Emergency Class Manager: Requirements Document

By: Katie Hoskins, Tanya Peacock, Marcus Strange

Table of Contents

Project Name

Our website is for schools or school districts that want to use a simple website instead of a basic spreadsheet to keep track of students in emergencies. It allows for teachers and administrators to manage their class which is why we came up with the name **Emergency Class Manager**.

Team Member Names

See **Figure 1** in the appendix for a detailed description.

Abstract

The Emergency Class Manager project will allow teachers to have a centralized space to keep track of their students in emergency situations. Teachers will have the ability to access the website on mobile or on a computer. They will have a place to check off students as present, missing, or absent and then submit that information to the administration. This will be a more streamlined process for teachers as well as administrators to keep track of students. The reports will be sent to the administrator at the end of the drill or emergency. When the need arises, this will be a much simpler and more effective way of accounting for students.

Tools and Technologies

See **Figure 2** in the appendix for a detailed description.

Requirements List

1. Home Page
    - 1.1. The home page will consist of three main components:
        - 1.1.1. An information box explaining the Emergency Class Manager
        - 1.1.2. A Sign-in button
            - 1.1.2.1. On click, this button will send the user to the sign-in page
        - 1.1.3. A Register button
            - 1.1.3.1. On click, this button will send the user to the registration page
2. Registration Form
    - 2.1. The registration form will be one component with the following:
        - 2.1.1. A hamburger-style pop-in with a clickable button for Sign-in and Registration.
        - 2.1.2. A label for First Name
        - 2.1.3. A box for First Name
            - 2.1.3.1. The box will accept a first name of at least two letters

2.1.3.1.1. On invalid input, the user will be notified that there must be at least two letters in the first name
2.1.3.2. The box will only accept characters
2.1.3.2.1. On invalid input, the user will be notified that there may only be characters used in the first name
2.1.3.3. The box will be marked as required with an *
2.1.4. A label for Last Name
2.1.5. A box for Last Name
2.1.5.1. The box will accept a last name of at least two letters
2.1.5.1.1. On invalid input, the user will be notified that there must be at least two letters in the last name
2.1.5.2. The box will only accept characters
2.1.5.2.1. On invalid input, the user will be notified that there may only be characters used in the last name
2.1.5.3. The box will be marked as required using *
2.1.6. A label for an email address
2.1.7. An input box for an email address
2.1.7.1. The box will accept a valid email address through the school system
2.1.7.1.1. Emails must contain an @ symbol
2.1.7.1.2. The email must end in the domain cmcss.net
2.1.7.1.3. The email will be verified by the server
2.1.7.2. The box will be marked as required using *
2.1.8. A button labeled “Create Account”
2.1.8.1. On click, the email is checked to make sure an account with that email has not already been created
2.1.8.1.1. If there is an existing account with that email, a message will be shown in a display box
2.1.8.1.1.1. The message will say “Email already taken”
2.1.8.1.2. If there is no existing account, no message will be displayed
2.1.8.2. If any of the fields are left empty, a dialog box will show containing the message, “Please fill out the required fields”
2.1.8.3. If all required fields are filled out, no message will be output
2.1.8.3.1. All information will be submitted to the back-end servers
2.1.8.3.2. The user will be logged into the account
2.1.8.4. The application will get sent to the dashboard.

3. Sign-in Form
    3.1. The sign-in form will be one component with the following:
        3.1.1. A hamburger-style pop-in with a clickable button for Sign-in and Registration.
        3.1.2. A label for an email address
        3.1.3. An input box for the email address
            3.1.3.1. The user input will be verified by the server
            3.1.3.2. If the user submits an unverified email, a dialog box will pop up that states “Could not find the email”
            3.1.3.3. If the user submits a verified email, no dialog box will pop up
            3.1.3.4. The user will be sent a link to sign in and verify the account via magic link.
4. Dialog Box
    4.1. Dialog box will have an “Okay” button that will close the dialog box
    4.2. Dialog box will provide and explain the error or confirm changes to page
5. Logo
    5.1. Homepage
        5.1.1. The logo button will send the user back to the homepage of the website
    5.2. On log in
        5.2.1. The logo button will send the user back to the dashboard of the website
6. Dashboard
    6.1. The dashboard will consist of a three main components:
        6.1.1. A logo
            6.1.1.1. On click, the logo will take the user back to the dashboard on all pages.
        6.1.2. A navigation bar
        6.1.3. A box for the week’s upcoming planned events
7. Navigation Bar - Hamburger
    7.1. The navigation bar will consist of the main components for teachers and administrators:
        7.1.1. The Account button
            7.1.1.1. On click, the Account button will take the user to the “Account” page
        7.1.2. The Roster button
            7.1.2.1. On click, the Roster button will take the user to the “Roster” page
                7.1.2.1.1. On click, the user will be sent to their student list
        7.1.3. Events button
            7.1.3.1. Teacher
                7.1.3.1.1. On click, the user will see a calendar with events they can click to see more info.

