Calander System

Contents

[Introduction 2](#_Toc169693732)

[Scope & Features 2](#_Toc169693733)

[Requirements 3](#_Toc169693734)

[Functional Requirements 3](#_Toc169693735)

[Non-Functional Requirements 3](#_Toc169693736)

[Architecture 4](#_Toc169693737)

[Data 4](#_Toc169693738)

[Visual Design 5](#_Toc169693739)

[Implementation 6](#_Toc169693740)

[Language(s) 6](#_Toc169693741)

[Stages of Development 6](#_Toc169693742)

## Introduction

A calendar utility desktop application intended for personal use and use within an organisation. At a basic level, a user should be able to create an account to a local database and create, modify and view events and invite other locally hosted users to events which then makes it visible on their calendar.

This document will focus on the version 1 of the project, with brief discussion of possible future continuation. Currently, there are no plans to continue the project past this version.

## Scope & Features

Once a user has created an account on a server, upon login they will be guided to their personal calendar where they can view and manage their events.

A user can create events, in which they can control the following parameters:

* Title
* Category
* Time (to/from)
* Date (to/from)
* Participants

Categories are specific to each user and can be created and modified freely by them. This allows them to mark events and their colour and filter what events they are viewing based on their categories. An event a user is invited to can be marked by the invitee with their own category which will not be visible to the other participants of the event.

#### *Accounts & Management*

There are access levels within a server, controlling what a user may be able to access and manage.

##### Terminology

*User group*: The grouping of users on the server. A user can only interact with users in the user group they have access to.

*Sub-group*: Subgroupings of a user group in order to manage events to specific groups of individuals. A user may be a member of multiple sub-groups, but only those present in the user group can be invited to a sub-group.

Within a server group, there should be three types of privilege ranked from high to low:

* Administrator
* Manager
* Regular

For a sever group, there must be at least one administrative account present. A user may relinquish their administrative rights if there is at least one other account with the privileges.

The difference in user privileges can be summarised by the following diagram:

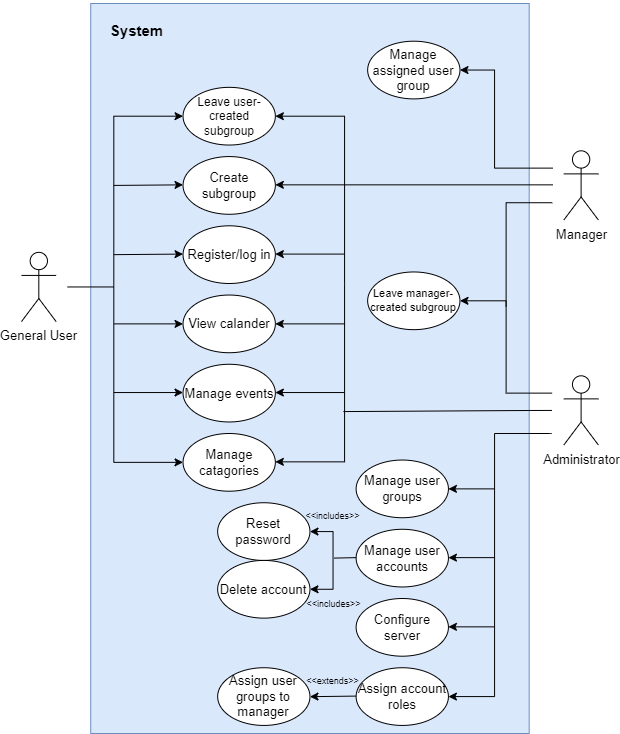


Figure 1 Use-case and Privileges diagram

Note: Privacy is maintained by all users, an administrator cannot view passwords, user events or user interactions outside of who is present in each user group, sub-groupings are completely private.

#### Future Scope

##### Interaction

Incorporating interactivity between members through a chat box. This would expand the application’s goal beyond just scheduling and must be scrutinised and planned for as a V2 feature.

