

# Teamup NEO Project Report

## Stakeholders

Professor David Kebo, Shashankit Thakur, and Florencia Mangini from Teamup Apps for Good Program.

## Client Requirements and Implementation Summary

The client envisioned Neo as a centralized web application designed to elevate students' experience participating in the Teamup Apps for Good program. The platform should empower students with a user-friendly dashboard that enables them to track project milestones, monitor their progress toward earning a certificate of completion, and seamlessly access information about their team members and mentors. Additionally, students should be able to stay informed about important deadlines, while easily navigating program resources and materials. For program administrators and mentors, Neo must provide tools to efficiently manage milestones, upload essential resources, and monitor student engagement and progress. By delivering a cohesive and intuitive solution, Neo aimed to streamline the onboarding process, enhance project management and accountability, and drive higher student engagement and program completion rates.

The Neo application was implemented using a modular and user-centric design approach, ensuring that it met the diverse needs of students, mentors, and administrators while maintaining scalability and efficiency. It is designed to provide tailored role-specific views for its stakeholders, ensuring that students, mentors, and administrators have access to features aligned with their needs and responsibilities. For students, the platform offers an intuitive dashboard where they can monitor project milestones, manage their assigned tasks, track progress toward certification, and access detailed information about their team members and mentors. Students are also notified of important announcements and have seamless access to program resources, which fosters organization and accountability throughout the program.

Mentors and admins, on the other hand, are equipped with tools to track the progress of their assigned teams, provide guidance, and access resources to support their mentoring efforts effectively. They also benefit from robust management features

integrated into Neo, which address key program needs. The application enables milestone management, allowing administrators to define and update project goals, track team progress, and ensure alignment with the program's timeline. Resource management tools facilitate the upload and organization of essential materials for students and mentors, while team management features streamline the onboarding of participants and assignment of mentors to teams. The milestone and task management enables mentors to create and manage milestones and tasks associated with it. These role-specific functionalities ensure that Neo provides a cohesive and efficient solution, meeting the diverse requirements of all its stakeholders.

## User Stories

### **Sprint 1 Stories**

#### **1) Setup Jira and Slack**

**Story Points:** 1

**Implementation Status:** Completed

**Tasks:**

- Create a JIRA account and Slack workspace
- Configure user roles and permissions in JIRA
- Configure JIRA Board for Sprint I
- Setup Slack communication channel for Sprint I
- Invite TA and Instructor to Slack and JIRA

#### **2) Setting up Git Repository**

**Story Points:** 1

**Implementation Status:** Completed

**Tasks:**

- Create a shared workspace/repository for the backend application
- Configure repository settings to require a minimum of 2 reviewers to merge pull request merges
- Create a simple Ruby on Rails application with the following version dependencies:
  - 1) Rails - v7.2.1
  - 2) Ruby - v3.3.4
  - 3) Bundle - v2.5.11

### 3) Heroku Deployment

**Story Points:** 1

**Implementation Status:** Completed

**Tasks:**

- Configure Heroku deployment changes to Git Repository.
- Add database to Heroku deployment.
- Deploy application.

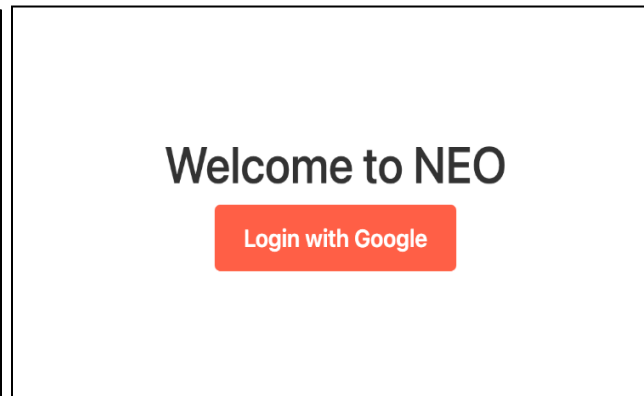
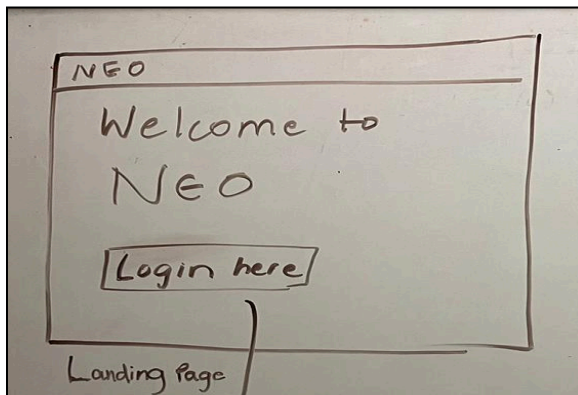
### 4) Feature Landing Page

**Story Points:** 3

**Implementation Status:** Completed

**Tasks:**

- Build a Landing page in the preferred Front end framework or Simple CSS/HTML
- Add a Login button to the Landing page
- Deploy Landing Page on Heroku



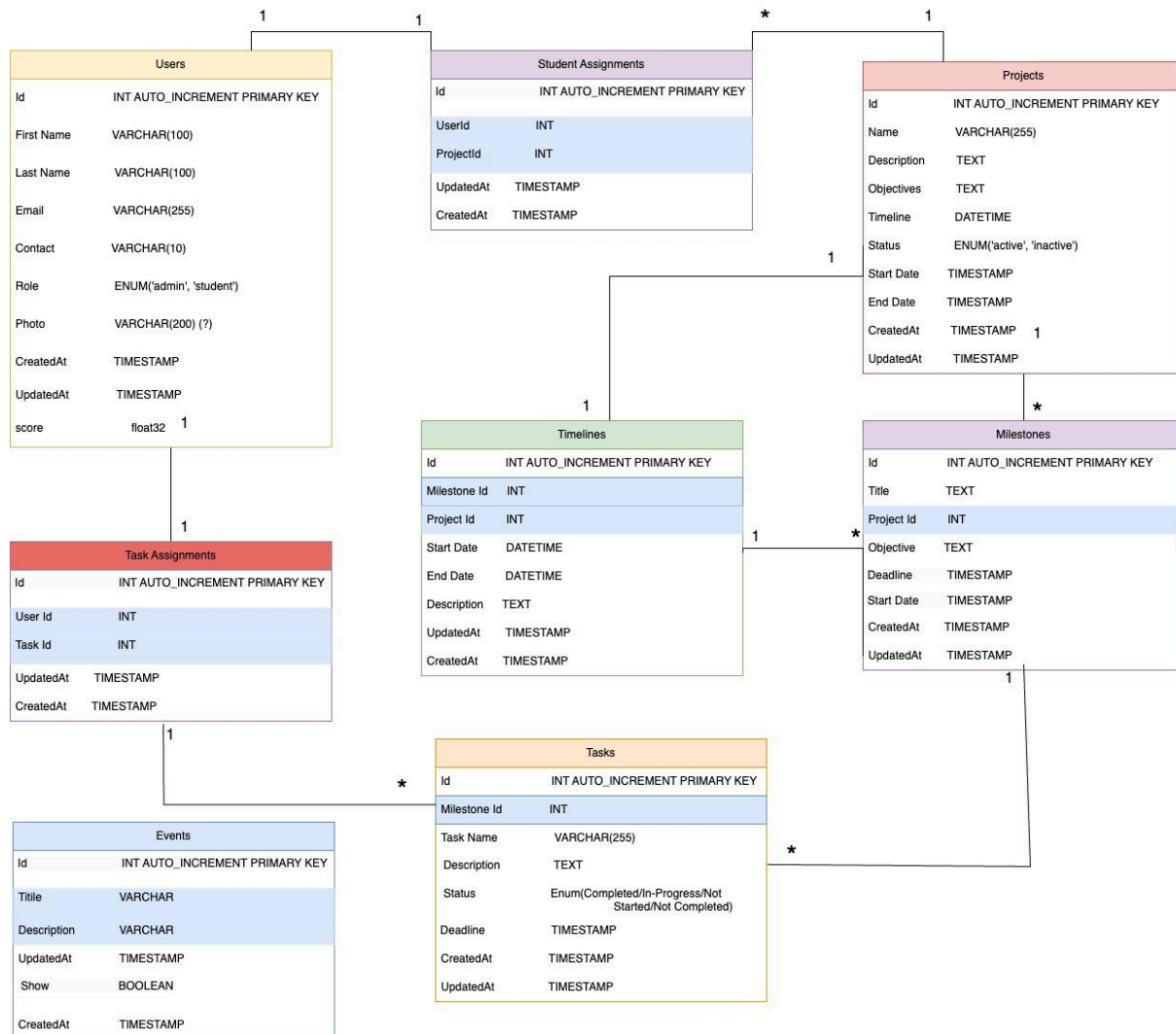
## 5) Feature Database Modeling

**Story Points: 3**

**Implementation Status:** Completed

### Tasks:

- Requirement Gathering
- Identification of Identities and Relationships
- Database Schema Design
- Deciding on an appropriate database to use
- Implementing initial models with association and validation
- Review and Iterate



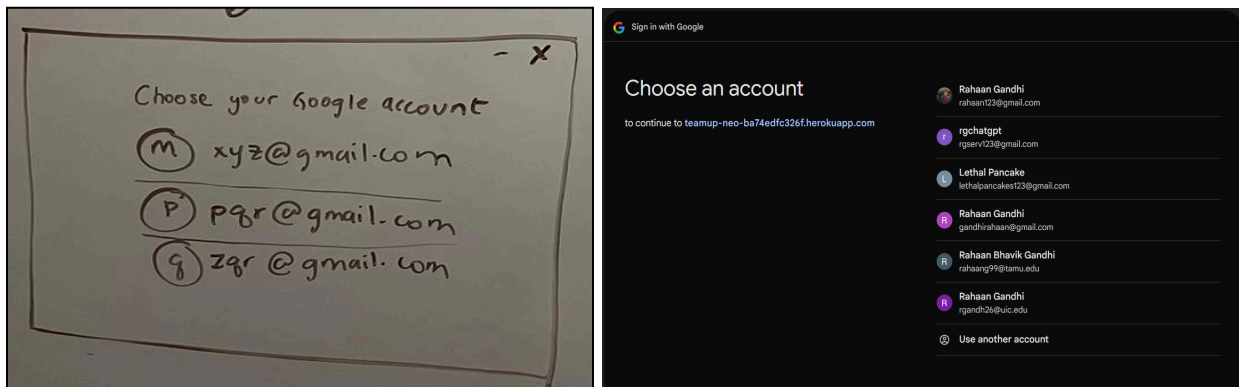
## 6) Feature Login using Google OAuth

**Story Points:** 3

**Implementation Status:** Completed

**Tasks:**

- Google OAuth client creation.
- Configuring omniauth in the rails application.
- Adding a callback to process Google's response.
- Handling session and its validation on login.



