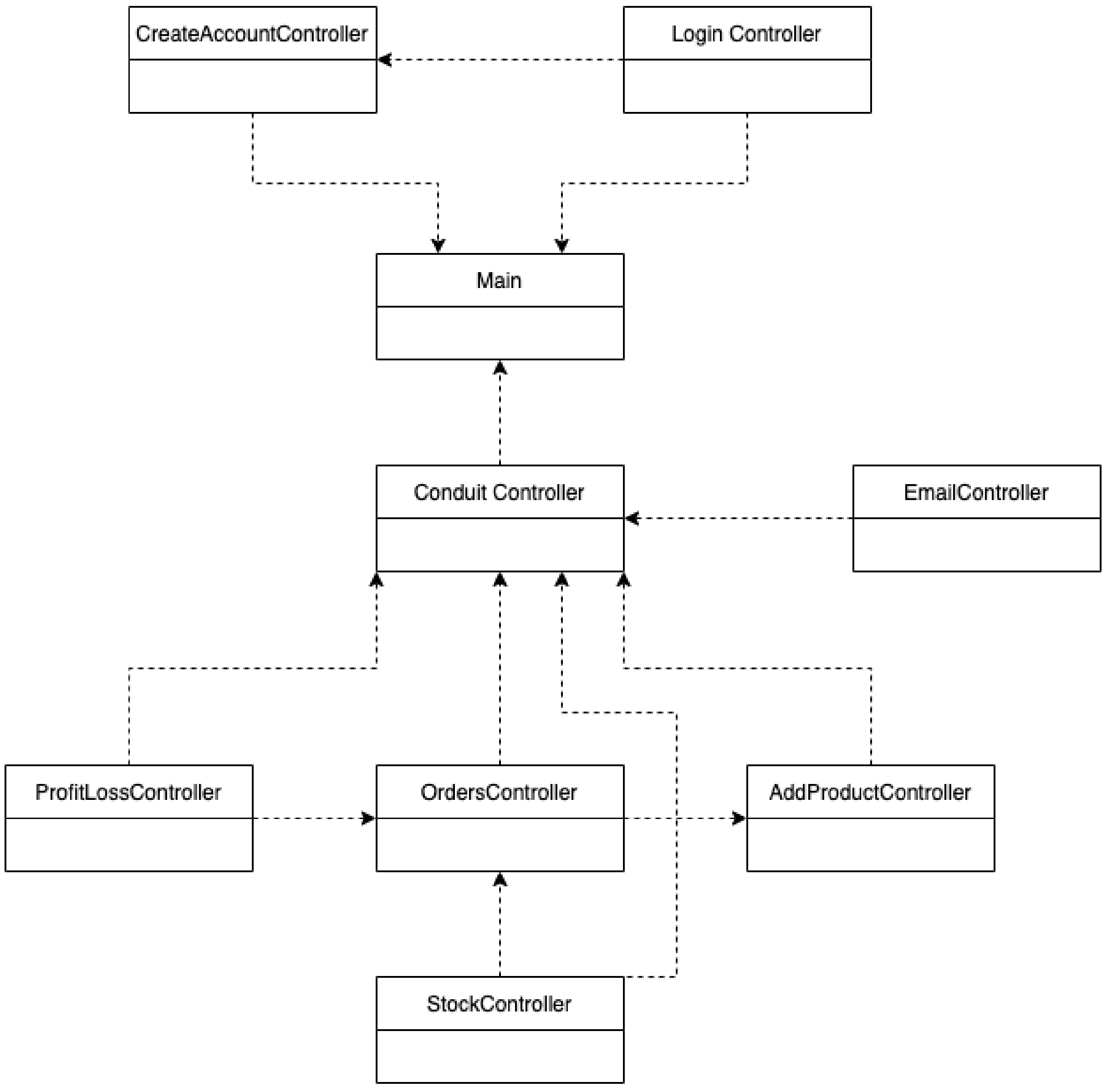
Criterion B: Design

Relationship between the classes:



Functionality of Each Class:

Main: Starts the program and allows for travel between GUI´s

Conduit Controller: Serves as a way to choose which GUI you want to go to.

Orders Controller: Creates “orders” with attributes such as Product name, Customer name, Date of Order, Price and Quantity and saves them to a table in the GUI. It also saves the information to a CSV File.

Add Product Controller: Creates “Products” with attributes such as Name, Total Cost for product, Price and Stock. Which is used in the Orders GUI and is also saved in a CSV File.

Stock Controller: Creates a table to visualize the amount of stock remaining of each product.

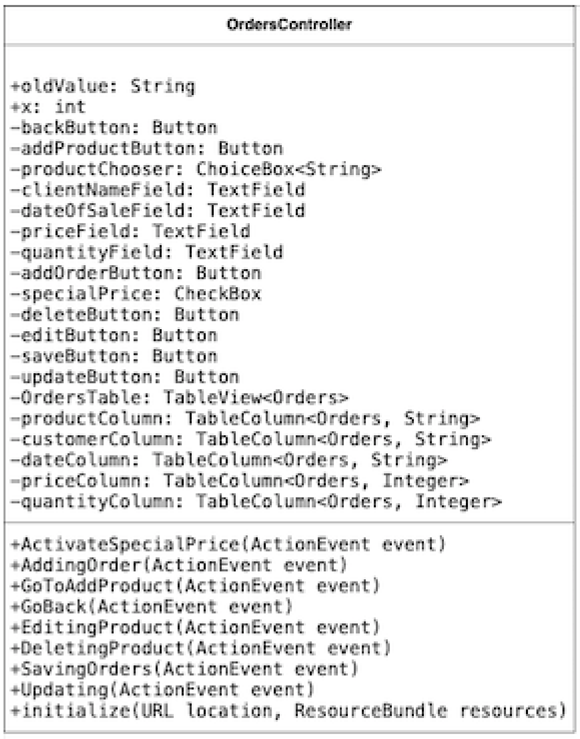
Profit Loss Controller: Creates a table to visualize the amount of profit and/or loss of each product.

Email Controller: Sends Emails with text to a mail provided by the user.

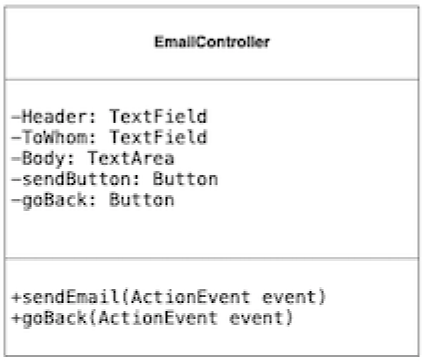
Login Controller: Verifies if the Username and/or Password is saved in the database and if so allows the user to access the complete application.

CreateAccountController: Allows the user to create a new account that is saved onto the CSV File database.

