MySTEMGrowth Survey Tool

Design Document

Sdmay 26-27

Dr.Rover

Dr. Rover, Yiqi Liang

Ethan Buenting: Backend/Cloud

Ethan Van Caster: Developer

Nina Gadelha: Cybersecurity Developer/Testing

Ryan Mamrot: Developer

Sam Craft: Developer/Testing

Caleb Hemmestad: Backend/Cloud

https://sdmay26-27.sd.ece.iastate.edu/

Revised: 09/30/2025

Executive Summary

The MySTEMGrowth Survey Tool is a web-based application designed to assess and visualize undergraduate students' growth during participation in STEM research programs. Many research programs rely on manual surveys or fragmented tools that make it difficult for students to meaningfully reflect on their development and for administrators and program coordinators to efficiently analyze outcomes. This project addresses that gap by providing a centralized, role-based platform that securely collects survey data, compares pre- and post-research results, and presents actionable insights for continuous program improvement.

The primary users of the system are students, program coordinators, and administrators. Students complete surveys at the beginning and end of their research experience and can view visualized results highlighting growth across key areas. Program coordinators manage student groups, distribute surveys, and review aggregated results to evaluate program effectiveness. Administrators maintain system-level oversight, managing user access, survey content, and data exports. The system is designed to support these varying needs through role-based access control and intuitive, purpose-driven interfaces.

Key design requirements for the MySTEMGrowth tool include secure authentication, protection of sensitive student data, scalability for multiple programs, ease of use across devices, and efficient data storage and retrieval. To meet these requirements, the team adopted a modern web architecture using a React/Next.js frontend and a Node.js/Express backend, supported by a MySQL database. Authentication is being strengthened through password hashing and token-based sessions, while role-based authorization ensures that users can only access appropriate data. The system is hosted on DigitalOcean using managed services to reduce cost and complexity while maintaining reliability and scalability.

Significant progress has been made in understanding the inherited codebase, planning architectural improvements, and implementing foundational security and testing enhancements. The team has designed updated UI layouts for mobile and coordinator views, developed architectural and data-model diagrams, and established testing strategies across unit, integration, and system levels. Initial backend and frontend test suites validate authentication flows, role enforcement, and core service behavior.

Overall, the proposed design effectively meets user needs by prioritizing security, clarity, and usability while remaining maintainable for future development teams. Next steps include completing the cloud migration from AWS to DigitalOcean with automated deployment, finalizing authentication and authorization improvements, enhancing mobile responsiveness, and incrementally deploying planned feature improvements. Through continued collaboration with stakeholders and iterative development, the MySTEMGrowth Survey Tool aims to deliver a robust, secure, and impactful platform for evaluating undergraduate research experiences.

Learning Summary

Development Standards & Practices Used

Throughout the MySTEMGrowth project, our team applied software engineering practices to ensure reliability, security, and maintainability. We followed a structured software life-cycle model aligning with IEEE 1448a-1996, working though through analysis, design, implementation, testing, and deployment phases. Configuration management practices based on IEEE 828-2012 helped guide our use of Git for version control, feature branching, issue tracking, and controlled merges. Verification and validation principles from IEEE 1012-2016 helped navigate our testing strategy, including unit, integration, system, and acceptance testing.

Summary of Requirements

- A web-based tool that client users can access
- Create charts based on survey data
- Role-based access for administrators, program coordinators, and students
- Secure account creation, login, logout, and account management
- Ability for students to take surveys and view personal results
- Ability for coordinators to view aggregated and student results within assigned groups
- Administrative control over surveys, programs, users, and exports
- Secure storage of user data and survey results
- Responsive UI supporting desktop and mobile devices
- Cloud-hosted deployment with scalable backend and managed database
- Automated build and deployment pipeline
- Data export and reporting functionality

Applicable Courses from Iowa State University Curriculum

- COM S 2270
- COM S 2280
- SE 3090
- SE 3190
- COM S 3630
- SE 3170
- SE 3390
- CPR E 2300
- CPR E 2310
- CPR E 3310
- ENGL 3140

New Skills/Knowledge acquired that was not taught in courses

- Configuring and migrating cloud

- Cookies and token-based authorization
- RBAC across frontend and backend

Table of Contents

List of figures/tables/symbols/definitions

List of Figures:

List of Tables:

List of Symbols and Definitions:

API - Application Programming Interface

CI/CD - Continuous Integration / Continuous Deployment

DB - Database

JWT - JSON Web Token

RBAC - Role-Based Access Control

UI - User Interface

UX - User Experience

1. Introduction

1.1. Problem Statement

The MySTEMGrowth Survey aims to provide students feedback based on their time they spent in undergraduate research. The tool accomplishes this by having students take the survey at both the beginning and end of their undergraduate research term, and share their results and improvements in 6 key areas. Specifically, the target group for this tool is undergraduate STEM students involved in research.

Students answer a series of questions before and after participating in research programs. The first set of results gives students a baseline, which they can use to compare their results with after completing the programs. Survey educators/administrators will use the students' results to make improvements to the research programs. The MySTEMGrowth Survey aims to make gradual improvements/innovations to undergraduate research programs, to benefit the learning experience of future students.

This survey was inherited from a previous design team. Although the basic structure is present and functional, it lacks security, modularity, depth, and personality. The problem will be solved by improving the currently lacking aspects in the current state of the survey tool. Implemented solutions will not only benefit the students taking the survey, but also survey administrators. Students' information will be more secure, and their overall experience taking the survey will be more efficient and straightforward. Administrators will be able to view and act upon results more efficiently, as the survey will provide information in a more user-friendly manner.

In addition, the current state of the tool is supported by web browsers on computers/laptops. However, current design/layout does not suit smaller sized screens (such as mobile devices). Improving the modularity of the tool is one of the main components we will look to improve. Another current undesirable component is the cloud situation, currently being held on AWS. After searching for alternatives due to the complexity and pricing of AWS, we are looking to switch to a new cloud service, Digital Ocean. We believe that Digital Ocean will provide all the security, storage and capabilities that the MySTEMGrowth Survey tool will require, and are in the early stages of migrating the database from AWS.

1.2. Intended Users:

We intend to support 3 different user types for the MySTEMSurvey Growth tool: Administrators, Program Coordinators, and Students. Each group will have certain roles, permissions, and level of access to information.

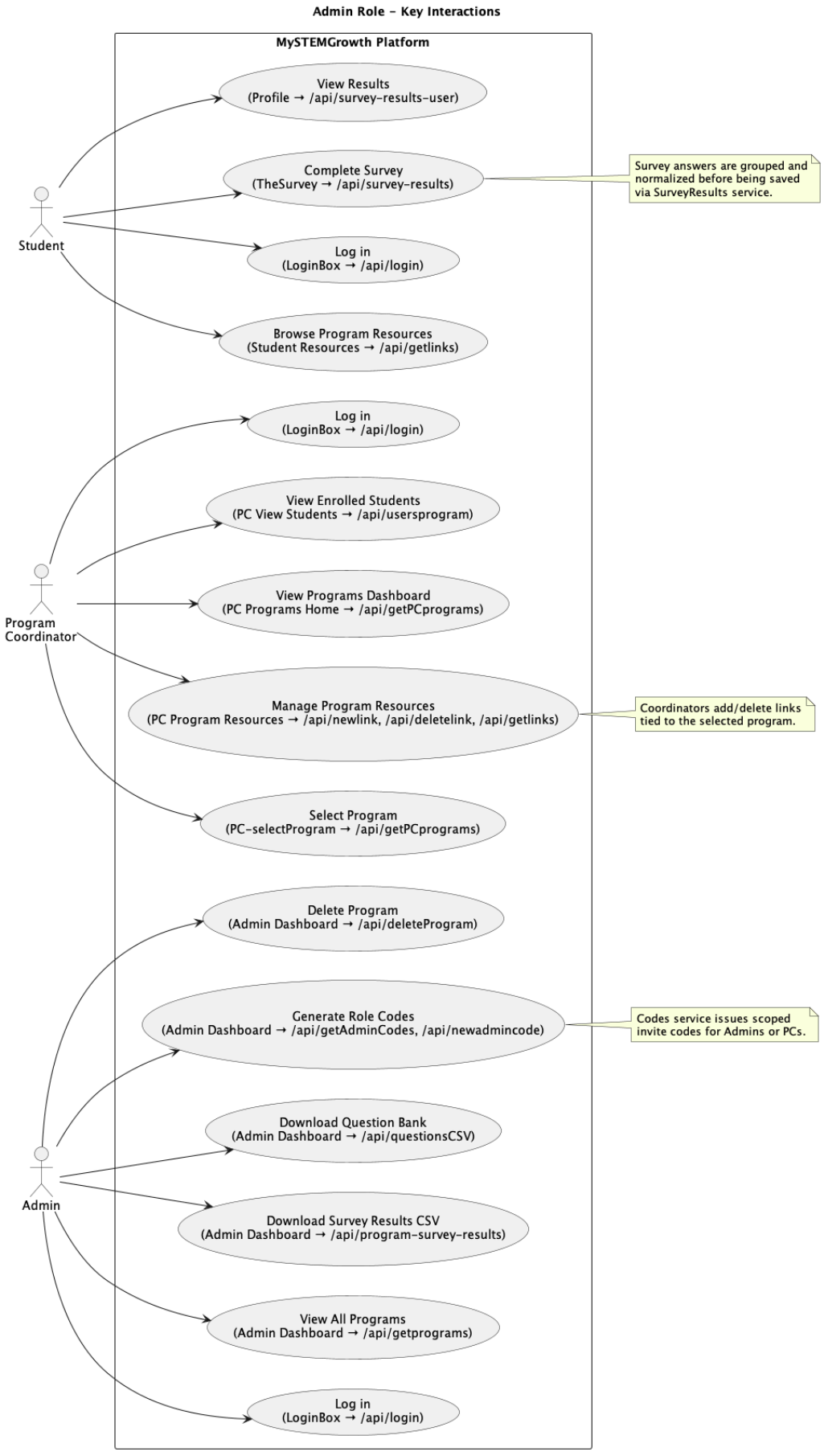
Admin Role – Key Interactions
MySTEMGrowth Platform
Student
View Results
(Profile → /api/survey-results-user)
Complete Survey
(TheSurvey → /api/survey-results)
Survey answers are grouped and normalized before being saved via SurveyResults service.
Log in
(LoginBox → /api/login)
Browse Program Resources
(Student Resources → /api/getlinks)
Program Coordinator
Log in
(LoginBox → /api/login)
View Enrolled Students
(PC View Students → /api/usersprogram)
View Programs Dashboard
(PC Programs Home → /api/getPCprograms)
Manage Program Resources
(PC Program Resources → /api/newlink, /api/deletelink, /api/getlinks)
Coordinators add/delete links tied to the selected program.
Select Program
(PC-selectProgram → /api/getPCprograms)
Admin
Delete Program
(Admin Dashboard → /api/deleteProgram)
Generate Role Codes
(Admin Dashboard → /api/getAdminCodes, /api/newadmincode)
Codes service issues scoped invite codes for Admins or PCs.
Download Question Bank
(Admin Dashboard → /api/questionsCSV)
Download Survey Results CSV
(Admin Dashboard → /api/program-survey-results)
View All Programs
(Admin Dashboard → /api/getprograms)
Log in
(LoginBox → /api/login)

Figure 1: Key Interactions

MySTEMGrowth Administrators (High access level):

Administrators will have the responsibility of setting up/distributing surveys to students, which requires them to have full access to the MySTEMGrowth Survey tool. Their roles include, but are not limited to:

- Creating/Modifying/Deleting survey groups
- Inviting/Removing participants from created groups
- Viewing results for all participants of a group
- Exporting/Sending result data
- Saving/Storing result data
- Giving access to others (certain staff) to view the created group's results

Administrators will play the 'coordinator' role. They are responsible for survey distribution, result management, and access delegation. The administrator interface will be organized, concise, and easy to navigate, as they will have a large amount of information to sift through.

MySTEMGrowth Program Coordinator (Medium access level):

Staff's responsibility is to create groups of students within their class and distribute surveys accordingly. They will be able to:

- Creating an account
- Gaining access to a group's data (created/given by administrators)
- Submit requests for survey modifications (adding/removing/editing current questions)
- View overall result data from multiple (owned) groups

Although their access will be more limited, their interface (like administrators) should be organized and easy to navigate through large amounts of data. They should be able to view results using different methods (text, graphs, charts), in order to better view a large volume of results. They should have access to view one specific student's results or an overall collection of their group's response data.

MySTEMGrowth Participants (Low access level):

Participants (mainly anticipated to be students) provide the data that will be collected and used to help improve undergraduate research programs. Their permissions include:

- Creating an account
- Take assigned surveys
- View personal survey results

Participants will have little access, as their main role is to provide the data. Their interface should be straightforward, easy to follow, and chronological. They should have access to view all of

their own personal data (account information, previous survey results, enrolled groups), but no one else's.

