Expense Manager

Project Design Document

Team

- Amit Bhorania
- Ankit Luv Mittal
- Vrushali Gaikwad
- Kamesh Kishanth Gopinath

Professor

- Prof. Dov Kruger

Git Repository

https://github.com/amitbhorania/ExpenseManager.git

Introduction

“Beware of little expenses. A small leak will sink a great ship. ” (Benjamin Franklin)

This famous quote of Benjamin Franklin shows the importance of Expense Management. In a Daily Life, we spends our money for various purposes. For example Grocery, Travelling and Shopping etc. To maintain the list of expenses by hand and to analyze is very difficult. But if we have a software which perform all this tasks, keep a record of expenses for us, support of graphs for visualization and user friendly attractive GUI will be very useful. Keeping this goal in mind, we have decided to create a windows application for Expense Management.

Features

Included Features

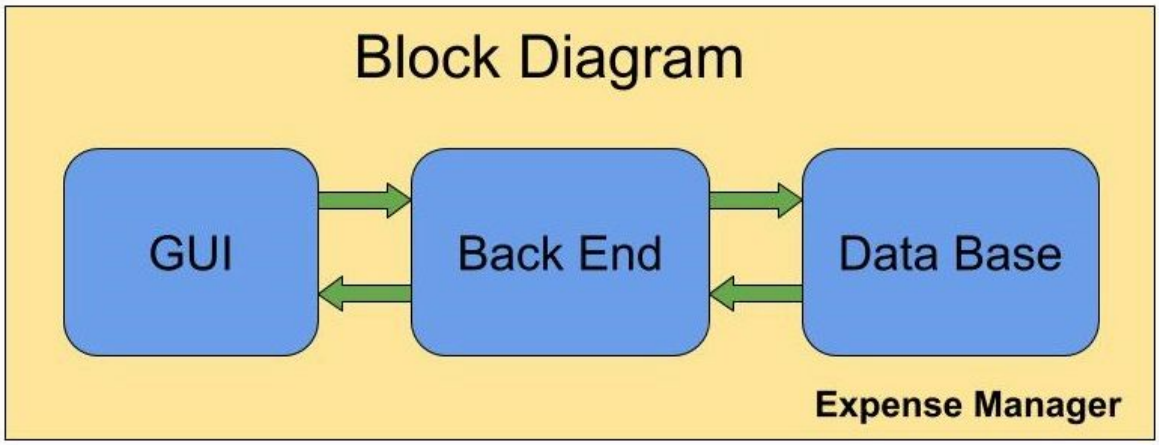
- Dashboard Display (2nd priority)
    - User name
    - Graphs of Income and Expense
    - Income Expense figures for Current time
    - List of Recent Transactions
    - Support for Daily/Weekly/Monthly/Yearly View Options
- Add Income and Expenses with (1st priority)
    - Amount
    - Description
    - Category of income/expense
    - Cash/Credit/Net Banking Transaction
    - Date
- User Account (3rd priority)
    - Register Multiple Users
    - Login with username and password
    - Forgot password with security question support
- Reset User Account

Optional Features

- View History of Income/Expenses
    - Daily/Weekly/Monthly/Yearly (Bar Graphs)
    - Category wise (Pie Graphs)
    - Payment Method wise (Pie Graphs)
    - Custom Graphs (Option to select the Start time and end time)
- Generate Report in printable format (.pdf, .excel, .csv or .jpeg format)
- Set Reminders (For example - Credit Card Bill Payment)
- Add Budget
- Add record of Money Transfer to Friends/Relatives (For example - $1000 given to friend)
- Option to select the Currency
- Add new Categories and Subcategories
- Desktop Notifications

As of now, we have planned to implement included features. If time permits then we will also try to implement optional features.

Block Diagram



Modules

Expense Manager would be designed as modular so that Each module will perform particular task. Modular design will also beneficial for easy debugging and bug fixing. Expense Manager consists following three modules

1. GUI
2. Back End
3. Data Base

Expense Manager will have a GUI from which user can access different features of the Application. GUI will internally pass the requests to the Back End and get the results and display the appropriate information to the user. Back End will have object oriented approach for handling different tasks. Back End will perform required tasks and if it requires information of Data stored then it will request Data Base Module. Database module will fetch the required information from the Files and send it back to Back End Module.

For example, User select the feature to display Graph of income and expense of last 6 months.
GUI - Request the Back end to provide it the last 6 months income and expense data. Once data received, plot the graph on GUI.
Back End - Query to database using the object oriented approach to fetch the data. Once data received, pass it to the GUI after processing.
Database - Read the required data from the files and send the information back to Back End.

Scope of Work

As of now, Modules assigned to each team member are as below.

- GUI - Vrushali and Kamesh
- Back End - Amit
- Database - Ankit

In future, work assignment may change as per the work requirement.

Tasks

Tasks required per module is discussed below and expected time required for each task is also mentioned.