7.1.3.2. Admin
7.1.3.2.1. On click, the user will be sent to a page with an implemented calendar where they can edit and add events.
7.1.4. Contact Information button
7.1.4.1. On click, the Contact Information button will take the user to the "Contact Information" page

8. Roster

8.1. The Roster page will consist of three main components:
8.1.1. A table label of "Roster"
8.1.2. A dropdown field that has the class name for selection
8.1.2.1. Each class is its own selection of students and allows multiple periods that a teacher might be teaching.
8.1.2.2. The last class selected is remembered for future page loading.
8.1.3. A search bar for the table to search for students
8.1.4. A table with the columns, Student ID, Last Name, First Name, Present status
8.1.5. Search bar that will search criteria: Student ID, Last Name, or First Name
8.1.6. Table will be able to do sort ascending or descending per header of: Student ID, Last Name, First Name.
8.1.7. A dropdown box in the Present column of the table.
8.1.7.1. Button Selection will be a drop down with P, A, M, and V
8.1.8. A check button for All Present?
8.1.8.1. This will set all student Present statuses to P
8.1.9. The last row is empty, which allows searching for students not on the roster to add them to the roster for visiting students.
8.1.9.1. Auto-fill by student id, and search for student id with both Last and First Name.
8.1.9.2. "Add" button will add the row to the list and adds a remove button to the newly created row.
8.1.9.3. Status will automatically be set to present.
8.1.10. A "Submit" button
8.1.10.1. On click, the roster will get submitted to the database and sent to administration.
8.1.10.2. The user will be prompted are they sure before submitting

9. Past Event Roster

9.1. The page is loaded with URL-encoded data for the roster which is generated by the events page.
9.2. The Past Event Roster page will have four main components:
9.2.1. A table label of Event Name
9.2.2. A dropdown field that has the class name for selection

9.2.2.1. Each class is its own selection of students and allows multiple periods that a teacher might be teaching.
9.2.2.2. The last class selected is remembered for future page loading.
9.2.3. A search bar for the table to search for students
9.2.3.1. Search criteria: Student ID, Last Name, or First Name
9.2.4. A table with the columns, Student ID, Last Name, First Name, Present status
9.2.5. Table will be able to do sort ascending or descending per header of: Student ID, Last Name, First Name.

10. Events

10.1. The Events page will consist of multiple components:
10.1.1. A calendar implementation
10.1.2. A modal for event and roster information
10.1.2.1. Event name
10.1.2.2. Search bar
10.1.2.3. Count
10.1.2.3.1. For teachers, this will hold 0/1 or 1/1 based on if they had submitted a roster for this event.
10.1.2.3.2. For administration, this will hold the number of submitted rosters compared to the number of teachers in the system.
10.1.2.4. Table with Class ID, Class Name, and Teacher
10.1.2.4.1. All inputs are able to be sorted by ascending or descending.
10.1.2.4.2. For administration, there will be an “Open” button where they can view the roster submitted by that teacher
10.1.3. A modal for extra event information
10.1.3.1. This modal is editable for administration but not editable for teachers
10.1.3.1.1. All editable items have an “Edit” button next to them for administration.
10.1.3.1.2. There will be a submit button at the bottom to save changes.
10.1.3.1.2.1. The user will be prompted if they are sure before submitting
10.1.3.2. This will provide the event name, time, and date of the event

11. Contact Information

11.1. The Contact Information page will consist of two components:
11.1.1. The phone number for the administrators
11.1.2. The email for the administrators

