IINSPIRE STEM Survey Visualization Tool

DESIGN DOCUMENT TEAM 24

Client

Dr. Diane Rover/University of Iowa

Advisors

Dr. Diane Rover

Team Members & Roles

Alex Moeller - Project Manager/Frontend
Nick Pinnello - Backend
Charlie Moreland - Backend
Max Strater - Frontend
Isabelle Raghavan - Frontend
Matthew Bennett - Cloud
Landon Gulotta - Cloud/Security

Team Email: sdmay25-24@iastate.edu

Team Website: https://sdmay25-24.sd.ece.iastate.edu/#

Executive Summary

The MySTEMGrowth Survey Visualization Tool is a web-based application designed to improve user engagement and enhance access to survey data. The platform allows student participants to complete surveys and visualize their individual results, while program coordinators can manage programs, invite students, and customize program resources. Administrative users have access to comprehensive data analysis and can extract statistical insights for research purposes. This project builds upon the work done by a prior Senior Design team, with this year's efforts focused on modernizing the user interface (UI), improving the user experience (UX), and enhancing overall functionality to support scalability and long-term sustainability.

Problem Statement and Importance

While tools like Qualtrics are primarily designed for collecting and aggregating survey responses, the MySTEMGrowth Survey Visualization Tool goes beyond simple data collection by delivering a user-centered experience that drives personal development and program improvement. Unlike standard survey platforms, MySTEMGrowth empowers students to visualize their own STEM capabilities, track their progress over time, and set personalized goals for growth.

For program coordinators, the platform serves as a comprehensive hub to manage all aspects of the program, including inviting students, assigning and managing surveys, and customizing program resources. Administrative users gain advanced capabilities to download, analyze, and interpret aggregated survey data for research and statistical analysis. Additionally, the tool supports user role management, enabling admins to promote users into higher-level roles such as program coordinators or fellow admins.

Key Design Requirements

Our project prioritizes usability, scalability, and maintainability. Our application should provide valuable survey experiences for students and dashboards/overviews for admin users. Additionally, the architecture should support future changes with minimal effort.

The existing MySTEMGrowth Survey Tool lacked a modern UI/UX, limiting its usability and appeal to target audiences. Additionally, its non-cohesive architecture posed challenges to maintainability and scalability, limiting its usefulness for future Senior Design groups. Addressing these issues is critical to enhancing user satisfaction and maintaining the program's value as a research and engagement tool.

Design Summary

We have adopted a modern technology stack to meet these goals:

- **Frontend:** React with the Chakra UI component library, TypeScript, Next JS
- **Backend:** MySQL
- **Cloud Infrastructure:** AWS ECS, Secrets Manager, and Amazon RDS

Our design strategy emphasizes modularity and reusability, ensuring ease-of-use for both current and future developers.

Learning Summary

Development Standards & Practices Used

IEEE Standards

● IEEE 26515-2018 - Agile Development Cycle

○ We plan to use this style of development for our project to ensure efficient development that allows for continual communication between the team and client in regards to the deliverables.

● IEEE 829 - Software Test Documentation

○ Following this standard for test documentation allows our team to easily document our tests to show what each test does and its expected results.

● IEEE Computer Society Code of Ethics:

○ We work professionally and ethically to further the advancement of software engineering.

Summary of Requirements

- A web-based tool that client users can access
- Create charts based on survey data
- Generate explanations for the created charts
- Provide the user with options to save or print the created charts
- Store statistical survey results in a database to be accessed by the web tool
- A web-based tool that client users can access

Applicable Courses from Iowa State University Curriculum

COM S 309, COM S 319, COM S 327, COM S 363, COM S 409, SE 317, SE 329, SE 339, SE 417, SE 421

New Skills/Knowledge acquired that was not taught in courses

- TypeScript programming language
- Front-End Framework such as React
- Data Visualization Library (AnyChart)
- Data Processing and Analysis tools
- Data Security
- UX design practices
- Amazon Web Services: ECS and RDS
- CI/CD Testing
- Infrastructure as Code (IAC)
- 3rd party Authentication

Table of Contents

1.Introduction

1.1. Problem Statement

The **MySTEMGrowth** program supports underrepresented minority students by encouraging their engagement in STEM fields. To better understand and improve the program's effectiveness, researchers need a modern tool to administer surveys and efficiently visualize the results. The current survey platform is outdated in both design and functionality, making it difficult to use and time-consuming to manage.

Our project addresses this problem by redesigning the frontend and upgrading the backend infrastructure to improve performance, usability, and responsiveness. Additionally, the restructuring of Cloud resources through AWS will provide a seamless transition for future project changes. This will streamline the survey-taking experience for students and simplify survey creation and management for researchers/developers. Ongoing user feedback will guide development and the addition of new features.

MySTEMGrowth participants can now take surveys, view results, and even download their data on a much more responsive web tool. By making data collection and interpretation more efficient, the tool will help demonstrate the program’s positive impact, such as increased student confidence in STEM subjects.

1.2. Intended Users

Admin:
Global Administrators manage the system and need access to high-level system functions.

MVP Capabilities

- Generate join codes for new Admins or Program Coordinators (PCs)
- View current Admins/Program Coordinators
- View all programs and download specific program data (.csv file)
- Delete programs

Administrators must access all participant responses, as this data is essential for their evaluations and research. Their interface should allow for easy, simple access to all needed data and functions, allowing them to manage users, surveys, and the system without unnecessary complexity.

Program Coordinator (PC):
Program Coordinators serve in the middle-privilege role. They act as the leader of a particular program instance (ex. Iowa State). They are in charge of all activities related to any program instances they may have created, but cannot view information about other program instances outside of their creation.
This role is likely the most time-intensive and has the most daily responsibility.

MVP Capabilities

- Create an account with the PC role
- Create a new Program Instance
    - Creates a unique join code for inviting Students
- Select a certain Program Instance to view
    - View information of a particular Program Instance via a drop-down menu
    - View the number of students attached to the selected Program Instance

- View a table of students (name, email, join date) attached to the selected Program Instance
- View, add, and delete program resources

Participants (Students):
Participants are students, often from underrepresented backgrounds in STEM, who provide key feedback through surveys.

They will be the primary survey-taking and result-viewing user.

Their interactions with the system include:

- Create an account
- Sign in/out of the account
- Take survey
- View personal survey results
- Download survey results to PDF
- Compare survey results
- View program-specific resources (assigned by their PC)

Students need an intuitive, easily accessible web app to access complete surveys from any device. The system should be accessible and allow for surveys to be easy and quick to take. They must be able to view assigned surveys, track progress, and review personal results.

NTH (Future) Items:

- Change/edit the join code for a Program Instance
- Allow multiple PC's for a single Program Instance
- View an average results graph for all students in the selected program
- Close a survey and set a new survey to be the default survey shown to students
- ⭐Allow a PC to move/transfer students from Program Instance to a new Program Instance without requiring a new account. This will allow for surveys taken under a different Program Instance to be viewed by the Student regardless of what Program Instance they are currently a part of.

Researcher:
Researchers will use this data to assess the impact of the STEM program on participants, identifying areas of improvement to adapt the program for future students. They will rely on the survey data to shape the program and provide insights into its effectiveness.

NOTE: THIS ROLE IS OUT OF SCOPE FOR 2024 - 2025.

Future Team MVP Ideas

- Create an account
- Store/View survey results
- Share/Download Data
- Methods to visualize survey results

Like admins, Researchers use the survey data to evaluate the program's performance and create different surveys to give to STEM students. Their role focuses more on operational tasks, such as creating and maintaining surveys and organizing results. The App should make it easy to manage and share the data, allowing for easy collaboration with researchers.

2. Requirements, Constraints, And Standards

2.1 Requirements, Constraints, and Standards

Functional Requirements

- Survey Participants must be able to:
    - View their results immediately after completing a survey.
    - Access and compare results from previous surveys.

- All users (Participants, Program Coordinators, Admins) must be able to:
    - Create accounts and log in using email or Single Sign-On (SSO).

- Admins must be able to:
    - Create, edit, and manage surveys.
    - Invite or remove users.
    - Send out surveys to participants.
    - Monitor and access all survey results.
    - Grant specific permissions to users, including Participants and Program Coordinators.
- Program Coordinators (PCs) must be able to:
    - Create and manage their program instances.
    - Invite students via join codes.
    - Access and manage resources and data related to their specific programs only.

- Researchers (for future implementation):
    - View and download survey results.
    - Request new survey questions or changes to existing ones.
    - Assign students to specific surveys.

UI/UX Requirements

- Separate, role-specific dashboards for:
    - Admins
    - Program Coordinators
    - Participants

- Essential UI features include:
    - Login and registration pages.
    - A consistent color theme and custom logo for the homepage.
    - A navigation bar with links to Home, Survey (with role-specific content), and About.
    - A redesigned survey interface that includes:
        - Large, readable text and buttons.
          A progress bar to show completion status.
        - A "Save" button to allow participants to save progress and return later.
        - A "Previous" button to navigate and edit previous answers.

Technical & Resource Requirements

- Frontend: React.js, Next.js, Chakra UI
- Backend: Node.js, Express.js, NextAuth.js, MySQL, PGAdmin, Jest
- Cloud/DevOps:

    - AWS ECS for containerized deployment
    - AWS Secrets Manager for managing sensitive configurations
    - AWS S3 for file storage (e.g., downloadable PDFs)
      AWS RDS for relational database services
    - AWS CloudWatch for logging and monitoring
      GitHub and GitHub Actions for source control and CI/CD

2.2 Engineering Standards

IEEE 26515-2018 – Agile Development Cycle

This standard emphasizes flexibility in the development process through Agile methodologies. It promotes rapid adaptation to changing requirements via ongoing stakeholder feedback and collaborative, iterative development. The goal is to allow for continuous improvement rather than waiting until the end of the cycle to make changes, particularly useful for evolving user needs.

Relevance to Our Project

Our project places a strong focus on user experience, making this standard highly applicable. We will implement continuous Agile cycles to refine our design based on frequent user feedback. Regular interviews with stakeholders will guide these iterations, keeping the development responsive and collaborative.

To align with this standard, we will:

- Use **Kanban boards** to manage and track tasks.
- Hold **weekly team meetings** to review progress and plan future work.
- Allocate time during meetings for **peer and advisor feedback**.
  This process ensures the project remains flexible and transparent throughout the development lifecycle.