2. Requirements, Constraints, And Standards

2.1. Requirements & Constraints

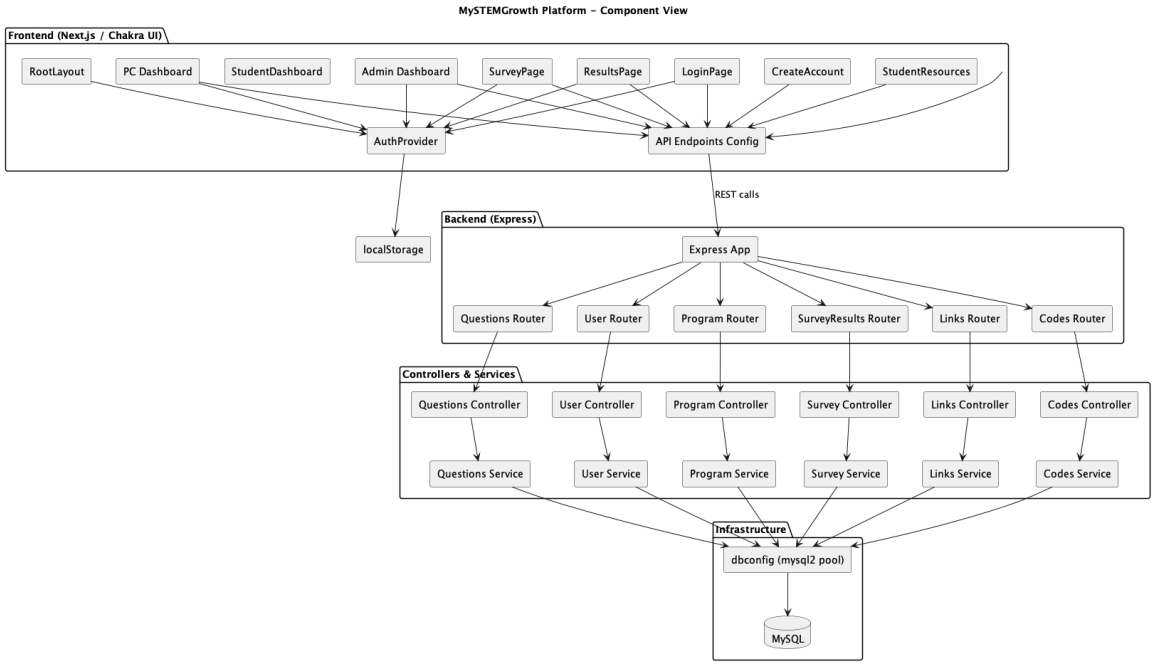

Figure 2: Component View

2.1.1. Functional Requirements

- Students should be able to take/view results from surveys (appointed to them by administrators)
- All users (administrators, program coordinators and students) will be able to create an account, login to their account, delete their account, change their passwords and view survey results
    - Students should only be able to view their own survey results (from any time period)
    - Program Coordinator should have access to view any/all results from students within groups they have assigned
    - Administrators should be able to view all survey results (any student from any program)
- Administrators will be able to modify the survey (add, edit or delete questions), have access to all results and enable program coordinators to create and distribute surveys to approved groups

2.1.2. Resource requirements

- Frontend
    - React.js
    - Next.js
    - Chakra UI
- Backend
    - Node.js
    - Express Framework
    - Next Auth
    - MySQL
    - PGAdmin
    - Jest
- Cloud
    - Digital Ocean
        - Managed Database
        - App Platform encrypted env vars
            - Use Web Services to function as components
        - DNS

2.1.3. UI requirements

- Each user type page should have different options based on granted permissions
- A universal Navbar should be present on every page (regardless of user type or specific page)
    - Navbar can have different options/buttons based on user access
- Login page will enable user to login or create a new account
- Each user type will have a manage account page allowing them to change their information (new feature)
- Change in theme (colors, images/icons, displayed text/information)
- Innovation to actual survey
- Add authentication for administrators
    - Fix typos/question duplication(s)
    - Possibly change the scale used to answer questions
    - If slider bar is used to answer questions, set default to 50%
    - After completing a page, ensure next page starts user at the beginning of the next page
- Allow program coordinators to view what students (in their groups) have completed the survey
- Allow tool to be viewed and taken on mobile devices (add modularity)

2.2. Engineering Standards

Engineering standards are important for two primary reasons. First is that they uphold safety and privacy standards for the general public. Secondly, they uphold a standard for both software and hardware to be compatible with other devices of different manufacturers, and uphold

quality standards while doing so. Engineering standards allow engineers to design systems that are interoperable between different devices, are safe for the consumer, and can be trusted by the general public.

IEEE 1448a - 1996: Standard for Information Technology: Software Life Cycle Processes

This standard ensures that our team has a specific plan to execute. It defines clear steps and guidelines for understanding, planning, developing and executing when innovating the MySTEMGrowth Survey tool. IEEE 1448a adds compliance methods and clarifications in order to make the cyclical process more adoptable to company practices which provides guidelines on development and software practices. The overarching goal is to provide stability, flow, consistency and organization in software projects.

IEEE 828-2012: Standard for Configuration Management in Systems and Software Engineering

The IEEE 828-2012 Standard defines processes and requirements for configuration management over the life cycles of a software application. It is a set of processes that provide guidelines for tracking and documenting changes to different parts of the program, such as source code, documentation, and databases. The guidelines set out a particular set of rules that require that changes are properly documented and implemented in a controlled manner. This standard helps to support practices such as version control and change tracking.

IEEE 1012-2016: Standard for System, Software, and Hardware Verification and Validation

This standard was created to provide guidelines for the verification and testing of different system applications. It helps to define what product verification looks like, and how to verify that it meets the intended purpose of development. The standard promotes performing multiple levels of testing to ensure that the system meets its intended requirements. It also promotes documentation and risk management to ensure that the application meets its pre-defined requirements and is reliable.

Project Relevance to IEEE Requirements

IEEE 1448a - 1996 (Standard for Information Technology: Software Life Cycle Processes) is very relevant to our project. Working in cycles will allow us to focus on a common goal and ensure everyone is on the same page about the plan moving forward. It ensures that one member won't rush ahead or fall behind, allowing everyone to participate in teamwork more effectively. It also will help us stay more organized and allows us to better retrace our steps in case of any issues.

IEEE 828-2012 is relevant to our project because it helps us to create a set of guidelines for our transition of cloud hosting from Amazon Web Services to Digital Ocean. We currently have a function application to begin with, but we are switching providers due to cost and simplicity of the hosting services. However, we want to have little to no data loss during the transition period with as little downtime as possible. Using this standard to create a set of guidelines, our team will be able to make the transition smoothly and document our progress along the way.

IEEE 1012-2016 will be used by our team to perform additional verification on the existing portion of the application, as well as the new features we implement throughout the next two semesters. While we won't be able to test for all possible scenarios, we will be able to place the application through a wide variety of tests in order to ensure that our product meets the expectations and intended usage as determined by our senior design team along with our advisor.

Additional IEEE Standards

Another IEEE standard our group thought about incorporating was IEEE 29148-2018. This standard is all about requirements engineering, which is relevant to our project due to our goal of adding additional features to the application. While this is a relevant standard, we felt that the previous three standards better incorporated our goals going forward with this project.

Project Modifications for Adherence

In order to adhere to standards that are demonstrated by IEEE 1448a - 1996 (Standard for Information Technology: Software Life Cycle Processes), our team will be completing this project in phases. These phases (currently) include: analysis, requirements, brainstorming/design, development, testing/maintenance and deployment. Stepping through this structure allows us to have a common understanding of the phase/stage we are currently working on. Although everyone will have their own responsibilities, we will be working on the same concept, so we can discuss, help each other and test features on a "scheduled timeline".

Adherence to the IEEE 828-2012 standard will not require many changes to the project, but more importantly the documentation and creation of guidelines for our transition of cloud hosting services. In order to adhere to the IEEE 1012-2016 standard, our group will be administering more forms of testing for the application, including both tests by our team and by actual undergraduate students. It is through adherence to this standard that we aim to develop a ready-to-release application by the end of our project timeline.

3. Project Plan

3.1. Project Management/Tracking Procedures

Our team has chosen to adopt a mix of both agile and waterfall project management style. In an overall broad sense, we believe that the waterfall approach is the most beneficial. We think identifying all requirements and desires for the project will help give us a clear vision when we start designing/implementing. Ensuring the design/implementation of the project is clearly laid out and agreed upon will ensure testing/maintenance are applied to all specified areas.

However, when working with specific individual issues, we think it is more beneficial to our team and overall project to ensure we are practicing the agile management style. It is important to ensure the defining/planning of an issue is fully understood and agreed upon by our client and team. Once this is the case, we can proceed to the design phase. Once the design is complete, we

can then test, and so on. When working with single features, we believe it is important to use this 'cyclical' motion to ensure we meet the desired requirements by our client and ensure we keep the overall flow of the project. Making small progress often for our features can allow the site to always be fully functional and reduces the chances of conflicts and incompatibility with other features/systems.

Our team has been utilizing the 'issue boards' feature available on Git in order to track progress. This feature allows us to create features we intend to add to the current state of the survey. We are able to label/categorize issues to ensure they get implemented in all required parts of the project (frontend, backend, cloud). We are also able to claim issues so we know who is responsible for what features and add timeline goals with dates. We can further organize details by labeling them with specific categories; such as security, UI, database, etc. Overall, the Git issue boards are an easy method we can use to ensure we are making progress in a timely manner and keep everyone organized and accountable.

3.2 Task Decomposition

Tasks are divided into 4 primary categories; Frontend, Backend, Security, and Cloud. Some of the tasks may overlap slightly, but we decide to place them within the list based on their primary area of needed development. The list is as follows:

- Frontend
    - Inform users of missed questions
    - Allow Program Coordinators to see who has completed survey
    - Set slider default to 50% for survey
    - Add change password feature
    - Mobile support
    - Explore using Qualtrics for survey
- Backend
    - Save survey progress for partial survey
    - Add delete user feature
    - Add tokens and cookies
    - Use hashing for login and new user
    - Reduce object size
- Security
    - Change passwords to store as hashed
    - Attempt Breaches
    - Protect Vulnerable Endpoints
- Cloud
    - Switch URL Hosting to Digital Ocean
    - Transfer databases from AWS to Digital Ocean
    - Fix CI/CD pipeline

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone 1: Switch from AWS to Digital Ocean + CI/CD pipeline fix

While this milestone is not easily quantifiable, we do have a very clear definition of success. By the end of this milestone all aspects of the application should run on Digital Ocean; Database, URL Hosting, Frontend, Backend. The CI/CD pipeline will also be updated to ensure that all future features changed or added by the development team will be automatically updated to the new hosting application. This will allow our team to cease all operations on AWS and make an effort to create a more intuitive host for future administrators as well as lower costs.

Milestone 2: Improving the Survey

We would look to quantify this milestone through user surveys and feedback from using the application. The goal would be to have over an 80% satisfaction rate with our user base. Throughout this milestone we would like to implement aforementioned features such as defaulting sliders to 50%, saving partial surveys for later completion, and informing users if a question was missed. The overall goal of this milestone would be to improve the user experience while taking the survey. While the current system is functional, it lacks some quality of life features that our team would like to implement.

Milestone 3: Security

Security is an important feature that any modern application should have, especially when said application is asking users questions identifying traits about themselves. The current application lacks most security features, including plaintext passwords and usernames when viewing the console, survey results being easy to find for other users, etc. The result of this milestone will be bringing the application up to expected security standards, ensuring that users information is only available to those who have permissions to access it, such as administrators and program coordinators.

3.4 Project Timeline/Schedule

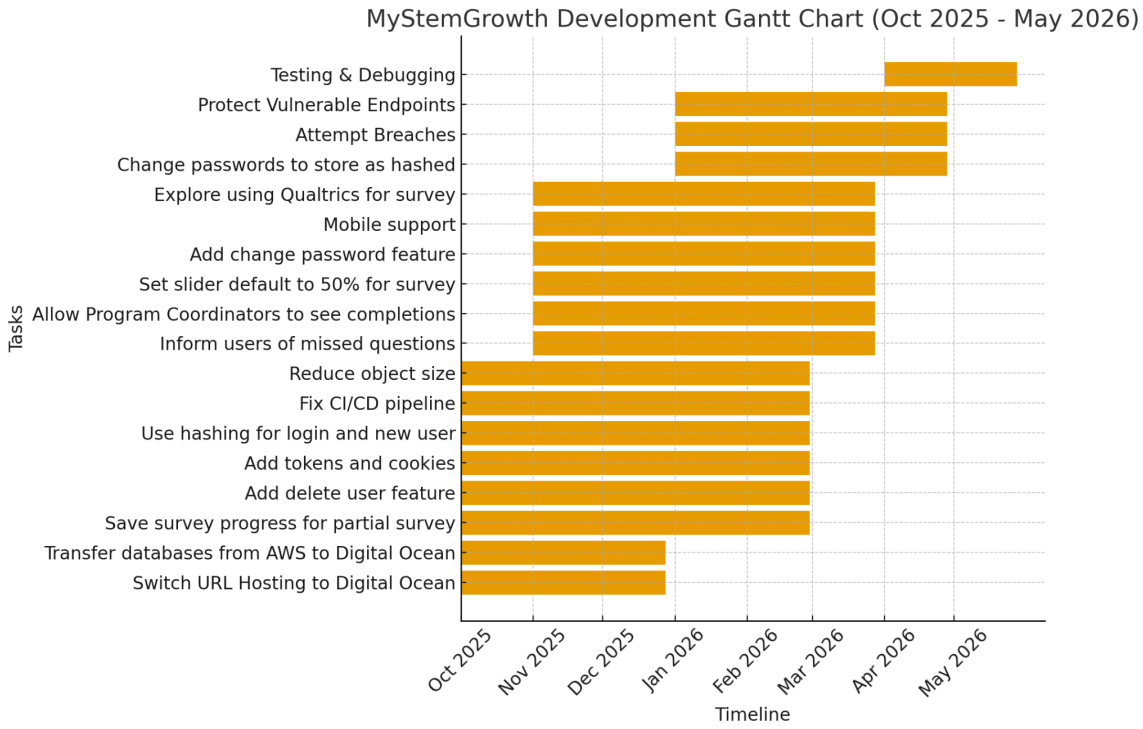

Figure 3: Gantt Chart

3.5 Risks and Risk Management/Mitigation

The cloud transition from Amazon Web Services to Digital Ocean holds a few risks in losing user data between database services. This risk can be estimated to be about 25%, but can be easily mitigated by maintaining the original database until we can ensure that all data has been successfully transferred.

