CIS 422

Software Design Specification (SDS) for YACC

The document in this file has been adapted from the IEEE Guide to Software Design Specifications (Std. 1016-2009)

CIS 422

Team Number 6

499ms

Zachary Bower, Chase Craig, Refael Yehuda, Noah Palmer, Benjamin Yin

Software Design Specifications

Version 1.2 Date: (02/04/2019)

Table of Contents

1. Introduction

1.1. Purpose of this Document

This software design specification is made with the purpose of outlining the architecture and design of YACC in detail. This document will provide developers an insight into meeting an end users needs. This document will provide developers an understanding of YACC inner workings to facilitate an efficient and accurate production.

Section 2 of this document describe the architectural design of YACC. The high level components, application programming interface, graphical user interface, subsystems, and systems, and algorithms used.

Section 3 of this document describes component design from a high level. This allows every developer to efficiently produce and test code for YACC.

1.2. Scope of the Development of this Project

This software design specification will demonstrate how YACC will meet the functional and nonfunctional requirement specified listed in the software requirement specification.

This document will provide a framework for all developers describing the high level components, application programming interface, graphical user interface, subsystems, and systems, and algorithms used.

1.3. Definitions, Acronyms, and Abbreviations

Software Design Specifications (SDS) : A document describing the built product. The SDS describes external visible behavior as precise as possible.

Initial Project Plan (IPP) : A document outlining the process from beginning to completion of YACC. This document will include : a management plan, multiple milestones including dates and items required, an outline of scheduled meetings, the responsibilities of each member, the coding style requirements for the development cycle, and a rationale behind each these choices.

Application Programming Interface (API) : A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service

Functional Requirement (FR) : A specification for the required behaviour of a system or component.

Non-functional Requirement (NFR) : A specification for the judgement of the operation of a system or component.

User Experience (UX) : The overall experience of the end consumer using YACC.

Coding Style Document (CSD) : A document outlining the expected coding style for all developers of YACC to follow. This will include: parentheses spacing, function naming, variable naming, file naming, comments, and more.

Citation Expectations Document (CED) : A document outlining the expected style of how a developer of YACC is to cite references used.

1.4. References

References related to the internal development

(1) **IPP** : https://docs.google.com/document/d/1ZszO4mlShik_dEfztpab-m4zGZj5UHmMHrU0SFNLGkg/
(2) **SRS**: https://docs.google.com/document/d/1sNOXpG85zZVb8jMq4xzn_6ZcFwR9cahqVJ5A0LunbMs
(3) **CSD**: https://docs.google.com/document/d/1KL8Q_9lhUDCnlcPMZtaPfaDTiaKeyBGpLgd7bN3cE5s/
(4) **CED:**
(5) **Project Requirements**
    (a) https://classes.cs.uoregon.edu/19W/cis422/P1.html
    (b) https://classes.cs.uoregon.edu/19W/cis422/P1_Grading.html

References used to create this document

(1) **SDS Template:** https://web.cs.dal.ca/~arc/teaching/CS3130/Templates/Design%20Templates/SDS.doc

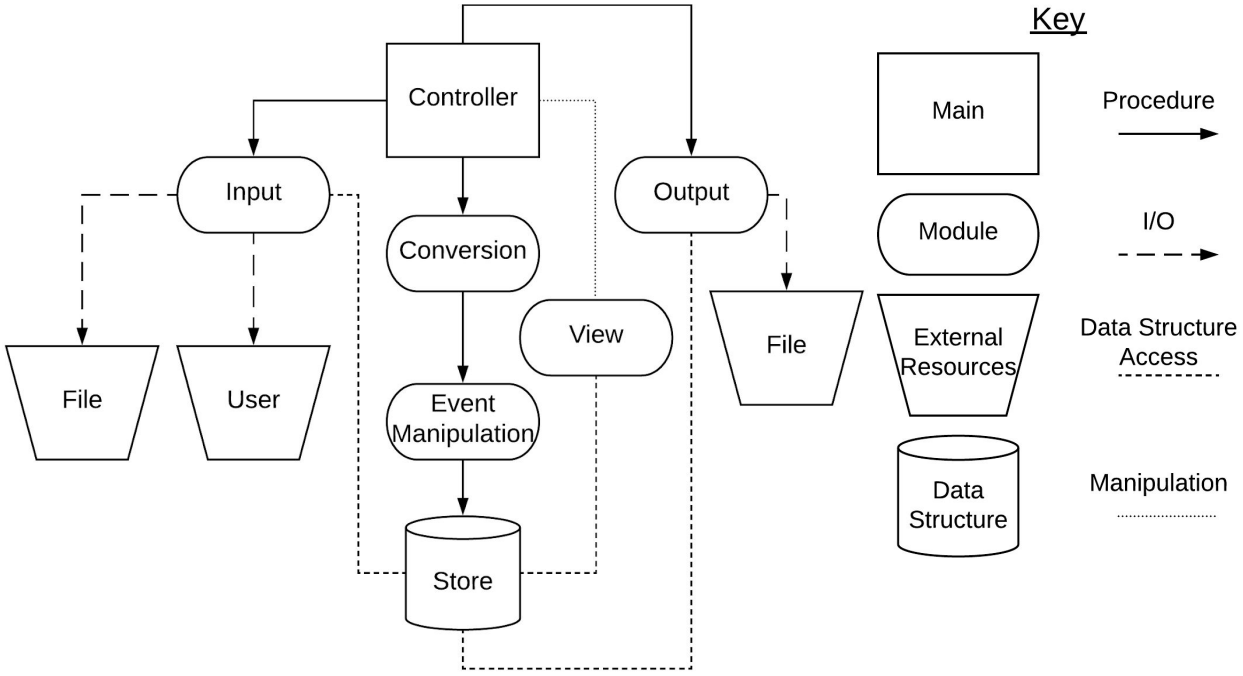
2. System Architecture Description

2.1. Overview of Modules/Components

There are 4 primary components to the system of YACC. The application is following the Model-View-Controller setup for designing the code. Additional to the given Model, View, and Controller, the last component specified in this document is the input/output (I/O). Focusing first on the Controller, it drives the program, acting as a liaison between the Model, View, and I/O. The second component in focus is the View. Within the view, there is 3 subcomponents: the MonthView, the DayView, and the EventView. The EventView in specific is the module the user uses to add/edit/delete events. The third component is the model. The model is composed of 2 subcomponents: the Calendar and the CalendarEvent. The last component is the I/O. The I/O is composed of 2 subcomponents: the file input/output and the user input.

2.2. Software Architecture

2.2.1. Static Model



This diagram is a static model of how the different components (I/O, Controller, Model, View) work together. The flow of the relationship from semi-left-to-right perspective. The first action to occur is the controller starting up and contacting the input module. Here, the module will look

to see if a file already exists. If the file exists, it will send the information back to the controller. The controller will then send it through the conversion interface module, that will then pass it to the module which will perform the event manipulation on the data. In this case, it adds all of the events and loads them into the store. From here, the controller will then update the view to reflect any added events.

