Programming III – Fall 2022
Course Project: Personal Finance Tracker

Team Information

Team name: MJ, Marcus Bloomfield 2264053, Joseph Lehman 2268268

Project Description

Application that manages finances starts with a menu with the ability to add income / expenses and load income / expenses from saved files. Expenses have categories that are also stored. The saved files store the income / expenses for every month. The program can also generate reports for incomes and expenses that contain statistics/information in a month. The reports can be shown in WPF and in file.

Development Approach

Explain how did you prepare for the project. You can use the 5 steps of algorithmic thinking to you help build this section (you will need to elaborate on each step).

1. Understanding the problem.

   List of what we're given and what we expect back:

   Input: For an expense, we are given the amount, the category and the date. For an income, we are given the amount and the date.

   Output: For an expense, the amount, category and date are saved in a file for the corresponding month, and in a file with every expense. For an income, the amount and date are saved in different files for the corresponding month, and in a file with every income. Reports for income files and expense files can be generated into file or interface form with various statistics (average amount, median, range, max, min, etc…)

2. Formulating the problem.

   How we're gonna represent the inputs and outputs in step 1:

   Input: Amounts for expenses and income are represented as a double value. Dates will be represented with date objects. Category will be a string from a predetermined list of categories to choose from. There will be expense and income objects to store the data.

   Output: All data will be saved in comma separated files. The reports saved in file will just be readable text that clearly labels the calculated data. The reports generated on the interface show the calculated statistics on the app interface for the user to see.

3. Developing the application \ algorithm.

   UML Diagrams

Detailed steps for the algorithms (in the repository)

4. Implementing the application \algorithm.
   Translate logic to C#
5. Testing.

OOP Design

Talk about the classes you need to create for the application and what is the purpose of each class. Include the UML class diagram in this section. The UML class diagram should include the relations between the created classes. Do not mention the WPF classes (Window, etc.)

Classes:

FinanceObject (parent of expense and income):
Abstract class with outline for a finance object.
Members:
-Constructor with amount and date
-Amount
-Date
-Getter / Setter for Amount
-Getter / Setter for Date
-ToString method

Expense (Inherits from FinanceObject):
Finance object representing an expense.
Members:
-Constructor with amount, date and category (calls the base constructor)
-Constructor with amount
-Category
-Getter / Setter for Category
-Override for ToString method

Income (Inherits from FinanceObject):
Finance object representing an income.
Members:
-Constructor with amount and date (calls the base constructor)
-Constructor with amount

Report (abstract):
Abstract class with data members and methods that are in a report
Members:
-Total
-Average
-Median
-List of finance objects

-Constructor that takes in a list of finance objects
-Getter / setter for the List of finance objects
-Getter for total
-Getter for average
-Getter for median
-Method to calculate total
-Method to calculate average
-Method to calculate median
-Method to calculate any other statistics that may be added in the future
-Add to list method to add a finance object to the list
-Remove from list method to remove a finance object from the list
-GenerateReportFromFile abstract method
-SaveReport method
-Abstract method to calculate the report
-Abstract ToString method

ExpenseReport (Inherits from IReport):
Report that contains statistics on the list of expenses
Members:
-most used category
-Constructor with no arguments to make empty report
-Constructor with expense list
-Method to calculate most used category
-Getter most used category
-GenerateReportFromFile method
-Method to calculate the report
-ToString

IncomeReport (Inherits from IReport):
Report that contains statistics on the list of incomes
Members:
-Constructor with no arguments to make empty report
-Constructor with income list
-Getter for total
-Getter for average
-Getter for median
-GenerateReportFromFile method
-Method to calculate the report
-ToString

FinanceTracker:
The class that contains all the functionality and needed data of the app
Members:

-Constructor with no arguments that gets the total expenses and income by accessing the files with the stored data
-All expenses file path (The csv file with the data on all expenses)
-All incomes file path (The csv file with the data on all income)
-Total expenses
-Total income
-Getter for total expenses
-Getter for total income
-All expenses list
-All income list
-Getter for all expenses list
-Getter for all income list
-Loaded expenses list
-Loaded income list
-Getter for loaded expense list
-Getter for loaded income list
-Load income file method
-Load expense file method
-Save loaded income file method
-Save loaded expense file method
-Add income method
-Add expense method
-Display expense report method
-Display income report method
-Clear loaded incomes method
-Clear loaded expenses method