Most of the frontend and backend tasks carry virtually no risk. They are all software based features that can be tested on a Git branch to ensure that they work before they are implemented. The biggest risk is that the carry would be unpredictably affecting another feature of the application, but in that case the version of the application can be rolled back quite easily. Due to the little risk this imposes we estimate that the risk factor is about 5%.

The final risk that the group identified was the switch to Qualtrics. As a group we have not committed to the switch to Qualtrics for a survey tool for a couple of reasons. First is that we want to have a smooth user experience, and we worry that using Qualtrics would result in extra windows needing to be opened and interrupt the user's interaction with the application. The second issue is that we are unsure if we will be able to automatically update the database with information from Qualtrics to allow users to view their survey program, or for higher level users to view their subordinates survey information. Due to these factors, we assess the risk of Qualtrics to be 50%. If

Qualtrics as a tool turns out to not be a valid approach, we can continue to use our custom implementation of the survey for data collection.

3.6 Personnel Effort Requirements

Task	Average time/week (per person)
Switch Clouds (AWS → Digital Ocean)	3
Improve security	3
Modify survey (questions + format)	3
Improve UI (for all users)	4
Improve storage	2
Add new user features	3

- Switch Clouds
  - Migrate all current data
- Improve security
  - Hash + salt passwords
  - Add tokens/cookies
  - Protect endpoints
  - Add authentication
- Modify Survey
  - Maybe migrate to Qualtrics
  - Fix answer methods
  - Fix questions (duplications/misspelling)
  - If keeping sliders - default at 50%
  - Add save progress button
  - Auto scroll to top
- Improve UI
  - Add modularity (correct format for small screens)
  - Change PC view from cards
  - Allow PCs to view who hasn't submitted survey (in managed groups)
  - Rework formatting for all views (students, PCs and admin)
  - Update About page
  - Add info to Resources page
- Improve storage
  - Delete unused fields
  - "Unsubscribe" from certain cloud options (saves money)
  - Remove abandon accounts
- Account features (all users)
  - Manage account info - (change password, email, name)
  - Delete account
  - Add a "test" mode for PCs and Admins

    - Add extra authentication method for Admin
- Fix Github CI/CD pipeline
    - Write new CI/CD script
    - Add endpoints of Digital Ocean server
    - Test CI/CD script
    - Implement script for push requests in Gitlab

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	Protects student data and promotes well-being by giving learners clear feedback on their research experience and growth. Security controls help prevent misuse of personal information.	Encrypted auth and least-privilege roles. Clear progress + "resume later" to lower survey fatigue. Admin tooling that avoids accidental exposure of student results.
Global, cultural, and social	Respects diverse backgrounds and programs by using inclusive language, accessible design, and flexible scales/question types. Supports different academic cultures and course structures.	Mobile-first layouts. Wording review to avoid cultural bias. Role views for admins, coordinators, and students.
Environmental	Digital-first delivery reduces paper and travel overhead; efficient cloud choices lower compute waste. Development practices consider energy usage of hosting and CI/CD.	Online surveys instead of printed packets. Adjusting the database size to match what we actually need. Static asset caching/CDN. Deleting stale accounts/data to reduce storage footprint.
Economic	Keeps costs sustainable for universities and research programs. Streamlines coordinator/admin time. Supports student success and program improvement that can attract funding.	Predictable hosting. Automated CI/CD to cut maintenance. Batch export of results for grant reports. low training/onboarding time due to simpler ops.

4.1.2 Prior Work/Solutions

One of the biggest advantages our application offers is the ability to collect survey data virtually, allowing for easier exportation for research purposes, or simply to make it easier to allow the student to view how their personal outlook and abilities have changed throughout their research experience. Using digital processes to collect information allows data to be processed more time efficiently, and has been used in a variety of different applications to help assess data and provide analytics [1]. The survey itself also holds similarity to previous projects that involved assessing undergraduate research program effects on students, such as the SURE survey. This survey was created to help collect student responses and evaluate things like student retention and

educational development based on the survey responses, similar to one of the potential benefits listed for our project [2]. One of the primary actions they performed to solve issues was to ensure data comparability and standardization to ensure that results collected would be effective if program coordinators or researchers wanted to use the data later on. The previous group, team sdmay25-24, has also had a lot of documentation and a solid code basis for us to work on and help us determine fixes and improvements we wanted to implement, as well as what worked well [3]. One of the biggest shortcomings we found was with the security of the application, as there was not much documentation, and there were notable security concerns with the application when we received it. However, there were many benefits to be had with having previous documentation as it made understanding the structure of the application very easy to understand and begin working on.

[1] M. M. Hanbury, E. L. Cox, J. M. Culley, and C. M. Pruinelli, "A web-based management application to improve data collection in an emergency care research network," *Journal of the American Medical Informatics Association*, vol. 26, no. 12, pp. 1547–1554, Dec. 2019. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC6351988/

[2] D. Lopatto, "Survey of Undergraduate Research Experiences (SURE): First findings," *CBE—Life Sciences Education*, vol. 3, no. 4, pp. 270–277, Dec. 2004. [Online]. Available: https://www.lifescied.org/doi/10.1187/cbe.04-07-0045

[3] Iowa State University, "MyStemGrowth – Senior Design Team SDMAY25-24," *Department of Electrical and Computer Engineering, Iowa State University*, 2025. [Online]. Available: https://sdmay25-24.sd.ece.iastate.edu/

4.1.3 Technical Complexity

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles

Our application consists of many different subsystems, all of which I will list and describe here:

1.1. Frontend: This component uses React and Javascript to implement the user interfaces that our undergraduate users and other roles will interact with. It is the primary tool that is used to collect data and display it in a clear and readable manner, whether that be for program coordinators to view how their program is influencing undergraduate students, or to allow students to see have they have grown according to their personal experience.

1.2. Backend: The backend is what allows our system to communicate with our databases and hosting environments, allowing for authentication and IO handling to create a seamless experience for users regardless of their role.

1.3. Database: The MySQL database management system allows for the application to store and sort data such as account information as well as survey questions and their response data.

1.4. Hosting: With the current use of AWS and soon to be Digital Ocean, these systems allow us to host our application as well as offer many industry standard tools for secure data transmission and role based authentication.

2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

    2.1. Scalable Design: Our application requires the use of industry level design for the purpose of scalability. Our application needs to be able to grow with its usage, as it has the opportunity to be used in a number of research programs.

    2.2. Data Privacy/Security: Our application will implement industry standard security features, such as hashing and the use of cookies, to ensure that user data such as passwords will be safe.

    2.3. Cross-platform accessibility: While the current application works on a browser, our team is working on UI improvements to help make the application easier to use, as well as offer a more readable view for admins or Program Coordinators to look over all responses for the respective programs that they oversee. In addition to that, we are also developing mobile support to allow users to take the survey or view their data on their mobile devices.

    2.4. Automatic Deployment: We are implementing an automated CI/CD pipeline to allow our team, or any team following, to continue to develop the application and seamlessly integrate these changes to the web application without any interruption to service.

4.2 Design Exploration

4.2.1 Design Decisions

Switching from AWS to Digital Ocean:

Our team chose to migrate the MySTEMGrowth Survey's infrastructure from AWS to Digital Ocean after coming to an agreement that AWS was overly complex and costly for our project needs. Digital Ocean's App Platform and Managed Database services offered a simpler setup, predictable pricing, and an easy way to integrate with our existing Node.js backend and MySQL database. This transition will enable easier onboarding for future developers with less of a learning curve to get started. It also improves security through encrypted environment variables and managed backups while maintaining scalability and reliability. Overall, the move to Digital Ocean supports our goals of affordability, maintainability, and long-term sustainability for the MySTEMGrowth platform.

Mobile layout support:

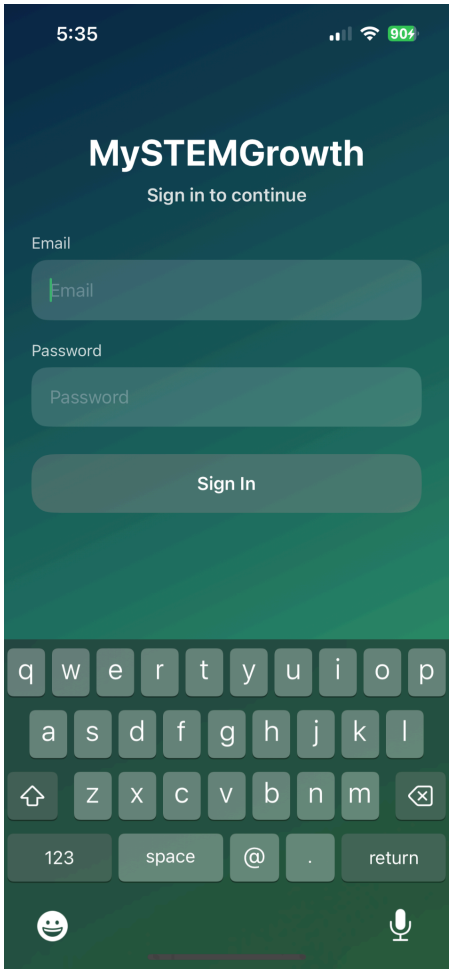

Figure 4: Mobile Login

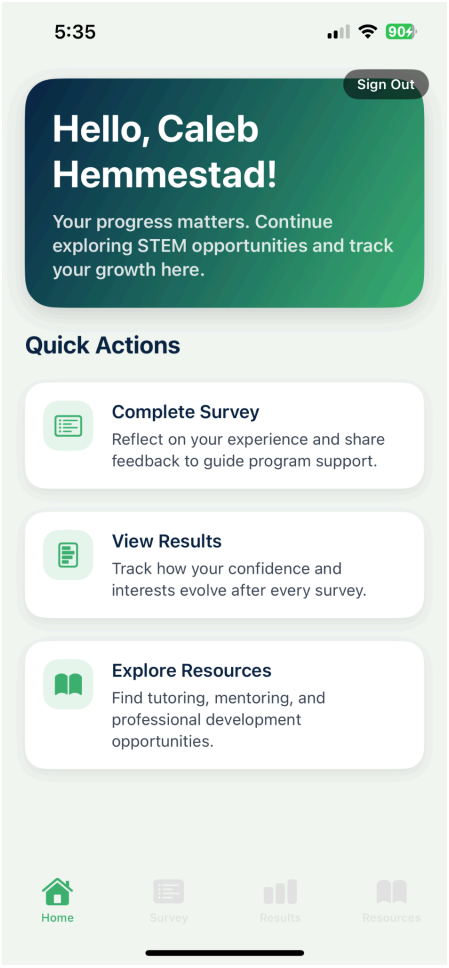

Figure 5: Mobile Home

Program Coordinator layout changes:

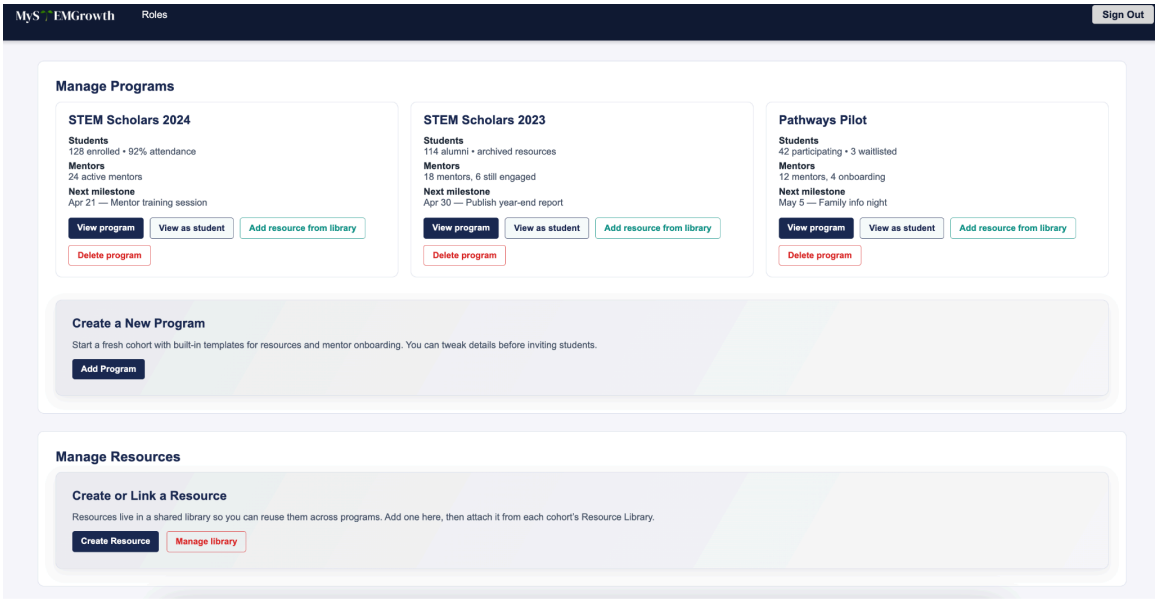

Figure 6: Prototype Program Coordinator View

Improved security with tokens and hashing:

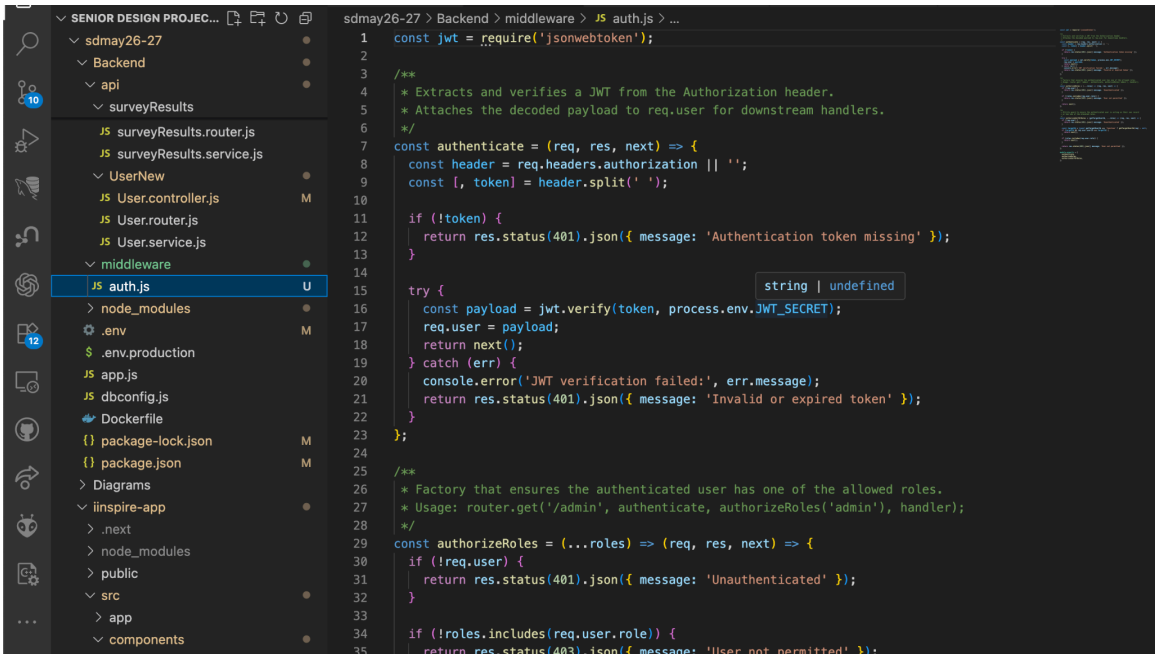

Figure 7: Token Code Snippet

4.2.2 Ideation

For the design decision of switching from AWS to Digital Ocean, our team used a collaborative brainstorming and comparison session to explore multiple hosting and cloud service options. We used a lotus blossom–style ideation approach, starting with the goal of finding a cloud

platform that meets our needs while reducing the cost and keeping the app easy to maintain and scalable in the future, then expanding outward with potential solutions and sub-ideas.

We initially identified five good options:

1. **Amazon Web Services (AWS)** – Continue using the existing infrastructure.
2. **Digital Ocean** – Migrate to a more developer-friendly and cost-effective environment.
3. **Google Cloud Platform (GCP)** – Leverage built-in integrations with CI/CD and analytics.
4. **Microsoft Azure** – Utilize advanced security and enterprise hosting capabilities.
5. **Heroku or Vercel** – Use a simplified app hosting service with auto-deploy pipelines.

After going through all the options, we eventually decided to go ahead with the switch to Digital Ocean which we will start working with as soon as we get access after being helped with ETG.

4.2.3 Decision-Making and Trade-Off

After looking at all five possible hosting options, our team evaluated each using a weighted-decision matrix based on five criteria: cost (25%), ease of setup (20%), integration with our stack (20%), security (20%), and scalability (15%). Each option was scored from 1 (poor) to 5 (excellent).

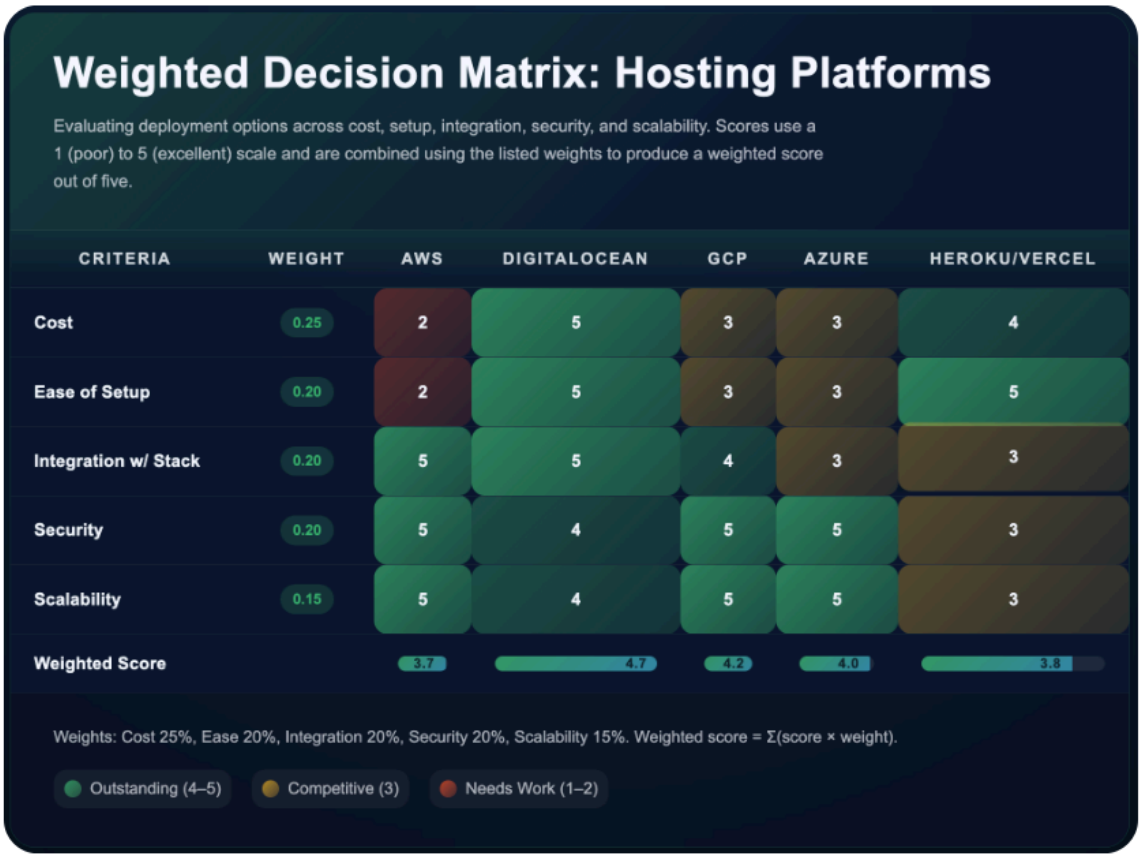

Figure 8: Weighted Decision Matrix

Digital Ocean achieved the highest overall score by meeting most of the criteria that we were looking for. While AWS and Azure offered stronger enterprise-grade scalability, their pricing and configuration complexity made them less practical. Heroku and Vercel were intuitive but lacked control over databases and back-end customization.

The trade-off in choosing Digital Ocean was accepting slightly fewer enterprise integrations in exchange lower cost and less of a learning curve for future developers.. We think this decision ultimately supports MySTEMGrowth's goals of maintainability, reliability, scalability, cost efficiency, and long-term sustainability.

4.3 Proposed Design

4.3.1 Overview

At a high level, the MySTEMGrowth tool consists of two main parts, the front end and the back end. The front end is the user interface of the program. This handles everything a user will need to interact with and use the tool. The other major component is the back end. The back end handles all data manipulation and storage. Inside the back end, the survey results are compiled and

calculated to give the proper graph to the user. The front end and the back end are very entwined. For everything the user does in the front end, something in the back end will be run to facilitate that. .

4.3.2 Detailed Design and Visual(s)

MySTEMGrowth is a role-based web application (Student, Program Coordinator, Admin) delivered as a responsive React/Next.js client that talks to a Node.js/Express API, backed by a managed MySQL instance on DigitalOcean. Persistent assets (exports/logs) live in object storage. Authentication uses hashed passwords and session cookies/tokens; authorization is enforced via RBAC on every API boundary. CI/CD builds and deploys the app platform components on push. See Figure 1 (System Architecture) and Figure 2 (Components and Data Paths).

Figure 1. System Architecture

High-level view of MySTEMGrowth showing client apps, backend services, and cloud resources on DigitalOcean.

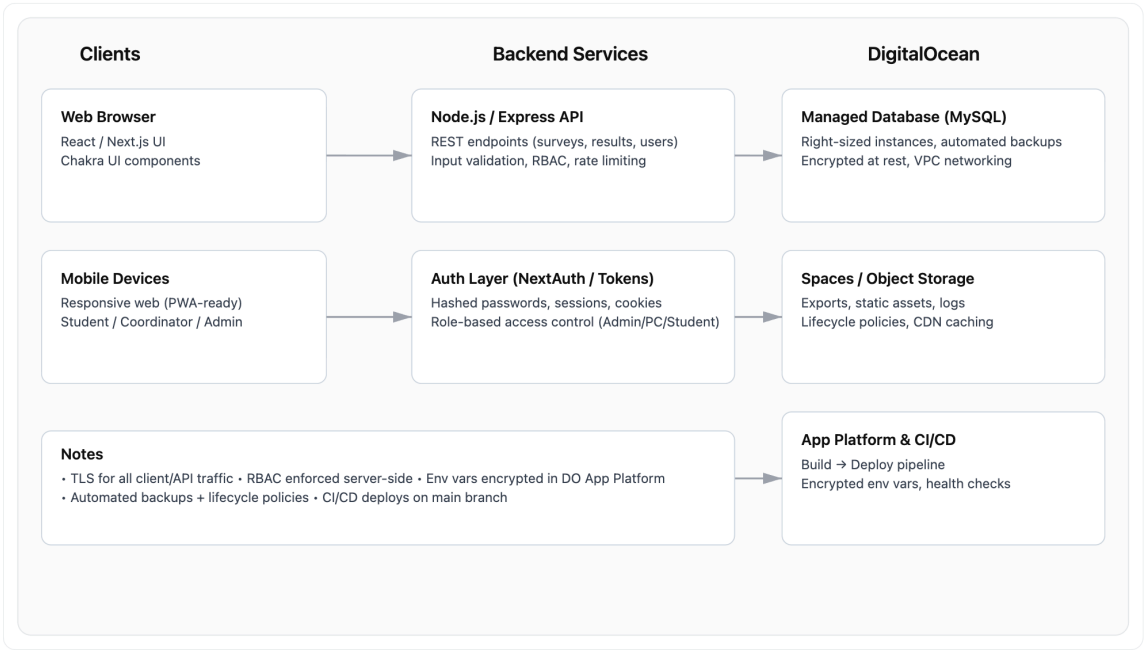

Figure 9: System Architecture

Frontend (React/Next.js)

- **Survey UI**: renders question types (slider/likert/text), validates inputs, supports “save progress.”
- **Results Dashboard**: charts/filters for individuals and cohorts; CSV export.
- **Role Views**: conditional navigation and routes per Student/PC/Admin.

- **HTTP Client Layer**: fetch wrapper adds auth headers, retries, and uniform error handling; maps DTOs to view models.

- **Integration**: communicates with `/auth`, `/surveys`, `/results`, `/users` endpoints over HTTPS; receives JSON.

Backend (Node.js/Express)

- **Controllers**: route handlers for auth, surveys, results, users; input validation (e.g., zod/express-validator), pagination.

- **Service Layer**: business rules (e.g., PCs can only view their groups), audit logging.

- **Auth & Sessions**: password hashing (bcrypt/argon2), secure httpOnly cookies, optional JWT for API clients, token rotation.

- **Data Access Layer**: parameterized SQL; transaction boundaries for multi-write ops.

- **Integration**: uses a pooled MySQL client; emits structured logs/metrics for ops.

Data Stores (DigitalOcean)

- **Managed MySQL**: users, groups, surveys, questions, responses, response_items; automated backups and VPC isolation.

- **Spaces/Object Storage**: export files, static assets, optional logs with lifecycle policies.

Operations (CI/CD & App Platform)

- **Build/Deploy**: on main branch, build containers, run unit tests, deploy API and static site; health checks and rollbacks.