Group Tasks

No	Task	Expected Time	Actual Time
1	Deciding the Project	5 hours	5 hours
2	Brainstorming for Possible Features	1 hours	1 hours
3	Project Design	3 hours	3 hours

Vrushali and Kamesh (GUI)

No	Task	Expected Time	Actual Time
1	Adding GUI Info in Class design document	2 hours	2 hours
2	Learning and getting used to the GitHub	6 hours	6 hours
3	Getting used to Qt and Environment Setup	4 hours	
4	Learning GUI in QT	16 hours	
5	Defining class and its methods	6 hours	
6	Feature Implementation		
	Creating User account login using GUI (Kamesh)	5 hours	
	Creating GUI Dashboard (Vrushali)	8 hours	
	Transactions (Income and Expense Entry) (Kamesh)	8 hours	

	Graphs (Vrushali)	6 hours	
6	Unit Testing and Error Fixing	2 hours	
7	Integration with Back End	4 hours	
8	Final Testing and Error Fixing	2 hours	
	Total Hours	Kamesh - 55, Vrushali - 56	

Amit (Back End)

No	Task	Expected Time	Actual Time
1	Preparing Class Design Document	6 hours	6 hours
2	Learning the GitHub Basics (Repo Creation, Branch, Pull, Merge etc.) Setting up the Work Environment with GitHub (master branch and individual module branch for each team member)	5 hours	6 hours
3	Framework Design (Source File, Header File, Top Level Class Creation etc.)	2 hours	
4	Defining various Class and its methods	4 hours	
5	Creating Expense Manager Simple Command Line Application without GUI for easy start and unit testing purpose	3 hours	
6	Implement Features		
	User Account	5 hours	
	Dashboard	6 hours	
	Add Income & Expense	6 hours	
7	Unit Testing & Bug Fixing	2 hours	
8	Integration with GUI and Database	8 hours	
9	Final Testing and Bug Fixing	2 hours	
Total Hours		49 hours	

Ankit (Data Base)

No	Task	Expected Time	Actual Time
1	Adding Database Class info in Class Design Document	3 hours	4 hours
2	Learning the GitHub Basics	6 hours	5 hours
3	Designing Database access	15 hours	
4	Defining various Class and its methods	3 hours	
5	Implement Features related Database		
	User Information	7 hours	
	Income & Expense Entry	4 hours	
	Dashboard	5 hours	
6	Unit Testing & Bug Fixing	4 hours	
7	Final Touch	4 hours	
Total Hours		51 hours	

Class Design

GUI

Login Account Class

```
// Class to Create GUI Login page
class GUI_login
{
private:
	string userNameGUI;
	string passwordGUI;
	// May require private variables to store the GUI widgets id

public:
	// Display Username
	ShowUserNameTextBox();

	// Display Password
	ShowPasswordTextBox();

	// Display Login Button
	ShowLoginButton();

	// Display Forgot Password Text
	ShowForgotPasswordText();

	// Display Forgot Password Text
	ShowSignUpButton();
};
```

Dashboard Class

```
// Class to Display the Dashboard on GUI.
class Dashboard_GUI {
private:
	// If required then will add the private members in future
public:
	// Displays the username given by the user
	showUsername();

	// Displays the Password given by the user
	showPassword();

	// Displays the Figure
	showFigure();

	// Displays the recent transaction done by the user
	showRecentTransaction();
```

```
        // Display the Tab for Options to view the data Daily, Weekly, Monthly and Yearly
        showOptionTab();

        // Displays the Income Button to Add new Transaction
        showIncomeButton();

        // Displays the Expense Button to Add new Transaction
        showExpenseButton();
};
```

Transaction GUI Class

```
// Class to Create Transaction Window
class Transaction_GUI {
private:
        double income_type;
        double payment;
        string category;
        string creditordebit;
        Transcatagory transaction;
        Paymenttype payment;

public:
        // Method to Display Transaction Window
        ShowTransactionPage();

        // Method to Display Amount Field
        ShowAmountTextBox();

        // Method to Display Description Field
        ShowDescriptionTextBox();

        // Method to Display Category Field
        ShowCategoryTextBox();

        // Method to Display Method of Payment Field
        ShowPaymentTypeTextBox();

        // Method to Display Date Selection
        ShowDateField();

        // Method to Display Submit button for Income/Expense Transaction
        ShowSubmitButton();
};
```

Enum Datatype for Graph type

```
// New Enum Datatype for Graph Type
enum graphType_t {
	DAILY = 0,
	WEEKLY,
	MONTHLY,
	YEARLY,
	MAX_GRAPH_TYPE
};
```

Graph Class

```
// Class to Create Graphs
class Graph{
private:
	graphType_t graph_type;
	int income_data[MAX_GRAPH_LENGTH];
	int expense_data[MAX_GRAPH_LENGTH];
	string time_data[MAX_GRAPH_LENGTH];

public:
	// Display Graph on the Dashboard
	// Main Method which will call other methods to plot whole graph
	showGraph();

	// Get Graph Data from Database
	getGraphData();

	// Method to Display Graph Window
	showGraphWindow();

	// Method to Display X Axis on Graph Window
	showXAxis();

	// Method to Display Y Axis on Graph Window
	showYAxis();

	// Method to Display Bar Graph on Window
	showBarGraph();

	// Method to get the graph type given by user
	getgraphtype();
};
```

