# Software Requirements Specification

**for**

# Status Reporting System

**Version 1.2 approved**

**Prepared by Team 5**

**CSCD 488**

**March 14, 2023**

# Table of Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| Shrimp | 2/13/23 | Initial Creation | 1.0 |
| Calamari | 3/14/23 | System Features and Feedback Changes | 1.1 |
| Tempura | 6/7/23 | Implementation Alterations | 1.2 |

# 1. Introduction

## 1.1 Purpose

The existing status reporting system from the Shelby website is out-of-date and requires an update to modern standards. Currently, the website can function but is becoming more unmanageable as it starts to show its age. The current user interface is complex and unclear, making it difficult for students to submit weekly reports. This standalone solution will entirely replace the status reporting system component of the Shelby website using modern frameworks and features, allowing for more effective error messages and interface design to help students complete their status reports. The new status reporting system will also give instructors and admins a more effortless way to create and manage active courses.

## 1.2 Document Conventions

IEEE,
Markdown (for design),
Markdown with mermaid extension (a diagramming tool) (for UML/diagrams),
Cargo doc (Rust standard for documenting code)
NOTE: As the system will be programmed in rust, rustdoc may be more appropriate.

## 1.3 Intended Audience and Reading Suggestions

There will be three documents released:

1. Development and software analysis documents will help guide development.
- Audience: Development Team
2. Design proposals will be for the client to better understand the product under development and determine viability.
- Audience: Client
3. Documentation to guide users, which will be updated after each deployment.
- Audience: User

## 1.4 Product Scope

This product will be a standalone component that will entirely replace the existing status reporting system within Shelby. The client wants a new status reporting system that utilizes new features from modern web frameworks to improve the use experience. Currently, the status reporting system doesn't give clear error messages to students when they fill out their report forms, causing confusion. The new system will provide improved error messages for students when they fill out their report forms. Ideally, students should also be able to view prior reports when they have completed their report forms. Instructors and admins also need an interface to manage and view active courses - such as an email or a downloadable file. A new database, form submission system, login system, email system, CSV parser, and several admin features will be necessary to satisfy all requirements.

### 1.5 References

The GitHub private repository contains all of the documentation
github.com/tztz8/EWU-CSCD488-490-Senior-Project
(Use this github pages website to view our repo):
tftinker.tech/EWU-CSCD488-490-Senior-Project/)
User Stories:
tftinker.tech/EWU-CSCD488-490-Senior-Project/Doc/UserStories/userStory.html
Requirements:
tftinker.tech/EWU-CSCD488-490-Senior-Project/Doc/Requirements/requirements.html
Info about the old system
https://www.tftinker.tech/EWU-CSCD488-490-Senior-Project/Doc/OldSystem/

## 2. Overall Description

### 2.1 Product Perspective

The status reporting system is a component of Shelby - a course management system made by the client. The status reporting component, in particular, allows students and teams to create a weekly status report to document project progression. However, the Shelby website was made many years ago and cannot satisfy new requirements. To meet the new requirements, the new status reporting system, as a self-contained product, will replace the existing status reporting system using new features from modern frameworks. However, the Shelby course management system will remain operational since it offers much more functionality than status reports.

### 2.2 Product Functions

1. Students and teams shall complete their weekly status report forms.
2. Students shall view prior status reports.
3. Instructors shall manage class rosters and teams.
4. Admins shall manage the web-server.
5. The system shall send emails.

### 2.3 User Classes and Characteristics

1. Students
- Students are the most frequent and important users of the status reporting system. Students will document their project progression using the status reports. They will also view prior status report submissions.
2. Instructors
- Instructors will view student status report submissions for grading and analysis. Instructors will also manage the class roster and teams for all their classes by uploading CSVs.
3. Admins
- The admins of the website will be responsible for maintaining the system and creating classes for instructors. As superusers, admins will need all permissions.

## 2.4 Operating Environment

A Linux-based web-server hosted by EWU. The web-server must operate on the same hardware as the existing Shelby course management system.

## 2.5 Design and Implementation Constraints

- The web-server must not run on port 80 or 443. The Shelby website will run on the same hardware and occupy port 80. If port 443 is available, some web browsers will not permit connections to port 80, which eliminates both options.
- The web-server must be on (Amazon Linux 2 AMI) due to the operating environment.
- The database must be MySQL as requested by the client.
- Virtualization is not an option (such as docker).
- EWU SSO is not an option for the login system. However, a workaround by filtering allowed emails to the ewu.edu domain may be a viable option.
- CSV file uploads and parsing must be used when creating and updating class rosters.
- CSV file uploads and parsing must be used when creating and updating project teams.

## 2.6 User Documentation

- In-depth documentation can be found within the repository.
- Students should be able to use the interface without the need for documentation.
- Instructors will receive documentation on how to create and update their class rosters, how to create and update project teams, and how to view student reports.
- Admins will receive documentation on how to set up and manage the web-server within the operating environment and how to create classes for instructors.