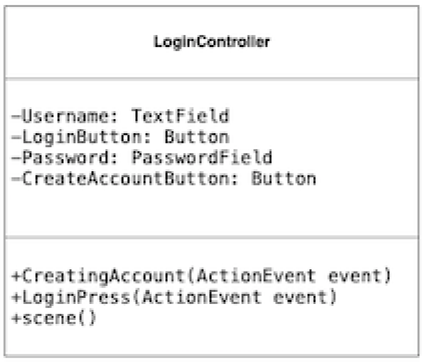
Class diagrams showing members of each class and Important Algorithms:



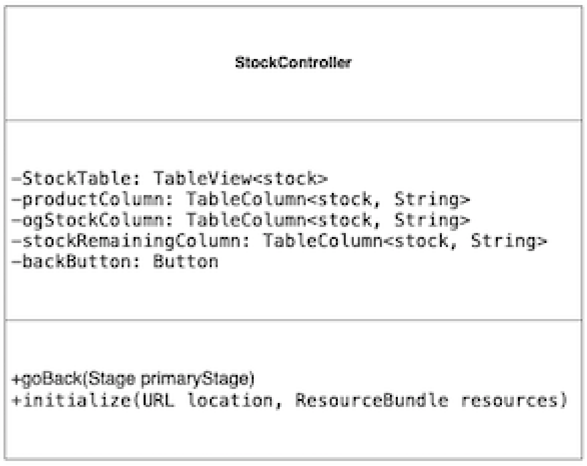
```
OrdersController

+oldValue: String
+x: int
-backButton: Button
-addProductButton: Button
-productChooser: ChoiceBox<String>
-clientNameField: TextField
-dateOfSaleField: TextField
-priceField: TextField
-quantityField: TextField
-addOrderButton: Button
-specialPrice: CheckBox
-deleteButton: Button
-editButton: Button
-saveButton: Button
-updateButton: Button
-OrdersTable: TableView<Orders>
-productColumn: TableColumn<Orders, String>
-customerColumn: TableColumn<Orders, String>
-dateColumn: TableColumn<Orders, String>
-priceColumn: TableColumn<Orders, Integer>
-quantityColumn: TableColumn<Orders, Integer>

+ActivateSpecialPrice(ActionEvent event)
+AddingOrder(ActionEvent event)
+GoToAddProduct(ActionEvent event)
+GoBack(ActionEvent event)
+EditingProduct(ActionEvent event)
+DeletingProduct(ActionEvent event)
+SavingOrders(ActionEvent event)
+Updating(ActionEvent event)
+initialize(URL location, ResourceBundle resources)
```



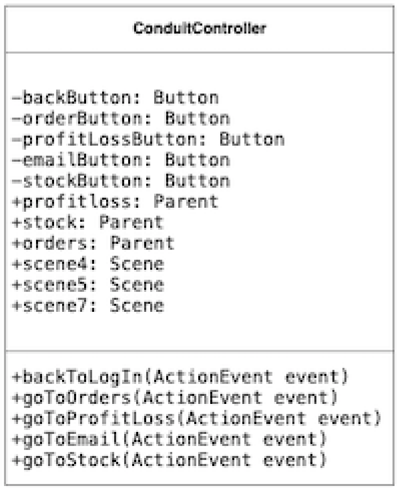
```
EmailController

-Header: TextField
-ToWhom: TextField
-Body: TextArea
-sendButton: Button
-goBack: Button

+sendEmail(ActionEvent event)
+goBack(ActionEvent event)
```



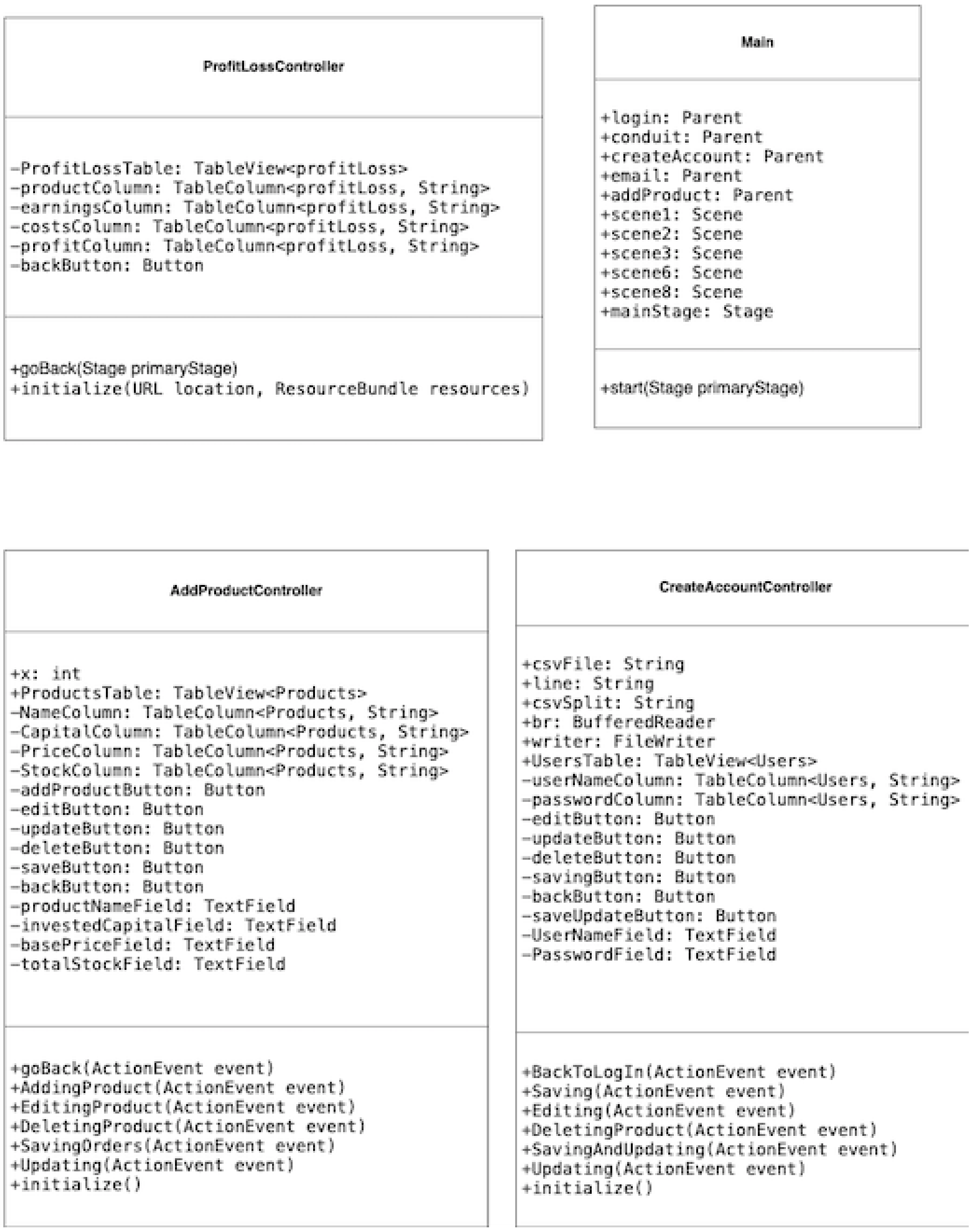
```
LoginController

-Username: TextField
-LoginButton: Button
-Password: PasswordField
-CreateAccountButton: Button

+CreatingAccount(ActionEvent event)
+LoginPress(ActionEvent event)
+scene()
```



```
StockController

-StockTable: TableView<stock>
-productColumn: TableColumn<stock, String>
-ogStockColumn: TableColumn<stock, String>
-stockRemainingColumn: TableColumn<stock, String>
-backButton: Button

+goBack(Stage primaryStage)
+initialize(URL location, ResourceBundle resources)
```



```
ConduitController

-backButton: Button
-orderButton: Button
-profitLossButton: Button
-emailButton: Button
-stockButton: Button
+profitloss: Parent
+stock: Parent
+orders: Parent
+scene4: Scene
+scene5: Scene
+scene7: Scene

+backToLogIn(ActionEvent event)
+goToOrders(ActionEvent event)
+goToProfitLoss(ActionEvent event)
+goToEmail(ActionEvent event)
+goToStock(ActionEvent event)
```

