

Name: David Alonge

Assignment: Final Report

Instructor: Dr. Kahanda

Date: 4/30/2024

Project Name: Course-Connect

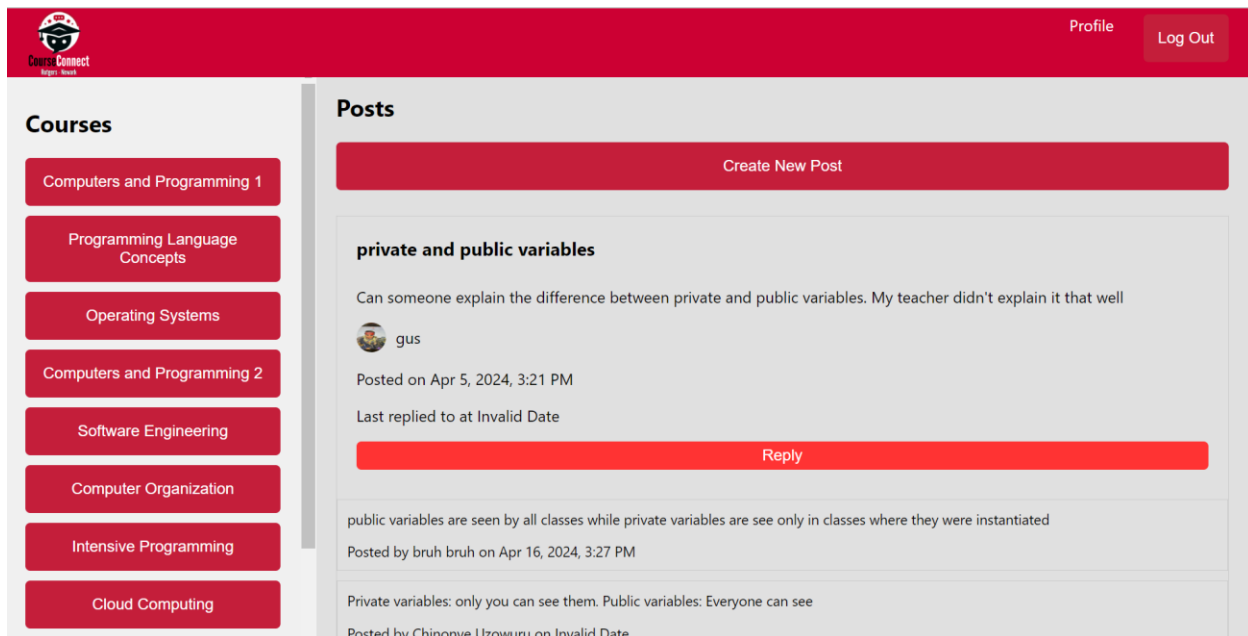
Project Description: Student-Tutor Interaction Platform

TABLE OF CONTENTS

Executive Summary	2
Introduction	3
Project Overview	4
Key Features	4
Methodology	5
My contributions.....	8
Role: Full Stack Developer	8
Implementation and Code Except	8
Deployment	11
Future Features.....	11
Conclusion.....	13
Work Cited	15

EXECUTIVE SUMMARY

The project aimed to address the need for a centralized platform facilitating student collaboration and assistance in academic endeavors. Leveraging Node.js for the backend and React for the frontend, we developed a user-friendly web application deployed on Vercel and Netlify at <https://course-connect.netlify.app> and <https://courseconnect-delta.vercel.app> respectively, ensuring scalability and accessibility. Key features include class-specific question threads, real-time notifications, and user authentication, enhancing user experience. Despite challenges integrating third-party APIs and optimizing performance, we successfully delivered a functional platform. Moving forward, our focus lies on enhancing messaging capabilities to connect students for peer tutoring and assistance. Additionally, for those interested in exploring the code implementation of our project, we have made it available on GitHub at <https://github.com/orgs/The-MetaMinds/repositories>. This repository offers insights into our development process and allows for further examination of our project's structure and functionality.



