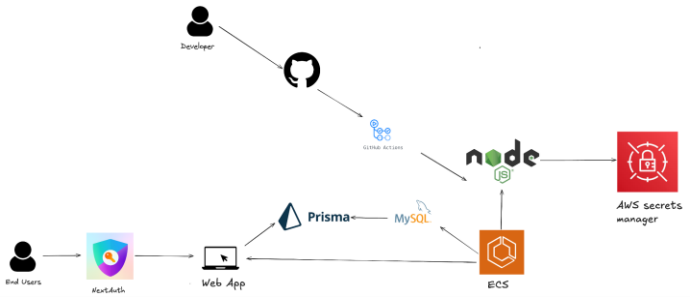
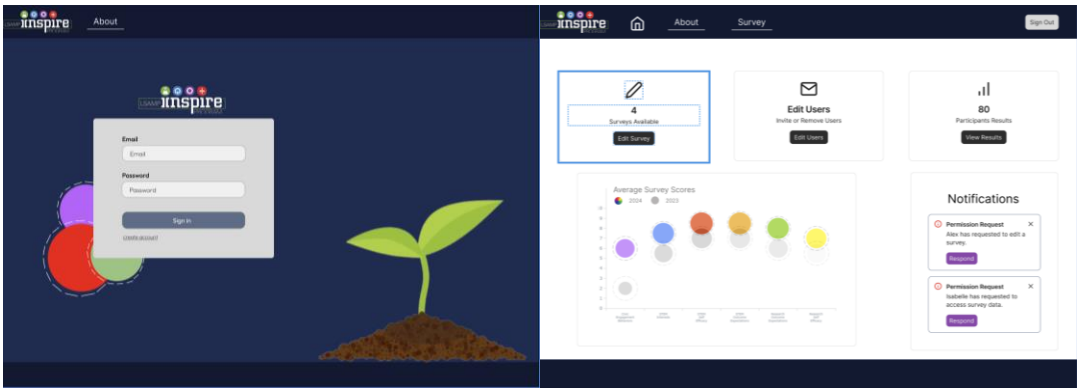
4.2 Design Exploration

4.2.1 Design Decisions

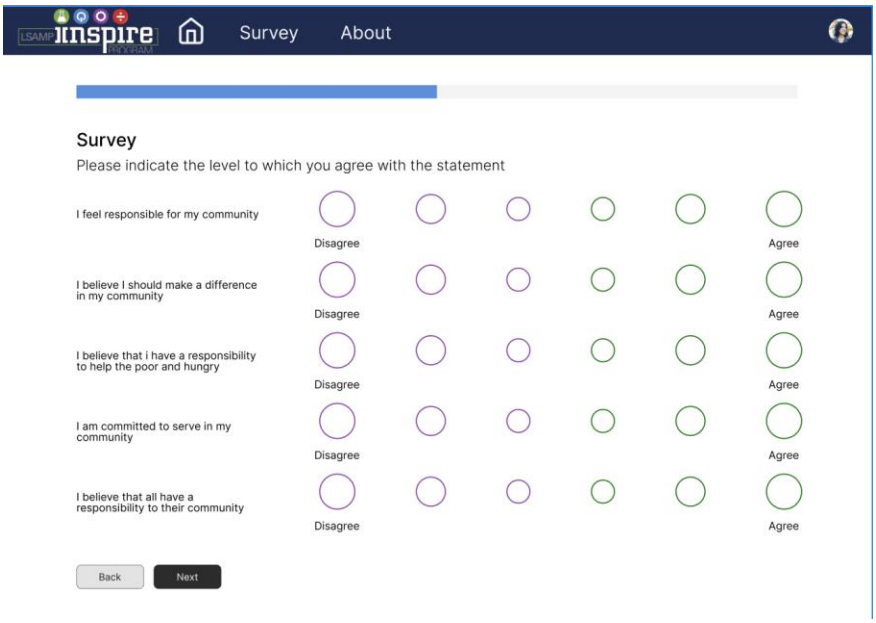
Cloud Server Rework:



Reworked Profile Page:



New Survey Markers:



4.2.2 Ideation

We considered many designs for the survey questions page to make it more visually appealing and user-friendly. We had an idea to add sliders and dropdown options for some of the questions to change the style and keep users engaged. Another idea we had was to add color to the survey bubbles themselves as shown in the picture. The color gives the user a better sense of range when deciding if they strongly agree or disagree on a question/ topic.

4.2.3 Decision-Making and Trade-Off

One apparent pros of the design choices we considered is that the page is more visually appealing, which was our goal. Users will be more engaged with these questions compared to the previous design of gray circles. One of these design choices will be the implementation of the various questions. The more we incorporate, the harder it will be to keep track of different question types, which may cause an error.