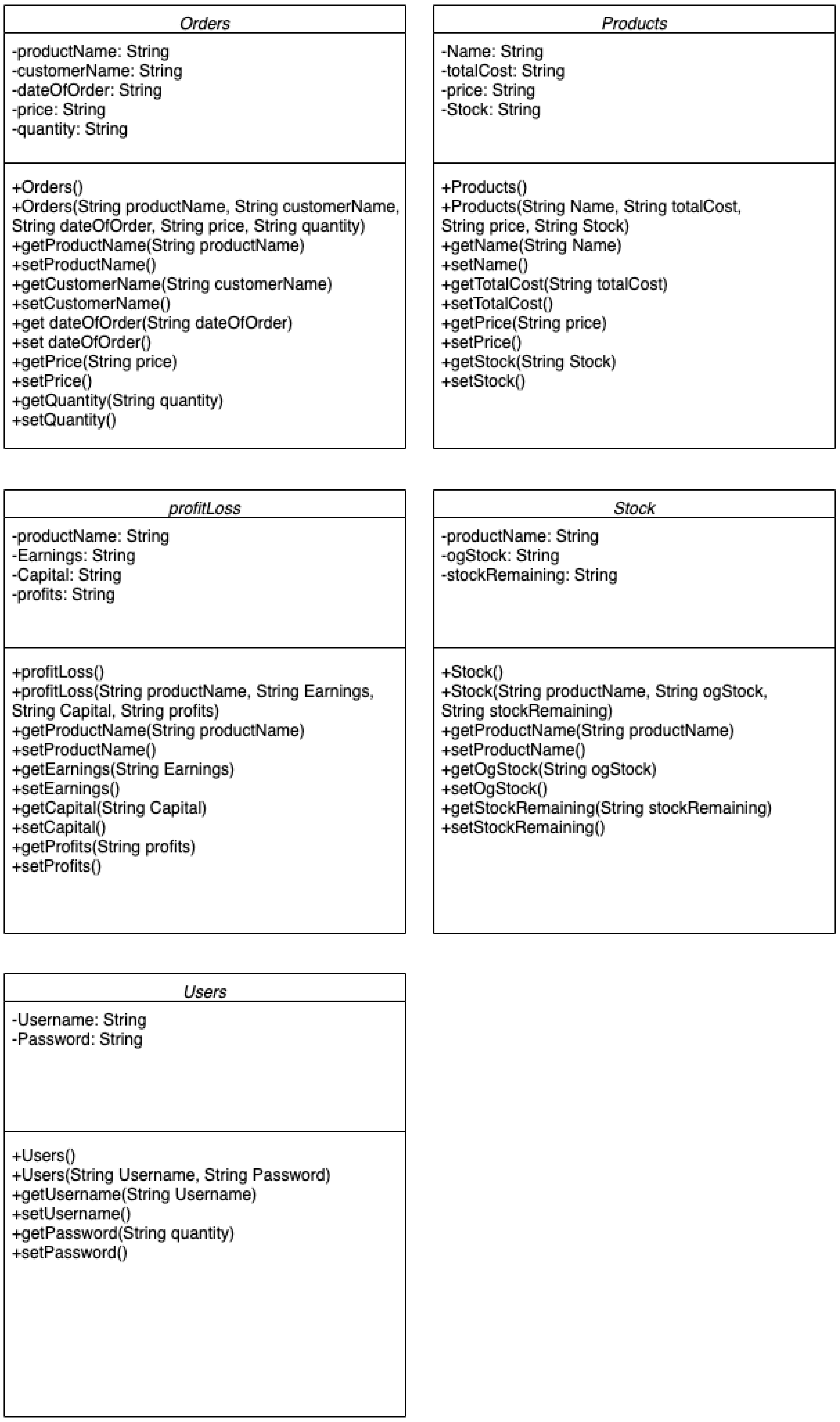


```
ProfitLossController
-------------------------------------------------
-ProfitLossTable: TableView<profitLoss>
-productColumn: TableColumn<profitLoss, String>
-earningsColumn: TableColumn<profitLoss, String>
-costsColumn: TableColumn<profitLoss, String>
-profitColumn: TableColumn<profitLoss, String>
-backButton: Button
-------------------------------------------------
+goBack(Stage primaryStage)
+initialize(URL location, ResourceBundle resources)
```

```
Main
-------------------------------------------------
+login: Parent
+conduit: Parent
+createAccount: Parent
+email: Parent
+addProduct: Parent
+scene1: Scene
+scene2: Scene
+scene3: Scene
+scene6: Scene
+scene8: Scene
+mainStage: Stage
-------------------------------------------------
+start(Stage primaryStage)
```

```
AddProductController
-------------------------------------------------
+x: int
+ProductsTable: TableView<Products>
-NameColumn: TableColumn<Products, String>
-CapitalColumn: TableColumn<Products, String>
-PriceColumn: TableColumn<Products, String>
-StockColumn: TableColumn<Products, String>
-addProductButton: Button
-editButton: Button
-updateButton: Button
-deleteButton: Button
-saveButton: Button
-backButton: Button
-productNameField: TextField
-investedCapitalField: TextField
-basePriceField: TextField
-totalStockField: TextField
-------------------------------------------------
+goBack(ActionEvent event)
+AddingProduct(ActionEvent event)
+EditingProduct(ActionEvent event)
+DeletingProduct(ActionEvent event)
+SavingOrders(ActionEvent event)
+Updating(ActionEvent event)
+initialize()
```

```
CreateAccountController
-------------------------------------------------
+csvFile: String
+line: String
+csvSplit: String
+br: BufferedReader
+writer: FileWriter
+UsersTable: TableView<Users>
-userNameColumn: TableColumn<Users, String>
-passwordColumn: TableColumn<Users, String>
-editButton: Button
-updateButton: Button
-deleteButton: Button
-savingButton: Button
-backButton: Button
-saveUpdateButton: Button
-UserNameField: TextField
-PasswordField: TextField
-------------------------------------------------
+BackToLogIn(ActionEvent event)
+Saving(ActionEvent event)
+Editing(ActionEvent event)
+DeletingProduct(ActionEvent event)
+SavingAndUpdating(ActionEvent event)
+Updating(ActionEvent event)
+initialize()
```