## 7) Feature Role-based Dashboard View and Basic Navigation

**Story Points:** 3

**Implementation Status:** Completed

**Tasks:**

- Set up routes and placeholder views for the Dashboard, Project hub, Settings, and Project management page
- Implement role-based access control for the Dashboard, Project hub, and Project management pages.

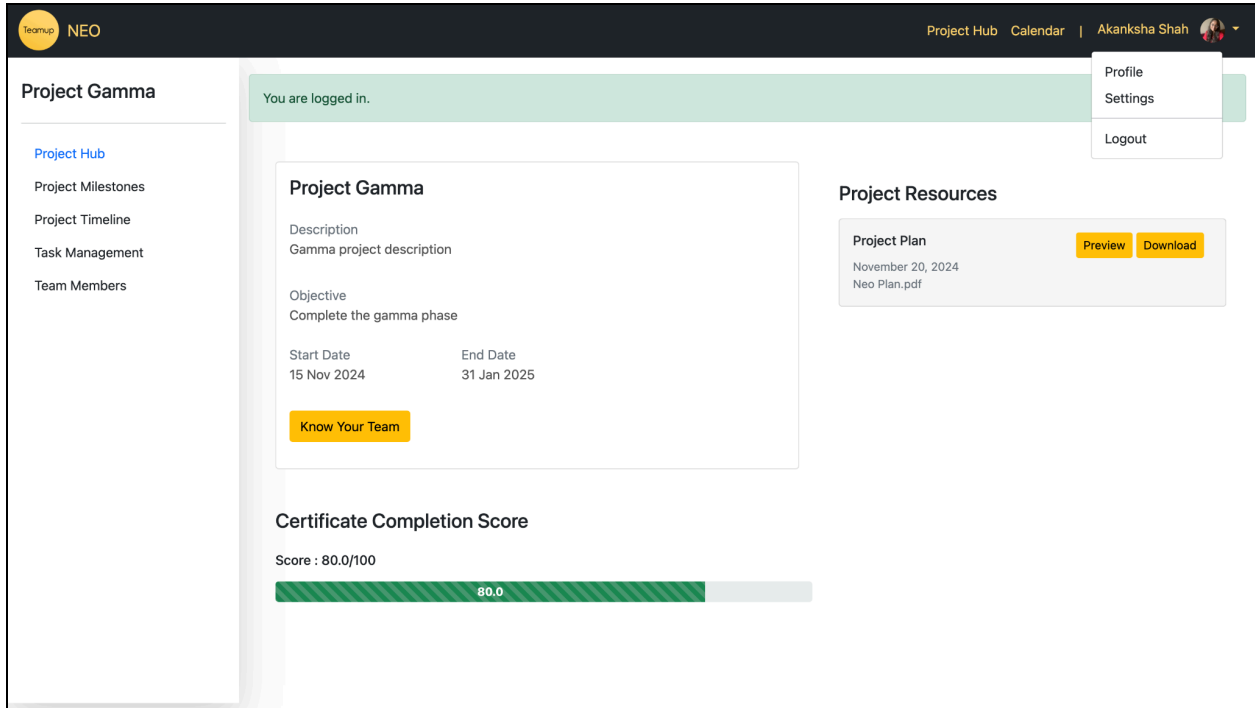
## 8) Feature Logout

**Story Points:** 1

**Implementation Status:** Completed

**Tasks:**

- Implement logout functionality to clear user sessions
- Add a Logout button throughout different pages
- Ensure proper session destruction and redirection after logout



## Sprint 2 stories

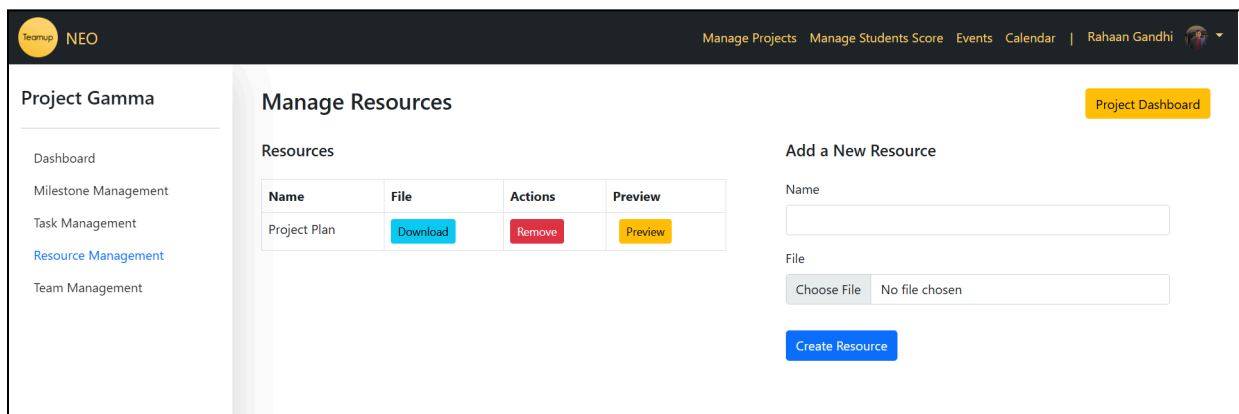
### 9) Resource Management

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Decide on storage (Github/S3 bucket)
- UI for Upload/View/Delete the resource
- Logic to handle and store



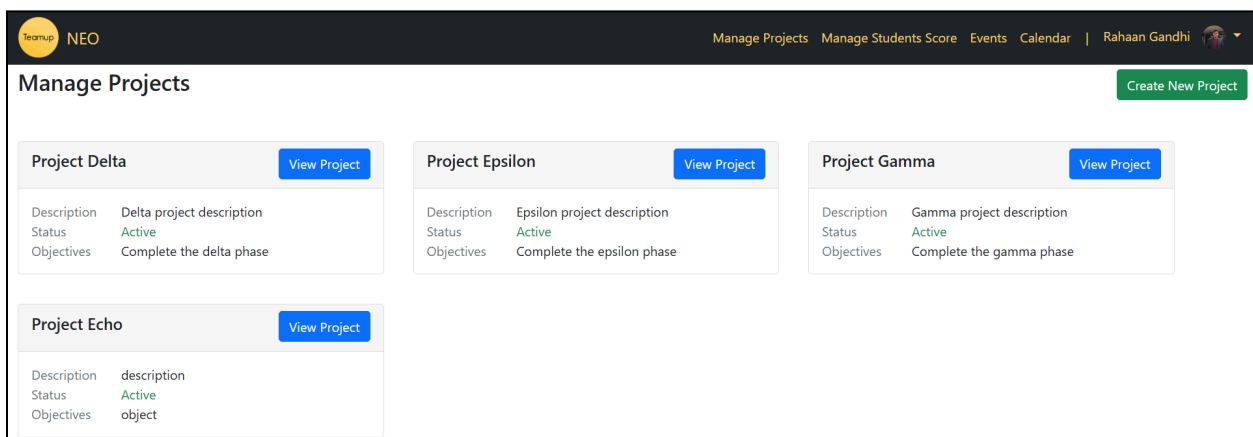
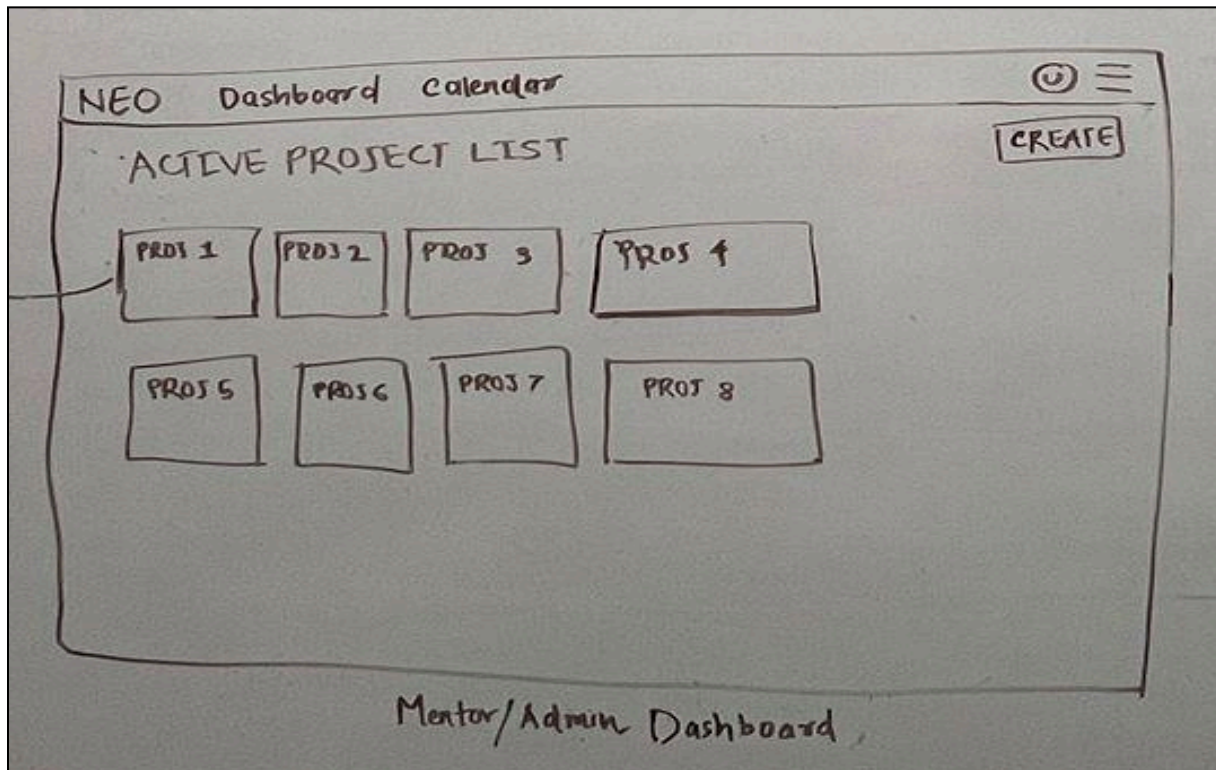
## 10) Display of Projects

Implementation Status: Completed

Story Points: 5

Tasks:

- Create a Project Display Dashboard
- Fetch and Display Projects in the Dashboard