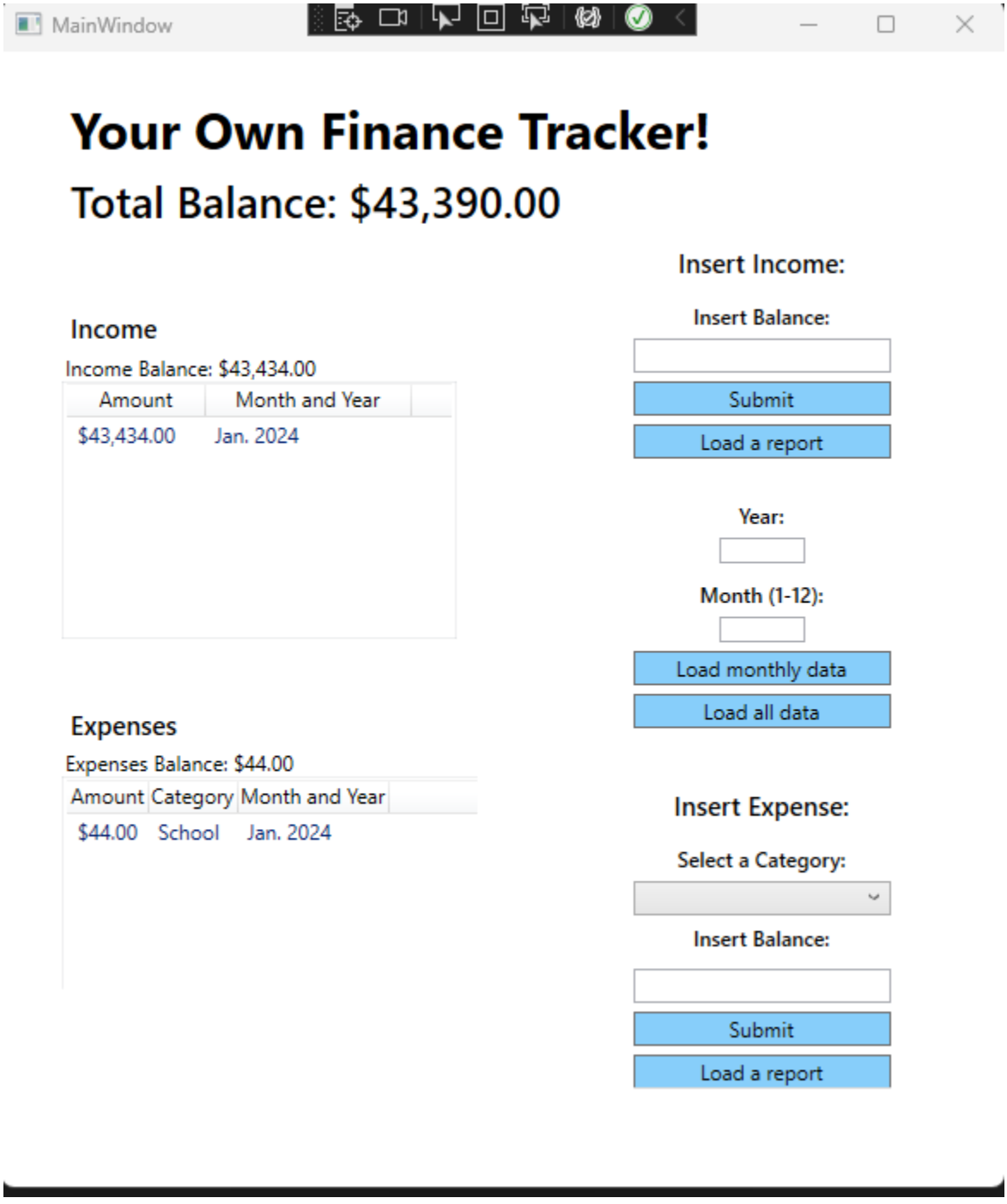
Contributions

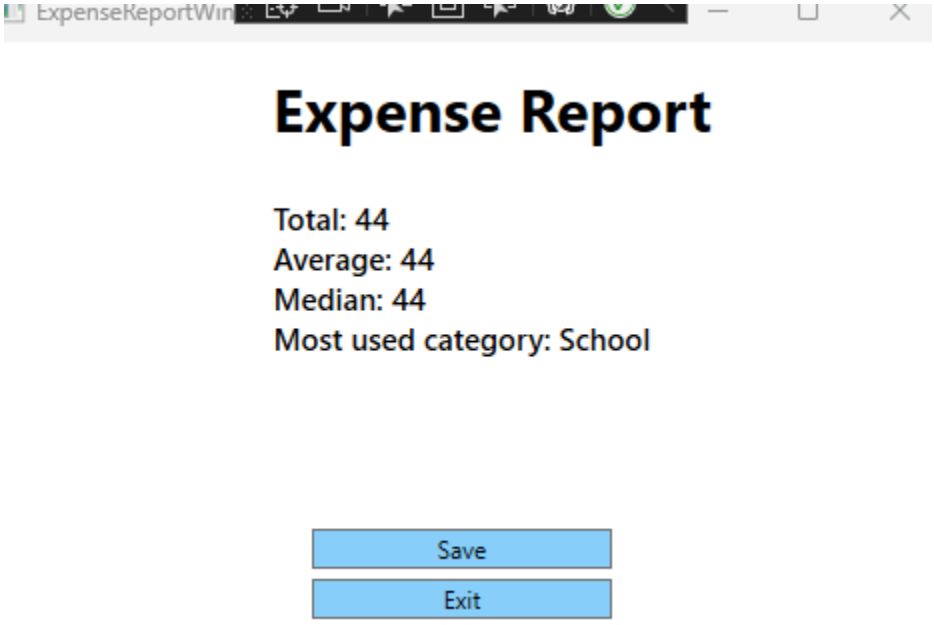
What did each team member do?

