Software Requirements Specification

for

HCGBST Tracking System

Version 0.1 approved

Prepared by:
Rachel Brower
Blake Culbertson
Enrique Lopez
Chan Rain
Garret Willis

Cal Poly Humboldt

March 8-9, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Creation	3/8/25	Created the initial document	.01

1. Introduction

1.1 Purpose

HCGBST is a system meant for storing scholarships, as well as their applications to streamline the application process for students and simplify the application processing for the organization. The system also acts as a database to manage and use the applications and quickly access information for applicants.

1.2 Document Conventions

The document follows the number scheme 1.1, 1.2, 1.3 to help anyone reading follow the flow of the document

1.3 Intended Audience and Reading Suggestions

The intended audience for this document is for the employees of the organization *Humboldt County Farm Bureau* and acts as the documentation for use of the software.

1.4 Product Scope

The scope of this product is to allow for the storage of scholarships offered by the *Humboldt County Farm Bureau* and act as the application portal for their provided scholarships. Additionally, the portal functions as a space to moderate the scholarships and applicants by the organization.

2. Overall Description

2.1 Product Perspective

This product is the application portal and tracker for the organization, it is to be associated with the organization itself, and acts in tandem with the websites homepage

2.2 Product Functions

- Note Detection
  - The notes played by the user will be detected and logged for an overall score

2.3 User Classes and Characteristics

The main user class for this product will be students applying to scholarships

2.4 Operating Environment

The product will be created using html, css, javascript, mongodb, and react.

2.5 Design and Implementation Constraints

The sole constraint of this project is the time limit imposed by the nature of the Cal Poly Humboldt Hackathon.

2.6 User Documentation

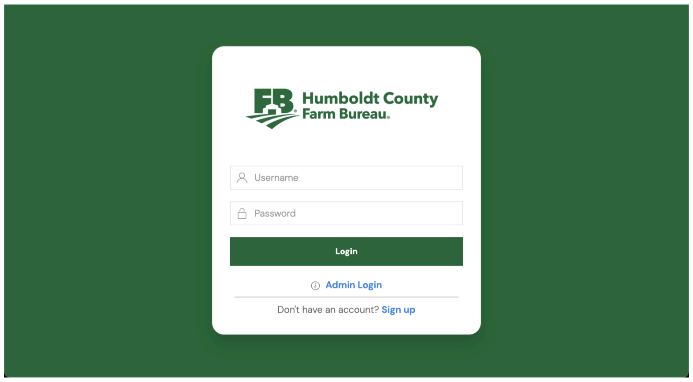
Outside documentation will be provided to allow for the upkeep of the program. This will include documentation regarding the frontend, backend, and database

2.7 Assumptions and Dependencies

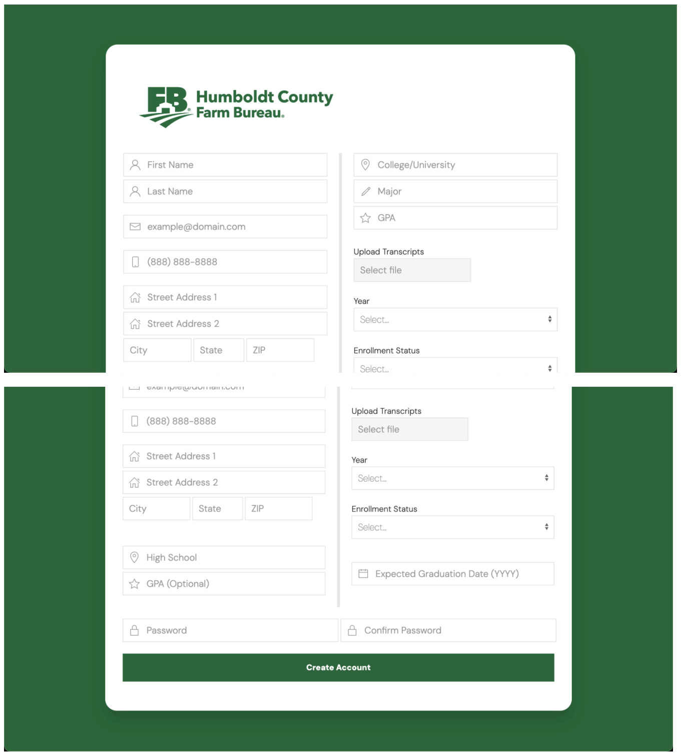
Unaware of any at this time.

3. External Interface Requirements

3.1 User Interfaces



- Example of log-in page, with admin login link, sign-up links



- Example of sign-up page

3.2 Hardware Interfaces

No hardware interfaces at this time

3.3 Software Interfaces

there will be a connection from the front-end and the back-end of the product. The front-end will be the game itself while the back-end will be the database storing user information and tracking performance over time.