INTRODUCTION

In today's rapidly evolving educational landscape, students often face challenges in accessing timely assistance and collaborative learning opportunities outside of traditional classroom settings. Recognizing this need, our team embarked on a software engineering project to develop a web-based platform aimed at facilitating peer-to-peer interaction and knowledge sharing among students.

The primary objective of our project was to create a centralized platform where students can post questions related to their coursework and engage in discussions with peers who can provide tutoring or assistance. While our initial implementation focused on core features such as user authentication, threaded discussions, and responsive design, we recognize that real-time messaging is a crucial component for fostering seamless communication and collaboration.

Despite facing constraints that prevented the implementation of real-time messaging in the current iteration of our platform, we remain committed to incorporating this feature in future iterations. By leveraging technologies such as WebSockets and asynchronous communication protocols, we aim to enhance the interactivity and responsiveness of our platform, thereby enriching the overall user experience and fostering more meaningful interactions among students.

In this report, we will provide a comprehensive overview of our software engineering project, including details on the methodologies used, implementation strategies, challenges encountered, and outcomes achieved. While real-time messaging remains a future-focused aspect of our project, we believe that documenting our progress and sharing our insights will contribute to the ongoing development and refinement of our platform.

PROJECT OVERVIEW

The Course-Connect Platform is a user-friendly tool exclusively tailored for Rutgers students, restricting access to non-Rutgers students during the initial phase. Our current scope focuses on providing a seamless environment for Rutgers

First Name:

Last Name:

Rutger's Email:

Password:

Confirm Password:

Major:

Contact Number:

Course Completed:

Computers and Programming 1

Programming Language Concepts

Computer Organization

Introduction to Pro

◀ ▶

Department:

Classes:

Computers and Programming 1

Programming Language Concepts

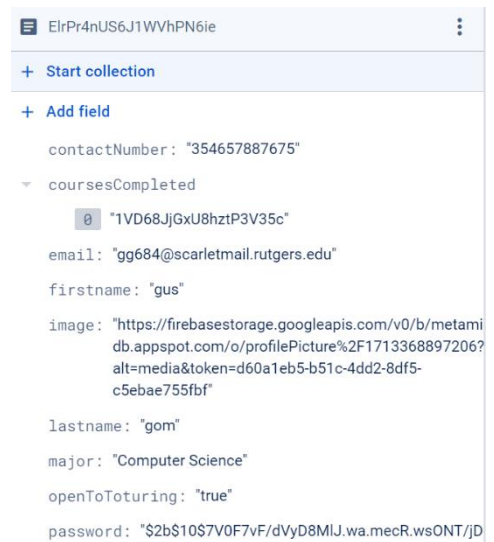
Operating Systems

Computers and Programming 2

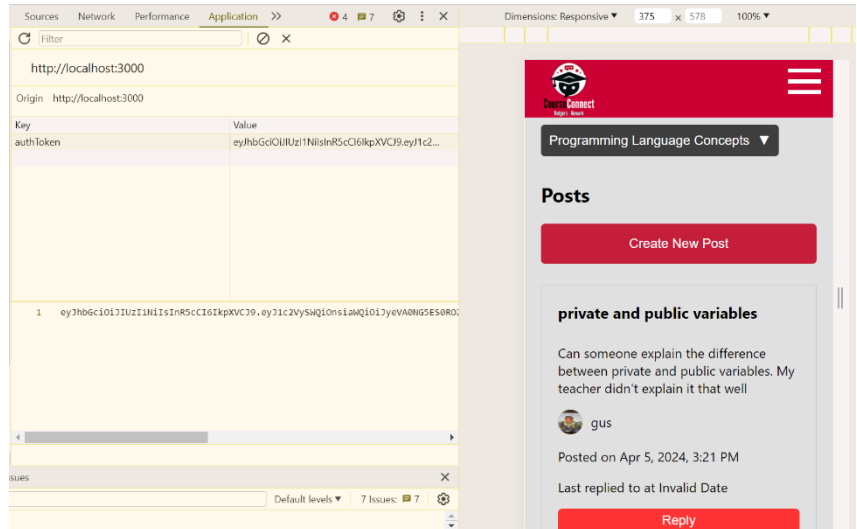
students to collaborate, seek assistance, and share knowledge in academic courses.