IEEE 829-1998 – Software Test Documentation

This standard outlines how to systematically document software testing to ensure thorough coverage and traceability. It includes the creation of test plans, test cases, expected outcomes, and results, helping teams identify gaps and maintain quality throughout the development process.

Relevance to Our Project
Our application contains many modular components that must be tested individually and as part of the full system. By following IEEE 829-1998:

- We will develop a **comprehensive testing framework**.
- Documentation will include **test plans, test cases, and test logs** for each feature.
- We will incorporate **automated testing tools** to verify consistency and behavior. All testing documentation will be reviewed and discussed during weekly team meetings to ensure alignment and accountability across the team.

IEEE 1448a-1996 – Standard for Software Life Cycle Processes

This standard, aligned with ISO/IEC 12207, provides a structured framework for managing the software life cycle from acquisition through retirement. It defines common terminology and best practices for development, operation, maintenance, and reuse.

Relevance to Our Project
We will use this standard to guide our planning and documentation processes, ensuring consistency and quality across the project. Its adaptability to business practices supports our iterative and modular development strategy.

IEEE 1621-2006 – Standard for User Interface Design and Management

This standard offers best practices for designing user interfaces that are consistent, accessible, and easy to use. It emphasizes usability and intuitive interaction patterns.

Relevance to Our Project
One of our core goals is to **redesign the current web application UI** to improve the overall user experience. By adhering to IEEE 1621-2006:

- We will ensure the interface is **clear, accessible, and consistent** for all user roles.
- The system will be designed for **ease of navigation and readability**, particularly for participants with diverse levels of digital literacy.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our team uses a hybrid (waterfall+agile) approach for project management.

Approach:

Waterfall for High-Level Planning: Each team (Cloud, Backend, Frontend) has a set list of base requirements and deadlines that are defined upfront.

Agile for Execution: Teams will break down the larger requirements into smaller, manageable tasks and work in agile cycles (sprints). Agile allows for flexibility in addressing unforeseen challenges and adapting to changes in requirements during development. Frequent iterations help to prioritize critical features and continuously deliver incremental value.

Coordination and Accountability: One team member will oversee progress, ensure deadlines are met, and assist individual teams with managing their stories. This role bridges the structured planning of Waterfall with the iterative adaptability of Agile, ensuring smooth coordination and delivery.

Justification:

- Clear Deadlines: The Waterfall approach ensures that key deadlines are established and adhered to, supporting project-wide consistency.
- Flexibility: Agile cycles allow teams to adapt to changing requirements without derailing overall project timelines.
- Collaboration and Ownership: Individual teams retain autonomy over their agile cycles, fostering ownership and accountability within their scope.
- Risk Mitigation: Frequent reviews and iterative progress help identify and address risks early, reducing the likelihood of significant roadblocks.

Tools:

- Gitlab: Primarily used for managing our repo and merge requests. We also had an issue board to keep track of each story that everyone was working on.
- Discord: Primary communication platform for the team and professor, with dedicated channels for team discussions, stand-ups, and file sharing.
- Git and GitHub: Version control and collaboration via GitHub repositories, pull requests ensure code review before integration.
- Automated AWS Deployment: GitHub Actions will automatically deploy changes to AWS servers and run tests, allowing for a stable live environment

3.2 Task Decomposition

Cloud

- **AWS ECS:**
  - Set up an ECS cluster and define a service for the Node.js server.
  - Configure the ECS task to run the Node.js server.
  - Expose necessary ports on ECS for public access.
  - Integrate ECS with GitHub Actions for automated deployments.
  - Configure auto-scaling and load balancing for high availability.
- **AWS RDS (MySQL):**
  - Provision and configure an RDS instance with MySQL.
  - Set up security groups to allow connections from ECS and specific IPs.
  - Define IAM roles for ECS to securely access RDS via Prisma.
  - Configure and enforce SSL for API encryption.
  - Finalize the connection between RDS and ECS for seamless communication.
- **GitHub/GitHub Actions:**
  - Set up a GitHub repository for CI/CD workflows.
  - Lock down the main branch for stricter code integration policies.
  - Define workflows in GitHub Actions to automate ECS deployment for the Node.js server.
  - Securely define secrets and environment variables for workflows.
  - Configure workflows to build, test, and deploy upon pushes to the main branch.

Frontend

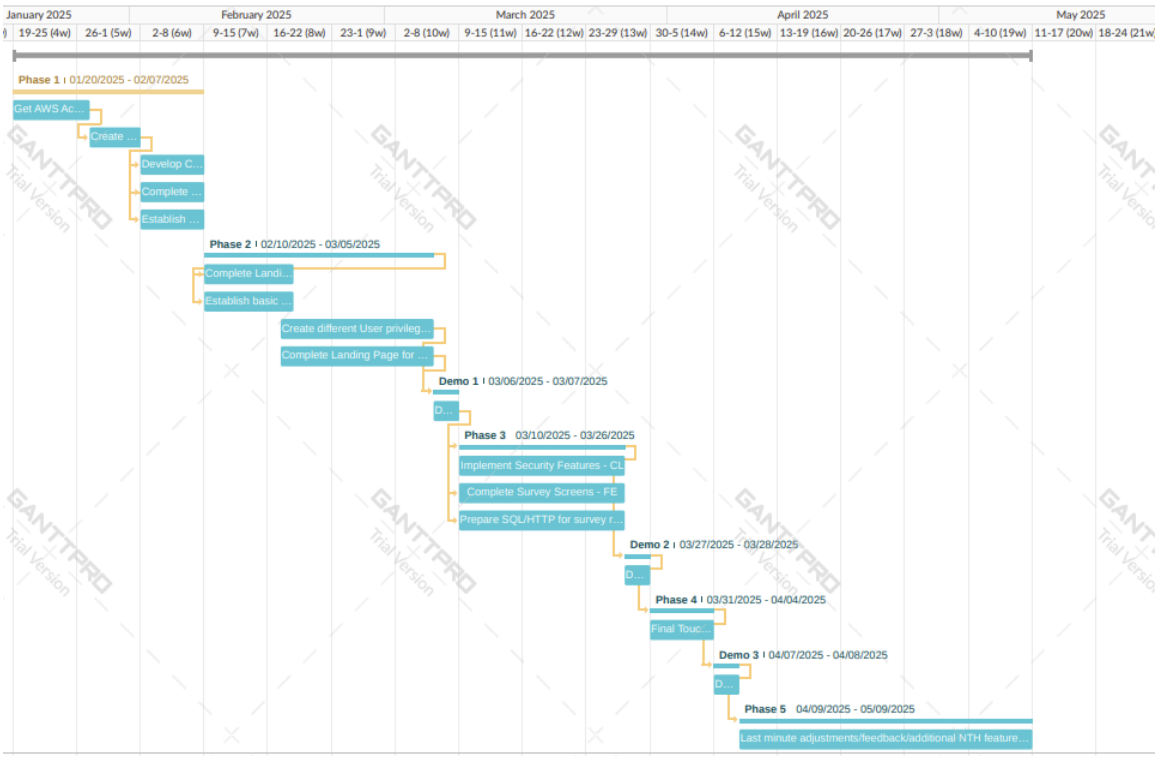
- **Improve user experience**
    - Import the frontend code from the previous team as the skeleton for this website.
    - Refactor the aesthetics of the website:
        - Create a Figma design as a blueprint for the website.
        - Use Chakra UI to implement the design created in Figma.
    - API Calls:
        - Refactor API calls and logic for improved efficiency.
        - Learn and implement Next.js for efficient API calls and server-side rendering.
        - Refactor API calls and logic for Next.js server-side rendering.

Backend

- **Exports for Data Results:** Create detailed downloadable CSV files of Survey Results, such as averages, demographics, and individual question responses. Formatting and accessible data are dependent on User roles, Admin vs Program Coordinator.
- **Test Coverage:**
    - Increase test coverage for backend functions.
    - Evaluate current testing strategies (e.g., automated Postman scripts) and explore Jest for backend testing.
    - Integrate API tests into the CI/CD pipeline for continuous validation.
- **Database Usage:** Use pgAdmin to manage user information and data storage in SQL.

Retain MySQL for consistency with the current architecture to continue to build off of the previous team's schema.

3.4 Project Timeline/Schedule



We have broken down the Spring Semester into 5 distinct phases, accompanied by 3 major demo targets. Each phase will approximately correlate to one month (4 weeks) of the semester, with a major demo occurring after phases 2, 3, and 4. Of course, there will be a final presentation, which could be considered demo 4.

Below, there is an easier-to-read breakdown of all the different tasks and phases, as well as who is responsible for each phase, denoted by CL for Cloud team, FE for Frontend Team, and BE for Backend Team.

Proper agile Software Engineering planning and organization would only plan 1-2 iterations out, which means our project will follow a hybrid agile/waterfall style. Therefore, tasks after Phase 2 are technically not in scope yet and are considered temporary planning items only.

Task Breakdown

1	⊟	**Phase 1**
1.1		Get AWS Account with ECS up and running - CL
1.2		Create shared Github Repo and attach to AWS - CL
1.3		Develop CI/CD Pipelines - BE
1.4		Complete Home Page - FE
1.5		Establish Basic HTTP Requests
	⊕	Add a task | Add a milestone
2	⊟	**Phase 2**
2.1		Complete Landing Page for Student - FE
2.2		Establish basic SQL Queries - BE
2.3		Create different User privileges - CL/BE
2.4		Complete Landing Page for Program Coordinator - FE
	⊕	Add a task | Add a milestone
3	⊟	**Demo 1**
3.1		Demo current project - All
	⊕	Add a task | Add a milestone

4	⊟	**Phase 3**
4.1		Implement Security Features - CL
4.2		Complete Survey Screens - FE
4.3		Prepare SQL/HTTP for survey results - BE
	⊕	Add a task | Add a milestone
5	⊟	**Demo 2**
5.1		Demo Survey Screens - All
	⊕	Add a task | Add a milestone
6	⊟	**Phase 4**
6.1		Final Touches and Testing
	⊕	Add a task | Add a milestone
7	⊟	**Demo 3**
7.1		Demo Final Project
	⊕	Add a task | Add a milestone
8	⊟	**Phase 5**
8.1		Last minute adjustments/feedback/additional NTH features - All

3.5 Risks and Risk Management/Mitigation

1. Scope Creep (0.6)
Risk: Client requirements are not fully finalized, and additional features, such as custom surveys, may be requested mid-project, leading to unplanned work and potential delays.
Mitigation Plan: Prioritize completing the core functionality of the application, focusing first on refining and finalizing the first iteration. Once the foundational features are in place, the team can assess and implement additional requests. After the core infrastructure is established, cloud development team members can be reallocated as needed to support other areas of development.

