CIS 422

Software Requirements Specification (SRS) for YACC

The document in this file has been adapted from the IEEE Guide to Software Requirements Specifications (Std. 830-1993)

CIS 422

Team Number 6

499ms

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

Software Requirements Specifications

Version 1.2 Date: (02/04/2019)

Table of Contents

1. Introduction

1.1. Purpose

This Software Requirements Specification (SRS) describes the computer program YACC that is currently being designed and built at the University of Oregon by students: Benjamin Yin, Chase Craig, Noah Palmer, Refael Yehuda,and Zachary Bower. YACC will implement features explained in the Project 1 description assigned by A. Hornof.

YACC will solve an end users organizational needs by providing a calendar application with the ability to add, edit, and remove events specified by the user. As the software evolves this document will contain the current specifications for said features. This document is for developers and end consumers of YACC. This document will allow all developers of YACC to organize, plan and track all aspects of development. This SRS will outline objectives of the software including functional/non functional features, an overview of expected user experience, a clear measure of accountability, and clear system requirements.

1.2. Scope

This software will be a calendar application for end users to store, edit, add and remove events. This software will be designed to maximize end users productivity while keeping a cohesive design. By focusing on these endpoints from the beginning of the development cycle this will allow YACC to meet an end users needs.

1.3. Definitions, Acronyms, and Abbreviations

This document will reference multiple acronyms definitions for each are defined below:

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.

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.

1.4. References

References related to the internal development

(1) **SDS:** https://docs.google.com/document/d/1WQpj-drVOHYIO6baY02JZA9RLvReq6zHXqaqdEbQN-Q/
(2) **IPP** : https://docs.google.com/document/d/1ZszO4mlShik_dEfztpab-m4zGZj5UHmMHrU0SFNLGkg/edit
(3) **CSD**: https://docs.google.com/document/d/1KL8Q_9lhUDCnlcPMZtaPfaDTiaKeyBGpLgd7bN3cE5s/
(4) **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) **NRL Dual Task SRS:** https://www.cs.uoregon.edu/research/cmet/Multimodal/files/SRS/070304/dualtask.html
(2) **SRS Template:** https://web.cs.dal.ca/~arc/teaching/CS3130/Templates/SRS%20and%20Project%20Plan%20Templates/SRS-template2.doc

1.5. Overview

Section two of this document gives an overview of the functionality of YACC. The intended audience for this section are end users of YACC. It describes the functional and nonfunctional requirements an end user should expect upon development cycle end. The functional and nonfunctional requirements are used to establish the technical specifications presented in section three of this document.

Section three of this document give an overview of the technical aspects of the project. The intended audience for this section are developers. This section shall describe in detail the functionality of the product.

2. Overall Description

2.1. Product Interaction

YACC will have one main interface for end users to interact with. The initial screen will have the current date within the current month displayed in a matrix. The month and year will be centered in this matrix display, with a left justified arrow and a right justified arrow.

From this main interface the user will be able to navigate to the previous month by clicking on the left arrow, the next month by clicking on the right arrow, or select the month and year to entered the date desired by the user.

The end user will have the ability to add or edit any event from this main view. This is accomplished by the user clicking on sections of the main view.

A detailed overview of the user software interaction are available in both the SDS and section 2.2.

2.1.1. System Interfaces

YACC is a GUI driven application. The user will perform all interactions described in the SDS and section 2.1. The GUI design and interactions are available in the SDS.

2.1.2. Software Interfaces

YACC is designed for and tested with the following software requirements. Deviating from these requirements may give undesired results.

(1) Java Runtime Environment
(2) GNU/Linux Operating System

2.1.3. Communication Interfaces

The end user will communicate through a mouse and keyboard which YACC will take as input through inputs located with the GUI.

YACC shall communicate to the user through a standard monitor or display. The display specifics are outlined in section 3.

2.1.4. Memory Constraints

YACC requires 128MB of memory
YACC requires at least 200 MB of disk space

2.2. Product Use and Case Scenarios

All examples assume the end user meets the minimum requirements for YACC. Unless otherwise specified the users assumed starting location is the main interface displayed on program startup described in section 2.1.

User wants to create an event

(1) The user issues a create event command
(2) The user must input the following required fields
    (a) Start/End date
    (b) Start/End time
(3) The user may input the following optional fields
    (a) Event Description
    (b) Event Name
    (c) Event Type
    (d) Recurring event
    (e) All day event
(4) The user issues a confirmation command

User wants to edit an event

(1) The user issues an edit event command to any event scheduled
(2) The edit must contain the following required fields, although all or none can be edited
    (a) Start date
    (b) End date
    (c) Start time
    (d) End time
(3) The user may input the following optional fields
    (a) Event Description
    (b) Event Name
(4) The user issues a confirmation and is notified

User wants to delete an event

(1) The user navigates to the desired event to be deleted
(2) The user issues a delete command
(3) The user gives confirmation of the action
(4) The event is deleted if confirmation is given. The event is preserved otherwise

User wants to save changes

(1) The user issues a save command
(2) The user confirms save command
(3) The users calendar event state is saved on confirmation, otherwise no changes are made

User wishes to exit the program

(1) The user issues an application exit command

(2) The user may save if unsaved changes are detected, cancel the application exit, or confirm
(3) If confirmation is received the application terminates.

