Title:

Architectural Document

Name of application:

Calendar Application

Authors:

Kyle Allen

Mhealyssah Bustria,

and Favian Quach

Date:

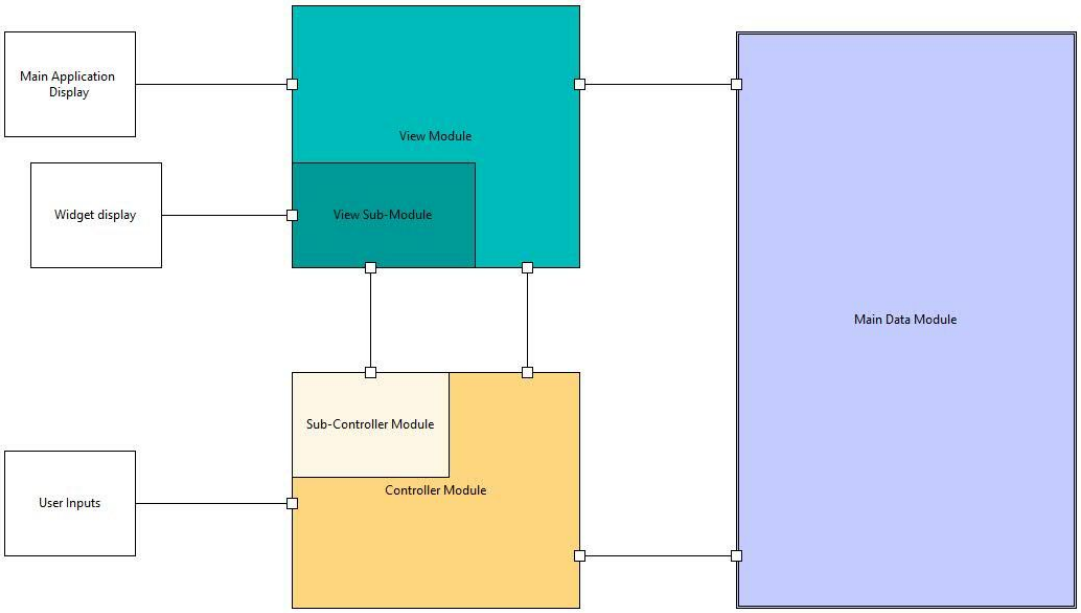
Sunday 10 March 2019

Table of Contents

1. Introduction

The system for the Calendar Organization Application (Calendar App) includes the following portions: The Controller module, the View module, and the Model module. Each of these portions performs the necessary functions for a user to be able to view a calendar filled with user-created events.

2. Architecture



The main architecture pattern used by the system is Model-View-Controller. This pattern was chosen because the main point of this application is to be able to graphically see and organize tasks based on user inputs and the state of the application.

The encapsulation of the main data allows for a more robust application to be available for the users. Also incorporating client and server connections allows other users to interact with each other through the application, such as sending and receiving tasks and comments.

The Controller module of the system acts as the mediator between the user and the program. The Controller takes in inputs from either the user or incoming connections. The Controller uses those inputs to either update the graphical representation (the View) or call another module to access the central data (the Model).

The Model module of the system is the main data module, which would contain the user's Calendar and events. The Model would also process any requests and changes that it receives from the Controller. The data fields that an event is composed of are accessible only by the event's creator unless the event's creator gives access to other user(s) or a group.

The View module of the system processes all of the display and works with the Controller and Model modules to appropriately organize the application for the user. The View displays any tasks that are in the Model, in addition to any updates that the Model has.

3. Component Design

Three main components are used in the application. The Controller module, the View module, and the Model module. The system is designed this way for ease-of-use, and so the user is able to manipulate the controller to change the view of the tasks.

3.1 Controller Module

Purpose

The purpose of this module is to allow user to manipulate data in order to create events and tasks within those events to keep track of important information. This module allows the user to experience a view that suits their preferences by letting the user change between different views. This module also give users the option to share events with groups or other users.