## 2.7 Assumptions and Dependencies

1. MySQL database
- A highly versatile and well-supported solution that will not restrict development. The client will manipulate and access the MySQL database manually with an IDE.
2. Linux environment
- A highly versatile and well-supported solution that will not restrict development.
3. Rust Language
- A modern programming language with well-supported web development frameworks and libraries. This will constrain us to only the frameworks and libraries supported by rust.
4. Actix (a rust crate, no external dependencies required)
- Rust's standard and most popular REST API web-server backend.
5. Diesel (a rust crate, no external dependencies required)
- A rust database connection management library.
6. env_logger (a rust crate, no external dependencies required)
- The standard universal rust logging library .
7. Yew (a rust crate, no external dependencies required)
- The most popular and well-maintained frontend web framework made with rust.
8. config (a rust crate, no external dependencies required)
- The standard and most popular rust library for reading config files. It supports reading from nearly all config file formats.

# 3. External Interface Requirements

## 3.1 User Interfaces

The website will function as a single all-encompassing user interface. All users will authenticate via a login interface to gain access. After authentication, users will proceed to a selection menu listing all classes they participate in or manage. Admins and instructors will have a separate page and interface for modifying class rosters and project teams. Instructors and admins will upload CSV files to update class rosters and project teams. Admins will also have an additional interface for creating classes. Upon selecting a course, students and instructors will view the weekly tasks organized in a calendar-based format, similar to the existing Shelby course management system. Students will then access and fill out their weekly status report forms. Students will also receive helpful error messages if any form fields are incomplete at submission. Instructors will also receive emails confirming any status report submissions.

## 3.2 Hardware Interfaces

The status reporting system will be a web application. Any device that is capable of accessing and interacting with the web will act as an interface (chrome base browsers and firefox will be tested). Otherwise, the website will not need to communicate with any other hardware interfaces.

## 3.3 Software Interfaces

- There will be two dynamic web pages. Students will access a single dynamic web page to complete their reports. Admins and instructors will share an isolated dynamic web page for managing the web-server and classes.
- MySQL for the database will be used to store the reports, login information, and logs.
- Linux will be for the operating system.
- Rust will be for the programming language and its web application libraries.
- Yew will be for the front end, which will enable the use of dynamic web pages and communicate with the backend. The frontend will send completed status report forms to the backend for verification and submission. The frontend will retrieve prior status report submissions from the backend for viewing. The frontend will communicate with the backend to authenticate users.
- Actix will be for the backend, which will communicate with the front end and any other web components.
- Diesel will be for communicating to and from the database. The backend will interact with diesel to manage the database connection.
- env_logger will be for the logging system. The backend will interact with env_logger and the database to log events.
- config will be used to read the config file (Ex login for db server and where to log to) to run the server.

## 3.4 Communications Interfaces

Chrome and Firefox will be officially supported - indirectly supporting most other web browsers. The status reporting system will not directly interact with the existing email system and instead use SMTP to send the emails. The status reporting system will share the same hardware as the Shelby system, forcing the web-server to use non-standard ports (instead of ports 80 and 443).