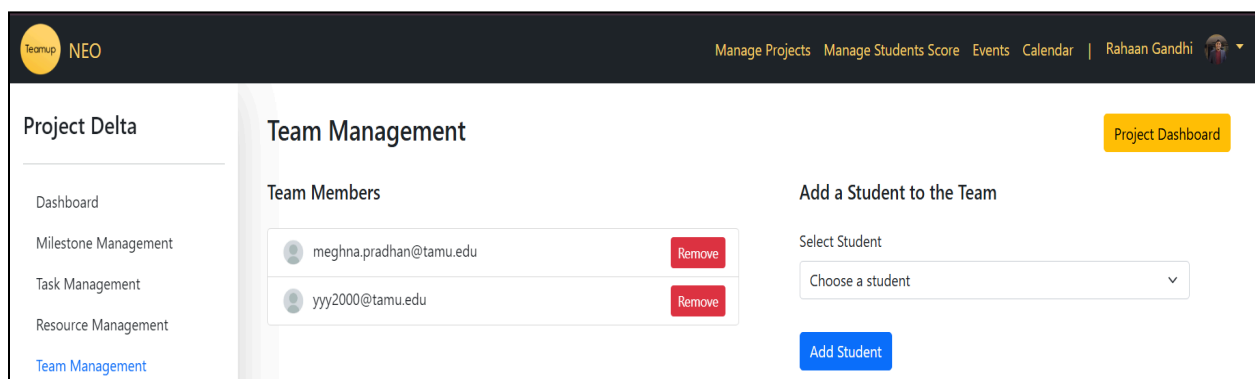
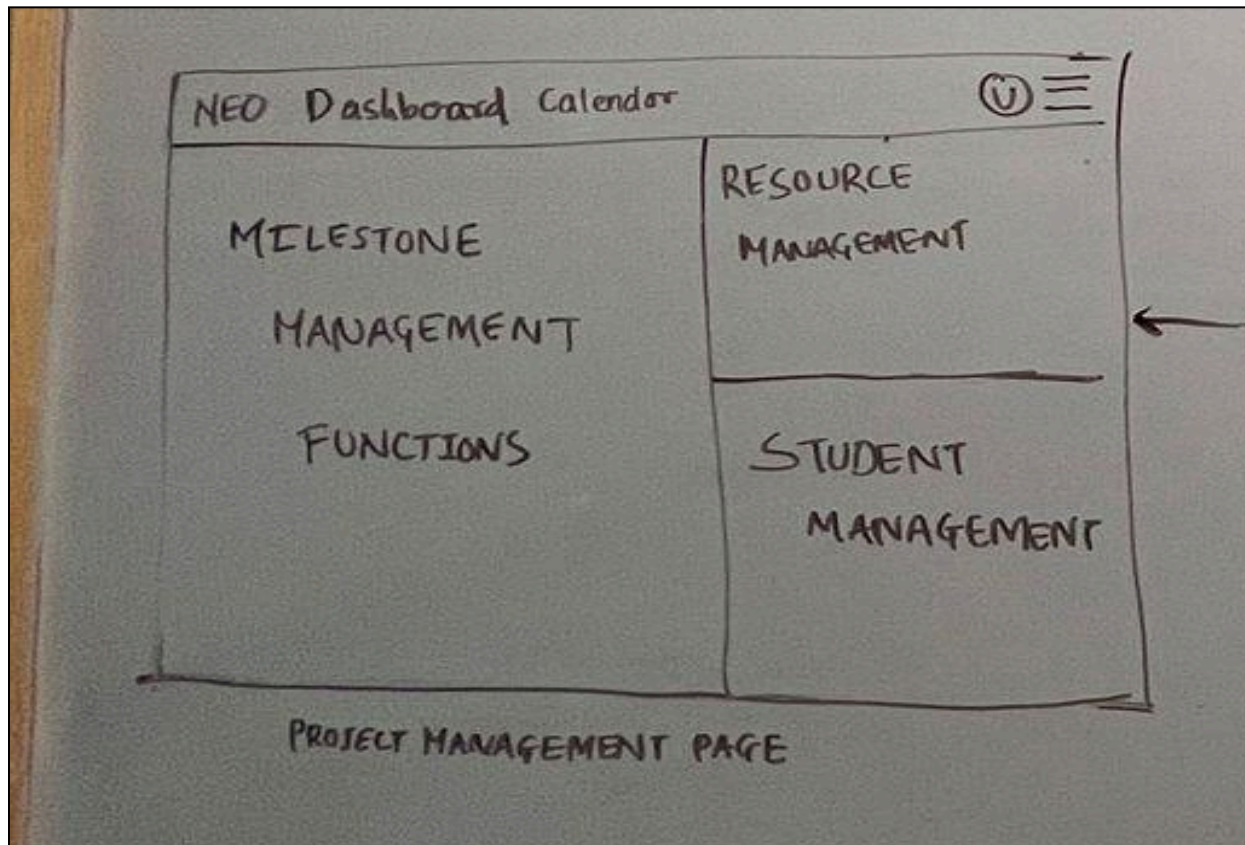
## 11) Student management

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Write operation for creating a team
- Write operation for Add/remove students





## 12) Bug fixes from Sprint 1 and increasing test coverage

**Implementation Status:** Completed

**Story Points:** 3

**Tasks:**

- Fix the dropdown issue with Bootstrap
- Raise cucumber and RSpec coverage to 100% or as high as possible
- Hiding of Link as per roles

## 13) DB table creation

**Implementation Status:** Completed

**Story Points:** 2

**Task:** Create the DB tables for all necessary data and relations

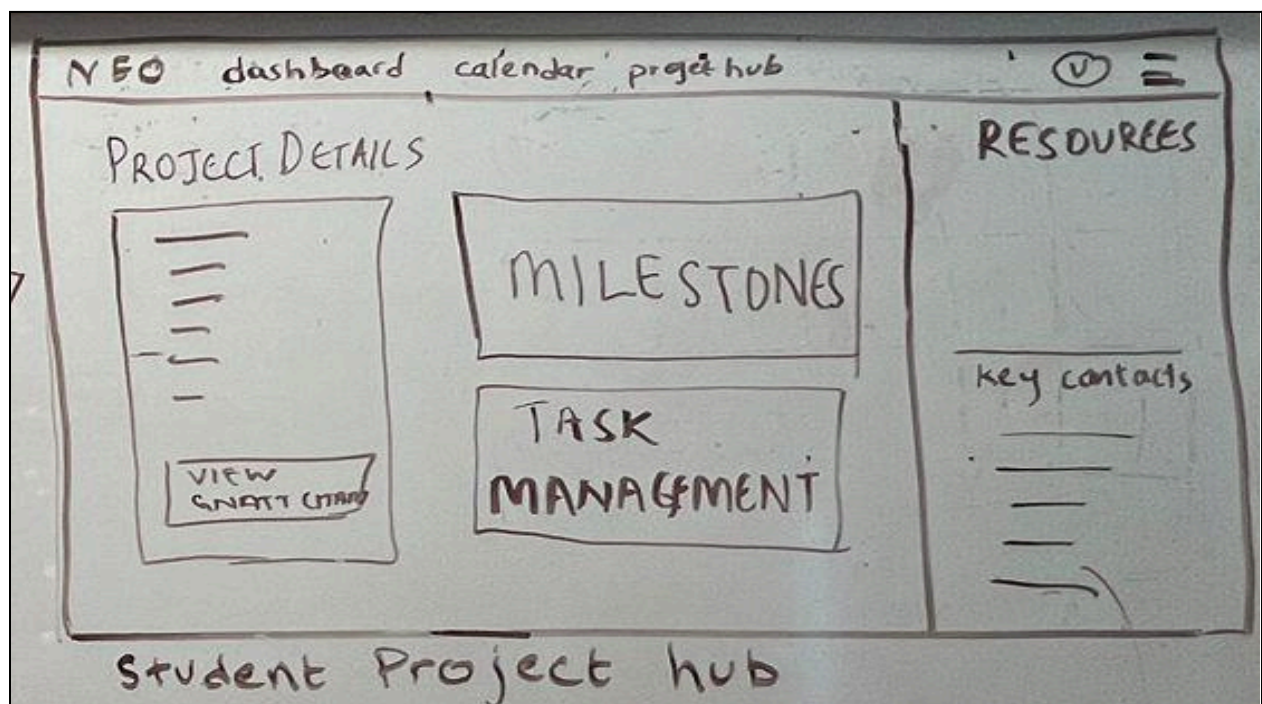
## 14) Project overview section

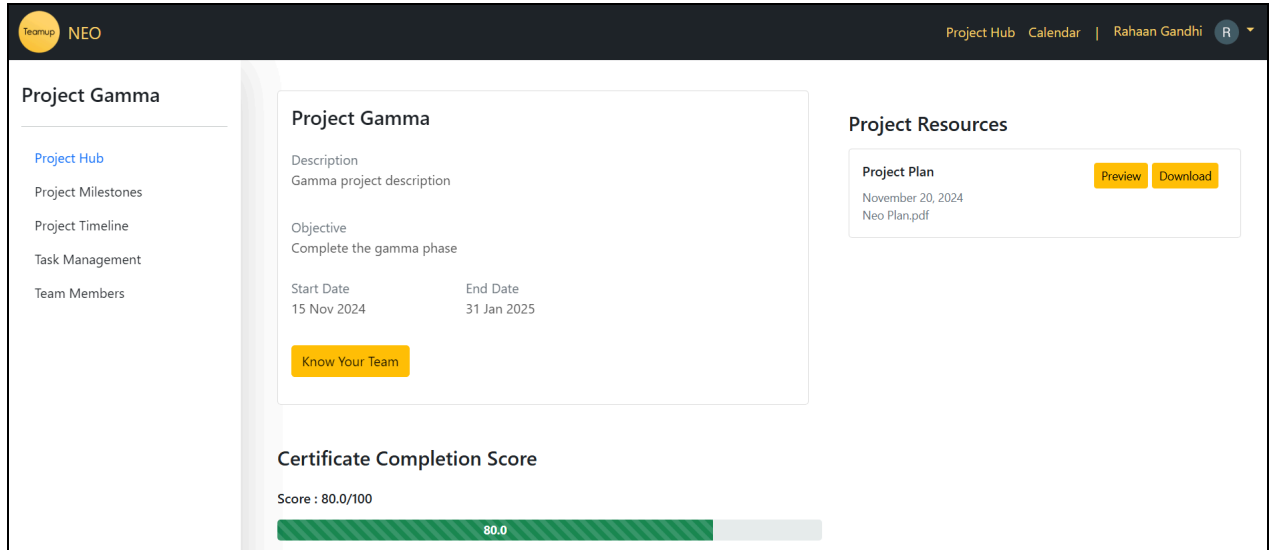
**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Create a view and controller for the student dashboard
- Write cucumber/RSpec tests for the actions





