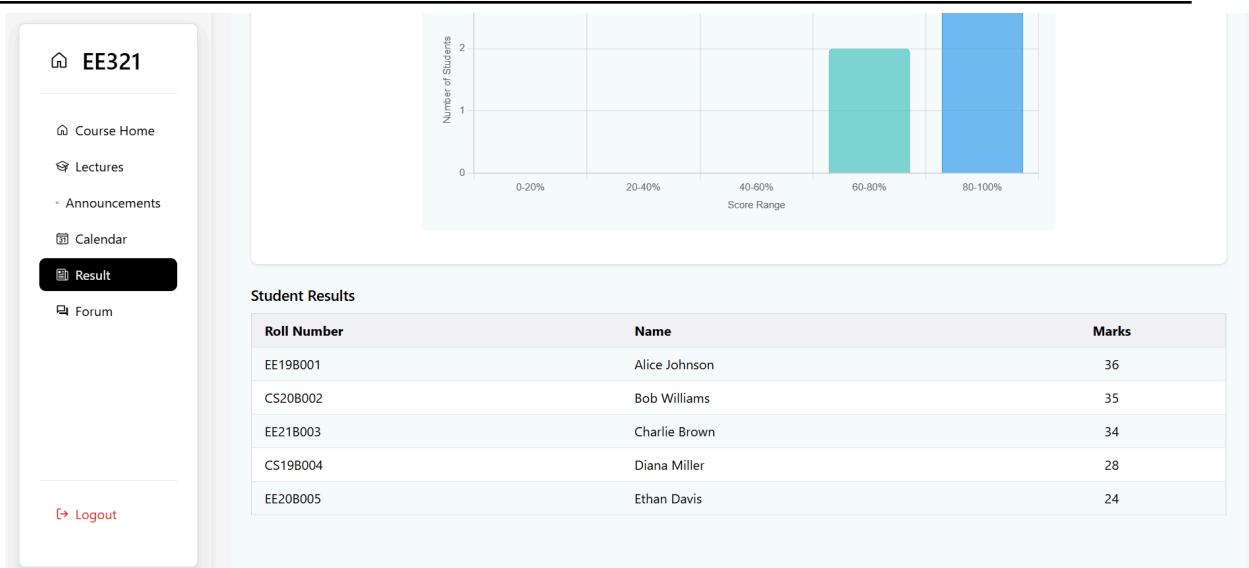
The screenshot shows the 'RESULTS' section for EE321. A dropdown menu is open, showing 'Quiz 3' selected. Below it, detailed statistics for Quiz 3 are displayed: Total Marks (40), Weightage (10%), Mean (31.4), Median (34), Maximum (36), and Standard Deviation (4.6). At the bottom, a bar chart titled 'Score Distribution' shows the number of students across different score ranges: 0-20% (0), 20-40% (0), 40-60% (0), 60-80% (2), and 80-100% (3).

Then it displays the graph of class performance which is basically a frequency distribution of marks. I have checked it with many data points and this unit correctly makes the graph for all of them so this unit is working as expected.

The screenshot shows the 'Student Results' section for EE321. It displays a table of marks obtained by every student enrolled in the course:

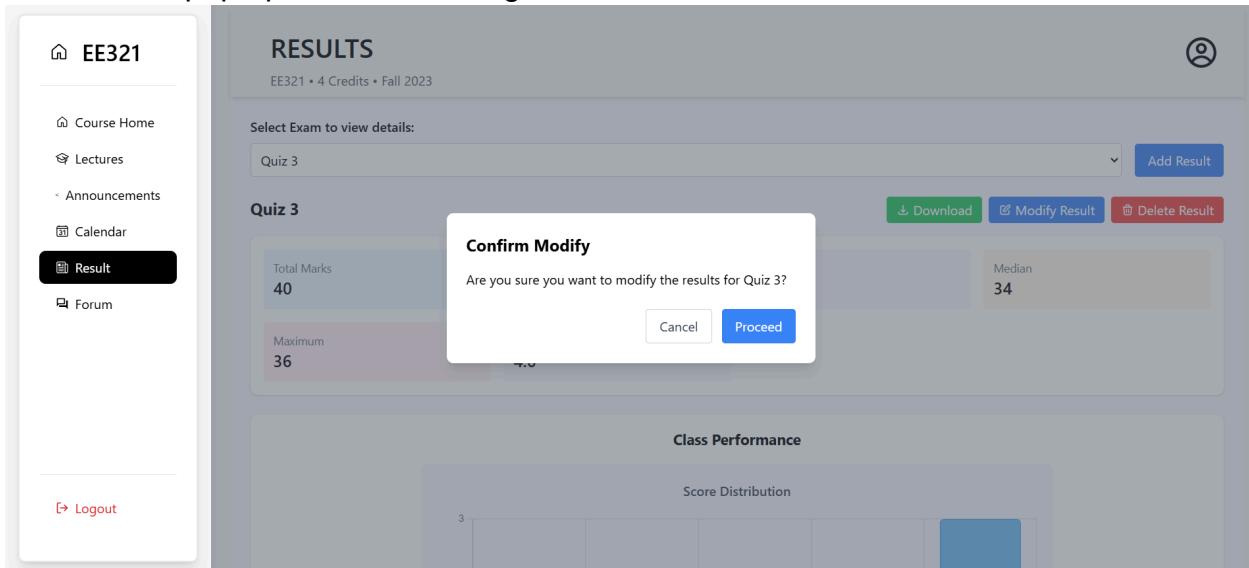
Roll Number	Name	Marks
EE19B001	Alice Johnson	36
CS20B002	Bob Williams	35
EE21B003	Craig Green	34

Then it display a table of marks obtained by every student enrolled in the course.



Modify result option-

When faculty click on the ‘modify result’ button the interface should give a confirmation pop up and it is working fine.



Once faculty click on the proceed button a form for modifying exam results opens as expected and faculty can update the details for the exam from here.

Roll Number	Name	Marks
EE19B001	Alice Johnson	36
CS20B002	Bob Williams	35
EE21B003	Charlie Brown	34
CS19B004	Diana Miller	28
EE20B005	Ethan Davis	24

Download result-

When faculty click on the ‘Download’ button result details for the exam get downloaded in an excel format. This helps faculty to export the data. This feature works fine in all the scenarios.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Exam Name:	Quiz 3																			
2	Total Marks:	40																			
3	Weightage:	10%																			
4	Mean:	28.8																			
5	Median:	28																			
6	Maximum:	36																			
7	Standard Dev:	5.5																			
9	Roll Number	Name	Marks																		
10	EE19B001	Alice Jof	36																		
11	CS20B002	Bob Wil	22																		
12	EE21B003	Charlie I	34																		
13	CS19B004	Diana M	28																		
14	EE20B005	Ethan D	24																		

Delete result-

Faculty can delete the result for any exam by clicking on the ‘Delete result’ button. Once faculty clicks on it a confirmation pop-up comes up as expected and when faculty clicks on the ‘Delete’ button in the pop-up also the result gets deleted. This unit works fine in most of the cases.

The screenshot shows the 'RESULTS' section of the EE321 course. On the left sidebar, the 'Result' tab is selected. In the main area, a 'Quiz 3' card displays 'Total Marks: 40' and 'Maximum: 36'. A 'Confirm Delete' dialog box is open, asking 'Are you sure you want to delete the results for Quiz 3? This action cannot be undone.' It has 'Cancel' and 'Delete' buttons. At the top right of the main area, there are 'Download', 'Modify Result', and 'Delete Result' buttons. Below the card, a 'Class Performance' section shows a 'Score Distribution' graph with a callout for the '60-80%' range (2 students, 40.0%).

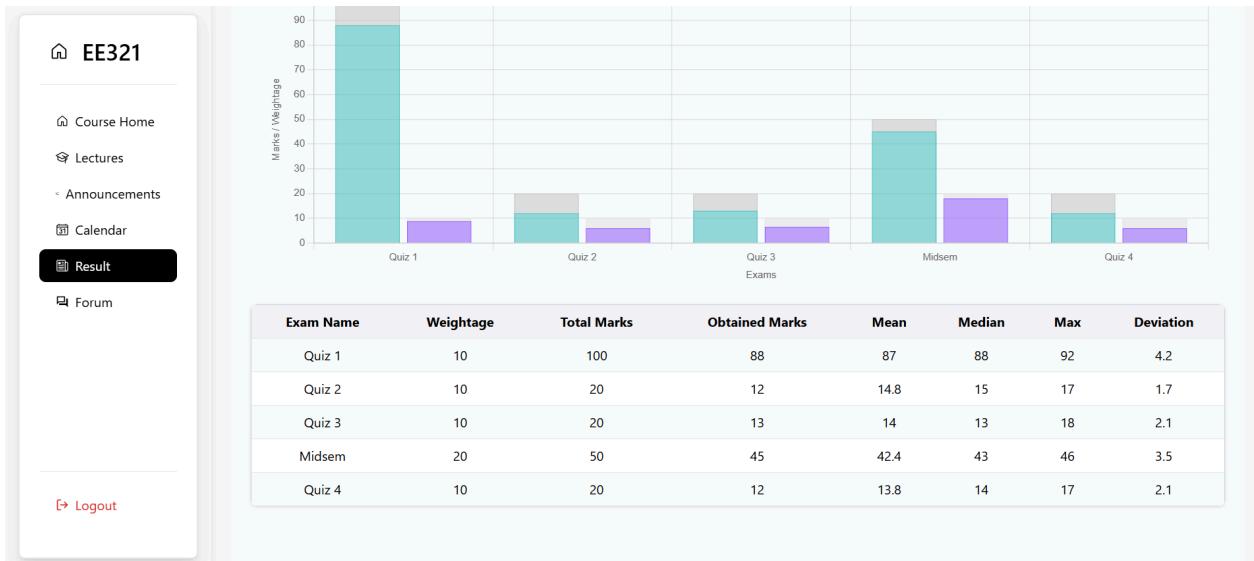
Student view-

The student only has read access to the results related to him. When a student clicks on the result tab he can see a graph telling his performance in various exams till then. The graph shows both the marks and the weightage that he has scored from the total.

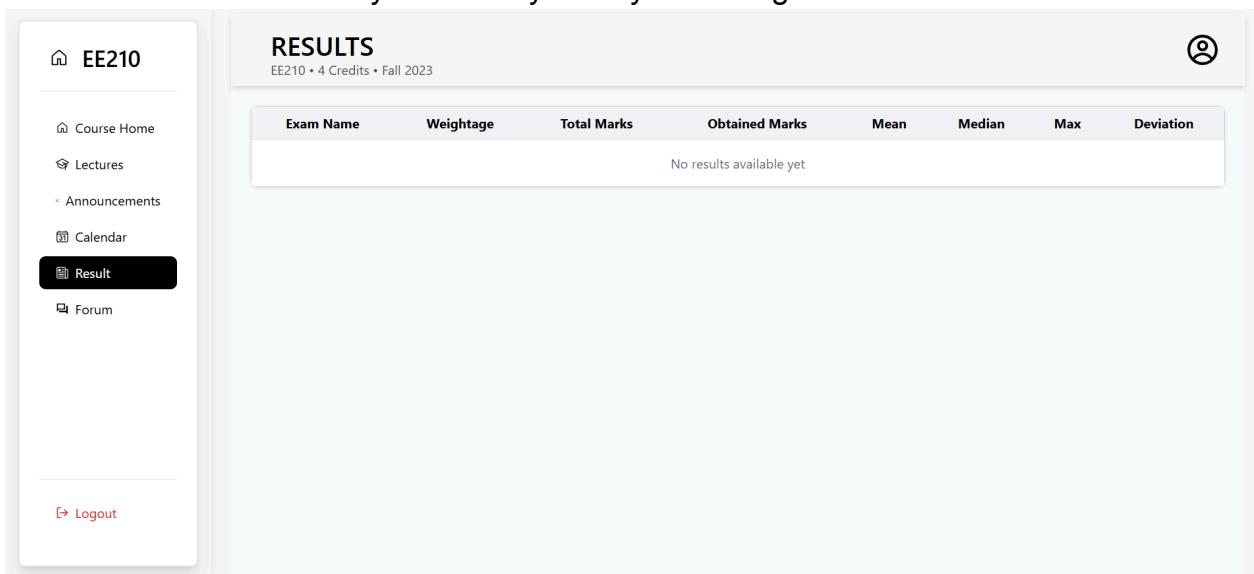
The screenshot shows the 'RESULTS' section of the EE321 course. The 'Result' tab is selected in the sidebar. The main area features a stacked bar chart comparing 'Obtained Marks' (teal) and 'Obtained Weightage' (purple) across 'Quiz 1', 'Quiz 2', 'Quiz 3', 'Midsem', and 'Quiz 4'. A callout highlights 'Quiz 1' with 'Obtained Marks: 88'. Below the chart is a detailed results table:

Exam Name	Weightage	Total Marks	Obtained Marks	Mean	Median	Max	Deviation
Quiz 1	10	100	88	87	88	92	4.2
Quiz 2	10	20	12	14.8	15	17	1.7
Quiz 3	10	20	13	14	13	18	2.1

Below the graph students can also view its result and other related information in a tabular format. This unit works fine in all the normal cases.



One edge case occurs when there are no results available in a course. This situation is also handled by our unit by clearly indicating the absence of results.



Structural coverage:

In most of the backend units, I have used **function coverage** and **branch coverage** as the criteria for exhaustive testing.

For frontend units I have used **branch coverage** as the criteria as it ensures the proper working in all the different scenarios.

Additional comments:

One minor bug was found in one of the backend unit and it was rectified then only.

Lectures

Unit details:

Involves the testing of the Lectures Database and APIs. The Lectures database is unique to each course, where a Heading object has a subheading and a subheading object has a lecturer along with a body, along with the corresponding frontend. Only Faculties associated with that course can create lectures in that course, but they can be viewed by all students and other Faculty members.

Test Owner:

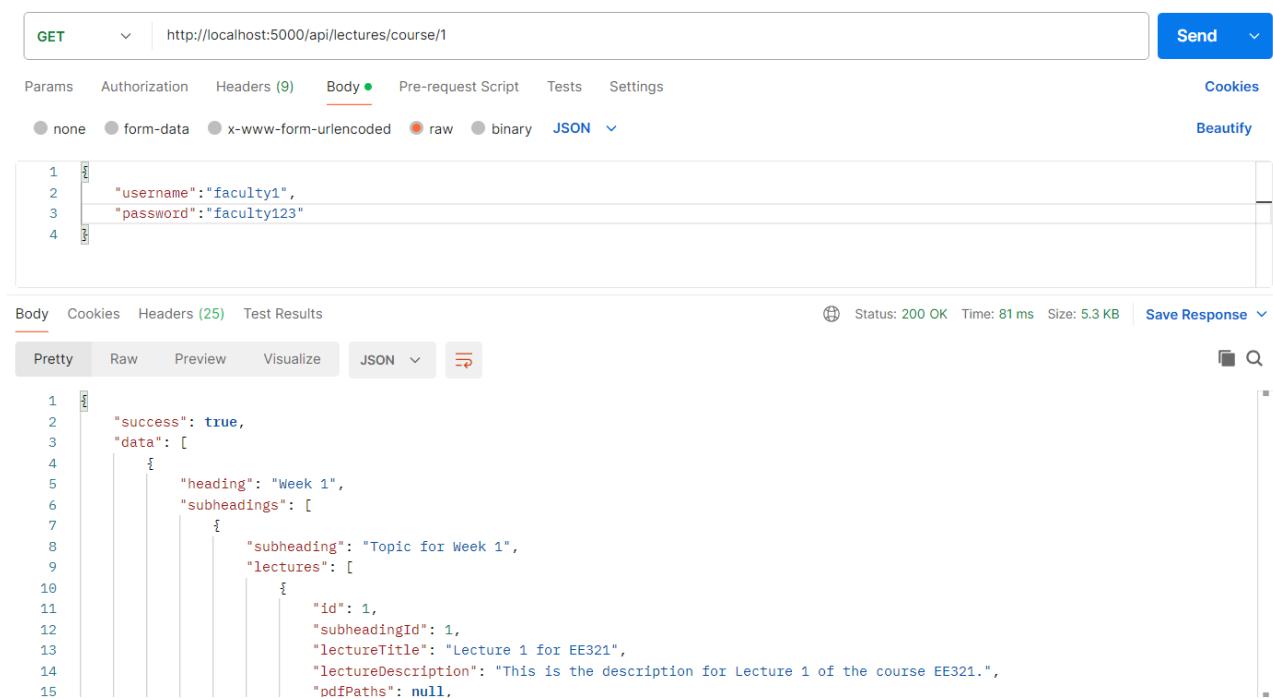
Abhijeet Agarwal, 210025

Test Date:

2 April 2025

Test Results:

Getting all Lectures: Test Successful

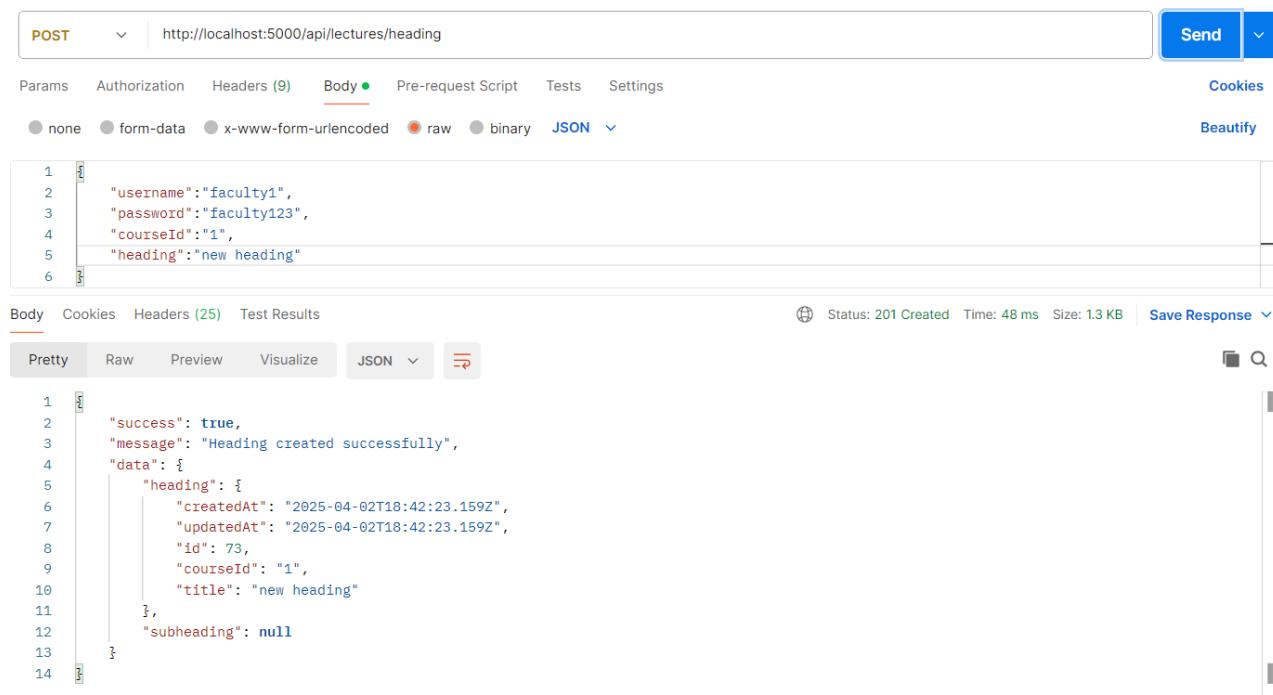


The screenshot shows the Postman application interface during a test run. The top bar indicates a 'GET' method, the URL is 'http://localhost:5000/api/lectures/course/1', and the 'Body' tab is selected. In the body section, there is a JSON payload with two fields: 'username' set to 'faculty1' and 'password' set to 'faculty123'. The 'Pretty' tab is selected in the bottom navigation bar, displaying the JSON response in a readable format. The response body is as follows:

```
1
2   "success": true,
3   "data": [
4     {
5       "heading": "Week 1",
6       "subheadings": [
7         {
8           "subheading": "Topic for Week 1",
9           "lectures": [
10             {
11               "id": 1,
12               "subheadingId": 1,
13               "lectureTitle": "Lecture 1 for EE321",
14               "lectureDescription": "This is the description for Lecture 1 of the course EE321.",
15               "pdfPaths": null,
16             }
17           ]
18         }
19       ]
20     }
21   ]
```

The status bar at the bottom right shows 'Status: 200 OK' and 'Time: 81 ms'.

Creating new Module: Test Successful



The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:5000/api/lectures/heading
- Body (JSON):**

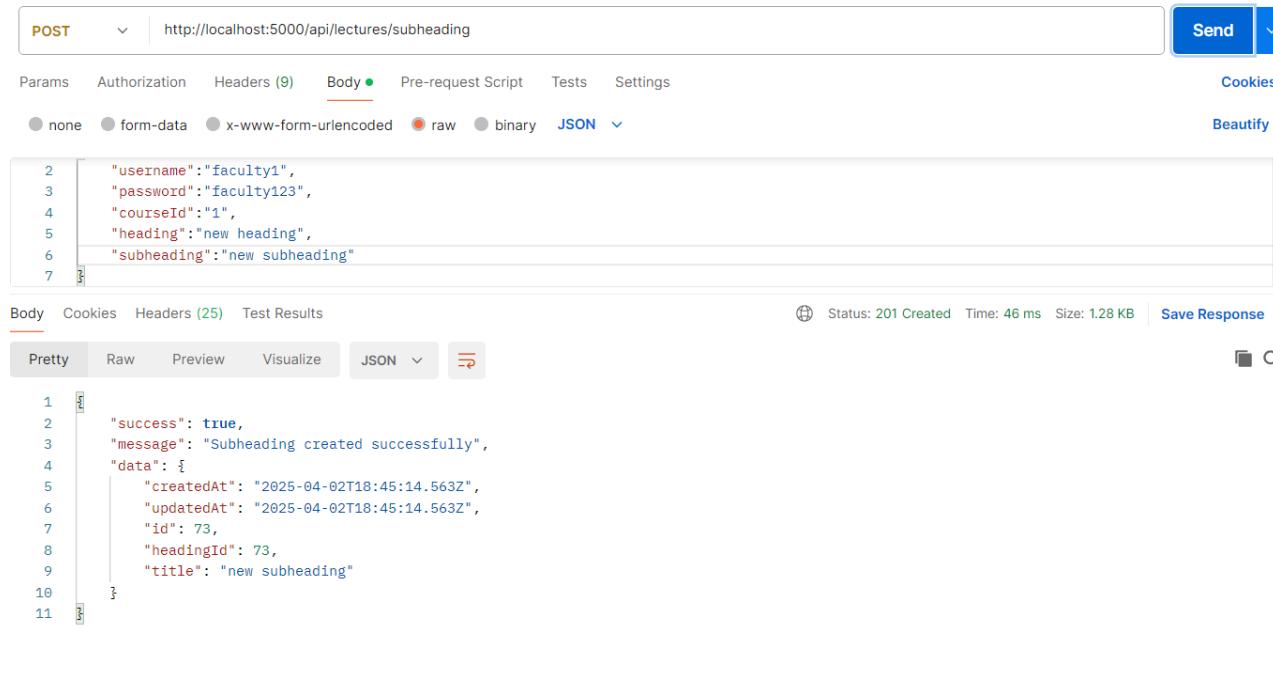
```

1
2   "username": "faculty1",
3   "password": "faculty123",
4   "courseId": "1",
5   "heading": "new heading"
6
    
```
- Response Status:** 201 Created
- Response Time:** 48 ms
- Response Size:** 1.3 KB
- Response Body (Pretty JSON):**

```

1
2   "success": true,
3   "message": "Heading created successfully",
4   "data": {
5     "heading": {
6       "createdAt": "2025-04-02T18:42:23.159Z",
7       "updatedAt": "2025-04-02T18:42:23.159Z",
8       "id": 73,
9       "courseId": "1",
10      "title": "new heading"
11    },
12    "subheading": null
13  }
14
    
```

Creating new Section: Test Successful



The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:5000/api/lectures/subheading
- Body (JSON):**

