Even Semester (2019)

BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		Surname	Given Names	Student ID Number
	1.	Alifio	Rasyid	2201798295

Course Code	**: COMP6510**	**Course Name**	**: Programming Language**
Class	**: L2AC**	**Name of Lecturer(s)**	: 1. **Jude Martinez** **2. Minaldi Loeis**
Major	**: CS**		
Title of Assignment (if any)	**: Report**		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 02 - 07 - 2019**	**Submission Date**	**: 02 – 07 -2019**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)
Alifio Rasendriya Rasyid

Contents

1. Project Specifications

A lot of university students such as myself still rely heavily on monthly allowance given by our parents. As such, we need to be wise upon spending our money. Some of us would take notes on expenses manually and some would take notes on an excel file on their computer (I'm one of them), others would use money managing app on their mobile phones.

While there are already many great financial managing app on the market right now, I find that a lot of them are simply too complicated to use on a daily basis and are more oriented for somebody who are working already, hence the reason why me and some others choose to track our finance manually. Tracking finances manually however isn't very time efficient.

For this reason, as my final project, I have made a financial tracker program. The program would store each user's daily financial expenses and help calculate their daily and monthly expenses.

2. Solution Design

2.1. Program Structure Description

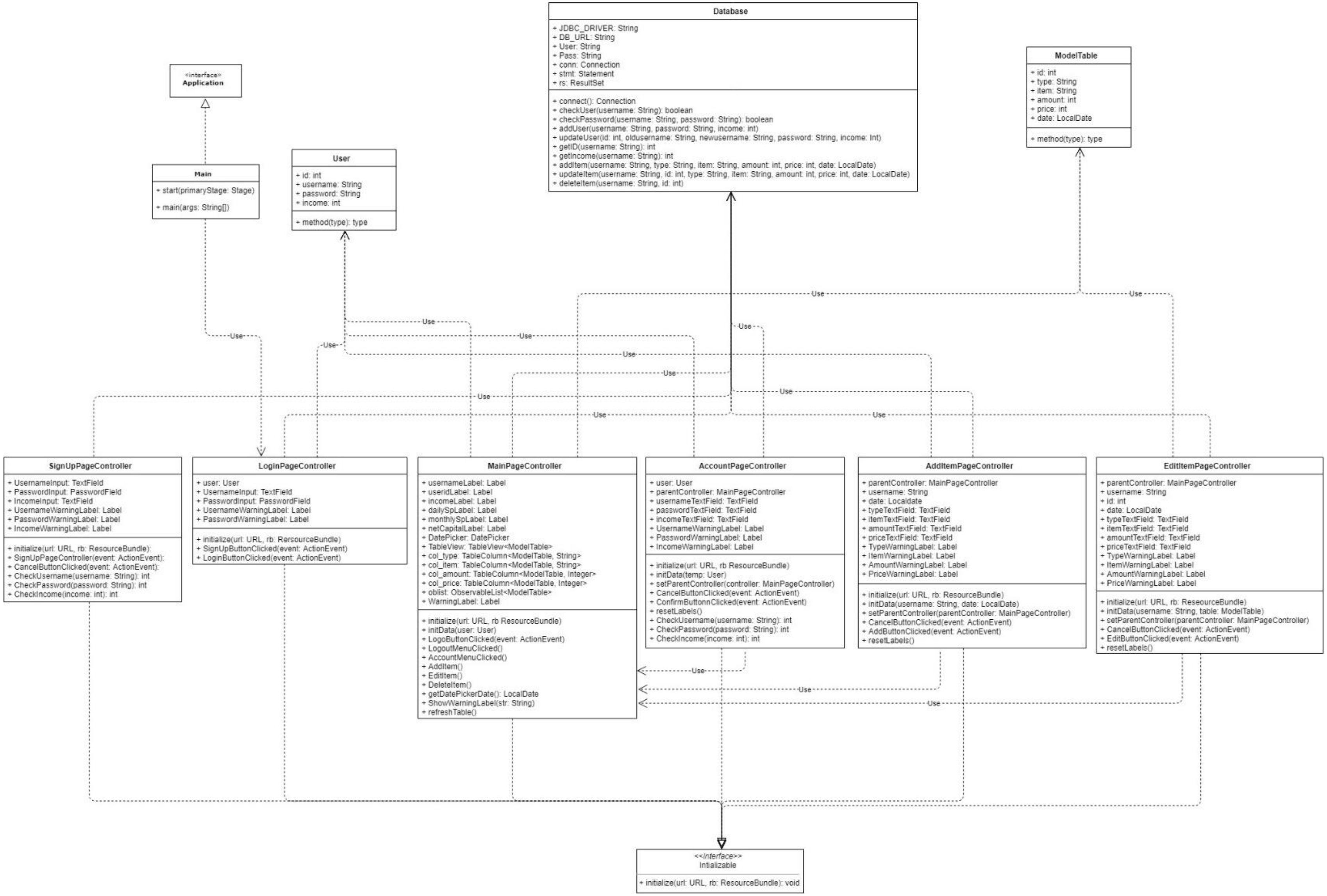
All of the database functions are stored in the `Database` class and all of the functions' accessibility there are also set to `public static` so that it can be accessed from any other classes.

