GAINS REQUIREMENTS DOCUMENT

Team members: Jonathan Gallo, Murat Yildiz, Yaroslav, Aiden, Braden, Kevin

CSC 230 Project – Requirements Final

Overview

- Stack: Next.js with React (frontend), Node.js with Express (backend), MongoDB (database).
- R and RStudio used for writing and testing R code.

Purpose

- Build a web app for running R analyses through a simple interface for social science students.
- Allow social science students to easily generate statistical regression outputs using their own inputted data.
- Allow social science students to view how the R code works and the context.

Functional Requirements

- **Must Have:** Users can paste or upload R code and submit it for execution.
- **Must Have**: Users can upload datasets and associate them with their jobs.
- **Should Have:** Users can login and view their account/job history.
- **Must Have**: System stores job history, status, and outputs for review and stores in database.
- **Must Have:** Results (tables, text, plots) can be viewed or downloaded.

Non-Functional Requirements

- Reliable and secure with clear error handling.
- Easy to use, with a simple and responsive interface.
- Reasonable performance with jobs completed in a short time.
- Accessible through common web browsers (Chrome, Safari, etc.).
- Accessible to students with disabilities (ADA compliance).
- System should be able to support multiple users concurrently.

User Stories

- As a student, I want to input data and receive linear regressions and the corresponding R code so that I can understand how the analysis works.
- As a student, I want to download my results so that I can include them in my assignments or research.
- As an instructor, I want to review outputs and logs to verify correctness so I can fairly evaluate students' work.
- As a team member, I want a clear interface to collaborate effectively.

System Architecture

- Next.js with React (CSS Styling) provides the user interface for submitting and viewing jobs.
- Express with Node.js manages requests, runs R code, and tracks jobs.
- MongoDB stores job data, parameters, and references to outputs.

Execution Strategy

- R code is executed through Rscript in the backend (/server folder).
- Outputs such as tables and plots are captured and returned to the frontend.

Milestones

- Scaffold frontend and backend setup.
- Create all required documents (Collab, Requirements, High Level Design, and Detailed Design) **[Benchmark 1].**
- Get temporary working frontend.
- Create temporary JavaScript backend code (Node.js/Express placeholder for R execution)
- Finalize polish and prepare for demo. **[Benchmark 2].**
- Add database via MongoDB for data storage.
- Login/Account system.
- Overall refactoring and optimization.
- Full integration (database, login, polished UI), final testing, deployment, and presentation prep. **[Benchmark 3].**

Definition of Done

- Users can submit R code with data and view accurate results.
- Outputs are saved, downloadable, and persist in MongoDB via account systems.
- Interface is clear, reliable, and meets all functional requirements.
- All major user stories are satisfied and verified through testing.