EcoCosts Final Review

Christian Schmachtenberger,

Hunter Johnson,

Jackie Diep,

Joshua McGehee,

James Rinehart,

Durlab Man Sherchan [1]

[1] assigned by Dr. Michaela Elliot

Introduction

An everyday struggle

Financial freedom is difficult.

With EcoCosts, our clients will be able to easily plan out their budgets to achieve the ability to free the burden of their finances.

The structure of the application puts key financial analysis in the hands of the masses.

What is EcoCosts?

We designed EcoCosts as a website for users to create and easily access budgets suited for them.

Not only will budgets be personally designed and optimized through user-created categories, but also EcoCosts implements a stock watcher for the more financially savvy users.

1

High Level Requirements

Back-end

Back-end Requirements

- Transaction input:
  - Operational
- Double-entry bookkeeping:
  - Fully implemented
- Cash flow analysis:
  - Not implemented
- Categories:
  - Fully implemented
- Stock:
  - Operational
- Transaction strategies:
  - Not implemented

2

High Level Requirements

Front-end

Front-end Requirements (1)

Navigation:

- Fully implemented

Login & Registration:

- Fully implemented

Account details:

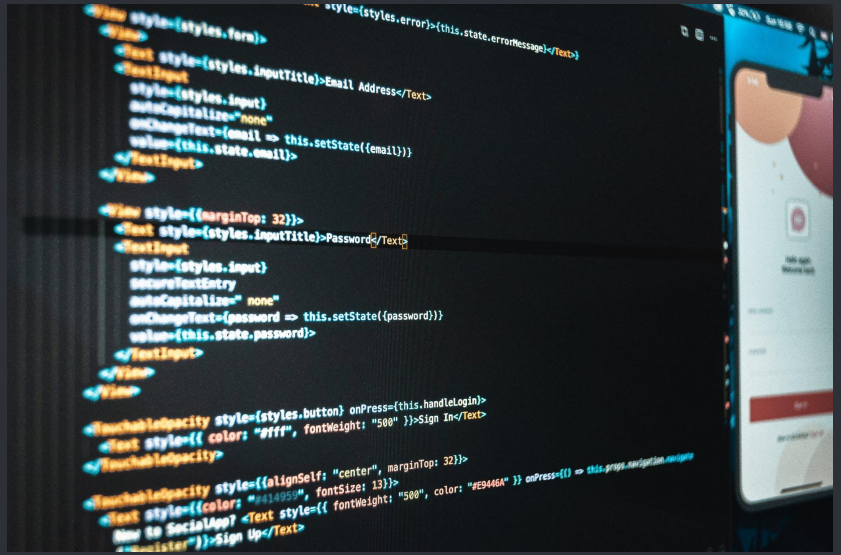
- Not implemented

Stock search:

- Partially implemented

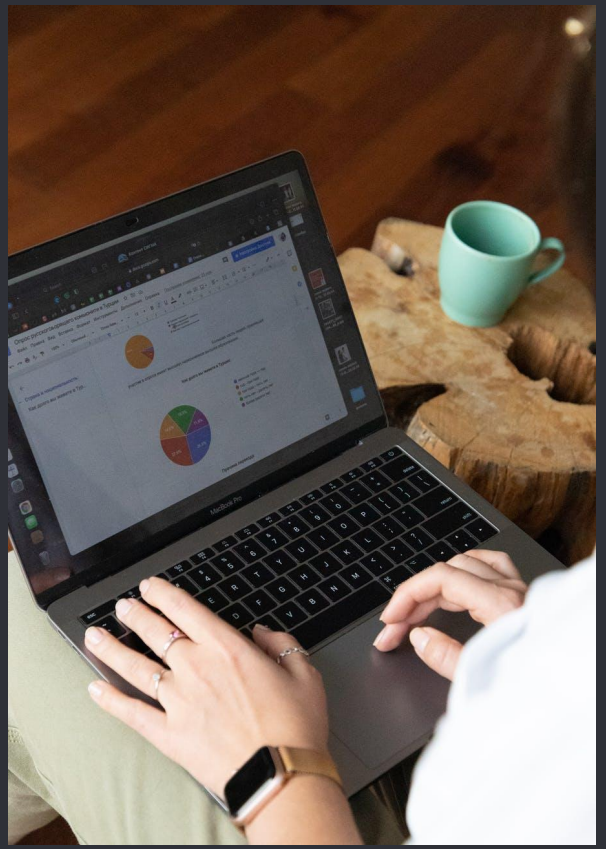
Stock watcher:

- Fully implemented



Front-end Requirements (2)

- Budget tracker:
  - Partially implemented
- Suggestions & tips:
  - Not implemented
- Categories:
  - Partially implemented
- Ledger:
  - Fully implemented
- Stock ticker:
  - Not implemented



3

High Level Requirements

Database

Database Requirements

Relational Schema:

- Fully implemented

Entity-relationship Diagram:

- Created

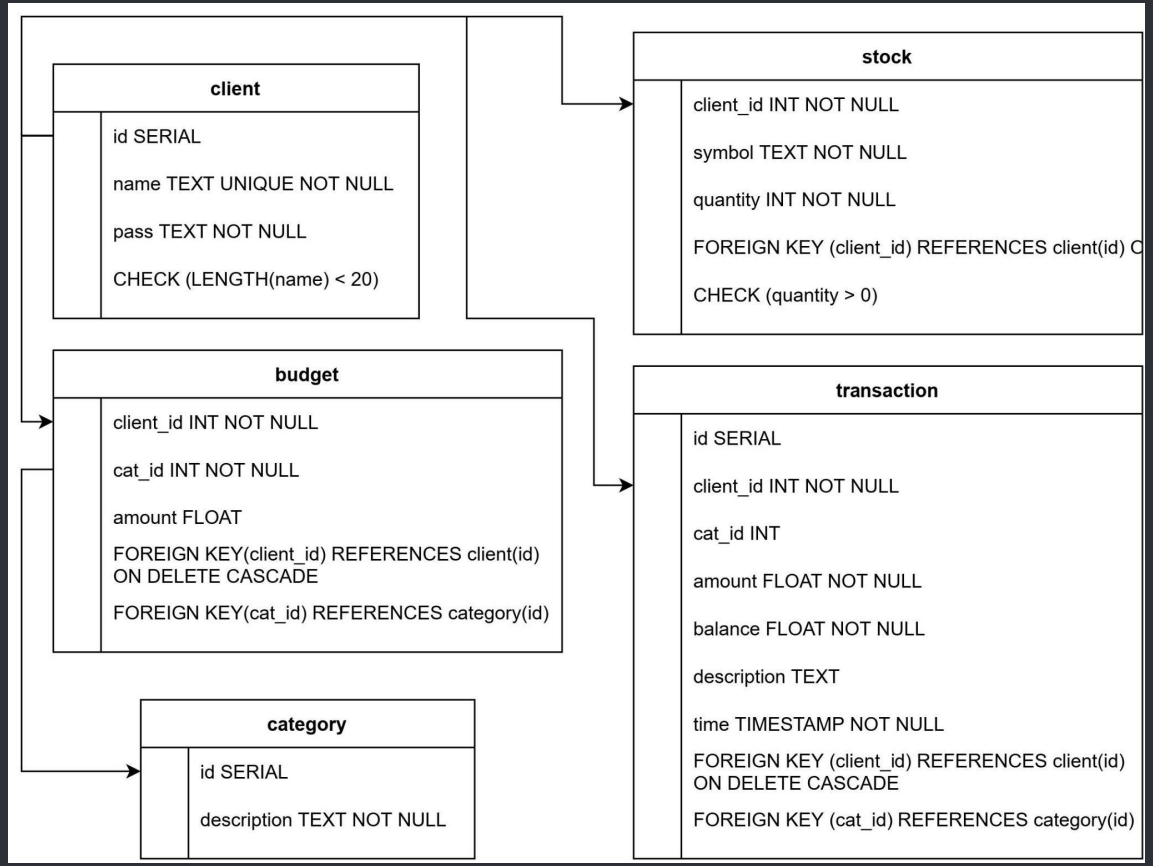
Sample SQL queries:

- Created

Transaction-based system for atomicity:

- Fully implemented

Entity-Relationship Diagram:

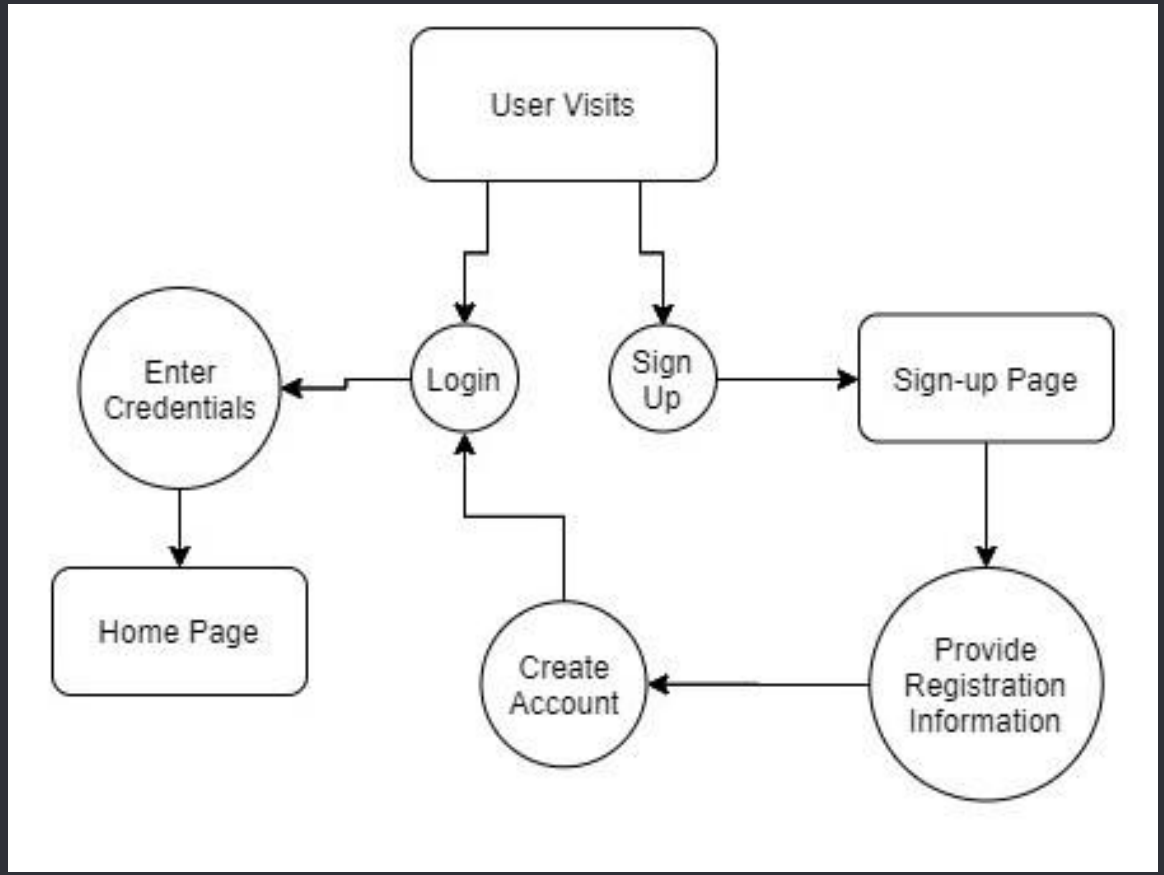


3

Final Design

Diagram and Visual Representations

State Diagram: Landing Page



State Diagram: Navigation

4

Final Design

Back-end

Back-end Design (1)

Transaction input shall:

