Calendar Application
Requirements & Architecture

•••

By: Kyle Allen, Mhealyssah Bustria,
and Favian Quach

Date: Monday 11 March 2019

Our Goal

Designed with the average user in mind, we are motivated to make your already hectic life a little more organized.

A Task we wish to accomplish by lifting the burden of organizing your life and making coordination with groups a little easier.

Project Introduction

Overview

- Main uses:
    - Event Organization
    - Task Organization
- Sharing events and tasks with other users
- Group schedule synchronization

Scope

- The system allows users to create, customize, and organize events and tasks

User characteristics

- easy to use for all levels of technical skill

Two Main Objects

Event

- An activity, ideally spanning a large interval of time.

- Event creator can edit the event and give other users access to the information.

- A group of users can be assigned to work on an event.

Task

- A smaller activity, defined under a larger event.

- Among a group assigned to the event, a task can be assigned to individual participants.

- Can be assigned to a color, allowing for more organization in the event

Creating Events

Some attributes set by the user:

- Title

- Start & End times

- Additional comments

The user becomes the Event Creator.

Sharing Events

Normal participant

Able to make changes to a personal copy of the event without affecting original

If the Event Creator grants admin access

That user can push changes which will be seen by all participants

Tasks

Some attributes set by the user:

- Title

- Start time

- Deadline

- Color

Tasks are tied to an Event.

Groups

- Will see shared events and tasks on their calendar.
- Tasks can be assigned to specific users within a group.
- Users within a group may have varying degrees of access.
- The event creator has the most features available.

Notifications

- Users can opt in or out at any time
- Reminders for:
    - upcoming events/tasks (start dates)
    - upcoming deadlines
- Users can choose how often a notification will occur
    - default: one notification one hour before the critical time

Calendar Views - Can be switched using a dropdown menu

The Year View

Displays the 12 months of the selected year. Exclamation marks indicate months in which at least one event exists.

The Month View

Displays the days of the selected month. A colored dot indicates an event that exists in a day.

The Week View

Displays the seven days of one week. Due dates are represented by a colored block with a time in it.

The Day View

Displays 48 blocks (each block represents a 30-minute time frame). Timestamps represent events, and a timestamp can be pressed to display more details.

To-Do View

- Displays a list of all incomplete tasks for the current day

- Displays future tasks that are marked as important or urgent

Main Functions

- Creating, editing, and deleting events/tasks

- Labeling and organizing tasks

    - setting a color to a task

- Sending and receiving events and tasks

- Setting and customizing notifications

- Group event synchronization

- Changing the view

System Interfaces

User Interface

- Main calendar display on startup
- Smaller widget display

Hardware Interface

- Mobile support
    - Android Pie (9.0)
- Written with JavaScript support

Communication Interface

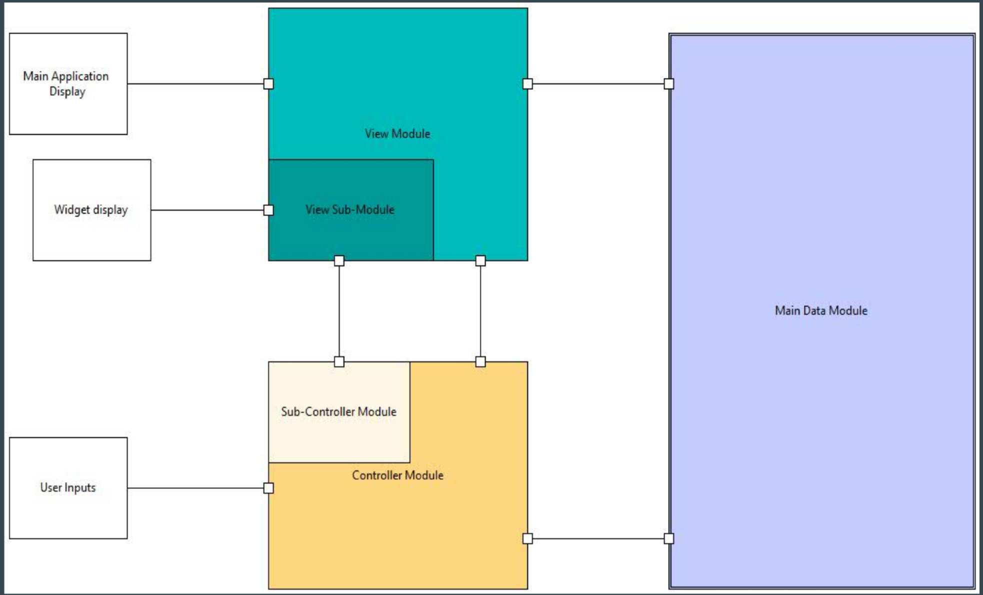
- Client/server system
- Internet
    - users can send and receive Events and Tasks