All of the FXML controller classes implements `Initializable` interface. This forces them to override the `initialize()` method; in short this method will be called when the FXML controllers are initialized, much like a constructor in a normal class.

There are also two classes to store the user's account data and user's expenses data, they're called `User` and `ModelTable` respectively. `User` and `ModelTable` objects are especially needed so that the program can passes user's data in between different classes (controllers).

`AccountPage`, `AddItemPage` and `EditItemPage` controllers all stores a `MainPageController` object inside them. This enables the three classes to control the Main Page from outside of its controller, in this case it's trying to refresh the table inside the Main Page.

2.2. UML Diagram


«interface»
Application
Main
+ start(primaryStage: Stage)
+ main(args: String[])
User
+ id: int
+ username: String
+ password: String
+ income: int
+ method(type): type
Database
+ JDBC_DRIVER: String
+ DB_URL: String
+ User: String
+ Pass: String
+ conn: Connection
+ stmt: Statement
+ rs: ResultSet
+ connect(): Connection
+ checkUser(username: String): boolean
+ checkPassword(username: String, password: String): boolean
+ addUser(username: String, password: String, income: int)
+ updateUser(id: int, oldusername: String, newusername: String, password: String, income: Int)
+ getID(username: String): int
+ getIncome(username: String): int
+ addItem(username: String, type: String, item: String, amount: int, price: int, date: LocalDate)
+ updateItem(username: String, id: int, type: String, item: String, amount: int, price: int, date: LocalDate)
+ deleteItem(username: String, id: int)
ModelTable
+ id: int
+ type: String
+ item: String
+ amount: int
+ price: int
+ date: LocalDate
+ method(type): type
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
SignUpPageController
+ UsernameInput: TextField
+ PasswordInput: PasswordField
+ IncomeInput: TextField
+ UsernameWarningLabel: Label
+ PasswordWarningLabel: Label
+ IncomeWarningLabel: Label
+ initialize(url: URL, rb: ResourceBundle):
+ SignUpPageController(event: ActionEvent):
+ CancelButtonClicked(event: ActionEvent):
+ CheckUsername(username: String): int
+ CheckPassword(password: String): int
+ CheckIncome(income: int): int
LoginPageController
+ user: User
+ UsernameInput: TextField
+ PasswordInput: PasswordField
+ UsernameWarningLabel: Label
+ PasswordWarningLabel: Label
+ initialize(url: URL, rb: RersourceBundle)
+ SignUpButtonClicked(event: ActionEvent)
+ LoginButtonClicked(event: ActionEvent)
MainPageController
+ usernameLabel: Label
+ useridLabel: Label
+ incomeLabel: Label
+ dailySpLabel: Label
+ monthlySpLabel: Label
+ netCapitalLabel: Label
+ DatePicker: DatePicker
+ TableView: TableView<ModelTable>
+ col_type: TableColumn<ModelTable, String>
+ col_item: TableColumn<ModelTable, String>
+ col_amount: TableColumn<ModelTable, Integer>
+ col_price: TableColumn<ModelTable, Integer>
+ oblist: ObservableList<ModelTable>
+ WarningLabel: Label
+ initialize(url: URL, rb ResourceBundle)
+ initData(user: User)
+ LogoButtonClicked(event: ActionEvent)
+ LogoutMenuClicked()
+ AccountMenuClicked()
+ AddItem()
+ EditItem()
+ DeleteItem()
+ getDatePickerDate(): LocalDate
+ ShowWarningLabel(str: String)
+ refreshTable()
AccountPageController
+ user: User
+ parentController: MainPageController
+ usernameTextField: TextField
+ passwordTextField: TextField
+ incomeTextField: TextField
+ UsernameWarningLabel: Label
+ PasswordWarningLabel: Label
+ IncomeWarningLabel: Label
+ initialize(url: URL, rb ResourceBundle)
+ initData(temp: User)
+ setParentController(controller: MainPageController)
+ CancelButtonClicked(event: ActionEvent)
+ ConfirmButtonnClicked(event: ActionEvent)
+ resetLabels()
+ CheckUsername(username: String): int
+ CheckPassword(password: String): int
+ CheckIncome(income: int): int
AddItemPageController
+ parentController: MainPageController
+ username: String
+ date: Localdate
+ typeTextField: TextField
+ itemTextField: TextField
+ amountTextField: TextField
+ priceTextField: TextField
+ TypeWarningLabel: Label
+ ItemWarningLabel: Label
+ AmountWarningLabel: Label
+ PriceWarningLabel: Label
+ initialize(url: URL, rb: ResourceBundle)
+ initData(username: String, date: LocalDate)
+ setParentController(parentController: MainPageController)
+ CancelButtonClicked(event: ActionEvent)
+ AddButtonClicked(event: ActionEvent)
+ resetLabels()
EditItemPageController
+ parentController: MainPageController
+ username: String
+ id: int
+ date: LocalDate
+ typeTextField: TextField
+ itemTextField: TextField
+ amountTextField: TextField
+ priceTextField: TextField
+ TypeWarningLabel: Label
+ ItemWarningLabel: Label
+ AmountWarningLabel: Label
+ PriceWarningLabel: Label
+ initialize(url: URL, rb: ReseourceBundle)
+ initData(username: String, table: ModelTable)
+ setParentController(parentController: MainPageController)
+ CancelButtonClicked(event: ActionEvent)
+ EditButtonClicked(event: ActionEvent)
+ resetLabels()
Use
Use
Use
<<interface>>
Intializable
+ initialize(url: URL, rb: ResourceBundle): void