Orders
-productName: String
-customerName: String
-dateOfOrder: String
-price: String
-quantity: String
+Orders()
+Orders(String productName, String customerName, String dateOfOrder, String price, String quantity)
+getProductName(String productName)
+setProductName()
+getCustomerName(String customerName)
+setCustomerName()
+get dateOfOrder(String dateOfOrder)
+set dateOfOrder()
+getPrice(String price)
+setPrice()
+getQuantity(String quantity)
+setQuantity()
Products
-Name: String
-totalCost: String
-price: String
-Stock: String
+Products()
+Products(String Name, String totalCost, String price, String Stock)
+getName(String Name)
+setName()
+getTotalCost(String totalCost)
+setTotalCost()
+getPrice(String price)
+setPrice()
+getStock(String Stock)
+setStock()
profitLoss
-productName: String
-Earnings: String
-Capital: String
-profits: String
+profitLoss()
+profitLoss(String productName, String Earnings, String Capital, String profits)
+getProductName(String productName)
+setProductName()
+getEarnings(String Earnings)
+setEarnings()
+getCapital(String Capital)
+setCapital()
+getProfits(String profits)
+setProfits()
Stock
-productName: String
-ogStock: String
-stockRemaining: String
+Stock()
+Stock(String productName, String ogStock, String stockRemaining)
+getProductName(String productName)
+setProductName()
+getOgStock(String ogStock)
+setOgStock()
+getStockRemaining(String stockRemaining)
+setStockRemaining()
Users
-Username: String
-Password: String
+Users()
+Users(String Username, String Password)
+getUsername(String Username)
+setUsername()
+getPassword(String quantity)
+setPassword()

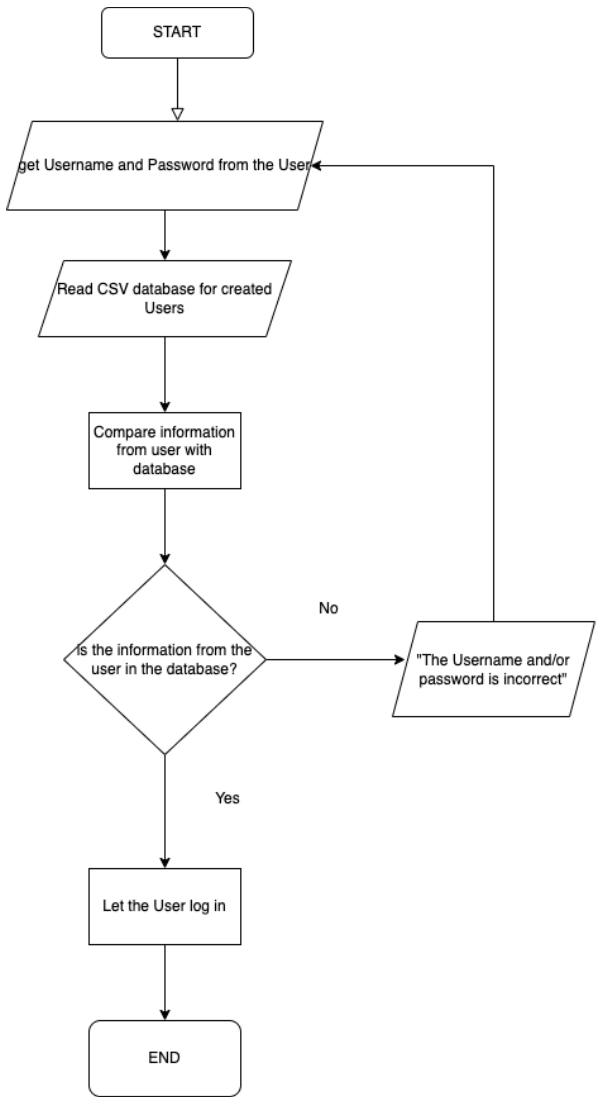

START
get Username and Password from the User
Read CSV database for created Users
Compare information from user with database
Is the information from the user in the database?
No
"The Username and/or password is incorrect"
Yes
Let the User log in
END

Figure 1 - Logging In Functionality

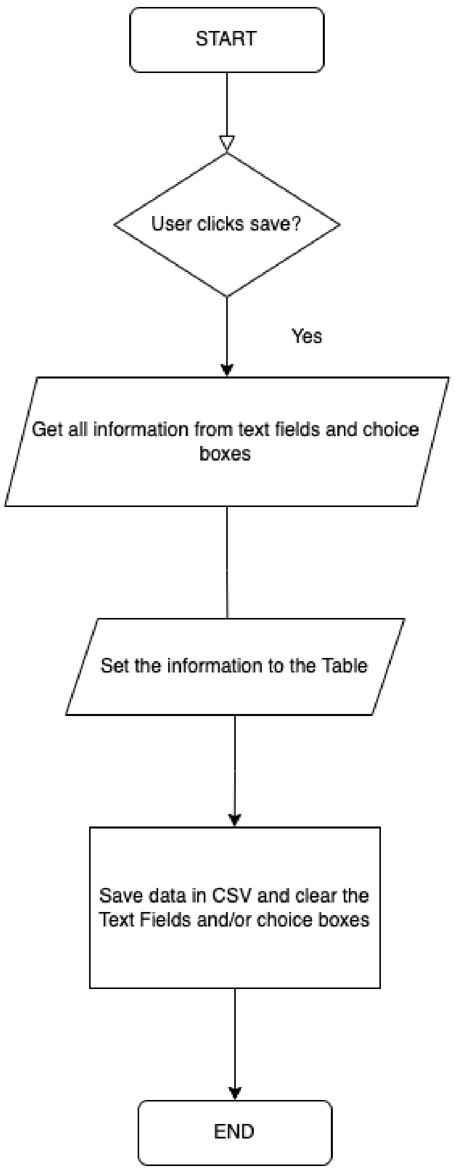

START
User clicks save?
Yes
Get all information from text fields and choice boxes
Set the information to the Table
Save data in CSV and clear the Text Fields and/or choice boxes
END

