Scientific paper kladd

Software engineering assignment 4

## Introduction

## Methods

## Requirements

Functional requirements

Take user input for filling checklist

Upload checklist to database

Indicate errors as alarm for supervisor

Display contents for supervisor

Let supervisor view and acknowledge alarms

Add date and current shift automatically

Allow for user to select different checklists

Non-functional requirements

Usability: User interface needs to be as simple as one on paper. It should be intuitive to select checklist and fill in the boxes.

Reliability: All data should be available for supervisor to view at any time. Checklists should be available for users to fill Friday through Sunday.

Performance: The system needs to operate smoothly and be quick in loading phase.

Supportability: The system needs to be scalable for further expansions.

A screenshot of a computer screen

Description automatically generated

Use case diagram

A diagram of a system

Description automatically generated

Domain model

There should only be one iteration of the elaboration phase. This iteration should take no longer than one week.

## Architecture

There are numerous different architectural structures that a software application can be built upon. Two-tier architectures includes two layers: a presentation layer, which in this case will be the user interface where the firefighters can interact with the checklist and the supervisor can acknowledge alarms. A data layer, database where the checklists, user information etc... will be stored.

Other structures like the tree-tier architecture could improve the system making it more robust and scalable. In the tree-tier architecture an application tier will handle the business logic. For this case that would include managing and validating user inputs, and coordinating interactions between the presentation and data tiers.

Four-tier architecture is commonly used for larger systems. Here we divide the application tier from tree-tier architecture into application and business layer. For this system that would mean the application layer handles the service with GRASP pattern control while the business layer handles the business logic.

For this particular system a two-tier architecture should be sufficient for its purpose. Keeping a simple structure will also lower the workload making it more likely that the system will be implemented.

## Chat GPT Requirements

Functional Requirements:

User Interface:

Provide an intuitive user interface for firefighters to fill out checklists.

Allow supervisors to review checklists for errors and deviations.

Support interactions on both laptops and tablets.

Checklist Management:

Allow users to select from pre-made checklist templates.

Enable users to easily complete checkboxes for all equipment items.

Provide fields for users to add comments and sign with their names.

Automatically include date, shift (A, B, C, or D), and user information for each checklist.

Save completed checklists in an archive for future reference.

Notifications:

Notify supervisors of any deviations or errors detected in the checklists.

Implement email or in-app notifications for timely alerts.

User Management:

Support user authentication and authorization mechanisms.

Allow administrators to manage user accounts and permissions.

Non-Functional Requirements:

Performance:

Ensure the application responds promptly to user interactions.

Minimize latency for data retrieval and processing.

Security:

Implement robust authentication mechanisms to protect user accounts.

Encrypt sensitive data such as passwords and user information.

Ensure compliance with relevant security standards and regulations.

Reliability:

Design the system to be highly available and resilient to failures.

Implement mechanisms for data backup and recovery to prevent data loss.

Scalability:

Design the system to handle a growing number of users and checklists.

Ensure scalability of both the application and database components.

Usability:

Ensure the application is user-friendly and easy to navigate.

Provide clear instructions and guidance for users of varying technological proficiency.

Accessibility:

Ensure the application is accessible to users with disabilities.

Comply with relevant accessibility standards to accommodate diverse user needs.

Compatibility:

Ensure compatibility with a range of web browsers and devices.

Test the application across different platforms to ensure consistent performance.

Maintainability:

Design the application with clean and modular code for ease of maintenance.

Provide documentation and guidelines for future enhancements and updates.

A screenshot of a computer

Description automatically generated