12. Classes - Administration

12.1. The Classes page will be made up of multiple components consisting of:

12.1.1. A search box

12.1.2. A table with Class ID, Class Name, and Teacher

12.1.2.1. All fields will be sortable by ascending and descending

12.1.3. An “Add Class” button

12.1.3.1. This button will open a modal for adding a class

13. Add Class Modal

13.1. This modal will contain a box for Class ID, Class Name, and Teacher input

13.2. A “Submit” button

13.2.1. This button will commit changes to the “Classes” table

13.2.2. The user will be prompted with a dialog box.

13.2.3. “Edit” buttons

13.2.3.1. These buttons will open a modal for editing a class asking if they are sure they want to submit

14. Edit Classes Modal

14.1. The edit class modal consists of:

14.1.1. A table with Class Name, Teacher, Student ID, Last Name, and First Name

14.1.1.1. All columns are able to be sorted by ascending or descending

14.1.1.2. This will hold information about the classes with all students and teachers in the table.

14.1.2. An “Add Row” button

14.1.2.1. This button will add a row to the table to add a new student

14.1.3. An “Add Teacher” button

14.1.3.1. This button will open a new modal for adding a teacher.

14.1.3.1.1. This will have Email, Name, and Class ID

14.1.3.1.1.1. All emails must have an “@” symbol

14.1.3.1.1.2. All names must be at least two letters and only letters

14.1.3.1.1.3. Class ID must consist of letters or numbers

14.1.3.1.2. A “Submit” button

14.1.3.1.2.1. On click, this data will be updated in the database.

14.1.3.1.2.2. The user will be prompted are they sure before submitting

15. Admin Panel - Administration

15.1. The Admin Panel will consist of multiple components consisting of:

15.1.1. A search bar

15.1.2. A table consisting of Email, Last Name, and First Name

15.1.2.1. All items can be sorted by ascending or descending

15.1.3. An “Add User” button

15.1.3.1. This button will open a modal for adding a new user to the table

16. Add User Modal

16.1. This modal consists of four main parts:

16.1.1. Email label

16.1.2. Email box

16.1.2.1. Email must consist of an "@" symbol

16.1.2.2. Email is required

16.1.3. Last Name label

16.1.4. Last Name box

16.1.4.1. Last name must contain at least two letters

16.1.4.2. Last name must contain only letters

16.1.5. First Name label

16.1.6. First Name box

16.1.6.1. First name must contain at least two letters

16.1.6.2. First name must contain only letters

16.1.7. A "Submit" button

16.1.7.1. On click, the information is submitted to the database and added to the Admin Panel

16.1.8. A check box for "Admin"

16.1.8.1. This is available only for Super Admin

16.1.8.2. On click, the user is set to administration and given the correct perms.

16.1.8.3. The user will be prompted if they are sure before submitting

17. Database

17.1. The database will consist of 6 tables:

17.1.1. Users

17.1.1.1. This table will hold userEmail, fName, lName, phoneNum, emergcyInfo, admin, superAdmin, and removed

17.1.1.1.1. userEmail is the primary key and char

17.1.1.1.2. fName and lName are both char

17.1.1.1.2.1. Cannot be null

17.1.1.1.2.2. Can have at max 100 characters

17.1.1.1.3. phoneNum is int

17.1.1.1.4. emergcyInfo is char

17.1.1.1.4.1. Can have at max 100 characters

17.1.1.1.5. Admin, superAdmin, and removed are bool

17.1.1.1.6. This information is saved for the classes

17.1.1.1.7. Zero to Many relationship with Classes

17.1.2. Classes

17.1.2.1. This table will hold class_id, className, teachers, and removed

17.1.2.1.1. Class_id is the primary key and int
17.1.2.1.2. className is char
17.1.2.1.2.1. Can have at max 100 characters
17.1.2.1.3. Teachers is array
17.1.2.1.4. Removed is bool
17.1.2.1.5. This information is saved for Classes and Roster
17.1.2.1.6. Zero to Many relationship with Users
17.1.2.1.7. One to Many relationship with Perm Roster

17.1.3. Perm Roster
17.1.3.1. This table will hold class_id and student_id
17.1.3.1.1. Class_id is a foreign key and int
17.1.3.1.1.1. Cannot be null
17.1.3.1.2. Student_id is another foreign key and int
17.1.3.1.2.1. Cannot be null
17.1.3.1.3. One to One relationship with Classes
17.1.3.1.4. Many to Many relationship with students

17.1.4. Students
17.1.4.1. This table will hold student_id, fName, lName, and removed
17.1.4.1.1. Student_id is the primary key and int
17.1.4.1.2. fName is char
17.1.4.1.2.1. Can have at max 100 characters
17.1.4.1.2.2. Cannot be null
17.1.4.1.3. lName is char
17.1.4.1.3.1. Can have at max 100 characters
17.1.4.1.3.2. Cannot be null
17.1.4.1.4. Removed is bool
17.1.4.1.5. Many to Many relationship with Perm Roster