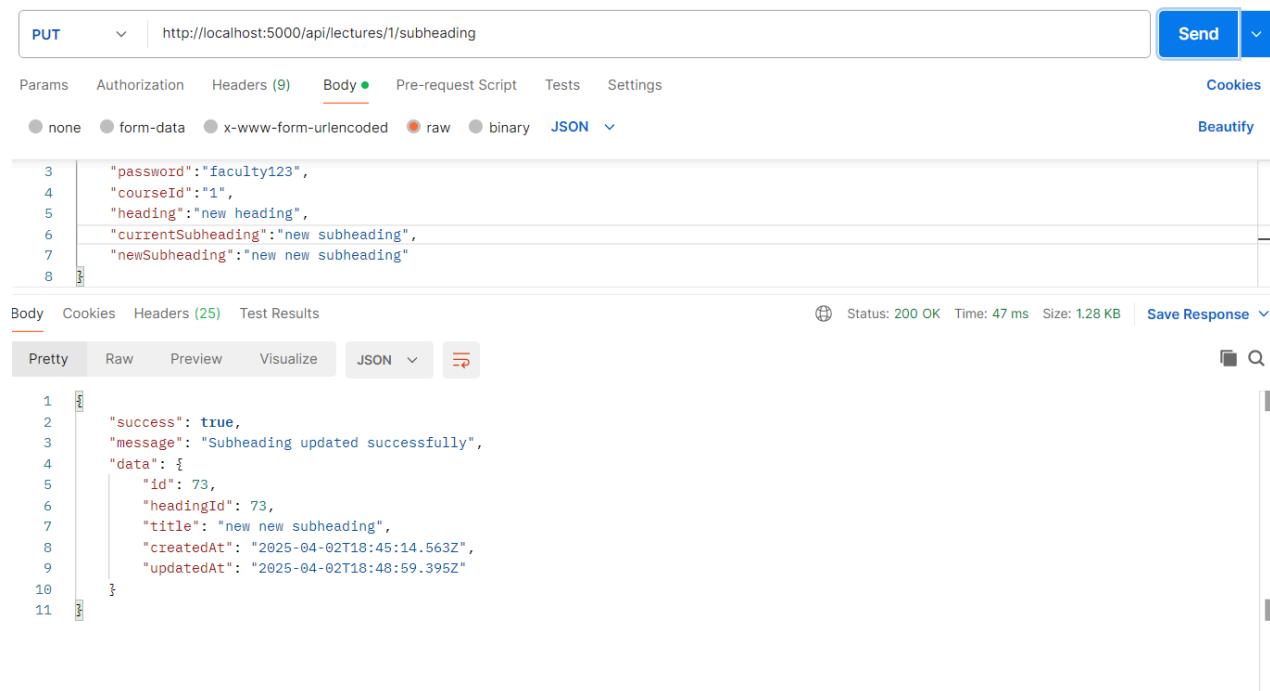
```

2   "username": "faculty1",
3   "password": "faculty123",
4   "courseId": "1",
5   "heading": "new heading",
6   "subheading": "new subheading"
7
    
```
- Response Status:** 201 Created
- Response Time:** 46 ms
- Response Size:** 1.28 KB
- Response Body (Pretty JSON):**

```

1
2   "success": true,
3   "message": "Subheading created successfully",
4   "data": {
5     "subheading": {
6       "createdAt": "2025-04-02T18:45:14.563Z",
7       "updatedAt": "2025-04-02T18:45:14.563Z",
8       "id": 73,
9       "headingId": 73,
10      "title": "new subheading"
11    }
12
    
```

Editing Section: Test Successful



PUT http://localhost:5000/api/lectures/1/subheading

Body (JSON)

```

3   "password": "faculty123",
4   "courseId": "1",
5   "heading": "new heading",
6   "currentSubheading": "new subheading",
7   "newSubheading": "new new subheading"
8

```

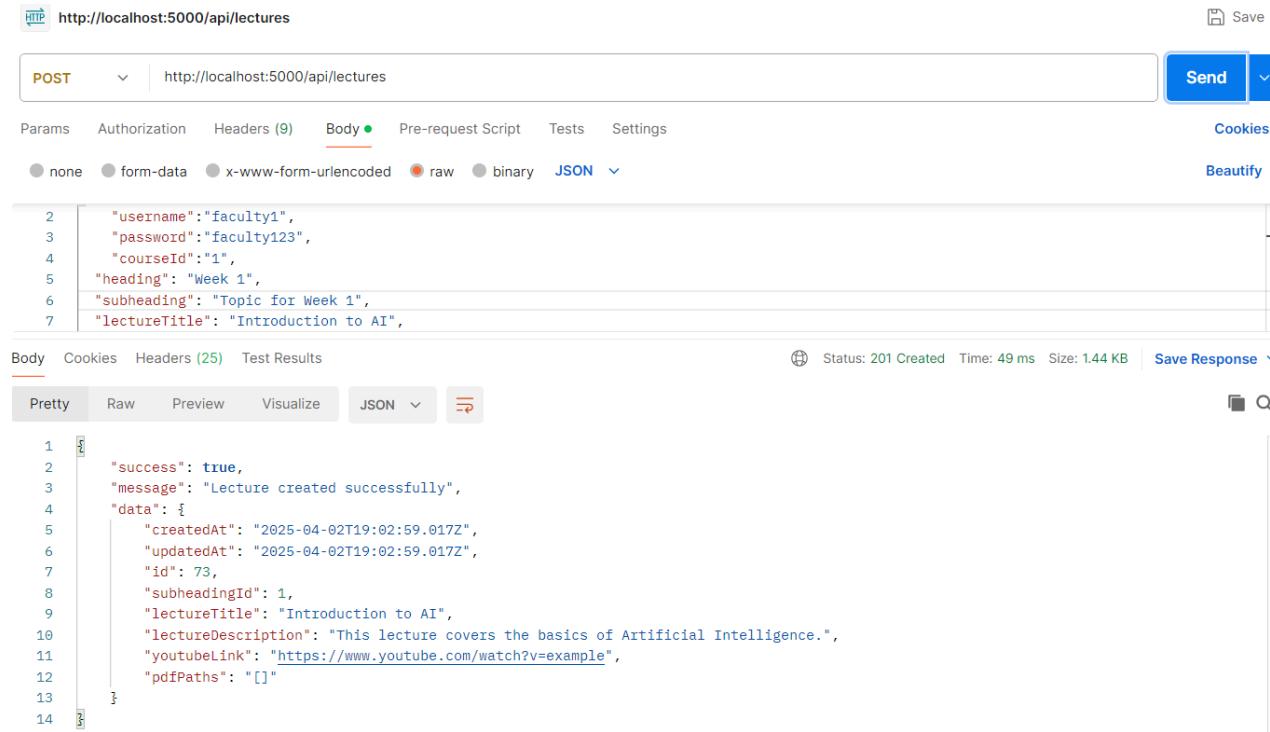
Status: 200 OK Time: 47 ms Size: 1.28 KB Save Response

```

1
2   "success": true,
3   "message": "Subheading updated successfully",
4   "data": {
5     "id": 73,
6     "headingId": 73,
7     "title": "new new subheading",
8     "createdAt": "2025-04-02T18:45:14.563Z",
9     "updatedAt": "2025-04-02T18:48:59.395Z"
10
11

```

Creating new Lecture: Test Successful



HTTP POST http://localhost:5000/api/lectures

Body (JSON)

```

2   "username": "faculty1",
3   "password": "faculty123",
4   "courseId": "1",
5   "heading": "Week 1",
6   "subheading": "Topic for Week 1",
7   "lectureTitle": "Introduction to AI",

```

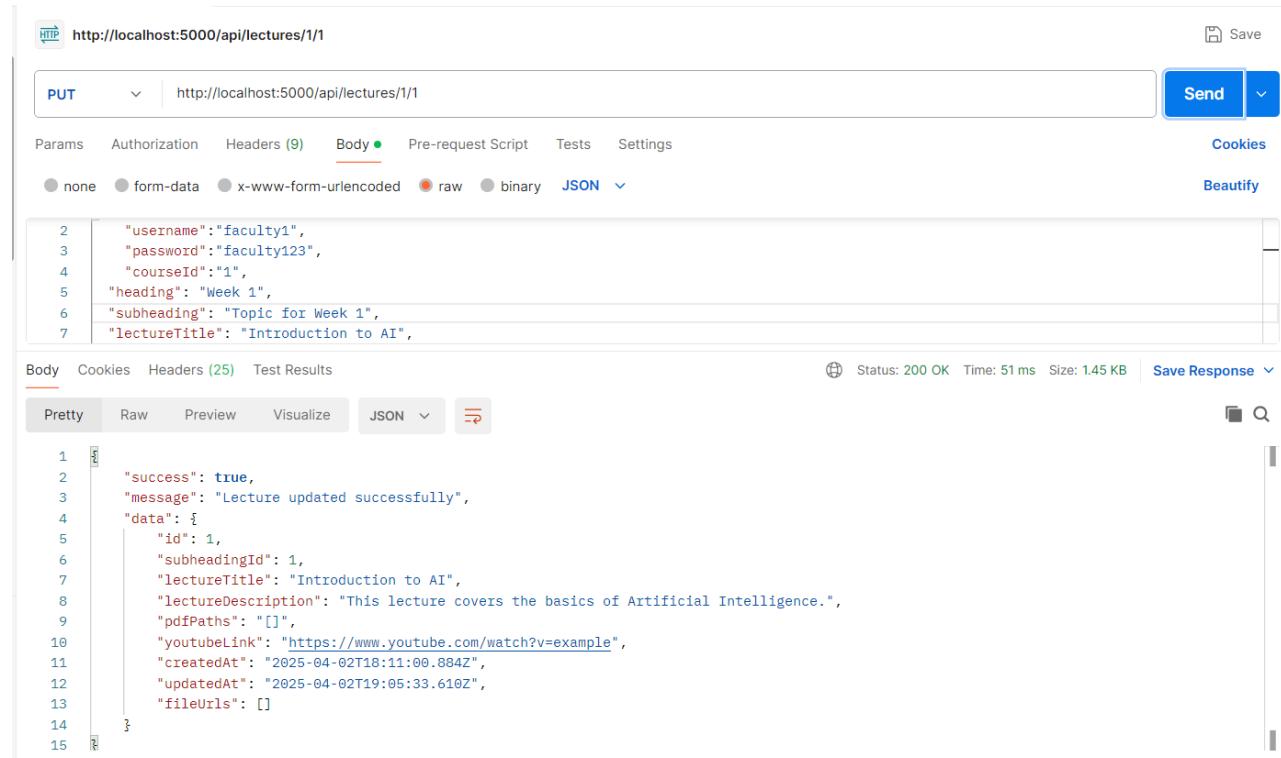
Status: 201 Created Time: 49 ms Size: 1.44 KB Save Response

```

1
2   "success": true,
3   "message": "Lecture created successfully",
4   "data": {
5     "createdAt": "2025-04-02T19:02:59.017Z",
6     "updatedAt": "2025-04-02T19:02:59.017Z",
7     "id": 73,
8     "subheadingId": 1,
9     "lectureTitle": "Introduction to AI",
10    "lectureDescription": "This lecture covers the basics of Artificial Intelligence.",
11    "youtubeLink": "https://www.youtube.com/watch?v=example",
12    "pdfPaths": []
13
14

```

Editing Lecture: Test Successful



The screenshot shows a POST request to `http://localhost:5000/api/lectures/1/1`. The request body is a JSON object:

```

2   "username": "faculty1",
3   "password": "faculty123",
4   "courseId": "1",
5   "heading": "Week 1",
6   "subheading": "Topic for Week 1",
7   "lectureTitle": "Introduction to AI",

```

The response status is 200 OK, indicating the lecture was updated successfully. The response body is:

```

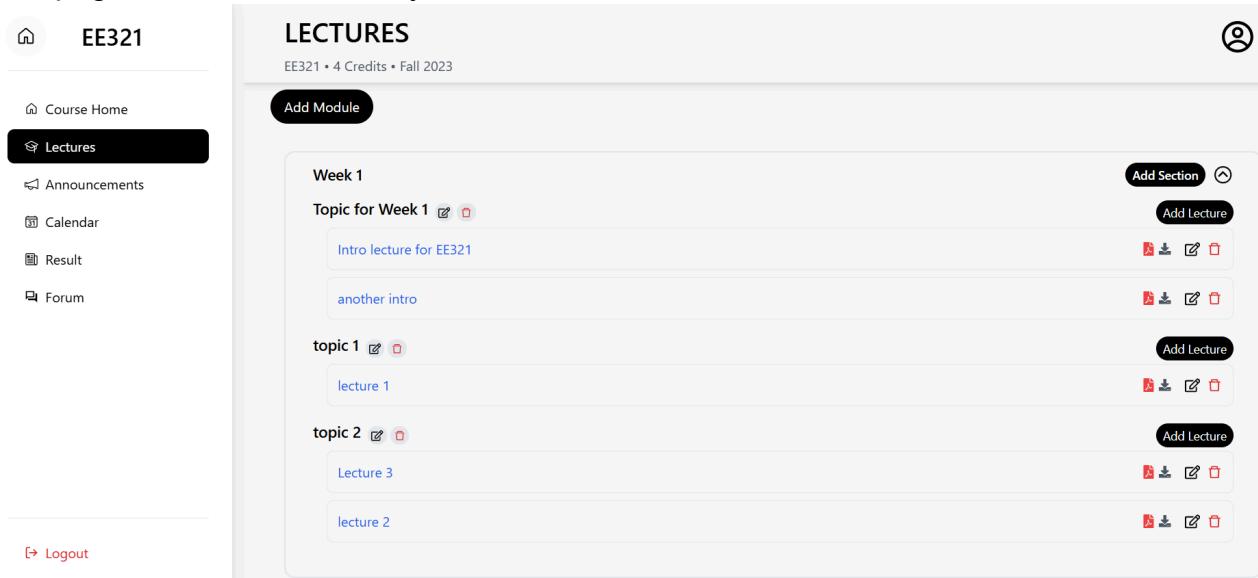
1
2   "success": true,
3   "message": "Lecture updated successfully",
4   "data": {
5     "id": 1,
6     "subheadingId": 1,
7     "lectureTitle": "Introduction to AI",
8     "lectureDescription": "This lecture covers the basics of Artificial Intelligence.",
9     "pdfPaths": "[ ]",
10    "youtubeLink": "https://www.youtube.com/watch?v=example",
11    "createdAt": "2025-04-02T18:11:00.884Z",
12    "updatedAt": "2025-04-02T19:05:33.610Z",
13    "fileUrls": []
14  }
15

```

The test results for the frontend units, which will serve as the user interface for the lectures page, are as shown. Since the backend is not yet connected, the tests were conducted using static data to verify that the frontend components are functioning as expected.

The faculty lectures page-

This page renders successfully in all the cases.



The screenshot shows the faculty lectures page for the EE321 course. The main content area displays a list of lectures under the heading "Week 1" and "Topic for Week 1". Each lecture item includes a title, a description, and a set of icons for managing the lecture (Edit, Delete, etc.). The sidebar on the left provides navigation links for the course.

- Course Home
- Lectures**
- Announcements
- Calendar
- Result
- Forum
- Logout

LECTURES

EE321 • 4 Credits • Fall 2023

Add Module

Week 1

Topic for Week 1

- Intro lecture for EE321
- another intro

topic 1

- lecture 1

topic 2

- Lecture 3
- lecture 2

Add Section

Add Lecture

Adding new module:

This page works successfully and shows the required module name for creating a new module.

The screenshot shows the LECTURES page for course EE321. On the left is a sidebar with links: Course Home, Lectures (which is selected and highlighted in black), Announcements, Calendar, Result, and Forum. At the bottom of the sidebar is a red 'Logout' button. The main content area has a title 'LECTURES' and a sub-header 'EE321 • 4 Credits • Fall 2023'. A 'Add Module' button is visible. Below it, there's a list of topics: 'Topic for Week 1' (with 'Intro lecture for E'), 'another intro', 'topic 1' (with 'lecture 1'), and 'topic 2' (with 'Lecture 3' and 'lecture 2'). A modal dialog box titled 'Add New Module' is open in the center. It contains a 'Module Name *' field with the placeholder 'Module Name'. At the bottom of the dialog are 'Cancel' and 'Save Module' buttons. To the right of the dialog, there are several small icons for adding sections and lectures, each with a red 'X' icon.

Adding new section:

This page works successfully and shows the required section name for creating a new section.

The screenshot shows the LECTURES page for course EE321, identical to the previous one but with a different modal dialog. The sidebar and course information are the same. The main content area shows the same list of topics. A modal dialog box titled 'Add Section' is open. It contains a 'Week 1' field with the value 'Week 1' and a 'Section Title' field with the placeholder 'Section Title'. At the bottom of the dialog are 'Cancel' and 'Save Section' buttons. The interface is consistent with the 'Add New Module' dialog, featuring the same set of small icons for managing sections and lectures.

Adding new lecture:

This page works successfully and shows all the required fields for creating the lecture.

The screenshot shows the 'LECTURES' page for course EE321. A modal window titled 'Add Lecture' is open. It contains fields for 'Lecture Title' (empty), 'Lecture Description' (empty), 'Upload Files' (button 'Choose Files' and message 'No file chosen'), and 'YouTube Link' (empty). At the bottom right of the modal are 'Cancel' and 'Submit' buttons. The background shows a list of lectures under 'Week 1' and 'Topic for Week 1'. To the right of the modal, there are sections for 'Add Section' and 'Add Lecture' with their respective icons.

Deleting lecture:

This page works successfully and asks for confirmation before deleting a lecture.

The screenshot shows the 'LECTURES' page for course EE321. A modal window titled 'Confirm Delete' is open, asking 'Are you sure you want to delete this lecture?'. Below the question are 'Cancel' and 'Delete' buttons. The background shows a list of lectures under 'Week 1' and 'Topic for Week 1'. To the right of the modal, there are sections for 'Add Section' and 'Add Lecture' with their respective icons.

Deleting section:

This page works successfully and asks for confirmation before deleting a section along with the warning that all the lectures within the section will be deleted.

The screenshot shows the LECTURES page for course EE321. On the left is a sidebar with links: Course Home, Lectures (which is selected and highlighted in black), Announcements, Calendar, Result, Forum, and Logout. The main area displays a list of sections and topics. A modal dialog box titled "Confirm Section Deletion" is open, asking if the user is sure they want to delete the section "Topic for Week 1". It also includes a warning message: "Warning: This will delete all lectures in this section." At the bottom of the dialog are two buttons: "Cancel" (gray) and "Delete" (red).

Downloading resources:

All the uploaded resources are correctly shown and are available for downloading.

The screenshot shows the LECTURES page for course EE321. The sidebar and main content area are similar to the previous screenshot. A modal dialog box titled "Lecture Supplements" is open, displaying a single PDF file: "ESO207_Midsem_2025-1743949894658-489408777.pdf". The dialog has a close button "x" in the top right corner.

The student lectures page-

This page renders successfully in all the cases and works correctly.

The screenshot shows the course page for EE321. The left sidebar includes links for Course Home, Lectures (which is selected and highlighted in black), Announcements, Calendar, Result, and Forum. The main content area is titled 'LECTURES' and shows 'EE321 • 4 Credits • Fall 2023'. It displays a list of lectures organized by week. Week 1 contains 'Topic for Week 1' with 'Intro lecture for EE321' and 'another intro', each with a download icon. Week 2 contains 'topic 1' with 'lecture 1', 'topic 2' with 'Lecture 3' and 'lecture 2', and 'Week 2' at the bottom. A user profile icon is in the top right corner.

Structural Coverage:

For most backend units, exhaustive testing was performed using both **function coverage** and **branch coverage** as criteria. For frontend units, **branch coverage** alone was used as the testing criterion, ensuring proper functionality across all possible scenarios

Forgot Password**Unit Details:**

Involves the testing of Forgot Password Database and APIs.

Test Owner:

Rahul Ahirwar, 220856

Test Date:

2 April 2025

Test Results:

- API to check username exists : Test Successful

The screenshot shows the Postman interface with a successful test run for the 'check-username' API. The request method is POST, the URL is <http://localhost:3001/api/auth/check-username>, and the response status is 200 OK. The response body is a JSON object indicating a user was found.

```

1 {
2   "success": true,
3   "message": "User found",
4   "userId": 13,
5   "email": "rahula22@iitk.ac.in"
6 }

```

- API to send OTP for password reset : Test Successful

The screenshot shows the Postman interface with a successful test run for the 'forgot-password' API. The request method is POST, the URL is <http://localhost:3001/api/auth/forgot-password>, and the response status is 200 OK. The response body is a JSON object indicating an OTP has been sent to the specified email.

```

1 {
2   "success": true,
3   "message": "OTP has been sent to your email"
4 }

```

- API to Verify OTP : Test Successful

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:3001/api/auth/verify-otp`. The request body is set to raw JSON, containing:

```

1 {
2   "email": "rahula22@iitk.ac.in",
3   "otp": "7853"
4 }

```

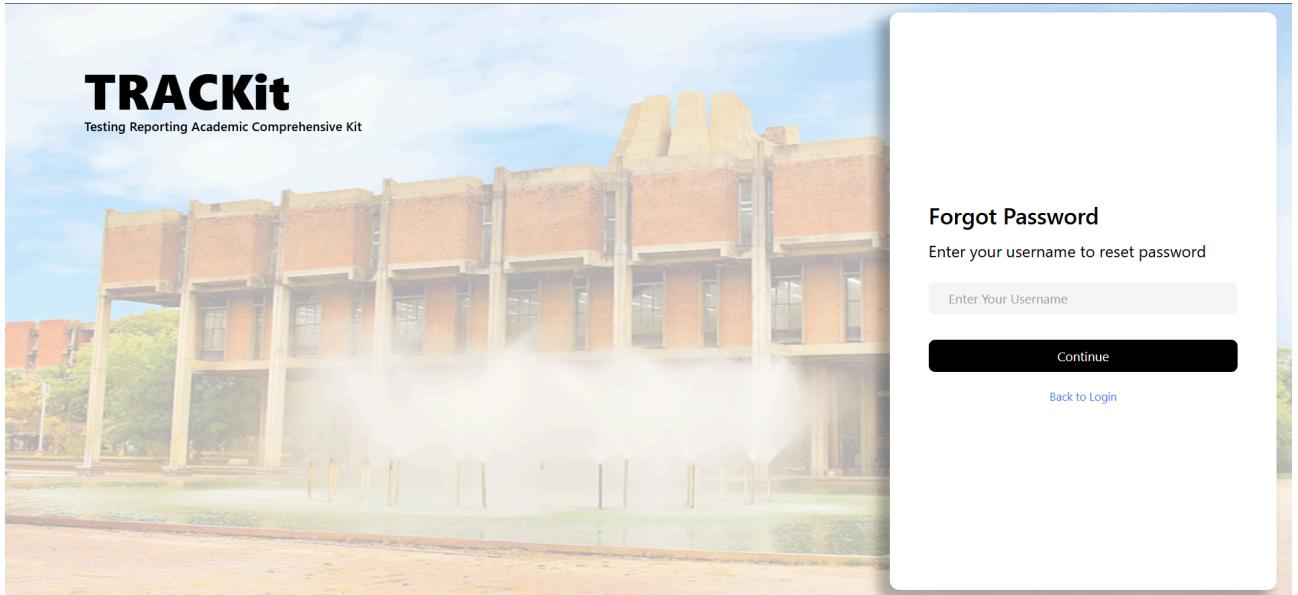
The response status is 200 OK, with a response time of 31 ms, a size of 1.33 KB, and a message body:

```

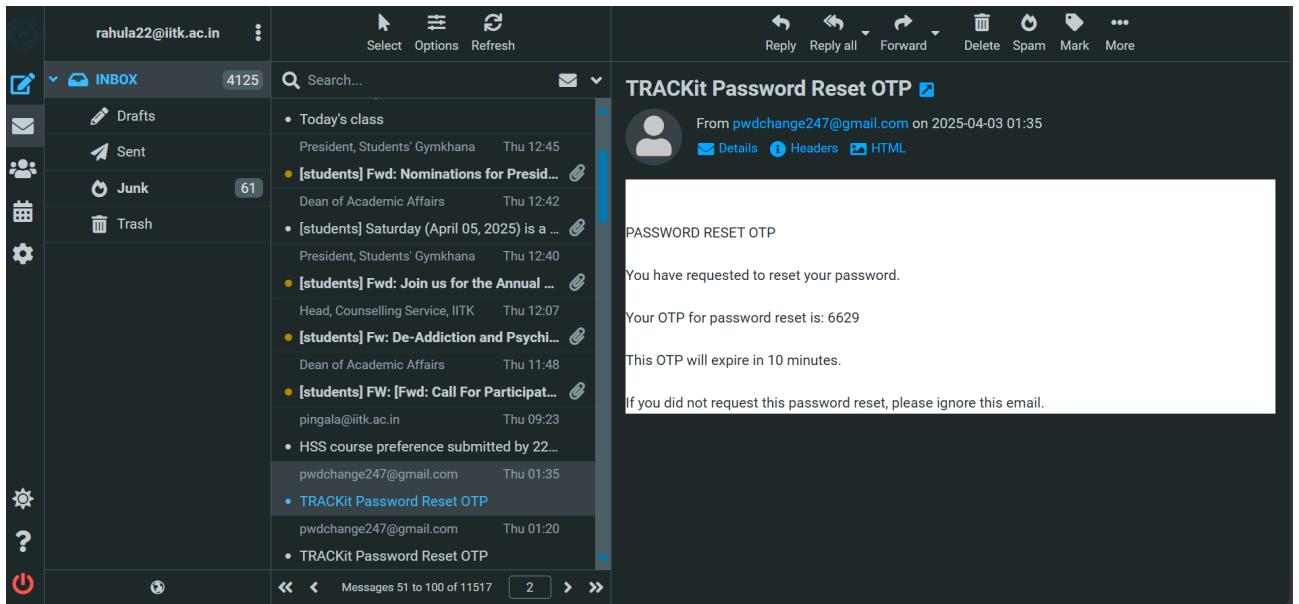
1 {
2   "success": true,
3   "userId": 13,
4   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
5         eyJpZCI6MTMsInB1cnBvc2UiOjEzNTE0MzYyMTI0NSwiZXhwIjoxNzQzNjIxODQ1fQ.
6         DymCPw1PCgLnGV_jTYK18Uh4yMsii3m_MVRv9N28Sv4",
7   "message": "OTP verified successfully"
8 }

```

On Clicking the forgot password link, the user is navigated to this page where he/she can enter the username and continue with the process to change password.