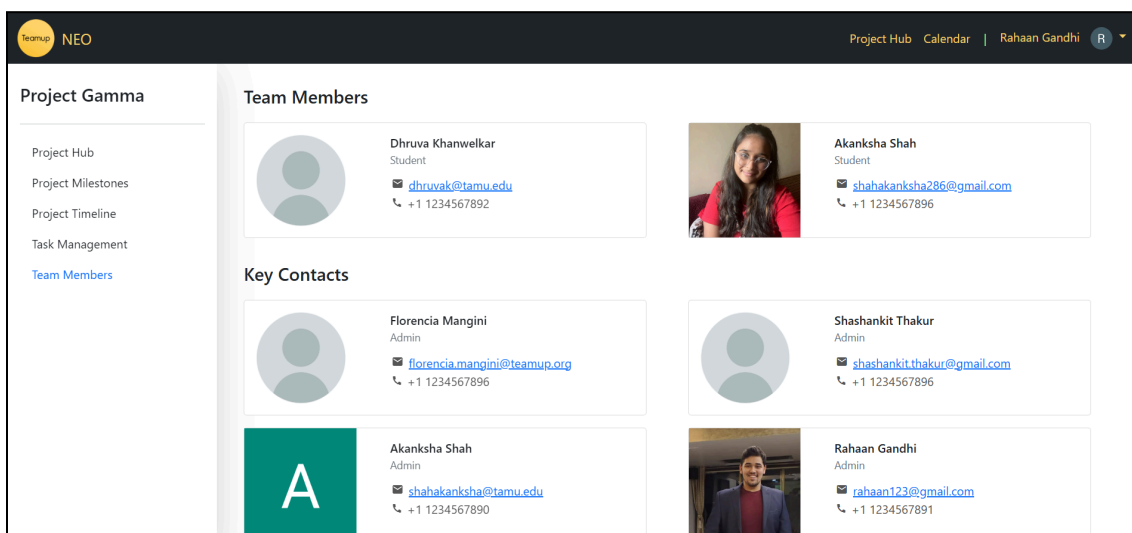
## 15) Display Team Members and Student Information on the Student Dashboard

Implementation Status: Completed

Story Points: 5

Tasks:

- Fetch the students' project team member information (name, email, contact, and profile photo)
- Fetch information on mentors associated with the project. (name, email, contact, and profile photo)
- Display the information on the student dashboard page.
- RSpec and Cucumber coverage for the feature.



## 16) Creation of projects (Create and indexing of projects)

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Create controller and logic for creating and indexing projects
- UI for project creation table
- Tests to cover project creation scenarios

The screenshot shows the 'Create New Project' form in the NEO application. The form is located on the right side of the screen, with a close button (X) in the top right corner. The form fields are as follows:

- Project Name:** A text input field.
- Project Description:** A text area.
- Project Objectives:** A text area.
- Project Start Date:** A date input field with a calendar icon, showing the format 'mm/dd/yyyy'.
- Project End Date:** A date input field with a calendar icon, showing the format 'mm/dd/yyyy'.
- Project Status:** A dropdown menu with 'Active' selected.

On the left side of the screen, there is a 'Manage Projects' section. It contains a table with two projects: 'Project Delta' and 'Project Echo'. Each project has a 'View Project' button. The table columns are 'Description', 'Status', and 'Objectives'.

Project	Description	Status	Objectives
Project Delta	Delta project description	Active	Complete the delta phase
Project Echo	description	Active	object

## Sprint 3 Stories

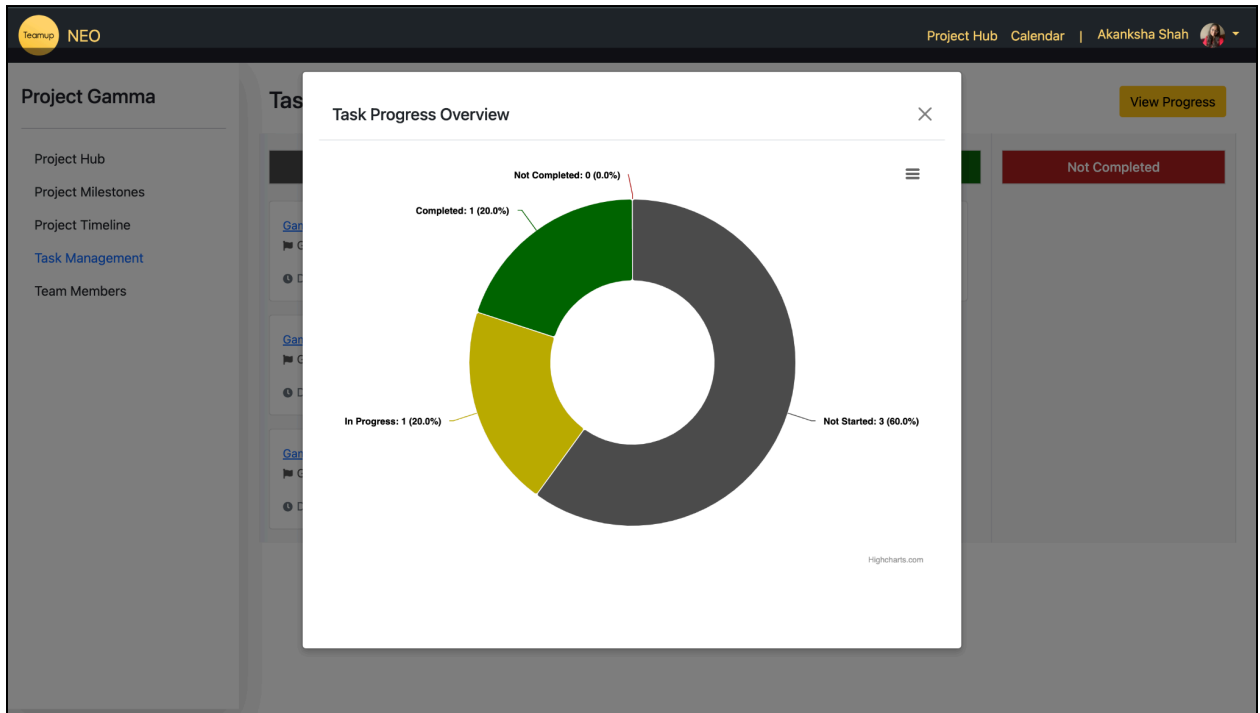
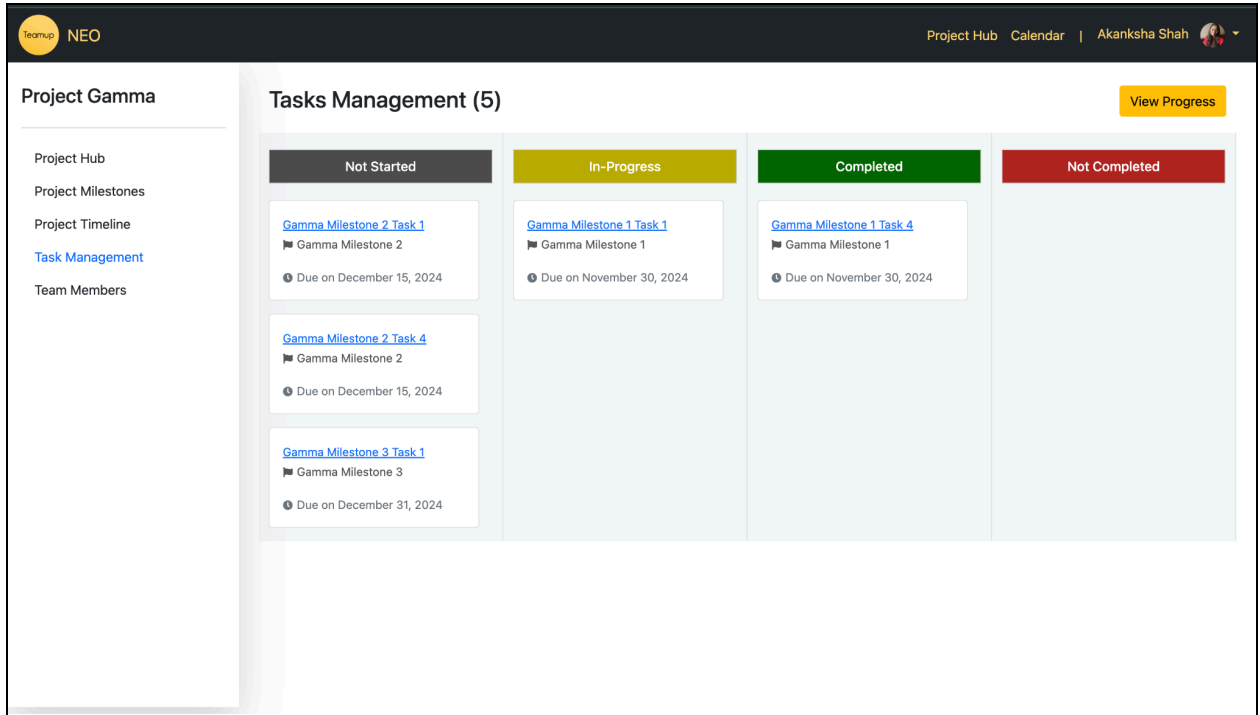
### 17) Student Tasks Management

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Display the list of the tasks assigned to a particular student by the admin.
- Display task information like task name, description, start date, end date, student assignment, and status for a particular task.
- Segregate the tasks based on the milestones.
- Add functionality to mark the tasks as complete.
- Implement charts to depict the student's progress.
- Deploy the application to Heroku.