Figure 2. Components and Data Paths

Key subsystems with their roles and how data moves from UI to database.

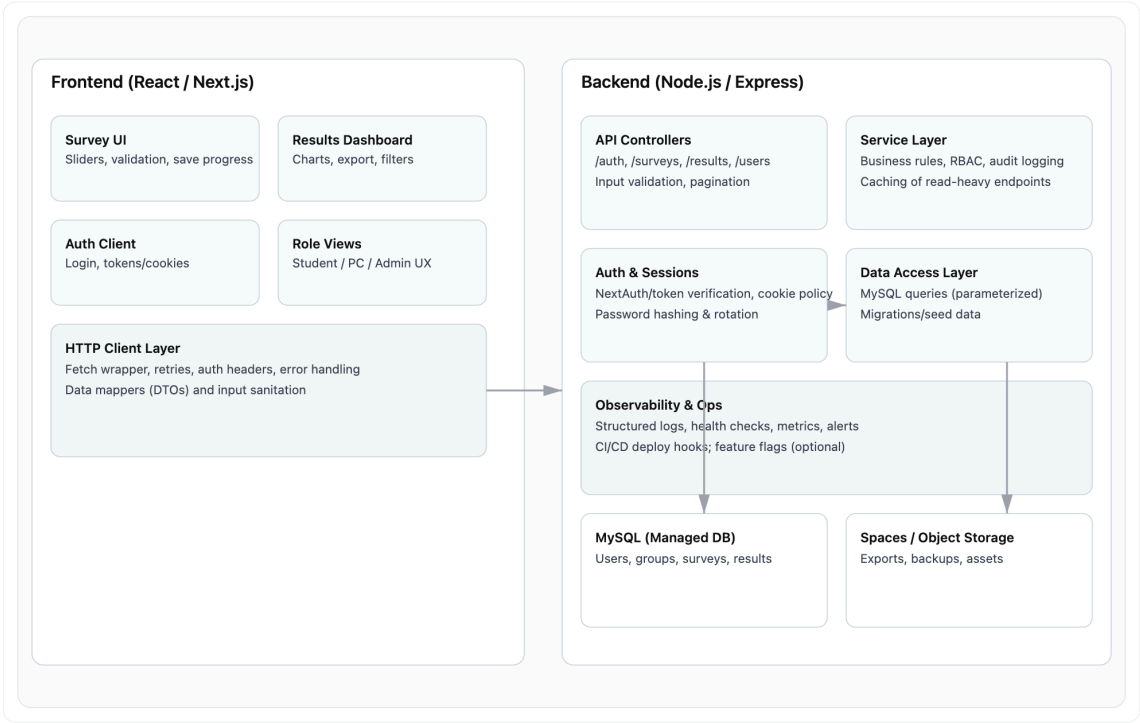

Figure 10: Components and Data Paths

Figure 4 (Mini ER Data View) shows the core entities and relationships: Users (PK `user_id`) join Groups via GroupMembers; Surveys contain SurveyQuestions; groups are linked to surveys through Assignments; Responses capture a user's submission for a survey (with denormalized `group_id` at submit time); ResponseItems store per-question answers. This schema supports role-scoped reads, longitudinal pre/post comparisons, and efficient aggregation. (Crow's foot can be added if your class prefers that notation.)

Representative API surface (abbrev).

- `POST /auth/login` → set session cookie after hash verify; returns user/roles.
- `GET /users/me` → returns profile + role; used to gate client routes.
- `GET /surveys/:surveyId` → survey + questions (role-agnostic).
- `POST /responses` → start or submit responses (supports partial save, `is_complete` on items).
- `GET /results?groupId=…` (PC/Admin) → scoped aggregate + per-student results; Students use `GET /results/me`.

- `GET /exports/group/:id` (PC/Admin) → generates CSV to Spaces (pre-signed download).

AuthN/AuthZ model.

- **Authentication**: email + password (hash stored), secure httpOnly cookie; optional short-lived JWT for programmatic access.

- **Authorization**: RBAC middleware on every protected route; Students only see their own data, PCs see assigned groups, Admins have elevated read/export scopes. See Figure 3 (Login & Session Flow).

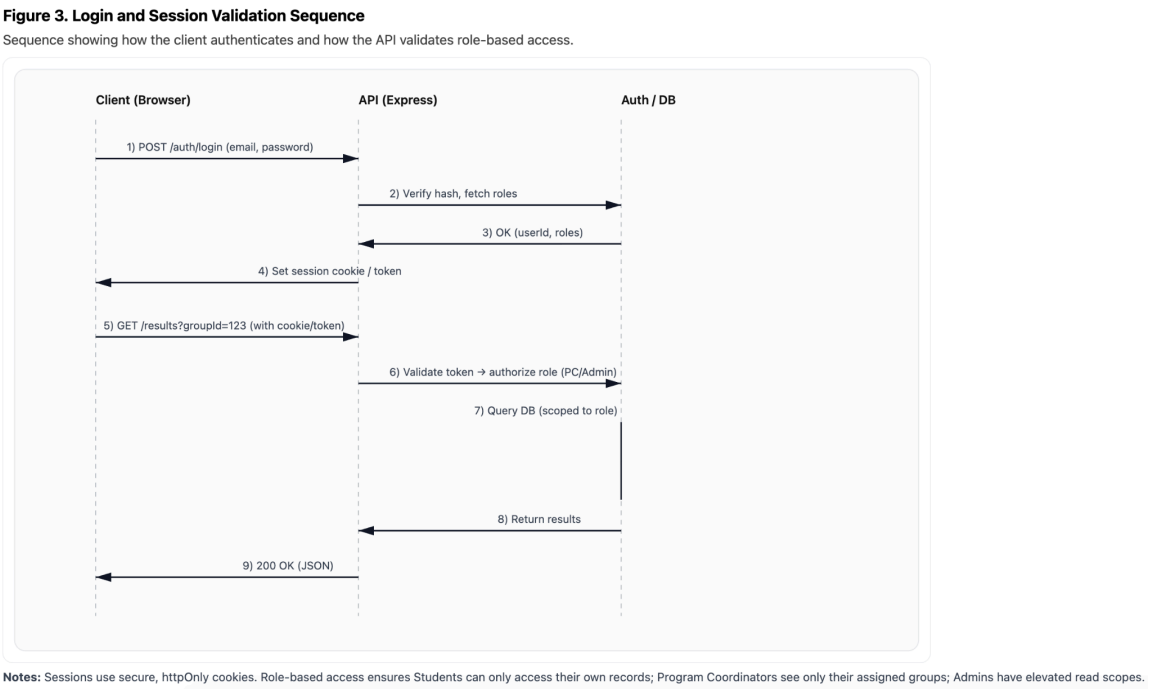

Figure 11: Validation Sequence

Validation, errors, and resilience.

- **Validation**: server-side schema validation; client mirrors constraints for better UX.

- **Errors**: uniform JSON shape `{code,message,details}`; 4xx for validation/permission, 5xx for server.

- **Resilience**: DB connection pooling; exponential backoff/retries for transient failures; idempotent POSTs where feasible.

Performance & scalability notes.

- Read-heavy endpoints (e.g., cohort dashboards) support **pagination and simple caching** at the service layer.
- **Right-sized DB instances** and query indexes (on `user_id`, `group_id`, `survey_id`) keep latency low.
- Static assets are cached; API uses gzip/br compression.
- Horizontal scaling path: add API replicas behind the App Platform; DB vertical scale as needed.

Security & privacy.

- Password hashing (argon2/bcrypt), TLS everywhere, least-privilege DB credentials, parameterized queries to prevent SQLi.
- Encrypted environment variables, role-scoped queries, audit logs for sensitive reads/exports.
- Backups tested via periodic restore drills; PII access limited to Admin/PC roles per policy.

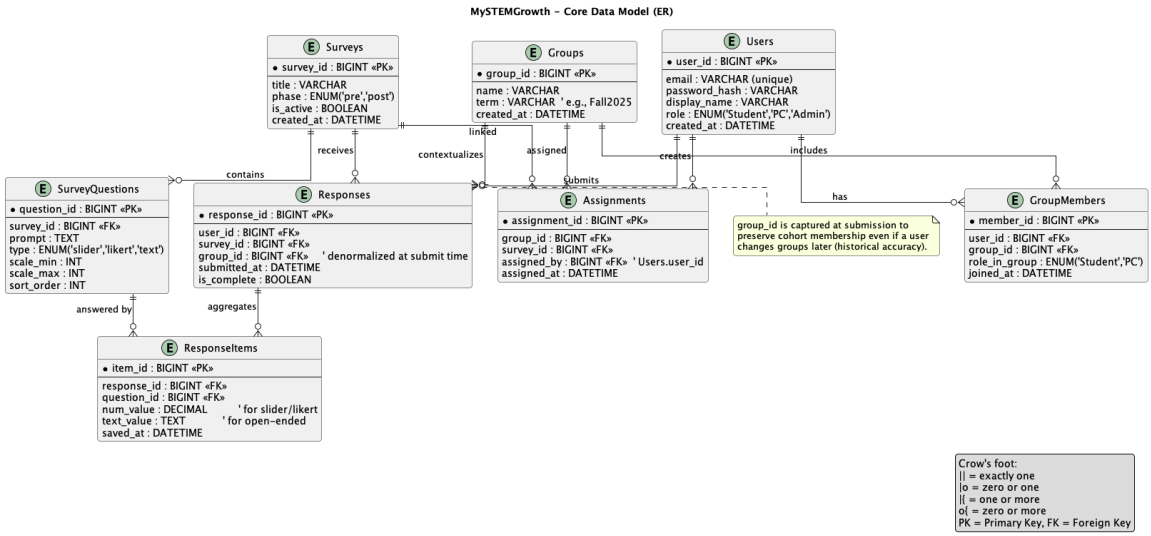

Figure 12: Core Data Model

4.3.3 Functionality

A standard user (student) would start by making an account with the code given to them by the program coordinator. After making the account, the student would have the potential to have extra required demographic information, depending on the program coordinator's choice. After that, the next major step would be for the student to take the survey itself. Once the survey is complete, the student will be able to see the data from their survey. If the student were to come

back and retake the survey at a later date, the graph at the end would show a comparison of the growth, rather than just an isolated result.

A program coordinator will create an account, and then create a program. Once the program is created, the code will become available for the program coordinator to give out to students. The program coordinator will also be able to demonstrate the survey to students from their program coordinator account. After students have taken the survey, the program coordinator will be able to go look at student's results to understand where they are at. This includes both the single data point graph and the comparison graph. Program coordinators will also be able to link resources to a program for students to access as necessary.

An administrator will be more removed from the survey itself. The focus of the administrator is to make sure that all programs are set up and running smoothly. They will be able to delete/modify accounts, change settings in specific programs, and look deeper into any program they desire. This role is the highest level of access of the three roles, and as such, the role given out to the least amount of people. Administrators will most likely not directly interact with students, but only with program coordinators.

4.3.4 Areas of Concern and Development

The current state of the application is operational, but there are 3 main areas of improvement that our group is focusing on in the immediate future. The application functions well for demonstration purposes, but lacks in some important areas that our team deems important before a full release. The first area that we want to improve is security. The primary security concern is that all variables are saved in local storage to allow for persistent login, but that is an easy vulnerability that allows a malicious user to change their role or view passwords in plain-text. Currently we have 2 people on our team working on hashing passwords and implementing cookies. The 2nd area we want to improve is mobile device usage. The application works well on a browser, but currently the mobile element is an awkward scrollable element that does not look very professional. The team is working within React to set pixel width settings to improve the mobile view. The final area would be simplifying the cloud hosting services. By switching to Digital Ocean, the hosting service will be more cost effective for a service, as well as provide a more simplistic service that will be easier for future hosts or developing teams to use.

4.4 Technology Considerations

Front-end:

React: JavaScript library used to create reusable components for interactive user interfaces

Pros:

- Reusable components
- Easy to integrate with other libraries
- Fast rendering

Cons:

- Steep learning curves for beginners
- Frequent updates can compromise feature compatibility

Trade-offs: Allows for fast, reusable UI development with complexity of frequent updates and steeper learning curves.

Next.js: React framework that enables server-side rendering and static site generation

Pros:

- Easy routing system
- Full-stack capabilities with API routes

Cons:

- Larger bundle size
- Steep learning curve
- Limited flexibility with custom configurations

Trade-offs: Offers enhanced performance and full-stack capabilities, but is more complex than using React by itself.

Chakra: React component library that provides customizable UI components for web applications

Pros:

- Prebuilt, responsive components
- Easy styling with consistent design system

Cons:

- Difficulties overriding when styling or complex layouts
- Limited design capabilities compared to competitors (CSS or Tailwind)

Trade-offs: Faster UI development with styled components, yet, can limit design flexibility and increase storage usage.

Front-end design solutions and alternatives: We inherited this design from two previous senior design teams. We have decided it would be in our best interest to continue working with the libraries/addons they have already implemented. We have spent a lot of time in CPRE 4910 reviewing the current code and learning more about the libraries used. This allows us to practice and ensure we understand the capabilities when the time comes to implement desired features, presented by our clients.

Back-end:

Node.js: Javascript runtime that allows developers to build applications using a non-blocking, event driven architecture

Pros:

- High performance
- Scales easily
- Large ecosystem with ‘npm’ packages

Cons:

- Frequent updates can compromise feature compatibility
- Callback based code can be complex

Trade-offs: Provides high performance, scalability for server side development but struggles with CPU intensive jobs/processes.