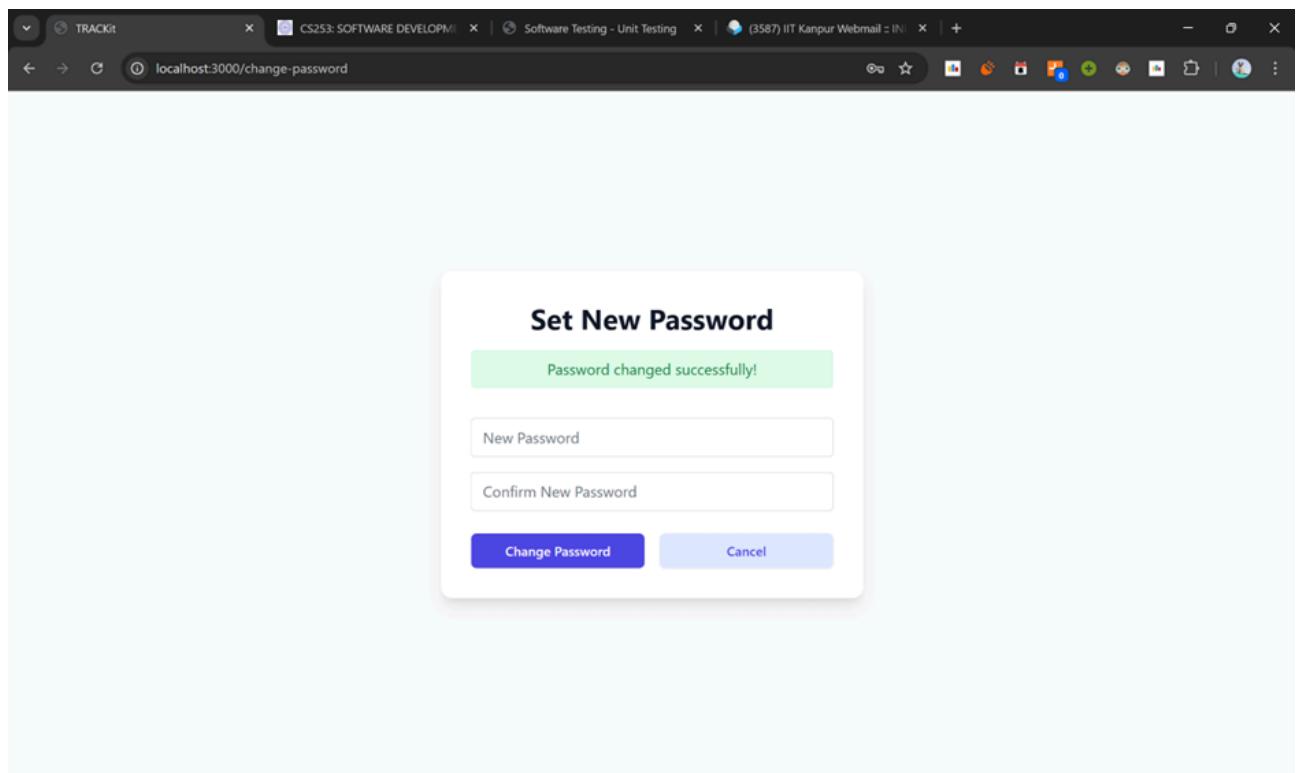
On verifying the username existence they will get the OTP on their email-Id to verify their identity.



OTP verification



Once the OTP gets verified, the user can change the password and get the message “Password changed successfully”.



Structural Coverage:

Functional coverage and **branch coverage** is used as the main criteria of coverage

3 Integration Testing

Integration testing involves testing whether the backend and frontend units interact in the right way and produce the desired behaviour. Both backend and frontend units have been individually tested by the unit testing team. We have performed integration testing as follows. We gave the input to the frontend and observed final changes that occur after the backend processes the request sent by the frontend. Correct final behaviour implies that the interaction between backend and frontend was alright. Apart from this, integration testing also involves the interaction between different components, redirection of pages and updation of pages whenever a change is made to the database from anywhere.

Admin : Add - Student

Module Details:

This unit focuses on the functionality for adding students to the system via two methods: manual entry and bulk upload using a CSV file. The manual entry form allows the admin to input individual student details, while the bulk upload feature enables adding multiple students at once by uploading a properly formatted CSV file. The unit also includes validation for input fields and file uploads, ensuring data integrity and error handling.

Key Features:

1. Manual entry of student details, including username, email, password, first name, last name, roll number, enrollment year, and major.
2. Bulk upload of student data using a CSV file with required fields.
3. Validation for required fields, file type, and file size (max 5MB).

Test Owner:

Sharique Ahmad, 221002

Test Date:

2 April 2025 - 3 April 2025

Test Result:

Manual Entry of User is successful as a New User named "Sharique Ahmad" has been added

Manage Users

Bob Williams	Role: student	View Details	Delete
Charlie Brown	Role: student	View Details	Delete
Diana Miller	Role: student	View Details	Delete
Ethan Davis	Role: student	View Details	Delete
Indranil Saha	Role: faculty	View Details	Delete
Sharique Ahmad	Role: student	View Details	Delete

Successfully Added 3 students using .csv file

Add Student

Successfully created 3 students

Add Students

Manual Entry Bulk Upload

Upload CSV File:
sample_students.csv

CSV file must include these columns:

- username: Unique username (required)
- firstName: Student's first name (required)
- lastName: Student's last name (required)
- email: Valid email address (required)
- rollNumber: Student's roll number (required)
- enrollmentYear: Year of enrollment (required)
- major: Student's major (required)
- password: Initial password (required)

Important: Make sure your CSV file uses commas (,) as separators and includes a header row.

[Upload and Add Students](#)

All the Added students were correctly shown in the Manage User.

The screenshot shows a list of users in the 'Manage Users' section of the TRACKit application. The users are listed in a grid format:

- Charlie Brown (Role: student) with 'View Details' and trash can buttons.
- Diana Miller (Role: student) with 'View Details' and trash can buttons.
- Ethan Davis (Role: student) with 'View Details' and trash can buttons.
- Sharique Ahmad (Role: student) with 'View Details' and trash can buttons.
- Aditya Gautam (Role: student) with 'View Details' and trash can buttons.
- Dhruv Varshney (Role: student) with 'View Details' and trash can buttons.

Admin : Add - Faculty

Module Details:

This unit focuses on the functionality for adding faculty members to the system. It provides two methods for adding faculty: manual entry and bulk upload using a CSV file. The manual entry form allows the admin to input individual faculty details, while the bulk upload feature enables adding multiple faculty members at once by uploading a properly formatted CSV file. The unit includes validation for input fields, file uploads, and error handling to ensure data integrity.

Key Features:

1. Manual entry of faculty details, including username, email, password, first name, last name, department, and position.
2. Bulk upload of faculty data using a CSV file with required fields.
3. Validation for required fields, file type, and file size (max 5MB).

Test Owner:

Sharique Ahmad, 221002

Test Date:

2 April 2025 - 3 April 2025

Test Result:

Successful addition of a Faculty through Manual Addition.

TRACKit ⌂

Manage Users

User Name	Role	Action
Alice Johnson	student	View Details Delete
Bob Williams	student	View Details Delete
Charlie Brown	student	View Details Delete
Diana Miller	student	View Details Delete
Ethan Davis	student	View Details Delete
Indranil Saha	faculty	View Details Delete

Successfully Added 3 Faculty through bulk addition using .csv file.

TRACKit ⌂

Add Faculty

Successfully created 3 faculty members

Add Faculty

Manual Entry [Bulk Upload](#)

Upload CSV File: [Choose a CSV file](#)

CSV file must include these columns:

- username: Unique username (required)
- firstName: Faculty's first name (required)
- lastName: Faculty's last name (required)
- email: Valid email address (required)
- department: Faculty's department (required)
- position: Faculty's position (required)
- password: Initial password (required)

Important: Make sure your CSV file uses commas (,) as separators and includes a header row.

[Upload and Add Faculty](#)

All the Added Faculty were Correctly shown in Manage Users with role “faculty”.

User Name	Role	Action
Sharique Ahmad	student	View Details Delete
Aditya Gautam	student	View Details Delete
Dhruv Varshney	student	View Details Delete
Shubham Sahay	faculty	View Details Delete
Abhishek Gupta	faculty	View Details Delete
Anonymous	faculty	View Details Delete

Admin : Create- Course

Module Details:

This unit focuses on the functionality for adding courses to the system. It provides two methods for adding courses: manual entry and bulk upload using a CSV file. The manual entry form allows the admin to input individual course details, while the bulk upload feature enables adding multiple courses at once by uploading a properly formatted CSV file. The unit includes validation for input fields, file uploads, and error handling to ensure data integrity.

Key Features:

1. Allows the admin to input individual course details
2. Enables the admin to upload a CSV file containing multiple course records
3. Validation for required fields, file type, and file size (max 5MB).

Test Owner:

Sharique Ahmad, 221002

Test Date:

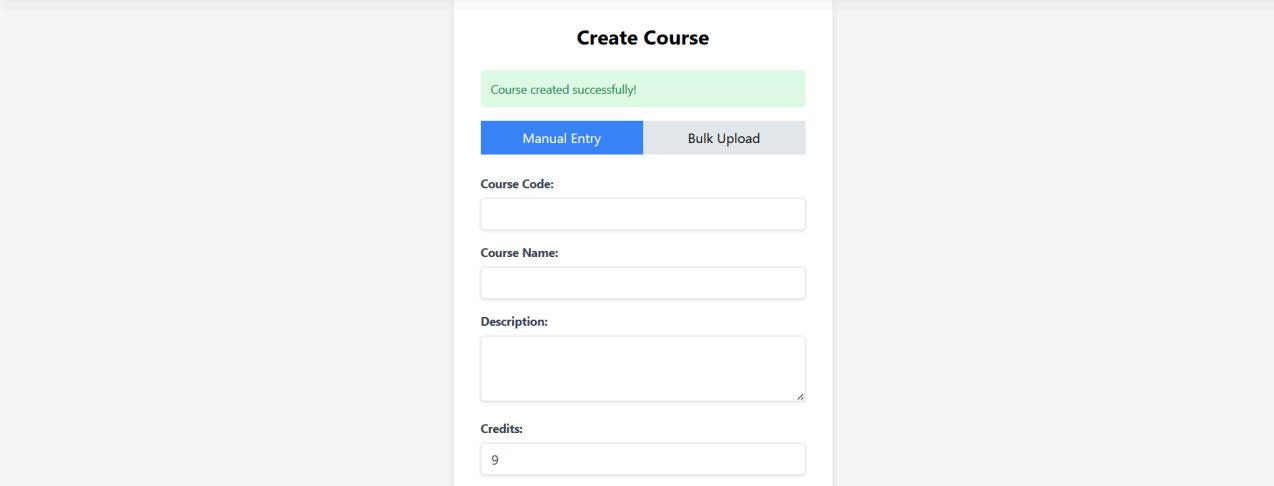
2 April 2025 - 3 April 2025

Test Result:

Successfully created a Course using manual Entry

TRACKit 

Create Course



The screenshot shows a 'Create Course' form. At the top, a green success message box displays 'Course created successfully!'. Below it are two buttons: 'Manual Entry' (blue) and 'Bulk Upload' (grey). The form fields are as follows:

- Course Code:**
- Course Name:**
- Description:**
- Credits:**

Added course is successfully shown in the course list.

TRACKit 

Manage Courses

Philosophy of Science	Course ID: 3 Code: PHI452	 
Digital Electronics	Course ID: 4 Code: EE210	 
Circuit Theory	Course ID: 5 Code: EE200	 
Principles of Economics	Course ID: 6 Code: ECO111	 
Microelectronics II	Course ID: 7 Code: EE311	 

Successfully created 5 courses in bulk using .csv file.

Create Course

Successfully created 5 courses

Manual Entry Bulk Upload

Upload CSV File:
Choose a CSV file

CSV file must include these columns:

- course code: Course code (required)
- course name: Course name (required)
- credits: Course credits (required, 2-14)
- semester: Fall/Spring/Summer (required)
- description: Course description (optional)

Important: Make sure your CSV file uses commas (,) as separators and includes a header row.

Upload and Create Courses

All the Added courses were correctly shown in the Manage Course List.

Course Name	Course ID	Code	Action
Intro to CS	8	CS101	
Circuit Analysis	9	EE202	
Thermodynamics	10	ME303	
Linear Algebra	11	MA404	
Quantum Mechanics	12	PH505	

Admin : Manage Users

Module Details:

This unit focuses on the "Manage Users" functionality within the Admin panel. It allows the Admin to perform CRUD (Create, Read, Update, Delete) operations on users, assign or remove courses for students and faculty, and manage role-specific fields such as department, position, roll number, and enrollment year. The unit also

includes search functionality for users and courses, ensuring seamless navigation and management.

Key Features:

1. Edit user details, including role-specific fields.
2. Assign and remove courses for students and faculty.
3. Delete users from the system.

Test Owner:

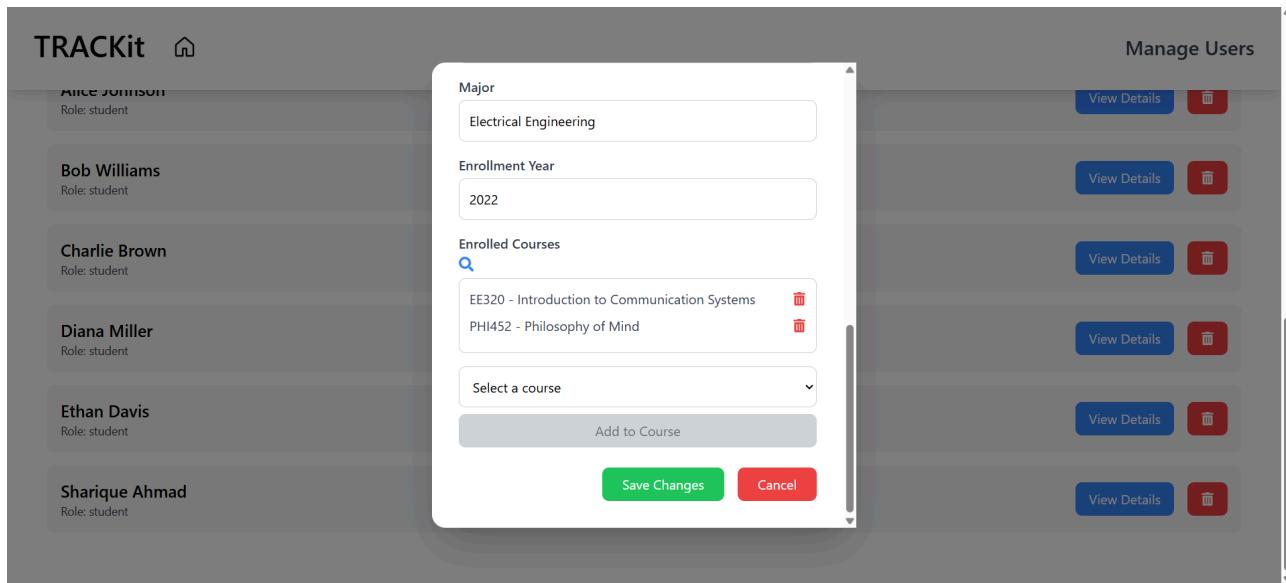
Sharique Ahmad, 221002

Test Date:

2 April 2025 - 3 April 2025

Test Result:

User added to some courses from Manage Users



Added courses were correctly reflected in the user's Dashboard.

The screenshot shows the TRACKit user interface. On the left, a sidebar menu includes 'My Courses' (highlighted in black), 'Performance', 'Profile', 'Contact Us', and 'Logout'. Below the menu is a URL: localhost:3000/PHI452/coursehome. The main content area displays two course cards: 'EE320 Faculty Introduction to...' and 'PHI452 Faculty Philosophy of ...'. A large calendar for April 2025 is centered, showing days from 30 to 03. Navigation buttons for 'This Month', 'Last Month', and 'Next Month' are at the top of the calendar, along with 'Refresh Calendar', 'Month', 'Week', and 'Day' buttons.

After Removing the User from a Course it is also correctly reflected in the User Dashboard.

The screenshot shows the TRACKit user interface. On the left, a sidebar lists users: Charlie Brown (Role: student), Diana Miller (Role: student), Ethan Davis (Role: student), Indranil Saha (Role: faculty), and Sharique Ahmad (Role: student). The main content area shows a user profile for 'Ronaldo' with fields for 'Major' (Electrical Engineering) and 'Enrollment Year' (2022). Under 'Enrolled Courses', there is a search bar ('Search Courses') and a list containing 'EE320 - Introduction to Communication Systems'. A modal window titled 'Manage Users' is open, listing several users with 'View Details' and 'Delete' buttons. At the bottom of the modal are 'Save Changes' and 'Cancel' buttons.

EE320
Faculty
Introduction L...

This Month Last Month Next Month

April 2025

Refresh Calendar Month Week Day

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	01	02	03	04	05
06	07	08	09	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	01	02	03

localhost:3000/EE320/coursehome

User details were updated from Manage User

User details updated successfully.

OK

Computer Science

Position

Professor

Teaching Courses

Search courses by name or code...

CS253 - Software Development

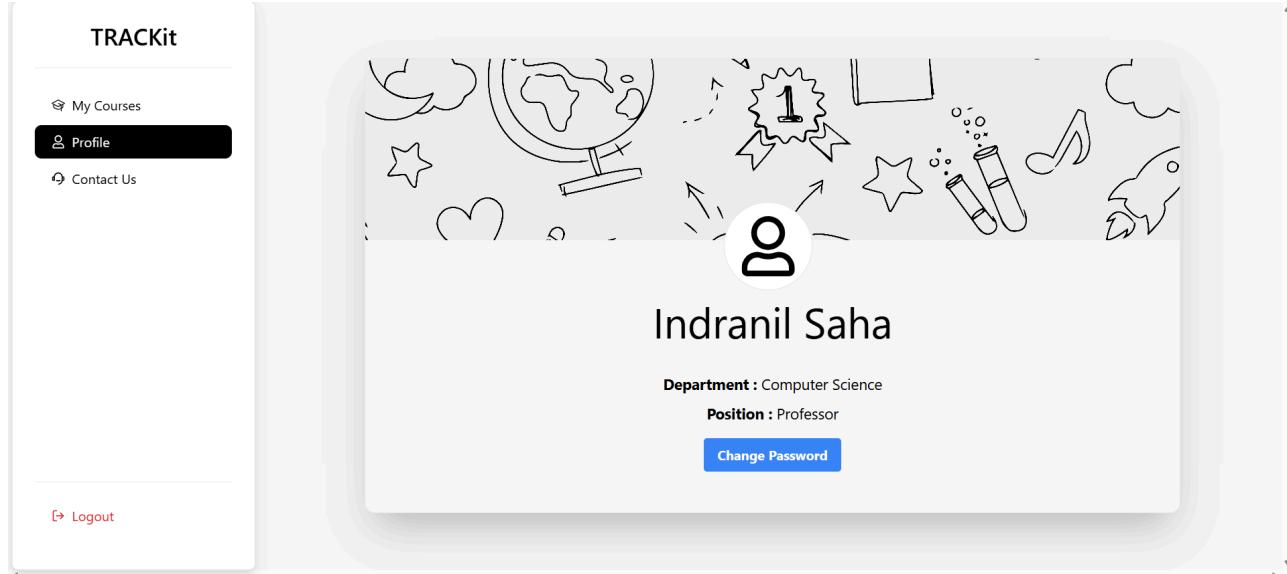
Select a course

Add to Course

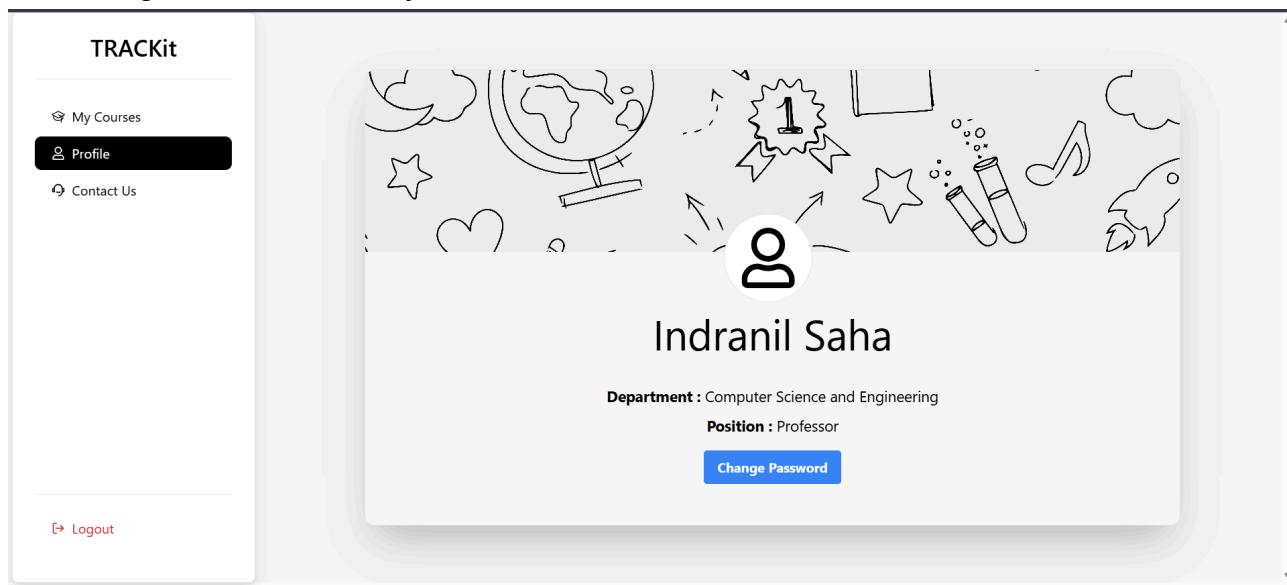
Save Changes Cancel

View Details Delete

The department was Changed.



The change was successfully reflected in the user's Profile tab.



User named "Sharique Ahmad" is being deleted.

Manage Users		
View Details		

After deleting it is removed from the list and all its information is removed from the database

Manage Users		
View Details		

Admin - Manage Courses

Module Details:

This unit focuses on the "Manage Courses" functionality within the Admin panel. It allows the Admin to perform CRUD (Create, Read, Update, Delete) operations on courses, manage enrolled students and assigned faculty, and handle bulk student

additions via CSV uploads. The unit also includes search functionality for courses, students, and faculty, ensuring efficient course management.

Key Features:

1. Edit course details, including code, name, description, credits, and semester.
2. Delete courses from the system.
3. View course details, including enrolled students and assigned faculty.
4. Add or remove students and faculty from courses.
5. Bulk adds students to courses using a CSV file.

Test Owner:

Sharique Ahmad: 221002

Test Date:

2 April 2025 - 3 April 2025

Test Results:

Before Updating the Course Details

The screenshot shows the course management interface for EE321. The left sidebar contains navigation links: Course Home (highlighted in black), Lectures, Announcements, Calendar, Result, Forum, and Logout. The main content area displays course details for "COMMUNICATION SYSTEMS" (EE321 • 4 Credits • Fall 2023 • student). A "New Events" section features a calendar for April 2025. The calendar grid shows dates from 30 March to 05 April. The week starting on Monday, April 01, is highlighted in light blue, while other weeks are greyed out. To the right, a "Your Attendance" box shows a 95% attendance rate with the message: "You have attended: 19/20 classes".

After updating the Course from the Manage Course all the changes were correctly reflected as the name and course code were changed.

EE320

INTRODUCTION TO COMMUNICATION SYSTEMS

EE320 • 11 Credits • Fall 2023 • student

New Events

Last Month	This Month	Next Month	April 2025					Month	Week	Day
Sun	Mon	Tue	Wed	Thu	Fri	Sat				
30	31	01	02	03	04	05				
06	07	08	09	10	11	12				
13	14	15	16	17	18	19				
20	21	22	23	24	25	26				
27	28	29	30	01	02	03				

Your Attendance

95%

You have attended:
19/20 classes

Bulk Addition of Students in a Course(EE320) is successful.

TRACKit

Manage Courses

Introduction to Communication Systems
Course ID: 1
Code: EE320

Software Development
Course ID: 2
Code: CS253

Philosophy of Mind
Course ID: 3
Code: PHI452

Digital Electronics
Course ID: 4
Code: EE210

Bulk students added successfully!

No students assigned to this course yet.

Faculty Assigned

Search faculty by name...

John Doe - Associate Professor (Electrical Engineering) X

Add Students in Bulk

Upload a CSV file with **userId** or **rollNumber** to add students to this course.

