Financial Database Manager

Cody Gonsowski

Abstract

This financial database manager is meant to help users keep track of their finances. It can display expense and income information over an extended timeframe and allows the user to navigate to a range of days' 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. Timeline and the baseline project were most recently completed.

1. Introduction

This is a financial database manager that allows users to track expenses and incomes over time. Most young adults, especially college students, have a hard time with money, either from a lack of funds or from spending irresponsibly. If bad financial habits go unchecked, a person can potentially get trapped in a vicious cycle of day-to-day living and high outgoing payments. 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 a database manager that allows for them to easily view how much they are making or spending over a given period of time. By allowing users to enter (any number of) financial data entries, they will be able to easily see their spending habits over a range of time. The program also provides a total for the range.

1.1. Background

The primary goal of this project is to keep a localized database of incomes and expenses that are entered by the user. This allows for the user to easily view any number of entries in a given range. Users will be able to see detailed descriptions of financial data they provide, given they input a sufficient description.

I believe that not only will this project will be challenging but also it will provide something practical. Suppose a person uses a paperback financial planner to keep track of funds. S/he would like to have an online planner to use in their day-to-day life to manage their own finances, but that seems a bit out of reach at this point.

1.2. Challenges

Because the program seemed relatively straightforward, I was unsure of all the challenges to come. However, challenges I expected to come across included as follows: on-demand creation of database entries and financial display that shows all entries in a given range.

The on-demand creation of database entries was relatively straightforward. Once I understood how to use OleDB stuff better, it became very easy. Financial display was handled via a FlowDocument + a Table. This was to show an organized representation of the data that is uniformly formatted.

2. Scope

The user will be able to add an entry to the database through simple form controls. By providing this information, they will be able to add an entry to the database through a simple click. The user will also be able to select a range of dates and display it as a formatted table on a large portion of the application window. The range of dates can vary greatly, and the table should be able to support it through a scroll bar. A total row will be displayed at the bottom of the table that shows the total of the given range.

Some stretch goals we would like to implement to provide a fuller user experience include as follows: clickable calendar interface and a graphical representation of the data.

2.1. Requirements

Users can keep the database stored locally so their own form(s) of security will help cover it. Ensuring that the database 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 local financial database
- User can edit the database directly, and the database should be changed when running the manager
- Calendar date selection should be locked to only valid, calendar-standard inputs

2.1.2. Non-Functional.

- Operability – interface should be easy to understand and use
- Scalability – database should be able to handle many years' worth of information

2.2. Use Cases

Use Case Number: 1
Use Case Name: Date range
Description: User selects range of dates they wish to view. They will click on the "Display" button. This will update the table accordingly.

1) User enters the initial date of the range
2) User enters the final date of the range
3) User clicks "Display"
4) Financials table will display information of dates in the range

Termination Outcome: The Financials table is updated to display the information in the given range.
Alternative: Input is null

1) User clicks "Display" without anything in start or end date
2) Error will appear indicating no start date
3) If there was a start date, then an error will appear indicating no end date

Termination Outcome: The user's planner display is left unchanged.
Use Case Number: 2
Use Case Name: Add entry
Description: A user would like to add an entry to the database. They will fill out the three fields (amount, date, and description) and click the "Add" button. This will add the entry to the database.

1) User selects an entry date using the date picker
2) User enters an amount
3) User enters a description (or doesn't)
4) User clicks on the "Add" button
5) Entry will be stored in the database

Termination Outcome: The database will now contain another entry made by the user. (NOTE: does not dynamically update Financials table if entry is in range)

2.3. Interface Mockups

Three objects that allow the user to input, DecimalUpDown (restricts to numerical values), DatePicker (calendar selection of date), and Description (textbox entry) "Add" Button allows user to add this in one click. Date range selection utilizes two DatePickers for easy selection. "Display" Button allows user to update Financials table in one click.

3. Project Timeline

Timeline vaguely looks like... See Figure 1.

4. Project Structure

MainWindow utilizes DisplayDataRange and AddEntry to provide easy-to-use interfaces for the user.