Postman: Collaboration platform for API development

Pros:

- Easy API testing and debugging
- Collaboration features for teams

Cons:

- Limited (free) testing/features
- GUI dependency

Trade-offs: Makes API testing and collaboration easy, but is resource heavy.

Back-end design solutions and alternatives: Many of our team members have experience with at least one of the two main backend technologies implemented. We could switch to alternatives, such as cURL or Django, however, since these technologies have already been implemented by previous teams, and we have experience with them, we decided it would be in our best interest to continue utilizing their features, already integrated with the site.

Security:

Bycrypt: Library that supports hashing and password verification

Pros:

- Easy to implement
- Hashing resistant to brute force attacks

Cons:

- Slower algorithm
- Adds computational overhead to authentication

Trade-offs: Requires careful/tedious configuration, but ensures very safe password security and provides easy hashing and salt implementation.

Tokens: Pieces of data used for authenticating/authorizing users (JWTs)

Pros:

- Carries user data securely
- Stateless authentication (no server-side needed)

Cons:

- Must be securely stored
- Revocation/expiration can be tricky to implement

Trade-offs: Require careful storage and tedious implementation but enables stateless, cross-service authentication.

Security design solutions and alternatives: Bcrypt and tokens are something new we are adding to the project, since security of the website was previously lacking. We could use alternatives such as multi-factor authentication or API keys, but our cybersecurity developers have already begun learning how to implement our chosen security features.

Cloud:

GitHub/Git: Platform used for version control and collaborative software development

Pros:

- Easy collaboration and code sharing
- Integrates with CI/CD
- Tracks changes/maintains history

Cons:

- Limited free usage
- Complex for advanced workflows

Trade-offs: Simplifies version control and collaboration but has limited free usage and can become complex for larger projects.

Digital Ocean: Cloud infrastructure provider for managing applications

Pros:

- Simple and user friendly interface
- Affordable pricing
- Fast deployment

Cons:

- Fewer advanced features than competitors (AWS)
- No team experience using this platform

Trade-offs: Offers simple, affordable cloud management solutions while lacking some advanced features and scalability.

Cloud design solutions and alternatives: One of our biggest decisions in the early stages of this project was switching cloud providers, from the previously implemented AWS. This switch was one of the first changes we started on, due to the time, complexity and learning curve increases. We would have stuck with AWS, but we believed that switching to Digital Ocean would reduce cost and decrease complexity for maintaining the site.

4.5 Design Analysis

Our cloud developers have begun transitioning from the old cloud service, AWS, to our new alternative, Digital Ocean. We are currently in the processes of moving databases, updating the CI/CD pipeline and implementing new features/protocols that Digital Ocean provides. Digital Ocean has all the capabilities as AWS (needed for the MySTEMGrowth Survey Tool), while making the process more user friendly. While researching this alternative, we also estimated we would be able to reduce the cost by switching clouds, as AWS charges more, due to its extensive feature options, most of which would not be utilized by the survey tool.

Another early focus has been security. Our cybersecurity developers have been testing misconfigurations in the current state of the survey tool. We have discovered major security vulnerabilities and have already started to implement fixes for them. We are working on adding tokens for user authentication and implementing hashing/salting user passwords. One of the main improvements we intend to complete is securing user data/information. Not only should sensitive account information (email, passwords, etc.) be protected, but also the results of students after completing the survey.

Not much has yet been physically changed and deployed within the website files itself (frontend and backend features), due to the extensive planning we have been conducting. We have been discussing improvements with our clients, to ensure we understand their vision for the product. We have presented a demo UI and a list of features we intend to implement into the current state of the app. Once we have clarification and confirmation, we intend to start implementing the changes and start making changes to the website. We also have been hindered to start writing/implementing code, due to the processes of changing clouds. We wanted to ensure we have the cloud setup before changing the functionality, to ensure the compatibility will work with Digital Ocean.

5 Testing

5.1 Unit Testing

- Backend service modules (e.g., Backend/api/UserNew/User.service.js (lines 20-134), Backend/api/surveyResults/surveyResults.service.js (lines 15-94), Backend/api/questions/questions.service.js (lines 16-18)) should have Jest test suites that stub the shared MySQL pool and verify each exported function: success paths, error propagation, and edge cases (missing parameters, zero results, constraint violations).
- Controllers such as Backend/api/UserNew/User.controller.js (lines 4-137) can be unit-tested with Jest + Supertest by mounting the router in an Express app and asserting status codes/payloads given mocked services and JWT secrets.

- On the frontend, utility hooks/components (iinspire-app/src/components/TheSurvey.tsx, iinspire-app/src/components/results/Profile.tsx) can be tested with React Testing Library + Jest DOM to ensure state transitions (phase changes, pagination, submission formatting) and derived text (card narratives) respond correctly to props/mock fetches.
- Where logic is mostly data transformation (e.g., grouping answers in TheSurvey), add pure helper functions and cover them with straightforward Jest assertions.

5.2 Interface Testing

- REST interfaces between frontend and backend (e.g., survey question fetch GET /api/questions, survey submission POST /api/survey-results, login POST /api/login) can be validated with pact-style contract tests: define OpenAPI schemas or JSON Schema expectations and ensure both consumer and provider tests stay in sync. Tools: Swagger/OpenAPI for spec, Jest + Supertest + jest-openapi for server response validation, and Mock Service Worker (MSW) for client-side interface simulations.
- Composition tests should exercise flows where multiple modules interact, such as user creation (frontend form → User.controller → User.service queries) or survey submission (React state → formatting payload → surveyResults.service normalization). Write Cypress component tests or Playwright interaction scripts that mount subsets of the UI with MSW stubs to assert full request lifecycles.

5.3 Integration Testing

- Critical paths include:
    - Authentication & Authorization: POST /api/login issuing JWT tokens (requirements: secure access by role). Integration suite should spin up the Express app against a test MySQL (or dockerized DB) populated with seed data, then run Supertest flows to ensure tokens gate endpoints like /api/usersprogram.
    - Survey ingestion and analytics pipeline: fetching questions, filling the survey (TheSurvey) and persisting results via surveyResults.service before the dashboard (Profile) renders normalized data. Cypress/Playwright end-to-end tests can automate a student logging in, completing the survey, and verifying stored results via API assertions.
    - Program management for coordinators/admins: API endpoints for program CRUD and student listing (/api/usersprogram, /api/newprogram, /api/program-survey-results). Integration tests should verify role-based access and data joins across Users/Program/SurveyResults tables.
- Tooling: Docker Compose (MySQL + backend) for consistent environments, Jest + Supertest for API integration, Cypress or Playwright for browser-level verification against the Next.js app.

5.4 System Testing

System level testing varies depending on what stage of the application pipeline is being tested. For frontend functionality changes, the changes are tested on a branch from within Android studio before being merged and implemented onto the main host. Tools included within Android studio allow the team to see and experiment with the elements as they are working on them, as well

as test how different visual or logic changes affect the flow of the user experience. For backend testing, tools like Postman are used to ensure that endpoints deliver expected results, ensuring that critical functionalities like login and survey results are reliable and functioning properly. For changes or reconfiguration in hosting, multiple instances of the application are used. For example, when adjusting resource constraints and utilization or uploading new application versions within the cloud server, a 2nd instance of the application is launched. It is only when that 2nd instance is tested and validated that the URL is pointed towards the new instance and the previous one is taken down. This style of testing ensures that the application has little to no downtime when implementing new instances of the application.

5.5 Regression Testing

We work to ensure that new additional features do not break old functionality by using standard software practice, as well as version testing and control. When implementing new features, we work off of branches, individually adding features and testing functionality. Before any of these branches are merged, we ensure that we create new, independent functions to provide any additional functionality we need to implement said new features. Each of these new functionalities are tested on their individual branches before being launched onto the main platform. In the case that functionality critical to the program made it past testing, we are able to roll back to a previous working version of the application.

Critical features to the application for students include basic functionality such as sign in, logout, ability to take the survey, and ability to view results. While the look and feel of such functions has changed slightly and may continue to be tweaked, the underlying logic has remained the same, and any future changes will undergo thorough testing before any attempt to launch new versions. Similar practices will be implemented for any changes made to Admin or Program Coordinator roles. Critical functionalities for these roles include ability to add or remove members from programs, view program information, and export program data. While updates may be made to these functionalities, each change in functionality is rigorously monitored to ensure that these critical functions are operations at all times.

5.6 Acceptance Testing

To ensure the tool meets all functional and non-functional requirements, acceptance testing will be carried out in collaboration with our client. We demonstrate completed features in meetings, walk through the application with the client, and validate that each component behaves as intended. During these sessions, administrators, coordinators, and participant workflows will be tested, as well as creating groups, distributing surveys, completing a survey, and viewing results to confirm the tool aligns with the operational needs.

5.7 Security Testing (if applicable)

Since we received an ‘inprogress’ system, the first thing we need to do is test the security of the current state of the website. From there, we can note possible vulnerabilities, research solutions to improve security flaws, design/implement solutions, deploy changes, test the new measures to ensure they fix previous vulnerabilities and refine implementation(s) if needed. Following this

process will allow us to continuously improve the security state of the system and ensure our designs function properly, in order to keep user data secure.

Process:

- Test the current state of the system
    - Before implementing any changes, test misconfigurations, vulnerabilities and exploits to find weak points in the system.
- Note vulnerabilities
    - Record any security flaws found
        - Found examples: unprotected endpoints, plaintext sensitive data, weak password practices, etc.
- Research solutions
    - Brainstorm/research techniques in order to fix found security vulnerabilities
        - Future improvement examples; bcrypt() for hashing, implementing token, designing middleware, etc.
- Implement solutions
    - Make changes in order to fix found security flaws in respective areas of the project (frontend/backend coding, cloud settings, security configurations, etc.)
- Deploy changes
    - Push solution creations to Git and rebuild the site.
- Test new security measures
    - Test misconfigurates, vulnerabilities and exploits used before in order to see if the problems have been patched.
- Refine implementation
    - If vulnerabilities still persist, refine implementation until security flaws have been fixed.

5.8 User Testing

We intend on asking our clients to walk through the system and perform the tasks they intend to use in the future. As we make substantial improvements, we will present the model to our clients and ask them to demo use the system as they intend to in production. Not only will we be able to obtain feedback regarding the design of our changes, but we will also have our target user, put the system to use and ensure it functions correctly.

Periodically presenting our system and allowing our clients to use it, will ensure we are always testing the tool. Although each team member will be testing changes made every time they contribute to the project, having our clients test the product will bring a new perspective to testing. Although it may technically work/function correctly through our eyes, the clients may have had a different functionality result in mind. They will also be testing system features in an order that will mimic functionality after deployment. This can lead to possible new vulnerabilities or system flaws discovered. Having them walk through the system, will allow us to observe and make changes based on their feedback. Having different perspectives test the system can also lead to new design flaws discoveries. The more we test the system before deployment, the more fixes we can make before the system is used in production; reducing future discovery of problems.

5.9 Results

Backend:

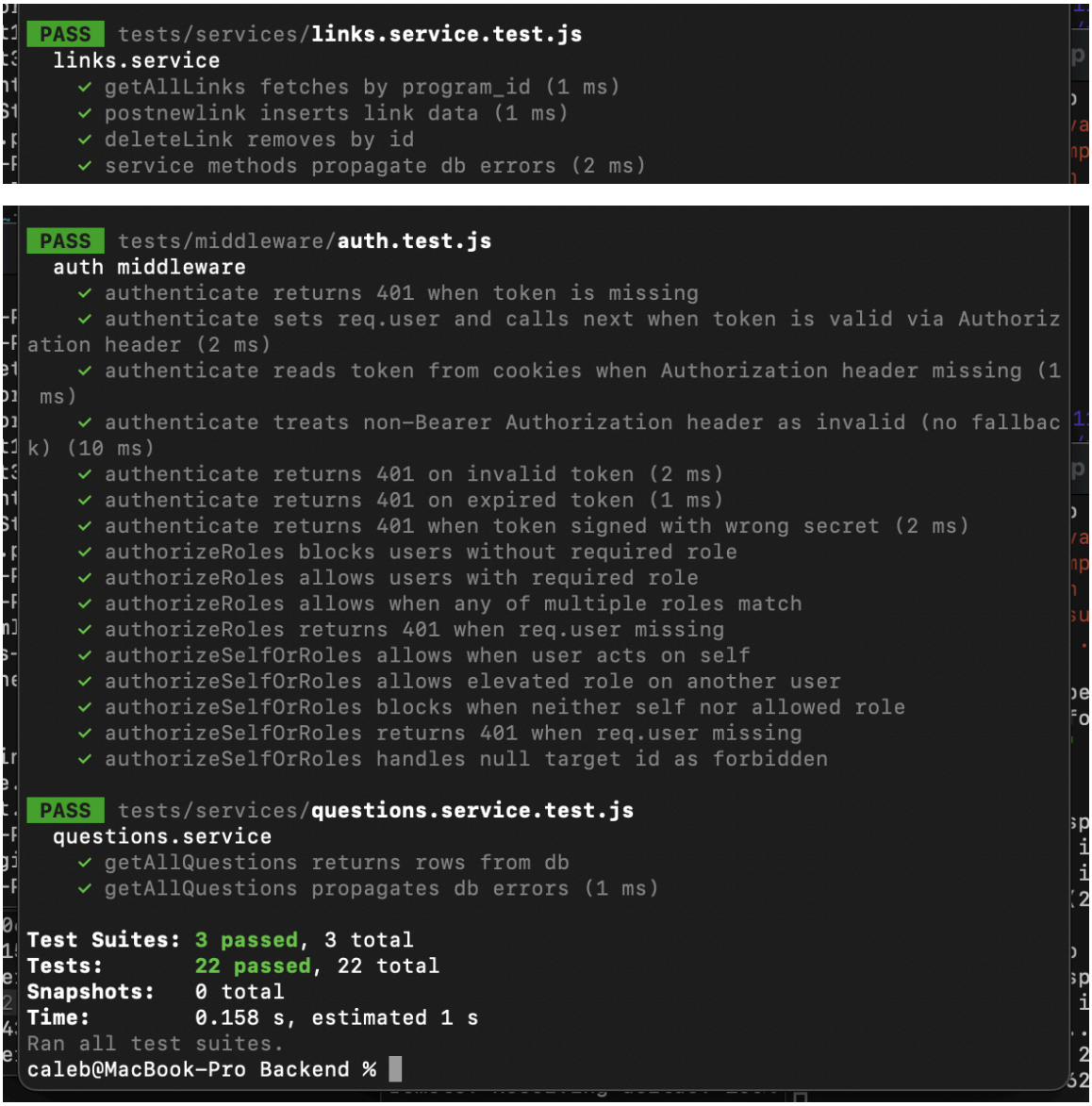

Figure 13: Backend Jest Tests

- Results: npm test -- --runInBand --verbose passes 22/22 Jest tests (middleware + services). DB calls are mocked; only in-memory JWT and query assertions run.
- What we validated: auth middleware enforces missing/invalid/expired token paths, role checks, self-or-role guard; services issue expected SQL/params and propagate errors.
- How it meets needs: Covers core security paths and DB interaction contracts without touching the real database, giving confidence in auth gating and service query shapes.
- Next steps: Optionally add router-level supertest specs to confirm status codes/JSON per route, and silence JWT console noise with a stub if desired.

Frontend:

Figure 14: Frontend Jest Tests

- Results: npm test -- --runInBand --verbose passes 23/23 Jest + RTL tests (middleware, auth context/provider, utils, ProgressBar). Warnings in console are from mocked auth hydration (expected) and don't affect pass/fail; no network or real storage used.
- What we validated: Next middleware role-based redirects for admin/PC/student and cookie clearing; auth token decoding edge cases; AuthProvider home-path logic, logout and localStorage usage; apiFetch credentials defaults; ProgressBar rendering math.

- How it meets needs: Confirms client-side access control logic and auth state handling behave as designed, and shared utilities/components render correctly.
- Next steps: Silence hydration warnings by stubbing fetch/json/wrapping effects in act, add integration tests for key pages/routes if desired.

6 Implementation

The MySTEMGrowth Survey Tool has entered the early implementation phase, with several critical components already developed or actively being integrated. The backend implementation includes restructuring authentication and authorization logic to support secure role-based access for students, program coordinators, and administrators. Password handling has been updated to use hashing techniques, and middleware has been introduced to enforce permissions on protected API routes. Initial work on token- and cookie-based session management has been completed to replace insecure client-side storage and strengthen overall system security.

On the infrastructure side, the team has begun migrating cloud services from AWS to DigitalOcean. This includes provisioning managed databases, configuring encrypted environment variables, and updating deployment configurations to support the new hosting environment. The CI/CD pipeline is being reworked to ensure automated builds, tests, and deployments function correctly within the new infrastructure. While the transition has required significant setup and validation effort, it creates a simpler, more maintainable, and cost-effective foundation for future development.

Preliminary frontend implementation focuses on improving usability and responsiveness. Updated layouts for mobile devices and redesigned program coordinator views have been prototyped and demonstrated to stakeholders to validate navigation and information hierarchy. These UI changes are being implemented incrementally to ensure compatibility with existing backend logic. Integration testing has confirmed that core features like, authentication, survey access, and results viewing, still function as new features are introduced.

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	**Definiton**	**IEEE Code of Ethics**	**Our Team's Interaction**
Work Competence	To produce high quality work using appropriate knowledge	"To maintain and improve our technical competence"	We test our functions and work to validate questions and result score logic.

Financial Responsibility	Use project resources effectively	“To avoid wasteful practices”	We changed hosting provider to reduce cost to the fund.
Communication Honesty	Be transparent and honest in all project communication	“To be honest and realistic in stating claims or estimates”	We be honest in our client-team meetings, even if it isn’t what our client wants to hear
Health, Safety, and Well-being	Ensure the tool promotes positive outcomes	“To hold paramount the safety, health, and welfare of the public”	We make sure our information and results encourages positive behavior
Property Ownership	Respect data ownership and rights of others	“To avoid injuring others, their property, reputation, or employment”	We ensure that the data students enter is protected, and that only people allowed to see it can see it
Sustainability	Solutions that use resources efficiently and consider long term effects on energy and maintenance	“To improve the understanding of technology; its appropriate application, and potential consequences”	We ensure our code is up to current standards with detailed documentation, as well as using minimal resources to fit the demand of the product
Social Responsibility	Designing the tool to benefit society	“To treat all persons fairly and with respect”	We prioritize inclusive language, and equitable use for students of all backgrounds

Our team is doing good in Communication Honesty. We have done a good job at being honest with our client, even if that isn’t how we wanted it to happen. We have explained how much work certain feature additions/rewrites would be so our client can make the decision that is the best use of everyones time.

We have work to do on Property Ownership. We have already improved some by removing the ability to call the API for the list of users. We have to still create a better method of exporting data for researchers that strips all identifying information. This protects both the students and the researchers.

7.2 Four Principles

Area	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	Provide support and opportunities for growth in all of the language in the tool. Work to promote self-confidence and interest in learning.	Avoid harmful and discouraging language that could make students less willing to learn and grow.	Make sure that students control both what information they include, and how they use the information they are given from the tool.	Work to ensure all students will receive the same level of support from tool, regardless of any circumstances.
Global, cultural, and social	Offer a tool that can be used by anyone to help learn and grow in their STEM and social proficiencies.	Stay away from any culturally biased or exclusionary language throughout the tool, allowing as many people as possible to access it.	Allow programs to use this tool as they desire, leaving features that promote individuality between programs	Ensure equity by being inclusive and accessible in the tool. Make sure everyone has access to this tool.
Environmental	Use the least amount of computational resources required for a good user experience.	Avoid unnecessary runtime and data storage. Only run and store the essentials.	Allow people to delete their information or account as they desire.	Ensure environmental impact is not stuck on one program.
Economic	Make sure to use the most cost efficient hosting site to reduce cost to keep the tool running, allowing more students to use it.	Avoid requiring a paid subscription on either a program or student basis. A paid model may create income, but it will reduce the number of students that can take advantage of the tool.	Programs can choose how much they want to use the tool without having to worry about paying for a subscription they won't get enough use out of.	Allowing all programs, regardless of financial state, to use this tool allows for more students to have a better chance at a STEM related job.

Public Health, Welfare and Safety beneficence is important to our project. Our goal is to make a tool that makes learning more about STEM and STEM principles easier to learn. Our tool

helps students learn about themselves, so they can work to improve through programs they are a part of. Providing this information from the survey they take in a simple and effective manner is critical to make the tool useful.

Our tool lacks Environmental nonmaleficence. While we are doing what we can to minimize resource usage, this tool still lives in a datacenter somewhere that is taking power and water to run. Our project can overcome this by helping students learn, which in turn could help students create a more sustainable method to host datacenters.

7.3 Virtues

Team virtues

- Commitment: Showing up prepared and present so decisions get made and work moves forward. Weekly reports with progress and blockers, and sharing notes/action items so attendance translates into progress.
- Responsiveness: Seeking and acting on client feedback quickly to ensure fairness to stakeholders' needs. Take notes, and track follow-ups to close the loop so their voices shape the product.
- Transparency: Communicating clearly across channels (face-to-face, text, email, Discord, reports) and aligning actions with what we say.

How we support them

- Commitment: We set agendas before meetings, showed up prepared, and rotated facilitation. After each session we shared notes/action items and posted weekly reports with progress and blockers.
- Responsiveness: We ran regular client touchpoints, took notes during demos and meetings, and turned their comments into GitLab tasks with owners/due dates. We followed up with brief summaries of what would change and when, so clients knew what to expect and then see their feedback reflected in the project.
- Transparency: We kept work visible in GitLab tasks/issues/PRs, used face-to-face for deeper discussions, chat/Discord for quick clarifications, and email for formal updates. Current work, tasks, plans for the future, and issues were described in weekly reports so everyone stayed up to date and could follow along with primary roles.

Caleb Hemmestad:

- Virtue demonstrated: Ownership
    - Why it's important: It keeps work moving and makes me accountable for delivering quality without waiting for prompts.
    - How I've demonstrated it: I took responsibility for the auth and middleware work, implemented and tightened cookie/token handling and role-based middleware, adding the related tests, and following through to verify everything passed.

- Virtue to strengthen: Tact
    - Why it's important: Delivering feedback and decisions with tact keeps collaboration smooth and helps ideas land without friction.
    - How I can demonstrate it: Be more deliberate in how I frame critiques and requests, offer options, and invite input before deciding, especially when suggesting changes to others' work.

Ethan Buenting:

- Virtue demonstrated: Communication
    - Why it's important: When working on a project in groups, especially over a long period of time, it is important that information is relayed to the team consistently to ensure all members have the same understanding of goals and progress.
    - How I've demonstrated it: I've coordinated with my team on a consistent basis to both relay progress within my role as well as ensure a uniform understanding of plans for future development

- Virtue to strengthen: Planning
    - Why it's important: Planning is important when it comes to accomplishing goals, both as a team and individually.
    - How I can demonstrate it: I can improve on individually delegating time for different class projects more efficiently, as well as working more closely with the team to define a schedule of deadlines for this application.

Ethan Van Caster:

- Virtue demonstrated: Integrity
    - Why it's important: Being honest and realistic in progress is important because uncertainty and struggles in progress are important to ensure a reliable product.
    - How I've demonstrated it: Being truthful in weekly reports on what progress has been made and where more work needs to be done, as well as where I might need others help.

- Virtue to strengthen: Consistency
    - Why it's important: In long term projects, steady predictable progress helps teammates plan around my contributions. It also allows for better more accurate planning of future work.
    - How I can demonstrate it: Create and follow a schedule with specific deadlines and log progress daily.

Nina Gadelha:

- Virtue demonstrated: Humility
    - Why it's important: It is important to openly receive feedback/critiques to ideas/designs, in order to produce an overall better final product.
    - How I've demonstrated it: I compiled a document consisting of changes (new features, functionality modifications, UI improvements), and presented it to our clients. I have been modifying that document to match the feedback from the

clients. This document is intended to be a checklist for site innovations, ensuring we make the expected changes to create a desirable final product.

- Virtue to strengthen: Collaboration
    - Why it's important: Collaboration is important in order to create an effective team environment. It is also useful when working on a feature that has multiple implementations, especially ones that you aren't familiar with. Working with a team member can help with speeding up the process, error correcting and building team chemistry.
    - How I can demonstrate it: I can demonstrate this by working with other team members on feature implementation. For example, I can ask a teammate who is familiar with backend functionality, to implement a feature that I have already functioning on fronted.

Ryan Mamrot:

- Virtue demonstrated: Reliability
    - Why it's important: Reliability is essential because the team needs to trust that work will get done consistently and done well. When deadlines are tight and tasks depend on one another, being someone they can count on keeps progress steady.
    - How I've demonstrated it: I've taken on tasks and followed through without needing reminders, whether it was implementing features, troubleshooting issues, etcetera, I made sure to communicate status updates, ask questions, and deliver my tasks on time.

- Virtue to strengthen: Communication
    - Why it's important: Keeping everyone aligned is what ensures the system comes together cleanly. Taking initiative in communication reduces assumptions, prevents duplicated effort, and can help catch design mistakes.
    - How I can demonstrate it: I can be more proactive about checking in with teammates before starting a new feature or making changes. I can communicate design intentions, ask for input, and provide feedback. Taking these steps will ensure the team is aligned without miscommunication.

Sam Craft:

- Virtue demonstrated: Dilligance
    - Why it's important: If you have something that needs to be done, you need to do it. And if you're doing it, you need to focus on it with the time you are giving to the project.
    - How I've demonstrated it: When I've been given a task I needed to complete, I made sure to start working on it immediately. I also made sure to ask questions as I went to be able to continue working if I ran into issues.

- Virtue to strengthen: Collaboration
    - Why it's important: In a team project, working together with your team is the most important thing you can do. It is essential to being an effective team to work together.

- How I can demonstrate it: I can be proactive with reaching out to my teammates on larger issues in the project. If I am working with something that is outside my expertise on the project, I will ask questions rather than try and learn it all myself.

8 Closing Material

8.1 Conclusion

During our semester in Senior Design I (CPRE 4910), our team has gained an understanding of the MySTEMGrowth Survey tool (functionality and purpose), heavily discussed improvements with our clients, compiled feedback to design the end product and began implementing innovative solutions.