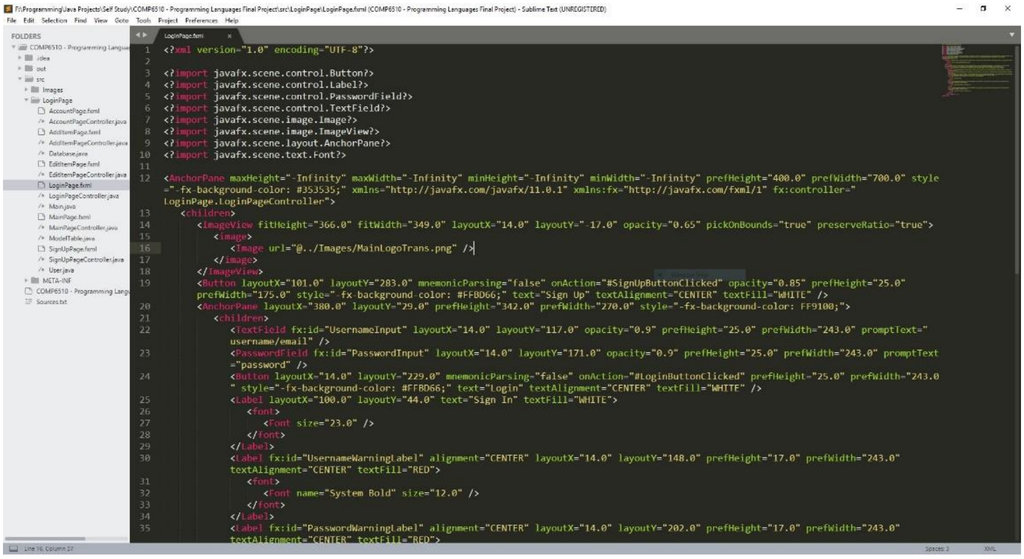
3. Code Implementation Discussion

3.1. Implementation

- **javafx**, library used to develop desktop application as well as Rich Internet Application (RIA). It is intended to replace swing in Java app as a GUI framework.
- **java.sql**, a package that contains all the JDBC API; enables java to send SQL statements to the database and retrieves the results.
- **java.time**, API for date and time.