Next, when the user provides input, the controller will contact the input module to receive any data pertaining to given input. This input could be adding/editing/deleting an event. For this, the controller then passes the data to the conversion interface module which then passes the parsed data to the event manipulation module. From here, it is updated and reflected in the store. After this, the controller then manipulates the view to update the necessary components. Once all input has ceased and the user would like to close the program, the controller will contact the output module to write to the file any information regarding the events currently stored in the store module.

2.2.2. Dynamic Model

Referring to Appendices 9 - 13, it provides an in depth view on the inner-working processes to perform the necessary operations. First looking at Appendix 9, it is of the procedure of opening and closing the application. Following the diagram, it can be seen that there is an interaction between the User [input], Controller, and the File System. Appendix 10 shows the procedure that the program will follow in conjunction with the user as an event is attempted to be added to a specific date. Appendix 11 goes into detail about the procedure that the program will follow in order to allow a user to edit an existing event. Appendix 12 gives the procedure for enabling the program to delete an event at the user's request. Appendix 13 describes the procedure for allowing the user to go into Day View mode for seeing all events that are occuring on that specific date.

2.2.3. Month View

The main view of YACC and the default view the user sees on program startup. The month view will contain the following basic layout items:

Navigation Arrows - In the top right corner of the window, a left and right arrow will allow the user to navigate forward (right) and backward (left) one month at a time. (See Appendix 5.1 item 1)
Month Display - In the center of the window, the name of the current month is displayed. (See appendix 5.1 item 2)
Day Display - Below the month display, the days of the week are listed as: Monday - Sunday (See appendix 5.1 item 3)

Calendar Matrix - A matrix filled with number representing the date. (See appendix 5.1 item 4,5)
Item Event - A box with the event name corresponding to an event the user has created. The text displayed will be taken from the user (See appendix 5.1 item 7)

2.2.4. Day View

The view a user is presented after left clicking a day in the Day Display listed in section 2.3.1. The day view will contain the follow basic layout items:

Navigation Arrows - In the top right corner of the window, a left and right arrow will allow the user to navigate forward (right) and backward (left) one day at a time. (See Appendix 5.4 item 3)
Date Display - In the center of the window, the current date selected is displayed in the following format Day as a decimal number, Month, Year. (See appendix 5.4 item 2)
Exit Button - In the top left of the window, an 'X' will appear to the user; left clicking on this leaves Day View and returns the user to Calendar View. (See appendix 5.4 item 1)
Day Edit - Below the Date display the user will be given multiple options to: enter an event name, set time, notes, location, and delete an event (See appendix 5.4 items 4,5,6,7,8)

2.2.5. Event View

The view a user is presented after left clicking an item event in the Calendar View, or left clicking the day edit section in the Day View. This view will allow a user to create a new event, edit an existing event, or remove an existing event.

Name Field - The top of the event view, this will take user input with 'ghost text' displaying the string ("Enter event name") (See appendix 5.4 item 4)
Time Field - Located below name field, this will allow a user to enter start and end times for an event (See appendix 5.4 item 5)
Delete Field - Located below location, this will take the user to a dialogue box asking the user to confirm deletion. If the user confirms deletion the event is deleted, otherwise no changes are made. (See appendix 5.4 item 8)

2.3. Design Decisions

The design structure of the calendar follows a basic calendar design that most people are already familiar with. Our structure was designed with simplicity in mind in order to make it easier for new users to adapt quickly to the interface. The first design choice that was agreed by all team members was having Sunday as the first day of the week. The main reason for this design choice

is due to the fact that most calendars use this design structure. Starting the week with sunday will help us achieve simplicity and familiarity with the software.

The second design choice was the pop out day-menu. The pop out menu was inspired by the calendar application on Mac OS X. The benefits of the pop out window design is that it is minimal and only block a small portion of the screen. It allows for a user friendly environment that is not going to overwhelm the user. Each pop out menu allows the user to either view an event or add a new event for the selected day. In addition, users can use the arrows to move to the next or previous day and close the window by simply clicking on the 'X' symbol on the top left side of the window. These button were placed in the top portion of the window and designed to be big enough in order for users to be able to find and click on easily while not taking too much of the available space. Similarly, users can find the same navigation buttons in the top right corner of the calendar. These buttons have similar functionality to move to previous or next month. Lastly in the monthly view of the calendar, users can clearly see which days have one or more events in them and which days are available.

2.4. Module Interface Specifications

2.4.1. Model

Component	Service	Parameters	Effects of Calling Service
Calendar	A Hashmap to store events	Takes CalendarEvent object.	Given a specific date, it will return a list of events.
Calendar Event	A Custom class to store event data	Takes the data of the event.	Given an event, it will return all of the data of that event.

2.4.2. View

Component	Service	Parameters	Effects of Calling Service
Month View	View that contains days and events	Contains all of the Days and Events within the Days of that month. Uses User Input	Refer to Appendix 1 for functionality.
Day View	View that contains events	Contains the specific events of that day. Uses User Input	Refer to Appendix 2 for functionality.

Event View	View that is editable	Contains the data of that event.	Refer to Appendix 3, 4, 5, 6 for functionality.

