Financial Management Application
for UTM CSCI 352 Fall 2017

Lucian Freeze and Brett Whitson

Abstract

This project is a graphical application which allows a user to view and manage financial accounts and budgets. The target user for the project is primarily a college-age student; however, the application should be useful to anyone who wishes to use it. The intent is to provide said users the ability to manage finances in an easy-to-use, versatile environment.

1. Introduction

Students like (read: ”need”) to save money. Having a simple and intuitive application to manage ones finances would make this much more acheivable. This application intends to serve that purpose by giving the user the tools needed to achieve this goal.

A section to keep track of accounts and balances is to be implemented, allowing a user to view this information and add/remove accounts with possible customizations. Users will also be able to enter and edit transactions, dynamically updating the initial balance.

Another section for budget creation and editing is also to be implemented, allowing users to track how their money is spent based on categories.

A reports section will be implemented to allow users to visually examine their transaction data within a certain time period.

1.1. Background

Inspiration for this project came from the stereotypical financial struggle of a college student. We wanted to create an application that might possibly help those in this situation by allowing them to better manage their finances in an easy to use application.

1.2. Challenges

Possible challenges for this project include implementing graphical components and managing information’s security within the application. The ”Reports” section is to have elements such as dynamic graphs, which we are not yet certain of how to implement. Information security on a real-world scale is also a challenge we have yet to face, so this could prove to be difficult.

2. Scope

Our project aims to develop a simple, operable program in which users can securely and easily manage their finances. When we have fully implemented account and budget functionality along with some aesthetic aids and security measures, the project will be at completion.

Additional goals for this project include incorporating real-time investment profiles and a possible implementation of remote server-based account and credential storage/management.

2.1. Requirements

2.1.1. Functional.

- Users need user accounts – each user needs to only be able to access his/her personal data/settings and no one else’s
- Users need financial accounts – checking and savings accounts will need to be implemented and able to be created/edited by the user
- Users need budgets – budgets will need to be implemented and able to be created/edited by the user
- Users need transaction functionality – user will need to be able to edit and add transactions per account; possible database will be needed to created
- Users need reports – opt-in; gives user a weekly/bi-weekly/monthly representation of spending in categories; bar/pie graph form

2.1.2. Non-Functional.

- Security – user credentials and account information must be encrypted, users should be able to reset their passwords if forgotten
- Data integrity – user and especially account data must be reliably stored and accessible for successive use

2.2. Use Cases

Use Case #: 1
Use Case Name: Create User Account
Description: A user would like to create an account within the application. They will click on a "Sign Up" button. This event will transition the user to a new page where they can enter their user information and confirm their account.

Flow:

1) User left-clicks on "Sign Up" button on the initial page.
2) User is presented with a new page to enter user account details such as username, password, etc.
3) If username is not taken, user account is created and the "Sign In" page is shown. Otherwise, new username must be chosen.

At Termination: The user now has a user account and access to the rest of the application's utlilities.

Use Case #: 2
Use Case Name: Create Financial Account
Description: A user would like to create a financial account within the application. They will click on a "New Account" button. This event will trigger a new page to appear, asking the user which type of account to create. The user clicks the appropriate button and is presented with another page to enter account details.

Flow:

1) User left-clicks on "New Account" button on the Dashboard's Accounts tab.
2) User is presented with a new page with "Checking" and "Savings" buttons. User left-clicks the appropriate choice.
3) A new page is presented which allows the user to edit details for the chosen account type (name, balance, etc.)

At Termination: The user now has a financial account of the chosen type.

Use Case #: 3
Use Case Name: View Budgets
Description: A user would like to view their budgets within the application. They will click on "Budgets" tab. This event will trigger the "Budgets" page to appear. The user can then view their budgets.

Flow:

1) User left-clicks on the "Budgets" tab.
2) User is presented with the "Budgets" tab/page.

At Termination: The user is able to view their budgets.

Use Case #: 4
Use Case Name: Create new budget
Description: A user would like to create a new budget to track specific spending. They will click on the "Budgets" tab. This event will trigger the 'Budgets" page to appear. The user can then click the "New Budget" button to enter the budget details and allocate funds.

Flow:

1) User left-clicks on the "Budgets" tab.
2) User is presented with the "Budgets" tab/page.
3) User clicks the "New Budget" button.
4) User enters budget information.

At Termination: The user now has a new budget.