3.2. How it Works

This program is a Java GUI based application, the GUI itself is entirely made using the JavaFX library and Scene Builder. There's two main components when using the JavaFX library; FXML files and a controller class to control the FXML files. The FXML files are responsible for the GUI of the program (Scene Builder helps controlling the FXML files, so designs are not hard coded) whereas the controller class would contain backend codes; for example, what a button does and when will a table be refresh.


F:\Programming\Java Projects\Self Study\COMP6510 - Programming Languages Final Project\src\LoginPage\LoginPage.fxml (COMP6510 - Programming Languages Final Project) - Sublime Text (UNREGISTERED)
File
Edit
Selection
Find
View
Goto
Tools
Project
Preferences
Help
FOLDERS
COMP6510 - Programming Langua
.idea
out
src
Images
LoginPage
AccountPage.fxml
AccountPageController.java
AddItemPage.fxml
AddItemPageController.java
Database.java
EditItemPage.fxml
EditItemPageController.java
LoginPage.fxml
LoginPageController.java
Main.java
MainPage.fxml
MainPageController.java
ModelTable.java
SignUpPage.fxml
SignUpPageController.java
User.java
META-INF
COMP6510 - Programming Langu
Sources.txt
LoginPage.fxml
Line 16, Column 37
Spaces: 3
XML