2. Project-Based Dependencies (0.4)
Risk: Certain components depend on others being completed before work can begin. For example, backend development requires the RDS database to be fully configured before integrating Prisma. Similarly, the database must be connected to the ECS environment to allow testing in a production-like setting.
Mitigation Plan: Maintain a clear dependency map and ensure early setup of foundational infrastructure (e.g., RDS, ECS). Assign tasks in a way that parallel progress can be made where possible, and schedule integration tasks with buffer time to accommodate upstream delays.

3. Timeline Slippage (0.8)
Risk: Due to the size of the team and the complexity of large deliverables, such as a completed Figma design or the foundational Next.js app, it is difficult to assign exact timelines. Development and testing delays may also occur unexpectedly.
Mitigation Plan: Break large goals into smaller, clearly defined tasks and assign them to individual team members with specific due dates. Use project management tools (alongside our project manager) to track progress against defined milestones. Build buffer time into the schedule and maintain a list of optional or stretch tasks for team members who finish early.

4. Unexpected or Out-of-Budget Costs (0.2)
Risk: With ECS's auto-scaling capabilities, operational costs may exceed the budget during high-demand periods, such as during load testing or when surveys receive a surge in responses.
Mitigation Plan: Monitor usage and costs closely through AWS billing dashboards. Set budget alerts and consider using scaled-down instances or scheduled scaling to manage cost spikes during testing phases.

3.6 Personel Effort Requirements

FE = Frontend (Alex, Max, & Isabelle)

BE = Backend (Nick and Charlie)

CL = Cloud (Matthew and Landon)

Task	Time Per Task (Hours)
FE - Create Figma Wireframe Designs	40
FE - Initial Implementation of Student	50
FE - Initial Implementation of PC	35
FE - Initial Implementation of Admin	10
FE - Misc. Bug Fixes/Extra Items	15
BE - API routes for pages	30
BE - Database setup/ updating tables	20
BE - Misc functions for frontend	35
BE - CSV files for survey results	15
CL - Teardown old Cloud infrastructure	10
CL - Setup new VPC & Repo	15
CL - Setup DB	20
CL - Setup ECR and ECS	30
CL - Setup ALB and Route 53	10
CL - Environments/Monitoring	15
CL - CI/CD	25
CL - Infra Planning	45
ALL - Weekly Advisor Meetings	30

3.7 Other Resource Requirements

This project involves building a survey system web application, there are no physical parts or materials that are needed to complete the project. Other than physical tools, there are essential resources that are needed, which include:

- AWS Tools
    - ECS to define the service and run the Node.js server as well as integrate with GitHub actions
    - RDS for configuring and provisioning MySQL Database as well as API encryption and IAM roles
    - SecretsManager for securely storing, managing, and retrieving user data
    - CloudWatch for monitoring and logging
    - ALB to distribute incoming HTTP/HTTPS traffic across the ECS tasks to avoid bottlenecks
    - Target Groups to manage and route traffic to specific ECS services based on health checks and load balancing rules.
    - Security Groups applied to the ECS and ALB to control inbound traffic
    - Route 53 to use a DNS name for the app and ACM for a certificate to use out app on HTTPS
    - Cloudshell to interact with AWS resources using the command line directly in the browser without the need for local configuration
- IDEs/Programs/Libraries
    - IntelliJ, Visual Studio Code, or any other IDE
    - MySQL database to create schemas that will be hosted on our AWS applications
    - Typescript for our programming language
    - Chakra UI for frontend styling
    - Node.js and the runtime environment to connect our frontend and backend logic
    - GitHub repository for CI/CD and to have a set main branch
    - GitHub Actions to set up CI/CD workflows to automate ECS deployment to the Node.js server
    - Postman for testing our API endpoints for frontend and backend communication

- Project Management Tools: Gitlab will be our team's source of project management. This allows us to create epics, stories, issues, milestones, and comments for our team's project development.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	The application collects sensitive user data (age, gender, ethnicity, etc.) and therefore must ensure privacy and security. This helps foster trust and psychological safety for users completing the survey.	Implementing encryption and secure authentication reduces the risk of data breaches; provides academic programs with insights to improve student well-being and performance.
Global, cultural, and social	The survey system must be inclusive of diverse identities and experiences. It should avoid bias in how data is collected and interpreted, particularly for underrepresented groups.	Including ethnicity and gender options beyond binary categories, ensuring UI/UX accessibility for users with different backgrounds or abilities.
Environmental	As a digital solution, the environmental impact is relatively minimal. However, hosting and infrastructure choices (e.g., cloud servers) contribute indirectly to energy use.	Choosing AWS regions with lower carbon footprints or using serverless functions to reduce idle computing resources.
Economic	The tool helps academic programs evaluate and improve curriculum based on real student outcomes, which can lead to better educational investments and funding use. It must also remain cost-effective for institutions to adopt.	Making the system open-source or low-cost for educational institutions, minimizing development costs through efficient planning and use of existing technologies.

4.1.2 Prior Work/Solutions

This project expands upon the initial version developed by an Iowa State University Senior Design team in 2023. That team established the foundation by implementing survey creation, user management, and response storage.

However, the initial version had several limitations:

- The user interface was outdated and lacked intuitive navigation.
- Students received minimal visual feedback after completing surveys.
- The platform was not optimized for scalability or cloud deployment.
- Cloud setup did not match previous technical documentation

Our team addressed these challenges through the following improvements:

- **Modernized UX/UI using** React for a more accessible and engaging student experience.
- **Visual feedback features** to help students reflect on their own data and progress.
- **Cloud deployment on AWS**, supported by **CI/CD pipelines**, to ensure scalability, reliability, and long-term maintainability.

These enhancements enable the platform to better serve its dual audience—students and administrators—while aligning closely with the goals of the LSAMP initiative.

4.1.3 Technical Complexity

Cloud

The infrastructure is built on AWS using a combination of Route 53, Application Load Balancers (ALBs), ECS Fargate, and Aurora MySQL, structured within a custom Virtual Private Cloud (VPC).

- Our new infrastructure separates public-facing components (DNS and ALBs) from our compute and data layers, which are placed on private subnets to minimize exposure and follow least-privilege network design.
- Services are deployed as independent ECS tasks for the frontend and backend, promoting fault isolation and scalability
- All infrastructure is deployed through an automated CI/CD pipeline that builds container images and registers new task definitions using Kaniko, supporting secure, repeatable deployments without requiring elevated privileges.
- Health checks and automatic rollbacks ensure that failed deployments do not disrupt availability, while internal service communication is encrypted and isolated.

This design reflects industry-standard practices for building reliable, secure, and maintainable cloud-native systems.

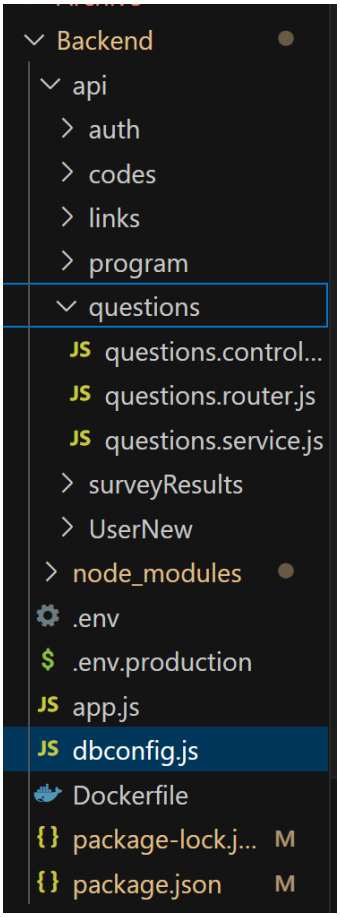
New User signup is protected by randomly generated 6-character codes that are generated by Admin and Program Coordinator Users. These codes validate a User to have access to the survey and assign the user's role without any manual elevation needed.

Survey Result data is formatted and downloadable for Admins by program for Program research, as well as proof of program effectiveness for funding.

Backend

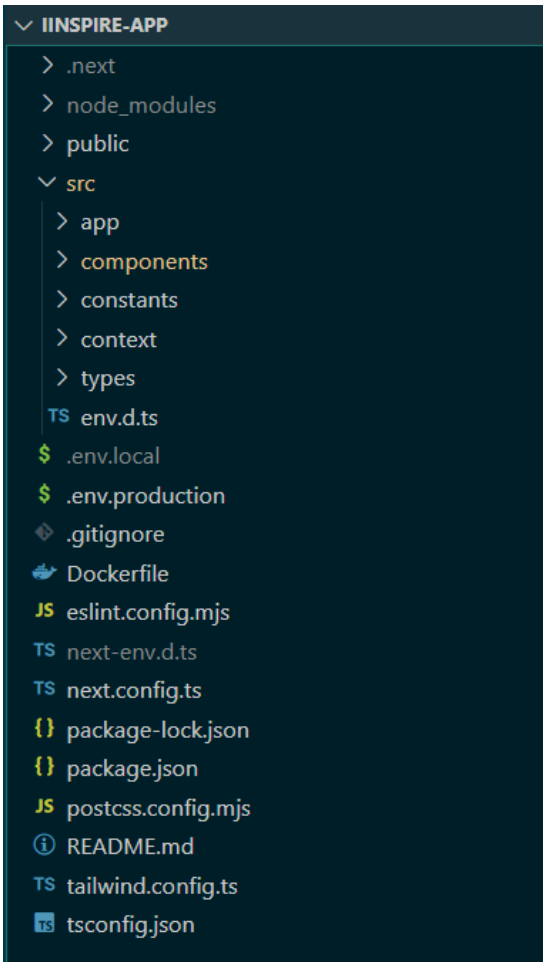
The backend is built using NodeJS and the Express framework, utilizing the controller, router, and service standard for backend systems. This architecture allows new routes to be scaled and developed quickly by reutilizing existing folder's routes. .