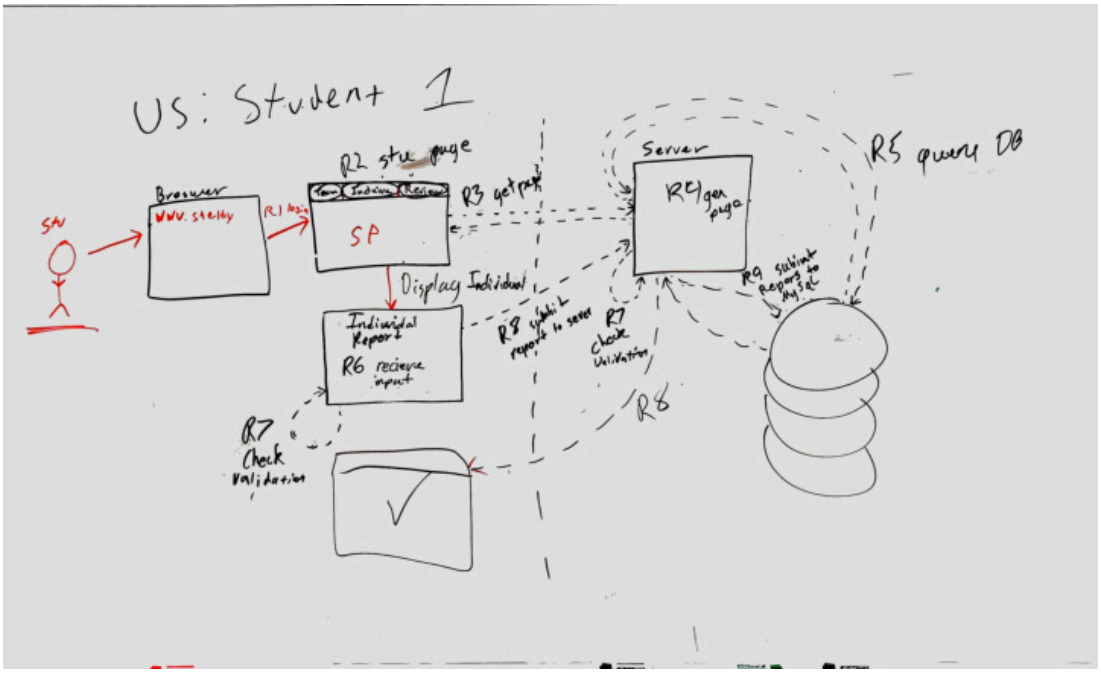
# 4. System Features

## 4.1 Student 1: Individual Reports

### 4.1.1 Description and Priority

- Students shall submit their individual reports by answering the questions on the status report forms. The individual status report forms will include questions for existing tasks from prior status reports, as well as for new tasks. The forms will update dynamically according to valid and invalid inputs to form fields.
- As the primary function of the new status reporting system, this is a high-priority feature.

### 4.1.2 Stimulus/Response Sequences




### 4.1.3 Functional Requirements

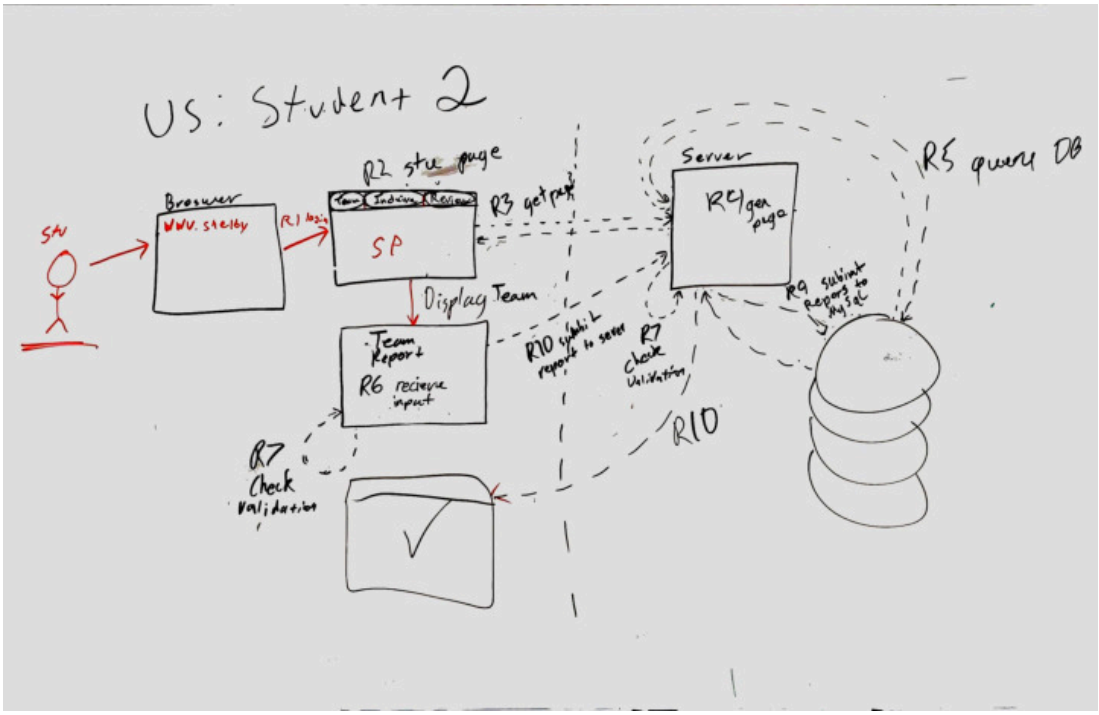
REQ-1: Students and teams shall complete their weekly status report forms.

## 4.2 Student 2: Team Reports

### 4.2.1 Description and Priority

- Students, as groups, shall submit their team reports by answering the questions on the status report form. Only one student from each team submits the weekly team report.
- As one of the primary functions of the status reporting system, this is a high-priority feature.

