

CS 320 Course Project Final Report

for

The Pro-Crastinator

Prepared by

Group Name: We Came As Programmers

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

|  |  |
| --- | --- |
|  |  |
| Date: | October 25, 2019 |
|  |  |
|  |  |
|  |  |

Contents

Contents ii

1 Introduction 1

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

2 Design 2

2.1 System Modeling 2

2.2 Interface Design 2

3 Implementation 3

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

4 Testing 4

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

5 Analysis 5

6 Conclusion 6

Appendix A - Group Log 7

# 

# Introduction

## Project Overview

*The Pro-Crastinator is a student planner web application. It allows a users to add classes, coursework, and events to a profile. There is also a feature that will calculate grades and even calculate how a weighted grade will affect the total grade for the class.*

*Pro-Crastinator uses an account system to keep track of which user’s data should be visible. The account also helps segregate each user’s data in their own profile and not visible to others.*

## Definitions, Acronyms and Abbreviations

<Define all the terms necessary to properly interpret the report, including acronyms and abbreviations.

TO DO: Please provide a list of all abbreviations and acronyms used in this document sorted in alphabetical order.>

* App = application
* API = application program interface
* CSS = Cascading Style Sheet
* DHTMLX – package and imports used to create scheduler
* DB = database
* HTML = Hyper Text Markup Language
* Iron-Meteor – package used to implement routing
* JS = JavaScript
* Meteor – javascript framework for easily creating web apps with universal code for cross-platform compatibiltiy
* MongoDB – package used to create backend database
* Routing – linking one app to another or having one part of an app redirect to another app
* UI = user interface

\*\*Do we need to define these\*\*\*

* Back end
* Client end
* Front end
* Server end

## References and Acknowledgments

<List any other documents or Web addresses to which this document refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.

TO DO: Use the standard IEEE citation guide for this section.>

[How To Create Meteor App with dhtmlxScheduler](https://dhtmlx.com/blog/create-meteor-app-dhtmlxscheduler/) [1]

Setting up accounts [2]

Creating task list [3]

Routing and setting up Iron [4]

Importing Collections2 and using “Schema” [5]

Setting up Meteor and tutorials on using Meteor and JavaScript frameworks [6]

# 

# Design

## System Modeling

< Update your UML diagrams in milestone 2, to reflect the real implementation of this software.

TO DO: Provide an updated version of the UML diagrams, including use case diagrams, sequence (or state) diagrams, activities diagrams, and class diagrams. If you don’t have an updated version, just mention: “our implementation strictly follows the design document (milestone 2)”. >

## Interface Design

<Provide several screenshots to illustrate your interface design.

TO DO:

For each subsystem, pick one or two representative screenshots and paste here.>

# Implementation

## Development Environment

<Describe the devleopment environment you were using for the project.

TO DO: List the programming lanagues, IDEs, tools, etc.>

## Task Distribution

*<Describ how the implementation tasks are distributed among team members.*

*TO DO: For each team member, describe his/her main implementation tasks in this project.*

*If this is a one-person project, mention: “all the work presented here is done by \*\*\* (your name).” >*

## Challenges

*<This section is optional. Describ the challenges in the implementation, if there are any, and how you dealt with them.*

*TO DO: If you don’t have anything to fill in, just leave this section blank.>*

# Testing

## Testing Plan

Had we gotten to the testing phase of this project, we probably would have used mocha and chai to run tests in the web browser outside of an IDE like IntelliJ. That way we could stick to editing the Meteor code in any text editor and see our tests running in the same browser that the application itself is running in.

## Tests for Functional Requirements

<Describe your test results for the functional requirements.

TODO: Provide a list of use cases or functions you have tested, as well as the testing results (whether or not the system passed the tests).>

## Tests for Non-functional Requirements

<Similar to the Section 4.2, but this section is for the non-functional requirements. >

## Hardware and Software Requirements

<Describe the hardware and software requirements for performing the tests. >

Required software:

* IntelliJ/Sublime Text
* Chrome
* Meteor and npm installed

We didn’t do any formal testing never got to this stage

# Analysis

<In this Section you need to analyze the effort that has been put on this project.

TODO: Describe how many hours (approximately) each team member spent on the project, for each milestone, which milestone takes the most effort and why. >

Sarah did:

* the majority of getting the project set up
* Front end: HTML, CSS, Mockup, Templates
* Leadership, Organizing the group
* UML, Structural diagram

Kyle did:

* Behavioral diagram
* Assignment list app
* MongoDB database
* Blaze server

Spencer did:

* Set up Git repository for project
* Activity diagram
* Scheduler app
* Account package for website
* Fused separate meteor projects into a unified project/app

We spent about 10 hrs/week/person working on this in between our regular assignments and studying for other classes. Easily most of our time was spent on research and learning from tutorials how to get started and how any of the concepts used for this project work.

We think that if we had a longer period of time and no other assignments and major duties outside of this project, we could have finished the whole application with a fairly robust feature set. Our effort toward this project was mainly held back by other responsibilties we had to homework, other programs, studying and our jobs.

# Conclusion

<Conclude the document with what you have learned through working on the project.>

We learned that teams can make a large task much simpler by breaking the workload into smaller, more manageable tasks. One thing we struggled with was judging how to properly divide the project. Because we all have never worked on a web application before, we got stuck doing a lot of research and learning how to get started.

One of the most challenging things we learned was how various pieces all connect together to make a unified application. Some things we now know how to do because of this project are:

1. How to create a basic Meteor app
2. Find and install packages to implement functionality
3. Route a webpage to multiple web apps for a dynamic website
4. How servers and clients are used and which parts of a web app they handle
5. How to set up a server to save data and changes made in an app
6. How apply an account system to give each user an individual experience

Appendix A - Group Log

< Describe how frequently the group meembers meet during the semester, and how effective the communication is. This is optional for one-person projects.>

Our team used a Slack workspace to communicate regularly. We used Slack to plan how to work on this project, delegate the workload, and schedule meet-ups for in person collaboration. We met as a group at least once a week in the computer labs and discussed our progress or struggles with the tasks we were working on. Toward the end of this development phase, we would sit in the same room and work together to get the multiple web apps into a single cohesive website.

Sources:

[1] <https://dhtmlx.com/blog/create-meteor-app-dhtmlxscheduler/>

[2] http://meteortips.com/first-meteor-tutorial/accounts/

[3] <http://meteortips.com/first-meteor-tutorial/databases-part-1/>

[4] <https://iron-meteor.github.io/iron-router/#using-javascript>

[5] <https://www.sitepoint.com/6-must-use-meteor-packages-almost-every-project/>

[6] https://www.meteor.com