Choose File Book1.csv

Upload

The User's which are already added are enrolled in the course. Rest are discarded.

The screenshot shows the TRACKit application interface. At the top, there's a navigation bar with 'TRACKit' and a home icon. On the right, a 'Manage Courses' button is visible. Below the navigation, there's a search bar labeled 'Search courses by name...'. A list of courses is displayed:

- Introduction to Communication Systems**: Course ID: 1, Code: EE320
- Software Development**: Course ID: 2, Code: CS253
- Philosophy of Mind**: Course ID: 3, Code: PHI452
- Digital Electronics**: Course ID: 4, Code: EE210

 In the center, a modal window titled 'Course Details' is open. It has tabs for 'Edit Course' (disabled) and 'User Management'. The 'User Management' tab is selected, showing two sections:

- Students Enrolled**: A search bar and two entries: Alice Johnson - EE19B001 (Electrical Engineering) and Bob Williams - CS20B002 (Computer Science), each with a delete button.
- Faculty Assigned**: A search bar and one entry: John Doe - Associate Professor (Electrical Engineering), also with a delete button.

The screenshot shows a Microsoft Excel spreadsheet titled 'Book1'. The data is organized into columns:

	Userid	rollNumber
1		CS20B002
2	13	
3		EE19B001
4		100
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		

 The status bar at the bottom indicates 'Accessibility: Unavailable'.

Course is successfully deleted from the Database.

Introduction to Communication Systems
Course ID: 1
Code: EE320

Software Development
Course ID: 2
Code: CS253

Philosophy of Mind
Course ID: 3
Code: PHI452

Digital Electronics
Course ID: 4
Code: EE210

Circuit Theory

The Dashboard of the Users enrolled in the Deleted Course was also updated correctly.

EE320

Faculty

Introduction L...

This Month			April 2025			Next Month		
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Mon	Tue
30	31	01	02	03	04	05		
06	07	08	09	10	11	12		
13	14	15	16	17	18	19		
20	21	22	23	24	25	26		
27	28	29	30	01	02	03		

localhost:3000/EE320/coursehome

Search Functionality is also working correctly in Manage Courses.

The screenshot shows a software application window titled 'TRACKit'. In the top left, there's a logo and a navigation menu with 'Manage Courses' highlighted. Below the menu, a course card for 'Introduction to Communication Systems' (Course ID: 1, Code: EE320) is displayed, featuring a title, a small thumbnail, and two action buttons (edit and delete). The main area of the window is currently empty, indicating no announcements have been made.

Announcements

Module Details:

Involves the testing of the Announcements Database, unique to each course, where an announcement object, which has a heading along with a body, along with the corresponding frontend. Only Faculties associated with that course can create announcements in that course, but they can be viewed by all students and other Faculty members.

Test Owner:

Aditya Gautam, 220064

Test Date:

2 April 2025

Test Results:

First, a dummy announcement was created using a faculty account. Then it was checked using the student accounts associated with that course. Both Tests were fully successful.

The screenshots illustrate the user interface for managing announcements in the EE321 course.

Top Screenshot (Creating an Announcement):

- The left sidebar shows navigation links: Course Home, Lectures, Announcements (selected), Calendar, Result, and Forum.
- The main header is "ANNOUNCEMENTS" with the subtitle "EE321 • 4 Credits - Fall 2023".
- A blue button at the top right says "+ Add Announcement".
- A modal window titled "Create New Announcement" is open, containing fields for "Announcement Heading" (with placeholder "This is a test announcement.") and "Announcement Body" (with placeholder "This is a Faculty Typing in a Test Announcement."). There are "Cancel" and "Post Announcement" buttons at the bottom of the modal.
- The background shows four existing announcements: "Announcement 4 for EE321", "Announcement 2 for EE321", "Announcement 3 for EE321", and "Announcement 1 for EE321". Each announcement has edit, delete, and view icons.

Bottom Screenshot (List of Announcements):

- The left sidebar shows the same navigation links as the top screenshot.
- The main header is "ANNOUNCEMENTS" with the subtitle "EE321 • 4 Credits - Fall 2023".
- The "Create New Announcement" modal is closed.
- The list of announcements is displayed:
 - "This is a test announcement." (Posted by: John Doe (faculty) on 4/2/2025, 8:50:02 PM)
 - "Announcement 4 for EE321"
 - "Announcement 2 for EE321"
 - "Announcement 3 for EE321"
 - "Announcement 1 for EE321"

Following this, we tested the editing and deletion of the announcements by the faculty, and both were successful.

The screenshot shows the 'ANNOUNCEMENTS' section of the EE321 course page. The sidebar on the left includes links for Course Home, Lectures, Announcements (which is selected and highlighted in black), Calendar, Result, and Forum. The main content area displays a list of announcements:

- Edits! This is a test announcement.**
Posted by: John Doe (faculty1)
Created: 4/2/2025, 8:50:02 PM
Updated: 4/2/2025, 8:55:26 PM
- Announcement 4 for EE321
- Announcement 2 for EE321
- Announcement 3 for EE321
- Announcement 1 for EE321

A small cursor icon is visible near the bottom right of the main content area.

The screenshot shows the same EE321 course announcements page as above, but with a modal dialog box overlaid. The dialog is titled 'Confirm Delete' and contains the message: 'Are you sure you want to delete this announcement?'. It has two buttons at the bottom: 'Cancel' (gray) and 'Delete' (red). The background of the page is dimmed to indicate it is not active while the dialog is open.

The screenshot shows a web interface for a course named 'EE321'. On the left, there's a sidebar with links: Course Home, Lectures, Announcements (which is selected and highlighted in black), Calendar, Result, and Forum. Below the sidebar is a 'Logout' link. The main content area is titled 'ANNOUNCEMENTS' and shows four announcements for 'EE321': 'Announcement 4 for EE321', 'Announcement 2 for EE321', 'Announcement 3 for EE321', and 'Announcement 1 for EE321'. Each announcement has three small icons to its right: a checkmark, a red square, and a circular arrow. A blue button at the top right says '+ Add Announcement' and a user icon is next to it. At the bottom right of the main content area, there's a green success message box that says 'Announcement deleted successfully' with a close button.

Course Description Entry

Module Details:

Involves the testing of the Course Description Entry Database, unique to each course, where an announcement object, which has a heading along with a body, along with the corresponding frontend. Only faculties associated with that course can create course description entries, but they can be viewed by all students and other faculty members.

Test Owner:

Aditya Gautam, 220064

Test Date:

2 April 2025

Test Results:

First, a dummy Course Description Entry was created using a faculty account. Then it was checked using the student accounts associated with that course.

Both Tests were fully successful.

The image consists of two vertically stacked screenshots of a software application named TRACKit.

Top Screenshot (Faculty View):

- Header:** EE321, COMMUNICATION SYSTEMS, EE321 • 4 Credits • Fall 2023 • faculty.
- Left Sidebar:** Course Home (selected), Lectures, Announcements, Calendar, Result, Forum, Logout.
- Main Content:** A modal window titled "Create New Course Description" is open. It contains two text input fields:
 - Description Heading: This is a test entry.
 - Description Body: This is a test entry body.A "Cancel" button and a "Post Description" button are at the bottom of the modal.
- Background:** A calendar grid for the month showing days 06 to 19. Below the calendar, four topics are listed: Topic 1, Topic 2 for EE321, Topic 3 for EE321, and Topic 4 for EE321, each with edit, delete, and refresh icons.
- Bottom Right:** "+ Add Section" button.

Bottom Screenshot (Student View):

- Header:** EE321, COMMUNICATION SYSTEMS, EE321 • 4 Credits • Fall 2023 • student.
- Left Sidebar:** Course Home (selected), Lectures, Announcements, Calendar, Result, Forum, Logout.
- Main Content:** A calendar grid for the month showing days 13 to 03. A callout bubble "19/20 classes" points to the days 19-20.
- Below Calendar:** "Course Details" section with four entries: Topic 1 for EE321, Topic 2 for EE321, Topic 3 for EE321, and Topic 4 for EE321, each with a refresh icon.
- Bottom Left:** "Logout" button.

Following this, we tested the editing and deletion of the Course Description Entry by the faculty, and both were successful.

The screenshot shows the course management interface for EE321. On the left, a sidebar lists navigation options: Course Home (highlighted), Lectures, Announcements, Calendar, Result, and Forum. The main area displays the course title "COMMUNICATION SYSTEMS" and a subtitle "EE321 • 4 Credits • Fall 2023 • faculty". Below this is a weekly calendar grid from Monday, September 11, to Sunday, September 17. A modal window titled "Course Details" is open, listing course topics: "Topic 1 for EE321", "Topic 2 for EE321", "Topic 3 for EE321", and "Topic 4 for EE321". Each topic has edit, delete, and refresh icons. A message at the bottom of the modal says "Edits! This is a test entry." and "Edits! This is a test entry body.". A green success message at the bottom right of the modal says "Course description entry updated successfully".

This screenshot shows the same course management interface for EE321. The "Course Details" modal is still open, displaying the four topics listed above. A confirmation dialog box titled "Confirm Delete" is overlaid on the page, asking "Are you sure you want to delete this course description entry?". It contains "Cancel" and "Delete" buttons. The "Delete" button is highlighted with a red background and white text.

The screenshot shows the TRACKit software interface for the EE321 course. On the left, there's a sidebar with navigation links: Course Home (highlighted), Lectures, Announcements, Calendar, Result, and Forum. Below the sidebar is a red 'Logout' button. The main area has a title 'COMMUNICATION SYSTEMS' and a subtitle 'EE321 • 4 Credits - Fall 2023 • faculty'. It features a monthly calendar grid from June to August. Below the calendar, under 'Course Details', there are four sections: 'Topic 1 for EE321', 'Topic 2 for EE321', 'Topic 3 for EE321', and 'Topic 4 for EE321'. Each section has three small icons: a checkmark, a red square, and a circular arrow. A blue button '+ Add Section' is located above the first section. At the bottom right, a green box displays the message 'Course description entry deleted successfully'.

Events

Module Details:

Faculty can create Events in their course, which will be displayed to all students in that course. There are 3 places where the calendar is displayed, Main Dashboard (where events of all courses a user is enrolled in are shown, Course Home (a mini calendar where events of that course are shown) and Course Calendar (full-fledged calendar with course adding and removing support).

Test Owner:

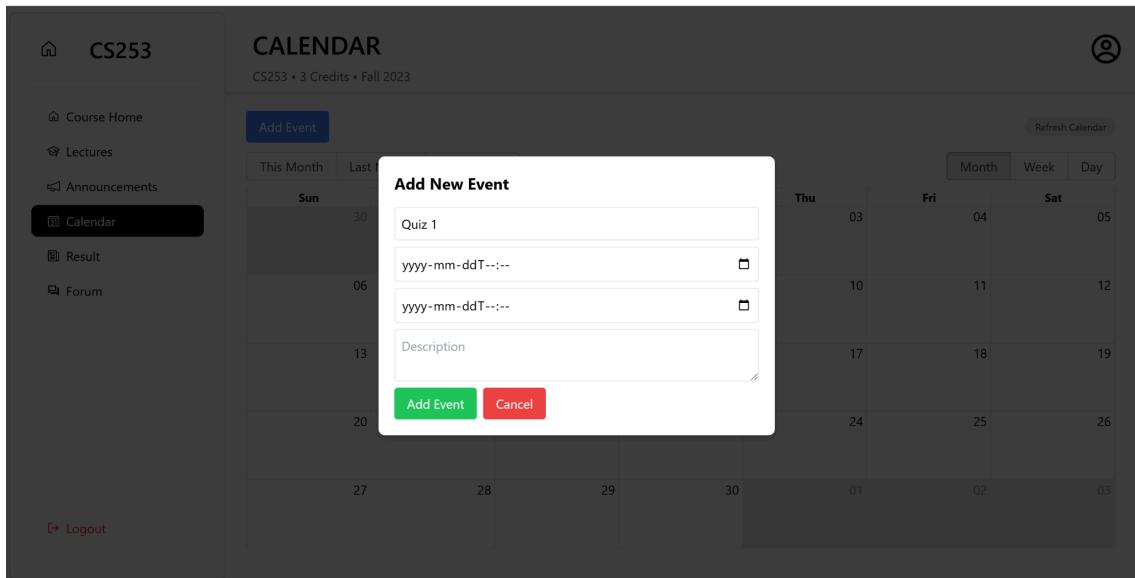
Aayush Singh, 220024

Test Date:

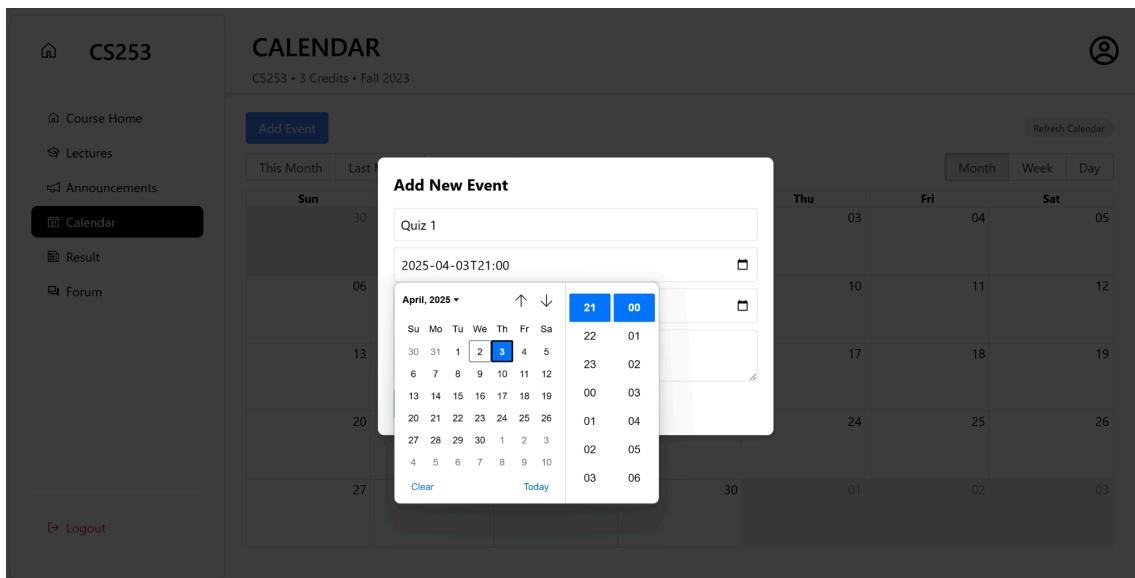
2 April 2025

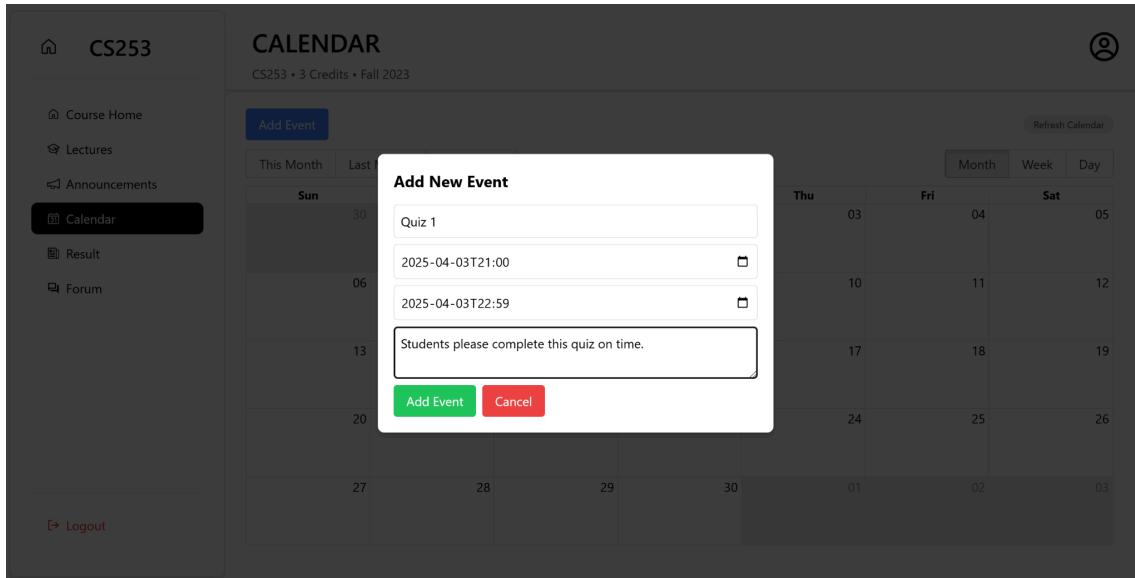
Test Results:

First, we added events in CS253 course from faculty1. We can put its Name, Start and End Time and date, and description

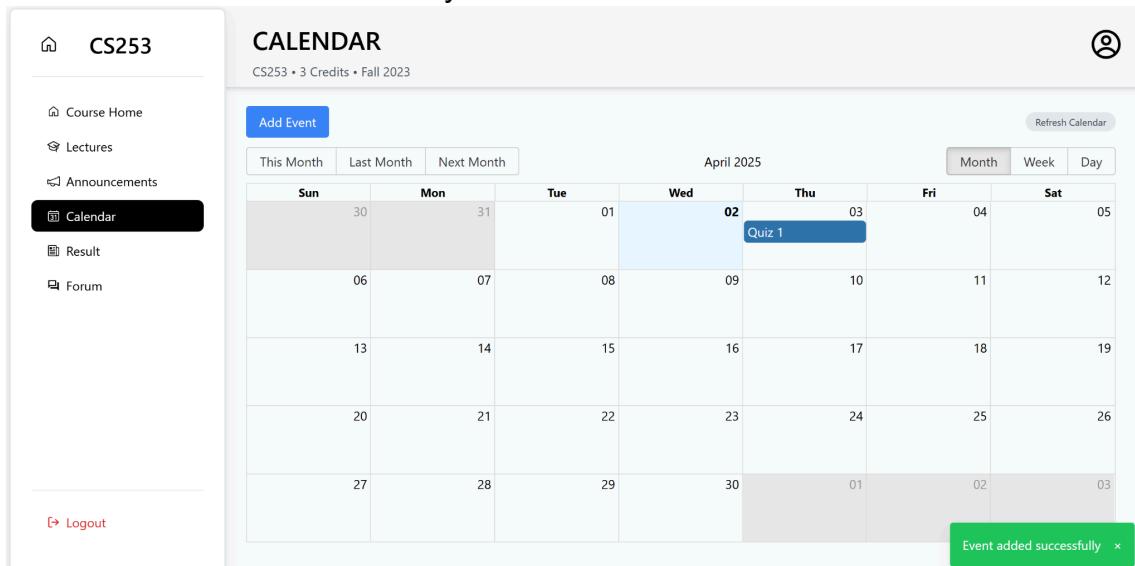


Time and Date selector UI.





Event added successfully.



What if we try to put end time before start time?
It shows a pop up which says “End time must be greater than start time”.

The screenshot shows the CS253 course calendar interface. On the left is a sidebar with links: Course Home, Lectures, Announcements, Calendar (which is selected and highlighted in black), Result, and Forum. The main area is titled "CALENDAR" and shows the month of April 2023. A modal window titled "Add New Event" is open, containing fields for "Event Name" (Reverse Event), "Start Date" (2025-04-02T21:35), and "End Date" (2025-04-01T21:35). Below these are "Description" and "Add Event" and "Cancel" buttons. A red error message at the bottom right of the calendar says "End time must be greater than start time".

Adding some more events In Course Calendar

The screenshot shows the CS253 course calendar for April 2025. The sidebar and basic layout are identical to the previous screenshot. The calendar grid now displays several events: "MIDSEM" on Wednesday, April 2, "Quiz 1" on Thursday, April 3, "PROJECT SUBMISS..." on Friday, April 24, and "PROJECT SUBMISS..." on Saturday, April 25. The days from April 1 to April 23 are currently empty.

In Course Home

The screenshot shows the Software Development module for the CS253 course. The left sidebar contains navigation links: Course Home, Lectures, Announcements, Calendar, Result, Forum, and Logout. The main area displays a calendar for April 2025 with the following events:

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	01	02 MIDSEM	03 Quiz 1	04	05
06	07	08	09	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24 PROJECT ...	25	26
27	28	29	30	01	02	03

In Main Dashboard (Color Coded for different courses)

The screenshot shows the main dashboard with the TRACKit logo. The left sidebar includes My Courses, Profile, Contact Us, and Logout. The right side features two course boxes: EE321 (Faculty Communication ...) and CS253 (Faculty Software Devel...). Below these is a color-coded calendar for April 2025:

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	01	02 Quiz EE321	03 Quiz 1 CS253	04	05
06	07	08	09	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24 PROJECT SUBMISS CS253	25	26
27	28	29	30	01	02	03

Weekly View

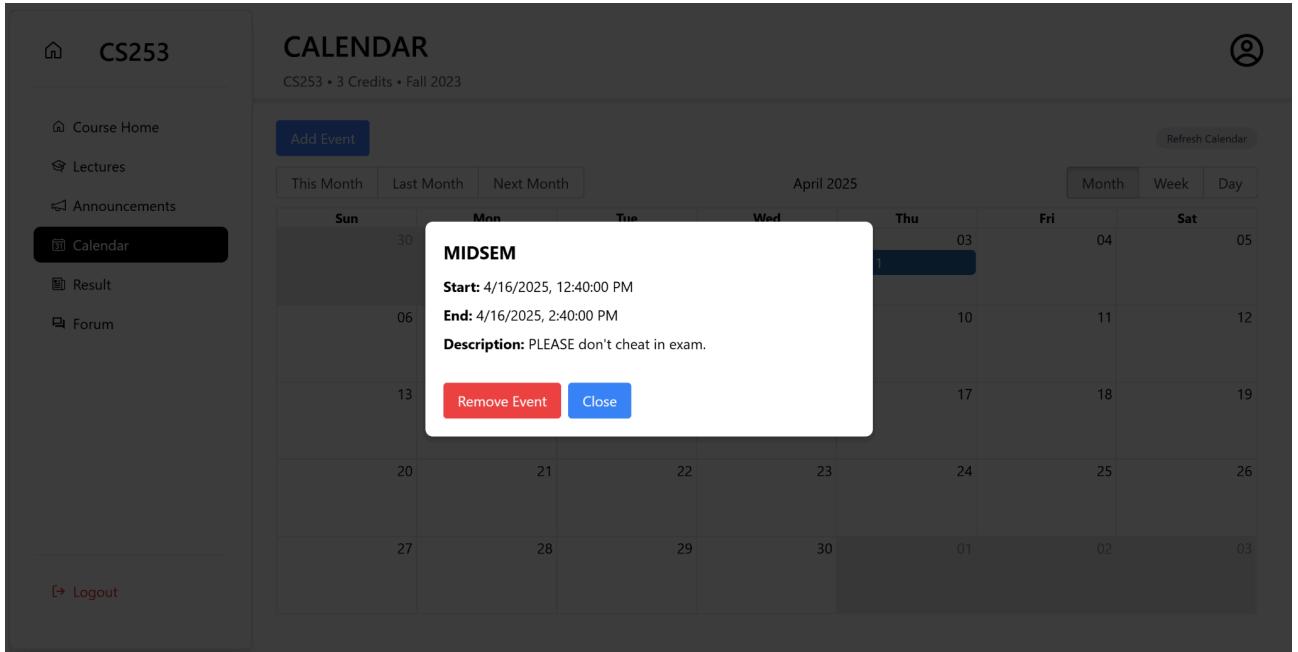
Day View

The screenshot shows the TRACKit application's Day View calendar. On the left, there is a sidebar with the TRACKit logo at the top, followed by links for 'My Courses' (highlighted in black), 'Profile', and 'Contact Us'. At the bottom of the sidebar is a red 'Logout' button. The main area displays a 24-hour grid from 6:00 AM to 2:00 PM. Two course blocks are visible at the top: 'EE321 Faculty Communication ...' and 'CS253 Faculty Software Devel...'. Below the grid, there are buttons for 'This Day', 'Last Day', and 'Next Day', and a date indicator 'Thursday Apr 24'. In the top right corner of the main area, there is a 'Refresh Calendar' button and tabs for 'Month', 'Week', and 'Day' (which is selected). A green event bar labeled 'PROJECT SUBMISSION' spans from 10:45 AM to 11:45 AM on Thursday, April 24.

Clicking on an event

This screenshot shows the same Day View calendar as above, but with a modal window overlaid. The modal is titled 'Quiz 1' and contains information about a quiz: 'CS253 Software Development', 'Start: 4/3/2025, 9:00:00 PM', 'End: 4/3/2025, 10:59:00 PM', and 'Description: Students please complete this quiz on time.' At the bottom of the modal is a blue 'Close' button. The background calendar grid shows dates from Sunday, April 29, to Saturday, May 5, with various events like 'Quiz 1' and 'PROJECT SUBMISSION' marked. The 'Day' tab is still selected in the top right.