2.3. User Characteristics

The end user should have a basic understanding of the GNU/Linux operating system, with the ability to change user permissions.
The end user will expect YACC to organize the end users events if the end user updates YACC consistently. The end users satisfaction will be proportional to their use of YACC.

2.4. Constraints

To ensure YACC performs as described in section 2.1. The end user is expected to meet or exceed the software, hardware, and communication requirements described in section 2.1.

2.5. Assumptions and Dependencies

YACC consists of functional and nonfunctional requirements described in section 3. An end user should expect all requirements to work as documented given the software, hardware, and peripheral requirements given in section 2.1 are met.

3. Requirements and Additional Specifications

3.1. Functional Requirements

The following features are defined for a viable product and thus are defined to be absolutely required.

3.1.1. Data Collection and Output

FR 1: Time Format

- The system shall ensure all dates and times are stored in a consistent format (such as “DD/MM/YYYY HH:MM”).

FR 2: Data Entry

- The system shall accept any valid date and time combination from 01/01/1970 00:00 to 31/12/2119 23:59.
- The system shall alert the user if their input is malformed or out of range.

FR 3: File Saving

- The system shall store all events in a local file (referenced hereafter as “save file”) on the user’s file system.
- The system shall store a version identifier with the save file for version matching.
- The system shall store a event starting token at the start of every event in the save file.

- The system shall accept any save file that has originated from a previous version of the application.
- The system shall, if it determines that the save file in the user's file system is lacking in required fields, it is to treat the event as malformed and it is to be skipped.

3.1.2. Events

FR 4: The system shall allow the user to create new events by specifying the date, a title, and the time the event shall occur on.

- The system shall reject the creation of a new event if it is missing either the date, the title or the time, however it should not clear what the user has already added to the event.

FR 5: The system shall define the minimum requirements for an event as an object holding a title (a string), a valid date and a valid time.

3.1.3. Program Interface Properties

FR 6: The system shall ensure that the program interface accept minimal window operations (such as minimize window and close window).
FR 7: The system should display any "week" like object as starting on Sunday.
FR 8: The system should keep the program interface limited to becoming no smaller than 600x800.

3.1.4. Calendar Operations

FR 9: The system shall allow for the addition of an event, which will be stored until saved.
FR 10: The system shall allow for the selection of an individual event.
FR 11: The system shall allow for selected events to be modified.
FR 12: The system shall allow for events to be deleted.

3.3.2. Calendar Layout

FR 13: The software shall contain two views to display to the user, a month overview and a day overview.

- The software shall ensure that in the month overview that the current day of the month is made differentiated from the other days of the month.
- The software should ensure that in the month overview that any days that have passed in the month are differentiated from days in the future.
- The software should allow for the selection of events by left clicking on the occurrence.
- The software should allow for adding an event by single left clicking a day in the month overview. The new event should have the date autofilled.

FR 14: The software shall give an indication to the user if more events are present than can be displayed

- The software shall have a way of presenting the user with overflow events.
- The software shall allow for the user to return to month overview through a simple key press (such as <Esc>) and through clicking an icon (such as a return arrow).

- The software shall allow for navigation through a simple key press (such as <Left Arrow> or <Right Arrow>) or through clicking an icon.
- The system shall ensure that any events listed shall be sorted in according to the time of which they occur in. In events of ties, the system will sort them in accordance to their titles.

3.2. Nonfunctional Requirements

The following features are defined for a viable product and thus are defined to be absolutely required.

3.2.1. Software Performance

NFR 1: The software shall not stall for more than 500 ms on any operation blocking user interaction besides initialization.

NFR 2: The software shall not require more than 256 MB of memory to run.

NFR 3: The software shall not require more than 200 MB of space for source code once compiled.

3.2.2. Hardware

NFR 4: The software will run on GNU/Linux Operating System.

NFR 5: The software shall be developed to run on most machines that can handle java environments that contains a native display.

3.2.3. Ease of Development

NFR 6: The software will be designed in a modular fashion such that simple changes will remain localized to a single system component.

3.2.4. Output File

NFR 7: The system shall use UTF-8 for all output files.

3.2.5. Security with Input Format

NFR 8: The system shall only allow standard ASCII characters to be typed into any textfield.

3.2.6. System Documentation

NFR 9: The SRS (the document you are reading now) must be editable by all developers. A numbering and indentation scheme will be used that permits easy reference, update and modification by all developers. All versions of the SRS will list the authors and the date.