­

Software Requirements Specification­

for

Pro-Crastinator

Version 1.0

Prepared by

Group Name: Once We Were Programmers

|  |  |  |
| --- | --- | --- |
| Spencer Ross | 11686231 | s.ross@wsu.edu |
| Sarah Mathes | 11065825 | sarah.robison-mathes@wsu.edu |
| Kyle Stennfeld | 10722379 | k.stennfeld@wsu.edu |

|  |  |
| --- | --- |
|  |  |
| Date: | December 13, 2019 |
|  |  |
|  |  |
|  |  |

Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behaviour Requirements 6

4 Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

5 Other Requirements 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Sarah Mathes, Spencer Ross, Kyle Stennfeld | First version of system require specification requirements document. System is in design phase. | 10/25/2019 |
| 2.0 | Sarah Mathes, Spencer Ross, Kyle Stennfeld | Second version of system requirement specification requirements document. System is in implementation phase. | 12/13/2019 |

# 

# Introduction

College students are in need of a concise schedule organizer in order to meet the demands of their busy schedules. The goals of this system are:

1. To be a time management system that is unencumbered by unnecessary features or complicated graphic user interface.
2. To improve the lives of students by organizing their classes, homework, and tests.

This system is intended to be a simple interface that the student can customize for their own workload.

This document outlines the high-level requirements for this system. The use-case of the system is also included, along with the overall description, and other non-functional requirements.

## Document Purpose

This document describes the requirements for the web application Pro-Crastinator version 1.0. The first section serves to introduce the scope of the system, its audience, and an overview of the document itself. The second section details overall description of of the system, including the operating environment, system functionality, and user documentation. The third section details the system’s hardware and software interfaces, and user interfaces.

The scope of this document encompasses all features of the system, and includes the system log-in, class management, grade calculation, and assignment organization. The system log-in is the process by which the student logs into their schedule. Class management is the sub-system where a student adds a class to their weekly schedule. Grade calculation is the sub-system where a student can edit the breakdown of their class grades and calculate a theoretical final grade based on the syllabus and submitted assignments. Assignment organization is the sub-system by which a student can add an upcoming assignment or test to their schedule.

## Product Scope

The software described in this document is the web application Pro-Crastinator. Pro-Crastinator is a schedule planner intended to be used by students in order to manage their school workload. The software system saves a user’s information, primarily their name, classes, and assignments, and recalls them upon the user logging in. Once logged in, the user can view a calendar of their classes and assignments, edit their class schedule, add a new assignment, edit existing assignments, and calculate their class grade.

Although many student planner applications exist, one of the main goals of Pro-Crastinator is to have a simple graphic user interface where the student can easily edit their class schedule and assignments from day to day. Upon logging in, the student is presented with options to either view, edit, or add to their class workload.

One of the core advantages to this system is that the user will not have to navigate through several menus, calendars, or course webpages in order to organize their workload. This lightens the burden of needing to mentally keep track of multiple assignments and tests across a number of classes. The system is a simple way to organize and then visually see the user’s school workload.

## Intended Audience and Document Overview

The primary intended audience for this document are developers of the system Pro-Crastinator. The purpose of this document is to formally state the functionality of this software system and its subsystems. Developers can refer to this document during development in order to solidify core goals, and to maintain consistency. The section that is most relevant to developers is section 3.1.3, Software Interfaces, as well as 2.2, Product Functionality.

Similarly, project managers should use this document to drive consistency as the system is developed and updated. Project managers should primarily refer to the section 2.1, Product Perspective, and 2.2, Product Functionality, in order to keep the main goals of Pro-Crastinator in mind. They should also familiarize themselves early in development with the Use Case View in section 3.3.1, which outlines the functionality of the system and the relationships of its subsystems.

The secondary audience for this document is the professor of CS 320 at Washington State University, Fundamentals to Software Engineering, Dr. Xinghui Zhao, who will be reviewing and grading the development of Pro-Crastinator. The section that is most useful for Dr. Zhao is section 3.3.1, the Use Case View, because it is the easiest way to visually understand the relationships between the system, its users, and its subsystems. Additionally, section 2.2, Product Functionality, will be useful for Dr. Zhao to refer to during system development, and when grading the final version of Pro-Crastinator.

This document is also intended for product testers of Pro-Crastinator. If at any point product testers have questions about the intended functionality of the system, they should refer to this document. Product testers should start with the use case view, section 3.3.1 which will give them an overall view of the system’s operation. They should then move on to section 3.1.1, User Interfaces, which provides guides to the user interface and its operation.

