

CS 320 Course Project Final Report

for

Expense Tracker

Prepared by

Group Name: KVC Inc.

|  |  |  |
| --- | --- | --- |
| Chayse Jorgenson | 11678623 | Chayse.jorgenson@wsu.edu |
| Kevin Tong | 11686700 | Kevin.tong@wsu.edu |
| Vitaliy Kudrik | 11603449 | Vitaliy.kudrik@wsu.edu |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
|  |  |
| Date: | 12/13/2019 |
|  |  |
|  |  |
|  |  |

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 Expense tracker is a useful tool to offer users insight into managing their finances more efficiently by displaying user input financial information in an easy to read format. The information displayed contains a graph of different expenses,

Once the expenses have been totaled and displayed to the user, Expense Tracker also displays tips or recommendations on where the user may need to adjust their finances or show the user where they have been doing well with their finances.

## Definitions, Acronyms and Abbreviations

There are no acronyms or abbreviations in this document.

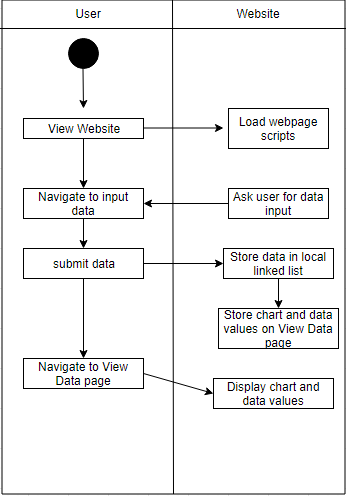
## References and Acknowledgments

NA

# Design

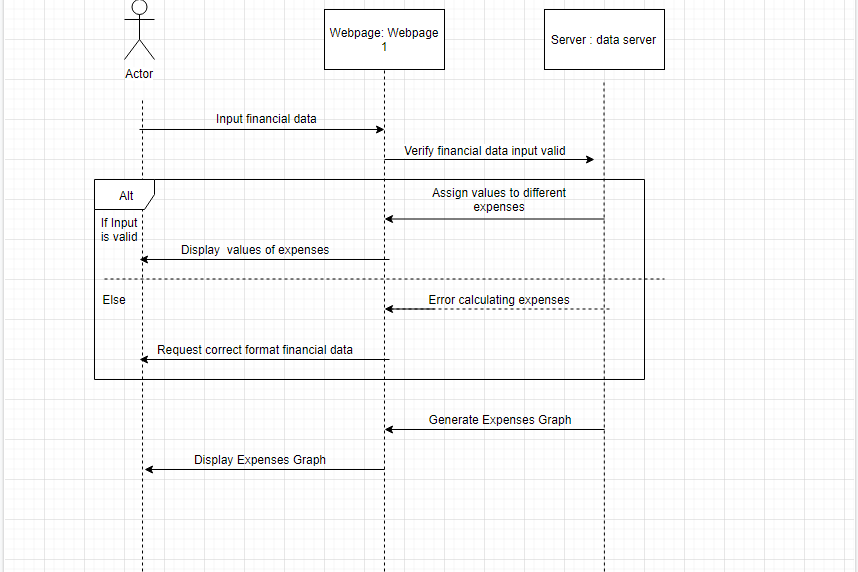
## System Modeling

Use Case Diagram:



Class Diagram implementation strictly follows the design document (milestone 2).

Sequence Diagram:



## Interface Design

# Implementation

## Development Environment

IntelliJ IDE, Notepad ++, Javascript, HTML, CSS, Semantic UI

## Task Distribution

Chayse: JS backend and integration of backend and front end. Implemented a Test Suite. Also contributed to all reports.  
  
Kevin: HTML and CSS, Contribution to SRS and final report.

Vitaliy: Html and CSS, also writing on the SRS and some of the final report.

## Challenges

Vitaliy: Just trying to get everything lined up and working properly was a challenge in the beginning but it got better later on. The biggest challenges was writing the SRS and the final report, because I just don’t know what to expect and if what we actually wrote is up to standard or not.

Chayse: I had a few challenges incorporating a save method that could be used to transport data between html pages. I attempted to use a txt file that the program would store data in and then read it back upon opening the second html. I was able to ge the program to read and write from the file great but but could not combine it with the html since it was using Node functionality. The second method I tried was doing all data entry and calculations on one html page and then storing the numbers we wished to display in local variables within the html. This worked great and allowed us to accomplish our goal. In the future I would consider a more client server approach which would allow for more flexibility.