Use Case #: 5
Use Case Name: Edit/Delete budget
Description: A user would like to view their budgets within the application. They will click on ”Budgets” tab. This event will trigger the ”Budgets” page to appear. The user can then click the”Edit” button and will be presented with the edit page where they can reallocate funds or delete the budget altogether.

Flow:

1) User left-clicks on the ”Budgets” tab.
2) User is presented with the ”Budgets” tab/page.
3) User clicks the ”edit” button on the budget they wish to edit.
4) User is presented with options to edit or delete the budget.

At Termination: The user is able to edit fund allocation or delete their budget.

Use Case #: 6
Use Case Name: Edit/Delete Bank Account
Description: A user would like to edit or delete a Bank Account within the application. They will click on ”Accounts” tab. This event will trigger the ”Accounts” page to appear. The user can then click the”Edit” button and will be presented with the edit page where they can edit account settings/details or delete the account altogether.

Flow:

1) User left-clicks on the ”Accounts” tab.
2) User is presented with the ”Accounts” tab/page.
3) User clicks the ”edit” button on the Bank Account they wish to edit.
4) User is presented with options to edit or delete the Bank Account

At Termination: The user is able to edit account settings/details or delete the account.

Use Case #: 7
Use Case Name: View Reports
Description: A user would like to view Reports within the application. They will click on ”Reports” tab. This event will trigger the ”Reports” page to appear. The user can then view their Reports.

Flow:

1) User left-clicks on the ”Reports” tab.
2) User is presented with the ”Reports” tab/page.

At Termination: The user is able to view their Reports.

3. Timeline

3.1. Deliverable 1

Due: 9/5/17

3.2. Deliverable 2

Due: 9/21/17

3.3. Deliverable 3

Due: 10/12/17

3.4. Deliverable 3.5

Due: 10/12/17

3.5. Deliverable 4

Due: 11/21/17

3.6. Deliverable 5

Due: 11/28/17

3.7. Deliverable 6

Due: 12/5/17

3.8. Deliverable 7

Due: 12/6/17

4. Project Structure

User accounts and bank accounts depend upon an Access database file for storage between sessions. Users login through the initial sign in page and their data is read from the database, then is passed to the Dashboard. If a user does not have an account they access the sign up page to create one which is subsequently stored in the database. In the Dashboard, functionality is highly graphical with users being able to interact with the application interface largely through clicking, with occasional typing for editing. Users have a Checking Account and a Savings Account stored in the database and viewable/editable within the Accounts section of the dashboard. Transaction data is stored within the database for each account and can be viewed/edited by the user within each bank account. Budget data for the object's category (transaction type) will adjust as Transaction data is entered by the user. Reports will be generated using data from the transactions.