Figure 2 - Saving data to tables and CSV

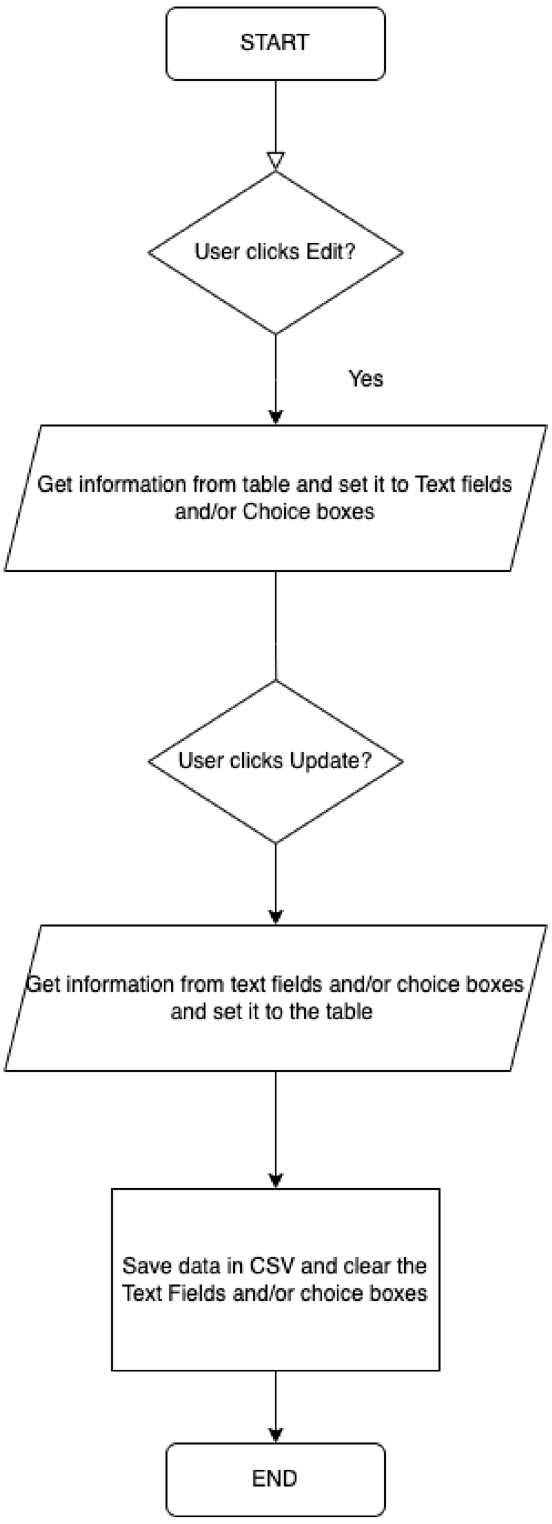

START
User clicks Edit?
Yes
Get information from table and set it to Text fields and/or Choice boxes
User clicks Update?
Get information from text fields and/or choice boxes and set it to the table
Save data in CSV and clear the Text Fields and/or choice boxes
END

Figure 3 - Editing and Updating data into Table and CSV

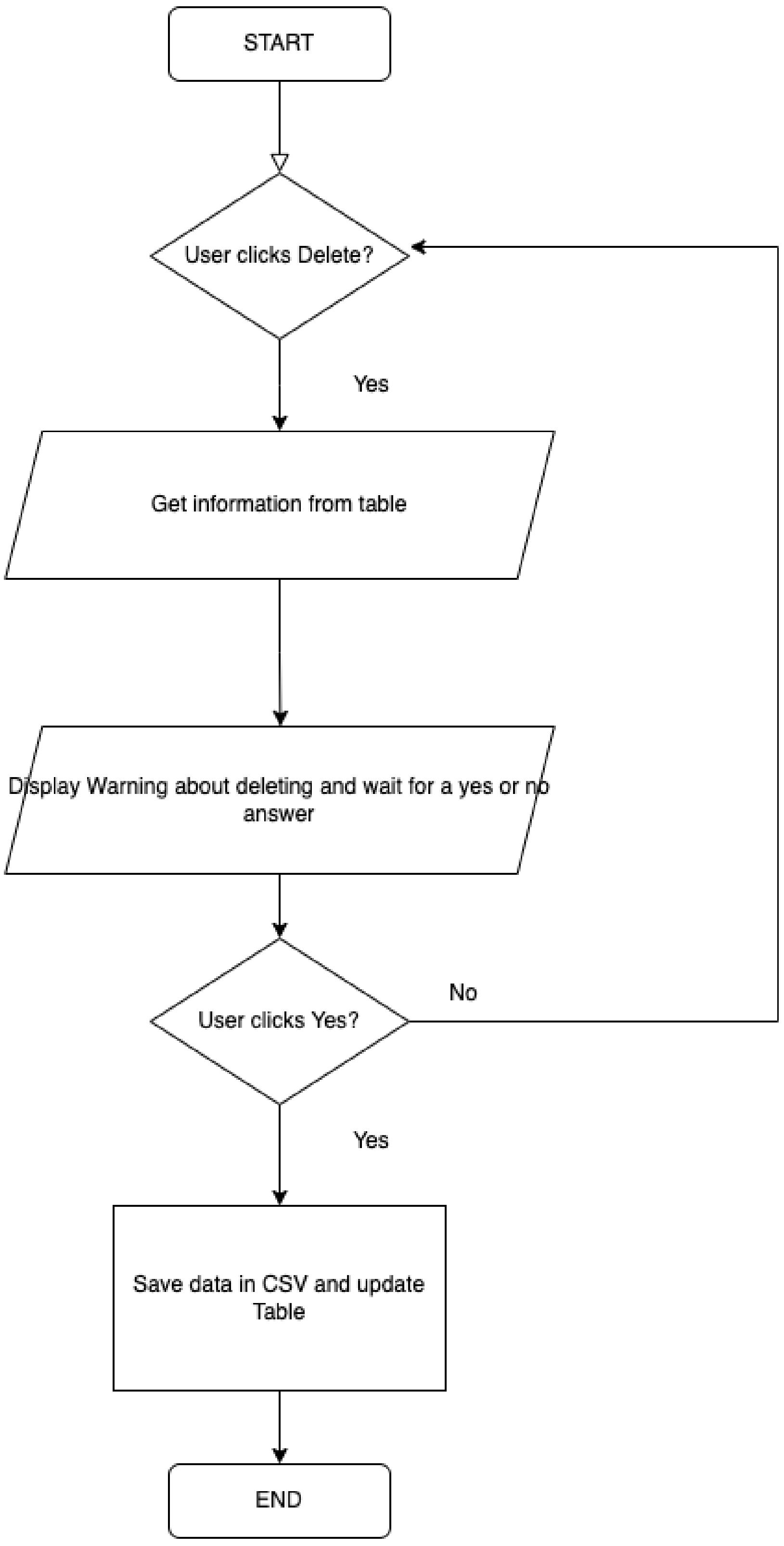

START
User clicks Delete?
Yes
Get information from table
Display Warning about deleting and wait for a yes or no answer
User clicks Yes?
No
Yes
Save data in CSV and update Table
END