- Database tables are kept modular with their own folder for any routes accessing or inserting data in the database ex. Codes, links, Program, questions.
- Each tables folder has a router, controller, and service. Router handles the routing of each request and calls the subsequent function in the controller. The controller methods handle any request parameter, the responses to send back to the client, and calls the correct Service function. The service function query the database based on any of the request parameters and handles any of the business logic of inserting or formatting data.
- The app.js file deploys the NodeJS server which is able to handle any of the routes in the API folder. The dbconfig.js file manages connection to the database for any necessary queries.
- Package.json manages any Node dependencies that need to be installed for the app to run, this allows for the web tool to be set up and tested locally seamlessly.



Frontend

The frontend is built using the Next.js framework, along with React using Typescript and Chakra UI.



Folder Breakdown

/**public**: images, fonts, colors

/images: contains all the different images used in the app
/styles: contains the styling for different components of the app

/**src**: Stores all “actual” code

/app: houses main route location and large components. Each subfolder defines a URL route. For example, /app/about refers to the “https://MyStemGrowth.com/about” page

/components: contains the implementation for the different components, such as the different dashboards, navbar, footer, etc.

/constants: Stores API endpoints

/context: Houses authorization and localStorage functions used with user verification and information accessed on later screens (Ex: getting the user’s ID on the home page)

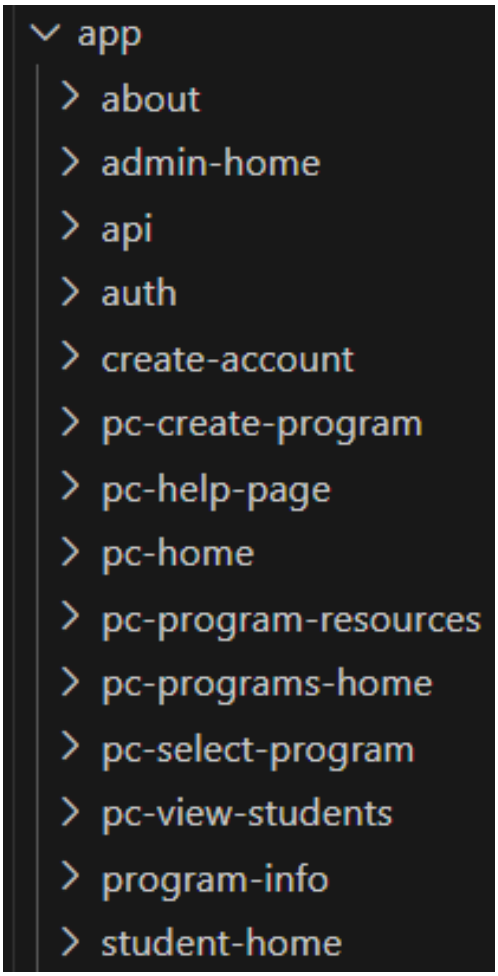
/types: defines modules for anychart and jsPDF

Package.json: stores dependencies for node (changes upon updating dependencies)

.gitignore: files or folders not pushed to Git with commits

Additional Information

Additionally, within the /app directory, the following naming convention has been defined for each URL route (subfolder):



Example: > (user role)-(brief-description)

4.2 Design Exploration

4.2.1 Design Decisions

Backend Role Management Design

To simplify access control and reduce the need for ongoing administrative overhead, we adopted a hands-off approach to user role management. Instead of manually assigning roles through an admin interface, the backend uses pre-generated signup codes, each mapped to a specific role or permission level. These codes are created and distributed by authorized administrators and are tied directly to backend logic that assigns the appropriate role upon registration.

When a user signs up using one of these codes, their role is automatically determined and stored securely in the database. This design ensures that only users with valid codes gain access to restricted functionality, without requiring human intervention post-signup.

By shifting role assignment to the signup flow, we eliminate potential points of failure associated with manual role management and enforce consistent privilege levels across all accounts. The system also allows for easy expansion of roles by generating new codes as needed, without requiring changes to the underlying codebase or user interface.

Separation of Frontend and Backend Services

We redesigned the architecture of the system to treat the frontend *(Next.js/React*) and backend *(Node.js/Express API)* as separate services. This modular design improves scalability, maintainability, and security by establishing a clear boundary between the user interface and application logic.

Both services are deployed as ECS Fargate tasks behind separate Application Load Balancers and operate within isolated private subnets inside a Virtual Private Cloud (VPC).

- This structure enforces the principle of least privilege: the backend is not directly exposed to the internet, and the database is only accessible to the backend service. Communication between components is encrypted and tightly controlled through security groups and IAM roles, ensuring that only authorized traffic flows between services.

- Impact: This change allows for a separate team structure, allowing for Cloud, Frontend, and Backend teams to work concurrently with minimal interdependency. Each team was able to develop, test, and deploy its respective components independently, which accelerated our development cycles and reduced the risk of cross-team bottlenecks. Overall, the redesign supports a secure, scalable, and production-ready cloud-native architecture.

Dashboard Design

We have intentionally tailored the dashboards to address the distinct needs and workflows of each user role. The student dashboard uses a scrolling layout designed for simplicity and mobile-friendliness, reflecting the way students are most likely to interact with the web application. This approach helps ensure an intuitive experience and minimizes potential confusion by presenting information progressively and in a clear, focused manner.

By contrast, the admin dashboard presents all key program information at a glance within a single view. This design recognizes that administrators typically require quick, comprehensive access to data and are less likely to engage in prolonged interaction with the application. By providing an at-a-glance overview, we support their need for efficiency and streamlined decision-making.

4.2.2 Ideation

We needed to determine the best way to deploy and manage our survey application infrastructure on the cloud. Our key objectives were:

- Security (minimal public exposure)
- Scalability
- Maintainability
- Developer independence (modular responsibilities)
- Automation through CI/CD

To identify the best path forward, we explored multiple deployment and hosting strategies, evaluating each based on the above criteria. Our ideation process included brainstorming sessions, trade-off analysis, and reviewing AWS architectural best practices. We loosely followed the Lotus Blossom technique, where our core design problem was deploying a secure, scalable cloud infrastructure—was placed at the center, and possible solutions branched out.

Options Considered:

1. Monolithic Deployment on a Single EC2 Instance

Pros: Simple to set up; all-in-one environment.

Cons: Difficult to scale; poor fault isolation; security concerns due to public exposure; not production-grade.

Reason for Rejection: Didn't support separate team workflows and lacked scalability/fault tolerance.

2. Serverless Architecture using AWS Lambda and API Gateway

Pros: High scalability and low maintenance; no need to manage infrastructure.

Cons: Complex to manage multiple functions; cold start latency; state handling for surveys would be difficult; vendor lock-in.

Reason for Rejection: Introduced complexity in coordinating frontend/backend integration and limited control over networking and compute lifecycle.

3. Docker Containers on Amazon ECS with EC2 Launch Type

Pros: More control over infrastructure; cost-effective for large workloads.

Cons: Requires EC2 instance management; overhead in scaling clusters and patching hosts.

Reason for Rejection: Introduced unnecessary infrastructure management overhead and reduced the benefits of containers as a service.

4. Managed Kubernetes via Amazon EKS

Pros: Industry standard; extremely flexible and powerful.

Cons: High complexity and learning curve; overkill for a relatively simple application; slower iteration for a student project.

Reason for Rejection: Complexity outweighed benefits for our use case; required more time and expertise than feasible.

5. Fargate-based Microservices with Independent Load Balancers

Pros: Fully managed compute; no infrastructure to patch; secure; ideal for isolated service deployment; integrates well with CI/CD and IAM.

Cons: Slightly more cost than EC2 but worth it for reduced management effort.

Reason for Selection: Met all goals: high security (private subnets), isolation of services (frontend/backend), automation (CI/CD with Kaniko), and scalability with low overhead. Enabled modular team development and deployment.

4.2.3 Decision-Making and Trade-Off

For our decision-making process, we began by clearly defining the key requirements and objectives of our project. We then conducted research to identify a range of potential solutions that could meet these needs. For each option, we developed a detailed pros-and-cons analysis to evaluate its strengths and limitations relative to our goals. This structured approach allowed us to systematically compare the alternatives and ultimately select the solution that best aligned with our project’s requirements.

4.3 Final Design

4.3.1 Overview

Backend Overview

Our backend is built using Node.js with the Express framework, providing a lightweight and efficient server environment for handling API requests. This server acts as the central hub for processing data submitted from the frontend, such as form inputs and survey responses.

Each API route is defined using Express, enabling clean, modular handling of different types of requests for example user authentication, program data retrieval, survey result graph generation. These routes are designed to enforce validation and security best practices, ensuring only authorized requests are processed.

The server runs as an isolated cloud service and communicates securely with our database and frontend interface. By keeping the backend logic separate and stateless, we ensure that the system remains scalable and easy to maintain, even as the application grows.

Frontend Overview

Our web application is developed using Next.js with TypeScript and uses Chakra UI for UI components. The application is structured around three distinct user flows: the student flow, the admin flow, and the program coordinator flow.

In the student flow, users can complete surveys, visualize their results through interactive graphs powered by AnyChart, and export their results as PDF documents for personal reference.

In the admin flow, administrative users have the ability to generate admin and program coordinator codes to manage user roles. Admins can also access detailed program information, including the option to download both survey questions and survey results in CSV format for further analysis.

In the program coordinator flow, program coordinators can view, edit, or create programs, generate program-specific codes to invite students, and customize the program’s resource page to tailor support materials for student participants.

Cloud Overview

Our project is hosted entirely in the cloud using Amazon Web Services (AWS), which allows us to deliver a secure, scalable, and highly available application without managing physical servers. The cloud infrastructure handles everything from routing website traffic, running our code, storing data, and automatically deploying updates when the code changes.

- **Frontend Web Server (User Interface)**
  The part of the website that users see and interact with (built using React) is hosted in the cloud and served securely through a system called a **Load Balancer**. When someone visits mystemgrowth.com, this service routes their traffic to the frontend application running in the cloud.

- **Backend Server (API)**
  The backend (built with Node.js) processes form submissions, survey responses, and handles secure communication with the database. It runs as a separate service in the cloud and is only accessible through authenticated channels (e.g., from the frontend app or admins).

- **Cloud Database**
  All user responses and program data are stored in a secure AWS database service through **Aurora MySQL**. This database runs privately in the background and is not directly exposed to the internet.