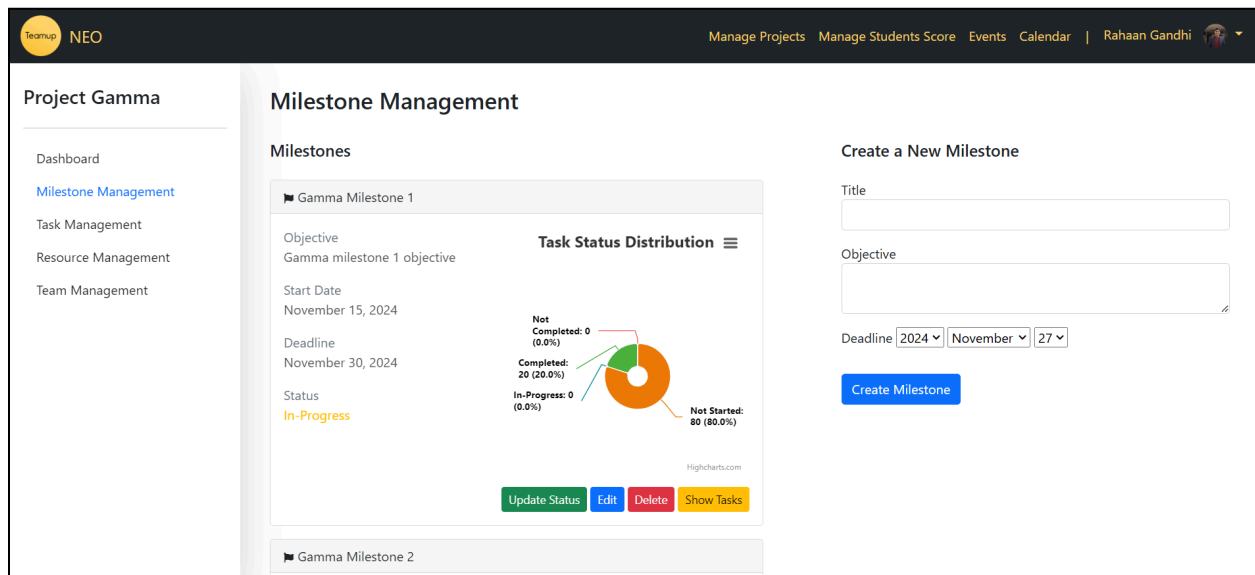


## 18) Admin Milestone Management (CRUD)

**Implementation Status:** Completed

**Story Points:** 5

- Implement CRUD operations from Admin for Milestones.
- The milestone tab for each project should display the milestones - Start date and end date, Description, and Status.
- The Admin should be able to mark milestones as complete.
- Progress view for milestones should include a chart or visual aid.



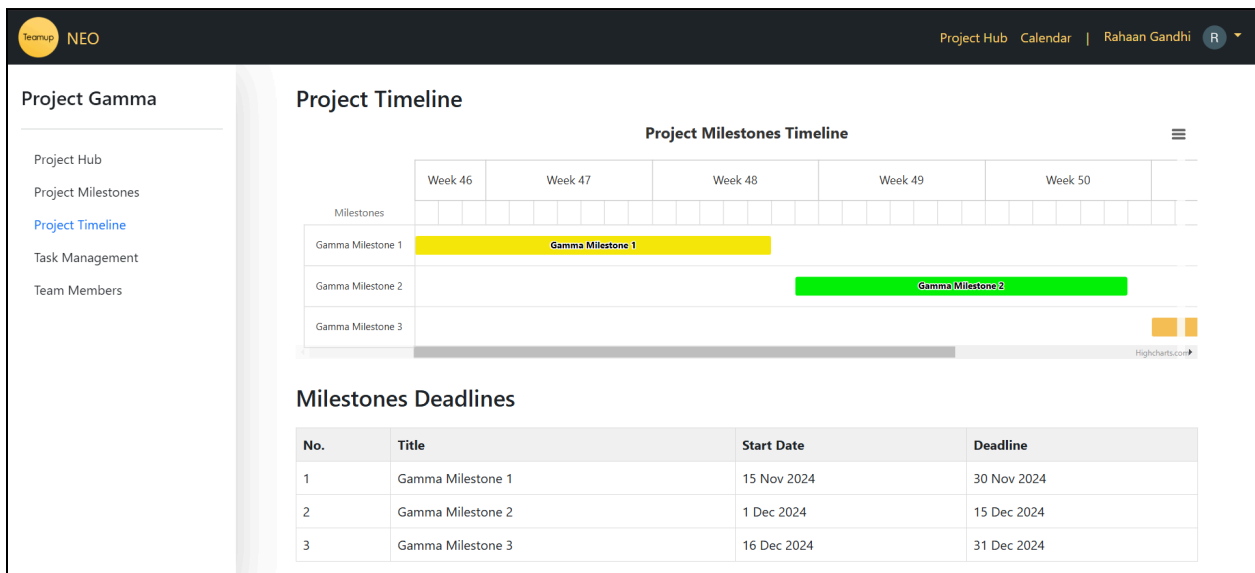
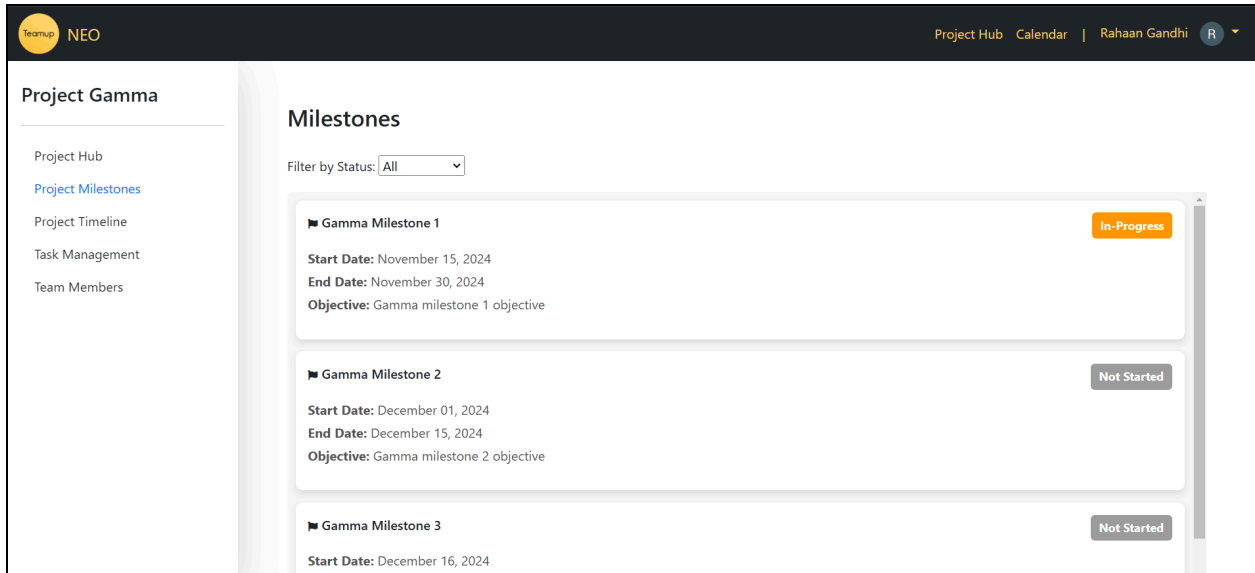
## 19) Student Milestones Management

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Create a scrollable list of the Milestones with all related information (Milestone name, status-completed/in progress/not started, start/end date, Description, and Objective) and display it on the Project Hub for students.
- The list should also allow for filtering based on the status of milestones.
- Implement a Gantt chart;
- Implementation of a table, next to the Gantt chart, with Milestone details such as Milestone Objectives, descriptions, and start/end dates.



20) Project Overview

Implementation Status: Completed

Story Points: 5

Tasks:

- Remove project objectives from the dashboard for students.
- Display project objectives on the Project Hub for students.
- Display student scores on the Project Hub for students.

## 21) Bug Fixes from Past Sprint

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Fix bugs of past sprint
- Test for deployed functionality of the last sprint
- Cucumber and Rspec testing to cover new fixes

## 22) Admin Tasks Management

**Implementation Status:** Completed

**Story Points:** 5

**Tasks:**

- Create admin side CRUD for task management for every student.
- Add the ability to tag tasks to milestones.
- Add functionality to view the task progress of each student.
- Write RSpec and cucumber tests for the tab.
- Deploy the application to Heroku.

The screenshot displays the 'Project Gamma' admin interface. The top navigation bar includes the 'Teamup NEO' logo, a user profile for 'Rahaan Gandhi', and links to 'Manage Projects', 'Manage Students Score', 'Events', and 'Calendar'. A sidebar on the left lists navigation options: 'Dashboard', 'Milestone Management', 'Task Management' (highlighted), 'Resource Management', and 'Team Management'. The main content area, titled 'Manage Tasks', is divided into three columns for students: Dhruva, Rahaan, and Akanksha. Each column features an 'Add Task' button and a 'Task Completion' progress bar. Below these, tasks are listed in a grid, each with a title, a status badge (e.g., 'Not started'), a folder icon, and a due date. For example, Dhruva has four 'Gamma Milestone 1 Task' entries (IDs 2, 5, 5, 2) and one 'Gamma Milestone 3 Task' entry, all marked 'Not started'. Rahaan has four 'Gamma Milestone 2 Task' entries (IDs 3, 3, 4, 5) and one 'Gamma Milestone 1 Task' entry (ID 5), with the last one marked 'Completed'. Akanksha has four 'Gamma Milestone 2 Task' entries (IDs 1, 4, 1, 1) and one 'Gamma Milestone 1 Task' entry (ID 1), all marked 'Not started'.

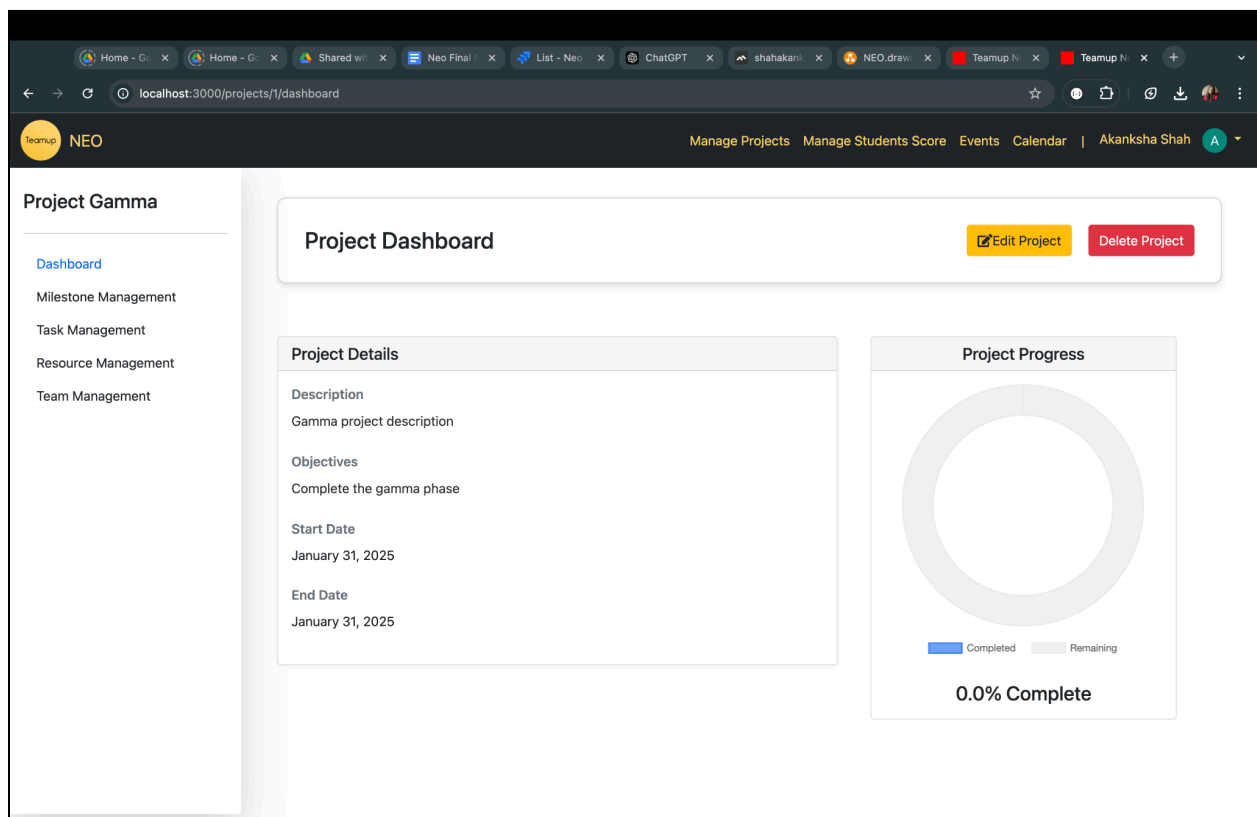
## 23) Project-Specific Dashboard Implementation for Admin