- take user input and import into database using POST requests
- check atomicity between transactions using Golang sync/atomic and database/sql packages
- CSV imports were not implemented

Charting module shall:

- show categorical separation of user expenses
- graphical representations were not implemented
- representations of portfolio diversity were not implemented
- cash flow analysis charts were not implemented

Back-end Design (2)

Stock module shall:

- fetch json from Yahoo API for relevant stock short codes
- allow the user to enter stock names and search for the appropriate NASDAQ short code
- chart history of stock since user added to the budgeting

Budgeting module shall:

- allow the user to set budgets on certain categories
- create categories
- the category section of the charting module was not implemented

5

Final Design

Front-end

Front-end Design (1)

Navigation shall:

- take user to separate pages via navigation bar.

Login & Registration shall:

- allow the user to register

Account Details shall:

- an account details page separate from other pages was not implemented

Stock Search shall:

- be present on a dedicated page
- home page stock search was not implemented

Stock Watcher shall:

- be present on home page & display user selected stocks

Front-end Design (2)

Budget tracker shall:

- be present on a dedicated page
- home page budget search was not implemented

Suggestions shall:

- suggestions were not implemented

Categories shall:

- be available for sorting different budgets

Ledger shall:

- show all transactional data

Stock Ticker shall:

- the stock ticker below the navigation and display stocks were not implemented

6

Final Design

Database

Database Design

Database Module shall:

- initialize the database
- handle failed connections
- create structs for necessary tables

Database shall:

- ensure atomicity
- use transactions to handle failures
- fetch only necessary columns and rows
- use proper views

Team shall:

- plan the database using relational schema
- document the relationships between tables
- create test data (See Slide 24)

7 Test Plan and Results

Pseudo Data SQL file

A data.sql file will be created that will store SQL commands to create a test database that will be the same on all developers computers.

This will ensure that mixed data will not cause issues to go unnoticed or issues to occur during revisions.

Of course, more data will be chosen at the final testing stage to ensure that edge cases do not go unnoticed.

Environment

Windows and Linux were used to host, run, and test the application.

The Golang version will stay static throughout the project; however, no regressions should occur due to Golang, since the project is backwards compatible on all minor revisions.

PostgreSQL will also stay the same version to combat regression.

Integration

The project will feature modules that cover specific tasks as to combat any integration testing that would come from changing modules.

Bugs are bound to happen so isolating them to a specific module will help with integration.

Furthermore, the preliminary design will cover a modules' specific tasks tied to certain functions and data collection.

Test Results:

Front end testing was performed using various browsers on both Windows and Linux.

- Formatting and scaling issues were tested by resizing windows
- CSS code was visually verified throughout development
  - Both passed testing for the current implementation of the project

Back end testing was performed through various inputs

- Modules correctly return errors when invalid characters or values are implemented
  - An exception handler was not implemented at this time for the budget, transaction, or stock modules.

Database testing was performed through SQL queries to pgAdmin 4

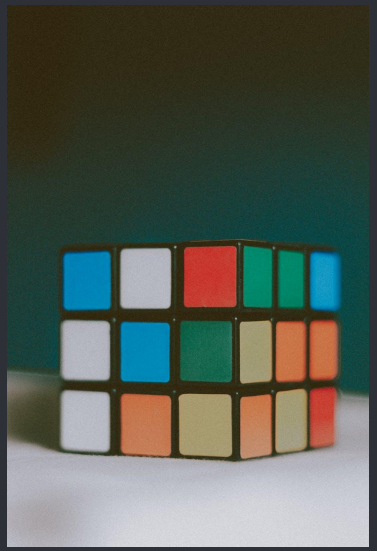
- The database correctly stores and returns information and ensures atomicity.

8 Problems and Challenges

Challenges

Things that could be done over time

- CSV Imports
- Graphical representations in all modules
- The home page portion of many modules
- A suggestions page to help with making decisions
- The dedicated stock ticker



- **ANY QUESTIONS?**

  Now is the time to ask!