## Document Conventions

Formatting coventions in this document follow the standard IEEE formatting requirements. Standard font for this document is Arial, size 11, with one inch margins. Lines are single spaced, with double spaces between paragraphs. New paragraphs are not indented. Comments are italicized.

# Overall Description

## Product Perspective

The Pro-Crastinator is a new, self contained product that utilizes existing Javascript libraries. The user is able to update their personal info, including their username, email address, and password. They can then add a class to their schedule, which Pro-Crastinator saves in the server. After a class has been added to their schedule, the user can create an assignment, of type homework or test, under that class. The assignment includes the class it belongs to and the date that its due. Pro-Crastinator also saves this information to the server. The system uses the class and assignment information to generate a calendar of class times and due dates. It also generates reminders for assignment due in the next twenty-four to forty-eight hours, which is delivered back to the user. The user also has the option to enter the grade breakdown for their class, so that the system can calculate a final grade based on theoretical assignment grades.

## Product Functionality

The first function that the user can perform upon creating a profile and logging in is setting up a class. When a new class is created, the user can save:

* The name of the class
* The name of the professor
* Contact information for the professor, such as an email address
* If it’s an in-person class or an online class
* If it’s an in-person class, they can add the start and end times and days that class is held
* Optionally, they can edit the final grade breakdown by percentages. For example, they can enter that attendance is worth 10%, exams are worth 30%, homework is worth 30%, and the final is worth 30%.

Once a class is created, the student can add an assignment. When they choose to add an assignment, the user edits:

* The class that the assignment belongs to
* The type of assignment: homework, or test
* Due date of the assignment (including date and time of day)
* Optional: notes about the assignment

After classes and assignments have been added, the system populates a calendar with visual representations of the classes and assignments. In the calendar view, the student can filter by:

* Assignment type
* Due date
* Class

The student can also click an assignment or class in the calendar view to see more information.

If the student has edited the grade breakdown for a class, the student can calculate their final grade based on theoretical grades of an assignment. This grade calculator is one of the options on the main landing page.

## Users and Characteristics

The users that Pro-Crastinator is designed for are middle schoolers, high schoolers, undergrad students, and grad students. The system’s features are applicable to students of all levels. The users that Pro-crastinator can help the most are undergrad and graduate students, because their classes schedules vary the most. These two groups of students also have the most demanding workload, and need to organize it around family and work life. The format of this application might also be useful to educators, but they are not an intended user for Pro-Crastinater.

This system may be used by a younger audience with limited technical skill, but the application will feature a brief tutorial to facilitate application use. Pro-Crastinator will also use non-technical vocabulary in order to make it useful to younger students as well.

The feature that is most helpful for students is the calendar view, which displays their class schedule and assignment due dates. This allows for a high-level view of their workload in an interface that is visually simple. It is also a way to consolidate assignments across multiple classes and institutions. This way, the student does not have to navigate through multiple menus, class websites, or third-party platforms.

## Operating Environment

Pro-Crastinator is a web application. It is intended to be run on a desktop or laptop computer. The computer must have internet connection, either wired or wifi, in order to access Pro-Crastinator. No downloads will be required to use the system, but the internet access should be reliable. The machine must also have an internet browser installed, with up-to-date Java plug-ins. Pro-Crastinator aims to be compatible across all internet browsers. It will not be optimized for mobile use on smart phones or on tablets.

## Design and Implementation Constraints

The primary constraints to this software is the limited experience of its developers. The developers have no expertise in web development, and no prior experience with Meteor, MongoDB, or React. As a result, a significant portion of this software’s development timeline has been spent researching. The developers will be responsible for maintaining the software after its completion. This include, but is not limited to, updating packages and maintaining code as updates are released.

The software is constrained by the Meteor packages that it uses. These currently include IronRouter, Semantic UI, Scheduler, and MongoDB. None of these packages have any technical requirements, although Semantic UI has compatability issues with other commonly used Meteor packages such as Less and PostCSS, or other packages used to render HTML and CSS.

## User Documentation

The instructional material for this software will be inside a ‘Help’ section on the website itself. A user does not have to log in or have an account in order to view the ‘Help’ material. This section includes information about the software’s main features, instructions on making an account and signing in and out, how to reset a password, and how to add and remove items to the user’s class and assignment list. Additionally, a small question mark icon will be placed on each page that, when clicked, provides instruction on how to use that page’s features. The functionality of the software is simple enough that an instruction manual would be unnecessary.