4.3 Proposed Design

4.3.1 Overview

The IINSPIRE STEM program supports underrepresented minority students, helping them engage in STEM fields. Researchers need a tool to administer surveys and visualize results to improve program support more efficiently. The current survey website, developed by the University of Iowa, is fundamental in design and needs essential functionality. The problem will be solved by redesigning the frontend and upgrading the infrastructure to improve the web app's speed and performance. This will make the survey-taking process faster for participants and allow researchers to create and manage different surveys easily. Continuous feedback from users will guide these improvements and the addition of new features.

4.3.2 Detailed Design and Visual(s)

Our system will be evenly distributed between our three teams of cloud, backend, and frontend. Because we are choosing to redo most of the previous design, each group has significant work to accomplish.

Cloud, Backend, and Frontend Breakdowns

Cloud: All changes have been made to improve simplicity and reduce our systems' complexity. ECS has been integrated to allow for automatic scaling of services, CI/CD

allows for changes to be automatically deployed, and Prisma/NextAuth reduces the need for other developers to create similar services

Backend: The team's lack of experience with new technologies like SSO integration, Prisma, and other security features poses a potential setback to project progress as developers get over the learning curve and implement different technologies. We are incorporating many industry standards to the design which is a great experience for all team members to gain new experience with commonly used web technologies.

Frontend: Providing revamped Web and Mobile interfaces for users. Significant visual changes include layout, functionality, style, colors, etc. None of us have significant experience in wireframe design, so we have to learn and experiment.

4.3.3 Functionality

The functionality of our web application can be divided among our four user types: admin, program coordinator, researcher and student.

For our admin, their goals when using our application are to be able to create and edit surveys, invite/remove users, update/patch the app and approve accessing/downloading survey data. Since many of their roles involve responding to requests, we decided to have a notification panel so they can easily approve/disapprove requests from lower tier users. In addition to overseeing requests, admins will also want to get a broad overview of how participants interact with the survey, which is why we display a chart showing the average of survey results over time and an option to view individual participants' survey results.

For the program coordinator their main responsibilities are to create/modify the surveys, view survey data results, share and download survey data, and modify survey visualization methods. Since their main concern is with the survey results itself , their dashboard is designed to allow different types of survey updates and modifications.

Researchers will also focus on the surveys, but with an emphasis on the results and collecting data. Considering this, we plan to allow survey results to be downloaded as csv files to allow for efficient data analysis. We will also give researchers the ability to submit requests for new survey questions and to request to edit current questions. Since they will need to be able to submit requests, we will have a "message" feature to allow them to send requests to admin for approval.

Finally, for students, their intended use of the application is to be able to take the survey and view survey results. After conducting a user feedback session, one common request from students was to have a page dedicated to resources at the University to help with their academic growth and community involvement. Considering this, we decided to add a page for resources divided into academics (such as tutoring and SI), clubs (such as WISE and Solar Car), and events like career fairs and company visits.

4.3.4 Areas of Concern and Development

Our current design heavily focuses on meeting the user's need for the same system functionality with an updated user interface. We have created a figma prototype of the revamped web interface and are iteratively updating this design through user interviews and user experience resource feedback. We are confident that this iterative process will leave us with an upgraded web design that meets user requirements. Our backend design focuses on recreating the previous team's solution using better technologies. We have captured all user functionality requirements from the previous team's design and will gather more information about backend requirements through user interviews with the University of Iowa research team.

The primary concern of our team's current design is the integration of new backend technologies. These technologies are new to the team and could pose a significant learning curve to getting a working version. However, these technologies will help create a more secure web application and have become the industry standard. Our team will actively monitor the timelines of implementing these technologies, and if some aspects of the backend design cannot be reasonably built our team has prioritized the most important of the backend features to be implemented. Another concern is fixating our design on meeting the student user group's need for an updated Web Interface and not spending enough time meeting the researcher user group's requirements for data collection and data exportation. Our client clarified to our team to prioritize the student user group first over the researcher user group. Our design can provide more functionality depending on the timeframe of our first prototype and the client's satisfaction with our design.

The team has several immediate plans to address these different concerns over our design. The team has scheduled a meeting with a Iowa State's Design Department member to receive feedback on our initial figma prototype. We hope to gain insight about the user experience of our web screen mockup as well as design thoughts regarding the default layout of each screen. The team has also scheduled a meeting with the University of Iowa researcher team. This meeting will help answer concerns over any research user group requirements and how to prioritize implementing them. This meeting will also offer insight into the standardization of data collection and exportation from our web application for the researchers to use. The immediate questions we have for our advisor is due to the slow process of working with ETG to get a new AWS account set up. How will this setback the original project timeline? For the University of Iowa team, we are wondering what the nature of the survey portability feature would look like? How different would a different universities program survey look like in comparison and what infrastructure would we need in place to support the addition of programs to IINSPIRE.

