MONEY MANGER

Cody Riniker
Advisor: Nathan Bean

Overview

- Money Manager is an application that helps users track their purchases, payments, and other financial transactions.
- Like a checkbook or journal.
- Allows users to manage transactions, set goals, and schedule transactions for a future date.
- Doesn't communicate with user's bank accounts or any other type of other sensitive account's directly.

Why Did I Make This

- Useful
  - I can use it personally and it can be an alternative to other budgeting apps.
- Reasonable scope
  - Everything planned for this project seemed reasonable with my current knowledge.
- Wanted to make something that could be expanded upon.
  - New features can be added relatively easily.

Orginal vs Final Plan

- **Original**
  - Have both a desktop application and a web application using .NET frameworks.
  - Have a back-end SQL database to store data so that it could be shared.
  - Less features with the limitations of a web application.
- **Final**
  - Just a desktop application was made in order to allow for more features.
  - Uses text file-based storage since all information is stored locally.
  - More features such as calendar and categories.

Specifications

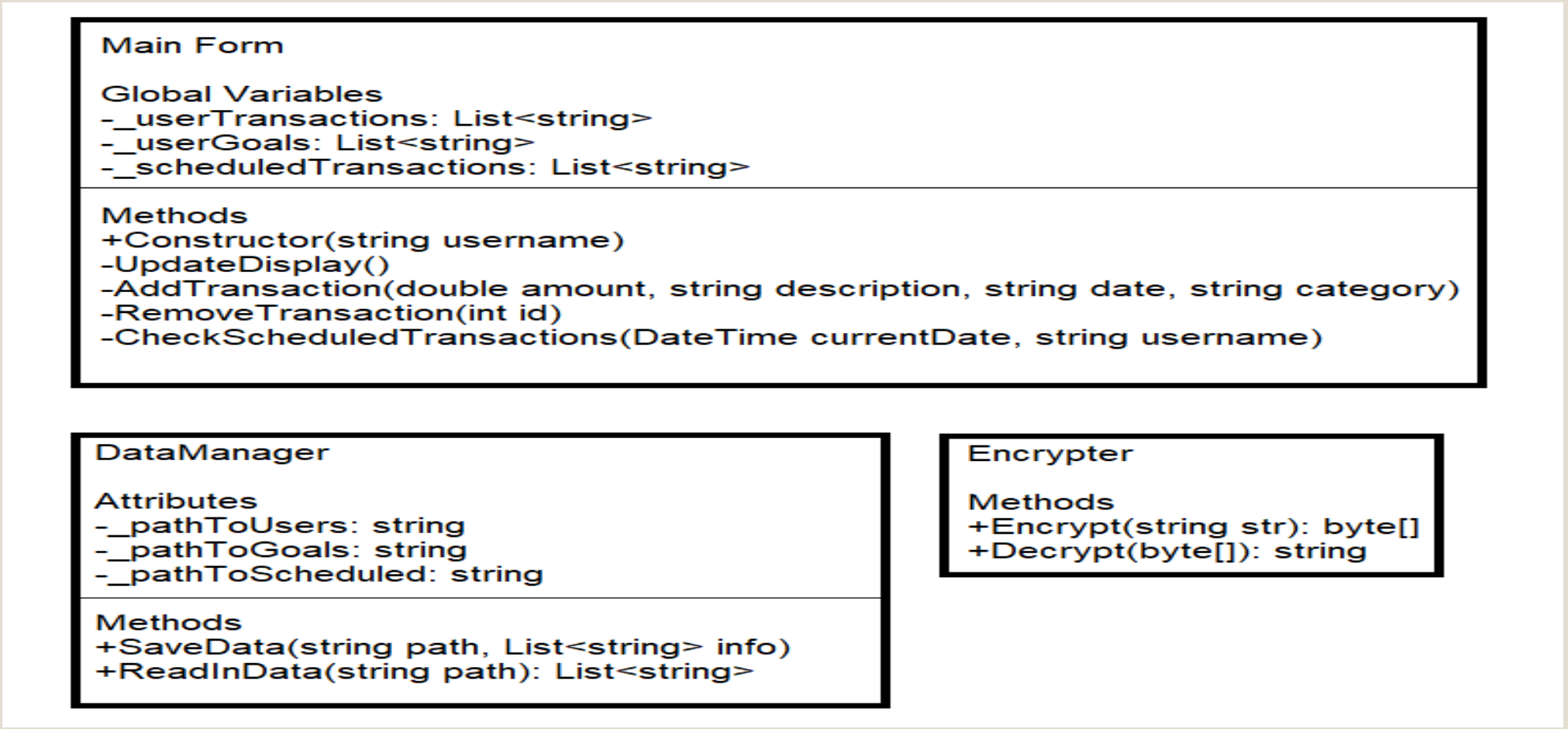
- Visual Studio 2017 and C#
  - There are currently no outside libraries.