17.1.5. Events
17.1.5.1. This table will hold event_id, date, scheduleTime, eventName, and description
17.1.5.1.1. Event_id is the primary key and int
17.1.5.1.1.1. Cannot be null
17.1.5.1.2. Date is data type date
17.1.5.1.2.1. Cannot be null
17.1.5.1.3. scheduledTime is data type date
17.1.5.1.4. eventName is char
17.1.5.1.4.1. Can have max of 200 characters
17.1.5.1.4.2. Cannot be null
17.1.5.1.5. Description is char
17.1.5.1.5.1. Can have max of 5000 characters

17.1.5.1.6. Zero to One relationship with Event Roster

17.1.6. Event Roster

17.1.6.1. This table will hold class_id, student_id, status, date, and submitted

17.1.6.1.1. Class_id is a foreign key and int

17.1.6.1.1.1. Cannot be null

17.1.6.1.2. Student_id is another foreign key and int

17.1.6.1.2.1. Cannot be null

17.1.6.1.3. Status is char

17.1.6.1.3.1. Can have max of one character

17.1.6.1.4. Date is data type date

17.1.6.1.5. Submitted is bool

17.1.6.1.6. One to One relationship with Events

Updated Timeline

See **Figure 3** in the appendix for a detailed description.

Appendix

Figure 1: Team Member Names

Name	**Role**
Katie Hoskins	Developer
Tanya Peacock	Developer
Marcus Strange	System Admin

Figure 2: Tools and Technologies

Place in System	**Dependency Name**	**Dependency Type**	**Why?**
Frontend	Vue.js	Frontend	Vue.js is a collection of premade web dev templates and uses Nuxt.js which is built on top of React
Data Source	Supabase	Database	We need a strong database for our project and this seems like the most reasonable one to use for our needs.

Figure 3: Updated Schedule

Week 1: August 28 - September 1	**All Members:** Project Proposal, discovering tools and technologies to use for the project, discussion about requirements, meeting with team and discussing strengths of team members.
Week 2: September 4 - September 8	**Sep 4: Labor Day - No class** **All Members:** Finish project proposal, discover requirements for the project, continue talking about tools and technologies available for the project and think about requirements.
Week 3: September 11 - September 15	**All Members:** Work on and finish requirements document, collaborate on what would be possible for the project, work on and finish requirements powerpoint.
Week 4: September 18 - September 22	**Katie:** Look into and learn more about the tools and technologies being used including Vue.js, Nuxt, React, and Supabase, start tinkering around with the technologies. **Tanya:** Research and learn more about the resources we are looking at using. **Marcus:** Provide insight in the tools and teach core practices in building the systems needed in the project.
Week 5: September 25 - September 29	**Katie:** Work on making website mobile friendly and working on the correct application, get GitHub created. **Tanya:** Start on the Registration Form.

	Marcus: Start on Registration backend, and setup the proper services in Supabase
Week 6: October 2 - October 6	**Katie:** Navigation bar, some frontend design. **Tanya:** Polish the Registration form, make sure all the buttons work like they should and labels line up with boxes. **Marcus:** Write out the documentation on Registration and start on admin controls with some frontend design.
Week 7: October 9 - October 13	**Oct. 9-10 - Fall Break No Class** **Katie:** Working with roster set up for teachers and designing classes for the teachers. **Tanya:** Learn about how to use the database and get the registration form to save to where it needs to. **Marcus:** Setup teacher's table with class structure.
Week 8: October 16 - October 20	**Katie:** Continue work on the roster setup for teachers. **Tanya:** Assist where needed. **Marcus:** Setup class's table with student's structure

Week 9: October 23 - October 27	**Katie:** Work in some frontend design for the website including making clickable icons and making it user friendly. **Tanya:** Assist where needed. **Marcus:** Setup student's table and setup rules for creating/editing students.
Week 10: October 30 - November 3	**Katie:** Help work on admin panel and discuss what it should be used for. **Tanya:** Assist where needed. **Marcus:** Setup process rules for events. Possibly complete admin process.
Week 11: November 6 - November 10	**Katie:** Continue work on admin panel and communication with teachers. **Tanya:** Assist where needed. **Marcus:** Help implement push notifications of events.
Week 12: November 13 - November 17	**Katie:** Work on archives for previous reports being sent to administration. **Tanya:** Assist where needed. **Marcus:**
	Katie:

Week 13: November 20 - November 24	Continue work on archives, start finishing up code and polishing frontend items. **Tanya:** Assist where needed. **Marcus:**
Week 14: November 27 - December 1	**All Members:** Finish up things we are unclear about and work on the project board.
Week 15: December 4 - December 8	**All Members:** Clean up code and finish submitting all items, finish project board.
Week 16: December 11 - December 15	**Finals Week** **All Members:** Present our fully functioning project.

This timeline is subject to change. *