Budget Watch

Tyler Phenix

UMKC – School of Computing and Engineering

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Last User to make Changes** | **Date** | **Summary of Changes** |
| T. Phenix | /8/11/16 | Vision statement, product backlog, sprint plan |
| T. Phenix | 8/25/16 | Review, Retrospective, Velocity |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Vision Statement 4](#_Toc459581887)

[Requirements 5](#_Toc459581888)

[Step #1: Identify Categories of Users 5](#_Toc459581889)

[Step #2: Create Actor-Goal List 5](#_Toc459581890)

[Step #3: Identify User Stories 6](#_Toc459581891)

[Product Backlog 7](#_Toc459581892)

[[Step #4: Write Use Cases] 8](#_Toc459581893)

[Sprint #1 12](#_Toc459581894)

[Review 12](#_Toc459581895)

[Retrospective 13](#_Toc459581896)

[Design 14](#_Toc459581897)

[Coding Standards 15](#_Toc459581898)

# **Vision Statement**

I will build an app that allows the user to enter a budget for a shopping trip and allow the user to input the items as they go to make sure they stay on budget called “Budget Watch”. This will help with people who overspend during a shopping trip or anyone who wants to be stay organized while shopping. This is most important for parents and people who don’t have much money to spend (college students, low income families, etc.). Unlike other apps I have seen, Budget Watch gives helpful feedback to user to help stay ion budget.

My project will include a calculation feature, for matching the target budget amount against what the user has already spent. My project will also include a pop-up to let the user know if they are getting close to the budget or over budget. Also once you are done shopping, it will give you a rundown of how far over/under budget you were. My project will include a category to put each item in, either “need” or “want”. My project will not include a barcode scanner because prices vary based on different stores, store location, sales, etc. Instead the user will manually put in each item’s name and price (possibly a picture of the item also).

Essential features of my project include calculating the total price and matching it against the target budget and feedback to the user (whether they were on budget or not).

Desirable features include allowing the user to take a picture of the item and saving the shopping trip list for future reference.

# **Requirements**

## Step #1: Identify Categories of Users

**User**

**Parents** – Adults with a child or children who can use smart phone apps well.

**College students/recent graduates** – People who are older teens up to about age 30 who are saving money to pay for/back school expenses.

## Step #2: Create Actor-Goal List

The actor-goal list shows the mid-level goals for each system actor. The categories of users above define human actors. In addition to human actors, there may be other non-human system actors such as external computer systems.

Probably the hardest part of creating an actor-goal list is deciding at what level of abstraction to express goals. For example, “make money” might be a goal but is too abstract to be useful as the only goal expressed in a requirements document. You can’t plan, estimate or design a software system based solely on the goal “make money”.

At the other extreme, “close dialog box” is too specific. No one is going to read a requirements document that lists goals at this level of specificity. To avoid having overly constraining goals, consider developing the goals in collaboration with the programmers that will be implementing the system.

Goals should be expressed at a level of abstraction that allows:

* Users and customer to see what they will be getting (users and the customer need to be able to review and validate requirements)
* The project manager to estimate the cost and schedule for the project
* The project manager to plan the project (that is, create a course-grain release plan and fine-grain iteration plans. Release plans express a tentative schedule for delivering features, iteration plans express a detailed schedule of tasks for implementing the features of an iteration.)

The detail included here will also depend on what other sections are included in the requirements document. Some of the details of the requirements can be expressed as system shalls. If there is a section listing “system shalls”, goals here can be more abstract.

Example actor-goal list:

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| User | Input budget |
|  | Input items as shopping trip continues |
|  | Input name of item |
|  | Input price of item |
|  | Take picture of item if needed |
|  | Decide whether item is a need or a want |
|  | See results at the end of shopping trip |
|  | See total of entire shopping trip |
|  | See how much over/under budget shopping trip was |
|  | See total of wants and total of needs |
|  | See how much shopping trip would have been without wants |

## Step #3: Identify User Stories

As a parent, I want to spend as little as possible at the store so that I can pay other bills.

As a college student/recent graduate, I want to save money on groceries so that I can afford housing.

Product Backlog

The collection of stories makes up your project’s product backlog:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Story** | **Story Points** | **Priority** | **Status** |
| S1 | Allow user to enter budget | 2 | 4 |  |
| S2 | Allow user to enter name of item and its price | 5 | 4 |  |
| S3 | Allow user to take picture of item | 7 | 2 |  |
| S4 | Allow user to select if item is need or want | 6 | 1 |  |
| S5 | Allow user to get feedback during shopping trip based on total relative to budget | 3 | 4 |  |
| S6 | Allow user to see totals at the end of shopping trip | 4 | 4 |  |
| S7 | Allow user to see how much they could have saved if they had no “wants” | 3 | 3 |  |

# **Sprint #1**

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Design UI | 3 | 3 |
|  | Add code for storing input | 3 | 4 |
|  | Add about menu | 2 | 3 |
| S2 | Add features to layout | 3 | 3 |
|  | Add code for running total | 4 | 4 |
| S5 | Design pop-up menus | 5 |  |
| S6 | Add button for checkout | 3 |  |
|  | Add code for calculating totals | 4 |  |
| S7 | Add code for calculated new totals | 4 |  |
| S3 | Import picture library | 3 |  |
|  | Link picture to item | 3 |  |
| S4 | Add selection menu for each item | 3 |  |
|  | Add code for storing selection of each item | 4 |  |
|  |  |  |  |

## Review

S1: Has actually taken about 10 hours and it is still not finished.

S2: Has taken about 7 hours as predicted but it is still not finished.

## Retrospective

**Iteration 1:**

For this iteration I planned on finishing two stories. I did not completely finish neither. The main reasons for this is I misinterpreted how long each task would take. My about menu was supposed to be fairly quick but I could not get it to work. Also for Sprint 2 I could not get the numbers to be parsed in correctly. The amount of time each task takes will have to be recalculated in the future to better manage my time on each sprint.

For Iteration 2 I plan on finishing Sprint 1 and 2 and also 5.

**Project velocity: 0**

# **Coding Standards**

Coding standards improve readability. They make it easier to understand code written by others. Good coding standards also improve reliability.