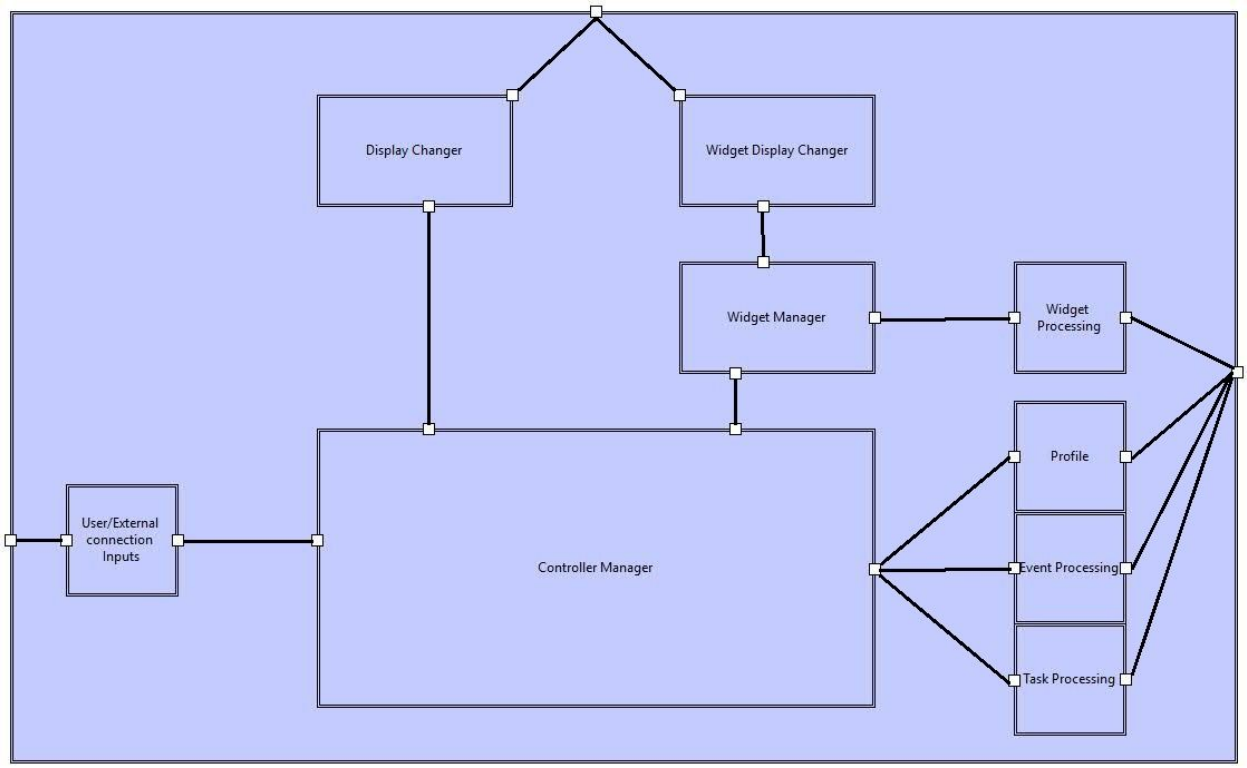
Rationale

This module allows users to carry out all data manipulation provided by the application. This includes: changing the view display, updating a profile, creating an event, creating a task, and creating/viewing a group.

High-Level Controller Design

The Controller module is broken down into all of the aspects the user should be able to manipulate within the program, as shown in the figure below.

All inputs come from the user and go through a controller manager that will then, based on the inputs, decide which process will take place next.



Required Interface

- User Input
- View Module
- Memory Module
- Model Module

Provided Interface

The provided interface will be allowing the user to carry out processes such as:

- Account Creating
- Event Creating
- Task Creating
- Group Sharing/Creating
- Display Changing
- `void createNewGroup(Group g)`
  This function will create a new group, creating an object g with all of the required information being provided by the user.
- `void addUserToGroup(String ID, string uName)`
  This function will add a user to the group based on the desired user's ID and first name.

3.1.1 Profile Module

Purpose
The purpose of this module is to allow users to create a profile that can be used to store information about themselves. The profile can also be used to make sure that the user's events and tasks are secure when sharing with other users/groups.