3.4 Communications Interfaces

This product will require a web browser and an internet connection in order to properly function

4. System Features

4.1 Typing In Time With Rhythm

4.1.1 Description and Priority:

Student:

- Game gives player a musical typing task and they must type words in time with the rhythm of a song
- Priority is high

Admin:

- Admin can view the player statistics for each song and game-setting
- Priority is low

4.1.2 Stimulus/Response Sequences:

Player:

- User chooses the song they want to play and the difficulty level, clicks the “Start song” button with the typing option selected(rather than math)
- New typing song starts
- The words that the user has to type are visually indicated with the song’s rhythm and the word’s movement across the screen indicates the time that the word should be typed by
- User types the words in rhythm and gets points for it, failure to type a word in time loses points
- Song ends, shows song summary(how the user did overall, where they messed up, etc) and adds data to user profile
- User either clicks “Replay” button to type the same song with the same game settings or “Back to menu” button to change the game settings

Admin:

- Admin chooses the song and game settings they want to see the data for
- Information is presented as a web page

4.1.3 Functional Requirements:

- P0: Basic Scholarship Database
  - P0.1: Schema for storing scholarship details (name, amount, criteria)
  - P0.2: CRUD (Create, Read, Update, Delete) operations for managing scholarships

4.2 Displaying User Data To Show Improvement

4.2.1 Description and Priority:

Player:

- Player can see their stored game statistics over time on their profile and compare to other players
- Priority is medium

Admin:

- Admin can see the overall game statistics for all players
- Priority is low

4.2.2 Stimulus/Response Sequences:

Player:

- Player selects "My Profile"/"My Games"(undecided) button on the web page and gets directed to a webpage containing their game statistics in the default time frame(overall)
- Player chooses different timeframes and the webpage changes accordingly
- Player can choose to show the highscores of other players as well as average scores for each statistic, to compare information

Admin:

- Admin selects the button to show player statistics, and is directed to a web document containing the overall player information across several different timeframes

4.2.3 Functional Requirements:

- REQ-1: Player should be able to view personal game statistics over different time frames
- REQ-2: Player should be able to compare their statistics to the statistics of others
- REQ-3: An admin should be able to view the game statistics and player information for all players, collected into one place and in an easily sellable form

4.3 User Account Creation/Login

4.3.1 Description and Priority:

Player:

- Player is able to create and login to an account
- Priority is medium

4.3.2 Stimulus/Response Sequences:

Player:

- Player clicks login or create account
- Player either enters login information or information to create an account
- Player is then logged into that account registered to them

4.3.3 Functional Requirements:

- REQ-1:Players should quickly and seamlessly login/create an account
- REQ-2: Players should be notified if they did not enter correct information and could not be logged into account

5. Other Nonfunctional Requirements

5.1: Low Latency

5.1.1: As a Player, I want low latency when it comes to playing the game so that my input registers as immediately as possible. This ensures that any input follows the math input of the game at the correct moments of the rhythm.

5.2: Frame Rate Stability

5.2.1: As a Player, I want my frame rate to be as stable as possible, roughly around the 60 Frames Per Second range to prevent any disruptions when it comes to the gameplay.

5.3: Audio Synchronization

5.3.1: As a player, I want the audio and visual elements of the game to be perfectly synchronized within 5 milliseconds, so that the notes and beats appear and sound in perfect alignment, ensuring the rhythm and timing are accurate.

5.4: Customizability

5.4.1: As a player, I want the game to allow me to calibrate my controller and screen settings for audio and input timing, so that I can adjust for any personal hardware or setup delays and play at my best.

5.5: Accuracy of Math Problems

5.5.1: As a player, I want the game to correctly evaluate my math problem inputs with zero tolerance for calculation errors, so that I can trust my performance is being assessed fairly and accurately.

5.6: Real-Time Feedback of Math Problems

5.6.1: As a player, I want the game to provide immediate feedback (within 100 milliseconds) on whether my math problem input is correct or incorrect, so that I can stay in sync with the game's rhythm.

5.7: Tolerance of Keyboard Errors

5.7.1: As a player, I want the game to handle minor input mistakes (e.g., accidental key presses or missed characters) without crashing or freezing, so that I can continue playing even if I make a small error.

5.8: Real-Time Scoring

5.8.1: As a player, I want the score to update immediately after each input without delay, so that I can see my progress in real-time.

5.9: Simple Start Menu

5.9.1: As a player, I want the start menu to have a simple and intuitive design, so that I can easily navigate and access game options without confusion or delay.

5.10: Error Handling

5.10.1: As a player, I want the game to handle errors gracefully, providing clear and helpful error messages and allowing me to continue playing or restart without losing progress.

5.11: Load Time

5.11.1: As a player, I want the game to load and start within 10 seconds of launching, so that I can quickly get into the gameplay without long waiting times.

5.12: In-game Tooltips

5.12.1: As a player, I want the tooltips to be clear, concise, and not obstructive to gameplay, so that I can easily understand their content without hindering my view or gameplay.

5.13: Readability of Math Problems

5.13.1: As a player, I want math problems and instructions to be presented clearly and legibly within the rhythm game, with appropriate spacing and formatting, so that I can solve them quickly and without confusion.

6. Appendix

6.1 Team Roles

- **Lead Technical Designer: Rachel Brower**
- **Technical Designer: Chan Rain**
- **Security Specializa: Blake Culbertson**
- **Quality Assurance: Enrique Lopez**
- **Team/Backend Manager: Garret Willis**