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="700.0" style="-fx-background-color: #353535;" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="LoginPage.LoginPageController">
   <children>
      <ImageView fitHeight="366.0" fitWidth="349.0" layoutX="14.0" layoutY="-17.0" opacity="0.65" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@../Images/MainLogoTrans.png" />
         </image>
      </ImageView>
      <Button layoutX="101.0" layoutY="283.0" mnemonicParsing="false" onAction="#SignUpButtonClicked" opacity="0.85" prefHeight="25.0" prefWidth="175.0" style="-fx-background-color: #FFBD66;" text="Sign Up" textAlignment="CENTER" textFill="WHITE" />
      <AnchorPane layoutX="380.0" layoutY="29.0" prefHeight="342.0" prefWidth="270.0" style="-fx-background-color: FF9100;">
         <children>
            <TextField fx:id="UsernameInput" layoutX="14.0" layoutY="117.0" opacity="0.9" prefHeight="25.0" prefWidth="243.0" promptText="username/email" />
            <PasswordField fx:id="PasswordInput" layoutX="14.0" layoutY="171.0" opacity="0.9" prefHeight="25.0" prefWidth="243.0" promptText="password" />
            <Button layoutX="14.0" layoutY="229.0" mnemonicParsing="false" onAction="#LoginButtonClicked" prefHeight="25.0" prefWidth="243.0" style="-fx-background-color: #FFBD66;" text="Login" textAlignment="CENTER" textFill="WHITE" />
            <Label layoutX="100.0" layoutY="44.0" text="Sign In" textFill="WHITE">