4.1. UML Outline

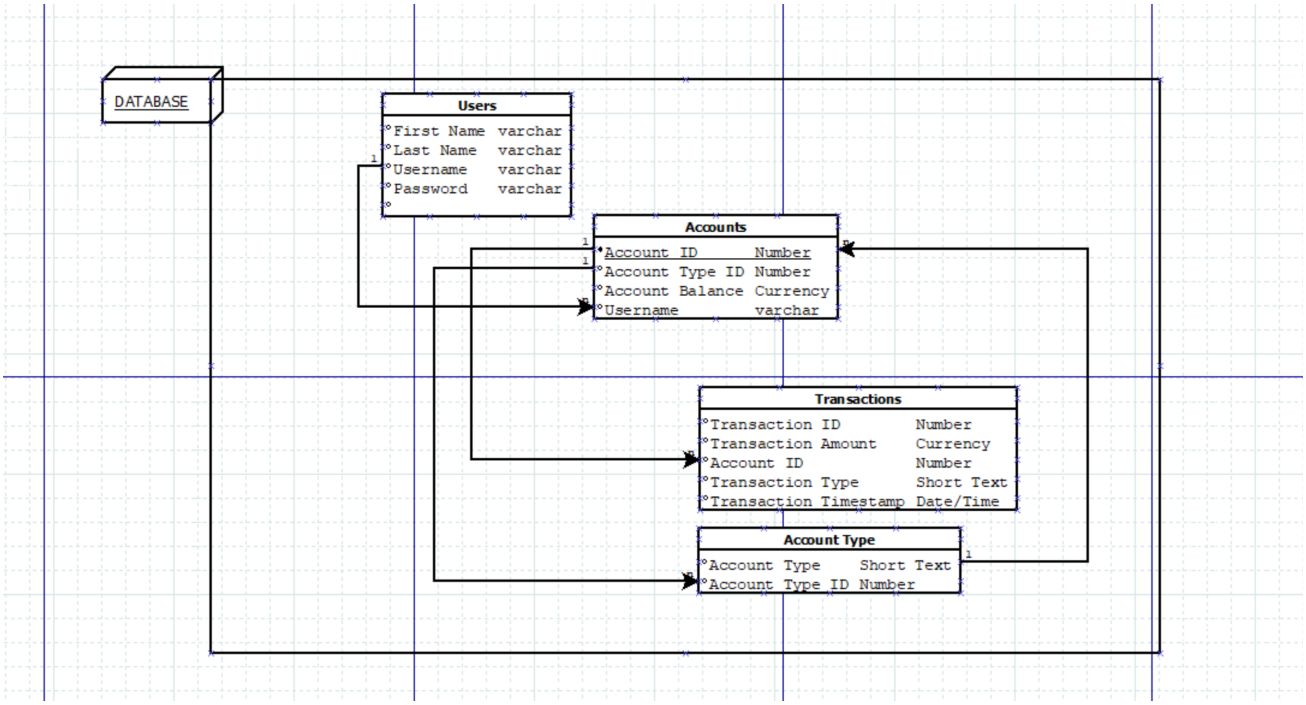

Figure 1. This is a DIA diagram of our program and database's structure.

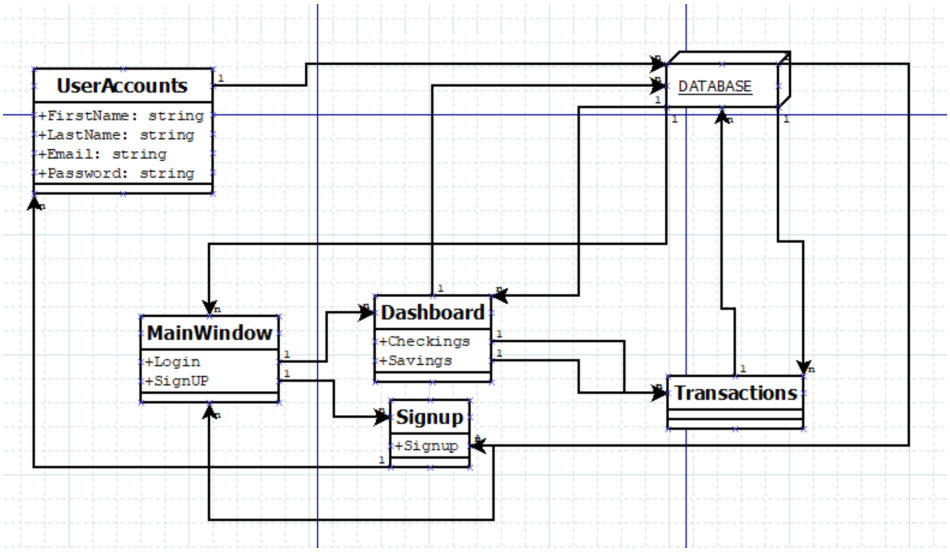

Figure 2. This is a DIA diagram of our program and database's structure.

4.2. Design Patterns Used

Though much of the application relies on interaction with a database, a theming feature is implemented with the Abstract Factory Pattern to give users the ability to change the application's appearance.

5. Interface Mockups

Figure 3. This image is a mockup of the initial sign in/up screen for our application.

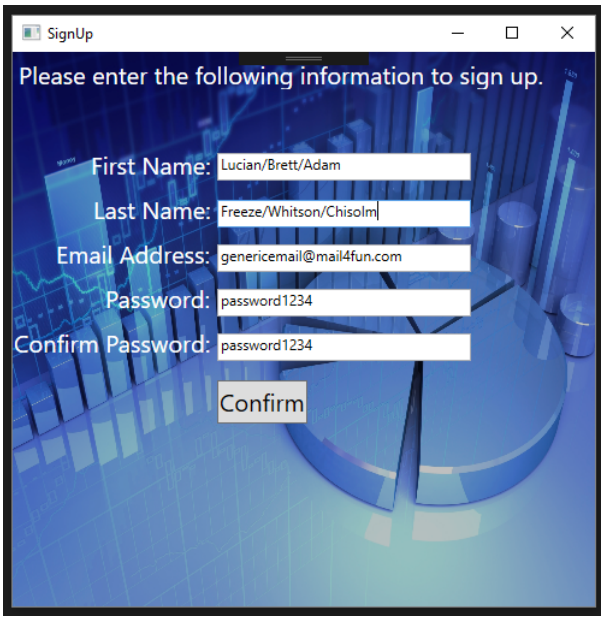

Figure 4. This image is a mockup of the sign up page for our application.