Figure 4 - Deleting Data from table and CSV

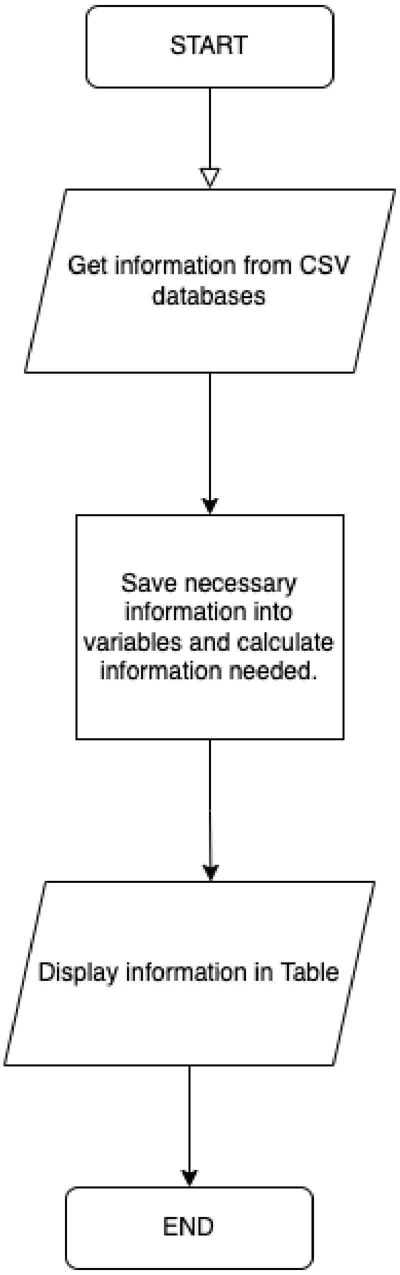

START
Get information from CSV databases
Save necessary information into variables and calculate information needed.
Display information in Table
END

Figure 5 - Initialize ProfitLoss and Stock screens

Databases and Files created before writing program:

A database called “Files” with csv files inside was created inside the project where all the classes are located with the following.

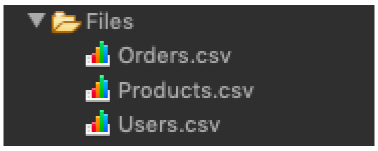


Figure 6 - Files folder

Design of Panels(Graphic User Interface):