### 4.2.2 Stimulus/Response Sequences




### 4.2.3 Functional Requirements

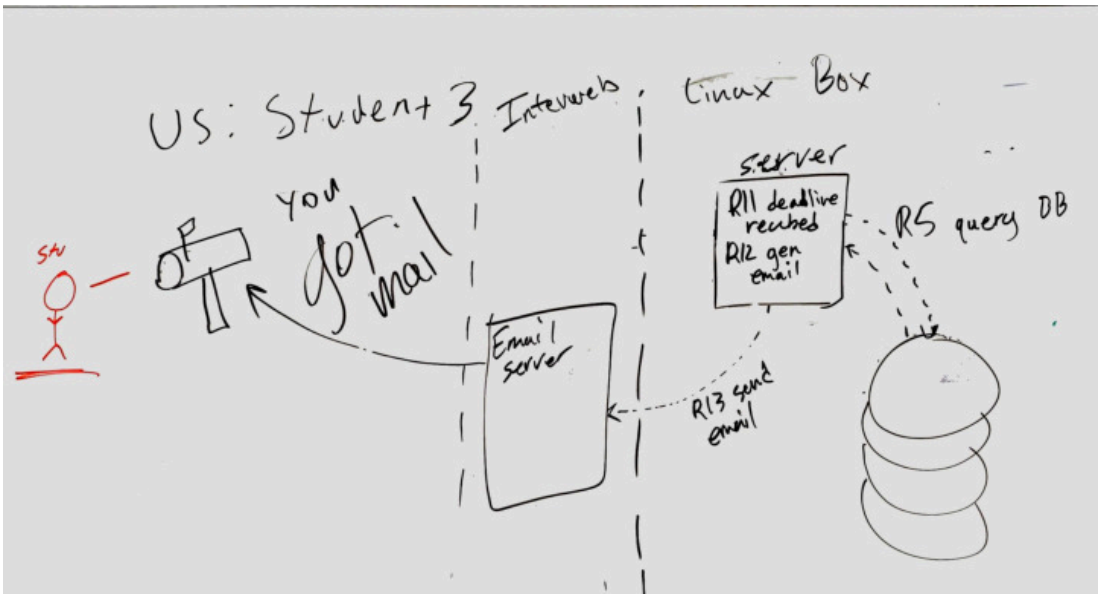
REQ-1: Students and teams shall complete their weekly status report forms.

## 4.3 Student 3: Email Confirmation of Status Report Submissions

### 4.3.1 Description and Priority

- Students shall receive emails confirming status report submissions.
- The emails will act as evidence of submission, making this a high-priority feature.

### 4.3.2 Stimulus/Response Sequences




### 4.3.3 Functional Requirements

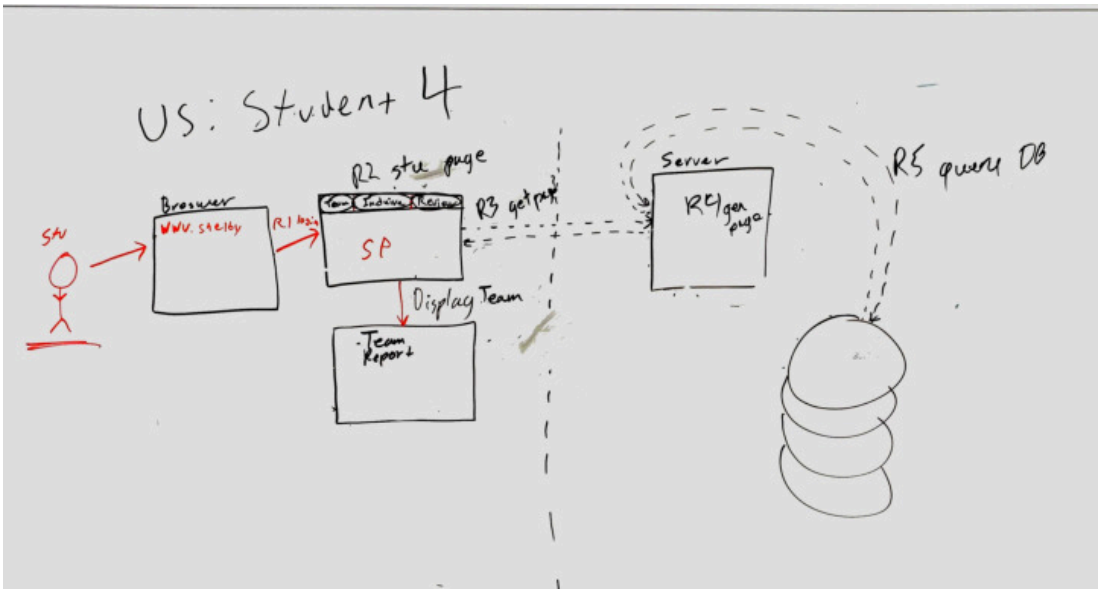
REQ-5: The system shall send emails.

## 4.4 Student 4: Viewing Prior Submissions

### 4.4.1 Description and Priority

- Students shall view their old reports and progress. Viewing old reports is necessary for creating new status reports since tasks will persist between status reports.
- As a necessary component of individual status reports, this is a high-priority feature.