Back End

Login Class

```
// Login Class to Handle Login Feature
class Login{
private:
	string userNameGUI;
	string passwordGUI;
	string userNameDataBase;
	string passwordDataBase;
	int result;
public:
	// Get the User Name from GUI
	getUserNameGUI();

	// Get the Password from GUI
	getPasswordGUI();

	// Get the Password from DataBase
	getPasswordDataBase();

	// Comapre the User Credentials with Data Base and Provide the result to GUI
	checkCredential();
};
```

Forgot Password Class

```
// Class to handle Forgot Password Feature
class ForgotPassword {
private:
	string userName;
	string securityAnswer;
	string newPassword;
public:
	// Get the User Name from GUI
	getUserName();

	// Get the Security Answer from DataBase
	getSecurityAnswerDataBase();

	// Get New Password from GUI
	getNewPassword();

	// Set New Password in Database
	setNewPassword();
};
```

Register New User Class

```
// Class to Register a New User
class RegisterNewUser {
private:
	string firstName;
	string lastName;
	string userName;
	string password;
	string securityQue;
	string securityAns;

public:
	// Get all the required Information from GUI
	getUserInfoGUI();

	// Check whether the Given Username already exists or not?
	checkForUserName();

	// Send the User Info to DataBase for Storage
	sendUserInfoToDataBase();
};
```

Date Class

```
// Class to Store the Date Format
class Date {
private:
	int month;
	int day;
	int year;

public:
	// Get/Set Methods
	getMonth();
	getDay();
	getYear();
	setMonth();
	setDay();
	setYear();
	getDateString();
};
```

Transaction Class

```
// Class for Income and Expense Transactions
class Transaction {
private:
	int type;	// Income or Expense
	double amount;
	string description;
```

```
        TranCategory_t category;     // Can take as String
        PaymentType_t paymentType;   // Cash/Credit/NetBanking
        Date date;     // Transaction Date

public:
        // Method to Get the Transaction details given by User
        getTransactionDetails();

        // Send the Transaction Details to DataBase to make an entry
        sendTransactionToDatabase();
};
```

Transaction Category Enum Data Type

```
// Type of Transaction
enum TranCategory_t {
        TRAVELLING = 0,
        FOOD,
        SHOPPING,
        EDUCATION,
        MEDICAL,
        RENT,
        OTHER,
        MAX_TRANCATEGORY
};
```

Payment Type Enum Data Type

```
// Type of Payment
enum PaymentType_t {
        CASH = 0,
        CREDIT_CARD,
        DEBIT_CARD,
        NETBANKING,
        MAX_PAYMENTTYPE
};
```

Reset Class

```
// Class to Reset the User Transactions
class Reset {
private:
        String userName;
Public:
        // Resets data for a user
        ResetData(string username, bool ack);
};
```

Database

Read Database Class

```
// Class to Read content from Database
class ReadDatabase {
private:
	int type;	// Income=0 or Expense=1
	double amount;
	string description;
	TranCategory_t category;	// Can take a String
	PaymentType_t paymentType;	// Cash/Credit/NetBanking
	Date date;	// Transaction Date

Public:
	// Check if user name is available
	ChkUserName(string userName, bool available);

	// Get User Info
	ReadUserInfo(string userName, string& firstName, string& lastName, string&
password, string& securityAns);

	//Get password from database
	ReadPassword(string userName, string& password);

	//Get Security Answer
	ReadSecurityAnswer(string username, string& securityAnswer, string& password);

	// Get the Transaction details from Database
	ReadTransaction(int& type,
	double& amount,
	string& description,
	TranCategory_t& category,
	PaymentType_t& paymentType,
	Date date);

	// Read Data for graphs
	ReadGraphData(Date firstDate, Date lastDate);
};
```

Write Database Class

```
// Class to Write content into Database
class WriteDatabase {
private:
	int type;	// Income=0 or Expense=1
	double amount;
	string description;
	TranCategory_t category;	// Can take a String
	PaymentType_t paymentType;	// Cash/Credit/NetBanking
```

```
        Date date;     // Transaction Date

Public:
        //Store User Info
        WriteUserInfo(string firstName,
        string lastName,
        string userName,
        string password,
        string securityAns,
        Date date);

        // Store the Transaction Details to DataBase
        WriteTransaction(int type,
        double amount,
        string description,
        TranCategory_t category,
        PaymentType_t paymentType,
        Date date,
        bool acknowledgement);

        // Store the Transaction Details to DataBase
        WriteNewPassword(string new_pass);

        //Reset the data
        ResetUserData(string userName );
};
```