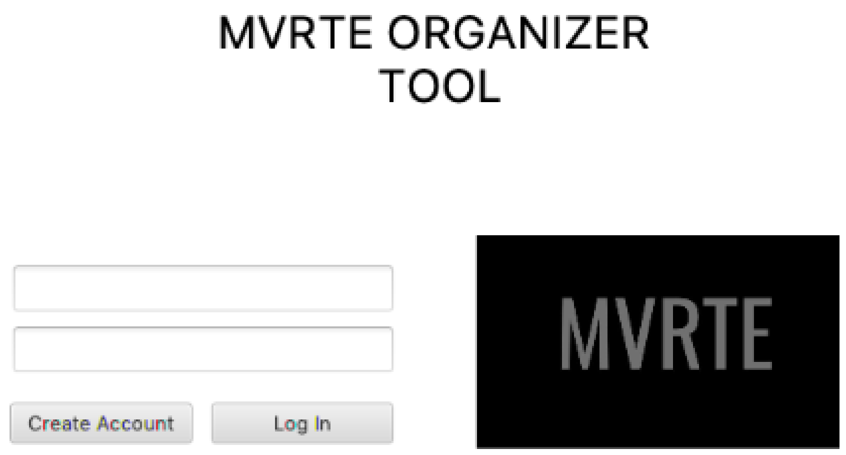


Figure 7 - Login Screen

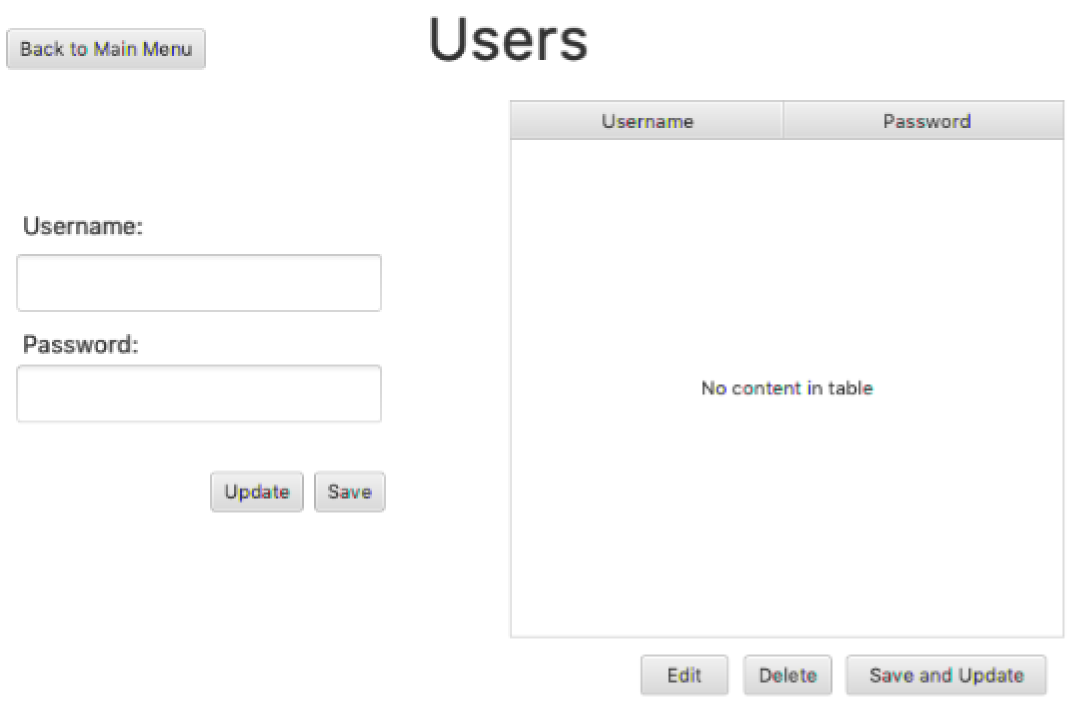


Figure 8 - Create Users Screen

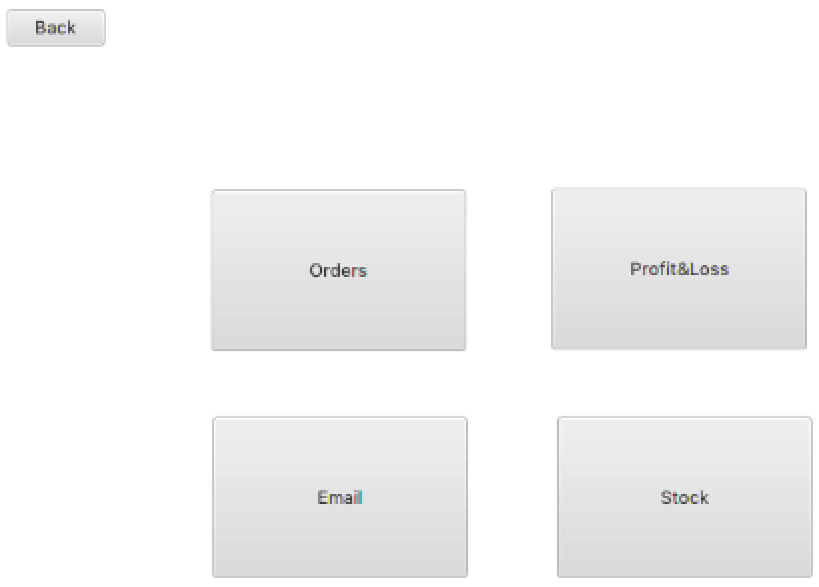
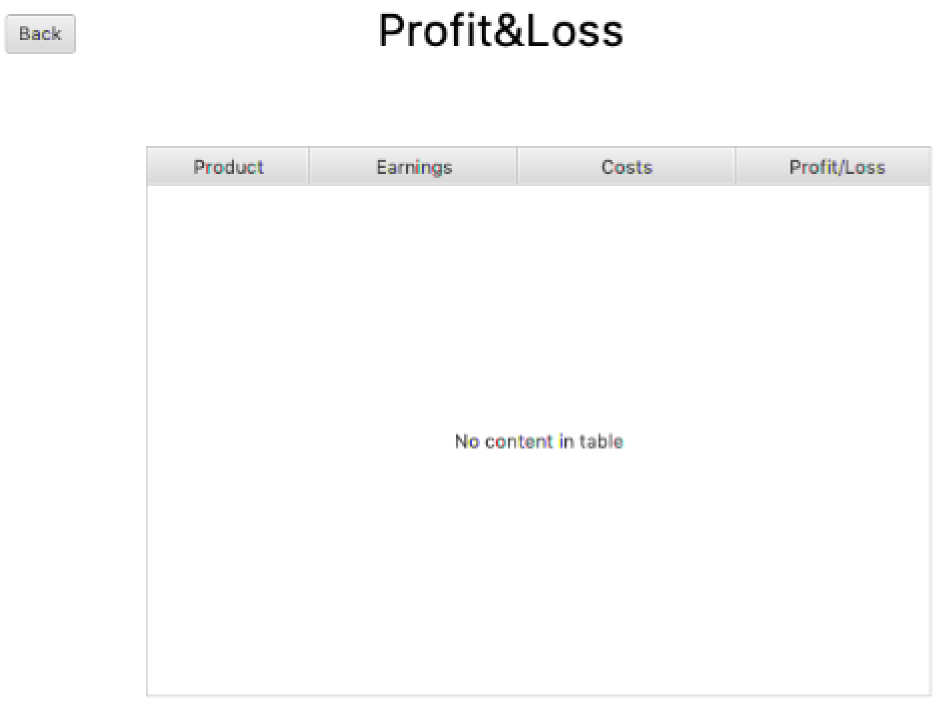