Removing an event (in course calendar page)



Testing whether the events are seen from a POV of student

The screenshot shows the student dashboard for TRACKit. On the left, there's a sidebar with links for My Courses (selected), Performance, Profile, Contact Us, and Logout. The main area displays a list of courses with their respective faculty and descriptions:

- CS253 Faculty Software Dev...
- EE210 Faculty Digital Electr...
- EE200 Faculty Circuit Theory
- ECO111 Faculty Principles of ...
- PHI452 Faculty Philosophy of ...

Below the courses is a calendar for April 2025. Several events are listed on the calendar:

- Wednesday, April 2: Quiz (EE210)
- Thursday, April 3: Quiz 1 (EE210)
- Wednesday, April 20: Assignment Submission (PHI452)
- Wednesday, April 24: PROJECT SUBMISS (EE210)

Comments:

No critical bugs were found. But improvements can be done by adding repeating events.

Results

Module Details

The module encompasses testing procedures for the result page which involve creation, modification, deletion of students' results. It also tests how the views of both students and faculty get updated once changes are made by the faculty. We also check the download button for faculty which fetch data and convert it in excel format. Additionally automatic calculation of mean, median, standard deviation is also checked.

Test owner

Dhruv Rai (220365)

Test date

02/04/25 - 05/04/25

Test results

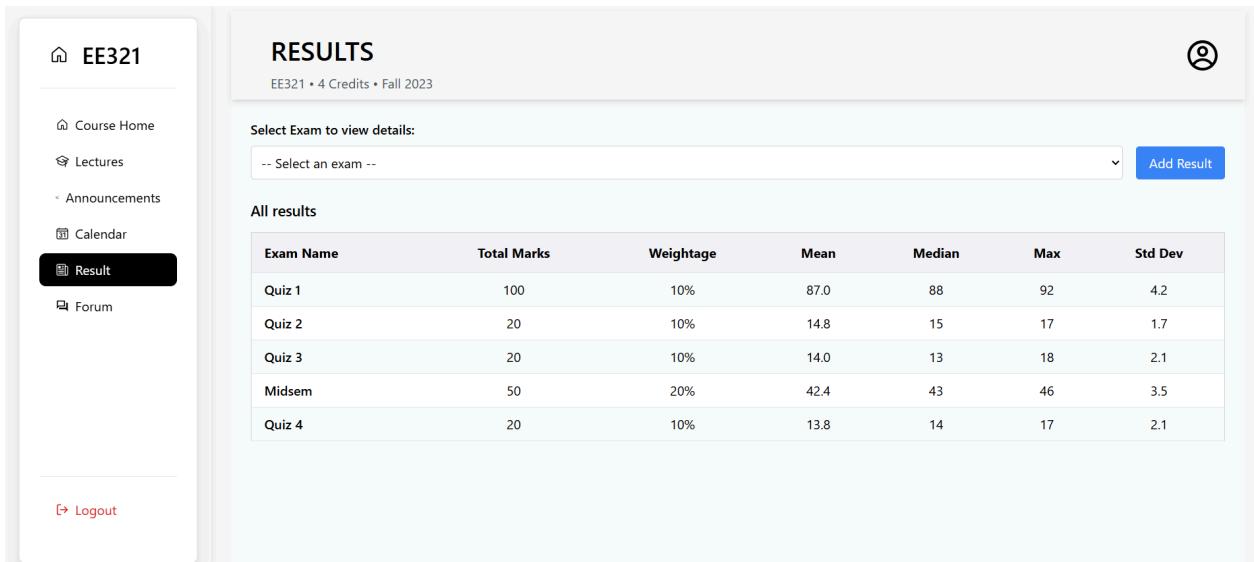
In this test we validated the successful integration of backend and frontend components of getAllResults, getAllExamsWithDetails, PublishExamResults, ModifyExamResults, getResultsForstudent, DeleteExamAndAssociatedResults and getEnrolledStudentsList. Apart from this, we successfully tested that the result table and the graph in both the student view and faculty view page represents the correct results for all students after all kinds of modifications. Throughout the testing process, we also scrutinized the redirection of users to the correct pages upon the completion of these actions.

Details : To verify whether all the results associated with a particular student enrolled in a course get fetched correctly or not in the student view.



Result : Displaying all the results with correct details for the student logged in and selected a particular course.

Details : To verify whether all the exam names for which result has been published are correctly shown or not in the faculty view after clicking on the result tab from the sidebar and also check the working of the dropdown menu.



	Quiz 3	Midsem	Quiz 4	Std Dev
Count	20	50	20	4.2
Percentage	10%	20%	10%	1.7
Avg	14.0	42.4	13.8	2.1
Min	13	43	14	3.5
Max	18	46	17	2.1

Result : Displaying all the exams with correct details once faculty get logged in and select a particular course and then click on the result tab. The dropdown menu is also working fine with a dynamic list of exams.

Details : To verify whether all the automatic calculations of mean, median, maximum, standard deviation are correct or not. Additionally also check the class performance graph.

Total Marks	Weightage	Mean	Median
20	10%	13.8	14

Maximum	Standard Deviation
17	2.1

Class Performance

Score Distribution

Score Range	Number of Students
0-20%	0
20-40%	1
40-60%	3
60-80%	3
80-100%	1

Result : All the calculations are done correctly for the new data and the graph is also correct.

Details : To verify whether the add result option is working fine and does this get updated in real time for both student and faculty of the course. Additionally check if all the edge case checks are working fine or not.

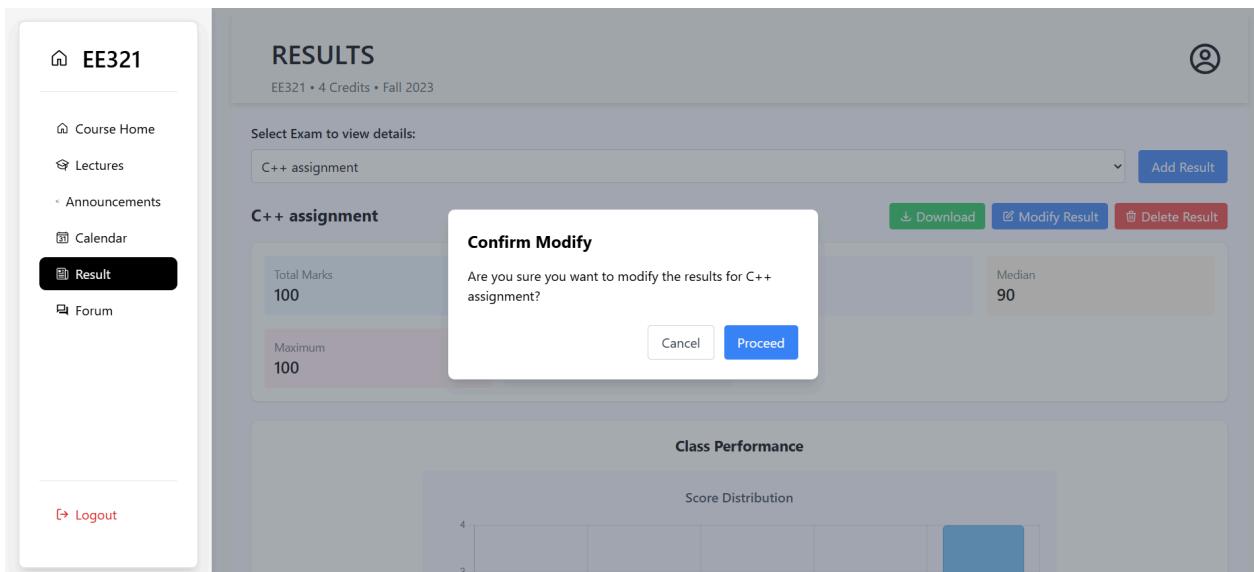
Roll Number	Name	Marks
EE19B001	Alice Johnson	90
CS20B002	Bob Williams	80
EE21B003	Charlie Brown	99
CS19B004	Diana Miller	100
EE20B005	Ethan Davis	65

	Quiz 3	Midsem	Quiz 4	C++ assignment		
Quiz 3	20	10%	14.0	13	18	2.1
Midsem	50	20%	42.4	43	46	3.5
Quiz 4	20	10%	13.8	14	17	2.1
C++ assignment	100	5%	86.8	90	100	13.1



Result : The add result option worked fine and correctly updated the database. Once the exam result is added it correctly refreshes the screen for the faculty loading the details of the added exam and also loads the same for all the enrolled students.

Details : To verify whether the modify result option is working fine and does this get updated in real time for both student and faculty of the course. Additionally check if all the edge case checks are working fine or not.



The screenshot shows the 'Modify Exam Results' page for EE321. On the left sidebar, 'Result' is selected. The main area displays a table of student marks for a C++ assignment. Alice Johnson has a mark of 65, which is being edited. Other students listed are Bob Williams (80), Charlie Brown (99), Diana Miller (100), and Ethan Davis (65). Buttons for 'Cancel' and 'Update Results' are at the bottom right.

Roll Number	Name	Marks
EE19B001	Alice Johnson	65
CS20B002	Bob Williams	80
EE21B003	Charlie Brown	99
CS19B004	Diana Miller	100
EE20B005	Ethan Davis	65



Result : The modify result option worked fine and correctly updated the database. Once the exam result is updated it correctly refreshes the screen for the faculty loading the details of the updated exam and also loads the same for all the enrolled students.

Details : To verify whether the download result option is working fine and does it provide data in expected excel format or not.

The screenshot shows the TRACKit software interface. On the left, a sidebar for 'EE321' contains links for Course Home, Lectures, Announcements, Calendar, Result (which is selected), and Forum. Below the sidebar is a 'Logout' button. The main area is titled 'RESULTS' and shows details for 'EE321 • 4 Credits • Fall 2023'. A dropdown menu 'Select Exam to view details:' shows 'C++ assignment'. Below this, a table provides summary statistics for the 'C++ assignment': Total Marks (100), Weightage (10%), Mean (81.8), Median (80), Maximum (100), and Standard Deviation (15.5). At the bottom, a chart titled 'Score Distribution' shows a blue bar representing the distribution of marks. Buttons for 'Download', 'Modify Result', and 'Delete Result' are also present.

The screenshot shows an Excel spreadsheet titled 'K23' with data for the 'C++ assignment'. The data includes:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Exam Name:	C++ assignment																	
2	Total Marks:	100																	
3	Weightage:	10%																	
4	Mean:	81.8																	
5	Median:	80																	
6	Maximum:	100																	
7	Standard Deviation:	15.5																	
9	Roll Number	Name	Marks																
10	EE19B001	Alice Johnson	65																
11	CS20B002	Bob Williams	80																
12	EE21B003	Charlie Brown	99																
13	CS19B004	Diana Miller	100																
14	EE20B005	Ethan Davis	65																
15																			
16																			
17																			
18																			
19																			
20																			
21																			
22																			
23																			
24																			
25																			
26																			

Result : The download button worked fine as expected and it correctly exported data in expected format.

Details : To verify whether the delete result option is working fine and does this get updated in real time for both student and faculty of the course.

The screenshot shows the EE321 course results page. On the left, a sidebar menu includes Course Home, Lectures, Announcements, Calendar, Result (which is selected and highlighted in black), and Forum. Below the menu is a Logout link. The main content area is titled "RESULTS" and displays "EE321 • 4 Credits • Fall 2023". A dropdown menu labeled "Select Exam to view details:" shows "Quiz 1". To the right of the dropdown are three buttons: "Download", "Modify Result", and "Delete Result". A modal dialog box titled "Confirm Delete" asks, "Are you sure you want to delete the results for Quiz 1? This action cannot be undone." It contains "Cancel" and "Delete" buttons. In the background, there's a "Class Performance" section with a "Score Distribution" chart and some summary statistics: Total Marks 100, Maximum 92, Median 88.

This screenshot shows the EE321 course results page with the "Result" button selected in the sidebar. The main content area is titled "RESULTS" and displays "EE321 • 4 Credits • Fall 2023". A dropdown menu labeled "Select Exam to view details:" shows "-- Select an exam --". To the right of the dropdown is a "Add Result" button. Below the dropdown, a table titled "All results" lists the following data:

Exam Name	Total Marks	Weightage	Mean	Median	Max	Std Dev
Quiz 2	20	10%	14.8	15	17	1.7
Quiz 3	20	10%	14.0	13	18	2.1
Midsem	50	20%	42.4	43	46	3.5
Quiz 4	20	10%	13.8	14	17	2.1
C++ assignment	100	10%	81.8	80	100	15.5

Below the table is a "Score Distribution" chart. At the bottom of the page is a "Logout" link.



Result : The delete result option worked as expected and correctly updated the database. Once the exam result is deleted the module correctly refreshes the screen for the faculty removing the details of the deleted exam and also updates the view for all the enrolled students.

Additional Comments : NA

Lectures

Module Details:

Faculty members can design lectures for their courses, which will be accessible to all students enrolled in that course. The lecture page is structured as follows:

1. Modules: Professors can create modules, such as weekly units, to organize the course content.
2. Sections within Modules: Each module can be divided into multiple sections to further categorize the content.
3. Lectures within Sections: Professors can upload multiple lectures under each section, providing detailed materials for students.

Adding a Lecture:

Professors can create new lectures by filling out the required fields:

1. Lecture Title: Specify the name of the lecture.
2. Lecture Description: Provide a brief or detailed explanation of the lecture content.

3. Upload Files: Attach relevant files (e.g., PDFs, slides, or other resources).
4. YouTube Link: Include a link to a YouTube video for additional learning material.
5. Click Submit to save the lecture or Cancel to discard changes.

Editing a Lecture:

Professors can modify an existing lecture by:

1. Navigating to the lecture they want to edit.
2. Updating fields such as the title, description, uploaded files, or YouTube link.
3. Submitting changes to ensure the updated lecture is saved.

Deleting a Lecture:

If a lecture is no longer needed, professors can delete it:

1. Locate the lecture in the section.
2. Use the delete option (typically represented by a trash icon or button).
3. Confirm the deletion to permanently remove the lecture.

Managing Modules:

Professors can also manage modules effectively:

1. Creating Modules: Add new modules (e.g., "Week 1," "Week 2") to organize course content systematically.

Test Owner:

Abhijeet Agarwal, 210025

Test Date:

2 April 2025

Test Results:

- When the "Add Module" button is clicked, and a module name is entered, the module is successfully created upon clicking "Save Module."

The screenshot shows the 'LECTURES' page for the course EE321. On the left sidebar, under the 'Lectures' section, there is a 'Logout' link. The main area displays a list of weeks from Week 1 to Week 6. An 'Add Module' button is located at the top right of the list. A modal window titled 'Add New Module' is open, containing a text input field for 'Module Name' with the placeholder 'Week n'. Below the input field are 'Cancel' and 'Save Module' buttons.

- When the "Add Section" button is clicked, and a section name is entered, the section is successfully created upon clicking "Save Section."

The screenshot shows the 'LECTURES' page for the course EE321. The left sidebar includes a 'Logout' link. The main area lists weeks from Week 5 to Week n. An 'Add Section' button is positioned at the top right. A modal window titled 'Add Section' is open, featuring a text input field for 'Section Name' with the placeholder 'New section'. It also contains 'Cancel' and 'Save Section' buttons.

- The section name is updated successfully after modifying it and clicking "Save Changes."

The screenshot shows the 'LECTURES' page for course 'EE321'. On the left, there's a sidebar with links: Course Home, Lectures (which is selected and highlighted in black), Announcements, Calendar, Result, Forum, and Logout. The main area displays a list of weeks from Week 6 to Week n. Each week entry includes an 'Add Section' button. A modal dialog box is overlaid on the page, titled 'Edit Section'. It contains two input fields: 'Module Name' with the value 'Week n' and 'Section Name' with the value 'New section'. At the bottom of the modal are two buttons: 'Cancel' and 'Save Changes' (which is highlighted in green).

- When the "Add Lecture" button is clicked, and a lecture is added with the uploaded files and other details, the lecture is successfully created upon clicking "Submit."

The screenshot shows the EE321 software interface. On the left is a sidebar with navigation links: Course Home, Lectures (which is selected and highlighted in black), Announcements, Calendar, Result, Forum, and Logout. The main area is titled 'LECTURES' and displays a list of weeks from Week 6 to Week n. A modal window titled 'Add Lecture' is open, containing fields for 'New Lecture' (text input), 'New description' (text input), 'Upload Files' (file input showing 'ESO207_MS_Q4.pdf'), and a URL input field ('Selected files: ESO207_MS_Q4.pdf' and 'https://www.youtube.com/watch?v=yFzYohKd-gQ'). At the bottom of the modal are 'Cancel' and 'Submit' buttons.

- Uploaded files are displayed with a "Download" button, allowing users to download them.

The screenshot shows the EE321 software interface. The sidebar and main layout are identical to the previous screenshot. A new modal window titled 'Lecture Supplements' is open, showing a single item: 'ESO207_MS_Q4-1743613156352-577413473.pdf'. Below this, the list of weeks from Week 8 to Week n is visible, each with an 'Add Section' button to its right. At the bottom of the page are several icons: a red square with a white document, a blue square with a white downward arrow, a green square with a white checkmark, and a grey square with a white minus sign.

- The YouTube video link opens in a new tab when clicked on the lecture name.

The screenshot shows the 'LECTURES' page for course EE321. The left sidebar includes links for Course Home, Lectures (which is selected), Announcements, Calendar, Result, and Forum. The main content area displays a list of weeks from Week 7 to Week n. Each week entry has a dropdown arrow icon. Below the weeks, there is a 'New section' button and a 'New lecture' input field with a file upload icon.

- Lectures are updated successfully with new resources and a modified name when edited by clicking on the edit button and then finally submitting the updated details with the submit button.

The screenshot shows a modal dialog for adding a new lecture. The dialog has fields for 'Module' (set to 'Week n'), 'Section' (set to 'New Section'), 'New Lecture' (text input), 'New description' (text input), and 'Upload Files' (file input set to 'Choose Files No file chosen'). Below these fields is a URL input field containing 'https://www.youtube.com/watch?v=AwZ8PtoqCeU'. At the bottom of the dialog are 'Cancel' and 'Submit' buttons, along with a row of icons for file operations.

- Upon clicking "Delete Lecture," a confirmation dialog box appears. When confirmed, the lecture is deleted successfully.

The screenshot shows the 'LECTURES' page for course EE321. The left sidebar includes links for Course Home, Lectures (which is selected and highlighted in black), Announcements, Calendar, Result, and Forum. The right main area displays a list of weeks from Week 7 to Week n. A 'New Section' button is visible. A 'Confirm Delete' dialog box is centered over the list, asking 'Are you sure you want to delete this lecture?' with 'Cancel' and 'Delete' buttons.

- When deleting a section, a warning dialog box appears stating that all lectures within the section will be deleted. Upon confirmation, the section and its associated lectures are removed successfully.

The screenshot shows the 'LECTURES' page for course EE321. The left sidebar includes links for Course Home, Lectures (selected and highlighted in black), Announcements, Calendar, Result, and Forum. The right main area displays a list of weeks from Week 7 to Week n. A 'New Section' button is visible. A 'Confirm Section Deletion' dialog box is centered over the list, asking 'Are you sure you want to delete the section "New Section"?'. Below it, a red warning message says 'Warning: This will delete all lectures in this section.' with 'Cancel' and 'Delete' buttons.

Structural Coverage:

The unit is working well and fulfils 100% of the requirements of the SRS.

Additional Comments:

A visual bug was present in the add lecture and edit lecture dialogue box. It is fixed.

Forgot Password

Module Details:

Users can reset their password using the "Forgot Password" feature, which involves OTP verification via email. The process consists of three steps: First, the user enters their registered username, and if it exists in the system, an OTP is sent to the registered email address of the user. Next, the user inputs the received OTP, and the system verifies its correctness. If the OTP is valid, the user is allowed to set a new password. Finally, after confirming the new password, the system securely updates it in the database, completing the reset process.

Test Owner :

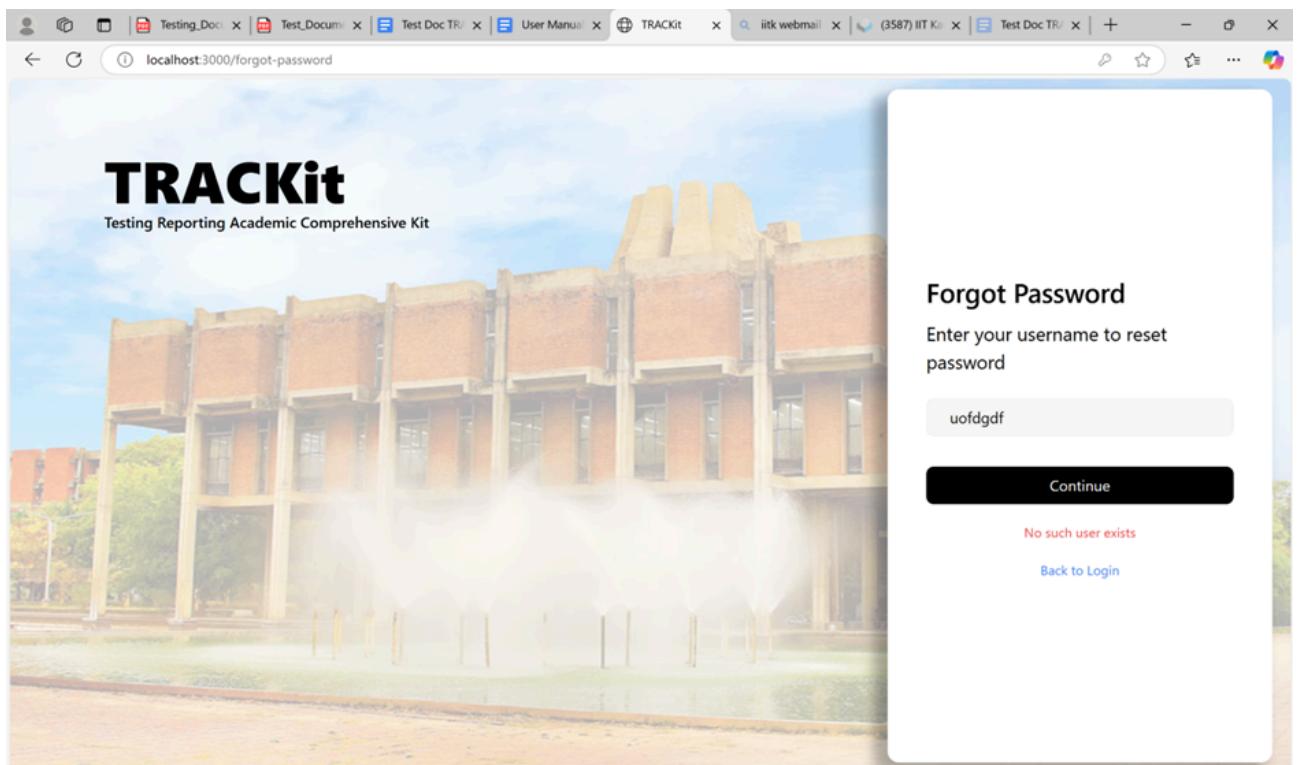
Akash Verma , 220097

Test Date :