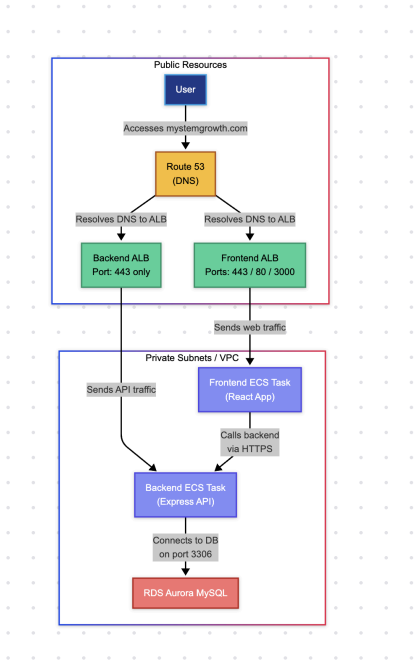
- **Automated Deployment (CI/CD)**
  Every time our team updates the code, it is automatically tested, packaged, and deployed to the cloud using GitLab's CI/CD pipeline. This ensures that the newest version of the website is live within minutes, with no manual setup needed.

- **Domain and Routing (DNS)**
  We use a service called **Route 53** to manage traffic for our custom domain, mystemgrowth.com. It knows whether to send traffic to the frontend (website) or the backend (API), depending on what's being requested.

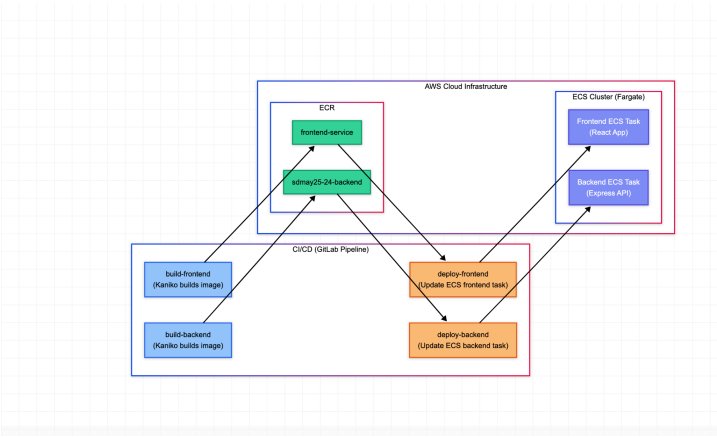
- **Monitoring and Logs**
  Our infrastructure is constantly monitored using **CloudWatch**, which helps us detect problems, view server logs, and troubleshoot issues quickly.

4.3.2 Detailed Design and Visual(s)

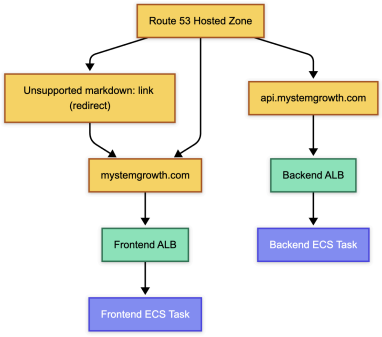
Cloud Network Architecture (VPC + ALBs + ECS + RDS)



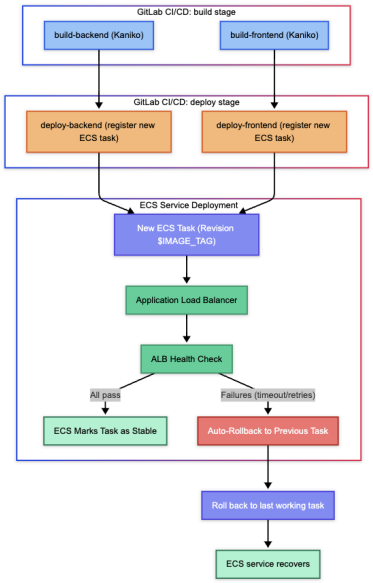
CI/CD Workflow with GitLab Pipelines and Kaniko



Route 53 DNS and Domain Routing Setup



Service Deployment Lifecycle



4.3.3 Functionality

Admin Functionality:

- The highest privilege role focuses on program management. “CEO”: Hands-off approach with minimal expected contact, effort, and time dedication. Primary Function: Invite future PC users and change user roles.
- They can:
    - Share join code with user intended to be a Program Coordinator
    - Share join code with user intended to be an Admin role
    - View current Admins/Program Coordinators
    - View all Programs and download specific program data (.csv file)

Program coordinator Functionality:

- The middle privilege role is to act as the leader of a particular program instance (e.g., Iowa State). One person is in charge of the activities of a single program instance. (NTH: multiple PC’s per Program Instance)Likely the most exhaustive role with the most responsibilities.
- They can:
    - Create an account with the PC role
    - Create a new Program Instance
    - Creates a unique join code for inviting Students
    - Select a certain Program Instance to view
    - View information of a particular Program Instance via a drop-down menu
    - View the number of students attached to the selected Program Instance
    - View a table of students attached to the selected Program Instance
    - Modify Program resources

Student Functionality:

- The lowest privilege role focuses on taking surveys and viewing results. Primary Function: Join the program, take and view survey results
- They can:
    - Create an account using a given Program Instance code (Ex: ISU2025)
    - View the program-specific “Resources” page
    - Take survey
    - View survey results
    - View the “About” page

4.3.4 Areas of Challenge

1. Managing Scope Creep
    a. As we gained technical momentum and clarity on implementation timeframes, the team started proposing more features than initially scoped.
    b. The original plan risked becoming too ambitious, jeopardizing deadlines and deliverable quality.
2. Balancing Technical Progress with User Requirements
    a. Some desired features were technically feasible but added significant complexity (e.g., real-time updates or user analytics dashboards).
    b. Users (students and admins) had varying expectations for what the platform should deliver—requiring prioritization.
3. Defining and Aligning Requirements
    a. Lack of clear definitions for user roles, permissions, and survey workflows early in the project.
    b. Differences in interpretation among team members and stakeholders caused implementation delays.

How We Addressed These Challenges

1. Re-scoping and Prioritization
    a. We held structured meetings to define a clear Minimum Viable Product (MVP) and set aside non-critical features.
2. Stakeholder Alignment
    a. Weekly check-ins with our advisor, Dr. Rover, helped validate our revised scope and ensure we stayed aligned with expectations.
3. Backlog Management
    a. We created a “Nice-to-Have Features” backlog to document ideas for potential future development by other teams.

.

4.4 Technology Considerations

1. Frontend: Next.js with React and Chakra UI
Strengths:

- Modern, performant framework with support for server-side rendering.
- React provides reusable component structure and integrates well with state management.
- Chakra UI offers accessible, responsive components out of the box, speeding up development.

Weaknesses:

- Learning curve for integrating Chakra UI's theme system.
- SSR and API routing in Next.js can be confusing to new developers.

Trade-offs:

- We chose Chakra UI over Tailwind for developer velocity and accessibility, even though it offered less design flexibility.

2. Backend: Node.js with Express
Strengths:

- Lightweight and fast, with non-blocking I/O suitable for handling survey responses.
- Massive ecosystem (npm), allowing rapid integration of middleware (e.g., validation, auth).

Weaknesses:

- Less structured compared to typed backends (e.g., Java Spring Boot), requiring extra care in code organization.

Trade-offs:

- Selected over Java or Python backends for rapid prototyping and the team's familiarity, despite its weaker type safety.

3. Cloud Hosting: AWS (Route 53, ALB, ECS Fargate, Aurora MySQL)
Strengths:

- Highly scalable and production-grade infrastructure.

- Fargate removes the need for server management; ECS task separation improves fault isolation.
- Aurora MySQL offers high availability and managed scaling.

Weaknesses:

- Complex setup requiring deep understanding of VPCs, IAM, and networking.
- Vendor lock-in and costs can increase as usage grows.

Trade-offs:

- Chose AWS for long-term maintainability and compatibility with industry practices, over simpler options like Heroku.

5 Testing

5.1 Unit Testing

For the front end created with React, the Jest Testing Library can write unit tests for React components, simulating user interactions and verifying their behavior. Jest can also be used to test functions and simulate HTTP requests. For the back-end, Node.js code can be tested using Jest to assess the correctness of API endpoints and the back-end logic. Additionally, AWS services can be incorporated into the testing process by setting up isolated environments for testing using Docker containers, enabling the emulation of real AWS resources. In this way, unit testing in this web project integrates seamlessly with the AWS infrastructure, React front-end, and Node.js back-end, helping to identify and rectify issues early in the development cycle to prevent problems from arising.

5.2 Interface Testing

There are multiple interfaces to be tested for our design. Users who access our application will be prompted to enter login credentials or create an account. These credentials will need to be veried and stored. We plan to use JSON Web Tokens to authenticate user access. A capability that will be tested is retrieving survey data from the back-end and displaying it to the user's prole on the front end. While the users take the survey, the information they enter and submit will be transmitted to the back-end to be stored.

5.3 Integration Testing

There are two critical integration paths we will need to test in our design:
1. Our Amazon EC2 instance and Amazon RDS will both need testing to ensure our web application can be built and run on our server with access to get and post data to our database. We will need network stress testing to ensure our web app is up and running, dependency testing to ensure the web application can be built, and testing over API calls and activity on the EC2. Some tools we may use are Amazon CloudTrail and Datadog.
2. Front-end tests over React and JavaScript. We plan on using Jest to test whether our React components are working properly. If a component were to regress by someone accidentally changing or adding to the UI, our tests can catch these changes.

5.4 System Testing

Considering that our project is entirely software, the "system" that we are testing is the entire web tool. Testing the entire system, in our case, involves ensuring an overall positive user experience. This includes what the user sees and how they can tell if they receive correct data. This can be tested using unit testing with Jest, as mentioned earlier, and it should be tested on parts of the web tool that heavily involve front-end and back-end communication. Examples of where this is present include when a user tries to log in and when a user imports data, expecting a result in graphics

5.5 Regression Testing

To prevent new updates from breaking existing functionality, we implemented a combination of version control, containerization, and CI/CD validation within our cloud infrastructure.

Code Review Process: All code changes require human review before being integrated. Smaller Features will require at least one team member to review and approve the changes. Larger Features: will require approval from two team members to ensure code quality, maintainability, and compliance with project standards.

Rollback Support via ECS and ECR:
After migrating to Amazon ECS, we are now storing versioned Docker images in Amazon ECR. This setup allows us to quickly revert to a stable version if a deployment fails or introduces regressions.

CI/CD Pipeline Safeguards:
Our GitLab CI/CD pipelines automatically build and test both the frontend and backend services prior to deployment. These steps include preliminary checks that help detect and prevent faulty builds from being pushed to production, maintaining overall stability and confidence in each release.