### 4.4.2 Stimulus/Response Sequences




### 4.4.3 Functional Requirements

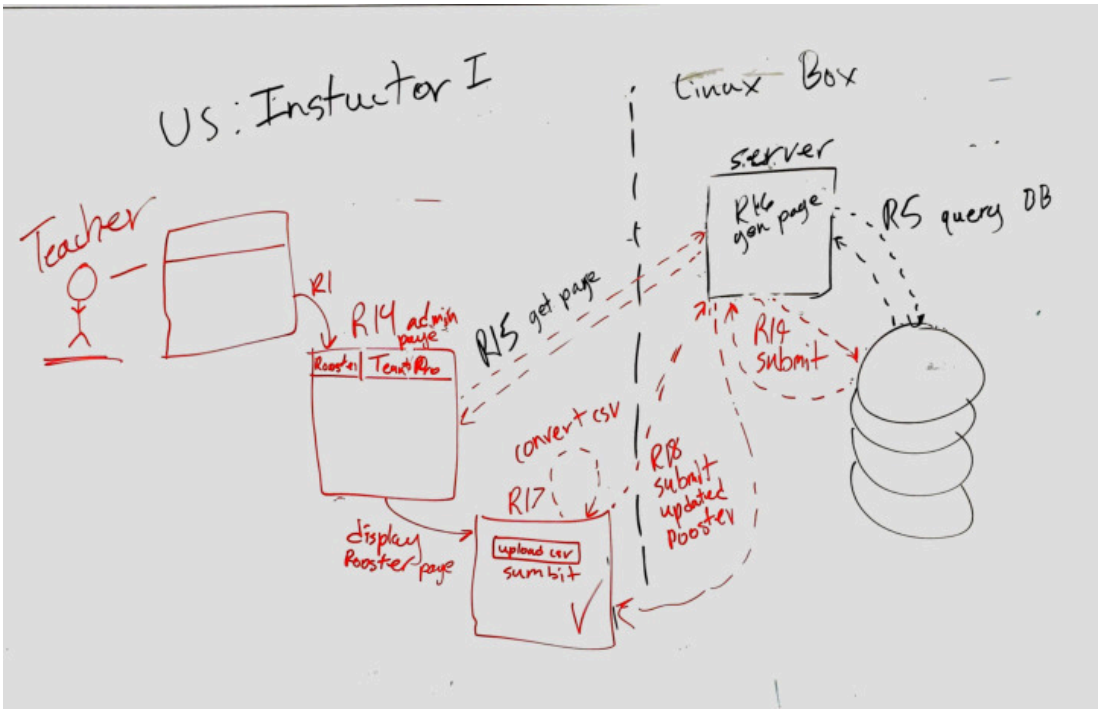
REQ-2: Students shall view prior status reports.

## 4.5 Instructor 1: Managing Class Roster

### 4.5.1 Description and Priority

- Instructors shall create and update the class roster by uploading a formatted CSV.
- This is required for the creation of student accounts and for the website to function, making it a high-priority feature.

### 4.5.2 Stimulus/Response Sequences




### 4.5.3 Functional Requirements

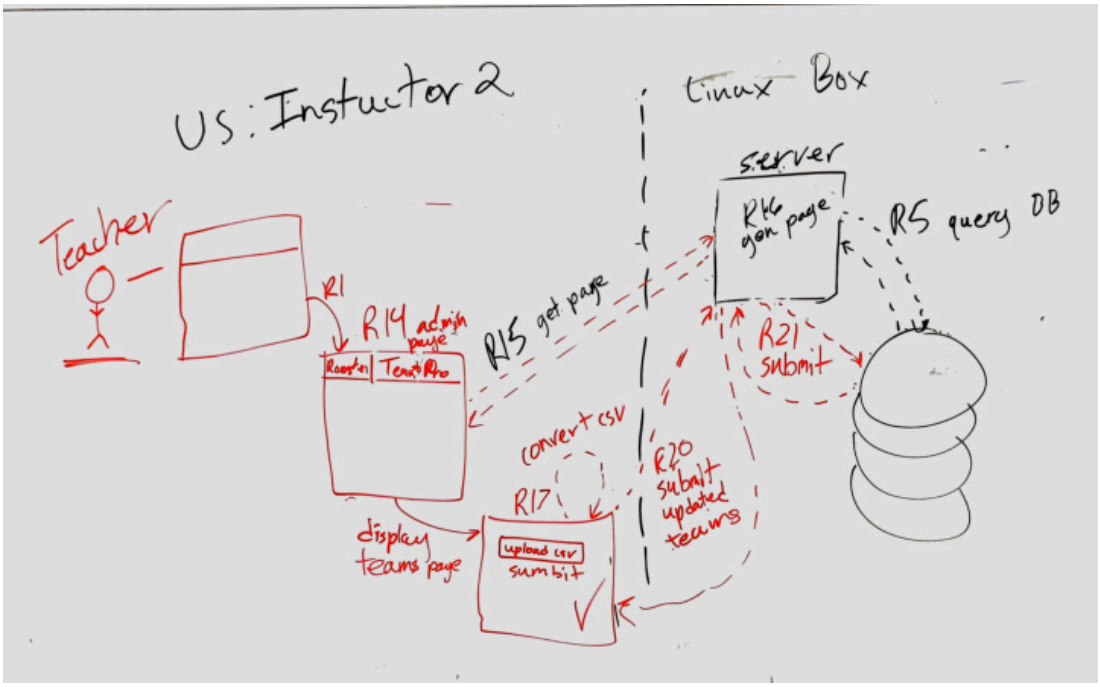
REQ-3: Instructors shall manage the class roster and teams.

## 4.6 Instructor 2: Managing Project Teams

### 4.6.1 Description and Priority

- Instructors shall group students into teams by uploading a formatted CSV. This is required for the creation of teams within the system and for the website to function, making it a high-priority feature.

