Coin Haven (Banking)

Steven Newbern
CSI 2300

Coin Haven: Application Scenario

Coin Haven is a small bank application that allows you to manage and utilize your money. Seamlessly track, manage, and spend your money with intuitive features designed for simplicity and control.

Why this application?

Managing money is a big responsibility for us all, especially as college students. Its a practical tool to effortlessly manage finances.

How is it useful?

With features like easy account creation, intuitive budget tracking, and transparent spending insights, it simplifies the complexities of student finances.

What problem is it solving?

Whether saving for textbooks, managing part-time job earnings, or tracking daily expenses, this app offers a user-friendly solution that aligns with the dynamic and busy lifestyle of college students and those alike.

Actions & Capabilities

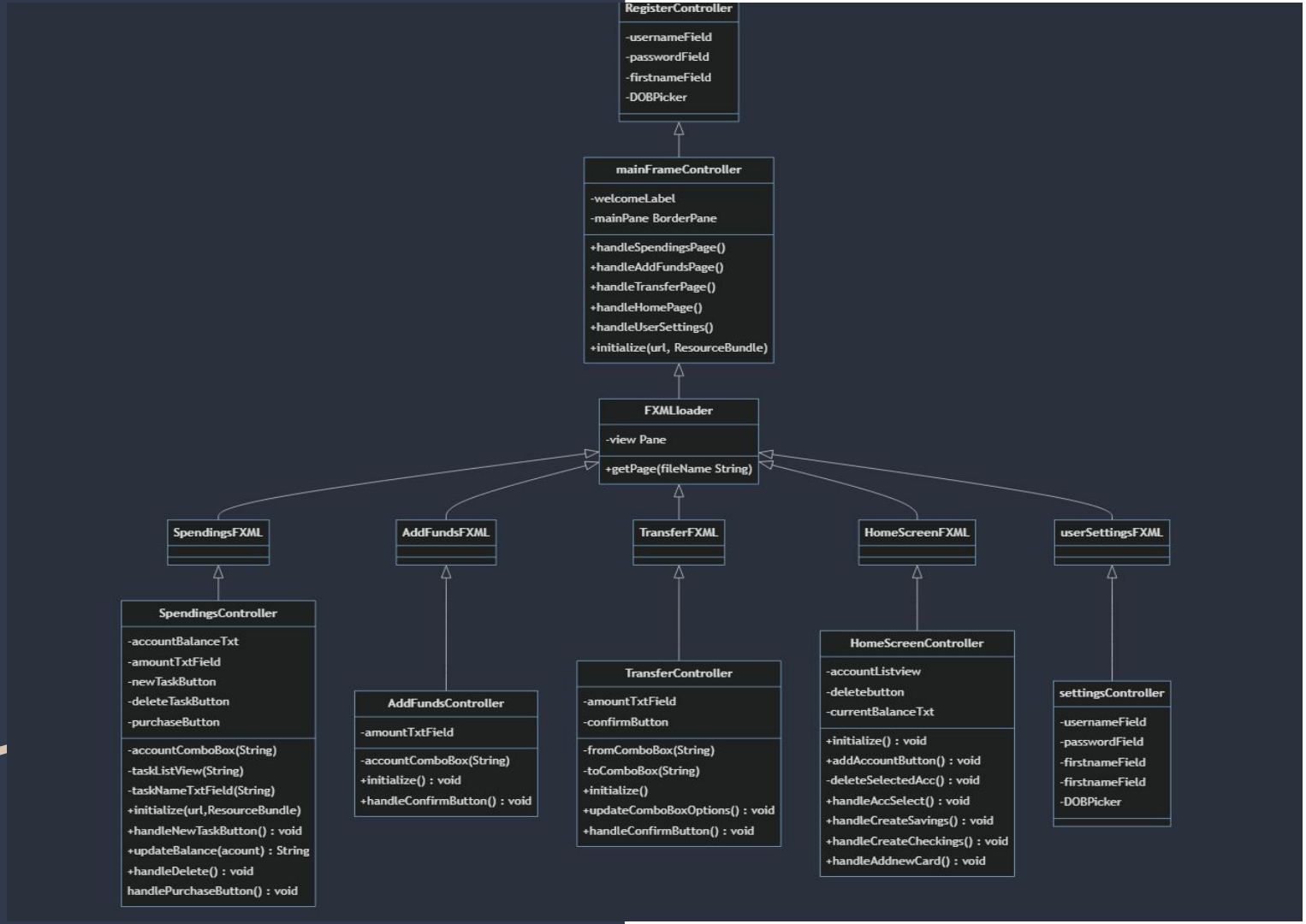
This application is capable of performing many actions to help users view and manage their savings when needed. These features are fairly simple and easy to use.