Security

- All information will default to private unless prompted otherwise
- User can opt in or out of receiving event or tasks entirely.
- Event and Task invitations will prompt user before being added to their calendar
- Event owner can choose the level of access for other added users.
- Important information can be locked by Event Owner or Admins
- Only Event Owner and Admin will be able to push Tasks unconditionally

How we want to design our project.

- We decided to use MVC (Model View Controller) architectural pattern
- The Modular design allows for easy encapsulation and protection of each modules' assets.
- The Widget will be handled as a separate entity that is managed by the main application.



Three Main Modules

- Each Module is encapsulated and changes should only be performed by their respective module.

Controller

- Processes inputs and calls the other modules as needed.

- Has a Sub-Module for handling Widget requests.

View

- Updates the display according to the Controller and Model

- Has a Sub-Module for handling Widget display

Model

- Manages and stores the data for the application.

Controller Module

```
void createNewGroup(Group g)
```

- Creates a new group using information provided by the user.

```
void addUserToGroup(StringID, string uName)
```

- Adds a user to the group based on a given user ID and first name.

Controller Module

Profile Module

```
void createUserProfile(User p)
```

- Creates a user profile. p holds all of the information required by the User class.

```
User findUser(string userID)
```

- Finds a user based on a given user ID. If the user does not exist, the function returns nothing.

Controller Module

Profile Module

`Boolean userExist(string userID)` - Finds a user based on a given user ID. -
- Takes a user ID to find. Returns true if the user exists, returns false otherwise.

`string changePassword(string userID, string pass)`

- Allows the user to change their password. Needs the user's ID and current password.

Controller Module

Event Module

```
Event createNewEvent(Event e)
```

- Creates a new event. e holds all of the information needed for an Event object.

```
string changeEventDescription(string descrip)
```

- Allows the user to change the description of an event.

```
int changeDueDate(string dueDate)
```

- Allows the user to change the due date of an event.

Controller Module

Event Module

```
string addEventComment(string eCom)
```

- Allows the user to add a comment to an event, separate from the description of the event.

```
Boolean isEventCompleted(string eName)
```

- Takes the name of an event and checks if that event is completed. Returns true if the event is marked complete, returns false otherwise.

Controller Module

Task Module

```
Task createNewTask(Task t)
```

- Creates a new task based on information provided by the user.

```
string changeTaskDescription(string descrip)
```

- Allows users to change the description of a task.

Controller Module

Task Module

```
string addTaskComment(string comment)
```

- Allows the user to add a comment to an task, separate from the description of the task.

```
Boolean isTaskCompleted(string tName)
```

- Takes the name of a task and checks if that event is completed. Returns true if the event is marked complete, returns false otherwise.

Controller Module

Display Changer Module

```
void changeView()
```

- Allows the user to change their current view to any other available current view.

- The available views are the four different calendar view options and the to-do view.

```
void makeDefaultView()
```

- Allows the user to choose a certain view to be set as the default view. This prevents the user from having to continuously change the view upon application startup.

View Module

Display Updater Module

```
void changeDisplay()
```

- Changes the current display based on the current information in the Model module.

```
void changeDefaultDisplay()
```

Model Module

- manages user data
- manages group requirements
- manages server aspects
    - storing information
    - sharing data

Closing Thoughts

With this design, we hope to create an environment that is secure and simple for everyday users to plan and organize their lives.

Secure enough to trust us with organizing your day, and simple enough so you don’t need a manual to do so.

With added functions to synchronize with a group, Setting deadlines and figuring out who is doing what should be much simpler and productive.

Thanks for your time!