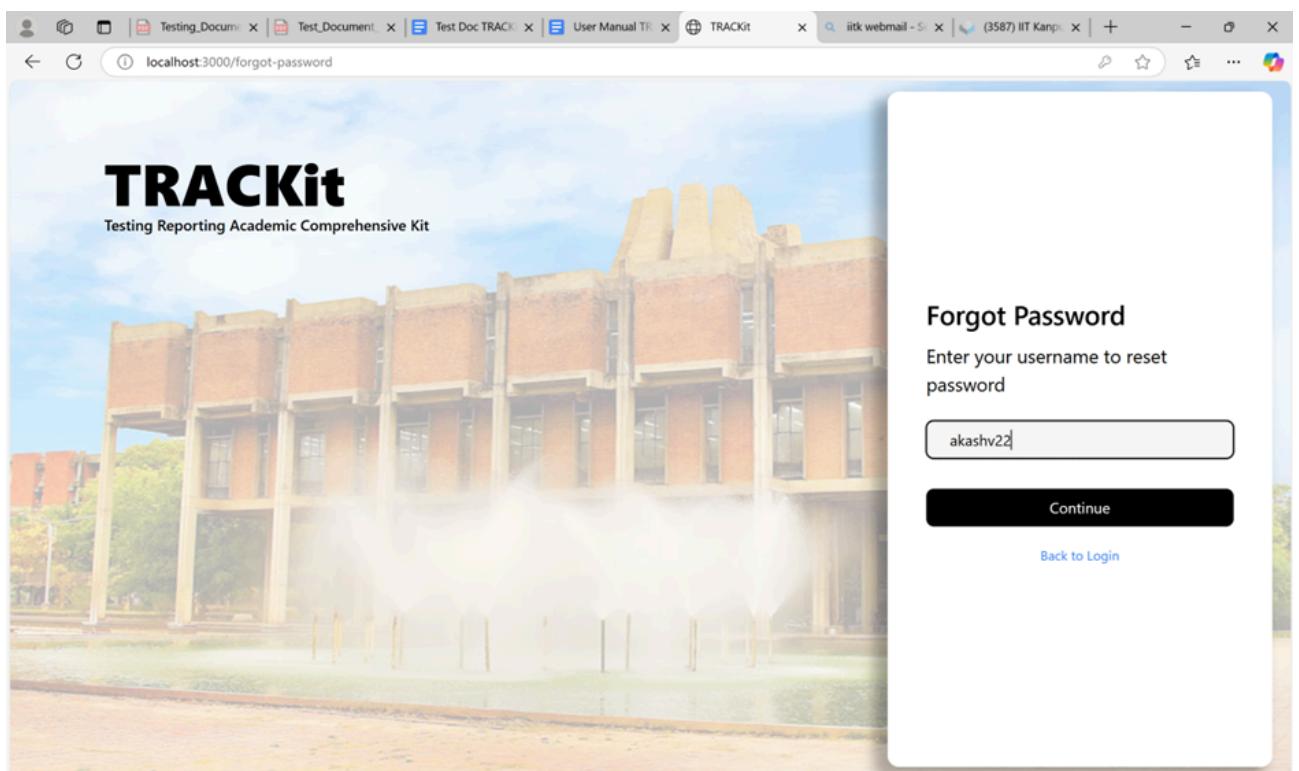
2 April 2025

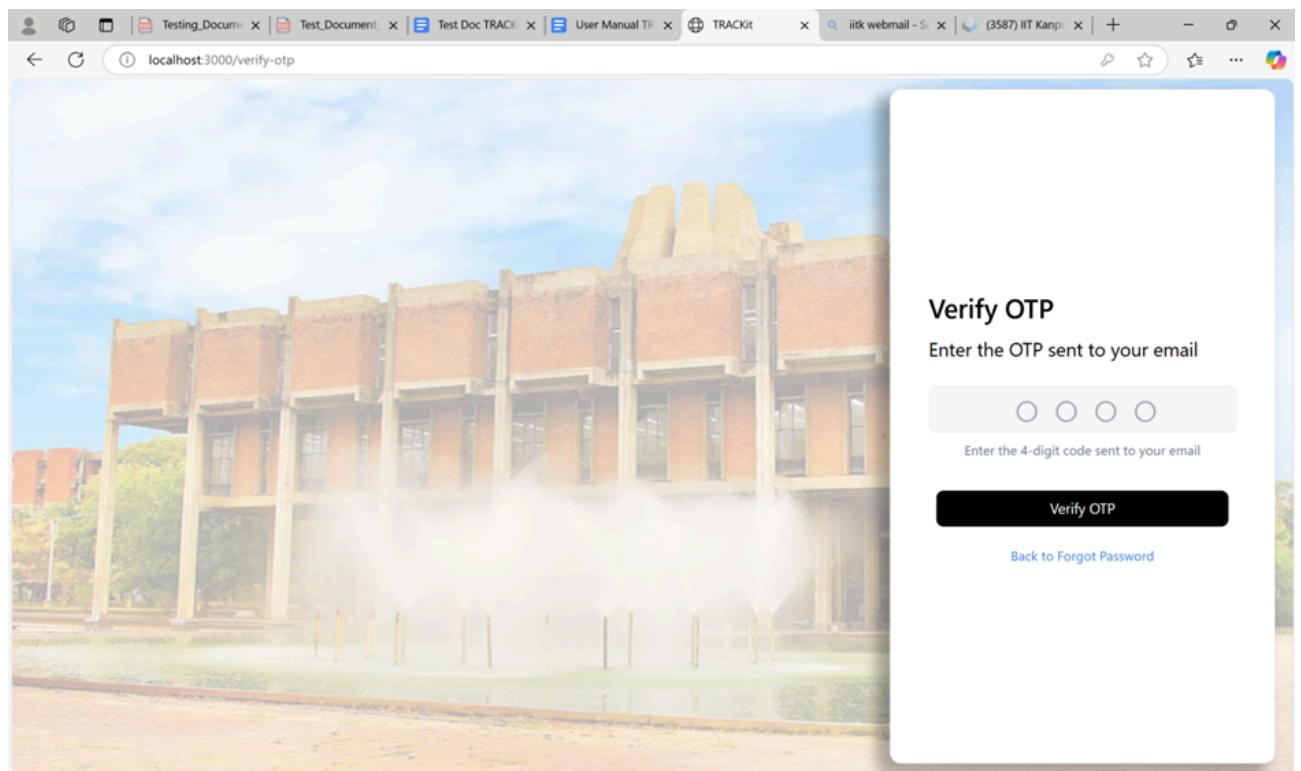
Test Results:

- On clicking the forget password button on the login page , the page for entering username correctly opens.



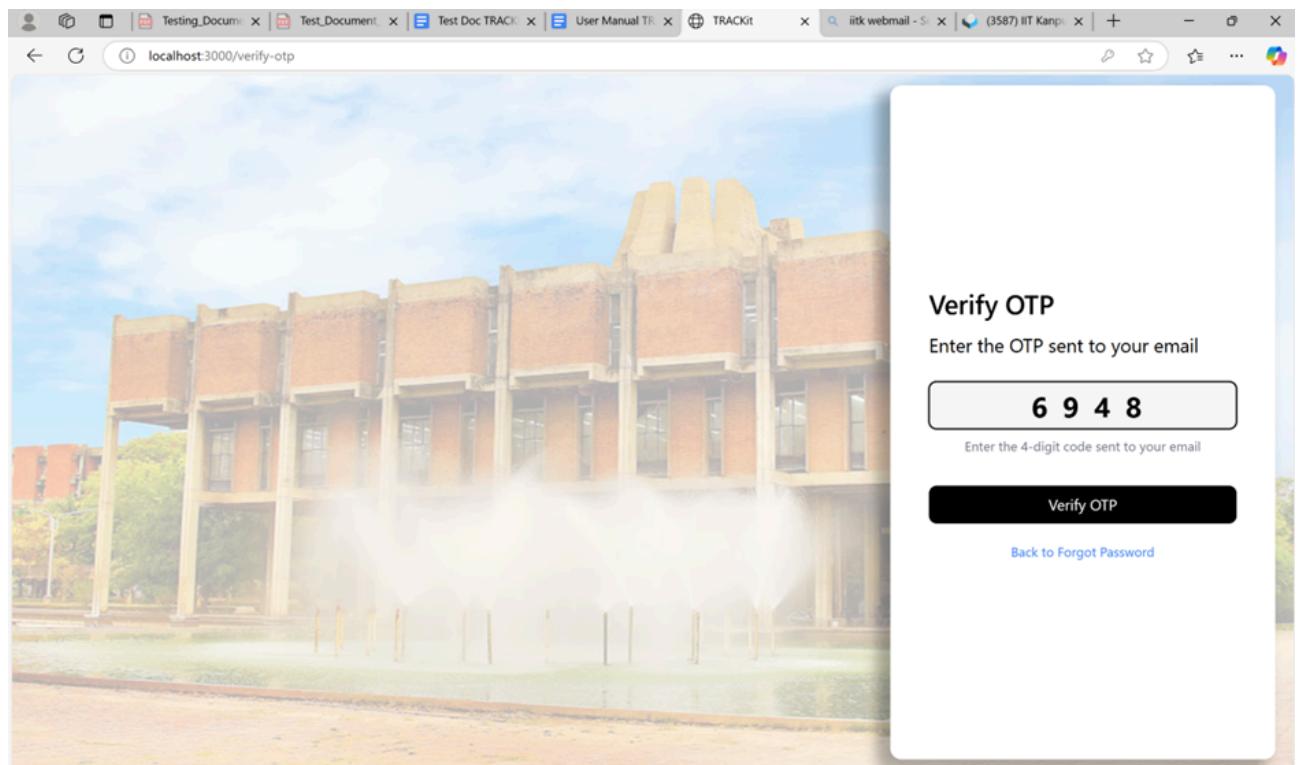
#Result1 : Correctly displays “No such user exists” on entering unregistered users.



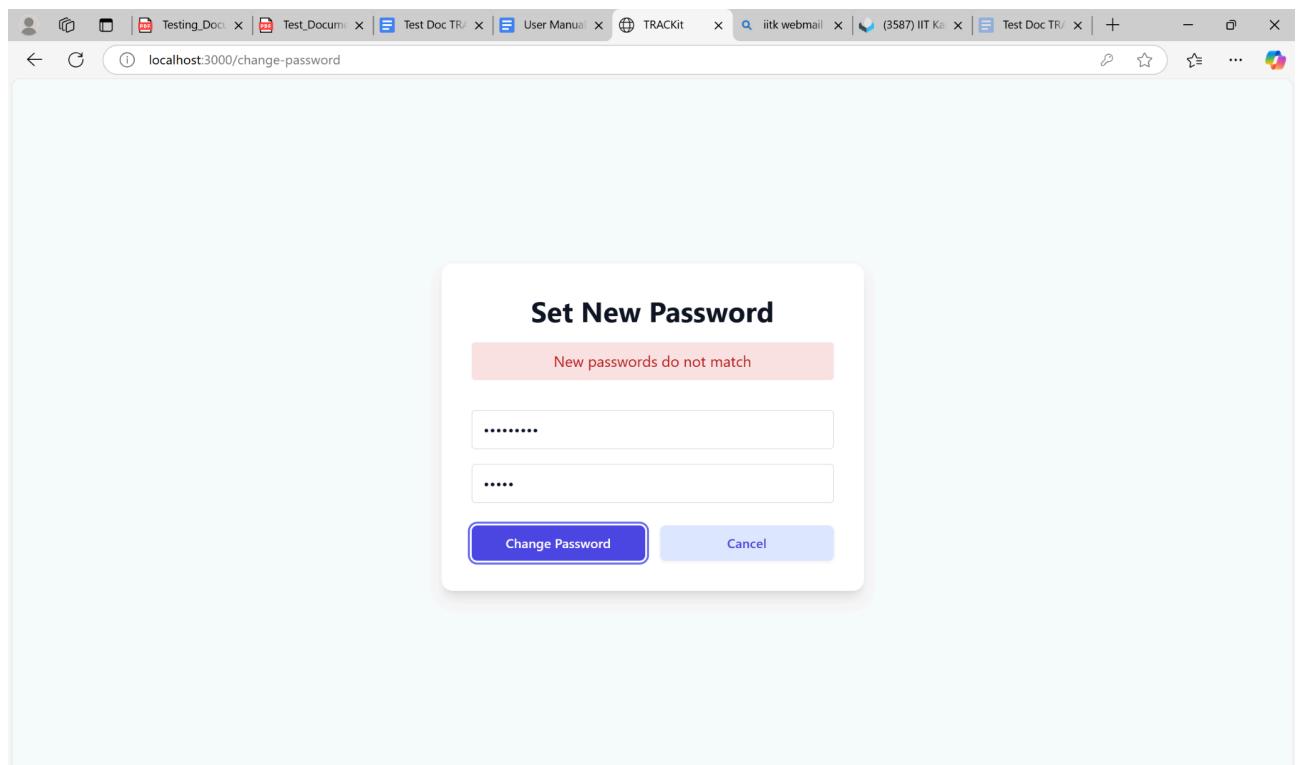


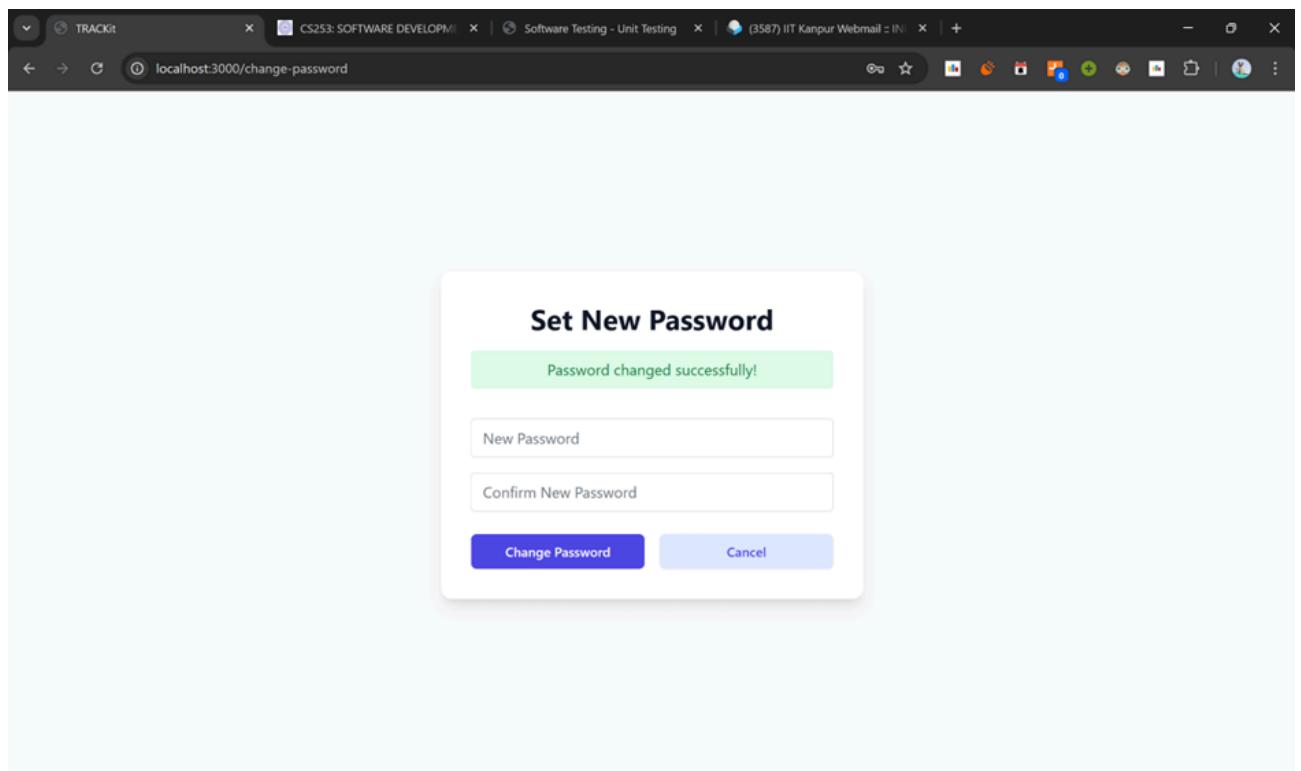
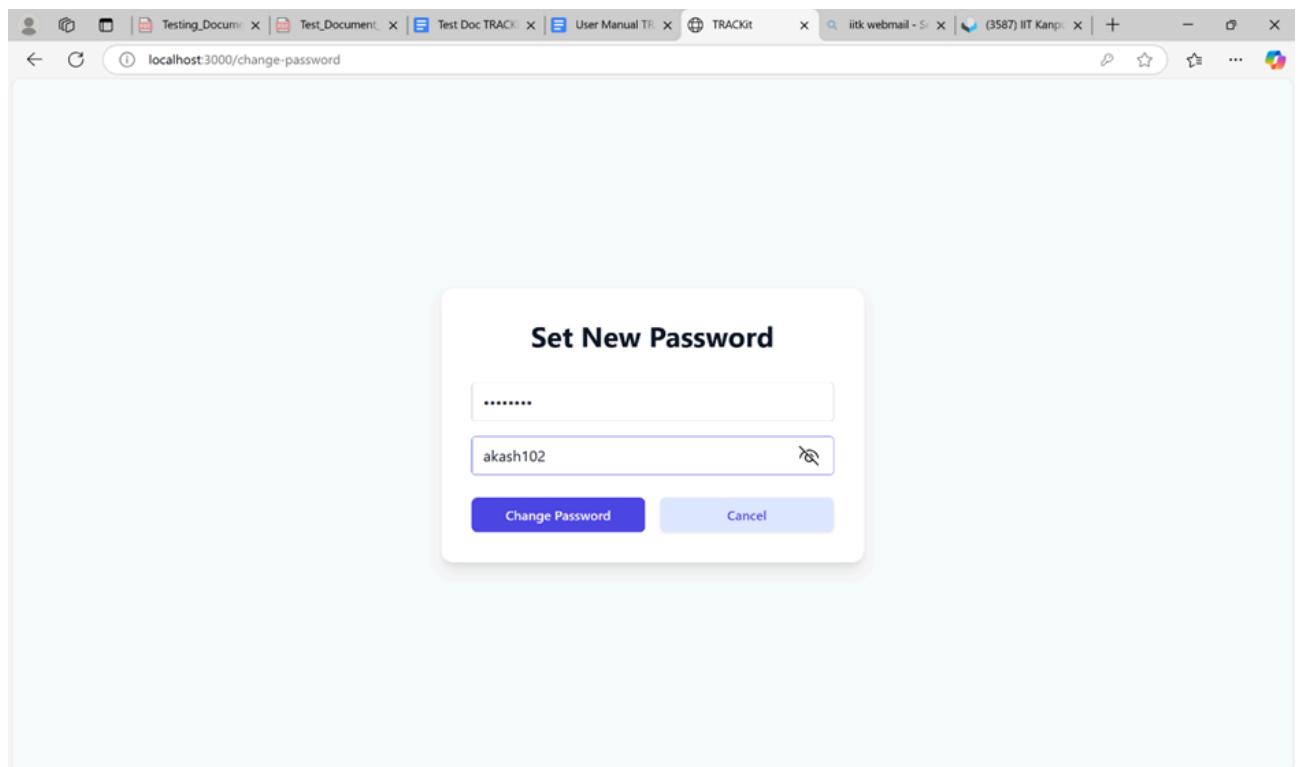
The screenshot shows an email client interface with a sidebar on the left and a main message view on the right. The sidebar includes icons for Drafts (3587), Sent, Junk, and Trash (25). The main view shows an inbox with several messages listed. One message is highlighted, showing the subject 'TRACKit Password Reset OTP' and the sender 'pwdchange247@gmail.com' from '2025-04-02 22:12'. The message content starts with 'You have requested to reset your password.' and provides an OTP of '6948'. It also states that the OTP will expire in 10 minutes and includes a note about ignoring it if not requested. The message list also includes other recent emails from various senders like Rohit Budhiraja, [students], and President, Students' Gymkhana.

#Result2 : On entering **registered username** and pressing continue email is sent to “**registered email address**” and page to verify otp opens up correctly.



#Result3 : On entering **Correct OTP** , otp is verified successfully and user is correctly taken to **set new password** page





#Result4 : On entering a matching new and confirmed password, the password is getting changed correctly .

The image consists of two screenshots of the TRACKit application interface.

Top Screenshot: The browser address bar shows "localhost:3000/login". The page features a large background image of a multi-story brick building under construction or renovation. Overlaid on the right side is a white sign-in form with the heading "Welcome to TRACKit". It contains fields for "Username" (filled with "akashv22") and "Password" (filled with "akash102"), a "Forgot Password" link, and a "Continue" button.

Bottom Screenshot: The browser address bar shows "localhost:3000/dashboard/courses". The page has a header "TRACKit" and a sidebar with links: "My Courses" (which is bolded), "Performance", "Profile", and "Contact Us". Below the sidebar is a "Logout" link. The main content area displays a message "No courses available" with the subtext "You are not enrolled in any courses yet.". A modal window titled "Update your password?" is open on the right, containing fields for "Username" (filled with "akashv22") and "Password" (filled with "akash102"), and buttons for "Update" and "Not now".

#Result5 : Password is successfully updating in the backend and the user is able to login with the changed password perfectly.

Structural Coverage:

The module is working well and fulfills 100% of the requirements of the SRS.

Additional Comments: No bugs were found. Works perfectly smooth and fine .

FORUM :

Module Details :

This module encompasses testing procedures for Forum services inside a particular course. Api testing is done previously , following tests include testing of features like posting a forum , replying someone , deleting your own post and for faculty the feature to delete any student's response(s).

Test Owner :

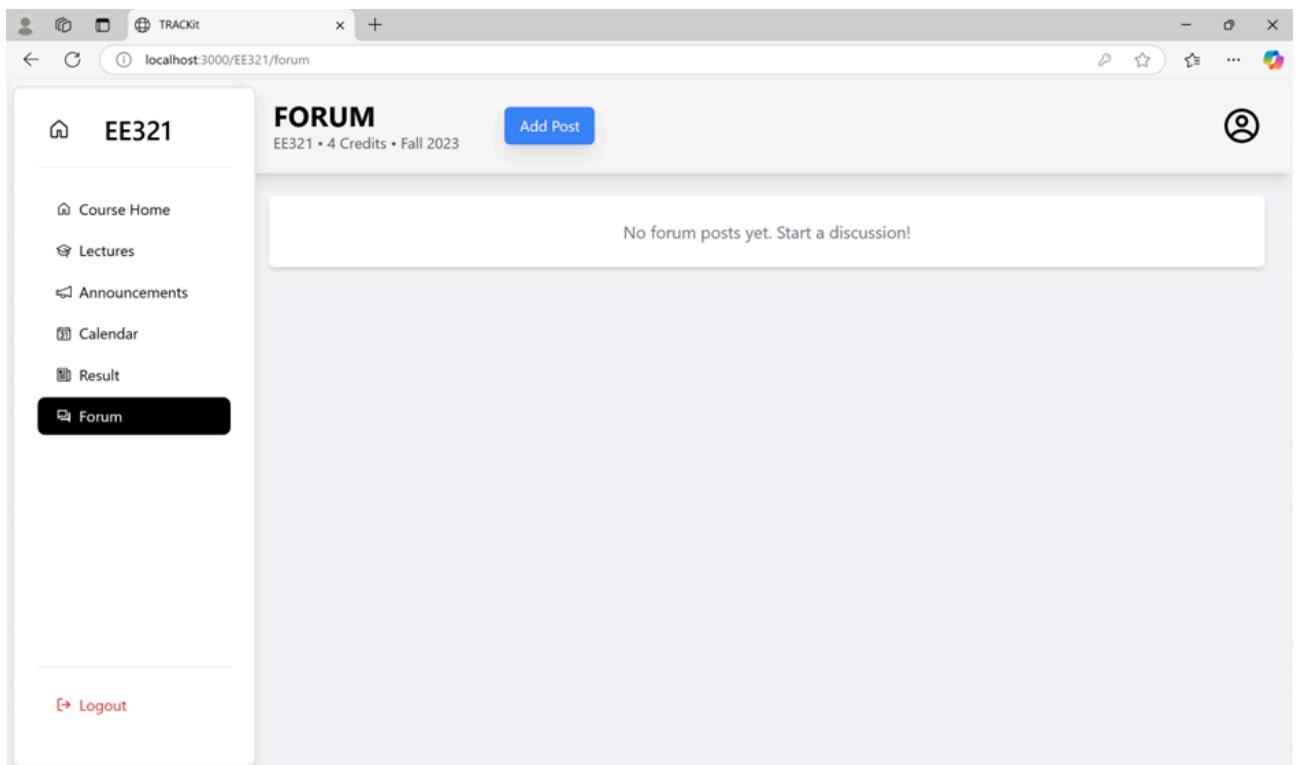
Akash Verma, 220097

Test Date :

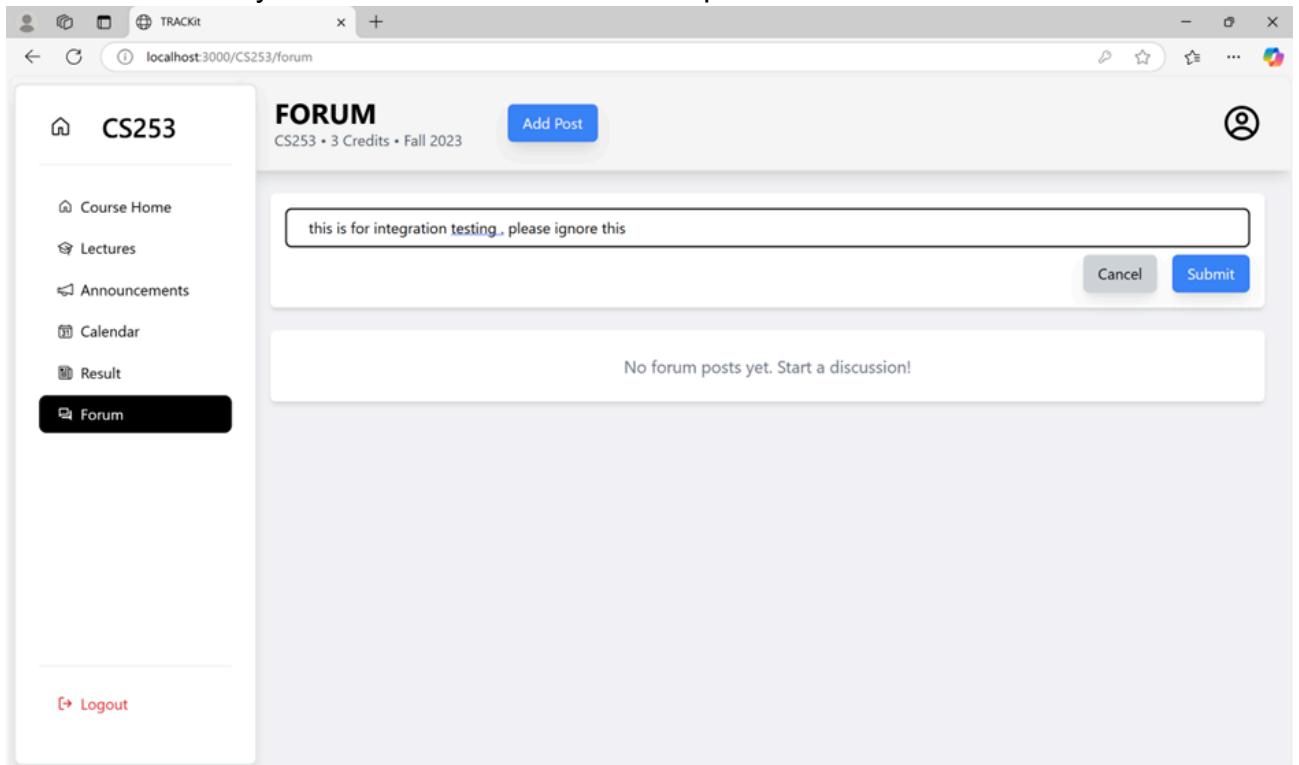
03/04/2025

Test Results :