               <font>
                  <Font size="23.0" />
               </font>
            </Label>
            <Label fx:id="UsernameWarningLabel" alignment="CENTER" layoutX="14.0" layoutY="148.0" prefHeight="17.0" prefWidth="243.0" textAlignment="CENTER" textFill="RED">
               <font>
                  <Font name="System Bold" size="12.0" />
               </font>
            </Label>
            <Label fx:id="PasswordWarningLabel" alignment="CENTER" layoutX="14.0" layoutY="202.0" prefHeight="17.0" prefWidth="243.0" textAlignment="CENTER" textFill="RED">
```

Image 3.2.1 LoginPage.fxml

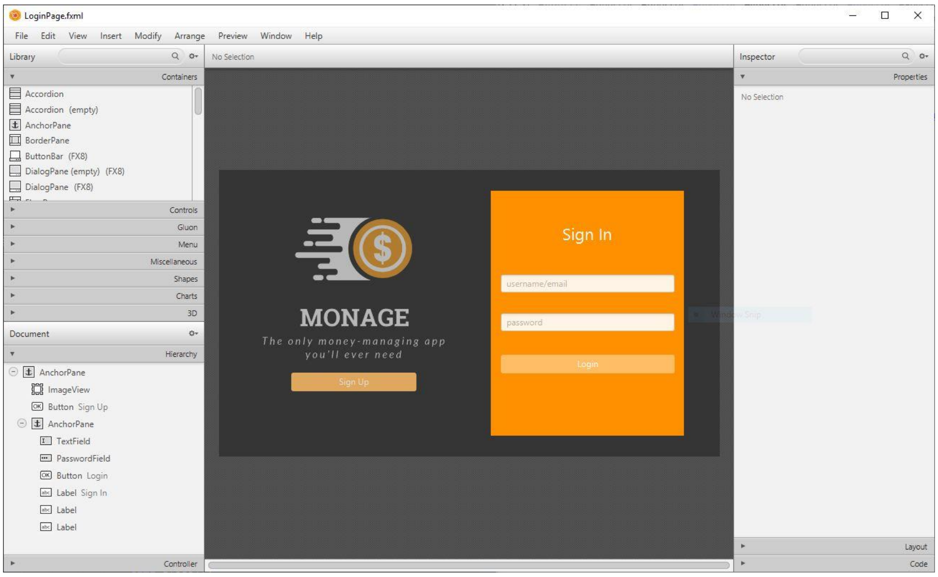
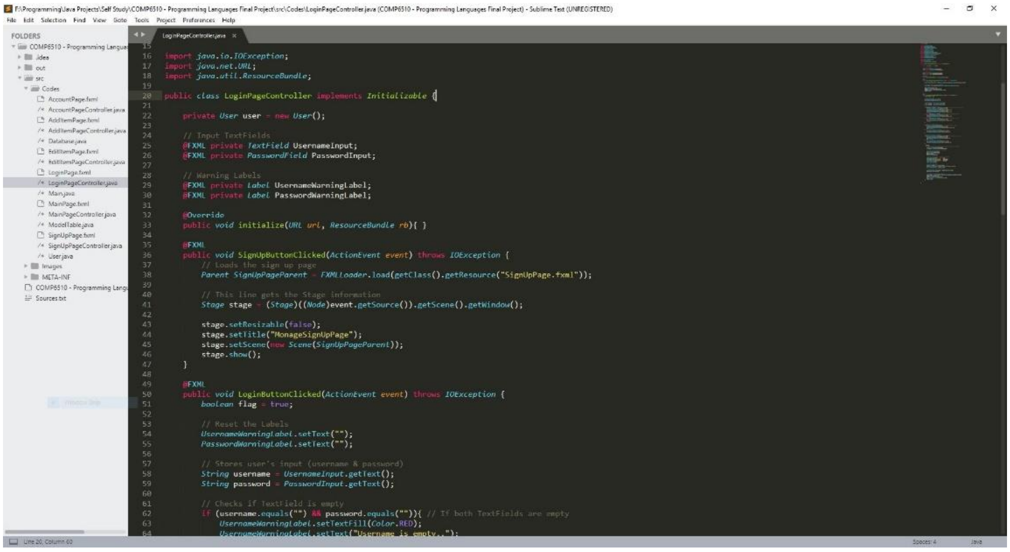


Image 3.2.2 Scene Builder


F:\Programming\Java Projects\Self Study\COMP6510 - Programming Languages Final Project\src\Codes\LoginPageController.java (COMP6510 - Programming Languages Final Project) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
COMP6510 - Programming Langua
.idea
out
src
Codes
AccountPage.fxml
AccountPageController.java
AddItemPage.fxml
AddItemPageController.java
Database.java
EditItemPage.fxml
EditItemPageController.java
LoginPage.fxml
LoginPageController.java
Main.java
MainPage.fxml
MainPageController.java
ModelTable.java
SignUpPage.fxml
SignUpPageController.java
User.java
Images
META-INF
COMP6510 - Programming Langu
Sources.txt
LoginPageController.java
Line 20, Column 60
Spaces: 4
Java

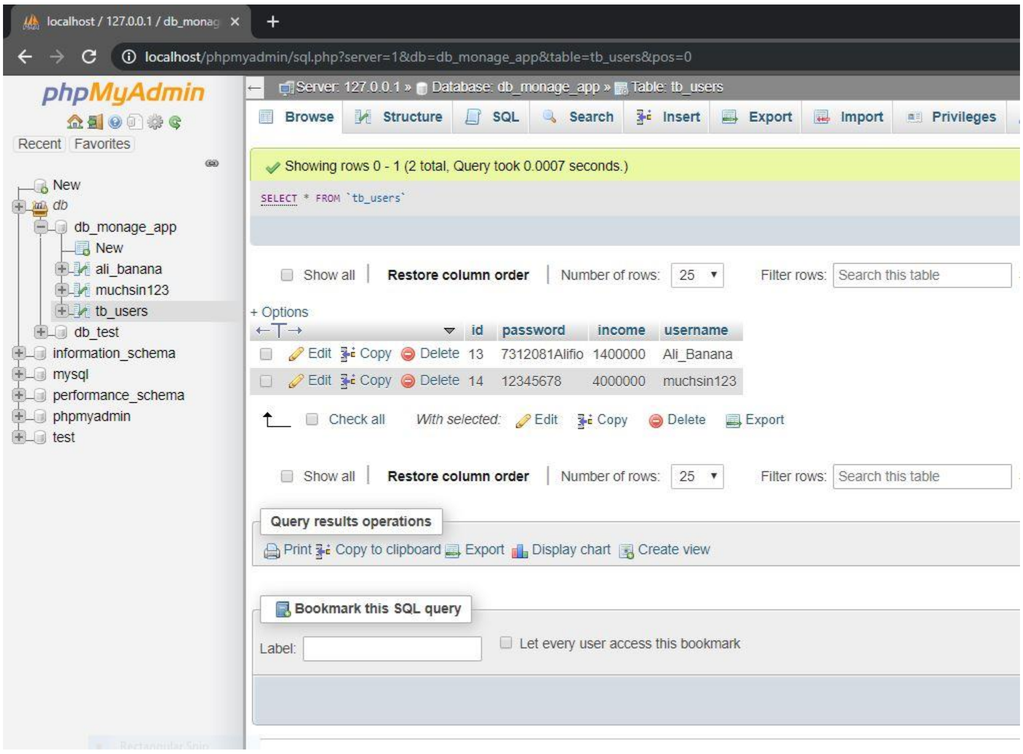
```java