Key Features

- Data Security:** Implementing robust security measures to protect users' data, including encryption of passwords stored in the database.
- Dynamic Registration Page:** Creating a user-friendly registration page that dynamically captures users' information and courses taken, ensuring ease of use, as we can see when selecting the courses they have taken.
- Scalable Backend:** Developing a backend system that connects to the database, providing authentication, authorization, and scalability for future expansion. (we will focus more on this later)



4. **Token-Based Authentication:** Implementing token-based authentication for login and logout functionality, ensuring secure communication with the backend and proper authorization.
5. **Responsive Design:** Implementing a dynamic and responsive design that optimizes the platform's functionality for mobile devices, ensuring accessibility and usability across different screen sizes. As we see above the classes becomes a navigable button on top, when on phones or smaller screens.



METHODOLOGY

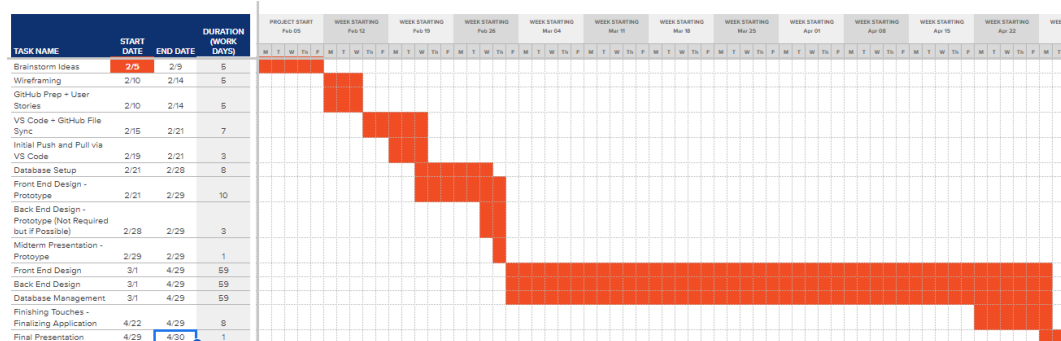
Before diving into the methodologies employed in the development of the Course-Connect Platform, it's crucial to establish the framework and approach guiding our project's execution. We recognized the significance of adopting a structured methodology to effectively manage our resources, streamline our development process, and ensure alignment with project goals and stakeholder expectations. In the following sections, we provide detailed insights into the methodologies, project management tools, testing strategies, and quality assurance practices employed throughout the development lifecycle of the Course-Connect Platform.

1. Software Development Methodology:

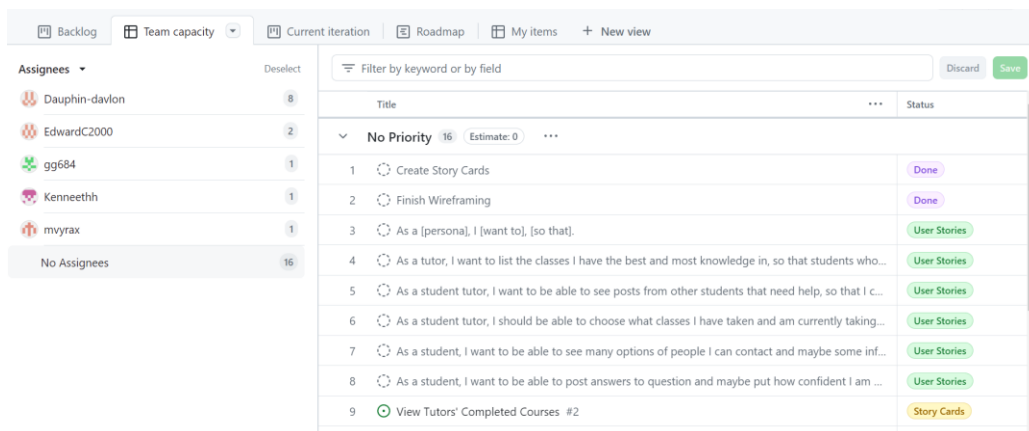
We implemented Agile principles, leveraging Scrum methodology to manage our project efficiently within the limited timeframe. Agile facilitated iterative development cycles, allowing for continuous feedback and adaptation to changing requirements. During our Scrum meetings, held regularly, team members discussed progress since the previous meeting, identified any encountered difficulties, and set goals for the upcoming week. This collaborative approach promoted transparency, accountability, and effective communication within the team.