2.4.3. Controller

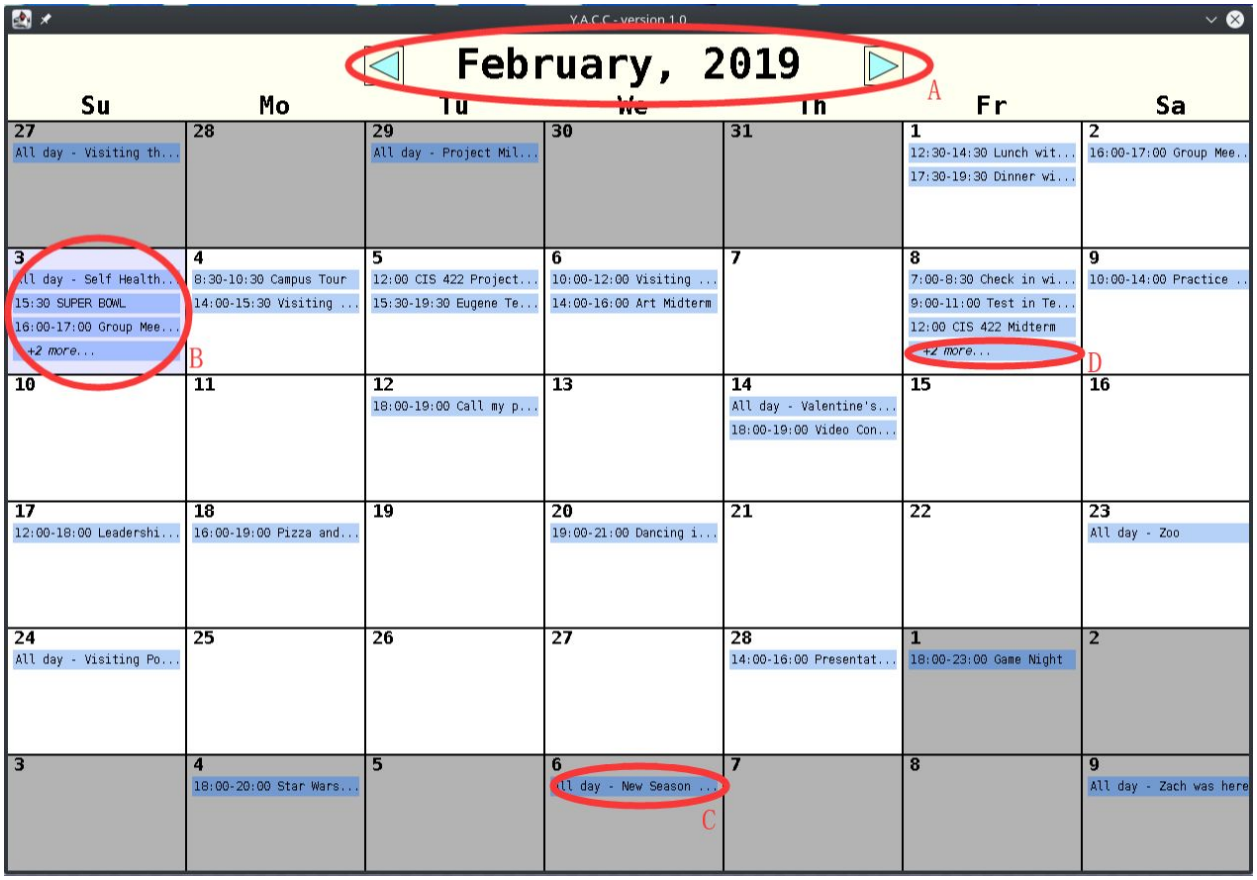
Component	Service	Parameters	Effects of Calling Service
Main	Controls the interaction of the various modules.	None	This service calls and triggers the other components

2.4.4. Input/Output

Component	Service	Parameters	Effects of Calling Service
Read File	Reads in .cal File	Takes a file and FileIO Status module.	This service will read and parse the file. It will utilize the FileIO Status module to flag any issues.
Write File	Writes to .cal File	Takes a file and FileIO Status module.	The service will clear the current file and then write to it with the current data. It will utilize the FileIO Status module to flag any issues.
FileIO Status	Passes message to Read and Write component IO Status	Takes Status Constants module.	The service will raise a flag and pass a message if there is success/waiting/failure on the input/output of the file.
Status Constants	Stores all possible statuses of FileIO	None	This service is used to provide FileIO Status module with the possible flags.
User Input	Passes message along of location of mouse click or keys pressed.	None	Module waits for a mouse click or a key press and pass the information to the Month View, Day View, or Event View.

3. Appendices

3.1. Appendix 1: Viewing the Month



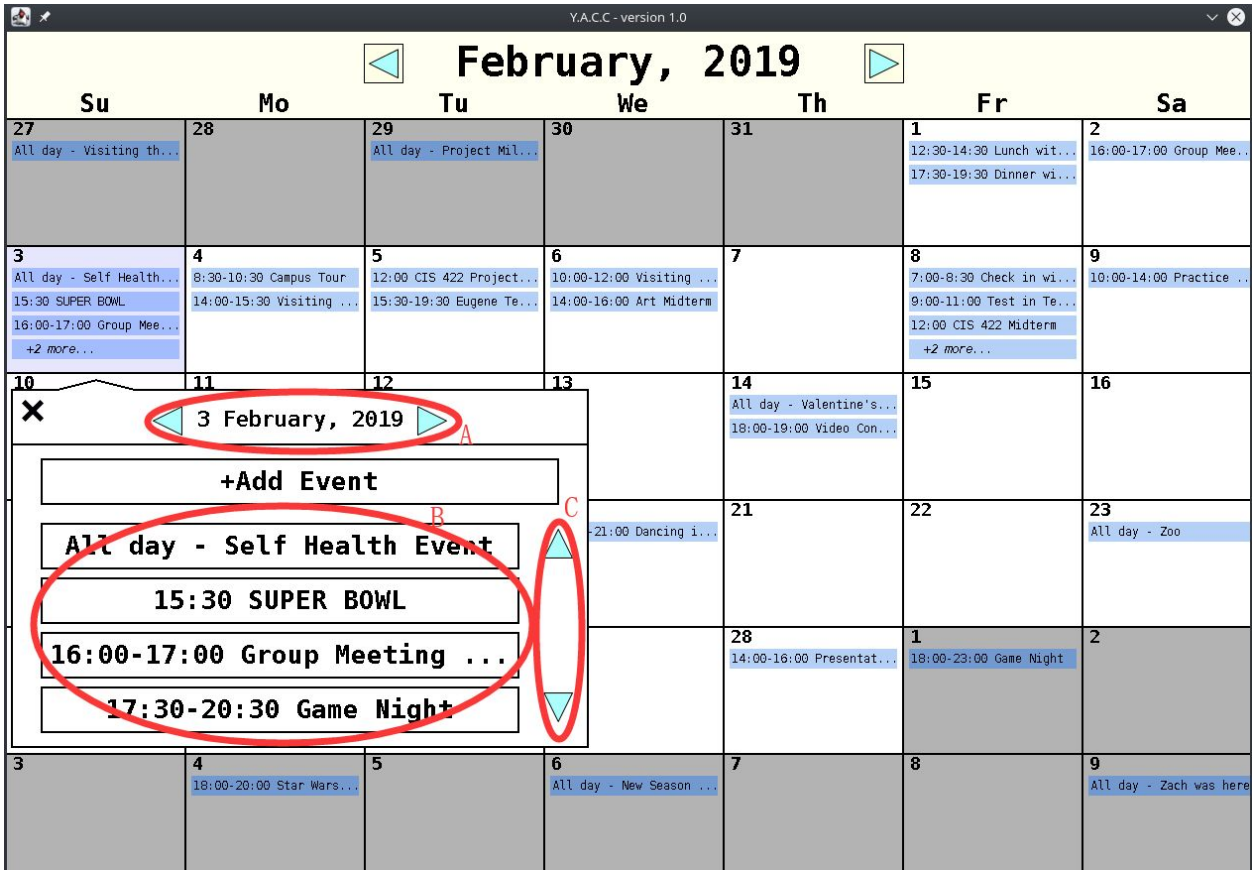
Screen Components:

A. Display the current month and year. User able to switch month by clicking left button or right button.
B. Blue box represents the current date.
C. Next/Previous month days are grey and when clicked will shift to that month.
D. Shows “+X more ..” for more than 4 events in one day.