ENTER START DATE: 9/5/2019

ACTIVITY	START	END	NOTES
Partnering & Brainstorming	9/1/2019	9/8/2019	financial planner idea
LaTeX Documentation 1	9/22/2019	9/29/2019	
Project Scope Established	9/29/2019	10/8/2019	
LaTeX Documentation 2	10/8/2019	10/15/2019	corrections made
Interface Mock-ups	10/15/2019	10/22/2019	
Presentation 1	10/15/2019	10/22/2019	presenting proposal to class
UML Layouts	10/29/2019	11/5/2019	
Basic UI Design	11/12/2019	11/19/2019	basic calendar layout and widget things
LaTeX Documentation 3	11/19/2019	11/26/2019	including timeline
Presentation 2	11/19/2019	11/26/2019	show that the project works vaguely
LaTeX Documentation 4	11/19/2019	12/3/2019	penultimate writeup
Abandonment	12/2/2019	12/4/2019	Ericson's approval to basically do the whole project differently and alone
Final Presentation	12/5/2019	12/5/2019	
Project End	12/7/2019		

Figure 1. This is a timeline!

4.1. UML Outline

UML vaguely looks like... See Figure 2. MainWindow utilizes two main functions: DisplayDataRange and AddEntry. DisplayDataRange takes the StartDatePicker's and EndDatePicker's entered dates to determine what should be displayed in the table. AddEntry takes the Amount, Date, and Description entered into the form to create a new entry in the database that can be loaded up later.

4.2. Design Patterns Used

AddEntry provides an easy to use interface (Facade) for the user to add new entries to the database, without the user having to go enter everything directly into the database: The program takes care of this. DisplayDateRange also allows the user to select two dates and display that range of dates, without the user having to sift through all of the database entries manually.

5. Results

Unfortunately, many of the project's original goals were not accomplished. The original interface needed a rework to account for change of plans, but hopefully this can be changed in the future to fit a calendar-based interface as originally planned. Some new challenges arose along the way as well, such as working with interworking GUIs, that I was unable to conquer within the timeframe. This should be something to overcome later when implementing the calendar-based interface.

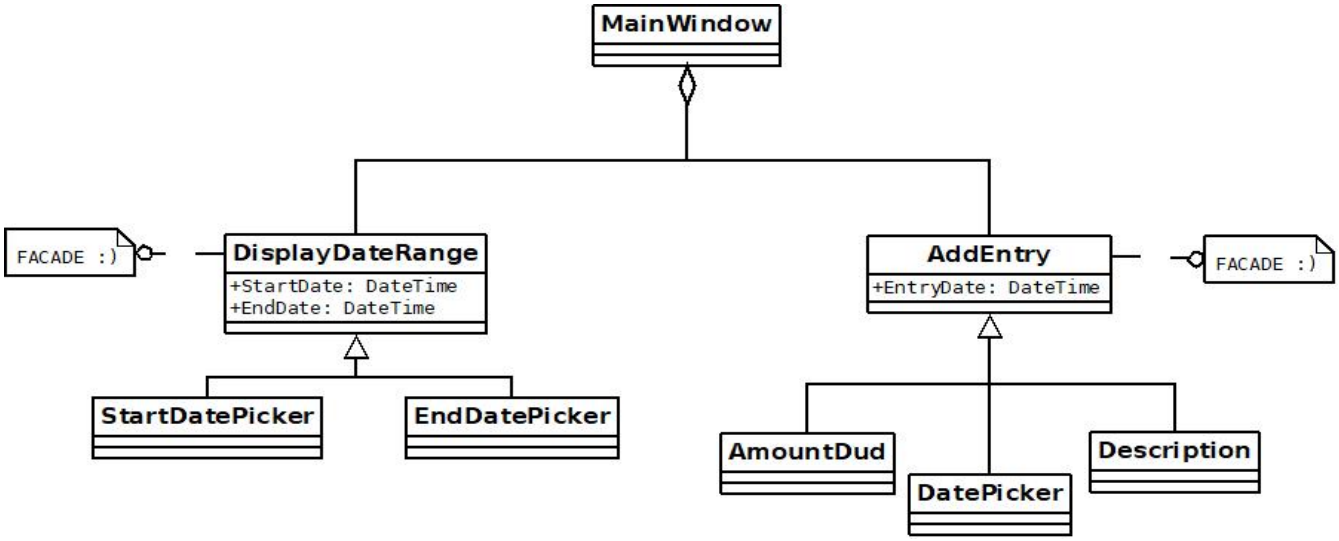

Figure 2. Poorly put together UML layout. Did not utilize classes as well as I should have.

5.1. Future Work

It would be ideal to have the planner further developed with a calendar-based interface. This would be easier to see all finances at a glance. Allowing multiple users to use the same build of the program through user accounts would be beneficial. On that same note, going online with the database storage would allow people with the build to access their personal database from anywhere.