4.4 Technology Considerations

Front-end:

Next.js: Next.js is a java framework built on React. This is what we will be using to create our web application.

Pros:

- file based routing: for developers, file based routing is easy to visualize and understand because you can organize your route structure via files and directories instead of through code
- built in cache: this allows for faster speed and enhanced performance

Cons:

- potential learning curve for more advanced features

Chakra: Chakra is a component library that is easily integrated with React frameworks, such as Next.js which is what we are using to develop our application

Pros:

- easy to learn
- components are built to work well on different screen sizes, which is important for our project since we will be hosting our web app on desktop and mobile devices

Cons:

- some computational overhead which can impact performance on large-scale projects (although this shouldn't be a concern for our project specifically)

Node.js: While Node.js is primarily a backend technology, the frontend will implement a Node.js package manager to manage dependencies.

Pros:

- npm: manage dependencies and allow the use of third party libraries in our project
- it can be used for server-side rendering with Next.js, which can improve performance

Cons:

- Node.js is a newer platform, so not all libraries are available yet

Front-end design alternatives: One of the drawbacks of Next.js is its steeper learning curve. An alternative platform is Angular since it is more beginner-friendly. However,

since the previous version of the application was built in React, this should simplify the migration to Next.js.

Back-end:

NodeJS: The existing project is already using NodeJS for the backend. We looked into other backend frameworks but given that the frontend will be done in React it makes the most sense to continue to use NodeJS. Tons of different libraries already exist.

Pros:

- Fast Execution with V8 Engine: Node.js is powered by Google's V8 engine, which compiles JavaScript into machine code, making execution fast. This speed is beneficial for handling high I/O-bound operations and real-time applications.
- Asynchronous and Event-Driven: Node.js follows an event-driven, non-blocking I/O model, meaning it can handle multiple requests concurrently without blocking. This makes it suitable for highly concurrent applications, like chat servers or live streaming.

Postman: Can continue to use automated scripts in Postman

Pros:

- User-Friendly Interface: Intuitive GUI that makes creating and managing API requests easy.
- Built-in Testing: Allows you to write tests in JavaScript directly within the application.

Cons:

- Limited to API Testing: Primarily focused on testing APIs, so it's not suitable for broader backend application testing.
- Dependency on GUI: While there are command-line options (Newman), the primary functionality is GUI-based, which can be less efficient for some users.

Jest:

Pros:

- Rich Feature Set: Provides a complete solution for unit and integration testing with built-in assertion libraries.
- Fast and Simple: Runs tests in parallel and has a straightforward API, making it easy to start.
- Snapshot Testing: A unique snapshot feature allows you to track changes in API responses or components over time.

Cons:

- JavaScript Focused: Primarily designed for JavaScript and TypeScript applications, so not suitable for other programming languages.
- Learning Curve: While simple for basic tests, more complex features (like mocks) can have a learning curve.

Cloud:

GitHub and GitHub Actions

- Pros: Automated CI/CD; integrated with GitHub; customizable workflows.
- Cons: Limited free usage; complex setup for advanced workflows; potential vendor lock-in.

Node.js Backend

- Pros: JavaScript across the stack; scalable; extensive package ecosystem.
- Cons: Single-threaded limitations; higher memory usage; potential for callback hell.

AWS Secrets Manager

- Pros: Secure credential management; automated secret rotation; easy AWS integration.
- Cons: Additional costs; slight latency; added complexity for small apps.

Prisma ORM

- Pros: Type-safe queries; simplifies database access; schema management.
- Cons: Learning curve; potential performance overhead; limited flexibility for complex queries.

Amazon RDS (MySQL)

- Pros: Managed service with automated backups; scalable.
- Cons: Costs; limited customization; potential latency.

NextAuth (Authentication)

- Pros: Supports multiple providers; secure by default; integrates well with Next.js.
- Cons: Learning curve; mainly for Next.js; limited customization for complex flows.

Amazon ECS

- Pros: Scalable; managed container orchestration; AWS integration.
- Cons: Complex setup; costs increase with scale; AWS-specific, limiting portability.

4.5 Design Analysis

The backend team has developed a system to support the functionality of Program user's editing and modifying the program's survey. This entailed creating a CSV file with all the data needed to create the default version of the survey. We then created a python script to parse the CSV file, allowing us to export an updated version of the survey to our database, which will then be displayed on our web application. Getting this functionality to work has allowed us to disregard any notion of redesigning screens for the admin to edit questions. Instead, this design allows us to provide the functionality of updating the survey without having to take the time to create a complex user interface for admins and instead focus front-end resources on the student experience. This feature of the design will impact the design of our AWS server. Knowing that we will have to execute a Python script means that we will need a Python interpreter and several Python libraries on the server on top of all of the node packages.