2. Project Management Tools:

- a. **Gantt Chart:** We utilized a Gantt chart to visualize project tasks, timelines, and dependencies. This tool provided a clear overview of project progress, enabling us to identify potential bottlenecks and adjust our schedule accordingly.



- b. **Story Cards:** Prior to implementation, we created user story cards to capture user needs, requirements, and acceptance criteria. These story cards served as a valuable reference point throughout the development process, guiding our efforts to deliver features that aligned with user expectations and preferences.



- c. **Project Prioritization Framework:** Our project prioritization framework incorporated various factors, including task complexity, time requirements, value, and associated costs. This systematic approach guided our decision-making process, ensuring that we focused on high-priority tasks aligned with project goals and stakeholder expectations.

Rutgers CS491 - 2024 Group Project: Prioritization Framework

	A	B	C	D	E	F	G	H	I
1	Value		Time Criticality	Cost To Delay		Size Estimate		Weighted Shorted Job First	
2	Create buttons for each course	5	5	10		1		10	DONE
3	Link Post to the backend	8	8	16		3		5.333333333	DONE
4	Approve @Rutgers emails and only allow	5	5	10		2		5	DONE
5	View your passwords while typing it	3	1	4		1		4	
6	Invalid email/password error during login	3	1	4		1		4	DONE
7	Being able to view someone else's profile	5	3	8		5		1.6	DONE
8	Login with gmail	1	1	2		2		1	
9	Profile Picture	1	1	2		3		0.666666667	DONE
10	Messaging between users	2	1	3		8		0.375	
11									
12									
13									
14									
15									

3. Collaboration Tools:

- GitHub:** We utilized GitHub for version control, issue tracking, and collaboration among team members. This platform facilitated seamless code management, code reviews, and discussion threads, enhancing team productivity and coordination.
- GroupMe and Discord:** Communication among team members and with the project manager occurred via GroupMe and Discord. These platforms provided real-time messaging and collaboration features, enabling efficient communication and decision-making.
- PowerPoint:** For presentations and meetings, we used PowerPoint to create visually engaging slides and effectively communicate project updates, milestones, and findings.

4. Testing and Quality Assurance:

In addition to internal functionality testing and user testing involving friends and family, we plan to expand our testing efforts in the future. This includes testing with strangers through targeted advertisement and marketing campaigns aimed at students. Furthermore, we intend to incorporate a feedback page within the platform to gather feedback directly from students, enabling us to identify areas for improvement and enhance overall user satisfaction.

By employing these comprehensive methodologies, we successfully managed our project, ensured quality and user satisfaction, and laid the groundwork for future enhancements and iterations.

MY CONTRIBUTIONS

Role: Full Stack Developer

As the Full Stack Developer on our team, my unique expertise in connecting backend and frontend development, along with your proficiency in using GitHub, deployment processes, React, and Node, makes me an invaluable asset to our project. I play a central role in ensuring the seamless integration of all components of our application, from user interface to server-side logic, database management, and deployment. My primary responsibility is to bridge the gap between frontend and backend development, ensuring that our application functions smoothly and efficiently. I will collaborate closely with other team members (frontend and backend developers), leveraging my skills to bring together the best of both worlds.

