# Financial Planner

Cody Gonsowski
Trever Hall

**Abstract**

**This financial planner is meant to help users keep track of their spending habits. It can display expense and income information over an extended timeframe and allows the user to navigate to a specific day's information through a user interface. The target audience is anyone seeking to have more financial stability, but people can easily use it for casual financial tracking. We have made corrections as recommended by our instructor and added use cases, functional and non-functional requirements, and interface mockups.**

## 1. Introduction

This is a financial planner that allows users to track expenses over time and create a budget. Most young adults, especially college students, have a hard time with money, either from a lack of funds or spending irresponsibly. If financial habits go unchecked, a person can get trapped in a vicious cycle of debt and high interest rates, just to afford to live. Even if people are not having a hard time with money, it still helps to develop good habits for the future as soon as possible.

The goal is to provide users with an outlet to track their expenses and plan a budget that allows them to live more comfortably within the income they have. By tracking and categorizing expenses, even all the small amounts, people will be able to see where their money goes and how much it adds up. Once people recognize how much they are actually spending, they are then able to tailor a budget to their needs.

### 1.1. Background

The primary goal of this project is to make users feel more secure with their financial decisions by allowing them to track expenses and income over time. Expenses include any money spent on bills, groceries, personal items, etc. Income is all money accrued, regardless of its source (i.e., interest from investments or an earned wage). Users will be able to know where their funds are at any given time by being as specific and inclusive as possible.

We believe that not only will this project will be challenging but also it will provide something practical. Trever Hall, a developer on the team, uses a paperback financial planner to keep track of funds. He would like to have an online planner to use in his day-to-day life to manage his own finances.

### 1.2. Challenges

Because the program seems relatively straightforward, we are unsure of all the challenges to come. However, challenges we expect to come across include as follows: on-demand creation of custom expense and income categories, graphical interface that scales from years to a specific day, and gathering APIs we would like to use to help the user experience.

## 2. Scope

The user will be able to locate a specific day within a year. Once there, the user can view all information for that day and add expense and income information. The user will be able to add expenses and income to a selection of preset categories, as well as custom user-created categories. Expense and income information will be displayed alongside each other for easy comparison. Total money spent or earned in each category, as well as net income, will be displayed on every view implemented (yearly, monthly, etc.).

Some stretch goals we would like to implement to provide a better, more convenient user experience include as follows: graphical representation of data for the current view (i.e., categorical pie chart) and a fullscreen calendar that allows the user to navigate to a specific day (more aesthetic).

### 2.1. Requirements

Each user should have their own planner independent from other users; by using an account-based system, this becomes very simple. When using accounts, however, security is something that cannot be overlooked. Ensuring that the planner scales well with large amounts of information allows for users to input data freely.

| Use Case ID | Use Case Name | Primary Actor | Complexity | Priority |
|---|---|---|---|---|
| 1 | Search date | User | Med | 1 |
| 2 | Add budget category | User | Med | 1 |
| 3 | Sign-in | User | Med | 1 |

TABLE 1. USE CASE TABLE

#### 2.1.1. Functional.

- User needs to have a personal financial planner – this will be on a per-user basis
- Users will have accounts that allows them to access their own planner from anywhere with an internet connection and the appropriate software – this allows users to remotely change their planner
- Calendar date selection should be locked to only valid, calendar-standard, integer inputs

#### 2.1.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- Scalability – planner should be able to handle many years' worth of information

### 2.2. Use Cases

Use Case Number: 1
Use Case Name: Search date
Description: A user enters the calendar date where they want to go. They will click on the "Search" button. This will adjust the calendar and info view accordingly.

1) User enters the month, day, and/or year they wish to jump to
2) User left-clicks on the "Search" button
3) Calendar and Info displays now display the chosen date and its information

Termination Outcome: The user's planner is updated to display information of the chosen date.
Alternative: Input is invalid

1) User enters the month, day, and/or year they wish to jump to
2) User left-clicks on the "Search" button
3) Calendar and Info displays are reset to the previous selected date

Termination Outcome: The user's planner display is left unchanged.
Use Case Number: 2
Use Case Name: Add budget category
Description: A user would like to add a custom budget category. They will enter a custom category in the "Add a category" text box. They will click "Confirm." This will add a custom category for the budget section.

1) User enters the name of the category they would like to add into the "Add a category" text box
2) User left-clicks on the "Confirm" button
3) Budget should now have a new available category

Termination Outcome: The budget section should now have an additional category to add budget info under.
Use Case Number: 3
Use Case Name: Sign-in
Precondition: User already has a pre-existing account.
Description: A user wants to sign-in to access their financial planner. The user enters their username in the "Username" text box and their password in the "Password" text box. They will click "Sign-in." This will allow them to access their personal financial planner.

1) User enters their username into the "Username" text box
2) User enters their password into the "Password" text box
3) User left-clicks the "Sign-in" button
4) User can access their financial planner

Termination Outcome: The user should be signed in and able to access and edit their own unique financial planner.
Alternative: User enters incorrect username or password

1) User enters their username into the "Username" text box
2) User enters their password into the "Password" text box
3) User left-clicks the "Sign-in" button
4) The username and password boxes reset to empty

Termination Outcome: The user is unable to sign in and needs to try again in order to access their unique financial planner.

### 2.3. 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 **??**.

Figure 1. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

### 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!

## 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.