Implementation Status: Completed

Story Points: 5

### Tasks:

- Display description and objectives
- Add functionality to update project details
- Add Project Progress chart ( progress depending on the milestones that are complete)
- Research on Calendar Integration





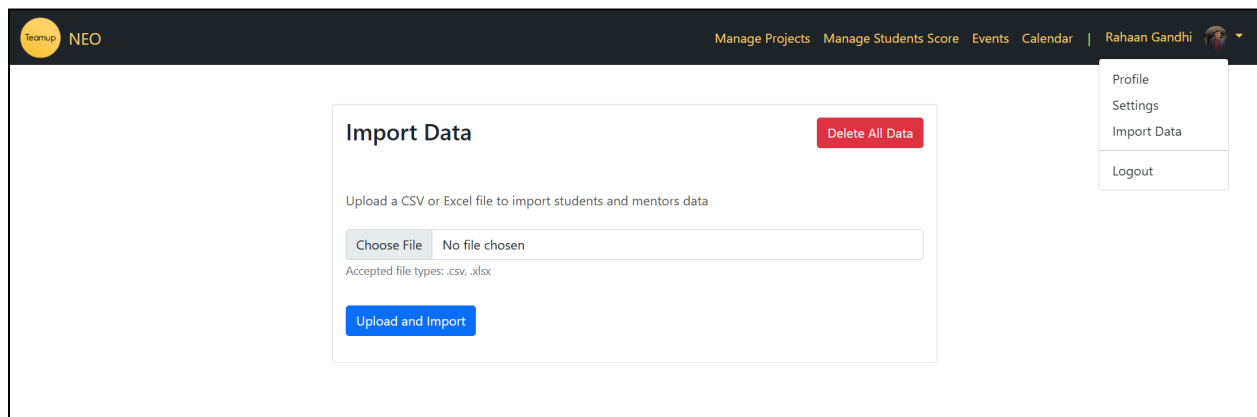
## Sprint 4 Stories

### 24) Import Data for Admins

**Implementation Status:** Completed

**Story Points:** 5

- Create a view that allows admins to register the mentors and students to the application by uploading the Excel.
- Add an option for admins to delete the students and mentors.
- Achieve  $\geq 90\%$  coverage on RSpec and Cucumber.
- Bug fixes from Sprint 3.
- Deploy the application to Heroku.



### 25) Event Broadcast Bar

**Implementation Status:** Completed

**Story Points:** 5

- An "Events" option in the admin navbar leads to a page for creating, updating, deleting, or setting an event as "Show."
- A banner on the student dashboard displays the details of the active event, keeping students informed.
- The admin interface is user-friendly, and event visibility updates automatically on the student dashboard when changed by the admin.

Teamup

NEO

Manage Projects

Manage Students Score

Events

Calendar

|

Rahaan Gandhi

Event Management

Events

Complete Project Selection

EditDelete

All students must select their project by the deadline

☐ Broadcast Event

Guest Lecture on Agile Development

EditDelete

Please join guest lecture on Wednesday at 5:30pm

☒ Broadcast Event

Create a New Event

Title

Description

Create Event

## Broadcasted event on the student dashboard

Teamup

NEO

Project Hub

Calendar

|

Akanksha Shah

Project Gamma

Project Hub

Project Milestones

Project Timeline

Task Management

Team Members

You are logged in.

Guest Lecture on Agile Development

Please join guest lecture on Wednesday at 5:30pm

Date: 20 Nov 2024

Project Gamma

Description

Gamma project description

Objective

Complete the gamma phase

Start Date

15 Nov 2024

End Date

31 Jan 2025

Know Your Team

Certificate Completion Score

Score : 80.0/100

80.0

Resources

Project Plan

November 20, 2024

Neo Plan.pdf

Preview

Download

## 26) Gantt Chart fixes and Project table fixes

**Implementation Status:** Completed

**Story Points:** 3

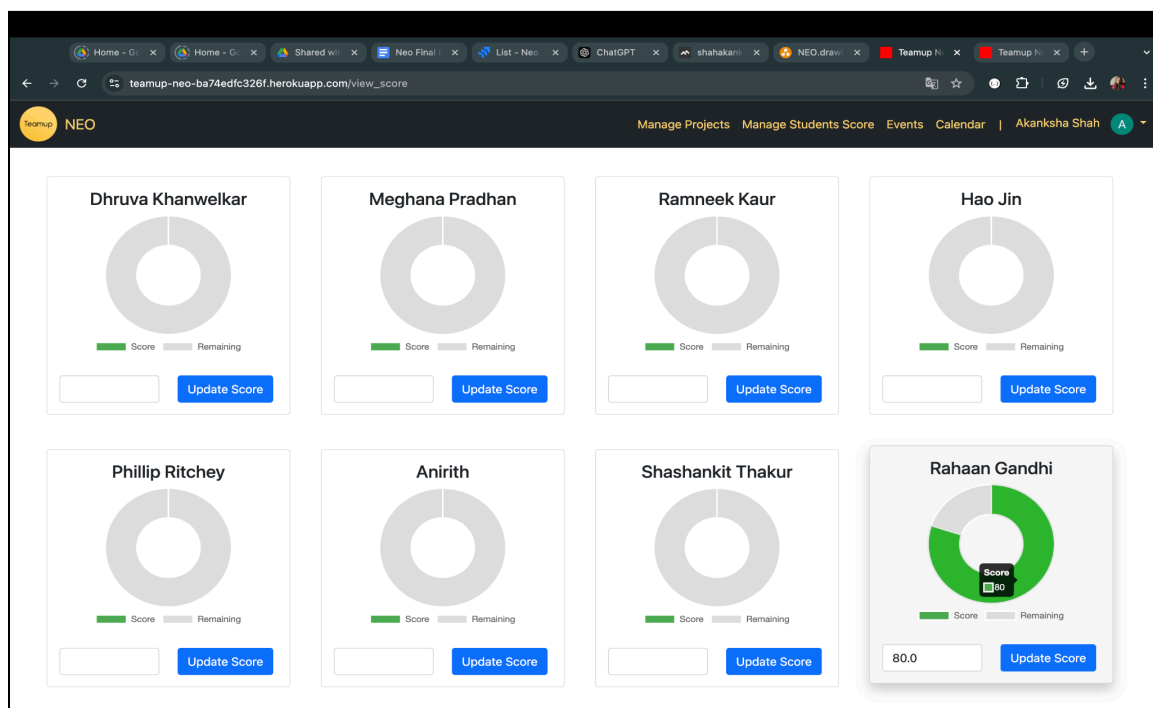
- Gantt Chart UI fixes:
  - Make the Gantt chart scrollable vertically and horizontally
  - Give each Milestone a different color in the chart
  - Enhance date information on the X-axis
  - Remove objectives from Timeline Table
- Remove Project status from UI
- Add the start and end dates to the project details on the admin and edit pages
- Ensure the project title is unique and not duplicate

## 27) Admin Score Management

**Implementation Status:** Completed

**Story Points:** 3

- Implement a pie chart to display student scores for admin
- Improve score edit page UI
- Integrate dashboard view to project hub



## 28) Bug Fixes from Past Sprint

**Implementation Status:** Completed

**Story Points:** 3

- Improve Resource Page UI
- Fix bugs from the last sprint:
  - Milestone start date and end date within the project start date and end date
  - Deleting milestones should cascade the delete to the tasks, along with student task assignments and the timeline table
  - Milestone status cannot be in progress before the start date
  - The milestone title must be unique

