ISENGARD

Premium Account Management Software

Created by

The Four Horsemen

Table of Contents

* Introduction
* Purpose
* System Overview
* Database Design
* System Security
* Roles of Each Member
* Project References
* Git Repository

Introduction

Isengard is designed to be a simple way to manage a single account that has multiple users making transactions. It has an intuitive layout and program flow, and is flexible to allow personalization.

Isengard stores your transactions and displays them in a user-friendly format. It gives you control to edit transactions, filter by user, create and save expense codes, and much more.

Administrators and regular users have different levels of visibility, and accounts can be managed by the administrator. The administrator has the control necessary to manage complex account user activity.

If you're looking for a program to meet your account management needs, look no further than Isengard!

Purpose

Isengard is your total package for keeping track of complex multi-user activity in a single account.

Transactions can be easily created and managed. They are formatted in a table on the main screen. Once created, they can be edited in a separate screen, and viewed in another.

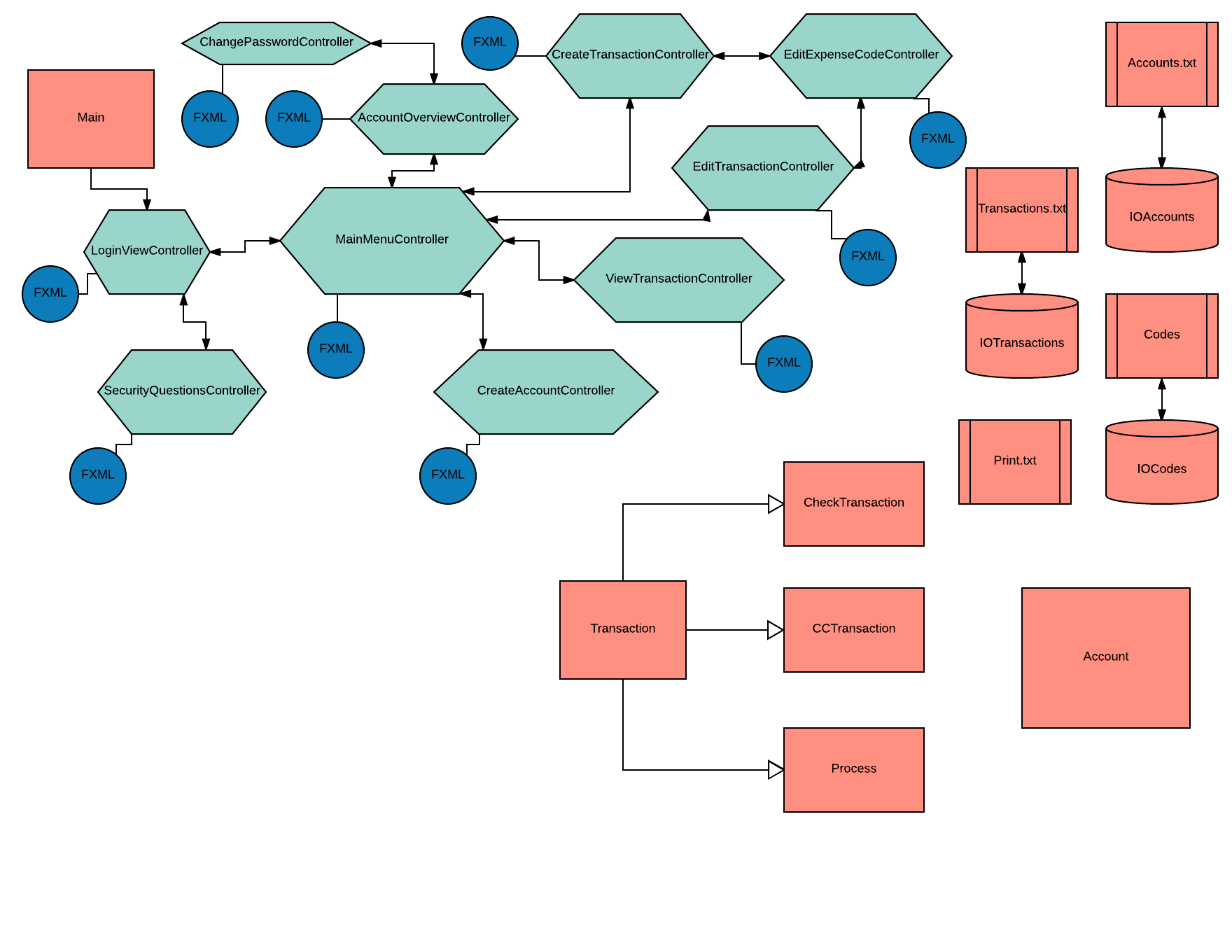
Fees are taken out automatically, and their amounts are saved separately. Expense codes can be easily created and edited from a handy menu inside of both the Create and Edit Transaction screens.

Accounts can be created and deleted by a single or multiple administrator. Administrators have unlimited control and view over the account, while users may only see transactions which they themselves have created. User data can be edited at any time.

The system is password protected for security. Other user features include a password recovery option, a date picker, and much more!

System Overview

Class Diagram and Program Flow



**Orange – Model Green – Controller Blue - View**

Functionality

1. Login
   * Password protected login
   * 'Forgot Password' option – Allows user to answer security questions to create new password
   * Ability to make visible characters in the password field
2. Main View
   * Formatted table for easy viewing of transactions
   * Tabs separating major program views
   * Side panel for selecting user accounts
   * Option to print a formatted transaction report
   * Ability to filter transactions by selecting user account in side panel
3. View Transactions
   * Allows user to see additional info about transactions, such as expense code and description
4. Create and Edit Transaction
   * Allows user to create or edit a transaction with following fields: user account, customer, date, type, expense code, amount, and description
   * Ability to create a new expense code
   * Ability to edit an existing expense code
   * Fields are auto-saved when going back and forth from 'add expense code' screen
   * Easy date selection with date picker
   * 'Edit Transaction' screen auto-fills fields with saved information
5. Account Overview
   * Displays easy view of user account data
   * Text fields are editable and can save account changes
6. Administrator
   * Create account allows creation of new user account
   * Delete account allows for deletion of existing account (does not remove transactions)

Non-functional Requirements:

1. Saves to and retrieves from text files all information including transactions, accounts, and expense codes every time program begins and ends, as well as when program saves information
2. All objects created in program (transactions, accounts, expense codes) are fully editable and may be deleted.

Database Design

Our database is a series of four classes (IOTransactions, IOAccounts, and IOCodes) that relay information to and from text files which are located inside the program folder.

System Security

The Isengard system is equipped with a password-protected login. If a password is lost, a user can create a new password after answering three personalized security questions. Internally, the administrator(s) can change account user data and edit transactions that are not their own; all other users may not.

Roles of Each Member

Scott McKay – Database design and management, GUI style editor, quality control

Jack Cummings – GUI design, user controller design and implementation

Dan Bailey – User account management, program flow management

Jake Wolfe – Transaction management, user controller implementation, scrummaster

Product References

Git Repository