Joseph	Marcus
Designed UML Diagram	Design XAML user interface
Wrote pseudocode for the classes	Add functionality to the interface
Code the classes in c# (not the wpf ones)	Create tables to display data from a text file
Made reports display on a new window	Modify data to display specific values
Added feature to load data from a month	Add a couple of properties to classes

App Snapshots

This section includes snapshots of the final application showing different features. It could be a guideline to using the application. You may include snapshots of the app while being developed. Remember to add explanatory captions to the snapshots.





Future Work

Discuss features or improvement that can be added to application.
Allowing the user to remove an income or an expense was something that we didn't consider that would be useful. Being able to sort the tables on the user interface.

Appendix A: Team Contract

Appendix A: Team Contract Template

This is an informal contract to ensure that all team members have a common understanding of what is expected in terms of work standards, communication, division or work, and conflict resolution.

Team Members (Name & ID)

	Name	Student ID
Member A:	Marcus Bloomfield	2264053
Member B:	Joseph Lehman	2268268

Strength & Weaknesses

Within the context of this project, what are the strengths and weaknesses that each member brings to the team?

Member A: Critical thinker, committed to the project, procrastination.
Member B: Good problem-solving skills, hardworking, often over critical on my work,

Definition of "good enough" for this project

What would the team collectively consider "good enough" of an achievement for the project?

(One response for the whole team)

Fully functional, no bugs or crashes, decent looking user interface, good OOP practices, utilization of all OOP pillars.

Picked Topic

Topic 4: Personal Finance Tracker

Division of work

How will each member contribute to the project?
Member A: Most of the WPF interface
Member B: Most of the object oriented programming

Frequency of communication

How often will the team be in touch and what tools will be used to communicate?
Around 3 times a week, on discord.
Response delays
Busy with other projects or personal situations, work.
What is a reasonable delay to reply to messages? Is it the same for weekdays and weekends?
Half a day on weekends (because of work), and within 2 hours on weekdays

Receiving feedback

Each member must provide a sample sentence for how they would like to receive constructive feedback from their peers.
(If unsure, assume a hypothetical situation such as you have not completed your work in time or you have not replied to a message in a timely manner).

Member A: Let me know if something is lacking, and I'll get on it.
Member B: Communicate to me if there is something you notice that I should improve or that I'm doing wrong in a respectful manner.