Keyboard Features:

- PGUP - Will shift the month view to the previous month.
- PGDN - Will shift the month view to the next month.
- HOME - Will shift the month view to the current month.

3.2. Appendix 2: Viewing the Events of the Day



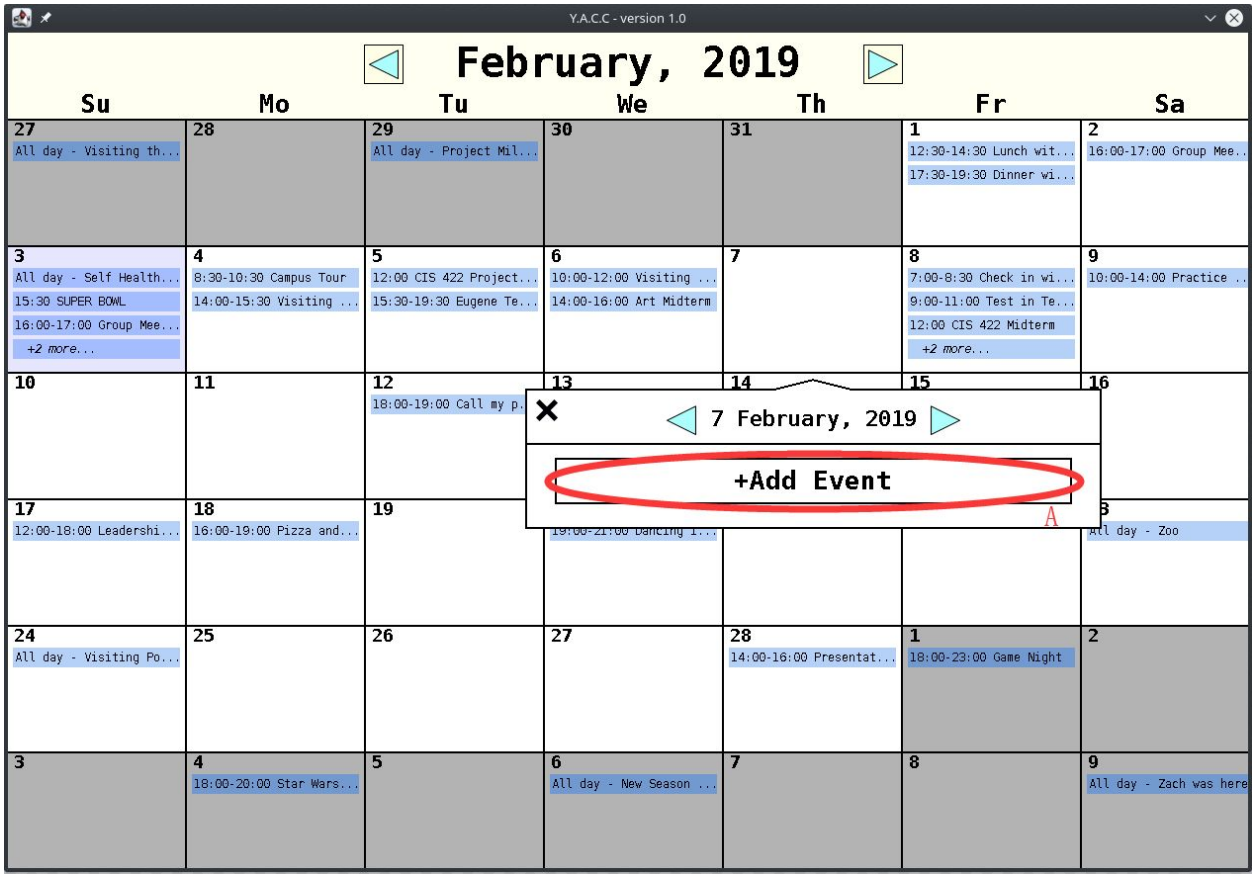
Screen Components:

A. User able to change the date by click left and right button.
B. Display the events of selected date.
C. Use up and down button to check more events on that day.

Keyboard Features:

- LEFTARROW - Will move the day view to the previous day.
- RIGHTARROW - Will move the day view to the next day.
- ESC - Will exit the day view.

