Study Board of the Faculty of Engineering

T630011402 - Fall 2024

Semester Project: Distributed Software Systems with Embedded Elements

Project Proposal

TITLE PAGE

Team members:
Ahmad M. Al-Qadiri
Mathias L.-Hansen
Patrik Muravski
Petar Spirovski
Tim L. Adam
Szymon J. Nowacki
Samuel Adjabeng

Date: 2024-08-10

1. Background/Motivation

Motorized desks have proven themselves to be significantly better than static desks, improving the user's posture, comfort and productivity.

1.1. Problem Statement

The Problem is that the user needs to manually adjust the height of each desk individually, which can be inconvenient and lead to incorrect usage that could contribute to discomfort or health issues over time. The lack of automatic, data-driven adjustments therefore consequently undermines the ergonomic benefits that motorized desks are designed to provide.

Also, desks are left at different heights after work. This becomes a challenge for maintenance staff during cleaning or reorganization as they have to manually adjust all desks.

The answer is a Desk Management System that we can use to analyze the data of the desk usage and we can continue with the improvement of the comfortability and productivity for the desk user’s. Through remote control the desks are able to be adjusted for easier maintenance of the office. This will address the needs of the employees connected with the desks like the everyday desk user and the cleaning staff.

2. Goals and deliverables

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 desks in specific rooms for easier cleaning. The embedded system will be able to control the desk through the included Pico W Explorer Base expansion board, communicate with the desk user and utilize its components e.g. Button, Screen, Potentiometer etc.

2.1. Personas in our system:

In order to design a solution that caters effectively to the needs of each user, it is important to conceptualize and highlight the key stakeholders involved and provide an overview of their expected roles.

Persona 1 (Decision-maker): The Office Manager (Sarah)

Age: 55

Responsible for:

1. managing and organizing the office space.
2. Obtain data on desk usage for health, occupancy, and maintenance.

Needs: A system that can analyze desk usage and be able to schedule adjustment desk heights automatically, for example, for cleaning.

Additionally, maintaining a consistent overall appearance and setting desks to the same height within a given time frame.

Goals: Cost-efficiency, space optimization, and employee satisfaction

Challenges: Difficulty managing cleaning schedules. Cleaner and employee complaints about uncomfortable workstations. keeping a uniform appearance overall.

Persona 2 (User): The Desk User (Maria)

Age: 35

Occupation: Accountant

Role: An employee who uses the desk on a daily basis.

Goals: A functional workspace with reliability, simplicity, and comfort.

Challenges: Difficulty adjusting the desk automatically by analyzing desk usage for health.

Needs: A system desk control with easy-to-use controls and clear user orders.

Persona 3 (User): The Cleaner (David)

Age: 25

Occupation: Cleaner

Goals: Ease of use, physical comfort, and time efficiency

Challenges: Difficulty cleaning under heavy desks and pain from bending and reaching under desks.

Needs: A system that can automatically adjust the height of a block of desks to a uniform level, making it easier to clean underneath and around the desks. Clear instructions on how to use the system.

Persona 4 (Tech): The Desk Technician (Alex)

Age: 30

Occupation: Desk Technician

Goals: Problem solving and customer satisfaction.

Challenges: Recurring equipment malfunctions and a variety of technical problems

Needs: A system that can analyze and diagnose desk problems, through desk usage data and downtime.

3. Objectives

To do this we've set these objectives:

3. 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.

4. Reliable communication between the frontend and backend which includes desk API requests and managing data storage in the database.
5. Use graph libraries to show desk statistics and trends intuitively and interactively.
6. Allow IT/System-Administrators to automate or schedule desk movements at specific times within different desk groups.
7. Use a Raspberry Pi Pico to control the desk through its expansion board.

4. Solutions

4.1. 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. Each part 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.

4.2. 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.

4.3. 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).

4.4. 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.

4.5. 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.

4.6. 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.

4.7. 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 have divided the requirements into Functional and Non-Functional requirements. We use MoSCoW for prioritization and each requirement is presented as a user story which shows who benefits and why:

5.1 Functional Requirements

ID	User Story	MoSCoW
F 1	As a user, I want to remote control my desk through a web app so that I can adjust its height remotely.	Must Have
F 2	As a user, I want visualization of desk usage statistics so that I can understand how they are being used.	Must Have
F 3	As a developer, I want to integrate the Raspberry Pi Pico W (Embedded system) with the web app (distributed system) so that it can communicate with the backend.	Should Have
F 4	As a System-Administrator, I want to automate desks movements at specific times so that maintenance can be done with the least effort.	Should Have
F 5	As a maintainer, I want predictive maintenance alerts so that I can fix issues before they become a problem.	Could Have

5.2. Non-Functional Requirements

ID	User Story	MoSCoW
NF 1	As a user, I want secure and reliable frontend and backend communication so that my data is safe and there are no interruptions.	Must Have
NF 2	As a System-administrator, I want a secure database for storing desk and user data so that it is guarded.	Must Have

NF 3	As a user, I want the web app to work on different device types and be accessible so that I can use it from my phone and accommodate my needs.	Could Have

Keep in mind these are initial requirements, we'll fill in missing tasks as we go following agile practices and break the current ones into ones that take a day at max.

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

Risk	Severity	Mitigation
Team Size and Coordination	High	Assign clear roles, hold regular meetings and have open channels to talk.
Skill Gaps (C++ and js)	Medium	Pair programming and knowledge sharing sessions.
Hardware and documentation access	High	Create an efficient desk access schedule and request API documentation.
Task Managing	Medium	Task Coordinator updating the Kanban board daily and weekly progress meetings.

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.
- **Developers:** They develop both the frontend and backend, where they can specialize in UI, Server-side logic, API implementation, Database management, and embedded programming.

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. Below are the planned sprints, but they have yet to be adapted into the schedule:

Sprints:

- 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.

Sample tentative outline:

- **Introduction:** Overview and scope
- **Background and Motivation:** Project relevance and affected stakeholders
- **Problem Statement and Project Objectives:** Challenges with the status quo and targeted remedies
- **System Architecture and Design:** Frontend, backend, database and Embedded system
- **Project Management and Methods:** Selected tools and development methodologies (e.g. agile, and GitHub for version control)
- **Risk Analysis:** Project and Team coordination challenges, and mitigation.
- **Testing and Validation:** Adopted testing strategies for system security and reliability.
- Results and Discussion: Analysis of project outcomes.
- **Conclusion**
- **References**
- **Appendices**