### 4.6.2 Stimulus/Response Sequences




### 4.6.3 Functional Requirements

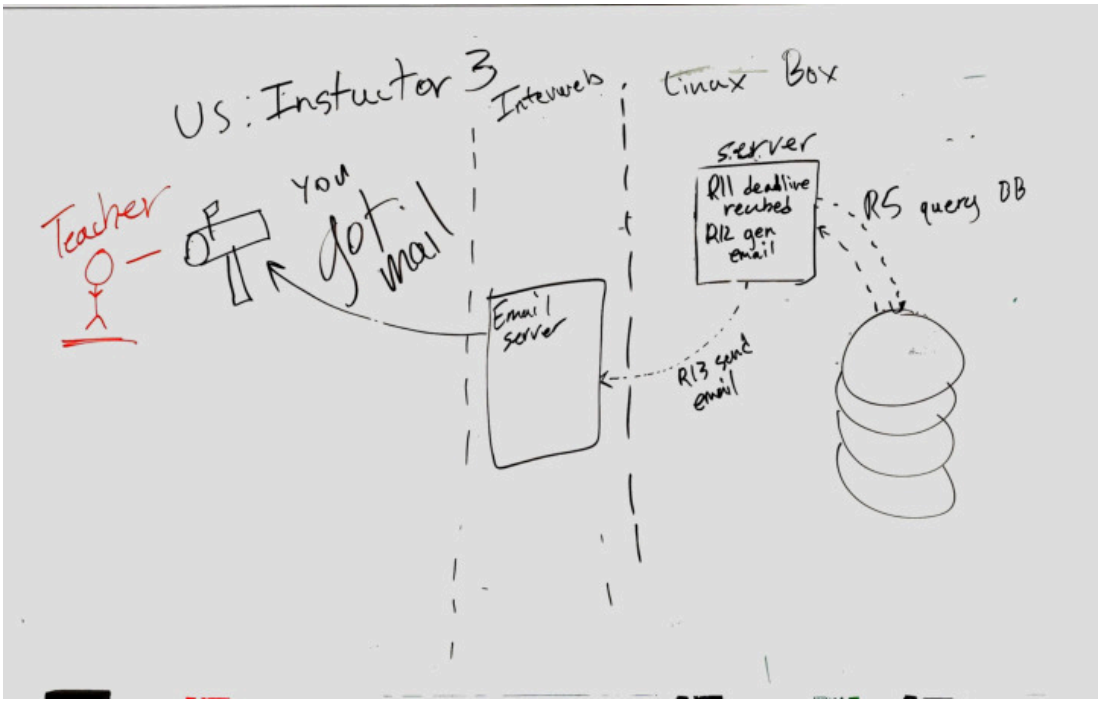
REQ-3: Instructors shall manage the class roster and teams.

## 4.7 Instructor 3: Email Confirmation of Status Report Submissions

### 4.7.1 Description and Priority

- Instructors shall receive emails confirming status report submissions. Instructors will use the emails when reviewing the status report submissions.
- The emails will act as evidence of submission, making this a high-priority feature.

### 4.7.2 Stimulus/Response Sequences




### 4.7.3 Functional Requirements

REQ-5: The web-server shall send emails.

## 4.8 Admin 1: Managing the Web-Server

### 4.8.1 Description and Priority

- Admins shall set up and manage the web-server on their hardware. They will need to configure the emailing and oath systems and manage releases. Documentation will be given on setting up the web-server.
- The product must be deployable, making this a high-priority feature.

### 4.8.2 Stimulus/Response Sequences


### 4.8.3 Functional Requirements

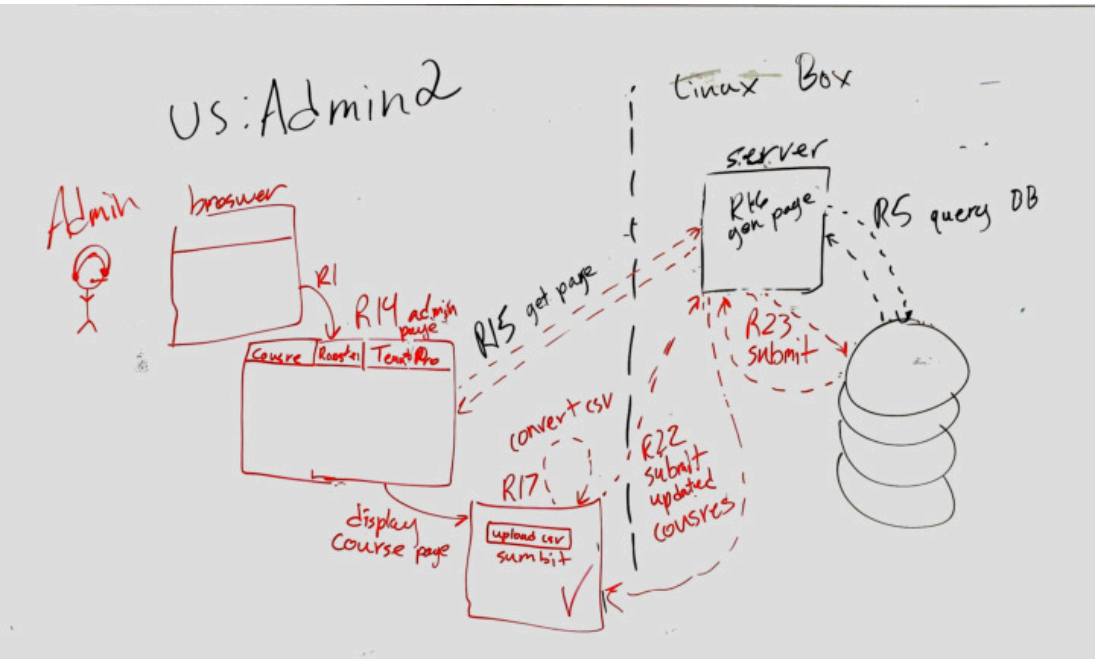
REQ-4: Admins shall manage the web-server.