- Creating Checkings Account(s)
- Create Savings Account(s)
- See available balance for accounts
- Add money to specified account
- Transfer money between accounts
- Manage Spendings / planning

Project Design

This program is made up of many different files each serving a distinct purpose to make the program run efficiently.
Most of the othe code / files can fall under one of four categories:

- ***.fxml file*** **-** XML base UI similar to HTML (creates page layout / structure)
- ***Controller files*** **-** Java classes responsible for managing the logic and behavior of different sections of the program
- ***In Memory Database*** **-** A class that acts as in memory storage for user data, account information, and tasks. It provides methods to interact with stored data.
- ***.CSS file*** **-** styling classes used by FXML to create a more visually appealing interface


RegisterController
-usernameField
-passwordField
-firstnameField
-DOBPicker
mainFrameController
-welcomeLabel
-mainPane BorderPane
+handleSpendingsPage()
+handleAddFundsPage()
+handleTransferPage()
+handleHomePage()
+handleUserSettings()
+initialize(url, ResourceBundle)
FXMLloader
-view Pane
+getPage(fileName String)
SpendingsFXML
AddFundsFXML
TransferFXML
HomeScreenFXML
userSettingsFXML
SpendingsController
-accountBalanceTxt
-amountTxtField
-newTaskButton
-deleteTaskButton
-purchaseButton
-accountComboBox(String)
-taskListView(String)
-taskNameTxtField(String)
+initialize(url,ResourceBundle)
+handleNewTaskButton() : void
+updateBalance(acount) : String
+handleDelete() : void
handlePurchaseButton() : void
AddFundsController
-amountTxtField
-accountComboBox(String)
+initialize() : void
+handleConfirmButton() : void
TransferController
-amountTxtField
-confirmButton
-fromComboBox(String)
-toComboBox(String)
+initialize()
+updateComboBoxOptions() : void
+handleConfirmButton() : void
HomeScreenController
-accountListview
-deletebutton
-currentBalanceTxt
+initialize() : void
+addAccountButton() : void
-deleteSelectedAcc() : void
+handleAccSelect() : void
+handleCreateSavings() : void
+handleCreateCheckings() : void
+handleAddnewCard() : void
settingsController
-usernameField
-passwordField
-firstnameField
-firstnameField
-DOBPicker


RegisterController
-usernameField
-passwordField
-firstnameField
-DOBPicker
mainFrameController
-welcomeLabel
-mainPane BorderPane
+handleSpendingsPage()
+handleAddFundsPage()
+handleTransferPage()
+handleHomePage()
+handleUserSettings()
+initialize(url, ResourceBundle)
FXMLloader
-view Pane
+getPage(fileName String)
SpendingsFXML
AddFundsFXML
TransferFXML
HomeScreenFXML
userSettingsFXML


SpendingsController
-accountBalanceTxt
-amountTxtField
-newTaskButton
-deleteTaskButton
-purchaseButton
-accountComboBox(String)
-taskListView(String)
-taskNameTxtField(String)
+initialize(url,ResourceBundle)
+handleNewTaskButton() : void
+updateBalance(acount) : String
+handleDelete() : void
handlePurchaseButton() : void
AddFundsController
-amountTxtField
-accountComboBox(String)
+initialize() : void
+handleConfirmButton() : void
TransferController
-amountTxtField
-confirmButton
-fromComboBox(String)
-toComboBox(String)
+initialize()
+updateComboBoxOptions() : void
+handleConfirmButton() : void
HomeScreenController
-accountListview
-deletebutton
-currentBalanceTxt
+initialize() : void
+addAccountButton() : void
-deleteSelectedAcc() : void
+handleAccSelect() : void
+handleCreateSavings() : void
+handleCreateCheckings() : void
+handleAddnewCard() : void
settingsController
-usernameField
-passwordField
-firstnameField
-firstnameField
-DOBPicker
InMemoryDatabase
-users List[User]
-savingsAccounts List[SavingsAccount]
-checkingsAccounts List[CheckingsAccount]
-cards list [Card]
-tasks List[String]
+addUser(username,password) : void
+addSavingsAccount(accountName,DateCreated,startingBalance) : void
+addCheckingsAccount(accountName,DateCreated,startingBalance) : void
+addCard(card#,cardHolder,expireDate,CCV) : void
+addTask(task Sting) : void
+getTasks() : List [String]
+getAccountNames() : List [String]
+getAccountType(accountName string) : String
+getStartingBalance(accountName string) : String
+updateSavingsBalance(accountName string, amount double) : void
+updateCheckingsBalance(accountName string, amount double) : void
+deductAmount(accountName string, amount double) : void
removeTask(task string) : void
+transferFunds(fromAccount, toAccount, amount double) : void

Challenges & Difficulties

While creating this application, there were many roadblocks and challenges that I struggled with. But, I managed to overcome most of these difficulties or atleast find an alternative

Design
A large amount time was spent constantly changing and altering the layout of the application. What helped me with this was creating a mock up sketch to get an idea of each frame before coding it.

Database complications
I had some trouble setting up a database. I originally started with *SQLite*, which I've used before in python, but it's a bit different in java (mainly syntax). Then I switched to *h2 database*, but similar issue came about. Finally, I decided to store data using a set of *ArrayList* of *Hashmaps*.

Application size
This project taught me how to better manage my files and code. There are dozens of files for this app consisting of FXML's, controllers, CSS (Styling), and javafx methods. This app was much larger than I initially intended, so I had to learn to manage and connect these files together.

What I’ve learned

Learned more about GUI’s and how to use JavaFX. Before, I had a simple understanding of this but now I know a bit more about the complexities of how a GUI interacts with backend data manipulation and storage.

I also have a better understanding on how to use ArrayList / hash functions. I didn’t really know how to use it in a practical way. So part of me feel that it’s a good thing that I had to resort to using it as I was forced to gain a better understanding.

The *SceneBuilder* application helped a lot when it came to create FXML and instantly getting visual feedback while designing the program.

Overall, I have a lot more knowledge and experience using java and the various aspects of OOP. Along with CSS (styling) and FXML.