Project Proposal Draft

Team members: Ahmad M. Al-Qadiri, Mathias L.-Hansen, Patrik Muravski, Petar Spirovski, Adam T. Lukas, Szymon J. Nowacki, Samuel Adjabeng
Date: 26.09.2024

1. Background/Motivation

Motorized desks have become very common in offices due to health benefits e.g. better posture and comfort. Which allows us to use the data from the desks to improve health, maintenance and to optimize space through data analysis and remote control. This addresses the needs of desk users, building owners, cleaning staff etc. by analyzing user behavior, desk movement patterns, work habits.

2. Aim

Our plan is to create a distributed system with embedded elements from which we can remote control and analyze the data of multiple desks in a building through the use of the desk web API. It will allow users to adjust their desk height remotely, admins to analyze desk usage, and cleaners to automate height of a block of desks for easier cleaning. The embedded system will be able to control the desk through the board, communicate with the desk user and utilize its components e.g. Button, Screen, Potentiometer etc.

3. Objectives

To do this we've set these objectives:
1. Create a Web-App that allows the users to register or login, from which depending on their role they can control desks and visualize usage data.
2. Reliable communication between the frontend and backend which includes desk API requests and managing data storage in the database.
3. Use graph libraries to show desk statistics and trends intuitively and interactively.
4. Allow Admins to automate/schedule desk movements at set intervals in different blocks.
5. Use a Raspberry Pi Pico to control the desk through its board.

4. Solutions

1. Frontend-Backend separation:

The user interface and server-side log are built as separate components. This allows independent development, which means that frontend and backend developers can work on their parts simultaneously. Every section can now easily update and manage their code. Additionally, it is simple to scale the frontend for increased traffic without affecting the backend, and vice versa.

2. Frontend made with asp.net razor pages?

It is a framework for developing web applications on the Microsoft .NET platform. It combines HTML with C# code to create dynamic web pages. We have previous experience with C# and need to learn and gain experience with Asp.net. We do not need complicated single-page application (SPA) features for our application.

3. Graph library:

library that provides tools and functions for creating visual representations of data. Visualizing desk occupancy data (e.g., floor plans with desks, representing relationships between desks or users).

4. Timed scripts (Allowing admins to set timed actions e.g. at 12 move desks up in block B):

It is a powerful tool for automating office tasks, improving efficiency, and enhancing the overall user experience. Scheduling cleaning tasks at optimal times, avoiding disruptions to users, and ensuring that frequently used desks are cleaned more often. Additionally, maintaining a consistent overall appearance and setting desks to the same height within a given time frame.

5. Buttons for individual desk control (up, down, off, on):

User interface that can control individual desks using buttons for up, down, off, and on functionalities. To send commands to the desk control box, use the web API provided by the desk control system or a communications protocol.

6. UAC, Auth and roles:

UAC (User Account Control) manages user permissions on the system. For our application:

- As admins verify identity when accessing to control timed actions and system settings.
- As users verify identity when accessing the application.

Roles define user permissions within the application, like:

- Admin: Can set timed actions and access all data and settings.
- User: Can control their assigned desk.

7. Integration with embedded system (Raspberry pi pico w):

It is a part of our distributed application connecting board that could possibly work:

- Control the desk notification.
- Collect data from users.
- Choosing the best solution for the user based on his decision.

5. Initial requirements analysis

We've ranked the following requirements from high to low. They should be broken into smaller tasks that take maximum 1 day at a later date:

High Priority:

- Working and secure frontend and backend communication.
- Remote control for users through a web app.
- Secure database for storing desk data, user data etc.
- Visualize desk usage statistics and trends.

Medium Priority:

- Integrate embedded system (Raspberry Pi Pico) with the distributed system (Web app).

- Allow Admins to automate desk movements.

Low Priority:

- Advanced analytics like predictive alerts for maintenance
- Optimize for different device types e.g. phones and accessibility.

Keep in mind these are initial requirements, we'll mainly start with high priority and down and we'll fill in missing tasks as we go following agile practices.

6. Methods

We'll be using the following tools and methodologies:

Project Management: Our main goal is for it to be simple and understandable. Which is why we will use agile practices, specifically Kanban, where we will organize our work into 1-week sprints. Weekly meetings will be held to review progress, address issues etc. And to ensure everyone works we'll send a single sentence a day on our communication tool about what we are doing.

Version Control and Repository: Git, GitHub and their derived services is what we will use for collaboration. GitHub Projects for task management, issues (tasks and bugs) and assigning them to members. GitHub Flow (main and feature branches) is the workflow we will do for our commits, pull requests, and code review for fast development.

Communication: Discord will be used for daily communication, brief updates and discussions. For longer conversations voice chat is used for clarity. Communication with Supervisors is via email.

Tech Stack:

- **Frontend:** ASP.NET Razor Pages, HTML, CSS, Javascript
- **Backend:** Asp.NET Core (C#)
- **Database:** SQLite or PostgreSQL
- **Embedded Systems:** Raspberry Pi Pico (C++)
- **Hosting:** UCloud (If available for students)

7. Risks

Identified risks: Team size and Coordination: Managing a large team of 7 people will likely lead to an uneven workload distribution and coordination issues. To try to mitigate this within our capabilities, we'll establish clear roles and responsibilities, hold regular meetings and an always open platform to talk.

Skill Gaps: This is the first Semester most members are working with C++ and JavaScript, so we will encourage pair-programming and knowledge sharing. And some members are transfer students and unfamiliar with C# and with them we'll do the same.
Hardware and Documentation Access: We have very limited access to the desks and yet to receive the Web API documentation, which may lead to false expectations on how the desks work. We will need to schedule desk access as efficiently as possible.

Tasks: Keeping tasks up to date on the board will be difficult. So we need to hold members accountable for their contributions and assign responsibility for task tracking.

8. Project Organization

Our team is organized into the standard Project Lead and Developers for Kanban, where there are subroles for developers depending on what they develop.

Roles and Responsibilities:

- **Project Lead:** Coordinates the project, Sets up meetings and makes sure progress is met while also contributing to development.
- **Distributed System Developers:** Implements the Web app (devs may end up working with both frontend and backend).
    - **Frontend Developer:** Focuses on UI development.
    - **Backend Developer:** Focuses on server-side logic, API implementation, and database management.
- **Embedded System Developers:** Programming the Raspberry Pi Pico.

Team Structure: We will try to keep it as flexible as possible allowing members to take on tasks matching their skillset and interest. We are using Kanban which is inherently simple and flexible where we utilize agile principles like the one week sprint.

Work Distribution: Tasks will be broken down to take no more than a day to complete and GitHub Projects Kanban board to track them.

9. Project Plan

We will use the standard Software Development Life Cycle (SDLC) for how we plan our project. we will have 1-Week sprints for the iterations, as for milestones, since I don't have the schedule beside me I'll write example milestones that can be adapted into the schedule.

Milestones:

- Project Setup and Initial Design
- Backend and Database Development
- Frontend Development and API Integration
- Security and Data Visualization
- Embedded System Integration and Testing
- Finalization and Documentation

10. Tentative outline for project report

We will use the standard report structure we learned in the Software Engineering Course. It should include the snapshots of our board, burndown chart, backlog etc.