## 4.9 Admin 2: Managing Courses

### 4.9.1 Description and Priority

- Admins shall create and update the active courses.
- Courses must exist before instructors can begin managing them, making this a high-priority feature.

### 4.9.2 Stimulus/Response Sequences




### 4.9.3 Functional Requirements

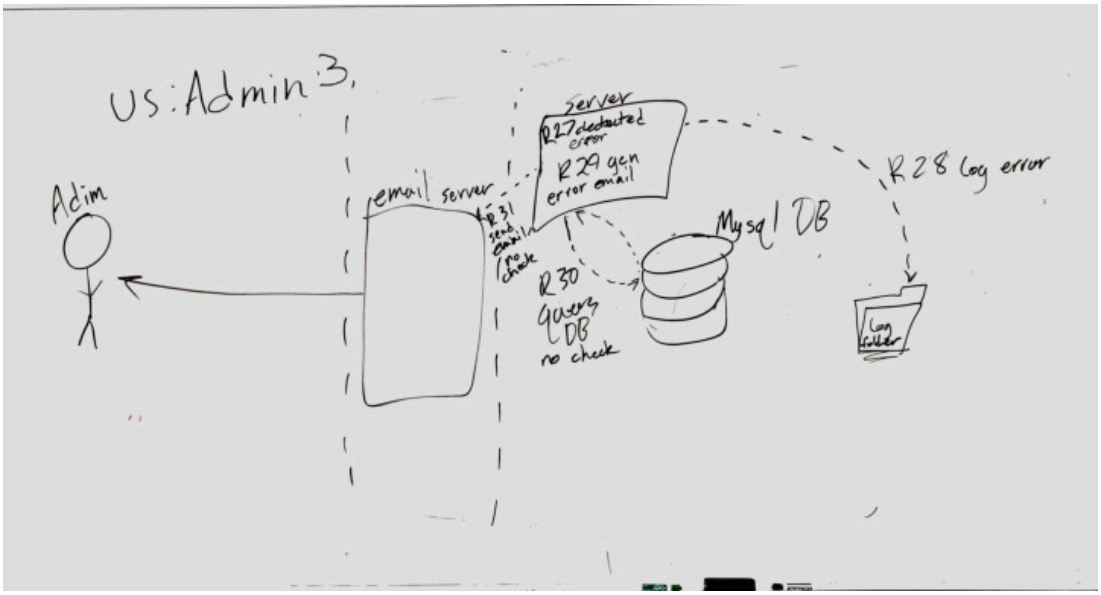
REQ-4: Admins shall manage the web-server.

## 4.10 Admin 3: Emails Reporting Web-Server Errors

### 4.10.1 Description and Priority

- Admins shall receive emails reporting any web-server errors, such as crashes.
- Emails reporting web-server errors will be crucial in maintaining the website, making this a medium-priority feature.

### 4.10.2 Stimulus/Response Sequences




### 4.10.3 Functional Requirements

REQ-5: The web-server shall send emails.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

The status reporting system will be a standalone web-server sharing the same hardware as the existing Shelby course management system. Due to using shared hardware, the system should ideally be as light and robust as possible. Using Actix for the Rust backend will far surpass the necessary performance requirements.

## 5.2 Safety Requirements

N/A

## 5.3 Security Requirements

The status reporting system will behave as a course management system within an educational setting. Harm could occur to students if private information, such as grades, is accessible by unauthorized actors. The login system should be robust to prevent any damages and ensure no unauthorized actors can access private information.

## 5.4 Software Quality Attributes

The client is primarily concerned about the ease of improving and maintaining the new status reporting system. A seamless integration with the current Shelby website is also preferred.

## 5.5 Business Rules

N/A

# 6. Other Requirements

N/A

# Appendix A: Glossary

**CSV:** Comma separated values files store data in a table structured format.

**Framework:** Application programming interfaces that provide all necessary components to build a complete application.

**Library:** Application programming interfaces that provide everything necessary to build an individual component or system within an application.

**IDE:** Integrated development environments are application building programs that provide a single unified graphical interface to help developers code software.

