3 Project Plan

3.1 Project Management/Tracking Procedures

We are using the waterfall method for our current work because we are trying to get the old groups prototype working. Once we reach that point, we will adopt the agile method and create actual stories for each group member to work on for each iteration.

3.2 Task Decomposition

Cloud:

- AWS ECS
    - Setup ECS cluster and define service for Node.js Server
    - Configure the ECS task to run the Node.js server
    - Expose Ports on ECS
    - Integrate ECS with GitHub actions
    - Set up auto-scaling and load balancing
- AWS RDS(MY SQL)
    - Configure/Provision RDS instance with MYSQL
    - Set up security groups to allow connections from ECS / specific Ips
    - Define IAM roles for ECS to allow access to RDS through Prisma
    - Configure and set up SSL for api encryption
    - Finizalze Connection between RDS and ECS
- Github / Github Actions
    - Set up Github Repository for CI/CD (need to wait on ETG)
    - Lockdown main branch
    - Setup CI/CD workflows in GitHub actions to automate ECs deployment for Node.js server
    - Define Secrets and Envs for workflows
    - Configure workflows to build, test, and deploy on pushes to the main branch

Frontend:

- Finalize Figma design
    - present to the University of Iowa to get feedback
- Go into the old codebase and diagnose + fix the errors
- Copy the fixed codebase
    - rework it to utilize Chakra UI and React
    - implement our figma designs
- Refactor API calls and logic from the old codebase
    - refactoring for Next.js server-side rendering
    - refactor the forms with Formik for better form state control

Backend

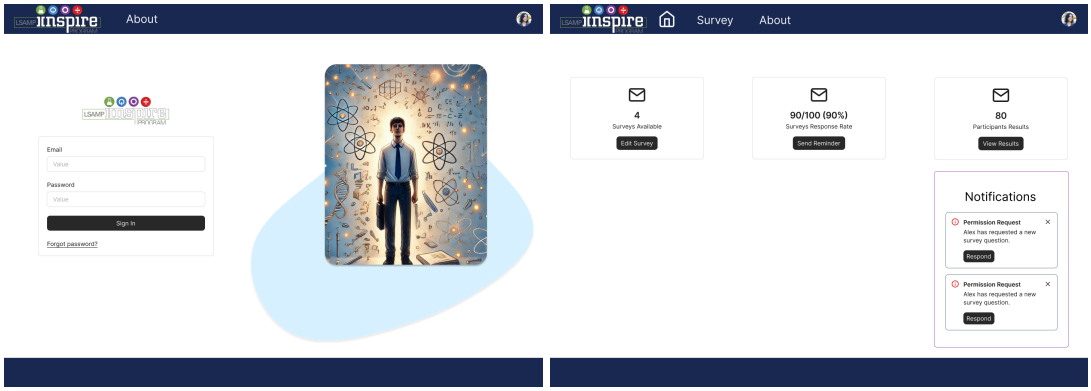
- Metrics For Data Results
    - Create better results for data collection for the admins/ researchers.
    - Averages, response rates, medians, and any other metrics that they may want to know
- Test Coverage

- Increase testing for functions in the backend to improve code coverage.
- Can continue using automated scripts in Postman or switch to Jest, another popular node js backend testing framework.
- Have the potential option to integrate API tests with the CI/CD pipeline

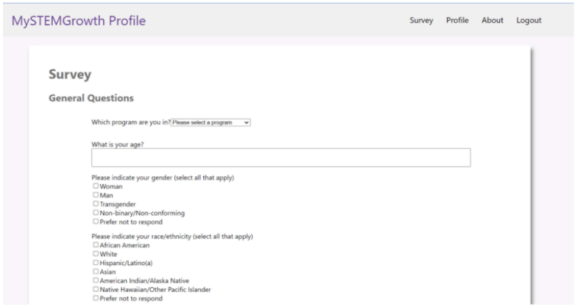
- Database Usage
  - Use pgAdmin in SQL to manage user information and other data storage.
  - Could use a different Non-relational database, but the current architecture is already in MySQL.

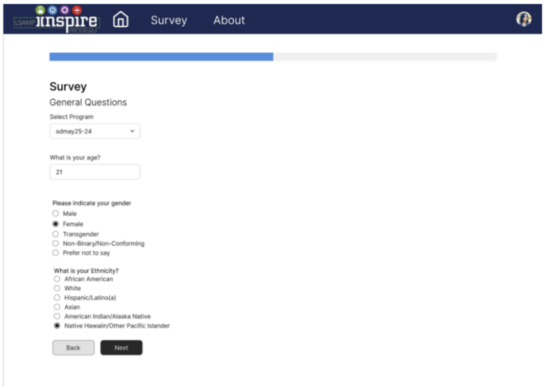
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Full UI redesign that is more appealing for users