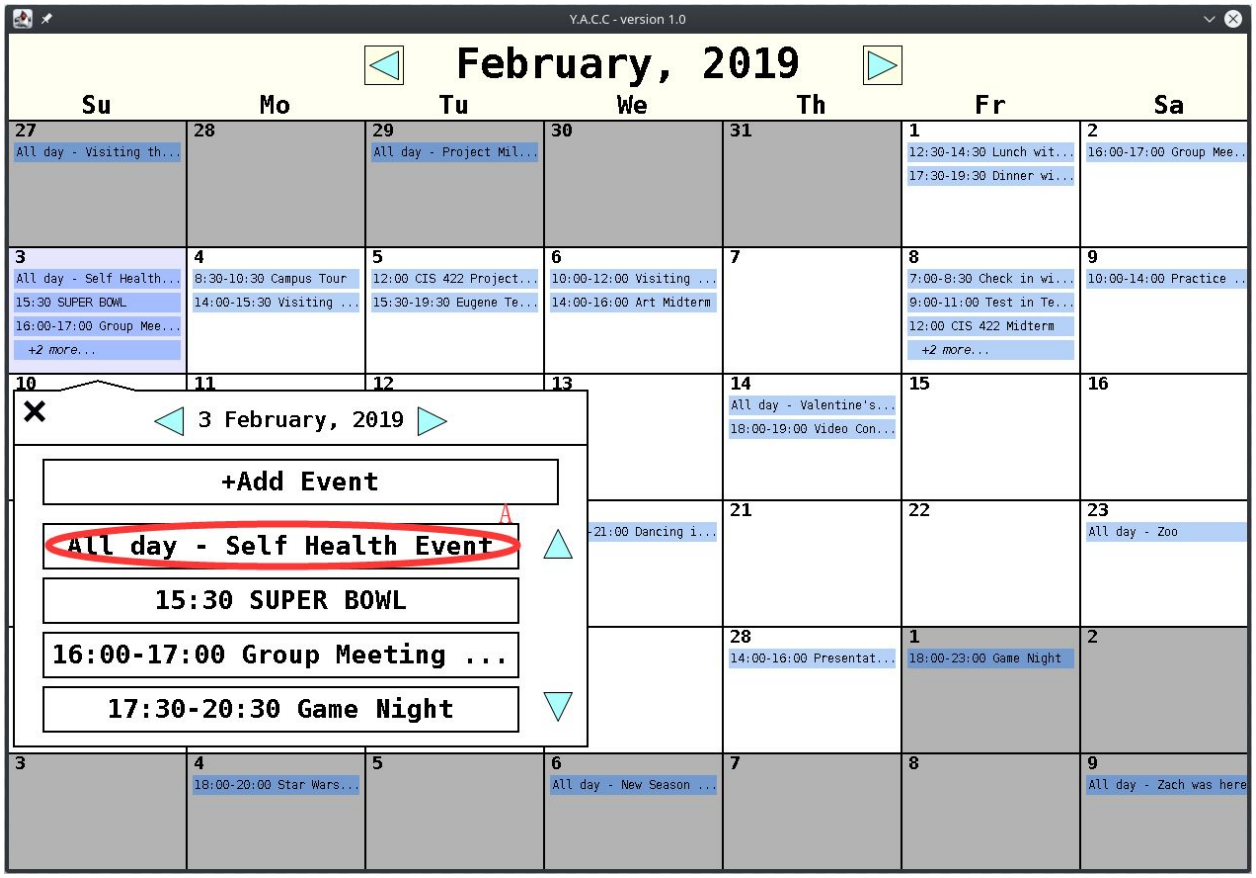
3.3. Appendix 3: Adding an Event



Screen Components:

A. Click “+Add Event” button to add event, for editing the details of the event go to page Editing an Event.

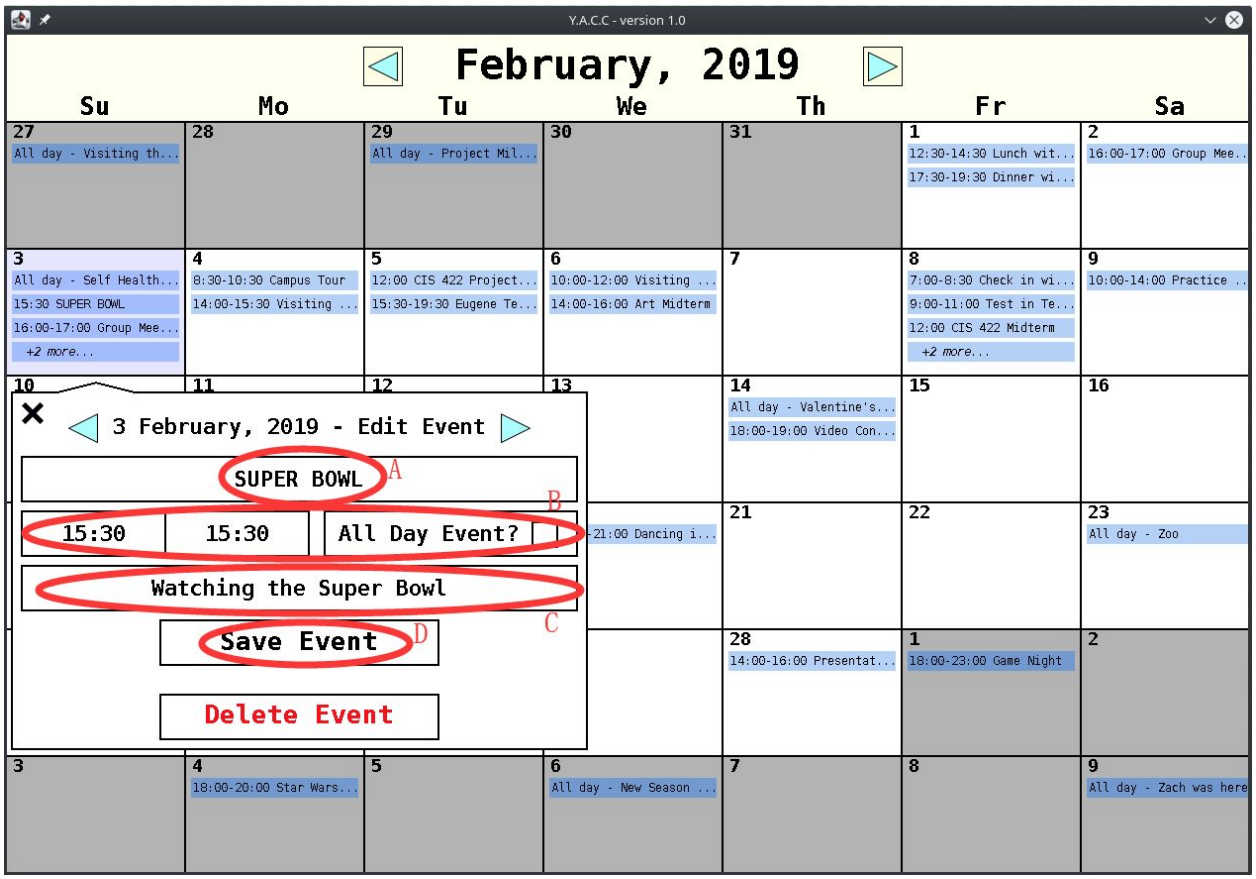
3.4. Appendix 4: Selecting an Event



Screen Components:

A. Click the event to select the event. User can either edit or delete the event.

3.5. Appendix 5: Editing an Event



Screen Components:

A. Edit event name.
B. Edit event time duration. User can click the button to make it 24 hours duration.
C. Edit description of the event.
D. After edit it, User need to click the button to save the edited details.

