PROJECT PROPOSAL

EIT Emergency Device Management System
ITPR7.508 Business Application Programming

Team Member	Roles	Signature	Date
Liam Palmers	Team Leader Lead Designer Developer		04/08/2024
Alex Scott	Team Leader Lead Architect Database Admin		04/08/2024
James Sadler	Developer		04/08/2024
Aidan Willis	Frontend Developer Tester		04/08/2024
Joe Yin	Backend Developer, Tester		04/08/2024

Version 1.0

Revision Date: 11/08/2024

Contents

Executive Summary

The Eastern Institute of Technology (EIT) is a tertiary education provider that needs a digital solution to manage and track emergency devices on its campus efficiently.

Given the institution's size and the necessity to comply with safety regulations, the implementation of a centralized system is crucial for ensuring that all devices are regularly inspected, maintained, and replaced when necessary.

Problem Statement

EIT currently relies on an Excel spreadsheet to track fire extinguishers, which is insufficient for their needs. Jonathan Bixley, the Facilities Operations Coordinator, is tasked with ensuring that each fire extinguisher is inspected four times annually, following a detailed checklist.

The current system makes it difficult to track these inspections and maintain up-to-date records, potentially leading to safety risks. Fire extinguishers have a lifespan of five years, so keeping track of their replacement is crucial. The limitations of the spreadsheet highlight the need for a more robust digital solution to ensure compliance and safety across the campus.

Solution Overview

The proposed solution is a web application specifically designed to manage and track emergency devices across the EIT campus.

This application will include a user-friendly interface to display device information, as well as features to add, delete, and update device records. Users will be able to search for specific devices, and the system will differentiate between user and admin accounts to control access to certain functions. Automated notifications will be sent to alert users of upcoming device expirations and required maintenance. Additionally, the application will include an interactive map to visually track device locations, enhancing the overall management and safety of the campus.

Requirements Analysis

Functional Requirements:

- The system must allow users to log in with a username and password.
- The system must support both regular users and administrators.
- Administrators must be able to add, update, and delete device information.
- The system must allow entering details like device type, location, manufacture date, and size.
- Users must be able to search for devices by different criteria (e.g., type, location, status).
- The system should allow filtering of devices based on maintenance or expiry status.
- The system must show real-time notifications on the admin dashboard for upcoming maintenance or device expiry.
- Administrators should be able to configure notification settings.
- The system must be easy to update with new device types, like fire hydrants or med kits, without needing big changes to the code.
- The system must include a user manual that explains how to use the application, accessible from the user interface.

Non-Functional Requirements:

- The system must keep data accurate and consistent across all actions.
- The system must be able to grow to handle more types of emergency devices.
- The system must support more users and data without slowing down significantly.
- The system must use secure connections (e.g., HTTPS) for data transmission.
- The system must have access controls to ensure only authorized users can change data.
- The system should work on different devices, including desktops, tablets, and smartphones, with a design that adjusts to different screen sizes.
- User passwords must be securely encrypted.
- The user interface must be simple and easy to use.

Technical Approach

Programming Language:

The application will be developed using Go (Golang), chosen for its performance, concurrency support, and robust standard library, which is well-suited for building scalable web applications. (The Go Programming Language, 2024)

Web Framework and Templating:

The front end will utilize the Go standard library "html/template" (tmpl) package for server-side rendering of HTML templates, providing a secure and efficient way to generate dynamic content. (Template Package - Text/Template - Go Packages, 2024)

jQuery will be used for DOM manipulation, event handling, and AJAX requests to enhance the interactivity of the application. (OpenJS Foundation - openjsf.org, 2024)

htmx will be incorporated to allow seamless, client-side interactions with the server, such as fetching and rendering partial HTML updates without requiring full-page reloads. (Htmx - High Power Tools for Html, 2024)

Database Architecture:

PostgreSQL was chosen for its easy integration with Go, reliability, and support for complex queries and relational data. (PostgreSQL: Documentation, 2024)

The pq driver will be used, a pure Go PostgreSQL driver for Go's database/sql package. (Lib/Pq: Pure Go Postgres Driver for Database/Sql, 2023)

API Architecture:

RESTful APIs built with Go, handling business logic and database interactions. The server will provide JSON responses and serve HTML templates.

Data Access Layer:

A data access layer will be implemented using Go's database/sql package, ensuring efficient and secure interactions with PostgreSQL.