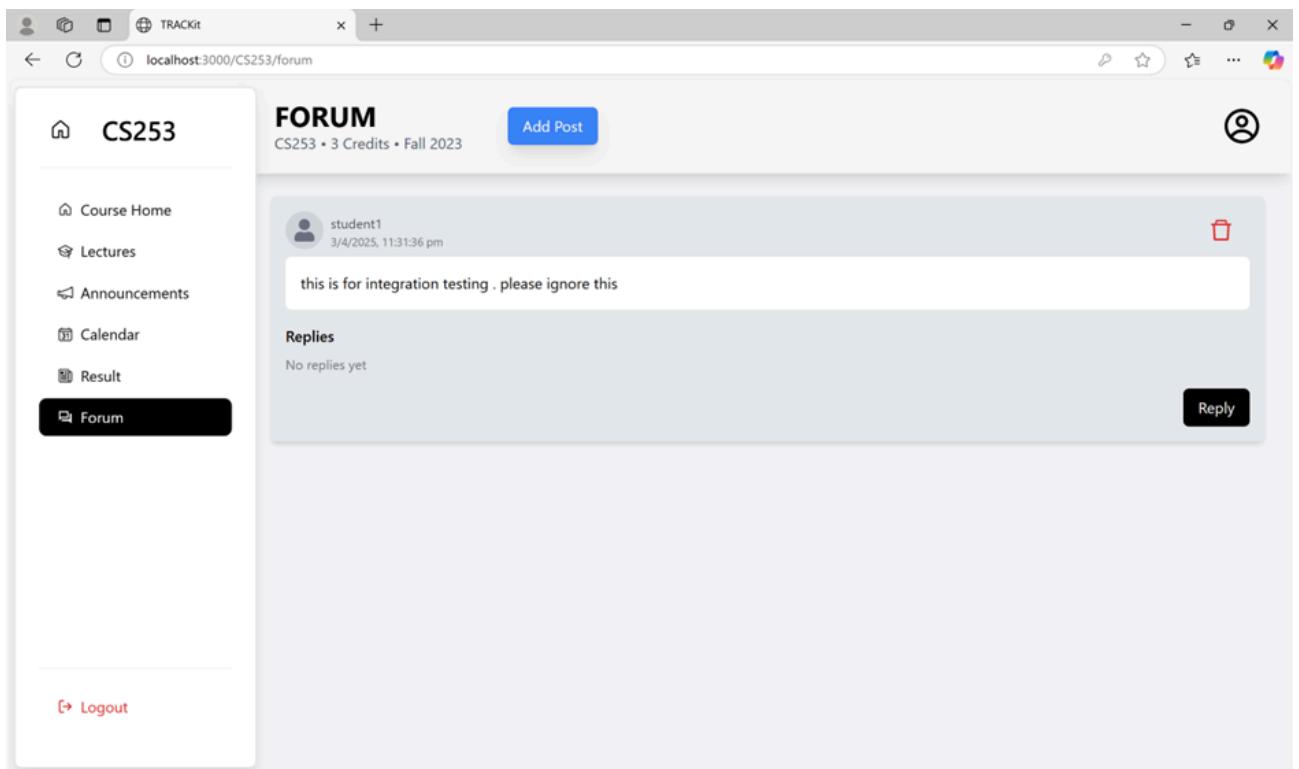
- After selecting an enrolled course in the main dashboard and selecting the forum option in the left panel , the forum page is opening correctly.



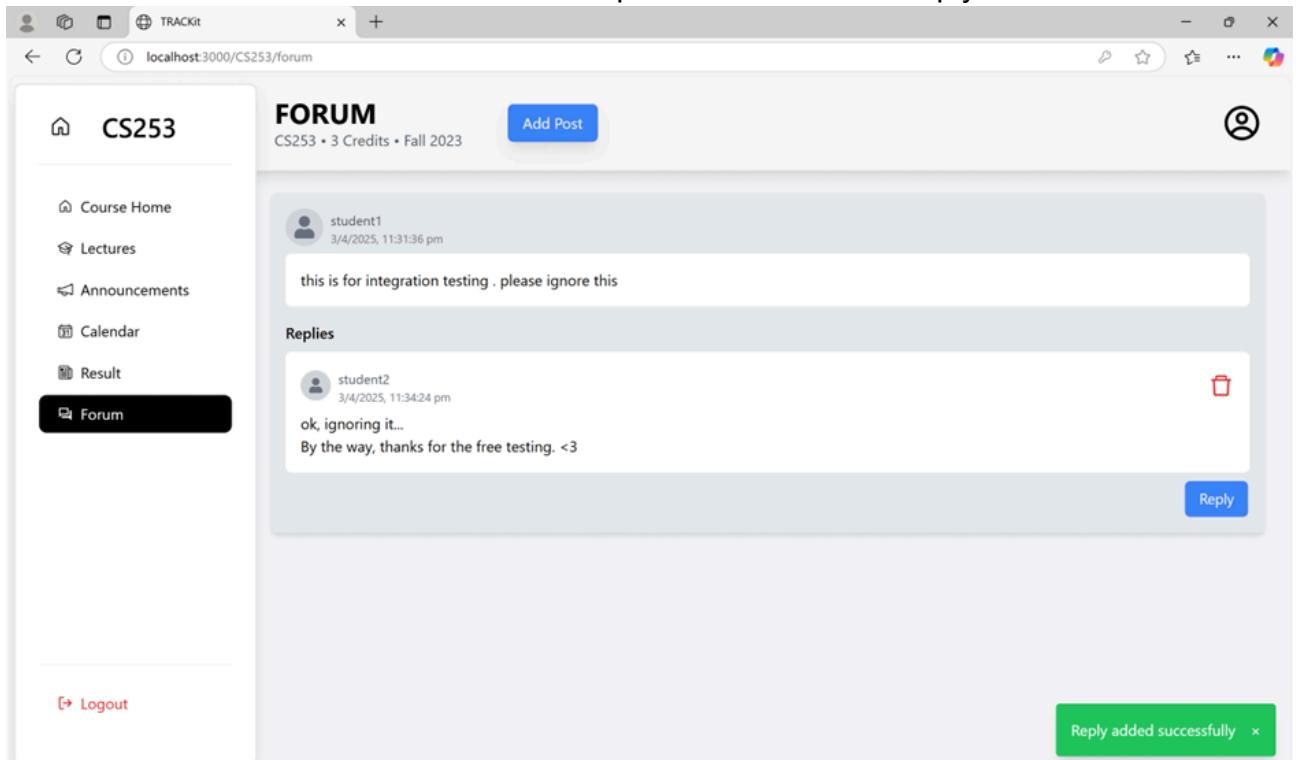
- On clicking upon Add post , a box to post new queries is opening successfully where one is able to write its queries:



- On clicking “submit” button , forum is getting posted :



- Other users are also able to see the post and are able to reply to it.



- Instructor is also able to see the post and able to reply on it :

The screenshot shows a web browser window for the CS253 forum. The left sidebar includes links for Course Home, Lectures, Announcements, Calendar, Result, and Forum, with 'Forum' currently selected. The main content area displays a post from 'student1' and its replies. A new reply from 'faculty2' has been added successfully, indicated by a green message box at the bottom right.

FORUM
CS253 • 3 Credits • Fall 2023

Replies

student1 3/4/2025, 11:31:36 pm
this is for integration testing . please ignore this

student2 3/4/2025, 11:34:23 pm
ok, ignoring it...
By the way, thanks for the free testing. <3

faculty2 Instructor 3/4/2025, 11:41:23 pm
Happy that you guys are engaging with each other..
But don't post unwanted things

Reply

Reply added successfully

- Instructor is also able to delete any students reply and also able to successfully delete the whole forum:

The screenshot shows the same CS253 forum page. The reply from 'faculty2' has been deleted, as indicated by a red trash can icon next to the reply and a green message box at the bottom right.

FORUM
CS253 • 3 Credits • Fall 2023

Replies

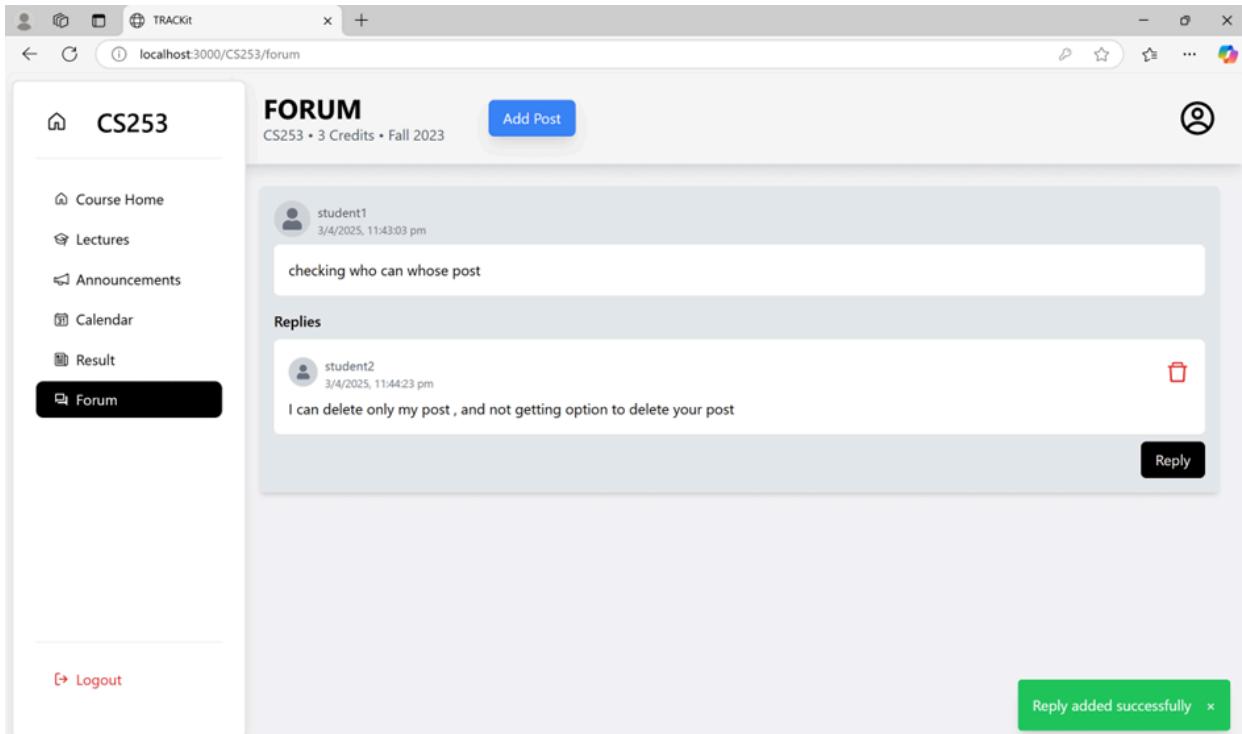
student1 3/4/2025, 11:31:36 pm
this is for integration testing . please ignore this

faculty2 Instructor 3/4/2025, 11:41:23 pm
Happy that you guys are engaging with each other..
But don't post unwanted things

Reply

Reply deleted successfully

- Students are not able to delete any post **other than of their own** :



Structural Coverage:

The module is working well and fulfills 100% of the requirements of the SRS.

Additional comment :

- A bug was found that while creating a new forum post , the forum creator was not able to create a multiline post. Bug is fixed and now it is working fine.
- Apart from these attached tests , other things like occurrence of messages in increasing order of date and time etc. is also checked carefully .

4 System Testing

1. Requirement:

When the system is first initialized, no user accounts are present by default except for the **admin account**. The admin holds the authority to add students, faculty members, courses, and manage system-wide settings. Therefore, it is critical to verify whether the admin account is created and functional upon system startup.

We tested the admin login process by navigating to the login page and entering the default admin credentials provided during system setup. Upon successful authentication, the system redirected the user to the admin dashboard, confirming access to administrative functionalities such as adding users and courses.

To ensure robustness, we also attempted login with incorrect admin credentials and verified that the system displayed appropriate error messages (e.g., “**Invalid credentials**”). No issues or bugs were encountered. The admin account functioned correctly, satisfying the specified requirement.

Test Owner: Aryan Bansal

Test Date: 28/03/2025

Test Results: The admin account was successfully verified on the deployed system. No bugs were reported.

2. Requirement:

The **admin should be able to add students and faculty members** (i.e., users) to the system by manually entering their details, including **username**, **email**, **password**, and relevant **academic information**. Additionally, once added, users should appear in the system database and on the user management page accessible to the admin.

To validate this functionality, we performed a comprehensive test where the admin added multiple users with all necessary details. We confirmed that the newly created users were correctly inserted into the system's database. We then navigated to the **User Management Page**, where the admin can view and manage all users, and verified that the newly added users were accurately displayed.

All operations executed as expected. No bugs or inconsistencies were found during testing, confirming that the functionality met the requirement.

Test Owner: Aryan Bansal

Test Date: 28/03/2025

Test Results: User addition, listing in the admin view, and password update functionality worked as expected. No bugs were reported.

Additional Comments: The username and password could be e-mailed on the registered email address.

3. Requirement:

The system provides a **Bulk Upload** feature that allows the admin to add multiple users (students and faculty) at once by uploading a properly formatted **CSV file**. The system should validate the uploaded file and display an error message if the format is incorrect or if the file content is invalid. If the file is correct, all users listed in the CSV should be successfully added to the system database.

First, we uploaded a **correctly formatted CSV file** containing valid user details for both students and faculty. After submission, we verified that all users were accurately added to the system database and were visible on the **User Management Page**.

Next, we tested the system's validation logic by uploading an **incorrectly formatted CSV file**—one with missing columns, invalid headers, or incomplete data. In each case, the system displayed an appropriate **error message**, ensuring that no incorrect entries were added.

Test Owner: Aryan Bansal

Test Date: 28/03/2025

Test Results: Bulk upload with correct CSV succeeded; errors were appropriately handled for incorrect files. No bugs were reported.

4. Requirement:

While the admin is adding users (students or faculty), the system should **enforce uniqueness** for certain critical identifiers to maintain data integrity. Specifically:

- No two users should have the **same username**.
- No two users should be registered with the **same email address**.

- In the case of students, the **roll number** must also be unique across the system.

To test this functionality, we initially attempted to add users with duplicate usernames, email addresses, and student roll numbers. The system should **prevent duplication** for usernames and emails by displaying an appropriate error message.

However, during early testing, we discovered that the system allowed the creation of **multiple student accounts with the same roll number**, which was a bug. This issue was reported, and the development team addressed it promptly. After the fix was applied, we re-tested the scenario and confirmed that the system **no longer allows duplicate roll numbers** for students.

The final outcome verified that all three uniqueness checks were functioning as required.

Test Owner: Aryan Bansal

Test Date: 29/03/2025

Test Results: Duplicate prevention for username, email, and roll number now works as expected. One initial bug with roll number duplication was resolved and re-tested.

5. Requirement:

Once a user (student or faculty) has been added to the system, they should be able to **log in using their username and password**. Additionally, the system must support a "**Forgot Password**" feature to allow users to reset their password if forgotten or to update the temporary password provided by the admin or according to their needs.

The password reset workflow includes the following steps:

- The user clicks on the "**Forgot Password**" option.
- The system prompts the user to enter their **username**.
- If the username is valid, an **OTP is sent to the user's registered email address**.
- The user must then enter the received OTP to proceed to **reset the password**.
- The system should only accept the **new password** for future logins, rendering the old one invalid.

During testing:

- We first verified that users could **successfully log in** using the correct username and password.
- We then tested the "**Forgot Password**" functionality. Initially, the OTP **email was not being sent**, which was reported as a bug.
- After this issue was resolved, we verified that the email with the OTP was successfully received by the user, and the OTP validation and password change process worked as expected.
- Lastly, we confirmed that users were **able to log in using the new password** and that the old password was no longer valid.

This test validated both the core login functionality and the robustness of the password recovery mechanism.

Test Owner: Mayur Agrawal

Test Date: 29/03/2025

Test Results: Login and password reset functionalities worked as expected after an initial email delivery issue was resolved. No further bugs were reported.

6. Requirement:

The admin should be able to **add new courses** to the system. These courses will later be assigned to faculty and students as part of academic scheduling. A **Course Management** page is provided, where all existing courses can be viewed and managed.

During testing, we verified the following:

- The admin could **access the Course Management page**.
- New courses could be added by entering course-related details such as course name, code, and description.
- After submission, the course appeared in the **list of existing courses**, confirming that the addition was successful.
- We also checked whether the data was correctly stored in the backend/database.

Test Owner: Mayur Agrawal

Test Date: 29/03/2025

Test Results: Course addition and listing features worked as expected on the Course Management page. No bugs were reported.

7. Requirement:

The system should provide a **User Management** page accessible to the admin. This page should allow the admin to **view and manage all users** (students and faculty) within the system.

Key functionalities include:

- Viewing user details such as **username, email address, and academic information** (excluding passwords).
- Editing user information, such as **username, email**, or academic details. Ensuring that **if a username is updated**, the affected user can only log in using the **new username**, and the old one becomes invalid.
- **Deleting users** from the system, which should remove their data from the database and revoke access (i.e., deleted users should not be able to log in with old credentials).

During testing:

- We navigated to the User Management page and confirmed that all users were correctly listed.
- We successfully **edited user details**, including changing usernames and academic data, and verified that the updated information was reflected both in the UI and the database.
- We **deleted sample users** and ensured they were removed from the database and could no longer access the system.
- We confirmed that after a username change, **login was only possible using the updated username**.

The entire flow was tested across different types of users, and the feature behaved as intended.

Test Owner: Aryan Bansal

Test Date: 30/03/2025

Test Results: The User Management page worked as expected. All edit and delete operations were successfully performed. No bugs were encountered.

8. Requirement:

The admin should be able to **manually assign students or faculty members to specific courses** through the User Management page.

Functionality tested:

- The admin locates a user (student or faculty) via the User Management interface.
- The admin selects the desired course from the available list and assigns it to the user.
- Upon successful assignment, the user should **see the course listed** in their dashboard after logging in, with access to its content.

During testing:

- We verified that users were successfully added to the selected courses.
- We confirmed that these users could access the course and its material upon logging in.
- Backend validations ensured the mapping between users and courses was properly stored in the database.

Test Owner: Aryan Bansal

Test Date: 30/03/2025

Test Results: The manual user-to-course assignment feature functioned correctly. Users were able to access newly assigned courses. No issues were encountered.

Additional Comments:

- A suggested improvement is to allow **students to request enrollment** in a course, which the corresponding instructor can then **accept or reject**. This would significantly reduce the workload on the admin.
- Another potential enhancement is the **auto-enrollment** of students into mandatory courses based on their department and academic year, streamlining the process further.

9. Requirement:

The **Course Management** page should allow the admin to **view and manage all courses** in the system.

Functionality tested:

- The admin was able to **view a complete list of courses** currently added to the system. Each course entry displayed key information such as **course code, name, and id**.
- The admin could **edit any of these course details** (e.g., correcting course codes or updating descriptions), and these changes were **immediately reflected for all users** (students and faculty) enrolled in that course.
- The admin was also able to **delete any course** from the system. After deletion:
 - The course was no longer visible on the Course Management page.
 - Users previously enrolled in the course could no longer access it from their dashboard.
- Additionally, the admin was able to **view the list of users** (both students and faculty) assigned to each course and had the option to **remove any user** from the course if necessary.

Test Owner: Aryan Bansal

Test Date: 30/03/2025

Test Results: The Course Management page performed as expected. All functionalities, including course modification, deletion, and user management within courses, worked correctly. No bugs were encountered during testing.

10. Requirement:

The **Course Management** page should allow the admin to **bulk upload students into a course** using a properly formatted CSV file.

Functionality tested:

- The admin was able to **select a course** and upload a CSV file containing a list of student usernames to be added to that course.
- The system validated the uploaded file. If the **CSV file format was incorrect** (e.g., missing columns, improper headers), or if the file contained **usernames not already registered in the system**, the system displayed **clear error messages** and **prevented the upload**.
- For **correctly formatted CSV files** containing valid student usernames, all users were **successfully enrolled in the selected course**.
- Once uploaded, these students could **see the course in their dashboard** upon logging in and access course content.

Test Owner: Aryan Bansal

Test Date: 31/03/2025

Test Results: The bulk upload feature worked as expected. Proper error handling was observed for invalid files, and students listed in valid CSV files were correctly enrolled in the course. No bugs were reported.

11. Requirement:

There should be a "**Contact Us**" page accessible to the **admin**, where the contact details of the personnel responsible for the maintenance of the *Track It* system are displayed.

Functionality tested:

- The page was accessible from the admin dashboard.
- It correctly displayed the **names**, **email addresses**, **contact numbers**, and **roles** of all relevant maintenance team members.
- The information was **read-only** to the admin, ensuring data integrity.

Test Owner: Aryan Bansal

Test Date: 31/03/2025

Test Results: The contact information page worked as expected. All details were displayed correctly and were accessible to the admin without any errors. No issues were reported.

12. Requirement:

All types of users — **admin**, **faculty**, and **students** — should be able to log out from their respective accounts. Upon successful logout, users should be redirected to the **login page**, and their session should be terminated.

Functionality tested:

- The **logout button** was visible and accessible from the navigation bar/dashboard for all user types.
- On clicking logout, users were redirected to the login page without delay.
- Logout functionality was tested on different user roles to ensure uniform behavior.
- No residual session data remained post logout.

Test Owner: Mayur Agrawal

Test Date: 01/04/2025

Test Results: Logout functionality worked as expected for admin, faculty, and student accounts. All users were redirected to the login page after logout, and no bugs were reported.

13. Requirement: Only faculty members assigned as instructors of a course should have the ability to post, edit or delete announcements on the Course Announcement page. Students should only have viewing access. All users (faculty and students) added to a course must be able to view any announcements posted.

Testing Description:

We verified the access control functionality by logging in as both an instructor and a student. As a faculty member assigned to a course, we tested the ability to add a new announcement and confirmed that it was successfully posted. We also checked that the faculty member could delete or edit the announcement, with a confirmation prompt appearing before deletion.

Next, we logged in using a student account added to the same course. We validated that the student could see the announcement but was unable to interact with any posting or deletion functionalities.

Test Owner: Aryan Bansal

Test Date: 01/04/2025

Test Results: The Course Announcement page functions as expected. Role-based access control is enforced correctly, and students are restricted to view-only access. Faculty members can post and delete announcements. No bugs were reported.

Additional Comment:

A notification email containing the announcement details can be sent to all users enrolled in the course upon creation of a new announcement. This feature was suggested to ensure timely communication.

14. Requirement: Once a quiz, assignment, lab test, project, or any evaluative event has been conducted and evaluated, instructors should be able to publish results on the **Result Page**. The functionality must allow instructors to:

- Add results by specifying the **Exam Name**, **Total Marks**, and **Weightage**.
- View a list of all enrolled students for that course during result entry.
- Manually input the **marks scored** by each student in the corresponding field.

During testing, we verified:

- The instructor could successfully **access the result page** and initiate result entry for a specific course event.
- Upon entering exam name, total marks, and weightage, a **list of students** was displayed with editable fields for entering marks.
- Manual entry of scores was working correctly and persisted after submission.
- Students could only **view** their results and were not able to modify them.
- We also ensured if a student is added to the course later after a quiz has taken place the marks scored by him are shown as **N/A** to both the student and instructor.

Test Owner: Aryan Bansal

Test Date: 02/04/2025

Test Results: Instructor was able to successfully enter and publish results. Data visibility and access were correctly enforced.

15. Requirement: Once results are published for an evaluative event (quiz, assignment, lab test, etc.), the Results Page should support the following features for faculty and students:

For Faculty:

- View marks scored by all students for any specific exam.
- Download the complete result set for offline use.
- Edit/Update marks if errors are identified post-publication.
- View a **histogram** that visualizes the distribution of marks for analytical insights.

For Students:

- Students can **only view their own marks** and not the marks of others.
- For each exam, students are also shown aggregate statistics : Maximum marks, Mean, Median, Standard deviation, Weightage & Total marks of the exam

Testing Details:

- Faculty successfully retrieved full student-wise marks for selected exams and verified data.
- Download option worked as expected; a structured xlsx was generated with appropriate data.