5.6 Acceptance Testing

Requirement Validation:

The team will conduct a walkthrough of the project, addressing each requirement. A comprehensive checklist will be used during this process to track progress and confirm that all larger specifications have been met. This step ensures internal confidence in the product's quality before presenting it to the client.

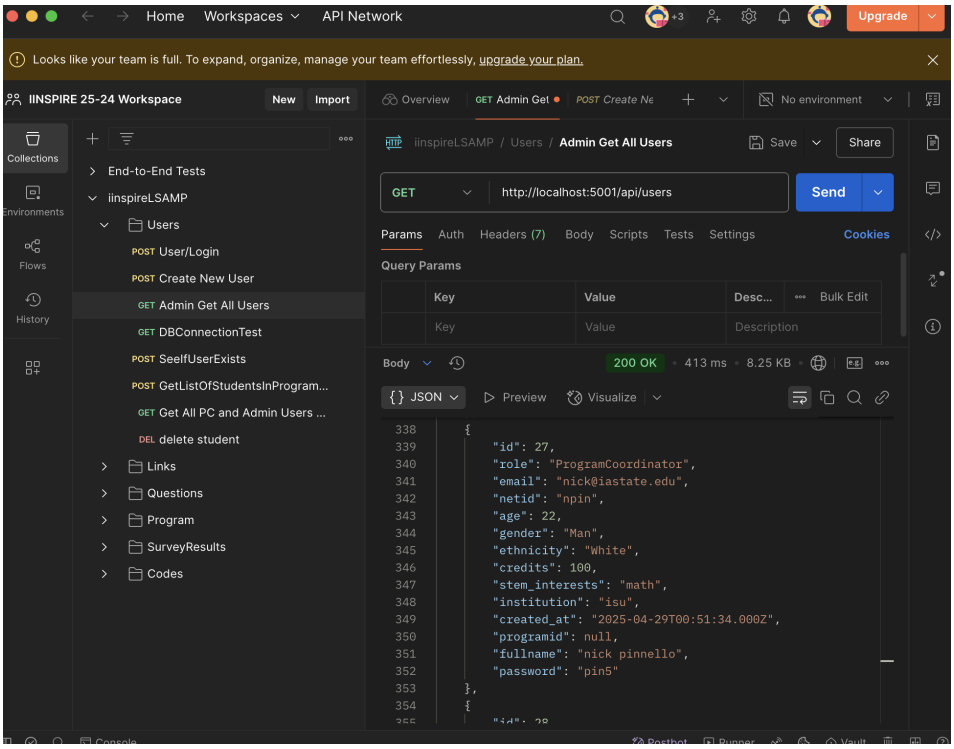
Client Presentation and Feedback:

The completed project will be presented to the client for review. This presentation will confirm that the project aligns with their expectations and to collect feedback. Following agile practices, we will hold regular feedback sessions throughout development. This approach ensures the final product is ready for approval upon completion

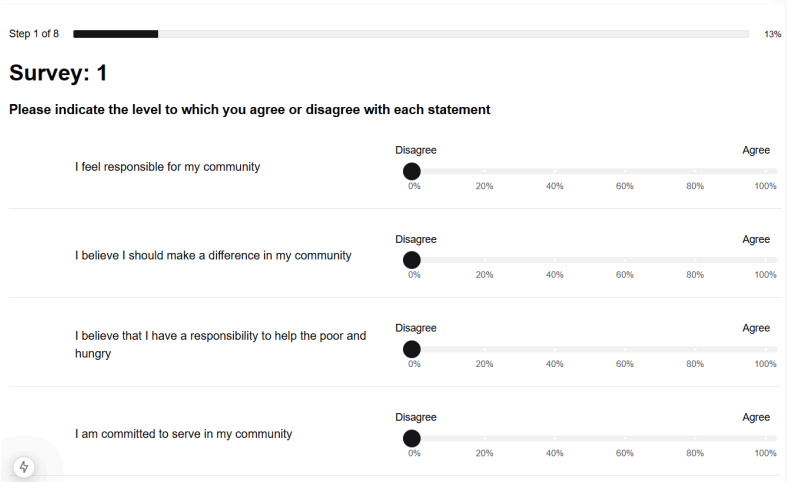
5.7 User Testing

As developers, we frequently put on our “user hats” and act as if we were a user using our software for the first time when testing. This process is good for nailing out initial kinks and bugs. However, it doesn’t get everything. That’s why we would frequently meet with Dr. Rover and Ally from the University of Iowa and show them our progress to get any and all feedback about design changes, tweaks to some user flows, or just some spelling changes we might’ve missed.

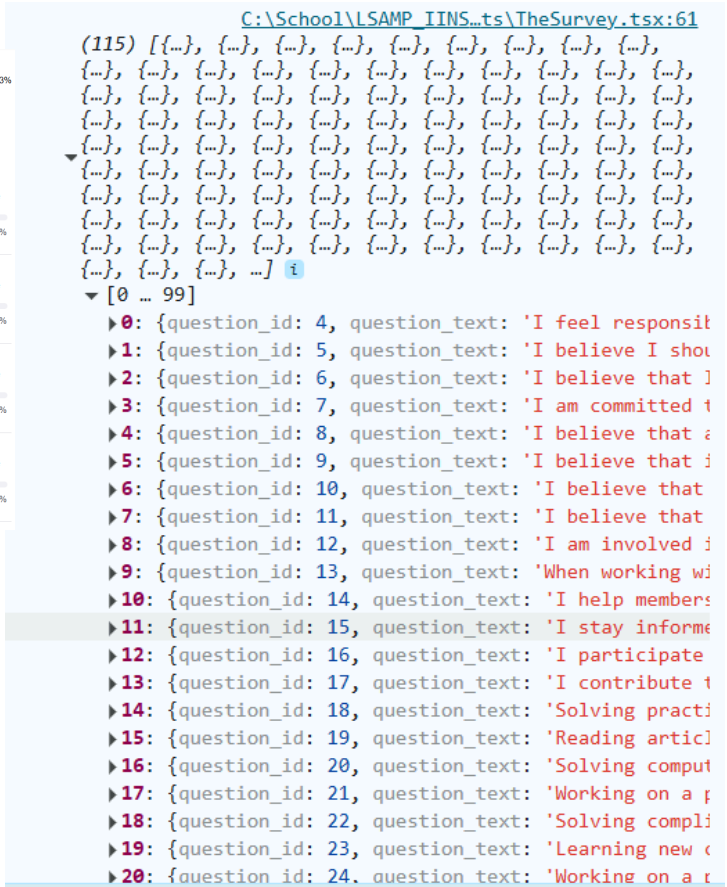
5.9 Results



Postman request for testing all users



Test fetching questions from backend



6 Implementation

6.1 Design Analysis

Our implemented design performs reliably and meets the client's core functional requirements, demonstrating a strong alignment with the original project goals.

What works well:

- User Interface/User Experience (UI/UX):
  - The interface is intuitive and easy to navigate, which both Dr. Rover and Ally from the University of Iowa responded positively to. This works well because we followed usability best practices and incorporated client feedback during development.
- Cloud Infrastructure
  - The cloud side of the project uses ECS with CI/CD to automatically update the production environment when a pipeline succeeds. Additionally, by using AWS ALB with Route 53 and SSL certificates, the production environment is set up with its custom DNS name through HTTPS for secure traffic. This was all done in the intent to not have the need for a cloud team in the future for any other teams that may work on this project.
- Backend
  - All API routes for each page of the design work with the frontend. Our SQL database properly stores users and whether they are students, program coordinators, or admins. It also stores the student's information/ question responses used for the survey results.

The main things that did not work as expected were features from our original design that we didn't have time to fully implement. Including the ability to edit survey questions after creation, full support for the researcher role, including managing and viewing survey results, Integration with NextAuth for login and account creation, which we initially planned but later replaced with a simpler approach due to time constraints.

On the **cloud infrastructure side**, we also encountered several unexpected challenges:

- We didn't realize our CI/CD runners couldn't use Docker in privileged mode, which broke our original plan to use Docker-in-Docker for builds,To address this, we switched to using BuildKit and Kaniko, and had to rework our CI/CD scripts accordingly.

- We also had to separate the ECS tasks for the backend and frontend, instead of running them together as originally designed, Additionally, we discovered we needed an Application Load Balancer (ALB) for the backend service in order to expose it properly, which added more configuration work and complexity.

7 Ethics and Professional Responsibility

Building Trust: Honest communication creates a foundation of trust among group members.

Enhancing Collaboration: Honest communication fosters an environment where group members feel comfortable sharing ideas and giving constructive feedback.

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	Definition	Relevant Item from ACM Code of Ethics	Team Interaction / Adherence
Professional Responsibilities	Fulfill commitments, communicate honestly, and maintain standards.	*1.3 Be honest and trustworthy.*	Our team consistently held weekly meetings and provided updates on progress and blockers. We documented delays or issues transparently and revised plans as needed.
Responsible Conduct of Research	Ensure integrity and accuracy in work and avoid fabrication or falsification.	*1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.*	We attributed third-party tools and clearly documented our references in both code and presentations. We avoided copying code from unauthorized sources.
Quality of Engineering Work	Aim for high-quality, robust, and sustainable solutions.	*2.5 Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.*	We performed multiple rounds of testing, including edge cases, and collected feedback from test users. We tracked issues with GitHub and addressed critical bugs promptly.
Leadership and Mentoring	Support development of peers and junior members and foster ethical behavior.	*3.1 Ensure that the public good is the central concern during all professional computing work.*	Our senior developers helped onboard new contributors and provided walkthroughs for backend setup and authentication modules.