Testing

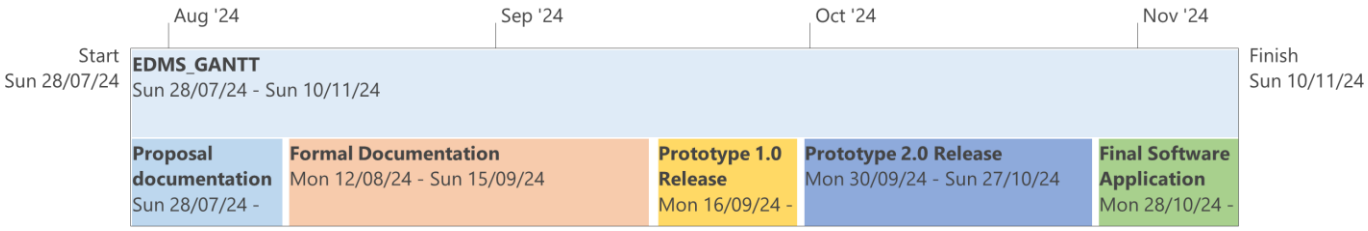
- Unit Testing: Go's "testing" package to test individual functions and methods, ensuring each component works as expected. (Testing Package - Testing - Go Packages, 2024)
- Integration Testing: Validate interactions between the frontend, backend, and database using integration tests to ensure proper system functioning.

Deployment

The current plan involves deploying the application into a single .exe file. Additionally, a separate .exe a portable version of PostgreSQL will be provided. This approach allows the application and the database to be deployed together on any suitable system without needing a separate installation of PostgreSQL.