Implementation and Code Excerpt

I worked on the full backend. This is the entry point of the backend. We use express for the routing and used “cors” for the allowing our front end have permission to communicate with the backend.

```
import { db } from "../firebase.js"; // Assuming your Firebase module file is named firebase
import { collection, addDoc, getDoc, deleteDoc, query, where, setDoc, doc, getDocs, limit } from "firebase/firestore";
import bcrypt from "bcrypt";
import generateAuthToken from "../auth.js";
import _ from "lodash";
import { auth } from "../middleware/auth.js"

router.post('/validate', auth, (req, res) => {
  // If the request reaches here, it means the token is valid
  res.status(200).send({ message: 'Token is valid' });
});

router.post('/', async (req, res) => {
  const { email, password } = req.body;

  try {
    // Check if user exists with the provided email
    const user = await getUserByEmail(email);

    // If user doesn't exist, return error
    if (!user) {
      return res.status(404).send('User not found');//change it to 400 later
    }

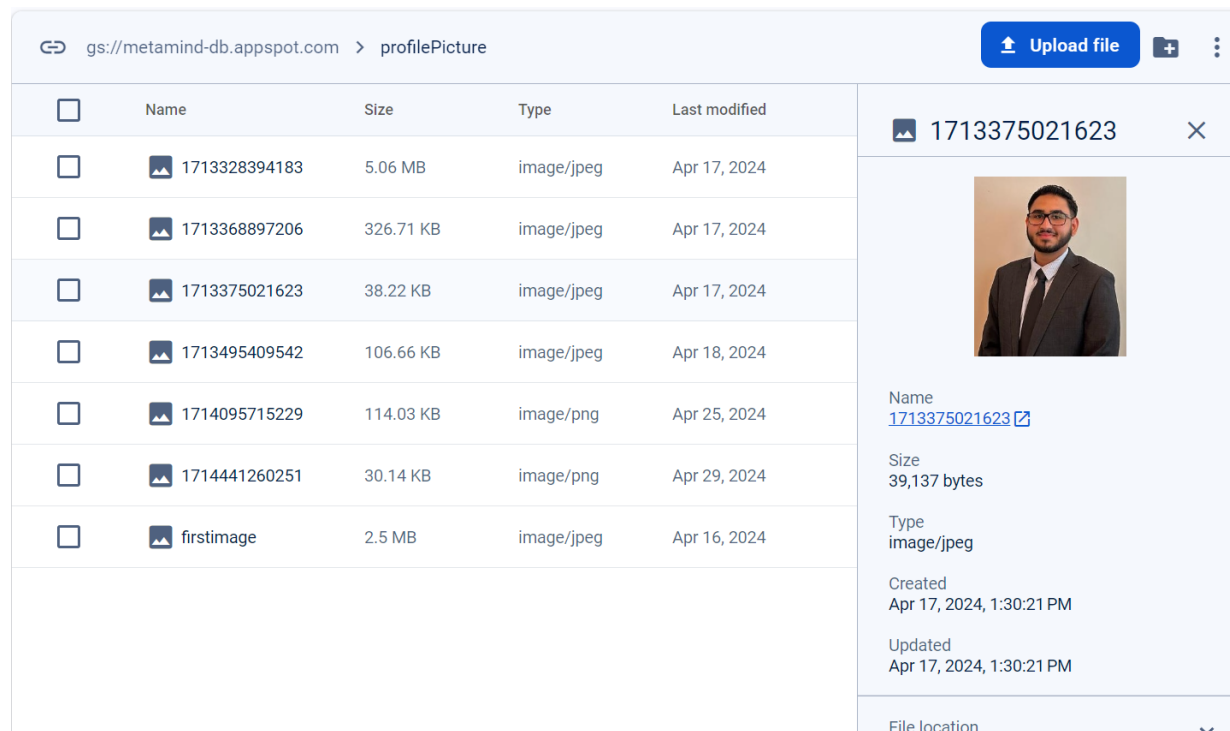
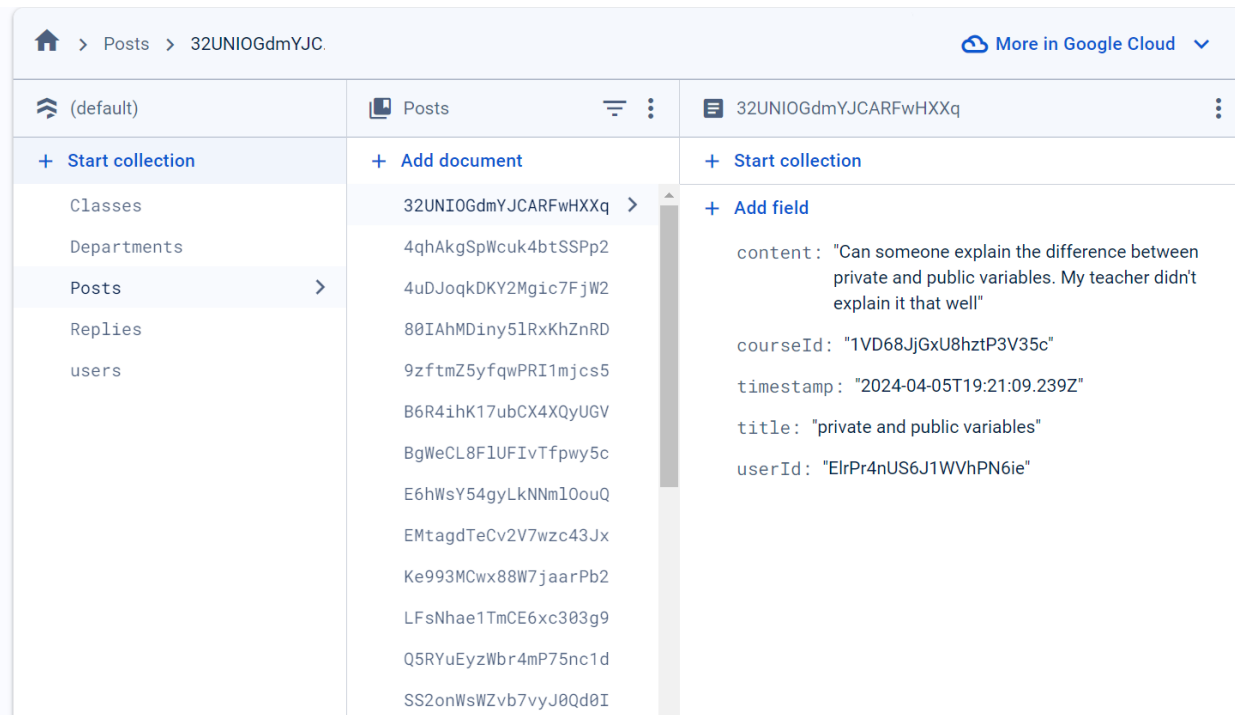
    // Compare the provided password with the hashed password stored in the database
    const isPasswordValid = await bcrypt.compare(password, user.password);

    // If passwords match, user is authenticated
  }
});
```

```
$ index.js > ...
1  import express from 'express';
2  import cors from 'cors';
3  const app = express();
4
5  import {router as learning} from './routes/learning.js'
6  import {router as users} from './routes/users.js'
7  import {router as departments} from './routes/departments.js'
8  import {router as courses} from './routes/courses.js'
9  import {router as posts} from './routes/posts.js'
10 import {router as auth} from './routes/auth.js'
11 import {router as replies} from './routes/replies.js'
12
13 app.use(cors())
14 app.use(express.json()); //middleware
15 app.use("/api/courses", courses)
16 app.use("/api/users", users)
17 app.use("/api/departments", departments)
18 app.use("/api/posts", posts)
19 app.use("/api/auth", auth)
20 app.use("/api/replies", replies)
21
22
23
```

