**UCN dmai0916**

A picture containing clipart

Description automatically generated

**FINAL PROJECT REPORT**

14th of January 2019

Andreas Richardsen

Zahro-Madalina Khaji

University College of Northern Denmark

Technology and Business

Computer Science AP Degree Programme

Final Project Report

dmai0916

Project participants:

Andreas Richardsen

Zahro-Madalina Khaji

Supervisor:

Simon Kongshøj

Submission Date:

14th of January 2019

Contents

[Introduction 4](#_Toc534879597)

[Problem statement 4](#_Toc534879598)

[Group contract 4](#_Toc534879599)

[Analysis 5](#_Toc534879600)

[Gathering requirements 5](#_Toc534879601)

[System vision 5](#_Toc534879602)

[Brief use cases 5](#_Toc534879603)

[Use case diagram 7](#_Toc534879604)

[Prioritization of use cases 7](#_Toc534879605)

[Non-functional requirements 8](#_Toc534879606)

[Domain model 8](#_Toc534879607)

[Patterns and associations 8](#_Toc534879608)

[Design 9](#_Toc534879609)

[Implementation 9](#_Toc534879610)

[Conclusion 9](#_Toc534879611)

# Introduction

In this report we will discuss the process of developing our final project in the context of this course. We chose to build a Personal Task Manager that can easily be used to replace the classic TO DO list. This project is a web application that we implemented using technologies that we learned through this course. Throughout this report we will present the different stages of development that we experienced, our choices of technology and the implementation of this project.

Our goal with this project was to offer an improved organization method that can help anyone live a simpler and more stress-free life.

## Problem statement

We believe the old TO DO list method of organising to solve tasks has several deficiencies and it might be improved.

To create a TO DO list the old way one requires access to paper and a writing instrument. In our modern age, access to the internet is very common and technology is very wide spread. Therefore, a digital TO DO list could be a better option with possibly more functionality.

Here are some as the aspects that is lacks: not easily accessible. E.g., when you go grocery shopping and you forget your list at home. Therefore, an improved list should be available to access anywhere, anytime with just an internet connection.

Another issue is that of security. Suppose you want to keep you TO DO list private – a list written on a plain piece of paper might be seen by anyone around you who might be snooping or can be lost and who knows who might find it. This shows the importance of keeping your data safe.

Writing the tasks on a list on paper means that it is hard to edit or rearrange them if the need arises, and takes unnecessary space. Important tasks for the user might also not be properly evident so they might be overlooked. Therefore, a method of organising, categorising and highlighting list tasks is necessary.

* What are some ways in which we can improve the old-fashioned TO DO list?
* How to present the tasks in a more organised manner?
* How to highlight the user’s most important tasks?
* What are some ways to make sure that our list is accessible to the user anytime and anywhere with an internet connection?
* Which are some ways to store our data securely?

## Group contract

Insert here

# Analysis

## Gathering requirements

For this project we have not partnered with a company to build a solution for their needs but instead we have decided to work on our own idea. This meant that during our preliminary study we could not obtain any requirements from a customer, such as a company. We put ourselves in both the place of the final user of the system and the developer to gather requirements. The way that our application is supposed to be used is a personal aid therefore we thought it should offer a high level of personalization. This ensures that the needs of the user are a priority. The method we used to generate requirements was through idea generation and through our own experience of what an application like this should offer.

## System vision

?

The way our system is built is with the idea of organization in mind. Users would have the ability to manage their tasks in several ways. One could choose to have individual tasks and they would also have the possibility of grouping their tasks into lists. Lists consists of one or more tasks. Each task can have other tasks called subtasks. To highlight a certain task a user can favorite it.

## Brief use cases

We decided to represent our functional requirements as brief use cases. During our analysis we discovered that none of our use cases were complex enough to necessitate making a fully dressed use case.

Manage account (Create account): User navigates to the login page, selects the create account option, the user inputs the required information for creating the account, the system creates the users account and the user is redirected to the log in page.

Manage account (View account): The user selects the account settings option, and then chooses the option View account; the system displays the account information on the page.

Manage account (Update account): User selects the account settings option, and then he changes the information he wishes to change and saves it, the system records the changes.

Manage account (Delete account): The user selects the account settings option, then selects Delete account and is prompted with a popup to confirm that he wants to delete his account; the system removes the users account information and redirects the user to the login page.

Log in: The user inputs their credentials and the system redirects them to their dashboard.

Log out: The user selects the log out option from the contextual menu and the system log out the user.

Manage list (Create list): The user presses the create list button and inputs the name of the list, the system creates and adds the new list to the dashboard.

Manage list (View list): The user selects a list and the system displays the list and all its details.

Manage list (Update list): The user selects the list settings, he is redirected to the view list page where he can change the name and/or add tasks.

Manage list (Delete list): The user selects the list settings, he is redirected to the view list page then he selects the delete list button, the system removes the list from the records and the user is redirected to the dashboard.

Manage task (Create task): The user presses the add task button on the view list page, then inserts a name for the task and presses save, the task is saved to the system and is displayed in the list.

Manage task (Update task): The user selects the task on the view list page, then updates the desired information and saves the task, the system records the task and the task is updated.

Manage task (Delete task): The user selects the task on the view list page, then presses the delete task where he is prompted with a popup that ask if he is sure, when the user presses yes, the task is deleted from the records and the view is refreshed.

Add subtask: The user presses the add subtask button on a selected task, then inputs the name for the subtask and presses save, the system records the task and the subtask is displayed on the task.

Remove subtask: The user presses the remove subtask button on a selected subtask, the system asks for confirmation and then deletes the subtask from the task.

Favourite task: The user presses the favourite button on a selected task, the system marks the task as a favourite and highlights it.

Search task: The user inputs the name of the desired task in the search bar; the system displays the results that match the input provided by the user.

Mark done: The user marks a selected task or subtask as done; the system displays a check mark next to it.

## Use case diagram

A close up of a map

Description automatically generated

Figure n: Use Case Diagram

## Prioritization of use cases

1. Mark done: The backbone of any TO DO list is checking off tasks as they are done.
2. Manage task: Adding and deleting tasks is quite important for managing a list.
3. Manage list: Being able to manage your list is also important in our system.
4. Favourite task: Tasks that are favourites should be properly highlighted therefore it necessary.
5. Manage account: Users use their account to save their data and there not much to do after that so not high priority.
6. Log in: Basic need for a log in.
7. Log out: Used for logging out.
8. Add subtask
9. Remove subtask
10. Search task: Not very important unless one has many lists

## Non-functional requirements

System must be:

* User friendly
* Accessible
* Fast
* Secure

## Domain model

A screenshot of a cell phone

Description automatically generated

Figure n: Domain model

### Patterns and associations

The domain model contains an Item-descriptor pattern between the Task class and the Subtask class. This means that a Subtask needs a Task to exist, otherwise when a task is deleted then all its subtasks are deleted with it. Between a person and a list there is an association relation which means that a person can have many lists, but a list can only belong to a person. A List can have many tasks and on the same hand a task can belong to many lists.

# 

# Design

Architecture

User interface

Design class diagram

Relational model

Normalization

# Implementation

Choices of technologies

Backend

Frontend

Code

Code standards

Testing

# Conclusion

References