Timeline and Milestones

Timeline



Milestones

Handover of Project Proposal:
Sun 10/11/2024

Completion of Formal Documentation:
Sun 15/09/24

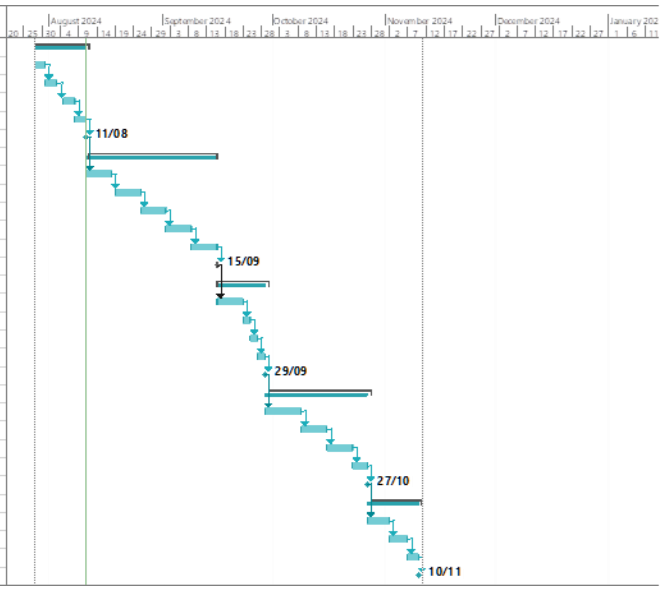
Release of Prototype 1.0:
Sun 29/09/24

Release of Prototype 2.0:
Sun 27/10/24

Release of Final Software Application:
Sun 10/11/24

Project schedule

ID	Task Name	Duration	Start	Finish	Predecessors
1	**Proposal documentation**	**15 days**	**Sun 28/07/24**	**Sun 11/08/24**	
2	Requirement Gathering	3 days	Sun 28/07/24	Tue 30/07/24	
3	Quote Preparation	3 days	Wed 31/07/24	Fri 2/08/24	2
4	Summary and Report Preparation	3 days	Mon 5/08/24	Wed 7/08/24	3
5	Proposal Finalization	3 days	Thu 8/08/24	Sat 10/08/24	4
6	Handover of Proposal documentation	0 days	Sun 11/08/24	Sun 11/08/24	5
7	**Formal Documentation**	**35 days**	**Mon 12/08/24**	**Sun 15/09/24**	
8	Business Issues Analysis	7 days	Sun 11/08/24	Sat 17/08/24	6
9	Program Specifications Analysis	7 days	Mon 19/08/24	Sun 25/08/24	8
10	Entity Relationship Diagram (ERD)	7 days	Mon 26/08/24	Sun 1/09/24	9
11	Design Documents	7 days	Mon 2/09/24	Sun 8/09/24	10
12	Test Framework and Plan	7 days	Mon 9/09/24	Sun 15/09/24	11
13	Completion of Formal Documentation	0 days	Sun 15/09/24	Sun 15/09/24	12
14	**Prototype 1.0 Release**	**14 days**	**Mon 16/09/24**	**Sun 29/09/24**	
15	Initial Development	7 days	Mon 16/09/24	Sun 22/09/24	13
16	Testing	2 days	Mon 23/09/24	Tue 24/09/24	15
17	Internal Review	2 days	Wed 25/09/24	Thu 26/09/24	16
18	Documentation	2 days	Fri 27/09/24	Sat 28/09/24	17
19	Release of Prototype 1.0	0 days	Sun 29/09/24	Sun 29/09/24	18
20	**Prototype 2.0 Release**	**28 days**	**Mon 30/09/24**	**Sun 27/10/24**	
21	Feature Enhancements	10 days	Sun 29/09/24	Tue 8/10/24	19
22	Advanced Testing	7 days	Wed 9/10/24	Tue 15/10/24	21
23	Feedback Integration	7 days	Wed 16/10/24	Tue 22/10/24	22
24	Update Documentation	4 days	Wed 23/10/24	Sat 26/10/24	23
25	Release of Prototype 2.0	0 days	Sun 27/10/24	Sun 27/10/24	24
26	**Final Software Application**	**14 days**	**Mon 28/10/24**	**Sun 10/11/24**	
27	Final Development	5 days	Sun 27/10/24	Fri 1/11/24	25
28	Comprehensive Testing	5 days	Sat 2/11/24	Wed 6/11/24	27
29	Deployment Preparation	3 days	Thu 7/11/24	Sat 9/11/24	28
30	Release of Final Software Application	0 days	Sun 10/11/24	Sun 10/11/24	29



Team and Resources

Team Member	Roles	Skills and Experience	Responsibilities
James Sadler	• Developer	Works well independently, enjoys programming, quick learner	Assist with backend development, and support in programming tasks under Alex's guidance
Alex Scott	• Team Leader • Lead Architect • Database Admin	Excels in programming, skilled and experienced in Go, extensive research, good at finding resources and libraries	Design the overall system architecture, ensure alignment with project requirements, oversee the integration of different components, provide technical guidance and direction
Joe Yin	• Backend Developer • Tester • Designer	Backend Development and design.	Contribute to backend and frontend development, support with coding and testing
Aidan Willis	• Frontend Developer • Tester	Programming and web development experience.	Develop and test frontend features, ensure user interface meets requirements
Liam Palmers	• Team Leader • Lead Designer • Developer	Frontend and backend programming experience, Good at design and modeling systems.	Oversee design and testing, coordinate team efforts, and ensure project milestones are met

Risks and Assumptions

Risks:

Deployment Uncertainty:

- Uncertainty about where and how the application will be deployed.

Maintenance:

- After the 14-week semester, there will be no one in charge of maintaining the application.

Interactive Map Integration:

- Technical difficulties in implementing and integrating an interactive map.

Technical Risks:

- Issues with technology integration and system development.

Assumptions

- The client will provide a suitable hosting solution. (local webserver or cloud-based)
- All users will have access to the necessary hardware and internet connection to use the web application.
- The client will handle the maintenance of the software after the project ends. (end of semester). We are willing to support the client with any queries regrading maintenance.

Conclusion

The proposed software solution aims to address the need for efficient management and tracking of emergency devices at EIT.

By implementing a web-based application, we will provide a centralized system that enables administrators and users to effectively monitor and maintain fire extinguishers and other safety devices across the campus.

Key benefits of the software solution include:

- **Centralized Management:** Streamlined tracking and management of emergency devices from a single platform.
- **Automated Notifications**: Automatic notifications for maintenance and expiry, ensuring devices are inspected and replaced as needed.
- **Scalability:** The application is designed to accommodate future additions of other safety devices, such as fire hydrants and first aid kits.
- **User-Friendly Interface:** Easy to use dashboard and search and filter functionality for efficient device querying.

This solution will improve on the current Excel spreadsheet and will increase, safety compliance, operational efficiency, and overall effectiveness in managing emergency devices at EIT.

References

htmx - High Power Tools for HTML. (2024). Htmx.org. https://htmx.org/

lib/pq: Pure Go Postgres Driver for Database/sql. (2023, April 26). GitHub. https://github.com/lib/pq

OpenJS Foundation - openjsf.org. (2024). jQuery. https://jquery.com/

PostgreSQL: Documentation. (2024). PostgreSQL.org. https://www.postgresql.org/docs/

template package - text/template - Go Packages. (2024). Go.dev. https://pkg.go.dev/text/template

testing package - testing - Go Packages. (2024). Go.dev. https://pkg.go.dev/testing