We divided the functionality so that each part had a single as a single responsibility. Therefore, there is a module that has code for authenticating and authorizing users (auth), and another module for department, another module for course, another module for posts, another module for replies, and another module to get users from the database. For more on how we implemented the backed you can check our github: <https://github.com/The-MetaMinds/courseconnect-backend>

Also, I worked on the database, ensuring that the information stores on the database from the backend. In this part, I used firebase storage for pictures (profile pictures) and used firestore database for the database. Below you will see some pictures from the database.



Also, I worked on the front end making sure that it connects with the backend so information go from the front end to the backend and then from the backend to the database and back and forth.

I will show some of the things I did in the front end. I worked on the making the profile picture connects and the posts so the posts you had can be added to the backend and can be retrieved.

```
const [courses, setCourses] = useState([]);
const [posts, setPosts] = useState([]);
const [replies, setReplies] = useState([]);
const [showNewPostForm, setShowNewPostForm] = useState(false);
const [newPostTitle, setNewPostTitle] = useState('');
const [newPostContent, setNewPostContent] = useState('');
const [activeClass, setActiveClass] = useState(null); //working on this
const [dropdownVisible, setDropdownVisible] = useState(false);

useEffect(() => {
  const fetchCoursesFromBackend = async () => {
    try {
      const response = await fetch('https://courseconnect-delta.vercel.app/api/depar');
      if (!response.ok) {
        throw new Error('Failed to fetch courses');
      }
      const data = await response.json();
      setCourses(data);
    } catch (error) {
      console.error('Error fetching courses:', error);
    }
  };
  fetchCoursesFromBackend();
}, [departmentID]);

const handleReply = async (postId, replyContent) => {
```

```
const createNewPost = async () => {
  setShowNewPostForm(true);
};

const handleFormSubmit: (e: any) => Promise<void>


const handleFormSubmit = async (e) => {
  e.preventDefault();
  if (newPostTitle.trim() !== '' && newPostContent.trim() !== '') {
    const newPost = {
      title: newPostTitle,
      content: newPostContent,
      username: userId,
      timestamp: new Date().toISOString(),
      course: activeClass.id
    };
    try {
      const response = await axios.post('https://courseconnect-delta.vercel.a
      console.log('Post sent successfully:', response.data);
      setPosts([response.data, ...posts]);
      setNewPostTitle('');
      setNewPostContent('');
      setShowNewPostForm(false);
    } catch (error) {
      console.error('Error while posting:', error.message);
    }
  } else {
    alert('Please enter a title and content for your post.');
```

For full implementation of the front end code, you can check our github: <https://github.com/The-MetaMinds/metaminds-courseconnect>

Deployment

I will show some deployment of what we have.

User Profile



First Name:
gus

Last Name:
gom


Email:
gg684@scarletmail.rutgers.edu

Major:
Computer Science

Contact Number:
354657887675

Courses Completed:
Programming Language Concepts


Open to Tutoring:
No

 [Registration](#) [Login](#)

[Sign Up](#)

Login

[Sign In](#)

 [Profile](#) [Log Out](#)

Courses

[Computers and Programming 1](#)

[Programming Language Concepts](#)

[Operating Systems](#)

[Computers and Programming 2](#)

[Software Engineering](#)

[Computer Organization](#)

[Intensive Programming](#)

[Cloud Computing](#)

Posts

[Submit Post](#)

Using while loop

Create supermarket program using only array lists and if applicable use functions too. When you run the program it should display something like this Product Name Quantity Price Amount ----- Milk 2 2.50 5.00 Bread 3 3.00 9.00 ----- Total Price 14.00 You will create array lists for each variable and get their amount from quantity and price. Use while loop to enter the data, remove data, insert by index, delete by index, update by name etc. When you decide to end the loop it should show total list of products, show highest and lowest price, search by name, and total amount of prices and average of all amount

FUTURE FEATURES

For future work, there are several key enhancements and features that we plan to implement to further improve the Course-Connect Platform:

1. **Code Refactoring:** We aim to conduct a comprehensive code review and refactoring process to enhance code quality, readability, and maintainability. This includes optimizing existing code structures, identifying, and resolving technical debt, and implementing best practices to ensure scalability and performance.
2. **Messaging Functionality:** Adding messaging capabilities to the platform will enable seamless communication between students and tutors, facilitating real-time interaction and support. This feature will enhance collaboration and foster a more dynamic learning environment for users.
3. **User Profile Editing:** Providing users with the ability to edit their profiles will enhance personalization and customization options. Users can update their information, preferences, and settings to better tailor their experience on the platform to their individual needs and preferences.
4. **Enhanced Learning Page:** We plan to create a more descriptive and informative learning page that provides users with comprehensive resources, tutorials, and guides to aid in understanding and navigating the platform effectively. This will improve user engagement and satisfaction by providing valuable educational content.
5. **Admin Dashboard:** Implementing an admin site will empower administrators to manage courses, departments, users, and content more efficiently. Admins can add, edit, and delete courses and departments, as well as oversee platform activity and user interactions.
6. **Likes and Filtering:** Introducing likes and filtering functionality to posts will enable users to express appreciation for helpful content and easily discover popular or relevant posts. This feature will enhance user engagement and promote the visibility of valuable contributions within the platform.
7. **User Role Differentiation:** Assigning different colors or visual cues to distinguish between users who tutor and those who do not will improve user experience and facilitate easier identification of tutors. This differentiation will streamline the process of seeking assistance and connecting with knowledgeable users.
8. **Marketing and Publicity:** To increase platform usage and user engagement, we plan to launch targeted marketing campaigns and publicity efforts to raise awareness and attract more students to the Course-Connect Platform. This includes leveraging social media,

campus outreach, and partnerships with educational institutions to promote the platform effectively.

By implementing these future enhancements and features, we aim to further elevate the Course-Connect Platform, enrich the user experience, and continue fostering a collaborative and supportive learning environment for students.

CONCLUSION

In conclusion, the development and implementation of the Course-Connect Platform represent a significant milestone in our efforts to enhance collaborative learning and academic support among Rutgers students. Through the utilization of Agile methodologies and Scrum principles, we successfully navigated the challenges of a tight timeframe, ensuring rapid development and iterative improvement of the platform. The incorporation of user-friendly features such as secure data handling, dynamic registration, and token-based authentication underscores our commitment to delivering a seamless and intuitive user experience.

As we reflect on our journey, it's evident that our project's success would not have been possible without the dedication and collaboration of our team members, who worked tirelessly to bring our vision to life. We also extend our gratitude to our project manager and stakeholders for their guidance, support, and valuable feedback throughout the development process.

Looking ahead, we are excited about the future possibilities for the Course-Connect Platform. Our plans to refactor the codebase, implement messaging functionality, enhance user profiles, and introduce an admin dashboard will further elevate the platform's functionality and usability. Additionally, initiatives such as the creation of a comprehensive learning page, integration of likes and filtering features, and differentiated user roles will contribute to a more enriching and dynamic user experience.

As we embark on the next phase of development, we remain committed to our mission of fostering collaboration, communication, and knowledge sharing among Rutgers students. With continued dedication, innovation, and collaboration, we are confident that the Course-Connect Platform will continue to evolve and positively impact the academic journey of students for years to come.

WORK CITED

- How to Write the Best User Stories with Story Cards. Leandog Blog. Leandog. N/A.
<https://www.leandog.com/blog/how-to-write-the-best-user-stories-with-story-cards>.
- "Surge." Surge. Surge. N/A. <https://surge.sh/>.
- "GitHub." GitHub. GitHub. N/A. <https://github.com/organizations/plan>.
- "Start a New React Project – React." React Documentation. React. N/A.
<https://reactjs.org/docs/create-a-new-react-app.html>.
- "An introduction to the Django ORM." Opensource.com. Opensource.com. N/A.
<https://opensource.com/article/17/11/django-orm>.
- "Firebase Documentation." Firebase Documentation. Firebase. N/A.
<https://firebase.google.com/docs/web/setup>.