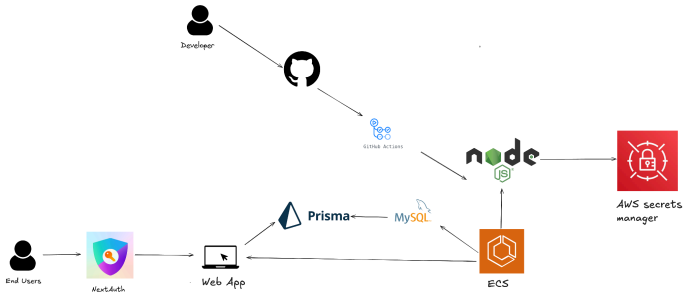


- Modular survey system

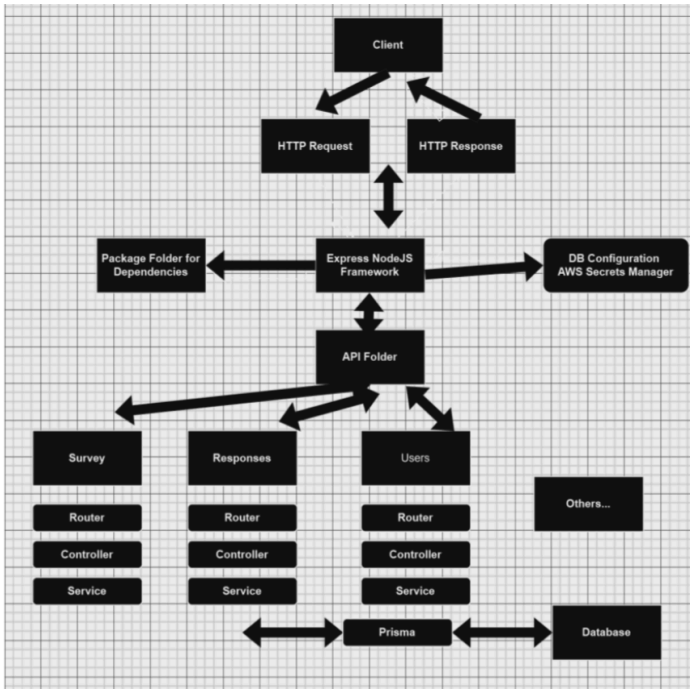




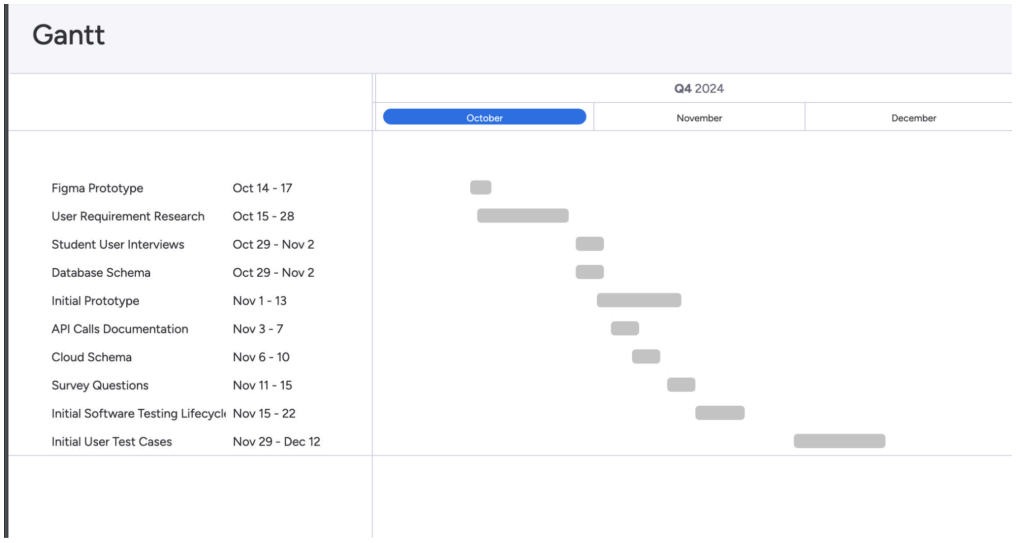
- Cloud Security System



- Backend Structure Rework



3.4 Project Timeline/Schedule



Gantt chart breakdown for the rest of semester 1

3.5 Risks And Risk Management/Mitigation

Scope Creep(.6): requirements from clients are not set in stone, and additional features i.e, custom surveys have not been planned out in stone

Plan: Focus on building the core application first, i.e., remaking the first iteration to its completion point, then move on to extra features. Also, we will be able to switch over the cloud developers to any needed role after the core infrastructure is set up.

Project-based Dependencies(.4): different sections of the project rely on other parts to be completed before being able to be started. For example, backend development needs the RDS database to be set up before starting to work with Prisma, and the RDS database needs to communicate with the ECS to test in production.

Timeline slippage(.8): significant project goals i.e. completed figma design, next js app foundation, cannot be fully time blocked as we have such a large team it is impossible to set concrete dates for large features. Also delays can happen in the development/testing phases.

Plan: Break significant goals into smaller goals for each member individually and set concrete dates. Use project management software (and our project manager) to track progress against milestones. Add extra time for delays and have tasks available for members who get done quickly.

Unexpected Costs/ Costs outside Budget(.2) : with ECS being available to scale, costs may randomly shoot up or exceed expected budget during times of extensive use like, for example, testing the application or large groups of survey participants.

3.6 Personnel Effort Requirements

Task	**Average Time Per Person (Hours per week)**
User Login	3
Participant Home Screen	2
Admin Dashboard Screen	2
Participant Survey Page	3
Admin Edit Survey Page	4
Participant Results Page	4
Admin Results Page	4
User Information Storage	4

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 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

- IDEs and Programs
    - 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
    - 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
    - NextAuth for username/password security using OAuth
    - Prisma toolkit for easy access when working with our databases

- Project Management Tool: Jira 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.