Figure 9 - Conduit Screen



Product	Earnings	Costs	Profit/Loss

No content in table

Figure 10 - Profit and Loss Screen

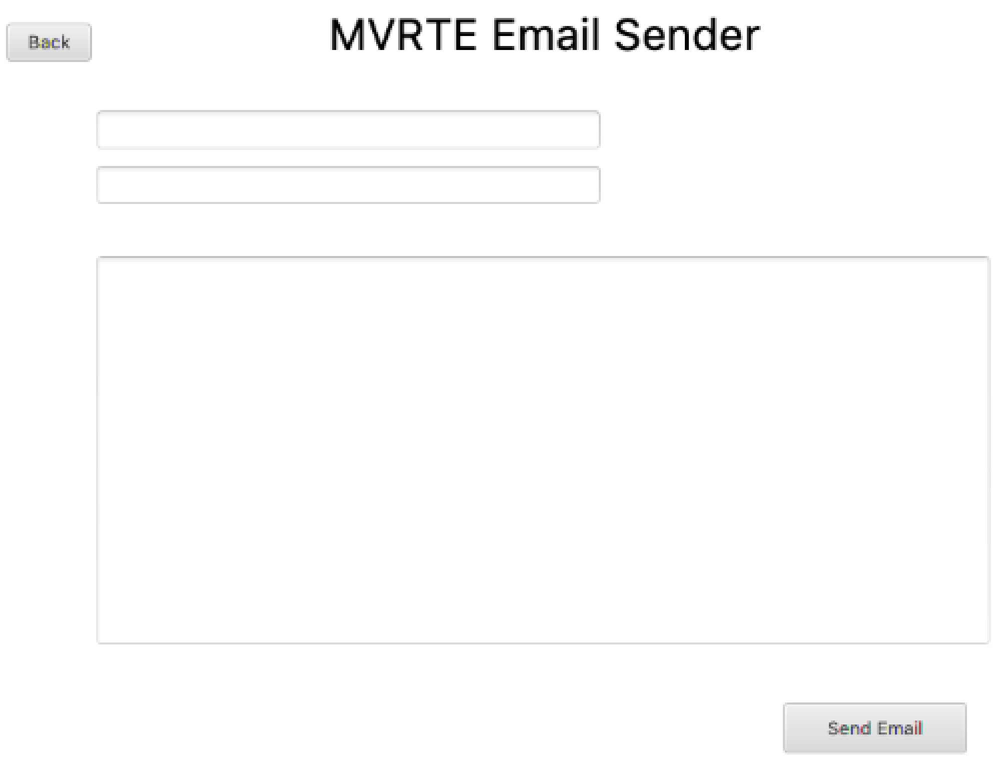


Figure 11 - MVRTE email sender Screen

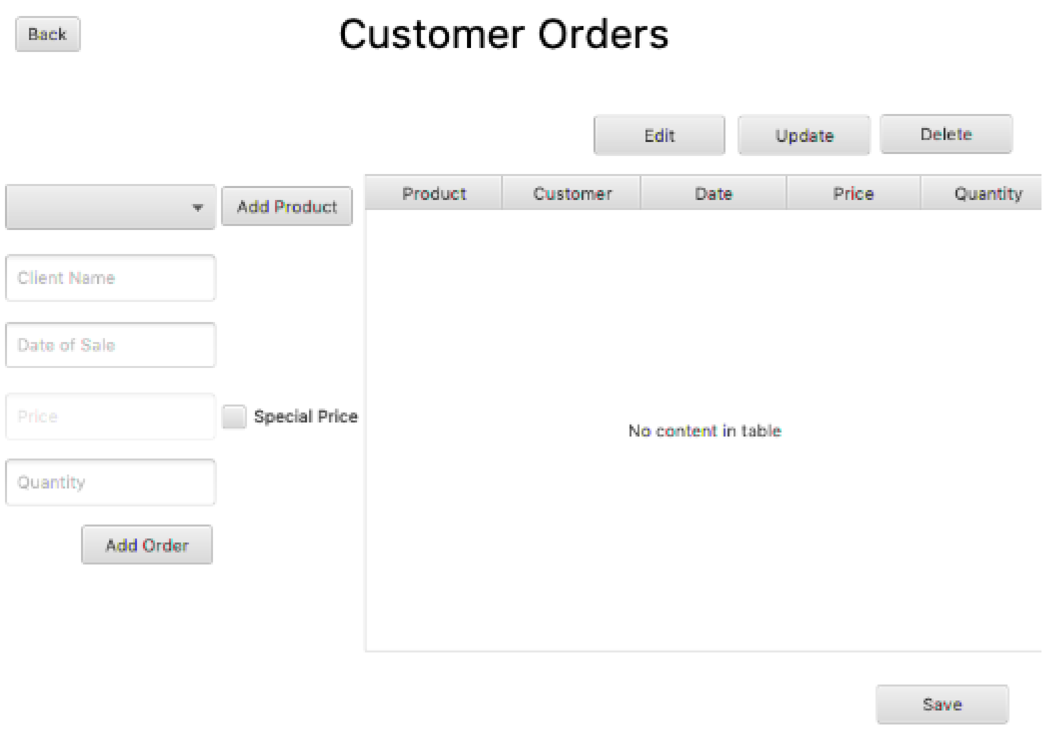
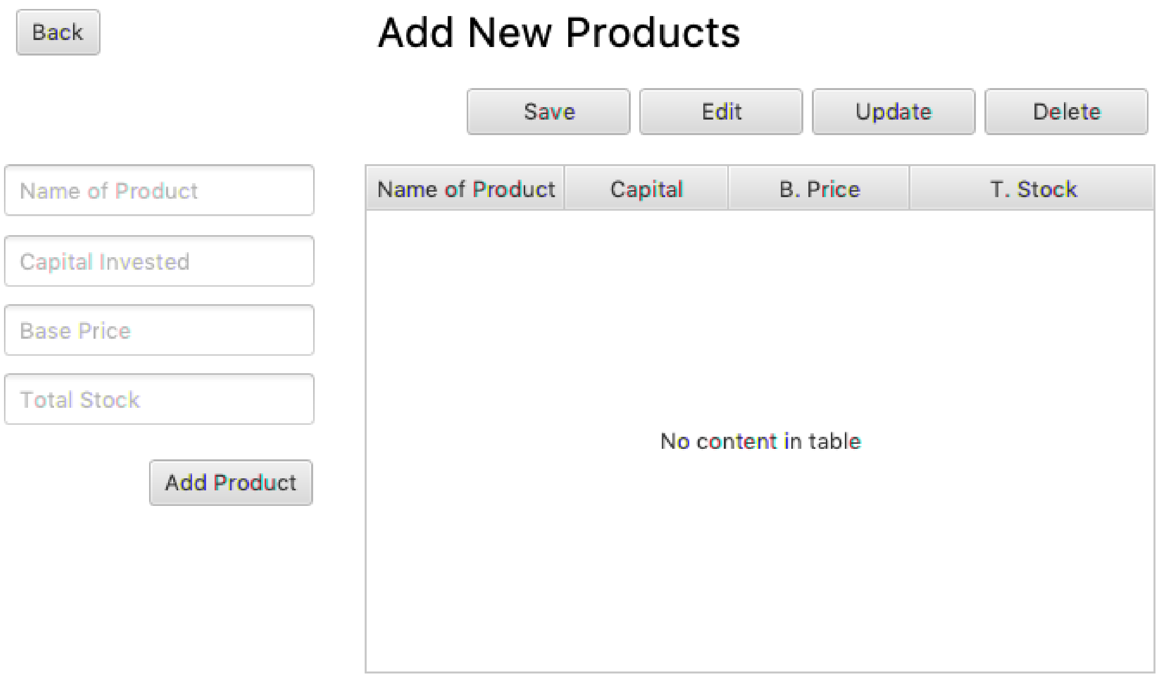


Figure 12 - Customer Orders Screen



Name of Product	Capital	B. Price	T. Stock

No content in table

Figure 13 - Add Products Screen

Test Plan:

Test Type	Nature of Test	Example
When starting the program a menu to put your username and password appear	Check if the program validates if you are a registered user in the program	User enters Michael and 1234 as username and password. The program validates it as a correct username and password and goes to next screen “Username: Michael Password: 1234 ”
In the Login screen there are buttons that allow editing, creating, deleting and saving Users.	Checks if the functionality of the buttons work	User clicks any of the buttons. Depending on the button he can create, delete, edit or save a user to the program
There is a screen with the information for Profit and Loss	Check if a screen exists that displays info for profit and loss with accuracy.	User clicks on the button to open the profit and loss screen and he is greeted with the information it's supposed to display.
There is a screen with the information of Orders, and the ability to edit, create or delete orders	Check if a screen exists that displays info for Orders	Click button to arrive to Orders screen and verify functionality and existence of such a screen
In the Orders screen there are buttons that allow editing, creating, deleting and saving Users	Checks if the functionality of the three buttons work	User clicks any of the buttons. Depending on the button he can create, delete or edit an Order.
There is a screen for sending emails	Checks if the screen can successfully send an email	user inputs text in all fields necessary and clicks the send button The send button sends the text in an email.
There is a screen with the information of the stock/inventory	Checks if there is a screen for stock/inventory	User clicks on the button to open the Stock/Inventory screen and he is greeted with the information it's supposed to display.

Word Count: 319