Kevin: Our first and main challenge was figuring out how we were going to implement our design when first writing out our SRS document. Luckily, we had an assignment that introduced us to HTML coding and we were able to use that knowledge to aid us in constructing our design. The other minor challenges we had were how to connect the backend data collected to our HTML design to present the data collected.

# Testing

## Testing Plan>

Testing will occur on 12/11/2019

Functions to Test:

Linked List Cosntructor()

AddToList()

TotalExpense()

TotalByCateg()

UI Testing:

Bounds tests for each field. Constraints are numbers only

Functional tests for each button and graph

## Tests for Functional Requirements

Automated:

Linked List Cosntructor() Passed

AddToList() Passed

TotalExpense() Passed

TotalByCateg() Passed

UI:

Bounds Passed

Functional Passed

## Tests for Non-functional Requirements

Program should be able to handle large amounts of expenses

Test creating 200 expenses and running TotalExpense() max computation time should be 10 ms. Passed (2ms)

## Hardware and Software Requirements

Test suite is implemented using Mocha and Chai.

Must be ran on a windows 10 machine. Connected to the internet.

# Analysis

Vitaliy: The SRS took a few hours to write up(Just my parts alone), because it was the initial document without any real tangible thing to work with/base on. The html and css also took a few days, mostly because there were lots of iterations of the final look. There were also lots of changes in how we presented the data which required a few extra hours of work. Also getting all the html elements everything to actually go where you want it to go was kind of a challenge. I believe the SRS and the Final report were the hardest because they’re the only things, I personally can’t verify if we are doing correctly or not. Mostly because we can see the website functioning properly and the testing showed that there were no errors, but we don’t know if the actual technical documents are up to par, because we haven’t had them graded yet.

Chayse: The SRS took around 3 hours between writing my section and reviewing others. The backend took about 4 hours to code. Integrating the JS with the html took around 9hrs total. With a lot of that time being related to road blocks faced along the way. The testing suite took around 45mins since I was able to structure it similar to our previous assigment. The final report took around 1.5 hrs. The part that took the most effort was finding a good way to allow data to pass between html pages. Aside from this class I had never written html or JS code before which made finding solutions to problems slightly harder. After doing lots of research I learned multiple ways of solving this problem and was able to find one that suited our project.

Kevin: Although the SRS didn’t take up the most time, I believe it was the one that required the most effort due to the fact that we would use most of this time to brainstorm ideas and decide how we were going to implement our design. This seemed to be the case as we had to decide what our program was to provide and display along with its functions. Once we figured that out, it everything else went smoothly as we were able to support each others work. Updating the HTML webpages was efficient as we were able to each contribute our ideas to the design and display.

# Conclusion

Vitaliy: Through this project, I learned the intricities of teamwork, and I also got a glimse of the actual work a computer scientist has to do, outside of just normal coding. I learned that writing SRS documents was not the most fun thing to do, and also learned that html does not like beginners. I tried out many different frameworks and styling packages that all were very interesting and showed me the possibilites of JS and frontend coding. Frameworks like Meteor JS, and frontend tools like Semantic UI. Actually writing a website and having a use for it, has opened up my eyes to the world of html and javascript. I’ve gotten a lot of ideas that would work extremely easily with a webpage and some javascript, that I can know implement, due to the knowledge gained from this project.

Chayse: Through working on this Project I have learned a a lot about JS, Node, and Html, all of which I had never used prior to this class. I learned a lot about the advantages and challenges that come with working in a team. I also learned a lot about the requirements side of Software Engineering and how important strict, explcit requirements are.

Kevin: From working on this project, I learned more about javascript and HTML coding. I learned how much configuration there is that goes into HTML from different sizing complications to different ways to implement graphics and how they are displayed correctly on wepbages. I learned how frustrating and how useful a tool Github is. I believe the most important takeaway from this project will have to be the SRS document as it shows the more professional/business side of programming.

Appendix A - Group Log

Kevin: Our group would usually meet up after class to discuss where we were at on the project and how we would proceed. Most of our work was done at home, while our communication was daily through social platforms.