**IEEE:** Institute of Electrical and Electronics Engineers, professional association for electronics engineering

**Markdown:** Markup language that enables advanced formatting of text within a plain text document. The status reporting system will include documentation utilizing markdown.

**Mermaid:** A markdown-derived diagramming and charting tool. The status reporting system will include documentation utilizing mermaid.

**MySQL:** A database server created and maintained by Oracle.

**MySQL workbench:** A visual database design tool that integrates SQL development.

**Parse:** Analyzing a sentence or word into its parts and describing their syntactic roles. The status reporting system will parse CSV files.

**REST API:** Representational state transfer is an architectural style for the web. It is the formal web architecture for all modern web applications.

**Shelby**: The website created by the client as a course management system, which helps run the class.

**SMTP:** Simple mail transfer protocol is a data transmission format used to send and receive emails.

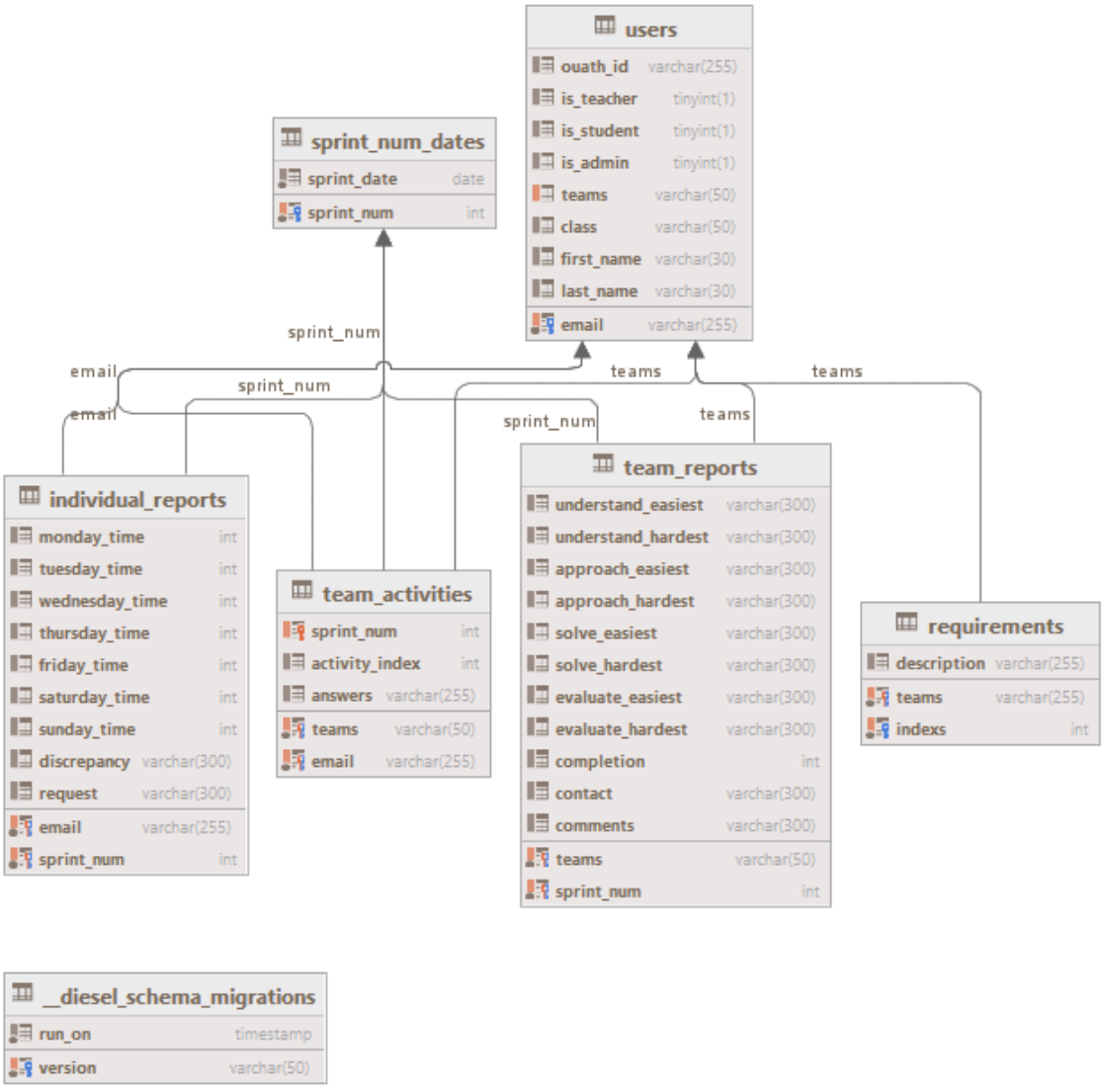
**Superuser:** Also known as root user on Linux - the administrator.

# Appendix B: Analysis Models

## API Docs:

github.com/tztz8/EWU-CSCD488-490-Senior-Project/blob/main/Doc/API/APIDOC.md

## Database Schema:




## User Stories:

User Story Diagrams

# Appendix C: To Be Determined List

1. Admin panel file uploads