- Text files are used for database.
  - These file are stored in folders on the user's local machine.
  - These files use a basic encryption method I made to mask data from the normal user.
  - In the future this would be switched to NoSQL database.

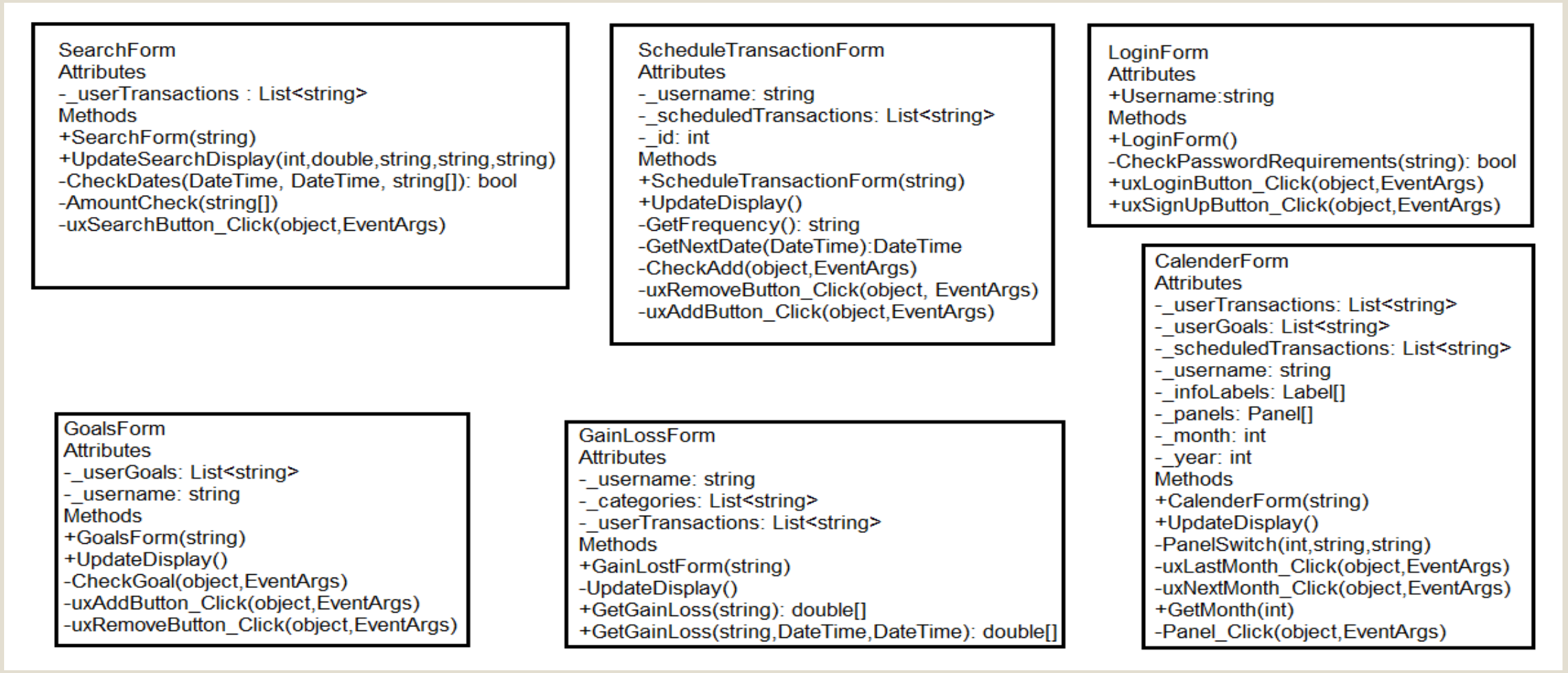
Main Design Goals

- Keep everything compact but not cluttered
  - Tried to not overwhelm the user while keeping things easy to use and navigate.
- Minimize the number of inputs the user must give.
  - I wanted to user to easily enter and remove data, so the amount of clicks required is about 3 on most forms for the minimum information.
- Avoid having the user type when not necessary
  - Adding a text box just gives the user another place to give bad input.
- Keep everything easy to read.
  - Having color coding and making sure everything is lined up properly.