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class LoginPageController implements Initializable {

    private User user = new User();

    // Input TextFields
    @FXML private TextField UsernameInput;
    @FXML private PasswordField PasswordInput;

    // Warning Labels
    @FXML private Label UsernameWarningLabel;
    @FXML private Label PasswordWarningLabel;

    @Override
    public void initialize(URL url, ResourceBundle rb){ }

    @FXML
    public void SignUpButtonClicked(ActionEvent event) throws IOException {
        // Loads the sign up page
        Parent SignUpPageParent = FXMLLoader.load(getClass().getResource("SignUpPage.fxml"));

        // This line gets the Stage information
        Stage stage = (Stage)((Node)event.getSource()).getScene().getWindow();

        stage.setResizable(false);
        stage.setTitle("MonageSignUpPage");
        stage.setScene(new Scene(SignUpPageParent));
        stage.show();
    }

    @FXML
    public void LoginButtonClicked(ActionEvent event) throws IOException {
        boolean flag = true;

        // Reset the Labels
        UsernameWarningLabel.setText("");
        PasswordWarningLabel.setText("");

        // Stores user's input (username & password)
        String username = UsernameInput.getText();
        String password = PasswordInput.getText();

        // Checks if TextField is empty
        if (username.equals("") && password.equals("")){ // If both TextFields are empty
            UsernameWarningLabel.setTextFill(Color.RED);
            UsernameWarningLabel.setText("Username is empty..");
```

Image 3.2.3 LoginPageController

The program works by storing its data in a MySQL database; this is also the main data structure for this program. Whenever a new user has registered a new account, a new table with their username as the name of the table would be made specifically for that user, this table would store user's expenses data (id, type, item, amount, etc.). All of the account data though (id, username, password), will be stored in a separate table called `tb_users`.

When a date is chosen from the DatePicker (MainPage), the program crosschecks the date chosen with the date inside the MySQL database and shows items which are of the same date. Shown items are stored in the `ModelTable` object which are stored inside an `ObservableList`.

Using the user's username as the name of each respective table also makes it easy to access and update them based on the user's data.

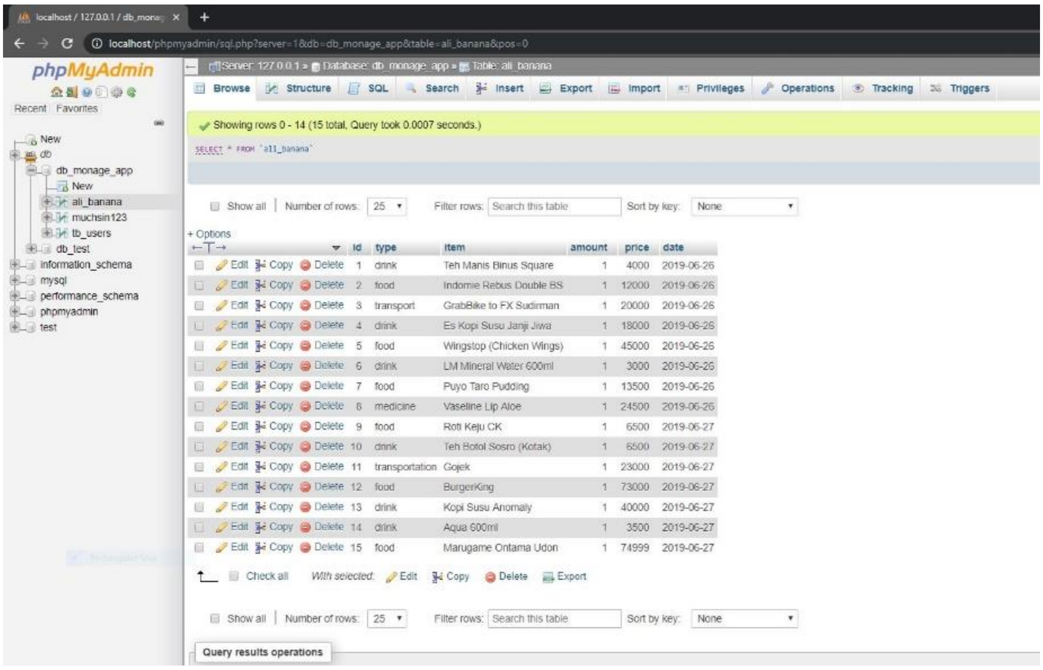


Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)

```
SELECT * FROM `tb_users`
```

id	password	income	username
13	7312081Alifio	1400000	Ali_Banana
14	12345678	4000000	muchsin123

Image 3.2.4 tb_users



Showing rows 0 - 14 (15 total, Query took 0.0007 seconds.)