## 29) Admin Tasks Management

**Implementation Status:** Completed

**Story Points:** 3

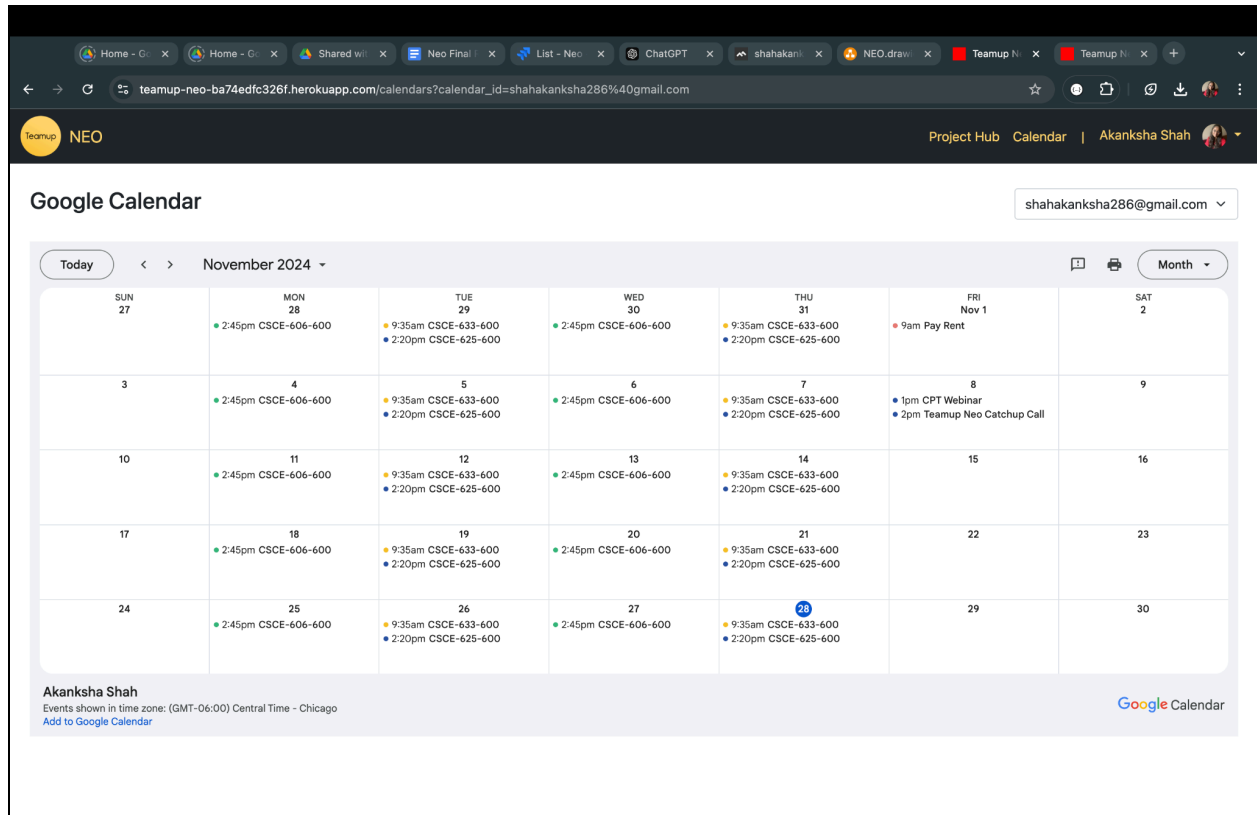
- Task Management UI Enhancement on Admin view
- Statistics for task completion per student
- Bug Fixes:
  - Start date and End date for tasks must be within Milestones start and end dates
  - Task Titles must be unique

## 30) Google Calendar Integration and Bug Fixes

**Implementation Status:** Completed

**Story Points:** 5

- Google Calendar Integration
- Bug Resolution
  - Deleting projects should cascade the delete to every table.
  - If a student is removed from the team, the tasks should be unassigned and not started - should delete the tasks as well from the database.



## Roles

Roles were changed every Sprint so everyone in the team had an opportunity to act as a Scrum Master and/or Product Owner.

### 1) Sprint I

Product Owner: Rahaan Gandhi

Scrum Master: Meghna Pradhan

### 2) Sprint II

Product Owner: Dhruva Khanwelkar

Scrum Master: Ramneek Kaur

### 3) Sprint III

Product Owner: Akanksha Shah

Scrum Master: Hao Jin

#### 4) Sprint IV

Product Owner: Rahaan Gandhi

Scrum Master: Yiyang Yan

### Summary and Achievements of Each Sprint

#### 1) Sprint I

**Total number of points completed = 21**

**Goal** - This sprint focused on implementing authentication and authorization using Google OAuth, as well as establishing the foundational structure for the dashboard and database modeling.

##### **Achievements**

- Designed low-fi UI mockups for all views.
- Developed secure login and logout functionality with Google OAuth.
- Set up basic navigation (navbar) and routing between views.
- Designed and modeled the database schema with object associations.
- Achieved test coverage of ~94%.
- Successfully deployed the application to Heroku.
- Discussed and documented the team's working agreement.

#### 2) Sprint II

**Total number of points completed = 35**

**Goal** - The sprint aimed at developing core features for both admin and student dashboards, with a focus on project management and resource handling.

##### **Achievements**

- Implemented resource management with AWS S3 for document uploads.
- Developed a Project Management Hub for mentors to edit teams and add resources.

- Created the Project Team Information page for students to view team member and mentor information.
- Implemented dashboard pages for both user and student views.
- Designed and implemented the database schema in a relational database.
- Achieved RSpec test coverage of ~94% and Cucumber test coverage of ~90.45%.
- Successfully updated the database on Heroku.

### 3) Sprint III

**Total number of points completed = 35**

**Goal** - This sprint focused on enhancing the dashboards with project, milestone, and task management features, including Gantt chart integration and resource access for students.

#### **Achievements**

- Enhanced the UI with a sidebar for both student and admin dashboards.
- Integrated features like milestone and task management, and certificate completion tracking for the admin dashboard.
- Displayed project and milestone progress in donut charts for admins.
- Developed student dashboard features like milestone lists, task management, and a Gantt chart view for project timelines.
- Added a resource listing view for students to preview or download project materials uploaded by admins.
- Achieved RSpec test coverage of 93.23% and Cucumber test coverage of 95.75%.
- Successfully updated the database on Heroku.
- Discussed and revised the team's working agreement.

#### 4) Sprint IV

**Total number of points completed = 27**

Goal - This sprint focused on calendar integration, broadcasting events, implementing data import for admins, and addressing bugs and UI improvements.

##### **Achievements**

- Integrated Google Calendar and Events Broadcast Banner.
- Implemented the import feature for admins to upload student data.
- Achieved RSpec test coverage of 97.05% and Cucumber test coverage of 96.81%.
- Fixed multiple bugs across the repository to improve app stability.
- Resolved issues with calendar deployment in production.
- Enhanced UI for task management and score management.
- Successfully deployed the application to Heroku.

#### Story Points Across Sprints

Name	Total points solved by the individual	Total points solved by the entire team
Rahaan Gandhi	18	118
Meghna Pradhan	19	
Akanksha Shah	20	
Dhruva Khanwelkar	16	
Hao Jin	14	
Ramneek Kaur	17	
Yiyang Yan	14	



## **Customer Meetings**

**Date: Wed Oct 9, 2024**

**Time: 9:30am - 10:30am**

**Place: Zoom**

### **Summary of Discussion**

- Demonstration of Current Project Progress and deployed Heroku app.
- Detailed review of Milestone 1 goals, current status.
- Discussion of the data model and the relationship between models.
- Discussion on the scope of Milestone 2 and expected deadlines.
- Recommendations for increasing development speed and expanding functionality coverage in the second sprint.
- Feedback on communication frequency with the client.
- Spoke on deliverables and how they expected more UI-related changes and deliverables than what was worked on.
- We explained more of our work was focused on the backend, which is crucial to the functionality of the frontend.

**Date: Wed Oct 16, 2024**

**Time: 9:30am - 10:30am**

**Place: Zoom**

### **Summary of Discussion**

- Demonstration of Current Project Progress.
- Discussion on the User Interface for the Project Management Hub.
- Discussion on Data Models and Their Relationships.
- Discussion on utilizing appropriate blob storage.

### **Feedback**

- Blob Storage like AWS S3 should be used to store Project Files

**Date: Wed Oct 23, 2024**

**Time: 9:30am - 10:30am**

**Place: Zoom**

### **Summary of Discussion**

- Demonstration of Current Project Progress
- Alternative options of storage for resources
- Use of Graph Packages for future data analysis
- Discussion of Student Scores and emphasis on displaying progress charts and graphs
- Discussion on Google Calendar integration

### **Feedback**

- Focus on key features of the application, including milestone tracking, task management, student score monitoring, and project progress visualizations.

**Date: Wed Nov 6th, 2024**

**Time: 9:30am - 10:30am**

**Place: Zoom**

### **Summary of Discussion**

- Presented the project's current progress, including a live demonstration of the deployed Heroku application.
- Reviewed the final sprint wrap-up and discussed usability expectations for the completed product.

### **Feedback**

- Ensure the hosted application is fully functional and user-friendly by the end of the sprint.
- The client communicated appreciation for the progress made in the last sprint.
- Implement the "Event Broadcast" feature, Google Calendar integration, and support for user detail uploads.

**Date: Wed Nov 13th, 2024**

**Time: 9:30am - 10:30am**

**Place: Zoom**

### **Summary of Discussion**

- Presented the project's current progress, including a live demonstration of the deployed Heroku application.
- Reviewed the final sprint wrap-up and discussed usability expectations for the completed product.

### **Feedback**

- Ensure the hosted application is fully functional and user-friendly by the end of the sprint.
- The client communicated appreciation for the progress made in the last sprint.
- Suggested improvements to events broadcast feature and calendar integration.

**Date: Wed Nov 20th, 2024**

**Time: 9:30am - 10:30am**

**Place: Zoom**

### **Summary of Discussion**

- Presented the project's final version, including a live demonstration of the deployed Heroku application.
- Discussed Documentation and handover procedures.

### **Feedback**

- Ensure the hosted application is fully functional and fix any necessary bugs before handover
- The client communicated appreciation for the team's hard work and the application
- Add a feature to allow the admin to set the maximum score possible for all students

## **BDD&TDD Process**

### **BDD Process (Behavior-Driven Development)**

#### **1. Collaborative Requirement Definition**

- Begin by collaborating with stakeholders (e.g., product owners, developers, and QA) to define the expected behavior of the application.
- Use plain language to write feature files in Gherkin syntax, focusing on "Given-When-Then" scenarios.

#### **2. Writing Scenarios (Cucumber)**

- Write feature files that describe how the system should behave from the user's perspective.
- Scenarios should be clear, and specific, and cover edge cases where applicable.

#### **3. Automating Scenarios**

- Implement step definitions in Ruby to automate the execution of the scenarios.
- Ensure each step directly maps to the intended user interaction or system behavior.

#### **4. Iterative Testing**

- Run Cucumber tests to ensure the application behaves as described.
- Collaborate to refine feature files or step definitions if tests fail or requirements evolve.

### **TDD Process (Test-Driven Development)**

#### **1. Write a Test First (RSpec)**

- Write a failing unit test in RSpec for a small piece of functionality.
- Tests should be focused on individual methods or classes, ensuring precise and modular design.

#### **2. Implement the Code**

- Write the minimal amount of code required to make the test pass.

#### **3. Refactor**

- Clean up both the code and the test to ensure maintainability without changing behavior.
- Repeat the process for subsequent features or bug fixes.

#### 4. **Continuous Integration**

- Combine unit tests with integration tests to ensure changes in one area do not break functionality elsewhere.

## **Configuration Management Approach**

Our approach to configuration management is structured and secure, ensuring seamless collaboration and reliable deployments.

### **Configuration Handling**

- We use **Rails' credentials.yml with a master key** for managing sensitive information such as credentials.
- This setup ensures that secrets are encrypted and version-controlled safely, with only authorized access to the master key.

### **Branching Strategy**

We follow a **Gitflow-inspired workflow** to maintain code quality and streamline development:

#### 1. **Main Branch (main):**

- Represents the stable version of the application.
- Only updated after successful releases.
- One main branch.

