CMSC 495 7980 Current Trends and Projects in Computer Science
Professor Christopher Howard
Back End Developer/Infrastructure: Nick Durkee
Front End Developer/Web Designer: Levi Riendeau

1 Introduction

1.1 Purpose

This document sets the requirements and plan for the development of a financial management web application for UMUC Class “Current Trends and Projects in Computer Science”. This document will set the preliminary design requirements and outline the technologies that will be used.

1.2 Scope

The financial management web application will be named GoFinance.

1.2.1 Required Features

- Separate User login
- Add new bank accounts
- Set initial bank account balance
- Remove Bank accounts
- Add spending categories
- Remove spending categories
- Add transactions (debit and credit) and assign to bank account and category
- Modify transactions
- Delete transactions
- View all transactions based on any criteria available (i.e. by name, account or category)
- Add budgets based on categories
- Assign dollar amount to budget
- Delete budget
- Modify Budget

1.2.2 Current Out-of-Scope Functionality

- Pulling transaction information from outside sources
- Scheduling bill payments

1.3 Objectives

This software is intended to assist an individual in managing their finances. It should be able to handle any transaction that a checkbook ledger should handle, but with the data being digital should save time as opposed to using it manually. Because of the data being digital repetitive tasks like copying transactions from the ledger to a budget will be automatic. Not only that but the data will be ready for any future reporting.

1.4 References

- IEEE Std. 1058-1998: IEEE Standard for Software Project Management Plans

1.5 Overview

This document contains the following diagrams:

3.1.2 Basic Page diagram

2 Overall Description

2.1 Product perspective

GoFinance, while intended as a basic finance application, will be entirely web based. Because of this it will be able to accessed by a wide variety of computer systems. There would not be many local dependences to run the application. The biggest requirement is that the system accessing the application is capable of viewing the web standard that the software is

built off of.

2.2 Interfaces

These are the requirements of the necessary interfaces

2.2.1 User Interface

User Interface will be defined by the web page. It must be simple and intuitive. Because the user could be using any number of devices to access the application, special care must be made to ensure the widest variety of accessibility. The front end must be able to be navigated whether accessed through a mobile phone, tablet or desktop computer.

2.2.2 Hardware Interface

Hardware interfaces will most likely be either a mouse and keyboard or through the touch interface on a mobile device.

2.2.3 Software Interface

Any web browser with JavaScript enabled will be the access point for our application.

2.3 Hardware Requirements

Server: A server capable of running docker (could be localhost on the client)
Client: Web Browser with JavaScript enabled

2.3.1 User Hardware

User is required to have a modern computing device. Any personal computer manufactured in the last 5 years should be able to work with the website and any mobile device newer than 4 years should suffice.

2.3.2 Server Hardware

Ideally, our "server hardware" will be handled by Kubernetes on a federated cloud infrastructure using Amazon Web Services (AWS) and Google Cloud Platform (GCP). Cloud hosting services offer a cost effective alternative to hosting with bare metal. They also provide painless infrastructure scaling horizontally. Kubernetes allows us to easily manage many future services associated with GoFinance and allows for great horizontal and vertical scaling when combined with cloud services. By federating Kubernetes across multiple providers, we are able to provide across more regions, we get to use the best of both services while maintaining a backup incase any one services goes down (like AWSs' us-east-1 did recently). However, for the scope of this project and for the sake of saving dollars, we will be hosting our application in a docker container on docker hub in a public repo for anyone to download and use!

2.4 Operations

We will be using GitHub for our source code hosting and version control. We will also be hosting the application (dockerized) on docker hub using dockers automated build process from the GitHub repo's master branch. Travis-ci.org will be used for automated testing, sending messages to the team when a build is failing. We will be using Git Flow for our workflow process (https://guides.github.com/introduction/flow/index.html).

2.4.1 User-initiated operations

User will initiate data entry for bank accounts, transactions, category creation and budget creation.

2.4.2 Scheduled operations

Releases will be based on passing builds, the latest release will always be available using the ":latest" docker tag on docker hub. A release schedule will be based on course requirements.