7.2 Four Principles

Area	Beneficence	Nonmaleficence	Respect for autonomy	Justice
Public health, safety and welfare	The purpose of this tool is to help students get the resources and education they need to grow academically but also personally.	Students will benefit from understanding what areas they excel in and what areas they need to improve in. This will help them in their careers and with their journey in school, as they will know what areas to spend more time improving.	Allowing students to understand their strengths and weaknesses will help them make an informed decision about their future without forcing them to choose a certain path.	Helping students get resources to improve in areas they may be weaker in because of a lack of exposure to STEM related courses or concepts will help level the playing field and give them a fairer chance at success.
Global, cultural and social	Helping underrepresented communities highlights our project's focus on humanity. The biggest goal of our project is to help those who traditionally have been excluded from STEM fields get opportunities they might not have otherwise.	Introducing communities who are underrepresented in STEM to our program will benefit those communities by giving them more information and resources. This complies with the idea of avoiding causing harm.	Our project aims to expose underserved communities to STEM majors and careers. We will disclose pertinent information about these career paths to help students decide if this is the right choice for them.	A major goal of our project revolves around giving equal opportunities to those in cultural, social and economic communities that are underrepresented in STEM fields.
Environmental	While our project itself doesn't directly impact the environment, If the students who participate in the program lead a career in environmental sciences or something related, they could contribute positively to	Our project does not deplete or harm our environment of its resources.	Introducing students to STEM majors involved in protecting our environment can equip students with the resources and knowledge to contribute towards this effort if it is something that interests them.	By involving individuals from underrepresented communities in environmental fields, we can help create opportunities for them to address environmental challenges that directly impact their communities. This not only benefits these communities but also contributes valuable insights and solutions to

	protecting our environment.			broader environmental issues.
Economic	Students who do a career in a STEM field will likely see a positive impact on their financial situation.	A career in a STEM field typically does not harm a person financially, and in fact tends to have the opposite impact.	Exposing students to STEM careers may present opportunities to better their financial situation through high-paying careers. Introducing these opportunities gives students autonomy over their finances.	Helping all students have more equal opportunities to STEM careers also gives more equal opportunities to financial well-being.

7.3 Virtue

Compassion: Combining empathy and action to alleviate the plight of others

Justice: Ensuring fairness by refraining from causing harm or disadvantage to others and actively working to restore justice when needed

Integrity: Aligning our conduct with the way we view the world by backing up our words and beliefs with our actions.

Our team has shown compassion by actively working to provide underrepresented students with resources and opportunities in STEM. This effort demonstrates a commitment to alleviating the disparities they face, combining empathy for their struggles with actionable solutions such as mentorship, academic support and access to STEM programs.

By focusing on bridging the gap for students traditionally underrepresented in STEM fields, our team promotes justice. We are making an effort to create equitable access to education, offering opportunities to address systemic inequalities, and building a supportive community, which reflects our commitment to restoring fairness and providing everyone with their due chance to succeed.

Our team has upheld integrity by aligning our actions with our mission to empower underserved communities in STEM. We have demonstrated this virtue by contributing towards the IINSPIRE program and ensuring all team members are held accountable for supporting the project's goals.

Alex:
Virtue I have demonstrated: Adaptability
Why it's important: Adaptability is important because the needs of our client and the structure of our team is dynamic, so flexibility is crucial
How I have demonstrated it: I stepped into a leadership role to address the evolving needs of our team and to act as a liaison between our group and our client.
Virtue I have not demonstrated: Patience
Why it's important: It's important to maintain a composed demeanor during challenging situations or when facing setbacks
How I can demonstrate it: I can demonstrate patience by taking a step back to assess the situation calmly and provide thoughtful responses to feedback

Charlie:
Virtue I have demonstrated: Dedication
Why it's important: Dedication ensures sustained focus toward achieving long term goals
How I have demonstrated it: I have demonstrated dedication by consistently preparing and organizing the necessary resources to ensure our project is fully equipped for implementation next semester
Virtue I have not demonstrated: Assertiveness
Why it's important: Assertiveness is important for effectively communicating ideas and opinions
How I can demonstrate it: I can demonstrate assertiveness by confidently expressing my thoughts during meetings and respectively challenging ideas when necessary

Isabelle:
Virtue I have demonstrated: Humility
Why it's important: Humility is important for recognizing areas for growth and embracing feedback to improve our project
How I have demonstrated it: I have demonstrated humility by actively seeking input from students in the program and experts in Human Computer Interaction and website design
Virtue I have not demonstrated: Perseverance
Why it's important: It's essential for pushing through challenges and staying committed to the project's goals even when progress feels slow or obstacles arise
How I can demonstrate it: I can demonstrate perseverance by continuing to work through technical difficulties or unexpected challenges

Landon:
Virtue I have demonstrated: Compassion
Why it's important: Compassion allows team members to build trust and support each other by addressing challenges with kindness and empathy
How I have demonstrated it: I have demonstrated compassion by being understanding toward my teammates' challenges and offering help when they need it
Virtue I have not demonstrated: Resilience
Why it's important: Resilience is important for bouncing back from setbacks and maintaining a positive outlook despite challenges
How I can demonstrate it: I can demonstrate resilience by staying motivated despite setbacks, learning from failures and encouraging others to persevere

Matthew:
Virtue I have demonstrated: Integrity
Why it's important: Integrity is important for establishing trust and credibility with others by aligning actions with our values and principles
How I have demonstrated it: I take ownership of my responsibilities and follow through on commitments such as meeting deadlines, participating in group meetings and helping other group members when needed
Virtue I have not demonstrated: Collaboration
Why it's important: It's essential for creating a cooperative team environment where everyone's contributions are valued
How I can demonstrate it: I can demonstrate collaboration by seeking input from others and working closely with teammates to achieve our common goals

Max:
Virtue I have demonstrated: Vision
Why it's important: Vision is important because it guides our efforts towards creating an impactful, lasting change
How I have demonstrated it: I have demonstrated this value by outlining the big picture for our project design and contributing ideas to improve our project
Virtue I have not demonstrated: Pragmatism
Why it's important: It is essential for making practical and realistic decisions that align with the team's goals and resources
How I can demonstrate it: I can demonstrate pragmatism by evaluating the feasibility of my ideas, considering potential challenges, and making adjustments based on practical considerations

Nick:
Virtue I have demonstrated: Inclusivity
Why it's important: Inclusivity ensures that everyone feels valued and heard. Inclusivity strengthens team cohesion and celebrates diverse perspectives.
How I have demonstrated it: I have demonstrated inclusivity by making sure all team members' opinions are included in discussions and decision-making
Virtue I have not demonstrated: Curiosity
Why it's important: It's important for driving innovation and continuous learning because it encourages the exploration for new ideas and perspectives
How can I demonstrate it: I can demonstrate curiosity by constantly asking questions and exploring different approaches to improve our project

8 Conclusions

8.1 Summary of Progress

Overall Design Achieved:

- We successfully developed and deployed a fully functional, cloud-native web application:
    - Frontend built with React/Next.js and Chakra UI for responsiveness and accessibility.
    - Backend built with Node.js and Express for efficient, RESTful API communication.
    - AWS-based infrastructure using ECS Fargate, Route 53, ALBs, and Aurora MySQL for secure, scalable deployments.
- Our architecture supports modular development and future expansion, allowing future senior design teams to build on a stable foundation.

Team Accomplishments:

- Created a secure, role-based user onboarding flow, using randomly generated codes that:
    - Assign proper access roles and eliminate manual user creation by admins.
- Implemented a dashboard for Admins and Program Coordinators:
  Allows them to view and export survey results by user and program.
    - Enables ongoing data collection for program effectiveness reporting.
- Developed clean API documentation and modular code structure to ease handoff to future teams.
- Overcame technical and team collaboration challenges through clear division of responsibilities across frontend, backend, and cloud groups.
- Developed a secure and scalable cloud platform that protects backend services and database access within private AWS subnets
- Set up an automated CI/CD pipeline using GitLab to deploy frontend and backend services with no manual steps
- Enabled new developers to deploy and test code in the cloud automatically with each push

8.2 Value Provided

User-Centered Design and Needs Met

For Students:

- Before: Students had no visualization or access to past surveys.
- Now: Students can log in and see interactive bubble graphs representing their STEM growth across six outcome areas.
- Value: This empowers students to reflect on their progress and increases engagement. Several test users reported the visualizations made their results “feel more real.”

For Admins/Coordinators:

- Before: Survey data was locked away in backend tables with no easy way to explore or export results.
- Now: Admins can view aggregated survey responses, filter by program, and export formatted results.
- Evidence: During demo sessions, our advisor noted the reporting features were essential for grant reporting and program evaluation.

8.3 Next Steps

Our team has consistently planned for future groups and their contributions. In the above sections describing each user functionality, you will see the NTH label, which means it is “Nice to Have”. These features are things either the Senior Design team or Dr. Rover have identified as items for the next team to tackle.

Here are the future items for each user:

Admin

- Invite/Remove Researchers
- Add, remove, or edit survey questions

Program Coordinator

- Change/edit the join code for a Program Instance
- Allow multiple PC’s for a single Program Instance
- View an average results graph for all students in the selected program
- Close a survey and set a new survey to be the default survey shown to students
- View a particular student’s graph (selected from the user list)
- ⭐Allow a PC to move/transfer students from Program Instance to a new Program Instance without requiring a new account. This will allow for surveys taken under a different Program Instance to be viewed by the Student regardless of what Program Instance they are currently a part of.

Student

- Compare multiple surveys
- Enable switching between programs
- Enable being transferred to a new program
- Change/Edit account information

Researcher

- Create an account
- Store/View survey results
- Share/Download Data
- Methods to visualize survey results

9 References

[1] J. Farmer, "Visual Analytics in Education: A Framework for Data-Informed Student Support," *Journal of Educational Technology Systems*, vol. 49, no. 2, pp. 145–161, Dec. 2020.

[2] P. N. Howard, A. Duffy, and D. Freelon, "Opening Closed Platforms: The Ethics of Interfacing," *IEEE Internet Computing*, vol. 20, no. 3, pp. 25–31, May–Jun. 2016.

[3] S. Krishna and S. B. Jha, "Survey Systems in Academia: An Evaluation Framework," *IEEE Transactions on Learning Technologies*, vol. 14, no. 1, pp. 80–90, Feb. 2021.

10 Appendices

Appendix 1 – Operation Manual

Cloud

The overall design goal of the cloud portion of this project was to make it never needed to be touched. This system has been designed so that the application can continue development without needing to change any cloud resources.

- **University Git repository:** this project is now hosted on a private gitlab repository owned by the university this will be passed down to future developers and has features allowing for easy continuation of development on this project.
- **CI/CD:** a major improvement to last year's design is the implementation of CI/CD pipelines. These have been designed to automatically run when there is a new commit in the master branch of the git repository. These will automatically build, upload and new versions of the application on the cloud meaning all future developers have to do is merge their changes into the main branch and those changes will be automatically deployed on the mystemgrowth website

Networking Setup (VPC, Subnets, Security Groups)

The system runs in a custom VPC (sdmay-25-24-vpc) located in the us-east-2 (Ohio) region. All ECS tasks, RDS, and the ALB operate within *private subnets*. The ALB handles incoming public traffic and routes it to internal services.