- Faculty were able to update marks for individual students; updates were correctly saved and reflected on subsequent views.
- Histogram of mark distribution loaded correctly.
- Students were restricted from accessing peer marks and only saw their own scores and the relevant statistics.
- All data was accurate, and no calculation or access issues were encountered.

Test Owner: Aryan Bansal

Test Date: 01/04/2025

Test Results: All functionalities worked as intended. Role-based data visibility was correctly enforced. Histogram and statistics were rendered accurately.

16. Requirement: The forum should allow authorized users (original poster or instructor) to delete their posts and replies. The functionality must enforce proper access control so that only the original poster or an instructor can remove content they have created.

During testing, we logged in as a student who created a forum post and, using the delete icon, removed the post after confirming the deletion via a prompt. In a separate test, we logged in as an instructor and deleted a reply made by a student. After each deletion, we verified that the corresponding content was completely removed from the forum thread and that no residual data was visible. We also confirmed that unauthorized users could not access the delete function on content they did not create.

Test Owner: Mayur Agrawal

Test Date: 02/04/2025

Test Results: Deletion operations for forum posts and replies were successful.

Additional Comments: NA

17. Requirement: Faculty should be able to upload and organize course materials into modules/sections and lectures on the Lectures page. This includes creating a module/section structure and adding lecture content (title, description, and file or link).

Description: On the EE201 course, **prof_smith** navigated to the **Lectures** page. We performed the following:

- Clicked "Add Module" and entered "Week 1" as the module name, then saved.

- Within "Week 1", clicked "Add Section" and named it "Introduction".
- Under that section, click "Add Lecture". In the add lecture form, entered Title: "Lecture 1: Basics", Description: "Introduction to Circuits", and uploaded a PDF file of lecture notes. Then click "Submit".

The expected outcome was that a new module "Week 1" with a section "Introduction" appears on the Lectures page, and inside it the lecture "Lecture 1: Basics" is listed with options to download the PDF.

Test Owner: Mayur Agrawal

Test Date: 02/04/2025

Test Results: The Lectures page updated immediately. "Week 1" module was created and could be expanded to show the "Introduction" section. Under Introduction, **Lecture 1: Basics** was listed. An icon for a PDF and a download link were present next to the lecture title. We verified the file by clicking download, which successfully downloaded the uploaded PDF. Students enrolled in EE201 were also able to see "Week 1 > Introduction > Lecture 1: Basics" on their lecture page (verified with a student login). The content addition feature worked flawlessly.

Additional Comments: NA

18. Requirement: The system should require essential fields (like lecture title) when adding lecture content, and it should not allow adding a lecture without them.

Description: We attempted to add another lecture in "Week 1" but left the **Lecture Title** field empty to test validation. A file was selected and description provided, then "Submit" was clicked. The expected behavior: the system refuses to add the lecture and indicates that the title is required.

Test Owner: Mayur Agrawal

Test Date: 02/04/2025

Test Results: The lecture was not added. The interface highlighted the Title field in red and displayed a small error message "Please enter a lecture title." We filled in the title and resubmitted, after which the lecture was added successfully. This confirmed that the application enforces required fields for lecture content (PASS).

Additional Comments: NA.

19. Requirement: Calendar updates and event notifications should synchronize in real time (within 1-2 minutes).

The tester updated events on the instructor's interface and verified that changes appeared on the student's calendar within 2 minutes. Manual refresh and automatic notification checks confirmed timely updates.

Test Owner: Mayur Agrawal

Test Date: 03/04/2025

Test Results: Calendar synchronization and notifications worked as required.

Additional Comments: NA

20. Requirement: The Contact Us page should allow both faculty and student users to submit queries and report issues.

During testing, we navigated to the Contact Us page as both a faculty member and a student. We verified that the user's name and email address were automatically populated. We then entered a valid subject and message in the designated fields. Upon clicking "Send Message," a confirmation message appeared, and we confirmed that the query was logged for support follow-up. Additionally, we checked that an email containing the submitted query was successfully sent to the support team for both user types.

Test Owner: Mayur Agrawal

Test Date: 03/04/2025

Test Results: Contact form submission worked correctly for both faculties and students; a confirmation was displayed and the email notification was successfully sent.

Additional Comments: NA

21. Requirement: Users should be able to change their password from the Profile section in a secure and user-friendly manner.

The system must ensure that:

- Users can access the change password form via the "Change Password" button on their Profile page.
- The user must enter their current password correctly.
- The new password and confirm password fields must exactly match.
- Passwords must meet strength criteria and be securely stored.
- Appropriate confirmation or error messages must be displayed depending on the outcome.

Testing Details:

We tested the password change functionality for both student and faculty users.

- Users successfully navigated to the Profile section and accessed the “Change Password” form.
- The current password was entered accurately, followed by matching entries in the new password and confirm password fields.
- Upon submission, the system validated the inputs and displayed a success message (“Password changed successfully”).
- We verified that users could log in using the new password immediately after the change, and that the old password was no longer valid.
- Additional tests were conducted with incorrect current passwords and mismatched new password entries. The system blocked the update in these cases and displayed appropriate error messages: “Current password is incorrect” and “New passwords do not match”.
- All validation logic worked correctly and prevented unauthorized or erroneous changes.

Test Owner: Mayur Agrawal

Test Date: 04/04/2025

Test Results: Password change functionality worked as expected for both students and faculty. All validations, success responses, and error messages were accurate and securely enforced.

21. Requirement: The Profile section should display accurate user details including name, roll number, and department.

We logged in as a user for different users, navigated to the Profile section, and confirmed that all personal details were correctly populated from the database. The interface allowed for clear viewing of information.

Test Owner: Mayur Agrawal

Test Date: 04/04/2025

Test Results: User profile details were displayed accurately with no discrepancies.

Additional Comments: NA

22. Requirement: Students should be able to view the course calendar with all scheduled events.

On the course home page, we verified that the calendar displayed a merged view of all events. Navigation controls (day, week, month views) were tested and events appeared with correct details (title, date, course code).

Test Owner: Mayur Agrawal

Test Date: 03/04/2025

Test Results: The calendar view was fully functional and events were displayed as expected.

Additional Comments: NA

23. Requirement: Students should be able to view all course announcements on the course home page.

We logged in as a student and verified that all announcements (posted by the instructor) for that course were visible, sorted in reverse chronological order. Each announcement could be expanded to view details and was accompanied by the correct posting date and author information.

Test Owner: Mayur Agrawal

Test Date: 03/04/2025

Test Results: Announcements were displayed accurately and in the correct order.

Additional Comments: NA

24. Requirement: Students should be able to access all lecture modules, sections, and materials that instructors have uploaded, and download resources as needed.

Description: We navigated to the **Lectures** page of EE201 as a student. We expected to see the structured content that the faculty added, organized by week/module. Specifically, "Week 1" module with the "Introduction" section and inside it "Lecture 1: Basics" (or its updated title). We attempted to open the lecture content and download the attached PDF to ensure the student has access. We also tested that lectures from other weeks (if any) can be expanded or collapsed.

Test Results: We accessed the Lectures page for EE201 from a student account and observed that "Week 1" was displayed with an expandable arrow. Upon expanding it, the "Introduction" section became visible, containing **Lecture 1: Circuit Basics**. We clicked on the lecture title, and the uploaded PDF opened directly in the browser. We also tested the download functionality by clicking the download icon, which successfully downloaded the file. The weekly sections were collapsible, allowing us to hide or display content as needed. As students, we only had read-only access—no edit or delete options were available, as expected. This confirms that students can effectively access and retrieve lecture materials as intended.

Test Owner: Mayur Agrawal

Test Date: 03/04/2025

Test Results: Lectures were displayed and accessible correctly; collapsible sections functioned properly.

Additional Comments: None.

25. Requirement: The Performance section should provide students with a consolidated overview of their academic standing across all enrolled courses. It must include an overall summary of performance, as well as course-wise breakdowns with detailed metrics and visual analytics.

Testing Details:

We accessed the **Performance** tab from a student account enrolled in multiple courses. At the top of the page, we verified the summary dashboard which displayed:

- Total number of currently enrolled courses (e.g., 5).
- Weighted average performance percentage across all courses (e.g., 68.5%).
- Relative-to-median metric indicating how the student is performing compared to peers (e.g., 12.3% below median).

Below the summary, we tested the **Course-wise Breakdown** section. Each active course was shown as a collapsible card displaying the course title, credits, and a dropdown arrow. On clicking the arrow for a course (e.g., CS253: *Software Development*), a detailed view appeared containing:

- A **bar graph** comparing obtained marks (in teal) and weightage (in purple) for each evaluation component such as quizzes, assignments, and midterms.
- A **tabular breakdown** listing the evaluation name, weightage, total marks, obtained marks, mean, median, highest score (max), and standard deviation.

We verified the accuracy of these statistics against known result data. The graph visually reflected performance trends across evaluations, and the table allowed clear comparison. The visualization accurately highlighted performance gaps and strengths.

Additionally, we ensured that:

- The breakdown was only visible to the logged-in student and did not expose any peer information.
- The student could access only their own scores and the aggregate statistics (mean, median, max, deviation), with no edit options available.
- Expanding and collapsing course cards worked as expected without interface glitches.

All metrics, graphs, and interactions functioned smoothly, and data presented was consistent with the results published for that student.

Test Owner: Mayur Agrawal

Test Date: 04/04/2025

Test Results: The Performance section worked as intended. Summary statistics, course-wise breakdowns, and analytics were accurately presented and securely restricted to the student's own data.

Additional Comments: NA

26. Requirement: The Forum section should allow students to initiate, view, and participate in course-related discussions. They should be able to post new queries, view threads, reply to posts, and delete their own content.

Testing Details:

We tested the Forum section functionality from a student account across the following use cases:

- **Creating a New Post:**

We navigated to the Forum tab within a course (e.g., EE210) and clicked the “**Add Post**” button. The interface expanded to reveal a text input field with the placeholder “Your Query.” We entered a question: “*When is the deadline for assignment 1?*” and clicked “**Submit**.” The post appeared instantly in the forum thread list, visible to all course participants.

- **Cancelling a Draft Post:**

We initiated a new post and clicked “**Cancel**” instead of submitting. The input field collapsed, and the draft was discarded as expected, confirming that users can back out of creating a post without errors.

- **Viewing and Replying to Posts:**

As a student, we opened an existing post and verified the following elements:

- Author name (e.g., *student5*)
- Timestamp of post creation
- Post content
- A red trash bin icon, visible only to the post's author, confirming access control

Below the post, a **Replies** section was visible. We clicked the “**Reply**” button, typed a response (“*It's on 4th April.*”), and submitted it. The reply appeared immediately below the original post in a threaded format, with

correct user info, timestamp, and a delete icon visible only to the reply's author.

- **Deleting Own Posts and Replies:**

We tested deleting a post created earlier. The red trash bin icon was visible next to our post, and clicking it triggered a confirmation prompt. After confirming, the post was removed from the thread. We performed a similar test for replies: the delete icon next to our reply allowed us to delete the comment, which was removed instantly upon confirmation. In both cases, deletion was restricted only to the content's original author.

- **Thread Structure & Layout:**

All posts and replies were displayed in a neatly threaded, hierarchical format. Usernames and timestamps were consistently visible, and there was no access to modify or delete others' content. The reply input box was correctly located beneath each post.

Additional Validation:

- Posts were not visible across unrelated courses.
- UI responded correctly without page reloads.
- Role-based permissions were strictly enforced.

Test Owner: Mayur Agrawal

Test Date: 04/04/2025

Test Results: All functionalities—post creation, reply, delete, and view—worked as expected. Proper access control and formatting were enforced, ensuring a smooth and collaborative discussion experience for students.

Additional Comments: NA

Testing the non-functional requirements

Scalability Testing:

1. Test to add 2,500+ users in the server:

We ran a script to add 2,500+ users to stress-test the website functionality.

The website was fully functional after such a high student load.

Manage Users				
FirstName2537	LastName2537	Role: admin	View Details	Delete
FirstName2538	LastName2538	Role: student	View Details	Delete
FirstName2539	LastName2539	Role: faculty	View Details	Delete
FirstName2540	LastName2540	Role: admin	View Details	Delete
FirstName2541	LastName2541	Role: student	View Details	Delete
FirstName2542	LastName2542	Role: faculty	View Details	Delete

2. Test to add 600+ courses:

We ran a script to add 600+ courses to stress-test the website functionality.

```

frontend > test_courses.js > ...
1  const axios = require('axios');
2
3  // Configuration
4  const API_URL = 'http://172.27.16.252:3001/api/admin/course'; // Adjust the endpoint if needed
5  const ADMIN_TOKEN = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16MSwidXNlIiR5cGUiOiJhzGpbiiImlhdCI6MTc0Mzg3MDA0MywizXhwIjoxNzQzODczNjQzfQ.ue4JQRlwB8TouET';
6
7  // Generate random course data
8  const generateCourseData = (index) => {
9    const semesters = ['Fall', 'Spring', 'Summer']; // Valid semesters
10   return {
11     code: `COURSE${index}`,
12     name: `Course Name ${index}`,
13     description: `This is the description for Course ${index}`,
14     department: `Department ${index % 10}`, // Rotate between 10 departments
15     credits: (index % 5) + 1, // Random credits between 1 and 5
16     instructor: `Instructor ${index}`,
17     semester: semesters[index % semesters.length], // Rotate between Fall, Spring, and Summer
18   };
19 };
20
21 // Add a single course
22 const addCourse = async (courseData) => {
23   try {
24     const response = await axios.post(API_URL, courseData, {
25       headers: {
26         Authorization: `Bearer ${ADMIN_TOKEN}`,
27         'Content-Type': 'application/json',
28       },
29     });
30     console.log(`Course ${courseData.code} added successfully:`, response.data);
31   } catch (error) {
32     console.error(`Error adding course ${courseData.code}:`, error.response?.data || error.message);
33   }
34 };
35
36 // Add 10,000 courses
37 const addCourses = async () => {
38   for (let i = 1; i < 10000; i++) {
39     const courseData = generateCourseData(i);
40     await addCourse(courseData);
41   }
42 };
43
44 // Run the script
45 addCourses();

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS node - frontend

```

Course COURSE41 added successfully: { success: true, message: 'Course created successfully', courseId: 49 }
Course COURSE42 added successfully: { success: true, message: 'Course created successfully', courseId: 50 }
Course COURSE43 added successfully: { success: true, message: 'Course created successfully', courseId: 51 }
Course COURSE44 added successfully: { success: true, message: 'Course created successfully', courseId: 52 }
Course COURSE45 added successfully: { success: true, message: 'Course created successfully', courseId: 53 }
Course COURSE46 added successfully: { success: true, message: 'Course created successfully', courseId: 54 }

```

The screenshot shows a list of courses on the 'Manage Courses' page. Each course is represented by a card with the following details:

- Course Name 592**
- Course ID: 600
- Code: COURSE592
- Actions: Edit (yellow), Delete (red)

- Course Name 593**
- Course ID: 601
- Code: COURSE593
- Actions: Edit (yellow), Delete (red)

- Course Name 594**
- Course ID: 602
- Code: COURSE594
- Actions: Edit (yellow), Delete (red)

- Course Name 595**
- Course ID: 603
- Code: COURSE595
- Actions: Edit (yellow), Delete (red)

- Course Name 596**
- Course ID: 604
- Code: COURSE596
- Actions: Edit (yellow), Delete (red)

3. Now we test adding all the 2,500 users in one course

```
frontend > js test_multipleusers_onecourse.js > ...
frontend > .env frontend > testjs > test_courses.js > test_multipleusers_onecourse.js > powershell - frontend > ...
```

The terminal output shows the execution of a JavaScript test file named `test_multipleusers_onecourse.js`. The code performs the following steps:

- Configures Axios with an API URL and an administrator token.
- Creates an Axios instance with an authorization header.
- Fetched all users.
- Assigned a single user to a course.
- Assigned multiple users to a course.
- Filtered students from the list of users.

The terminal also displays log messages indicating successful assignments for each user:

```
User 2568 assigned to course 1: { success: true, message: 'Student added to course successfully' }
User 2571 assigned to course 1: { success: true, message: 'Student added to course successfully' }
User 2574 assigned to course 1: { success: true, message: 'Student added to course successfully' }
User 2577 assigned to course 1: { success: true, message: 'Student added to course successfully' }
User 2580 assigned to course 1: { success: true, message: 'Student added to course successfully' }
All students have been assigned to the course.
```

The screenshot shows a web-based application titled "TRACKit" with a "Not secure" warning in the browser's address bar. The URL is 172.27.16.252:3000/admin/manage-courses. The main page displays a list of courses:

- Communication Systems (Course ID: 1, Code: EE321)
- Software Development (Course ID: 2, Code: CS253)
- Philosophy of Mind (Course ID: 3, Code: PHI452)
- Digital Electronics (Course ID: 4, Code: EE210)
- Circuit Theory (Course ID: 5)

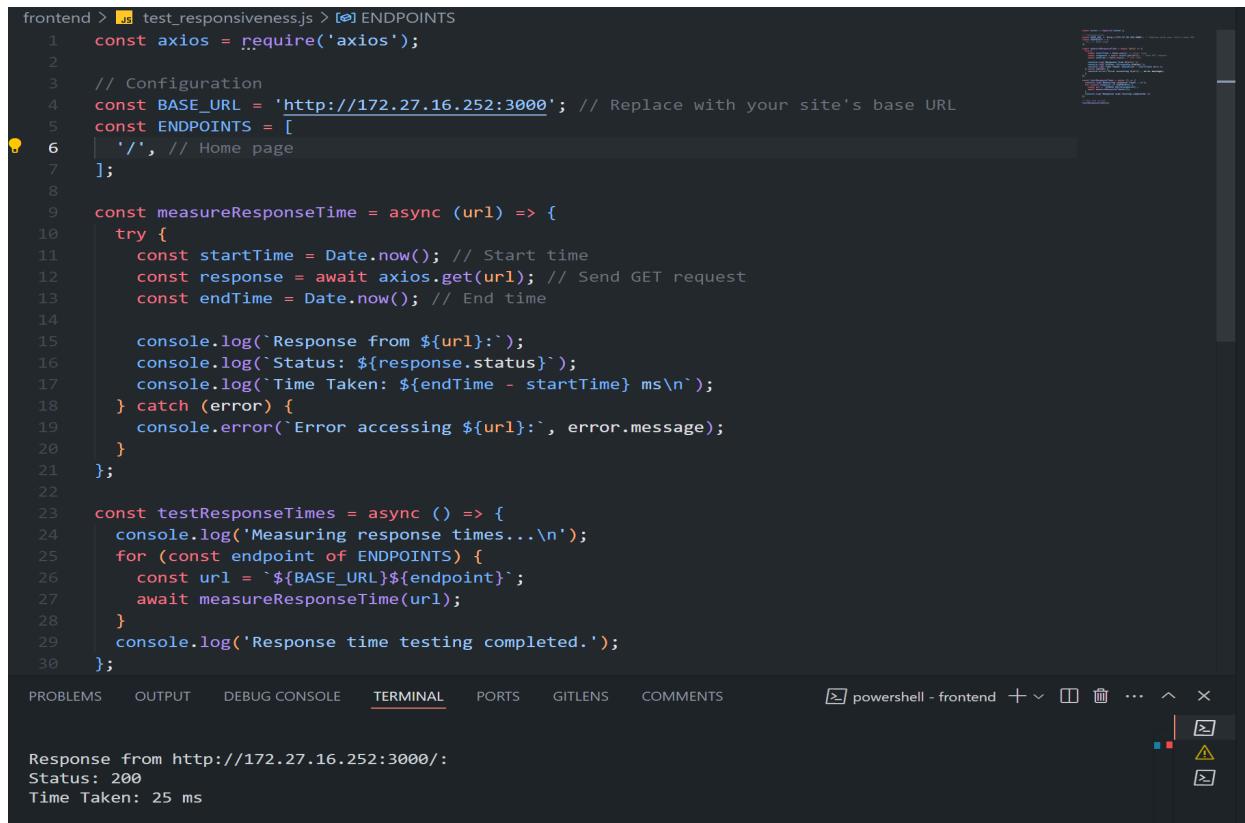
A modal window titled "Course Details" is open for the "Communication Systems" course. The modal has two tabs: "Edit Course" (disabled) and "User Management" (selected). The "User Management" tab displays a section titled "Students Enrolled" with a search bar and three listed students:

- FirstName2535 LastName2535 - ROLL2535 (Computer Science)
- FirstName2538 LastName2538 - ROLL2538 (Computer Science)
- FirstName2541 LastName2541 - ROLL2541 (Computer Science)

Below the list is a red "Remove Selected Students" button. At the bottom right of the modal is a red "Close" button.

Performance testing:

All APIs take less than 50ms on average, and loading the page for the first time takes about 25 ms.



```

frontend > test_responsiveness.js > [ENDPOINTS]
  1 const axios = require('axios');
  2
  3 // Configuration
  4 const BASE_URL = 'http://172.27.16.252:3000'; // Replace with your site's base URL
  5 const ENDPOINTS = [
  6   '/',
  7 ];
  8
  9 const measureResponseTime = async (url) => {
 10   try {
 11     const startTime = Date.now(); // Start time
 12     const response = await axios.get(url); // Send GET request
 13     const endTime = Date.now(); // End time
 14
 15     console.log(`Response from ${url}:`);
 16     console.log(`Status: ${response.status}`);
 17     console.log(`Time Taken: ${endTime - startTime} ms\n`);
 18   } catch (error) {
 19     console.error(`Error accessing ${url}:`, error.message);
 20   }
 21 };
 22
 23 const testResponseTimes = async () => {
 24   console.log('Measuring response times...\n');
 25   for (const endpoint of ENDPOINTS) {
 26     const url = `${BASE_URL}${endpoint}`;
 27     await measureResponseTime(url);
 28   }
 29   console.log('Response time testing completed.');
 30 };

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS COMMENTS

powershell - frontend + ×

```

Response from http://172.27.16.252:3000:
Status: 200
Time Taken: 25 ms

```

Maintainability testing:

The software has been designed in an organized manner so that new features can be added and modifications can be done very easily within two working days. Various components of the software are well organised into folders and comments are added wherever required streamline future improvements

Reliability testing:

The system operates with remarkable robustness, consistently delivering reliable performance. The anticipated downtime of the website is significantly minimized, ensuring uninterrupted accessibility for users. This reliability enhances the overall user experience and fosters trust in the website's functionality.

Portability testing:

Being a web-based app, the software is easily accessible from any device with browser support. We tested the deployed website, and it works perfectly fine on Google Chrome, Mozilla Firefox and Microsoft Edge web browsers.

5 Conclusion

- **How Effective and exhaustive was the testing?**

Testing was done almost exhaustively. Each API underwent testing to ensure that they were modifying the database and returning the desired information when needed. We have taken care of all cases, thus ensuring complete branch coverage. The frontend was also tested and bugs were resolved. Validity of each input has been checked in all forms. Appropriate alerts have been shown in such cases. Edge cases were also tested and were confirmed to work as expected. For example, when faculty tried to enter marks more than total marks of the exam, the system correctly prevented him from doing so.

The testing was quite effective. Each of the modules has been tested by at least 2 members ensuring double check for every sub-system. Testing helps us to find many small bugs mostly at the time of integration testing and we tried to resolve most of them, however some are still pending and will be resolved soon.

- **Which components have not been tested adequately?**

Our test coverage for non functional requirements outlined in the SRS document could be further expanded. Tests for non-functional requirements like reliability, maintainability and ease of learning could not be performed due to time constraint. Integration testing almost tries to check every interaction between the units but due to manual nature of testing some cases might got ignored so it can also be improved further.

- **What difficulties have we faced during testing?**

The main difficulty faced during testing was ensuring non-functional requirements are satisfied, in system testing. Apart from this we also feel that testing every api manually for each possible scenario is quite time consuming and we could have same some time on that if we use automatic testing

- **How could the testing process be improved?**

The testing could have been improved by doing the unit testing or integration testing as automated rather than manual. The developer might subconsciously assume a few crucial things which may be not so obvious to an end-user or someone who was not involved in the development of the software.

Appendix A - Group Log

"The following are the minutes of the collective group meeting held during the document submission. In addition to these discussions, each team member has contributed significantly through individual efforts that may not be fully captured in this documentation. The entire group sincerely appreciates and acknowledges each other's hard work and dedication."

Date	Timing	Duration	Minutes
01/04/25	14:00-16:00	2hrs .	<ul style="list-style-type: none"> ● Discussed Various Types Of testing. ● Distributed Tasks Of Various types of testing among subgroups/individuals within the team. ● Divided work of making user manual
02/04/25	17:00-21:00	4 hrs.	<ul style="list-style-type: none"> ● Discussed Various Doubts faced by each subgroup/individuals in their respective tasks.
03/04/25	22:00-02:00	4 hrs.	<ul style="list-style-type: none"> ● Every testing unit updated and organised their work in doc and reported bugs to respective developer
04/04/25	16:00-18:00	2 hrs.	<ul style="list-style-type: none"> ● Resolved bugs , reconducted the testing and updated the docs ● Revised the structure and content of the user manual.
05/04/25	20:00-23:00	3 hrs.	<ul style="list-style-type: none"> ● Finalised the test document and User Manual