A close up of a logo

Description automatically generated

Michael Fisher & Braydon Hampton

**Software Design**

**Document**

**Version: 1.2** **Date: 11/30/19**

**Table of Contents**

1. Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, Acronyms, and Abbreviations 3

1.4 References 3

1.5 Overview 3

2. The Overall Description 4

2.1 Use Case Diagrams 4

2.1.1 Login 4

2.1.2 Main Menu 6

2.1.3 Credit Menu 7

2.1.4 Debit Menu 8

2.1.5 Backup/Flush 9

2.2 Class Diagram 10

2.3 ER Diagram 11

2.4 Interface Description 11

2.5 Extra Visual Aid 14

# 1. Introduction

## 1.1 Purpose

The general purpose of this document is for developers to have a blueprint of the software design. It serves as documentation for developers Fisher & Hampton to build off of as well. The final purpose is for grading by Dr. Ekin in the Columbia College Software Engineering class.

## 1.2 Scope

1. Budgeting software for personal finance to be named BudgetME
2. Track user balance & transactions and organize the data into an ease of use user interface (UI). Software will NOT communicate with any bank accounts.
3. Goal of the software is to provide users with a place to keep track of their finances in an easy to use UI. Users can view how their financial standing holds, as well as if they have been allocated any spending money after calculating and comparing credits and debits.

## 1.3 Definitions, Acronyms, and Abbreviations.

**User:** The person, or persons, who operate or interact directly with the product. User(s) and customer not often the same person(s).

**Developers (devs):** Group developing the software project.

**Supervisor:** In this instance, our professor Dr. Ekin who is giving feedback on the documentation and development process for the project.

**F/H:** Fisher & Hampton respectively.

## 1.4 References

IEEE Std 1016‐1998: IEEE Recommended Practice for Software Design Descriptions

By IEEE Computer Society

## 1.5 Overview

The rest of the document contains a pinpoint representation of how the software should operate with detailed diagrams presenting the overall flow of the software.

## 2. The Overall Description

## 2.1 Use Case Diagrams

## 2.1.1 Login

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## 2.1.1 Main Menu

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## 2.1.3 Credit Menu

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## 2.1.4 Debit Menu

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## 2.1.5 Backup/Flush

A screenshot of a cell phone

Description automatically generated

## 2.2 Class Diagram

A screenshot of a cell phone

Description automatically generated

## 2.3 ER Diagram

A close up of a device

Description automatically generated

## 2.4 Interface Description–

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function Name | Class | Parameters | Return Types | Description |
| convert\_money() | Account | self, money | string | converts the value of the integer balance into a formatted string for displaying |
| pull\_transaction\_data() | Account | self | void | pulls the transactions for this account from the database - not mutating |
| push\_transaction\_data() | Account | self | void | pushes the transactions for this account to the database- mutating |
| create\_transaction() | Account | self, user\_id, account, account\_type, amount, date, datetime | void | creates a new transaction in the database. this method also pushes the transactions to the databases- mutating |
| convert\_money() | Goal | self, money | string | converts the integer value of the amounts into the formatted string version |
| create\_user() | User | self, fname, lname, username, password | void | creates a new user in the user- mutating |
| delete\_user() | User | self | void | deletes the user from the database - mutating |
| pull\_accounts() | User | self | void | pulls accounts from the database, whilst simultaneously pulling the transactions for each account - non mutating |
| push\_accounts() | User | self | void | pushes accounts to the database, whilst simultaneously pushing the transactions - mutating |
| create\_account() | User | self, user\_id, account\_type, balance, tag | void | pushes all current accounts, and then makes a new account on the database. finalizes by re-pulling accounts from the database - mutating |
| delete\_account() | User | self, account | void | deletes account from the database - mutating |
| pull\_goals() | User | self | void | pulls all the goals from the database - non mutating |
| push\_goals() | User | self | void | pushes goals to the database - mutating |
| create\_goal() | User | self, user\_id, account\_type, goal\_amount, tag, deadline | void | pushes all current goals, and then makes a new goal on the database, and then repulls all the goals from the database  - mutating |
| delete\_goal() | User | self, account | void | deletes goal from the database  - mutating |
| dosql() | DB\_Mediator | self, sql | list of dictionaries | executes an sql command and fetches the requested data from the database - non mutating |
| is\_user() | DB\_Mediator | self, username, password | bool | sees if the user exists |
| is\_username() | DB\_Mediator | self, username | bool | sees if the username exists |
| is\_user\_id() | DB\_Mediator | self, user\_id | bool | sees if user id exists |
| get\_user\_id() | DB\_Mediator | self, username, password | string | gets the user id based on the given username and password |
| get\_user\_data() | DB\_Mediator | self, user\_id | string | retrieves user data and returns it |
| create\_user() | DB\_Mediator | self, fname, lname, username, password | string | creates a user in the database |
| delete\_user() | DB\_Mediator | self, user\_id | void | deletes user from the database by their id number |
| get\_account\_data() | DB\_Mediator | self, user\_id, account, account\_type, balance, tag | string | gets an account's data from the database |
| set\_account\_data | DB\_Mediator | self, user\_id, account, account\_type, balance, tag | void | adjusts an account's dataset |
| create\_account() | DB\_Mediator | self, user\_id, account\_type, balance, tag | void | creates an account in the database based of the parameters |
| delete\_account() | DB\_Mediator | self, account | void | deletes an account from the database |
| get\_transaction\_data() | DB\_Mediator | self, account | string | fetches transaction data from the data base and returns it |
| set\_transaction\_data() | DB\_Mediator | self, user\_id, account, account\_type, amount, date, datetime | void | adjusts the data for a specific transaction |
| delete\_transactions() | DB\_Mediator | self, account | void | deletes all transactions for a specific account from the database |
| delete\_transaction() | DB\_Mediator | self, amount, date | void | deletes a specific transaction from the databsae |

## 2.5 Extra Visual Aids

A screenshot of a cell phone

Description automatically generated