Our goal is to deliver a completely functional version of the website that meets our client's expectations, is fully functional for all users and is a design that we are genuinely proud of. In order to achieve these, we need to continue having open communication with our clients, distribute the workload evenly, implement the desired changes, frequently test compatibility and continue demonstrating excellent teamwork/collaboration skills.

One of the largest constraints we had this semester was our transition of Cloud providers. Not only was gaining access, setting up and transfer of data very time consuming, but it was also challenging to transition the infrastructure without breaking functionality of the tool. This was the main reason we weren't able to start implementing earlier, however, it did give us ample time to discuss and plan our future design with our clients. This extra time was used to clearly communicate expectations and distribute roles/features throughout our team.

Next semester in Senior Design II (CPRE 4920), we intend to regularly be implementing changes to the site. This semester, we focused on developing a clear plan that would meet the client's needs/expectations. Moving forward, we intend to slowly but surely modify the state of the site, in order to meet the client's vision of the tool. In order to achieve this goal, we must keep open communication with our clients, demonstrate effective teamwork, hold ourselves accountable and frequently test functionality of new and existing features/functionality.

8.2 References

[1] S. Crawford, E. R. McCabe, and L. A. Smith, "Applying web-based survey design standards," *Journal of Prevention & Intervention in the Community*, vol. 44, no. 3, pp. 181–194, 2016.

[2] D. A. Dillman, J. D. Smyth, and L. M. Christian, *Mail and Internet Surveys: The Tailored Design Method*, 4th ed. Hoboken, NJ, USA: Wiley, 2014.

[3] IEEE Computer Society, "Challenges in developing research-based web design guidelines," *IEEE Xplore Digital Library*, 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7304204.

[4] IEEE Computer Society, *IEEE Standard for Software Life Cycle Processes*, IEEE Std 12207-2017, 2017.

[5] IEEE Computer Society, *IEEE Standard for Software Configuration Management*, IEEE Std 828-2012, 2012.

[6] IEEE Computer Society, *IEEE Standard for System, Software, and Hardware Verification and Validation*, IEEE Std 1012-2016, 2016.

[7] OWASP Foundation, "OWASP Top 10 Web Application Security Risks," 2023. [Online]. Available: https://owasp.org/www-project-top-ten/

[8] DigitalOcean, "DigitalOcean App Platform Documentation," 2024. [Online]. Available: https://docs.digitalocean.com/products/app-platform/

8.3 Appendices

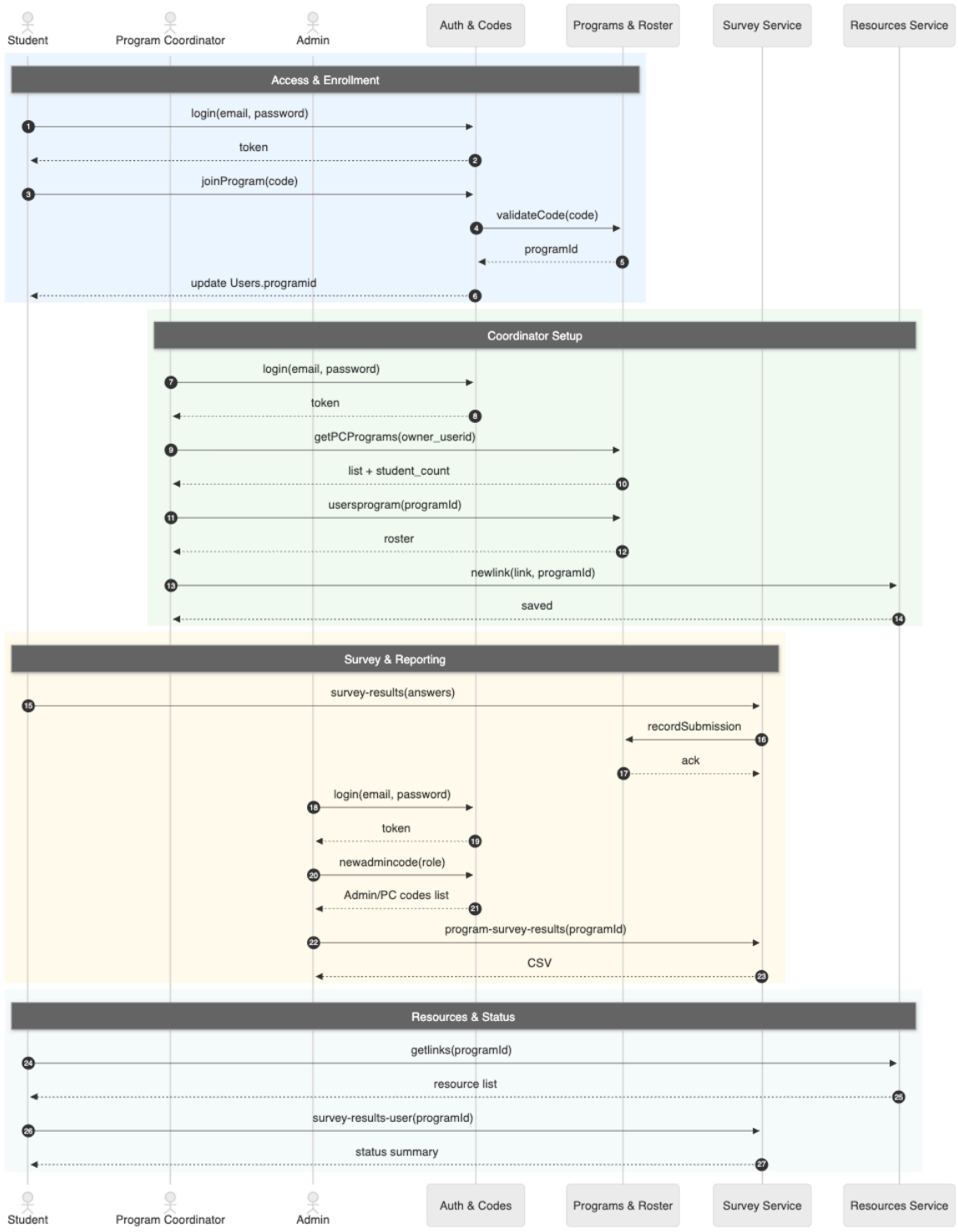

Figure 15: User Interaction Timeline

9 Team

9.1 Team Members

Caleb Hemmestad: Backend/Cloud

Ethan Buenting: Backend/Cloud

Ethan Van Caster: Frontend Developer

Nina Gadelha: Cybersecurity Developer

Ryan Mamrot: Frontend Developer

Sam Craft: Backend Developer

9.2 Required Skill Sets for Your Project

- Frontend: React/Next.js (TypeScript), Java, Chakra UI/Tailwind, accessibility basics, Jest/RTL testing, GitHub workflows.
- Backend: Node.js/Express, REST API design, MySQL/SQL, Jest/Supertest testing, GitHub workflows.
- Cloud/DevOps: CI/CD and releases, Docker basics, environment/secrets management, deployment to hosting platforms.
- Security: Password hashing (bcrypt), JWT/token auth, secure cookies (httpOnly/secure/sameSite), input validation/OWASP awareness, CSRF/HTTPS hardening, dependency patching/CVE response, vulnerability testing.

9.3 Skill Sets covered by the Team

Caleb Hemmestad: JS/TS, Next.js, React, Express/Node, REST API, JWT, Cookies, Middleware, MySQL/SQL, Jest, Docker, PlantUML/Mermaid, Documentation, Prototyping, GitLab

Ethan Buenting: Next.js, Node.js, JavaScript, Java, SQL, AWS, GitLab, Digital Ocean

Ethan Van Caster: React, Next.js, Java, JS, Chakra UI, HTML, CSS, SQL, environment management, GitLab

Nina Gadelha: Object-oriented programming, password hashing, vulnerability testing, React, Java, Typescript, HTML/CSS, github

Ryan Mamrot: Web/app development, frontend/backend development, database, Data Analysis, and virtual machine experience

Sam Craft: Object-oriented programming, Node.js, Express, SQL/MySQL, Gitlab, Docker

9.4 Project Management Style Adopted by the team

Our team has decided to implement the Agile development process during Senior Design I (CPRE 4910). Throughout this semester in Senior Design I, we believe we have discovered many adjustments/innovations our team is able to make to the website, in order to satisfy our clients and future users. Working on features in a 'cyclical' process ensures our improvements are continuously tested. We want to ensure all changes we make benefit the overall state of the website, work as expected for all types of users and do not compromise any other feature functionality. Adopting the Agile process allows us to frequently check our features, while integrating innovations frequently.

9.5 Initial Project Management Roles

Cloud transition + backend management: Ethan Buenting and Caleb Hemmestad

Mobile screen configurations + team contact: Ethan Van Caster

Security implementations + misc. features: Nina Gadelha and Caleb Hemmestad

Frontend + survey development: Ryan Mamrot

Backend development + testing: Sam Craft

9.6 Team Contract

Team Members:

1) Nina Gadelha
2) Caleb Hemmestad
3) Ethan Buenting
4) Ethan Van Caster
5) Ryan Mamrot
6) Sam Craft

Team Procedures

1. Team meetings will be held primarily face-to-face in whatever rooms we can reserve in the SIC or the TLA. Occasionally due to schedule conflicts we will hold impromptu meetings virtually or communicate over our Discord channel. The meeting times we set aside were:
    a. Monday 4-5 pm
    b. Tuesday 4:30 - 5:30 pm
    c. Thursday 2 - 3 pm
2. Texting, Discord, Email, Face-to-face meetings, and lecture are all valid methods of communication for the group
3. We will use majority vote as our decision making policy
4. We will create a Google Drive folder for all shared documents, such as class documents. There will be a minutes folder created for all meetings in case a member of the group is unable to participate. We will also have a shared Git repository for all relevant code

Participation Expectations

1. Attendance, punctuality, and participation are expected in all meetings. If someone must miss a meeting, at least an hour notice is expected. From there, the team can either reschedule the meeting or continue as scheduled. The person who missed the meeting should reach out and get the information they missed.

2. All team members are expected to contribute to all assignments and portions of the projects that are relevant to their major. While certain assignments or tasks may be taken on by individuals who feel it is within their expertise, it is expected that overall this project is an equal group effort.

3. Regular communication is expected regarding project progress/problems, meeting attendance and communication from advisors/clients.

4. Each team member has a say when making decisions. Depending on the role/experience/expertise of a certain member, their input may weigh more (dependent). On average, every member will get an equal say.

Leadership

1. The roles of the group are dedicated as follows:

    a. Caleb: Cybersecurity Development

    b. Nina: Cybersecurity Development

    c. Sam: Testing

    d. Ethan: Team Organizer

    e. VC: Client Contact

    f. Ryan: Developing

2. We will use Gitlab's Tasks function in order to monitor progress and member contributions as well as delegate work. During team meetings we can work on adding Tasks for features we want to begin implementing, as well as helping us to plan for the future.

3. Members will receive the praise of the team for their contributions, as well as the satisfaction of a successful project.

Collaboration and Inclusion

1. Individual skills and expertise that individual members bring to the team:

    a. Caleb: Full-Stack Website Development with a Cybersecurity mindset and Cloudflare/AWS experience.

    b. Nina: Frontend App development, VM experience, some web development, linux proficient

c. Sam: Web Development, Frontend/Backend, Database, DigitalOcean, AWS

d. Ethan: Frontend/Backend, MySQL and PostgreSQL, Render

e. VC: Previous web development, databases, data analysis

f. Ryan: Web/App development, Frontend/Backend, database, Data Analysis, and some VM experience

2. Individuals are encouraged to share their ideas and thoughts on project features or potential implementations at team meetings, where the team can vote on them and/or make changes/improvements to the proposition that the group agrees on with the majority.

3. If any member feels that the team environment or an individual is obstructing their ability or opportunity to contribute, we would expect that they bring the conflict up at a team meeting so it can be resolved swiftly and amicably. If the individual does not feel comfortable presenting the issue in a group discussion, they can bring it to the attention of the group coordinator or an instructor of the course.

Goal-Setting, Planning, and Execution

1. Our team goals for this semester are to create a ready-to-launch deliverable, have fun, learn new skills, create new friendships, and build new connections amongst our team.

2. We plan to divide work among members heavily based on their skill sets to allow them to build within their respective areas, as well as work as a team to create a clear and decisive list of tasks and goals to help us work concisely as a team.

3. To ensure an efficient execution, we plan on keeping the team mutually accountable between group members both within team meetings and weekly progress reports.

Consequences for Not Adhering to Team Contract

1. We will handle infractions of any obligations by individual members within team meetings by having a group conversation with that member to determine reasoning and developing an actionable plan to fix the issue.

2. If infractions continue past the previously mentioned expectations and attempted solutions, we will contact the course instructor(s), Dr. Fila and Dr. Rover, to help with an outside opinion to provide us with a potential solution or course of action.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Nina Gadelha ` DATE 9/4/2025

2) Caleb Hemmestad DATE 9/5/2025

3) Sam Craft DATE 9/8/2025

4) Ethan Van Caster DATE 9/8/2025

5) Ethan Buenting DATE 9/8/2025

6) Ryan Mamrot DATE 9/9/2025