3 Specific Requirements

3.1 Interface

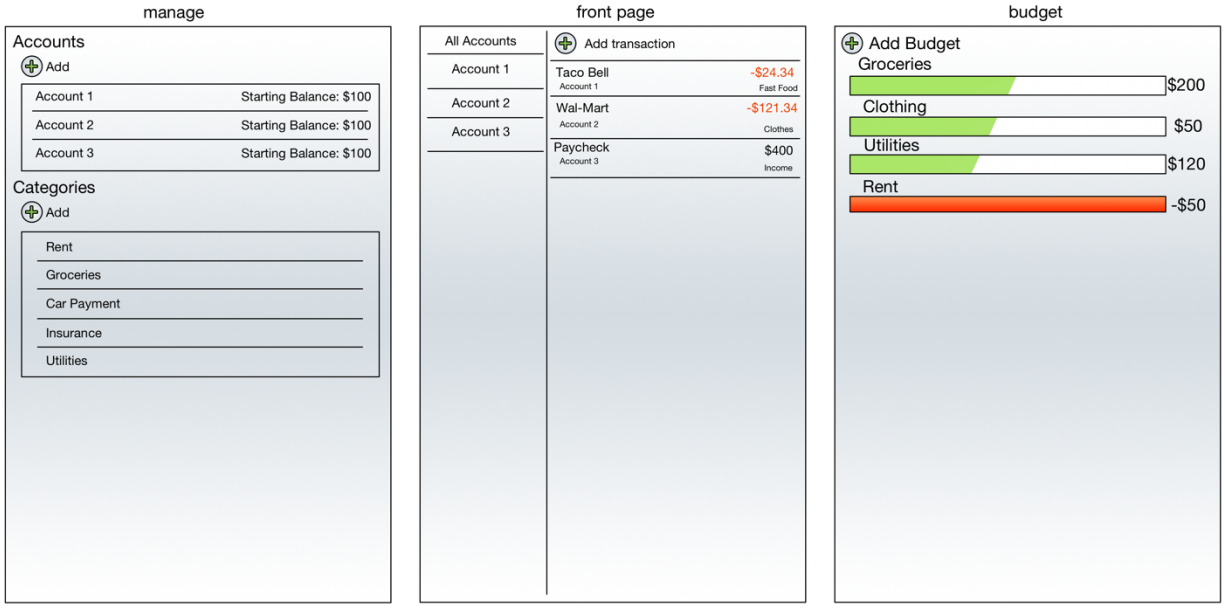
Web-interface

3.1.1 Title Bar

Title bar will be a basic navigation device that will allow the user to login in, switch between other pages and log out.

3.1.2 Pages

Basic Page diagram



3.1.2.1 Front Page

The front page displays all transactions between accounts. It will have the transaction amounts, what accounts they're implemented against and what category they fall under. All transactions need a category.

3.1.2.2 Manage

Manage page allows you to enter or remove bank accounts and set their starting balance. You can also add or remove categories through this page.

3.1.2.3 Budget

Page to set up budgets. Budgets can be added or removed from here. Budget is set by category of transactions. Bar is calculated by summing up transactions by month.

3.2 Objects

3.2.1 Accounts

This object holds the information of the bank accounts listed in the manage page.

3.2.1.1 Attributes

3.2.1.1.1 id

This is the id for the account on the database. This is unique for each account.

3.2.1.1.2 name

The name of the bank account

3.2.1.1.3 initialBalance

initial balance of bank account.

3.2.2 Budget

3.2.2.1 Attributes

3.2.2.1.1 id

Unique id for budget

3.2.2.1.2 name

name of budget

3.2.2.1.3 categoryId

ID of the category the budget uses

3.2.2.1.4 budgetAmount

Dollar amount set as budget

3.2.2.1.5 usedAmount

Total dollar amount of transactions this month that are attributed to the budgets categoryId

3.2.3 Transaction

3.2.3.1 Attributes

3.2.3.1.1 id

id for transaction

3.2.3.1.2 accountId

id of account the transaction is associated with

3.2.3.1.3 categoryID

id of category the transaction is associated with

3.2.3.1.4 description

description of transaction (i.e. name of store transaction happened)

3.2.3.1.5 amount

dollar amount for transaction

3.2.3.1.6 credit

boolean, if set to yes then transaction is a credit, if not then it is a debit

3.2.4 Category

3.2.4.1 Attributes

3.2.4.1.1 id

id for category

3.2.4.1.2 name

name of category

4 Technologies

4.1 Code

4.1.1 Back End

Backend will be program with the Go programming language. It will use standard libraries for the language. Methods will generate front end code that will be dynamically built from templates. Data will be stored and retrieved from a MySQL database.

4.1.2 Front End

Front end will be served from Go methods and HTML will be dynamically generated from the Go methods. Pages will be generated from templates that will use a combination of HTML and JavaScript with basic style information being stored in a standard CSS file.

4.2 Hosting

All code will be hosted on Github. This will allow coordination between team members

and versions control.

Once code is posted to GitHub repo it will be pushed to Travic IC where it is compiled and tested.

From there the code can pulled through Docker and tested on any system.

5 Code Access

5.1 Github

Link: https://github.com/durksauce/GoFinance

6 Schedule

	March 20 to March 26th	March 27th to April 2nd	April 3rd to April 9th	April 10th to April 16th	April 17th to April 23rd	April 24th to April 30th	May 1st to May 7th
Project Plan							
Test Plan							
Project Design							
Phase 1							
Phase 2							
Phase 3							
Final Phase							

6.1 Test Plan

A comprehensive test plan will be built with input from all team members. It will be used to ensure correct operation of all portions of the website

Team Members involved: Nick Durkee, Levi Riendeau

Infrastructure setup during this stage. All portions setup to run together (git, Travis CI and Docker) and sent to the rest of the team.

Team Member involved: Nick Durkee

6.2 Project Design

Overall design for the whole project. Must be completed by April 9th. This will outline exactly how the software will be built and all relationship between technologies.

Team Members involved: Nick Durkee, Levi Riendeau

6.3 Phase 1

Build database tables and database queries: This phase will setup up all of the groundwork for managing our data.

Team Member involved: Nick Durkee

Build static examples of all webpages.

Team Member involved: Levi Riendeau

6.4 Phase 2

Build methods for data pull and manipulation.

Team Member involved: Nick Durkee

Convert static web pages to templates

Team Member involved: Levi Riendeau

6.5 Phase 3

Build backend API calls and maintenance methods for database. Ensure backend calls are presenting information as needed.

Team Members involved: Nick Durkee, Levi Riendeau

6.6 Final Phase

Finish webpage formatting and design

Team Member involved: Levi Riendeau

Testing and final changes to code

Team Members involved: Nick Durkee, Levi Riendeau