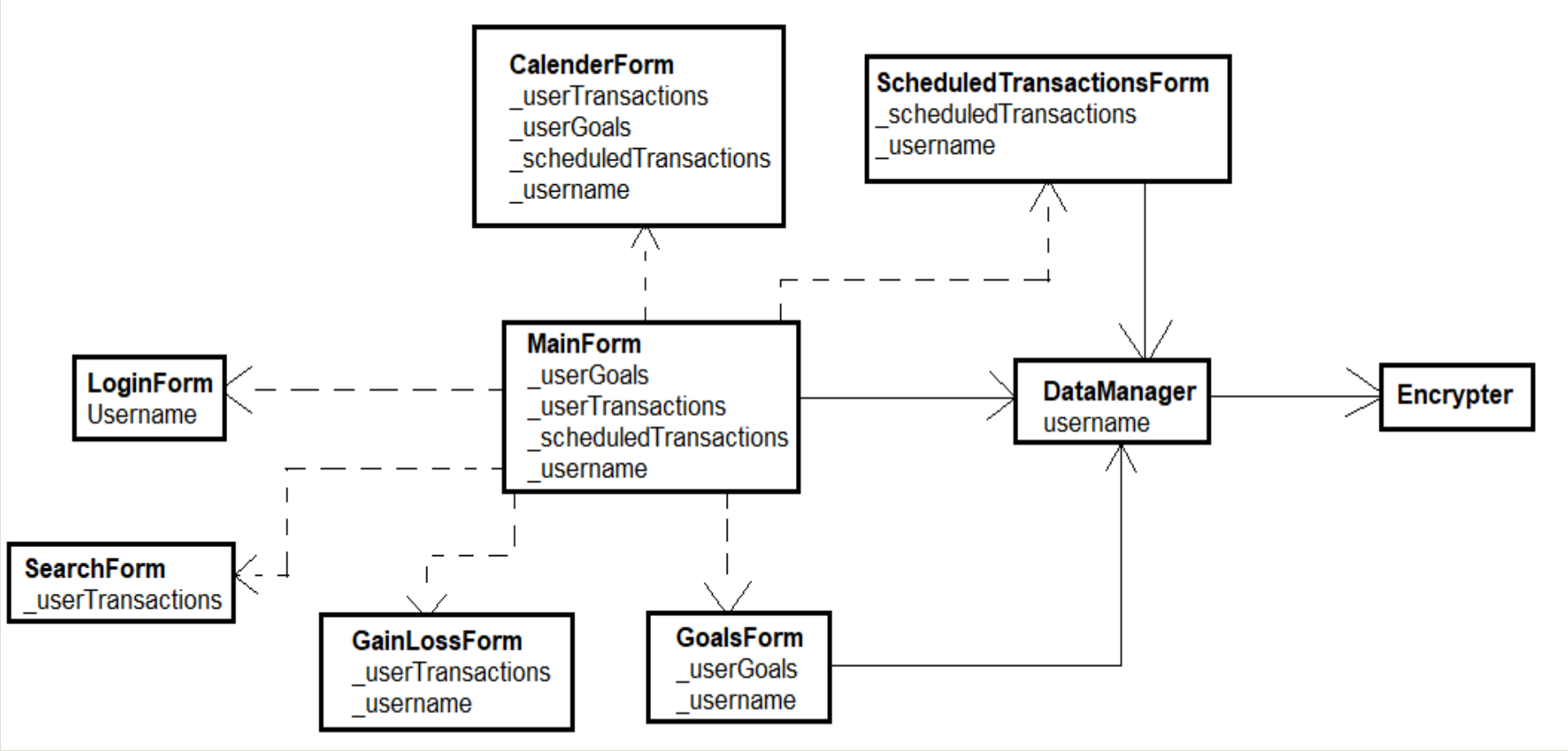
UML Diagram Part 1



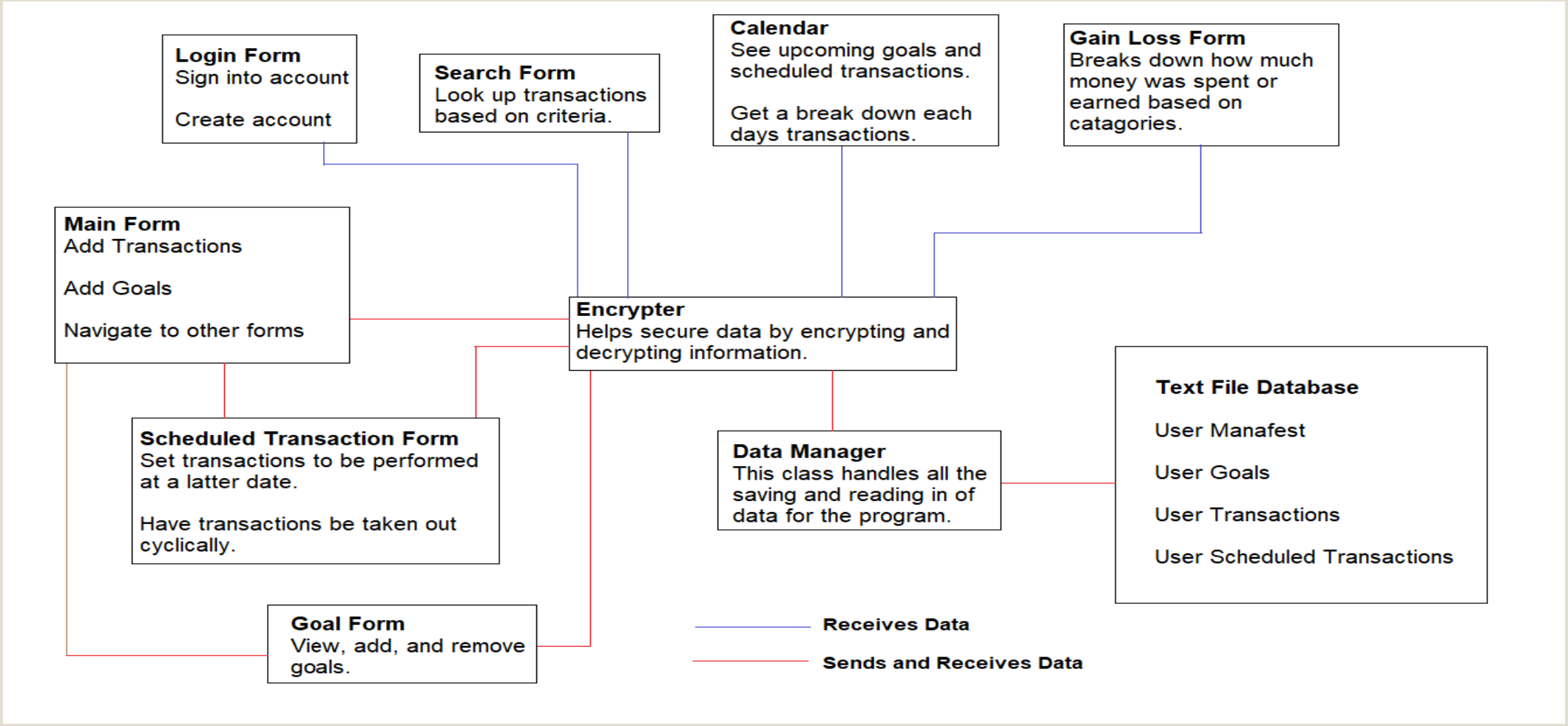
UML Diagram Part 2



Dependency Diagram



Flow Diagram



How It Works

- Data Manager is called before and after updates to keep data consistent.
  - This reduces that need of tossing anything but the username between classes.

- Main Form is like the home page of the program and will call all the other forms when needed.

Notes On Set Up

- The code is not as clean as I would have liked
  - Added and changed features during development.
  - Auto saving features made some original code redundant.

- I like to use static classes when I can.
  - I like to do it this way instead of making an instance of an object.
  - This lead to some problem's latter with Encryption.

DEMO

Testing

- All testing was done manually
  - C# and Visual Studio have unit testing library's but, the setup on these is a bit of a hassle when I must manually check the data anyway.

- Testing Steps
  1. Make sure only valid inputs could have the ability to be added
     Parsing numbers from text and checking string lengths.
  2. Enabling/disabling buttons when bad input is detected.
     Does the program know the input is bad before the user has a chance to add it.
  3. Make sure data was saved.
     Check if changes were saved in all locations and can be used by other forms.

Challenges

Dealing with scheduled transactions.

Making sure cyclical transactions were performed correctly

How to store data

Where to put data and how to hide it from the user.
Tossing away the SQL database.

How to handle multiple users

Even though this is a desktop app multiple people might use it, so I had to manage passwords and usernames.

Encrypting data

Pre-made encryption methods didn't work well with my schema so I had to make a simple one myself.
For a real launch this would be updated to AES standards.

Plans for future work

ASP.NET web app

I like ASP.NET and want more web development experience.

Get an actual back end database

Had a back-end SQL database but didn't really fit needs.

Add receipt checking

A cool feature that was suggested but I couldn't find a good image reading library. The text was always inaccurate.

Updates to UI

Keep updating forms as needed.

Questions