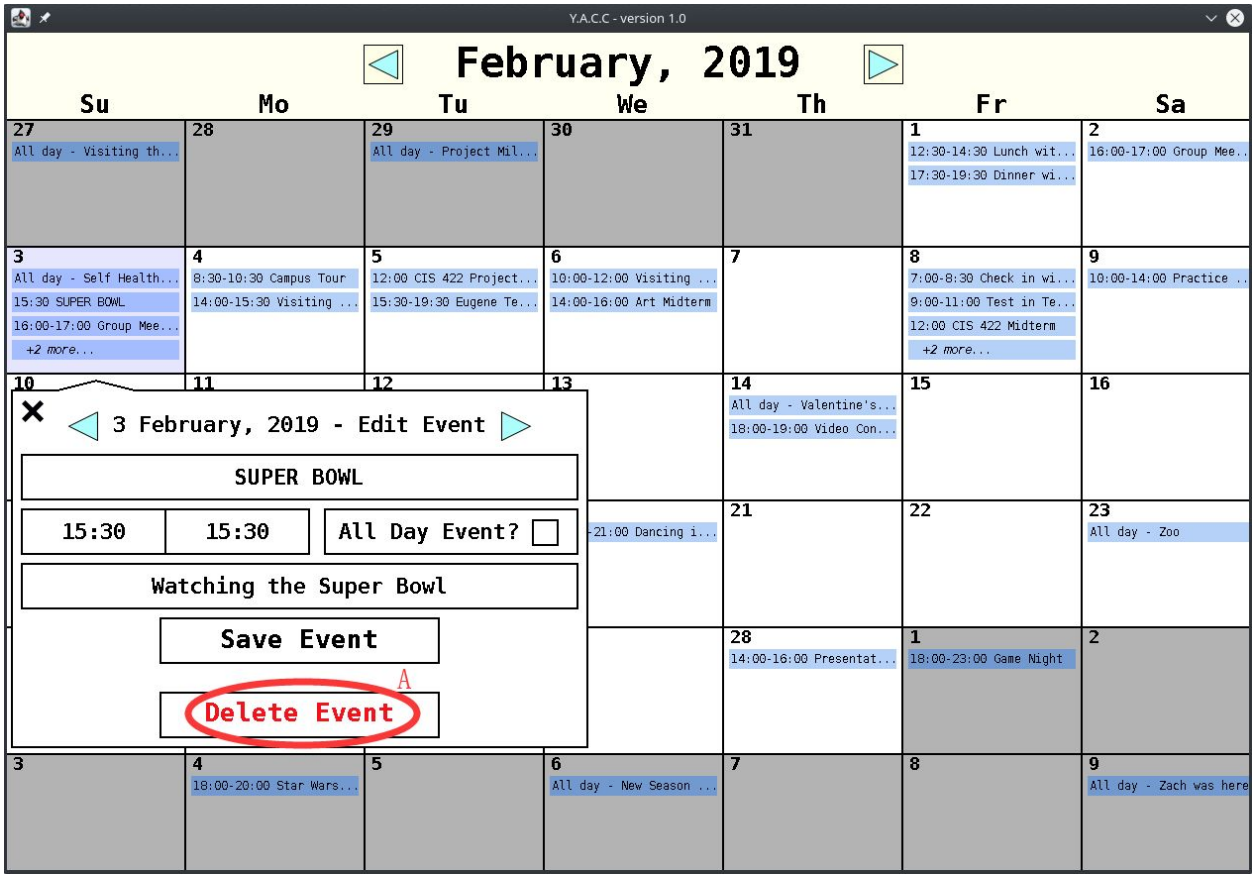
Keyboard Features:

- TAB - Will cycle through the editable features of an event.
- ESC - Will exit the event editor.
- LEFTARROW - Moves event to previous day.
- RIGHTARROW - Moves event to next day.

Types of Events

- All Day Events - Can Click “All Day Event?” button to set as all day event.
- Single Time Events - Can type same time for both start and end for single time event.
- Time Span Events - Can type different times for stand and for time span event.

3.6. Appendix 6: Deleting an Event



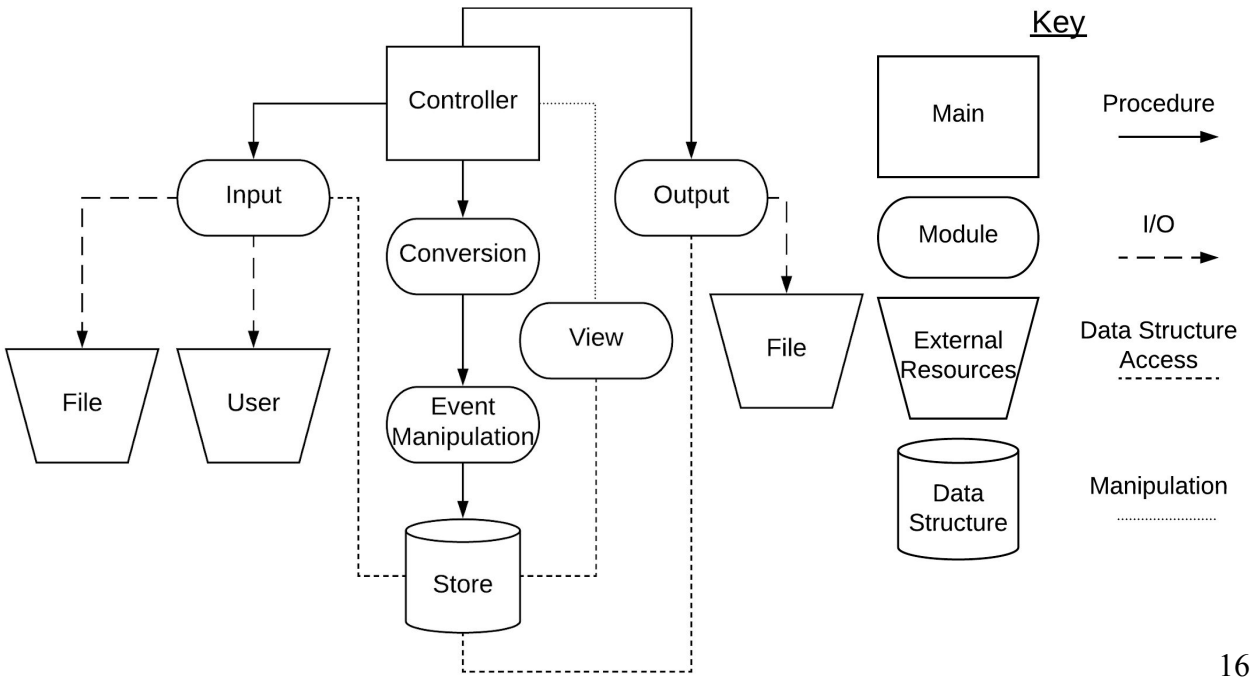
Screen Components:

A. Click button to delete the selected event.

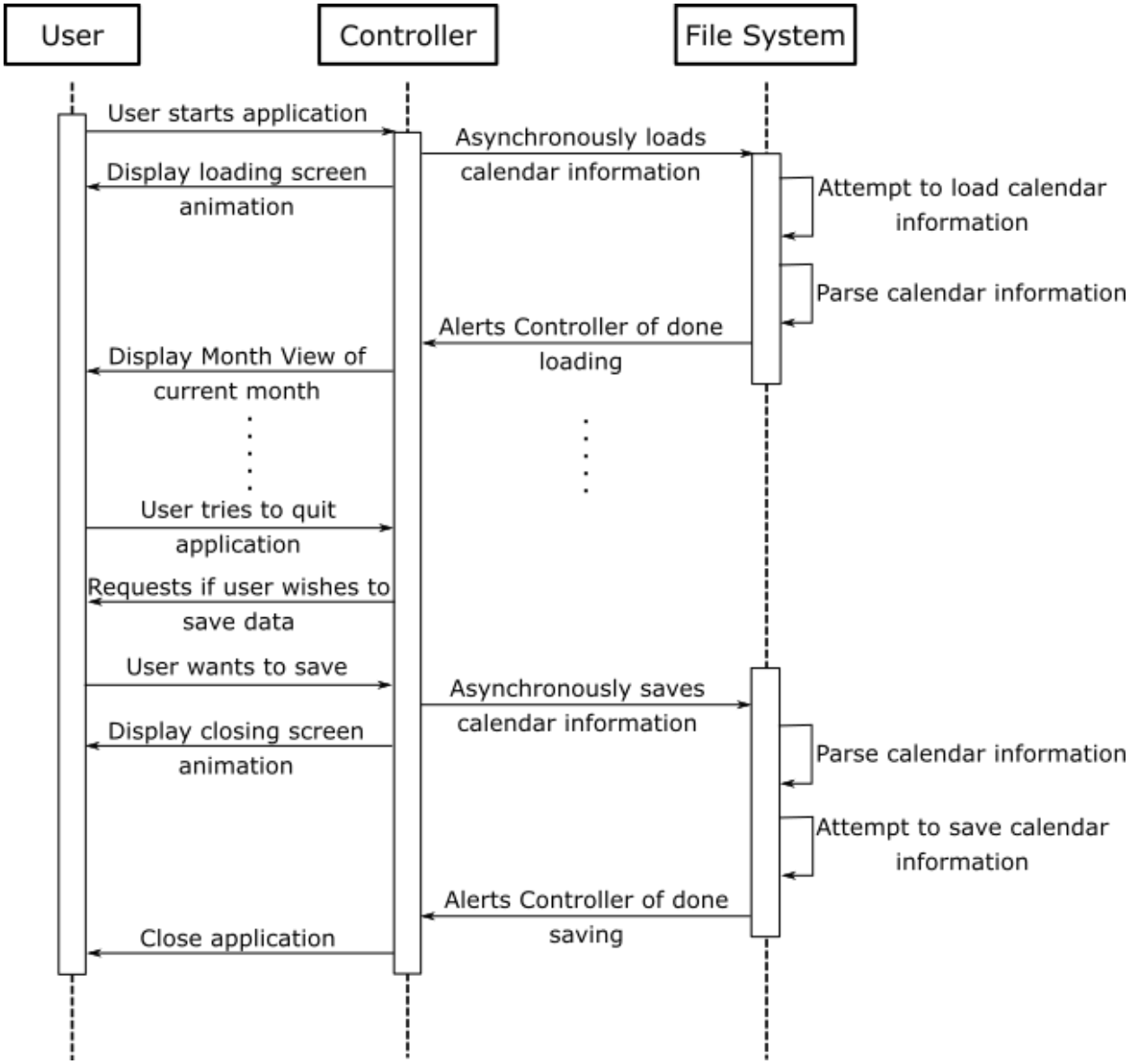
3.7. Appendix 7: Sample File Structure

```
{
     days:[
          <INDEX>:{
               month: <MONTHNUMBER>,
               year: <YEAR>,
               day: <DAYOFMONTH>
               events:[
                    <INDEX>:{
                         note: <STRING>,
                         start_hour: <INT>,
                         is_all_day: <BOOLEAN>,
                         end_minute: <INT>,
                         title: <STRING>,
                         end_hour: <INT>,
                         start_minute: <INT>
                    }
               ]
          }
     ]
}
```