Rationale
This module will store user information such as: names, passwords, and events tied to each account.

Required Interface
- User Input
- Controller Module

Provided Interface
Provided will be functions that allow the user to create their own profile:
- `void createUserProfile(User p)`
  This function will create a user profile with p being all of the information required by a User class to create a profile.
- `User findUser(string userID)`
  This function will find a user. ID is required to find the user. If the desired user does not exist, the function will return nothing.
- `Boolean userExist(string userID)`
  This function returns true if the user with the specified ID exists and false if the user does not exist. Requires user ID to find.
- `string changePassword(string userID, string pass)`
  This function will allow user to change their password. Needs their user ID and current password in order to change.

3.1.2 Event Module

Purpose
The purpose of this module is to allow users to create events based on real-life situations. Examples of events that may need to be completed are: chores, group assignments, party planning, etc.

Rationale

This module will allow users to create an event that is tied to a certain day and has an end date or end completion time of their choice. The user will be able to create the event and then change certain parameters if needed.

Required Interface

- User Input
- User Profile
- Task Module

Provided Interface

These functions will be provided for use:

- `Event createNewEvent(Event e)`
  This function will create a new event based on what is required to create an event object, then sends e to be created.
- `string changeEventDescription(string descrip)`
  This function will allow users to change the description of an event, allowing users to input a string.
- `int changeDueDate(string dueDate)`
  This function will allow users to change the due date of an Event. The user inputs the new due date as the parameter.
- `string addEventComment(string eCom)`
  This function will allow users to add an additional comment that is separate from the description of the event. Passes a string.
- `Boolean isEventCompleted(string eName)`
  This function will check if an event is completed. The function will search for the event by the name given. If the event is completed, the function will return true. Otherwise, the function will return false.

3.1.3 Task Module

Purpose

The purpose of this module is to get instructions to the completion of an event. Users can create tasks that give individual instructions for what steps need to be completed to finish an event as a whole.

Rationale

Tasks can be assigned to specific users, with parameters that help the users complete the task. Each task is tied to an event; a task contributes to the completion of the event as a whole.

Required Interface

- User Input
- User Profile
- Event Module (Event needs to be created to assign a task to it).

Provided Interface

- `Task createNewTask(Task t)`
  This function will create a new task t based on the required information inputted by the user.
- `string changeTaskDescription(string descrip)`
  This function will allow users to change the description of a task, allowing users to input a string.
- `string addTaskComments(string comment)`
  This function will allow users to add an additional comment that is separate from the description of the task. Passes a string.
- `Boolean isTaskComplete(string tName)`
  This function will check if a task is completed. The function will search for the task by the name given. If the task is completed, the function will return true. Otherwise, the function will return false.

3.1.4 Display Changer Module

Purpose

This module will allow the user to change between different displays for viewing the calendar.

Rationale

This module gives users the ability to change between the following views: year, month, week, and day. This module gives the users options on how they would like to view their events.

Required Interface

- View Module
- User Profile

Provided Interface

- `void changeView()`
  This function allows users to change their current view to any other available view.
- `void makeDefaultView()`
  This function allows users to choose a certain view to be set as the default view upon application startup. This prevents users from having to continuously change the view upon startup.

3.2 View Module

Purpose

The purpose of this module is to change the user interface (UI) view of the program based on what is selected by the user.