In case of conflict

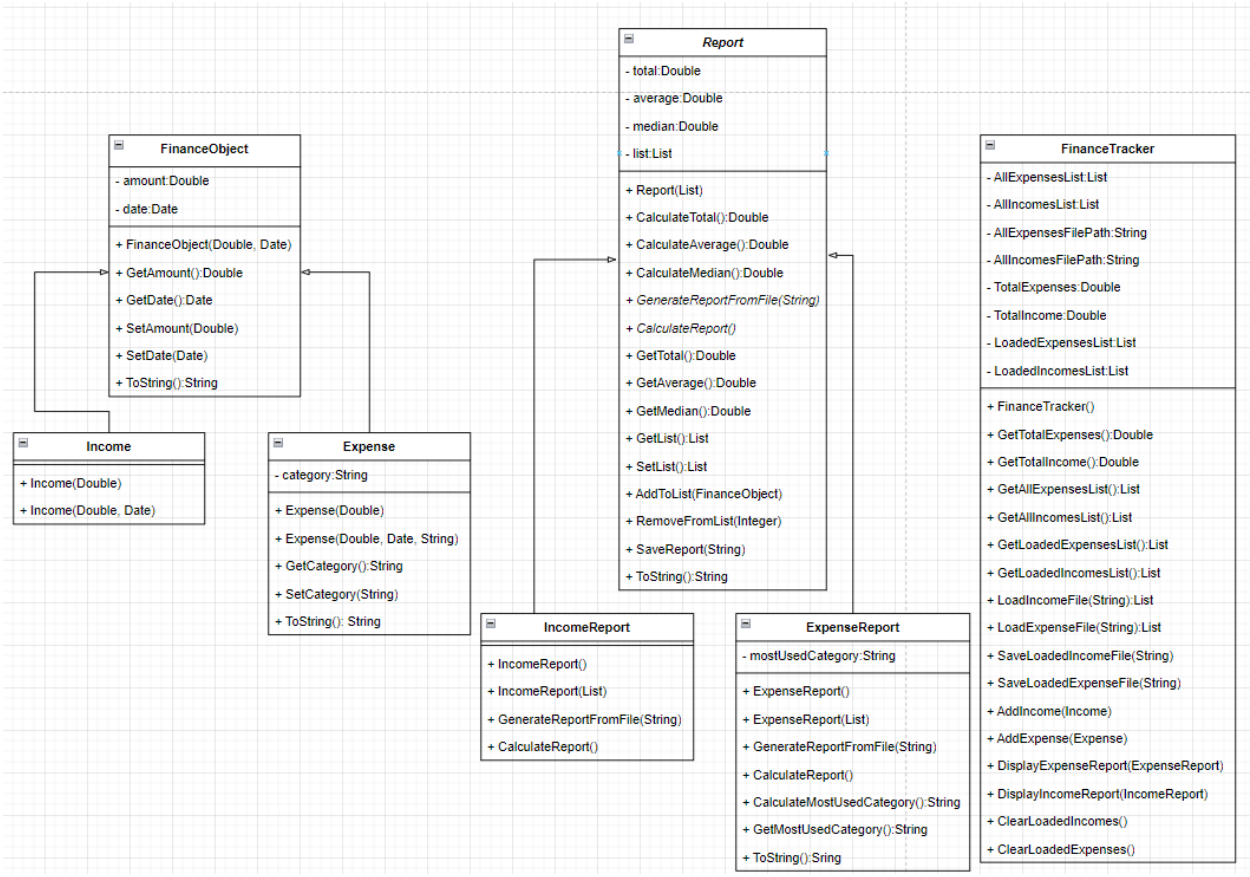
If a team member fails to communicate as described in this contract or does not respond to constructive feedback, what measures should the other teammate take?

Talk to the teammate about it to try and resolve the issue in a peaceful manner. If there is no collaboration, reach out to the teacher to say that the teammate is not cooperating and explain the situation.

(One response for the whole team)

Appendix B: UML Class Diagram

- DO NOT PLACE A LINK TO THE DIAGRAM.
- Do not include WPF created classes in the class diagram.
- The diagram should be placed in the document.


Report
- total:Double
- average:Double
- median:Double
- list:List
+ Report(List)
+ CalculateTotal():Double
+ CalculateAverage():Double
+ CalculateMedian():Double
+ GenerateReportFromFile(String)
+ CalculateReport()
+ GetTotal():Double
+ GetAverage():Double
+ GetMedian():Double
+ GetList():List
+ SetList():List
+ AddToList(FinanceObject)
+ RemoveFromList(Integer)
+ SaveReport(String)
+ ToString():String
FinanceObject
- amount:Double
- date:Date
+ FinanceObject(Double, Date)
+ GetAmount():Double
+ GetDate():Date
+ SetAmount(Double)
+ SetDate(Date)
+ ToString():String
FinanceTracker
- AllExpensesList:List
- AllIncomesList:List
- AllExpensesFilePath:String
- AllIncomesFilePath:String
- TotalExpenses:Double
- TotalIncome:Double
- LoadedExpensesList:List
- LoadedIncomesList:List
+ FinanceTracker()
+ GetTotalExpenses():Double
+ GetTotalIncome():Double
+ GetAllExpensesList():List
+ GetAllIncomesList():List
+ GetLoadedExpensesList():List
+ GetLoadedIncomesList():List
+ LoadIncomeFile(String):List
+ LoadExpenseFile(String):List
+ SaveLoadedIncomeFile(String)
+ SaveLoadedExpenseFile(String)
+ AddIncome(Income)
+ AddExpense(Expense)
+ DisplayExpenseReport(ExpenseReport)
+ DisplayIncomeReport(IncomeReport)
+ ClearLoadedIncomes()
+ ClearLoadedExpenses()
Income
+ Income(Double)
+ Income(Double, Date)
Expense
- category:String
+ Expense(Double)
+ Expense(Double, Date, String)
+ GetCategory():String
+ SetCategory(String)
+ ToString(): String
IncomeReport
+ IncomeReport()
+ IncomeReport(List)
+ GenerateReportFromFile(String)
+ CalculateReport()
ExpenseReport
- mostUsedCategory:String
+ ExpenseReport()
+ ExpenseReport(List)
+ GenerateReportFromFile(String)
+ CalculateReport()
+ CalculateMostUsedCategory():String
+ GetMostUsedCategory():String
+ ToString():Sring