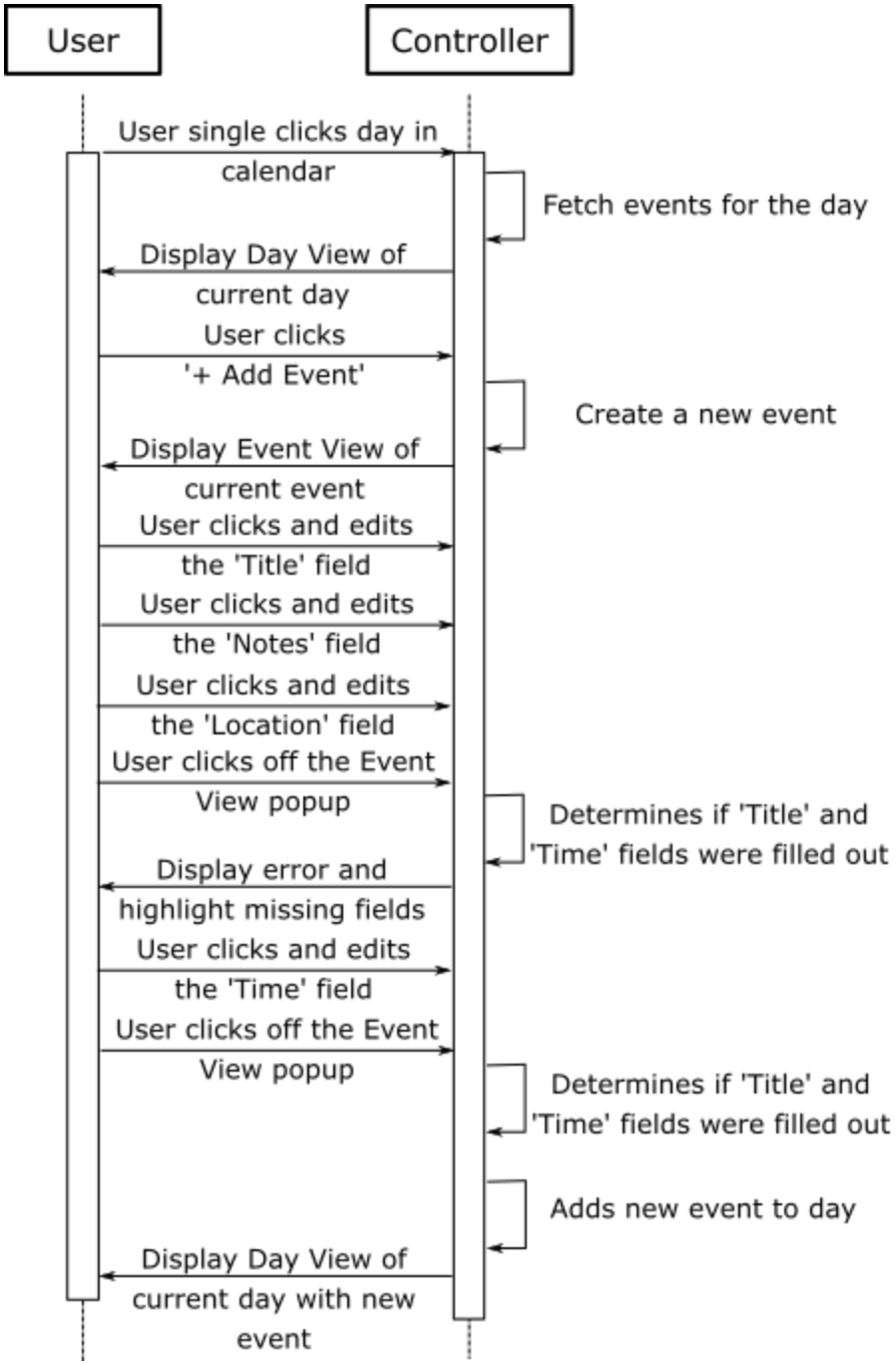
3.8. Appendix 8: Static Model and Key



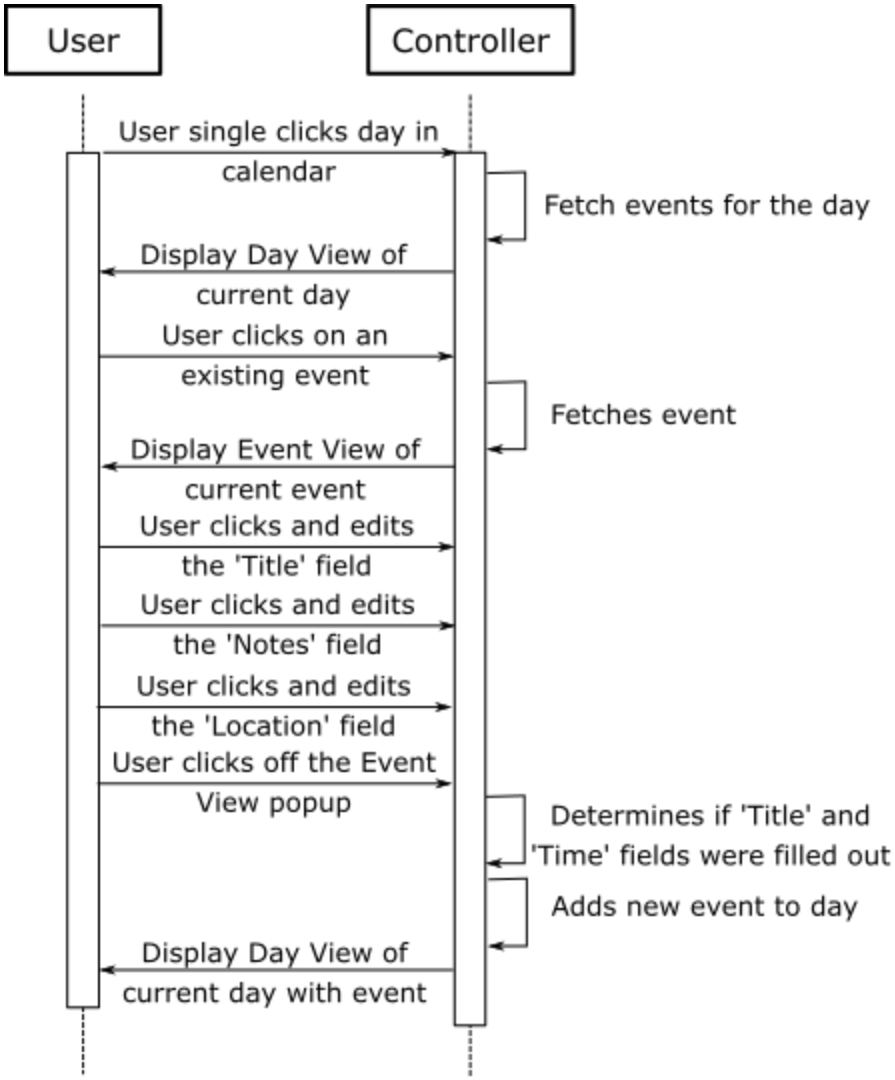
3.9. Appendix 9: Dynamic Model - Open/Close



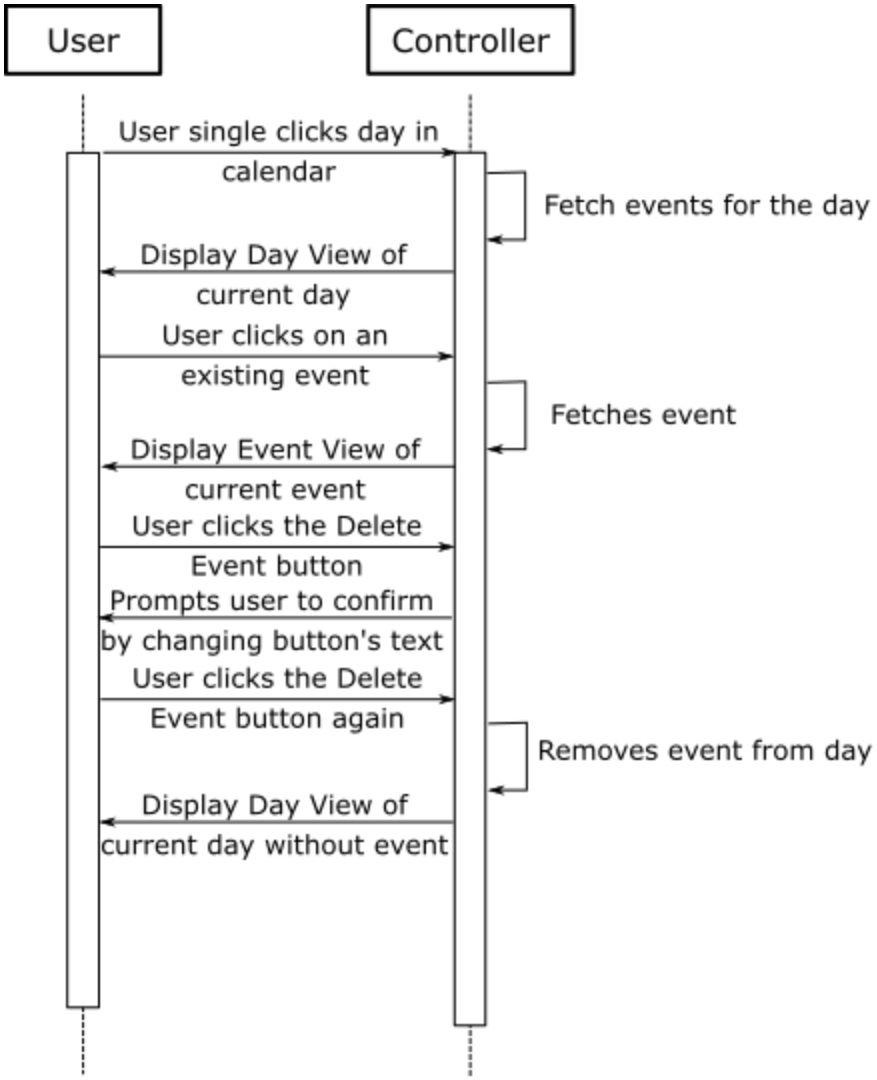
3.10. Appendix 10: Dynamic Model - Add Event



3.11. Appendix 11: Dynamic Model - Edit Event



3.12. Appendix 12: Dynamic Model - Delete Event



3.13. Appendix 13: Dynamic Model - Day View