Private Subnets Used:

subnet-0080eb9ac67ee186b, subnet-06db68f7d1f548081, subnet-0f8497e99daf802ea, subnet-00b37daf24bd29e9d

Security Groups:

- ECS-SG – ECS tasks, allows HTTP/HTTPS
- ALB-SG – ALB, allows public access on ports 80/443/3000
- RDS-SG – Allows MySQL (port 3306) from ECS
- default – Used temporarily during setup (avoid in production)

Database Setup (Amazon RDS – Aurora MySQL)
Aurora MySQL is used for persistent data storage. It is configured for high availability and runs entirely within private subnets.

Endpoint: sd-may2524-db.c588auok61gp.us-east-2.rds.amazonaws.com

- Port: 3306
- User: root
- Instance Class: db.t3.micro
  Multi-AZ: Enabled across us-east-2a and us-east-2b
- Security Group: RDS-SG

Elastic Container Registry (ECR)
Docker images are stored in Amazon ECR, one repository for each service.
CI/CD pushes both latest and commit-tagged images on each update.

- frontend-service – Used by the frontend-service-task-new ECS service
- sdmay25-24-backend – Used by the backend-service ECS service

Application Load Balancer (ALB)
The ALB routes public traffic to the frontend service.

- **Name:** sdmay25-24-ALB
- **DNS:** sdmay25-24-ALB-777576476.us-east-2.elb.amazonaws.com
  Listeners:
  - Port 80 → HTTP → Target group sdmay25-24-NEW-ALB-TG
  - Port 443 → HTTPS (ACM cert for mystemgrowthprofileserver.com)
  - Port 3000 → For development only

Note: Backend is not exposed through the ALB. It is only accessible within the private VPC.

Elastic Container Service (ECS)
AWS ECS (Fargate) runs containers for both frontend and backend services.

- **Cluster:** sdmay25-24-ECS
- **Launch Type:** FARGATE
- **Subnets:** Private subnets only
- **Security Groups:** ECS-SG, ALB-SG, RDS-SG, default
- **CloudWatch Logging:** Enabled for both services
- **Services:**
  - backend-service – Private-only, no ALB integration
  - frontend-service-task-new – Routed via ALB to serve public traffic

CI/CD Pipeline (GitLab + AWS)
GitLab CI/CD automates building and deploying containers.

- **Stages:** build, deploy
- **Builder:** Kaniko (Dockerless)
- **Image Tags:** latest, Git commit SHA
- **Environment Variables:** Set in GitLab CI/CD and in .env files
  Trigger Rules: Pipeline only runs when changes are detected in the relevant service folder

Frontend

The primary goal of the frontend team was to create clean, modern, reusable components as building blocks for the UI/UX. Additionally, the student role had a large focus on responsiveness for mobile devices, since students may take the survey on their phones.

To view and run the frontend locally, navigate into the iinspire-app folder from the Git Repo. You will need to run “npm install” to ensure your dependencies are updated.

Then, use the command “npm run dev” to start a local development environment and click the link using “https://localhost:3000”.

Upon changing code, use “Ctrl + S” to save and the localhost server will update with your changes.

When your changes are completed, use the branch naming convention of “issue #-briefDesc”. For example: 23-createPCHome.

View Additional Frontend Documentation here:
https://docs.google.com/document/d/1xCZM9W_9N0Qq_JYrChEArAEm881zDPkQYJYdEE3PMPs/edit?usp=sharing

Backend

The Primary goal of the backend team was to create modularity between API endpoints and database tables while continuing to scale up features and stored data.

The **modularity** was achieved using the **Router, Controller, Service architecture**, giving each database table an individual folder with these three files handling different aspects of responding to client requests. This architecture kept files simple and lightweight so code can be easily reused.

The **dbconfig** file creates a DB connection pool, this limits the number of connections the Node server must make to the sql server for queries, this way service functions can borrow a connection, make their query and then return it back to the pool.

App.js handles creation and deployment of the nodejs server, The API folder is exported to the app.js folder so all request starting with /api will automatically be routed and handled to the correct table's router without the app.js file needing any changes if new api endpoints are created.

Appendix 2 – alternative/initial version of design

- Versions considered before client's specifications have changed
    - **AWS EC2 Instance:** Initially we considered using the AWS Elastic Compute Cloud (EC2) to manually provision production deployments. After more consideration we decided to use the AWS Elastic Container Service (ECS) since it would automatically update our production environment after it passed the CI/CD.
    - **NextAuth:** We considered using NextAuth for user authentication to get into the web application. However, after talking to the client, they wanted other functionality working better so we decided to go with a normal username and password combination to log into the application.
    - **GitHub:** At the beginning we were going to choose to use GitHub for our code repository. This changed when we had more conversations about GitLab's CI/CD pipeline capabilities and after talking with the client we chose to change to GitLab over GitHub.
- Versions considered before learning more about the project
    - **GitHub Actions for CI/CD:** Before understanding ECS and AWS-specific deployment needs, we started with GitHub Actions. But managing secrets, ECS task updates, and ECR was more complex there, so we migrated to GitLab CI/CD where we had better control and flexibility.
    - **Public Subnet Deployment:** We first assumed it would be okay to run everything in public subnets for easier access and testing. After looking into how we can improve security, we re-architected the setup to use private subnets for ECS and RDS, with only the ALB exposed.
    - **Researcher Role:** Initially, our plan was to have 4 roles: Admin, Program Coordinator, Researcher, and Student. After discussing the roles and responsibilities of each user type with our client, we decided that the researcher role was not necessary for this iteration of the web-app. We included some of the key roles from the researcher into the admin role to reduce complexity and meet the more pressing needs of our client.

- Versions that resulted in failure to achieve specifications, etc.
    - **Aurora MySQL (again):** The increased costs did not make sense for the project's budget and didn't offer clear benefits over regular MySQL for our workload.
    - **Single ECS Task for Full Stack:** This design didn't allow independent scaling or updates of backend and frontend, which became a problem once development diverged between the two.
    - **CI/CD Running on Every Commit:** Early pipelines built and deployed the entire stack every time, even if the change was small. This led to slow builds and unnecessary resource usage. Now, CI/CD only runs for changes in relevant directories.

Appendix 4 – Code

- https://git.ece.iastate.edu/sd/sdmay25-24
    - Note: for security reasons this is is not publicly accessible without being a project member assigned to this repo by ETG

Appendix 5 – Team Contract

Team Members

1) Landon Gulotta - Cloud
2) Alex Moeller - Project Manager/Frontend
3) Max Strater - Frontend
4) Nick Pinnello - Backend
5) Isabelle Raghavan - Frontend
6) Matthew Bennett - Cloud
7) Charlie Moreland - Backend

Team Procedures

Team Meetings: Monday, 10:00am, Virtual (Teams)
Client/Advisor Meetings: Monday, 2:30pm, In-Person (Durham)

Our preferred method of communication is primarily Discord between team members and our client/advisor. Additionally, we use text messaging for quicker responses.

When making any decisions, the team first talks to their group members (frontend/backend/cloud). Once a decision is made between them they then send the decision to the rest of the team for a majority vote.

We use Microsoft Teams to track all of our meeting minutes. For each meeting, we have someone responsible for logging and reporting the meeting minutes each week.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
3. Expected level of communication with other team members:
4. Expected level of commitment to team decisions and tasks:Leadership

Strategy for Guiding Work

Utilizing GitLab issues as well as making an updated task list every week after our advisor meeting.

Strategy for Recognition

To ensure that we are recognising the contributions of all of the team members we start each meeting by going one by one stating what they have accomplished for the week. We also state what the next task will be and if they need any assistance with anything related to the project.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team

- **Alex Moeller (Project Manager/Frontend Developer):** Skilled in leadership and organization, Alex ensures team coordination and clear communication with the client.
- **Nick Pinnello (Backend Developer):** Strong in backend development, Nick specializes in secure database management and API design.
- **Charlie Moreland (Backend Developer):** Charlie focuses on backend optimization and reliable database systems.
- **Max Strater (Frontend Developer):** Max combines technical skill and creativity to design functional and visually appealing user interfaces.
- **Isabelle Raghavan (Frontend Developer):** Isabelle excels at creating intuitive, user-friendly designs with attention to detail.
- **Matthew Bennett (Cloud Engineer):** Matthew is experienced in designing scalable and efficient cloud infrastructure.
- **Landon Gulotta (Security Engineer):** Landon brings expertise in cybersecurity, ensuring the application is secure and resilient. Some AWS experience to help get a working, automatic updating production environment. 2. Strategies for encouraging and support contributions and ideas from all team members:

- Regularly scheduled weekly meetings (both team and advisor-led) to discuss progress, challenges, and new ideas.
- Open communication channels, such as Discord, to encourage informal brainstorming and collaboration.
- Recognition of individual contributions during meetings and in project communications to motivate and foster engagement.

3. Procedures for identifying and resolving collaboration or inclusion issues

- Team members can report concerns during meetings or directly to the project manager for confidential discussions.
- A constructive feedback system ensures all team members feel heard and valued.
- Persistent issues are escalated to the advisor for impartial mediation and resolution.

Goal-Setting, Planning, and Execution

Spring Semester Goals

Our primary goals for this semester are to develop the application to meet our advisor's expectations in both functionality and design. We want to make sure it aligns with their vision and feedback. Additionally, we aim to implement a fully automated production environment, including CI/CD pipelines so that future updates can be pushed with minimal user intervention.

Consequences for Not Adhering to Team Contract

Handling Infractions

If there were to be an infraction, we will bring it up with the team as well as our advisor in our weekly meetings. We will also reach out to the individual for a separate meeting to discuss why their work effort is not meeting standards. If they continue the rest of the team will have a private meeting with the advisor to talk about next steps with the individual.

Continuing Infractions

If the fractions continue after we take steps to mitigate them, we will set up a meeting with the professors as well as our advisor for additional help. We will also discuss grading for the team without the individual's help and what their grade should be based on their contributions.

****

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) Landon Gulotta - DATE:4/28/2025
2) Max Strater - DATE: 4/28/2025
3) Nick Pinnello - DATE: 4/28/2025
4) Isabelle Raghavan - DATE: 4/28/2025
5) Alex Moeller - DATE: 4/28/2025
6) Charlie Moreland - DATE: 4/28/2025
7) Matthew Bennett - DATE: 4/28/2025