#### 2. **Development Branch (develop):**

- Serves as the integration branch for new features and bug fixes.
- Code from individual developer branches is merged here after peer reviews and successful testing.
- One develop branch.

#### 3. **Release Branches:**

- Created at the start of each sprint for preparing production-ready code.

- Used for final testing, bug fixes, and polishing features before merging into the main.
- There were a total of 4 release branches, each corresponding to the total of 4 Sprint.

#### 4. **Developer Branches (dev-username):**

- Each team member has their branch for working on assigned tasks.
- These branches are frequently rebased to develop to avoid conflicts and ensure up-to-date work.
- There were a total of 7 dev branches, one corresponding to each team member.

### Releases

- We aimed for **one release per sprint**, ensuring that deliverables were well-tested and aligned with sprint goals.
- Hotfixes, if needed, are branched from the main and quickly merged back after resolving critical issues.

### Spikes

We performed **spikes** (short exploratory tasks) when introducing new tools or features:

- **Using credentials.yml:** Ensured compatibility with our CI/CD pipeline and cloud hosting provider before implementation.
- **Branching Strategy:** Conducted an initial spike to decide the best workflow that balances team collaboration and deployment speed.

### Heroku Production Issues

During the production release process on Heroku, we faced two significant issues that required targeted solutions. The first challenge arose during database management. After resetting the database on Heroku, running `db:migrate` failed because the reset process disrupted the database state. To resolve this, we utilized `db:schema:load`, which directly loaded the pushed schema into the database. This approach bypassed the need to apply migrations incrementally, enabling us to seed the database and

ensure that it was fully synced with the application code. This workaround ensured a consistent database state but highlighted the importance of careful database resets and proper schema versioning.

The second issue involved Google OAuth and its integration with the application. On the deployed version, the OAuth consent screen did not appear during user logins, as the system retained previous consent information in cookies and skipped the consent step. This behavior caused the Google Calendar integration to fail, as explicit user consent is required to access calendar data. To fix this, we modified the OmniAuth configuration to include the `prompt: 'consent'` parameter, ensuring the consent screen was displayed during every login attempt. This adjustment resolved the issue and restored calendar functionality. Together, these challenges emphasized the need for meticulous configuration and alignment between database management and third-party integrations to ensure a smooth production deployment.

## Tools/Gems Used

### 1) GitHub

**Benefits** - GitHub served as the central version control system, facilitating collaboration, code reviews, and issue tracking. Its pull request workflow allowed team members to review and discuss code changes before merging, ensuring high-quality code integration. Additionally, GitHub Actions automated tasks such as running tests and deploying the application, streamlining the development process.

**Problems** - Merge conflicts occasionally arose when multiple contributors worked on overlapping features, requiring careful resolution to avoid breaking the codebase. Moreover, effective use of GitHub depended heavily on disciplined commit practices, which required consistent adherence by all team members.

## 2) CodeClimate

**Benefits** - CodeClimate was used to analyze code quality and maintainability. It provided metrics on code complexity, duplication, and potential vulnerabilities, allowing the team to identify and address issues proactively. The tool's integration with GitHub ensured that feedback was accessible during code reviews, fostering a culture of clean, maintainable code.

**Problems** - While CodeClimate provides metrics and highlights areas for improvement, it does not always explain the reasoning behind the issues it identified, leaving us to interpret the results without sufficient guidance. Furthermore, CodeClimate lacked several key features that would have improved the feedback process. It did not offer the ability to categorize issues based on their severity, making it difficult to prioritize them according to their impact on the codebase. It also did not provide deeper analysis, such as identifying code smells, security vulnerabilities, or coverage gaps. More importantly, the tool lacked actionable steps or specific recommendations for resolving these issues.

## 3) SimpleCov

**Benefits** - SimpleCov was used to measure test coverage, ensuring that the application was adequately tested. It Generated detailed reports, which allowed the team to identify untested parts of the codebase, encouraging comprehensive test writing. This increased confidence in the stability of the application, especially during deployment.

**Problems** - Achieving high test coverage often revealed gaps in the test suite that required additional effort to address. However, SimpleCov only measured coverage and did not assess the quality of the tests, so the team needed to remain vigilant about writing meaningful, effective tests.

## 4) OmniAuth Gem

**Benefits** - OmniAuth simplified the integration of third-party authentication providers like Google OAuth into the application. It provided a streamlined way to handle the



complex OAuth workflow, including generating authorization URLs, exchanging tokens, and managing user authentication. This allowed us to enable secure login functionality and integrate features such as Google Calendar without having to build the authentication mechanism from scratch.

**Problems** - Configuring OmniAuth required careful attention to the provider-specific settings, such as API keys and scopes, to ensure functionality. A specific issue we encountered was with cookies retaining prior consent, which caused problems with Google Calendar integration. We addressed this by adding the `prompt:'consent'` in the configuration to enforce explicit user re-consent during each login.

## 5) Faker Gem

**Benefits** - The Faker gem was invaluable during development and testing, providing realistic yet fake data to populate the database. It allowed us to generate random names, emails, and other data fields, which helped create test scenarios that mimicked real-world usage. This was particularly helpful for testing features like team management and resource sharing without relying on actual user data.

**Problems** - While Faker made testing easier, over-reliance on randomly generated data occasionally led to inconsistencies in test cases, requiring the use of fixed seed values to ensure reproducibility. Additionally, some generated data could inadvertently conflict with validation rules, requiring careful configuration during test setup.

## 6) aws-sdk-s3 Gem

**Benefits** - This gem was used to integrate Amazon S3 for file storage and management within the application. By leveraging S3, we were able to efficiently upload, store, and retrieve files such as documents and resources that were needed by students and mentors. S3 provided a scalable, reliable, and secure solution for managing assets, ensuring that files were safely stored and easily accessible without burdening the application server. The gem's straightforward API allowed us to interact with S3 buckets directly from the application, simplifying the file-handling process.

**Problems** - One potential challenge was configuring the proper AWS credentials for secure file access, which had to be managed carefully to avoid any exposure to sensitive data. Misconfigurations or lack of proper permissions could lead to errors, such as access-denied issues when trying to upload or retrieve files. Furthermore, managing file storage costs on S3 required monitoring, especially if there was a large volume of data being uploaded, to avoid unanticipated charges.

## **Limitations**

### **1) Google Calendar API**

The application utilizes the Google Calendar API, which involves sensitive scopes and requires verification by Google to allow unrestricted access in a production environment. Currently, the application is operating in testing mode, which limits access to only those users explicitly added to the testing list. This restriction ensures controlled usage during the development and testing phases. However, accounts under the TAMU organization have been configured to bypass these limitations, enabling stakeholders and team members with TAMU accounts to access the calendar in the production environment. New users with personal Gmail accounts will not be able to access the calendar in production until the application is verified and transitioned to production mode.

### **2) AWS S3 Bucket**

The Resources tab currently utilizes AWS S3 to store files uploaded by admins through the portal. While the S3 bucket allows unlimited uploads via the application, the free tier imposes a limit of 2,000 files. To address this constraint, alternative solutions must be explored to enable unlimited file storage. Additionally, security measures need to be enhanced to ensure that only registered admins can access the bucket. It is also essential to implement segregated buckets for different environments, as the current setup has all three environments pointing to a single bucket. This arrangement risks quickly exceeding the file limit and can lead to management challenges.

## Repository Contents

The GitHub repository contains the following files

- 1) Application/Source code under app folder.
- 2) Dependencies under Gemfile.
- 3) Rspecs under spec folder.
- 4) Cucumber features under the features folder.
- 5) Google OAuth credentials in the config/credentials.yml.enc file.
- 6) README.md file explaining the project setup and deployment steps.
- 7) Sprint Documentation and Final Report under documentation/Fall2024 folder.

Project Deployment steps:

Login to Heroku through CLI If you have not already done so, install the Heroku CLI.

*heroku login*

### **Create a Procfile**

*echo "web: bundle exec rails server -p \$PORT" > Procfile*

This file tells Heroku what to do when the application is deployed. In this case: spin up a web process to host the application.

### **Create a Heroku App**

*heroku create <app\_name>*

### **Verify that the git remote is set:**

*git config --list --local | grep heroku*

### **Make Master Key Available on Heroku**

*heroku config:set RAILS\_MASTER\_KEY=`cat config/master.key`*

This enables Rails on Heroku to decrypt the credentials.yml.enc file, which contains the OAuth credentials (and any other secrets your app needs).

### **Provision a Database**

*heroku addons:create heroku-postgresql:essential-0*

An essential-0 PostgreSQL database costs \$5 a month, prorated to the minute.

### **Add PostgreSQL to Gemfile**

```
group :production do
  gem 'pg'
End
```

Run this so that your app won't try to install production gems (like postgres) locally:

*bundle config set --local without 'production' && bundle install*

### **Add the Heroku App to Authorized redirect URIs on Google Cloud Console**

- Go to the Google Developer Console.
- Click on Credentials in the left navigation panel
- Click on your OAuth 2.0 Client ID for this project
- Add your Heroku app's callback address as an authorized redirect URI e.g.  
[https://your-apps-name.herokuapp.com/auth/google\\_oauth2/callback](https://your-apps-name.herokuapp.com/auth/google_oauth2/callback)
- Click "SAVE"

### **Deploy the App to Heroku**

*git push heroku main*

### **Migrate the Database**

*heroku run rails db:create*

*heroku run rails db:migrate*

### **Seed the Database**


*heroku run rails db:seed*

### **Launch the application:**

*heroku open*

Your app is now successfully deployed to the production environment.

## Links

<b>GitHub Repository</b>	<a href="https://github.com/shahakanksha-tamu/teamup-neo-backend">https://github.com/shahakanksha-tamu/teamup-neo-backend</a>
<b>Slack</b>	<a href="https://app.slack.com/client/T07PC1QMZRN/C07NP8T7UE6">https://app.slack.com/client/T07PC1QMZRN/C07NP8T7UE6</a>
<b>JIRA Stories</b>	<a href="https://tamu-teamneo-606.atlassian.net/jira/software/projects/JIRA/list">https://tamu-teamneo-606.atlassian.net/jira/software/projects/JIRA/list</a>
<b>Heroku Deployment</b>	<a href="https://teamup-neo-ba74edfc326f.herokuapp.com/">https://teamup-neo-ba74edfc326f.herokuapp.com/</a>
<b>Demo</b>	<a href="https://youtu.be/Bhu5eubuGm0">https://youtu.be/Bhu5eubuGm0</a>
<b>Presentation</b>	 Neo Presentation.pptx