##### Remote Hosting Service

An expansion of this project should see the ability to host user accounts on a remote server hosted by the application vendor making user data accessible outside of their chosen database. This will not be attempted in this scope of the project.

## Requirements

### Functional Requirements

* User must be able to create an account.
* User must be able to log into their account with the credentials they defined.
* User must be able to create events:
  + Events must have a title and date.
  + Events must have optional parameters:
    - Description
    - Category
    - Time begin and end and/or date end
    - Invitations to other users
  + Events must be modifiable and deletable.
* User must be able to create categories for events.
  + Categories must have a title and colour.
* User must be able to create user groups.
* User must be able to remove themselves from a group.
* User must be able to remove themselves from events if invited individually.
  + If invite is a group invite, user cannot remove it from their events.
* User must be able to filter events by category.
* User should have privileges associated with their level.
  + Admins should:
    - Be able to modify user groups.
    - Be able to remove/grant user access to creating user groups (including group limits).
    - Be able to delete user accounts.

### Non-Functional Requirements

* Application must be able to send and retrieve data from a database.
* Database must store password securely.
* Releases must be cross-compatible (ie, an event created on a newer release must still be visible to a user running an older release and visa versa).
* When in calendar view and active, application should check for remote event changes every minute and update them visually in the calendar.
* Each account must be associated with a user group, which must have at least one admin account present.

## Architecture

The application is, as per the scope requires, is to be usable in both personal use case and business case and therefore must provide options for both with an easy-to-setup default configuration. This affects how data should be handled as well as how the application should present its features and configuration settings to accommodate its variation user base and [user types](#_Accounts_&_Management).

The application is to run by default on a desktop device without the requirements of any additional devices or software.

[more headers n stuff here?]

### Data

#### Requirements

* Must be non-volatile.
* Must be configurable, saved locally to user’s computer by default or to a defined server.
  + Application must have full CRUD privileges.
* Security (such as encryption) for handling of sensitive data, ie password.
* By default, should not require external (third-party) services or software.

#### Structure

The application of data is ideal for a relational database, as there is expected to be 1+ users with shared properties and pre-defined structures for the program in which the data is held (ie, events will have set parameters) and data must be fetched by parameter such as user ID.

The structure can be represented by the following (not including details such as privileges user settings):

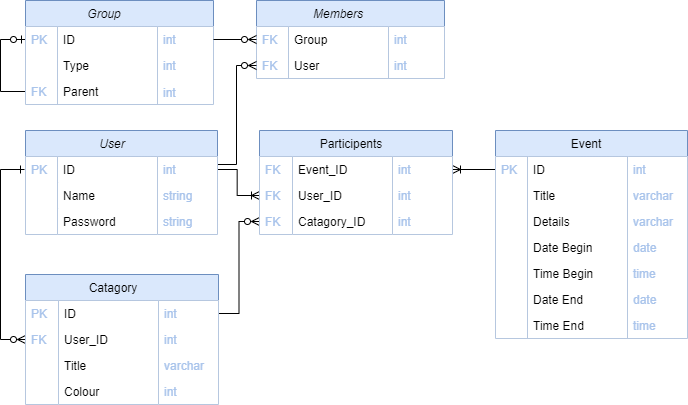


Figure 2 Database Entity Diagram

## Visual Design

[To complete]

## Implementation

### Language(s)

This project shall be conducted using C# for not much other reason than I wanted to practice more with it and its high-abstraction provides an easier format to explore other concepts like SQLite (spoilers for the database side of it) and GUI.

The database for which holds and manages the user’s data set by default will be setup with SQLite, as it handles the structure necessary and does not require the user to download or use any additional services. This of course is configurable by the user and the user may wish to switch to a server of their choosing to store the application’s data.

### Stages of Development

* Database setup
  + Testing
* GUI
  + Toolbar
  + Calendar
  + Categories
  + Additional windows (config windows)
* Account privileged configuration