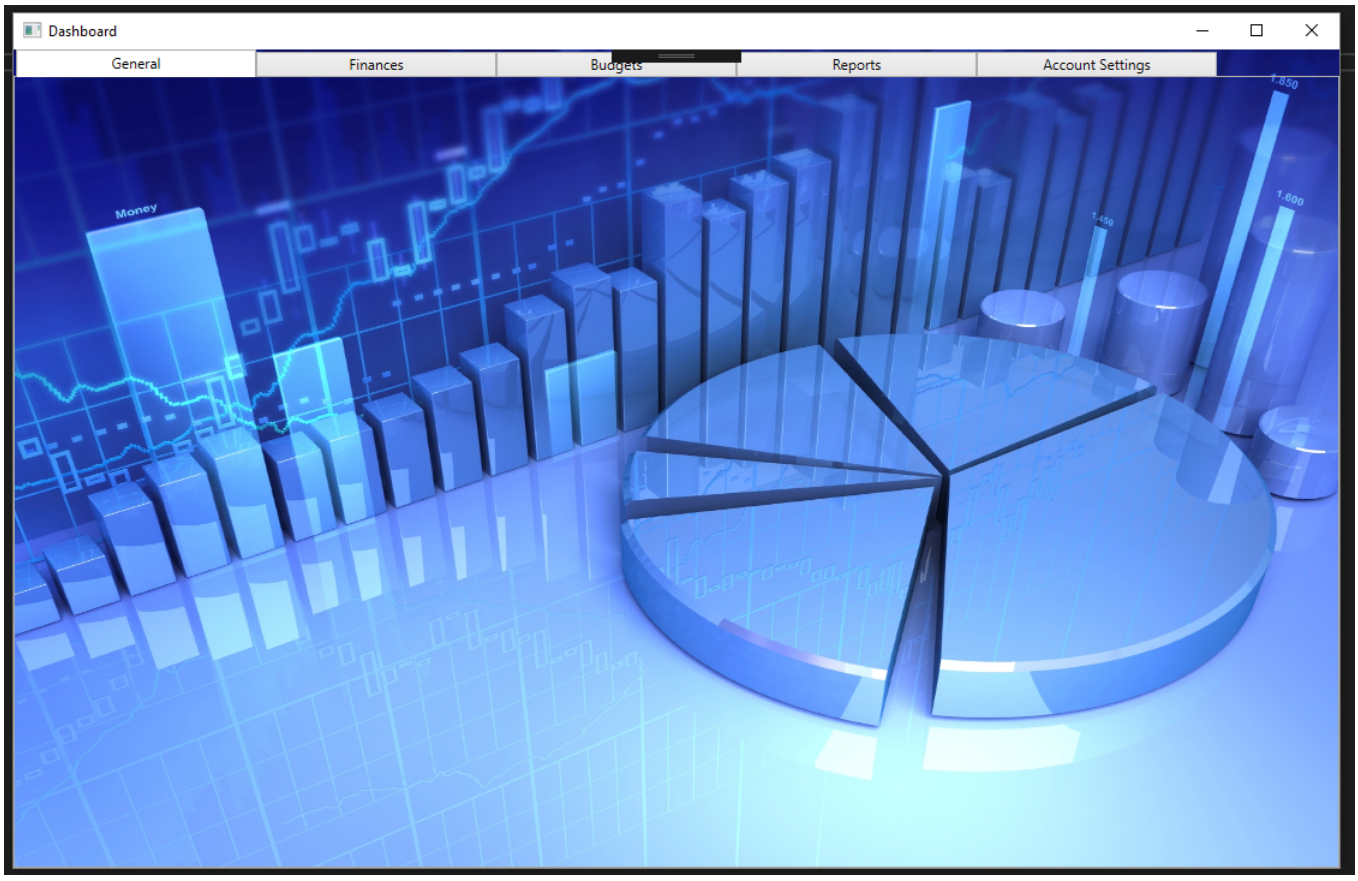

Figure 5. This image is a mockup of the Dashboard (the main user interface) for a logged-in user.