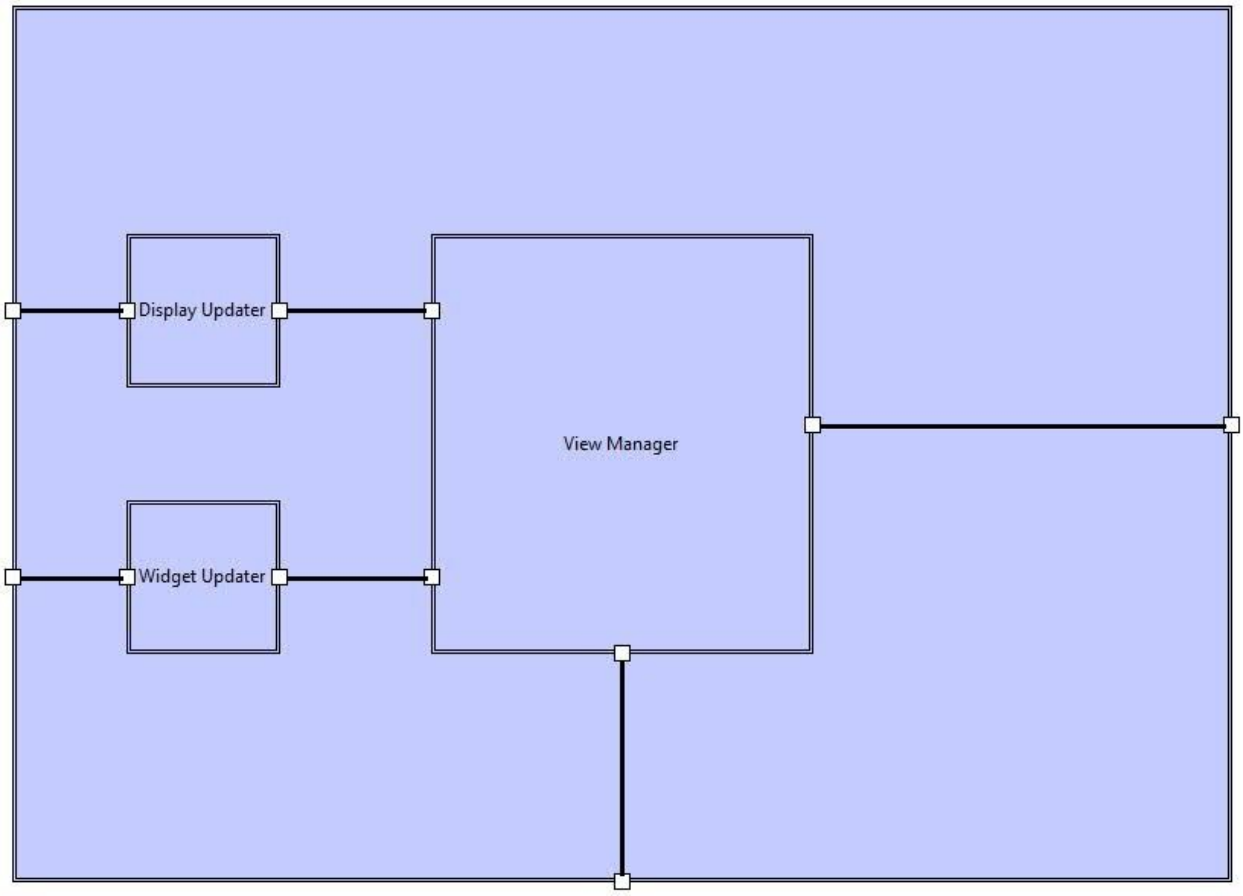
Rationale

The user will have a choice between multiple graphical calendar interfaces based on a year, month, week, and day view that will allow the user to see all their events and tasks in whichever way is most efficient to them. This module acts as a bridge that allows the controller to have this ability to change these views.

High-Level View Design

As seen below, the View module takes in inputs coming from either the Controller module or Model module, then determines what process shall be carried out using the View Manager.

The only options to change are the Display updater and the Widget Updater.



Required Interface

No required interface. (View will just default on start unless change by user)

Provided Interface

Provided is everything needed for the view to be updated through the view displayer.

3.2.1 Display Updater

Purpose

This will update the displays for the user.

Rationale

Similar rational to the view module

Required Interface

- View Module
- User Profile

Provided Interface

- `void changeDisplay()`
  This function will change the current display.
- `void changeDefaultDisplay()`
  This function will change the user's current default display.

3.3 Model Module

Purpose

This module manages all the data that users have. This module also manages all of the group requirements such as assigning other users to groups. This module also manages server aspects needed in order for the program to allow for sharing of the data.

Rationale

This module will be the main management system for all of the functions that need to take place and all the data that needs to be utilized.

High-Level Model Design

The design of this module is to store all the information on the server and also allow users to connect to their created group as well as be able to change their options within their group.

Required Interface

The required interface will be the control module.