Financial Planner

Cody Gonsowski and Trever Hall

Abstract

This project is a financial planner to help users keep track of their spending habits. It can display expense and income information over the course of years, and the user can navigate the user interface to see the information for each day. The target audience is for anyone seeking to have more financial stability. We have begun documentation of what our goals are and what we believe is the stopping point of the program.

1. Introduction

This is a financial planner for users to track expenses over time and create a budget. Most college students have a hard time with money, either from a lack of funds or spending irresponsibly. If financial habits go unchecked then a person can get trapped in a vicious cycle of debt and high interest rates, just to afford to live. The goal is to provide users with an outlet to track and plan a budget that allows them to live within the income that they have.

1.1. Background

The hope of this project is for the user to be able to feel more secure with financial decisions by being able to track expenses and income over time. Expenses include any money spent on bills or even person items. Income is all money accrued wether it is interest from investments or an earned wage. Being as specific and inclusive as possible will allow the user to know where every dime is made and spent.

We decided to do this not only because it would be challenging but also have a practical application in our day to day lives. I, Trever Hall, have a paper financial planner that I write and track funds in. I would like to have an online planner, so personally I plan on using this in my day to day life to manage my own finances.

1.2. Challenges

At face value the program seems pretty straight forward. We are not sure of all the challenges we will face, but challenges we do know of include:

The creation of new expenses and income on demand

A graphical interface that can support scaling from the span of years to financial events of a specific day.

Gathering the API's we want for example one that will allow us to know which day of the week each day fall on.

2. Scope

In terms of the scope of the project what we would define as done would be the user having the ability to locate a specific day within a year. Once there, the user would be able to view all information for that day and also be able to add expenses and income information. For each expense the user should be able to put it into a preset category, and the money spent on each category should be displayed on every view implemented (yearly, monthly, etc.). Income should be able to be put under the categories of Salary or Other, and displayed with the expense information. Along with that, a difference of the money earned (net gain or loss) for that time period and the money spent should also be in view.

2.1. Stretch Goals

Stretch goals for our project are things that we think would give the user a better experience, but are not necessary for the over all function of the program. Those would include:

Graphical representation of data for the current view (perhaps a pie chart).

A fullscreen clickable calendar that allows the user to navigate through years all the way down to a specific day. It will be more of a ”pretty” way to navigate the information, but not edit.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.2. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.2.1. Functional.

- User needs to have a private shopping cart – this cannot be shared between users, and needs to maintain state across subsequent visits to the site
- Users need to have website accounts – this will help track recent purchases, keep shopping cart records, etc.
- You'll need more than 2 of these...

2.2.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

2.3. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table **??**.

Use Case Number:	1
Use Case Name:	Add item to cart
Description:	A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure **??**). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number:	2
Use Case Name:	Checkout
Description:	A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

Figure 1. First picture, this is a kitten, not a use case diagram

2.4. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure **??**.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.