## Assumptions and Dependencies

If Procrastinator is sold to schools and universities as a productivity tool or communication tool between educators and students, it is assumed that there would be thousands of accounts created. In this case the developers may have to use an account manager other than Accounts UI, which currently handles all log-in credentials and user accounts. The developers would also have to purchase server space and maintain a server in order to provide reliable service to their users. In any case, there is no cap on the number of users or accounts that will be created.

It is also assumed that Procrastinator will be used on desktop machines, and not in mobile environments. If opened in a browser on a mobile device, the software will be formatted for mobile dimensions but also state that it is best used on a desktop device.

# Specific Requirements

## External Interface Requirements

### User Interfaces

Upon loading the software, the user is prompted with a description of the software, and also to create an account and log in. Software content, aside from instructions, is not displayed until the user is logged in to their account. After logging in to their account, the main content is loaded. This includes a menu of four items on the left hand side of the screen: Classes, Assignments, Calendar, and Calculator. The active menu item is highlighted. The ‘Classes’ page lists all created classes, along with buttons to quickly edit or delete them, and an ‘Add Class’ button that takes the user to another page to add a new class to the list. The ‘Assignments’ page has a very similar user interface and functionality. The ‘Calendar’ page shows all created classes and assignments in a week-view of a calendar. The ‘Grade Calculator’ provides a form interface to enter a current overall grade, an assignments percentage of the final grade, and the desired final grade in the class. On pressing ‘submit’, the software reports what grade the user needs on the assignment in order to get their desired final grade. All graphical user interface components are Semantic UI components, and are assembled with HTML and CSS. Form components are generated with Meteor.

### Hardware Interfaces

The software is designed to be viewed on an internet browser on a desktop machine. It can be viewed on a mobile device, but the user interface is designed and best viewed on a desktop device. The Meteor libraries used in this software are optimized for desktop browsers and there may be incompatability if used on a mobile device. There is no library required for the software to operate on different interfaces.

### Software Interfaces

This software uses several Meteor packages, including Semantic UI, IronRouter, AccountsUI, MongoDB, and Scheduler. These packages have been installed in the Meteor project. AccountsUI facilitates account creation and maintenance. Scheduler is a calendar interface that saves appointments and displays them in their assigned time block. Data is shared across several components. When an assignment is created, the user selects which class it belongs to, from a list of created classes. The user can also filter the list of assignments by class. In the calendar page, all created classes and assignment due dates are displayed in the Scheduler calendar interface. No data is displayed until the user logs in to their account via the software from AccountsUI.

### Communications Interfaces

Account creation is done by creating an account with an email address as the username, and a password. This email is then used for all communications with the user. If changes or updates are made to the software, the user is notified via this email. They are similarly notified if there is planned outages for server maintenance, etc. If the user needs to recover their password its done through email communications. Account encryption is handled by the AccountsUI package.

## Functional Requirements

*< Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. This section is the direct continuation of section 2.2 where you have specified the general functional requirements. Here, you should list in detail the different product functions with specific explanations regarding every function.*

*TO DO: Break the functional requirements to several functional areas and divide this section into subsections accordingly. Provide a detailed list of all product operations related to these functional areas.*

## Behaviour Requirements

### Use Case View

<A use case defines a goal-oriented set of interactions between external actors and the system under consideration.

TO DO: Provide a use case diagram which shows the entire system and all possible actors. Do not include detailed use case descriptions (these will be needed when you will be working on the Test Plan), but make sure to include a short description of what every use-case is, who are the actors in your diagram.>

# Other Non-functional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

TODO: Provide relevant performance requirements based on the information you collected from the client. For example you can say “1. Any transaction will not take more than 10 seconds, etc…>

## Safety and Security Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied. Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements.

TODO:

* Provide relevant safety requirements based on your interview with the client or, on your expectation for the product.
* Describe briefly what level of security is expected from this product by your client and provide a bulleted (or numbered) list of the major security requirements.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

TODO: Use subsections (e.g., 4.3.1 Reliability, 4.3.2 Portability, etc…) provide requirements related to the different software quality attributes. Base the information you include in these subsections on the material you have learned in the class. Make sure, that you do not just write “This software shall be maintainable…” Indicate how you plan to achieve it, etc.>

Appendix B - Group Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist the Teaching Assistant to determine the effort put forth to produce this document>