Indhold

1 Introduction

1.1 Purpose of the system

The purpose of the calendar system is to make a calendar system for a workplace, capable of sharing entries and organizing meetings etc. The system is intended to be used in a workplace environment, where people know each other, or at least are acquainted in some way. Thus the program has no need for contact lists and blocking of people, since it is used by people working together, to organize meetings etc.

1.2 Design goals

Usability: The system should be easy to learn.

Fault tolerance: The system should be fault tolerant to loss of connectivity with the calendar server.

The system should use a low amount of bandwidth.

The client part of the system should be compatible with older windows versions.

1.3 Definitions, acronyms, and abbreviations

1.4 References

1.5 Overview

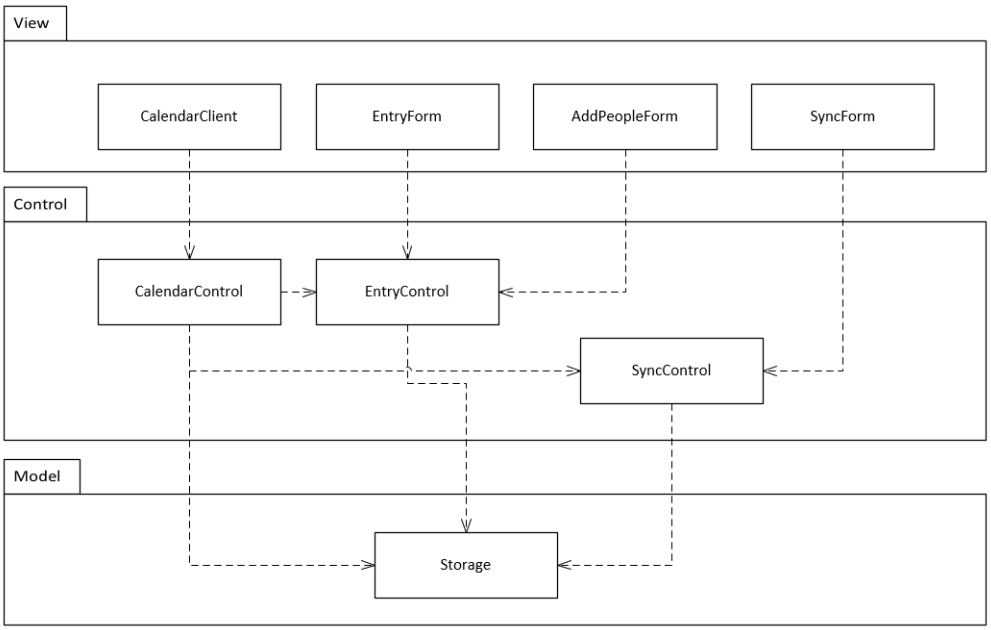
2 Proposed software architecture

2.1 Overview

The following chapter will go through: the chosen architecture patterns and design of the calendar system, the design patterns used to implement it and how the control flow of the program.

2.2 Subsystem decomposition

The figure below shows the subsystem decomposition of the User part of the program.



Figur 1: Subsystems and their decomposition

2.3 Hardware/software mapping

2.4 Persistent data management

Identifying persistent objects
The calendar system works with four entities, which need to be stored: The administrator and user entities, which need to be persistent, so accounts doesn't have to be made every time the system starts again. The

workgroups also have to be saved since they would be useless unless persistent. Since it's a calendar, the entries also need to be persistent, else it wouldn't be a calendar, which is used to write down and remember events.

Selecting a storage strategy
A mixed strategy has been chosen for the system, between relational database and flat file. The relational database will save all persistent information, which is then available online, while the flat file will save all relevant information to the specific user locally. This makes the system able to function off- and online. The relational database will be updated when online while also updating the local file.

2.5 Access control and security

There are only two kinds of actors that will be using the Calendar System: User and Administrator. Below is the access matrix, which specifies which parts of the Calendar System the two different actors will have access to.

Objects Actors	User	Workgroup	Entry	Sync
User	addUserToEntry	addWorkgroupToEntry	manageEntry	manageSync
Administrator	manageUser	manageWorkgroup		

Figur 2: Access matrix for the Calendar System

2.6 Global software control

The server should be using threads as to allow multiple clients connected, updating their calendar at the same time. While the local client should be event-based, since it's simpler than threads and there is no reason for the clients to use threads.

2.7 Boundary conditions

Configuration use cases:

InstallClient	An administrator creates the calendar server, allowing the creation of new administrators and users, who can make use of the systems functions.

Start-up and shutdown use cases:

StartCalendarServer	An administrators starts the calendar server. As soon as the server is up and running, users will be able to synchronize their local calendar with the servers.

ShutDownCalendarServer	An administrator stops the calendar server. Any changes made to entries, by users will instead be saved in the local storage and uploaded when the server becomes available again.

Exception use cases:
The calendar system can experience two major classes of system failures

- A network failure between the client and the server.
- A failure causing the calendar server to unexpectedly terminate.