```
SELECT * FROM `ali_banana`
```

id	type	item	amount	price	date
1	drink	Teh Manis Binus Square	1	4000	2019-06-26
2	food	Indomie Rebus Double BS	1	12000	2019-06-26
3	transport	GrabBike to FX Sudirman	1	20000	2019-06-26
4	drink	Es Kopi Susu Janji Jiwa	1	18000	2019-06-26
5	food	Wingstop (Chicken Wings)	1	45000	2019-06-26
6	drink	LM Mineral Water 600ml	1	3000	2019-06-26
7	food	Puyo Taro Pudding	1	13500	2019-06-26
8	medicine	Vaseline Lip Aloe	1	24500	2019-06-26
9	food	Roti Keju CK	1	6500	2019-06-27
10	drink	Teh Botol Sosro (Kotak)	1	6500	2019-06-27
11	transportation	Gojek	1	23000	2019-06-27
12	food	BurgerKing	1	73000	2019-06-27
13	drink	Kopi Susu Anomaly	1	40000	2019-06-27
14	drink	Aqua 600ml	1	3500	2019-06-27
15	food	Marugame Ontama Udon	1	74999	2019-06-27

localhost / 127.0.0.1 / db_monag
localhost/phpmyadmin/sql.php?server=1&db=db_monage_app&table=ali_banana&pos=0
phpMyAdmin
Server: 127.0.0.1 » Database: db_monage_app » Table: ali_banana
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options
Edit
Copy
Delete
Check all
With selected:
Query results operations

Image 3.2.5 table ali_banana

4. Evidence of Working Program

4.1. Login Page

When the program is run, it will direct the user to the Login Page, here the user will have a few options, if they don’t have an account yet they can click the Sign Up button; this will direct them to the Sign Up Page. Otherwise, they could log in with their existing account and password.

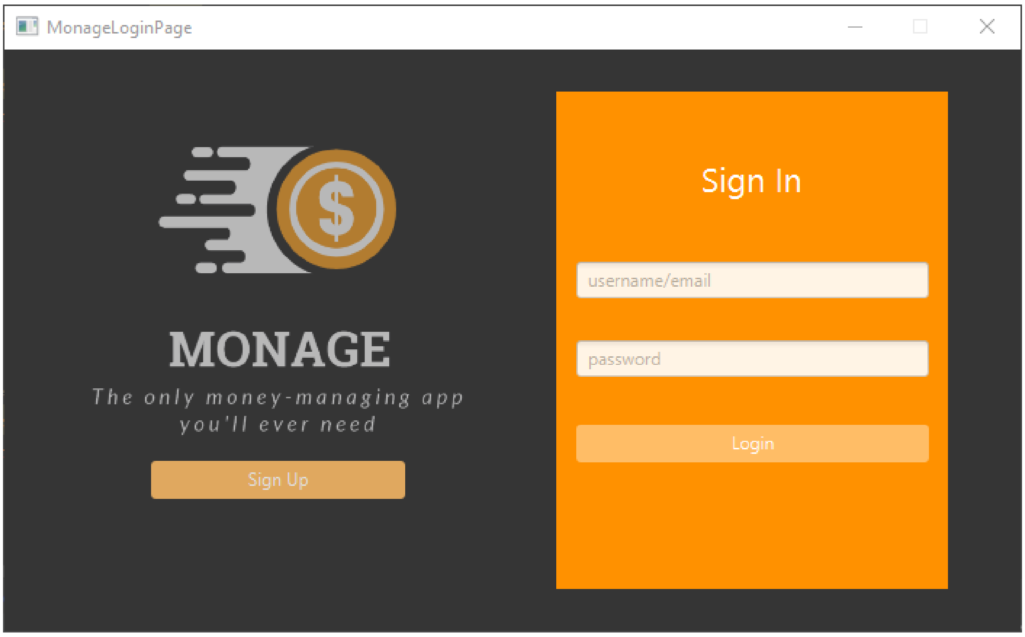
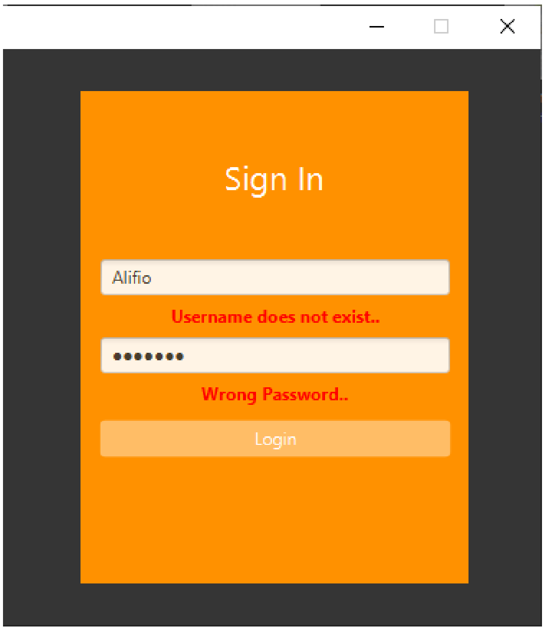


Image 4.1.1 LoginPage



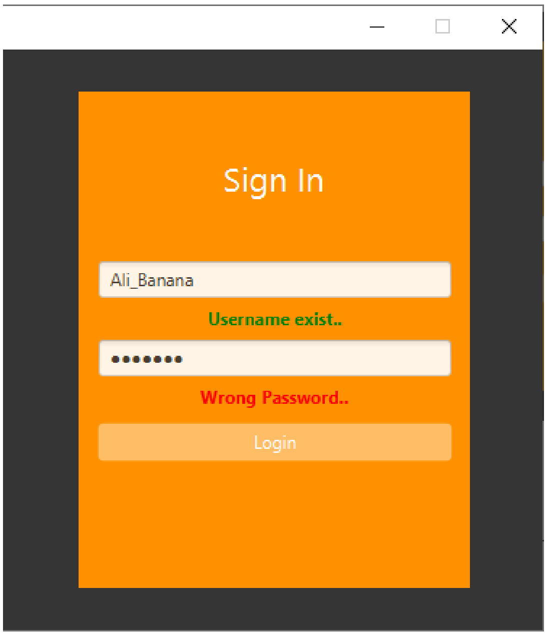
Alifio

Username does not exist..

•••••••

Wrong Password..

Image 4.1.2 LoginPage Warnings



Ali_Banana

Username exist..

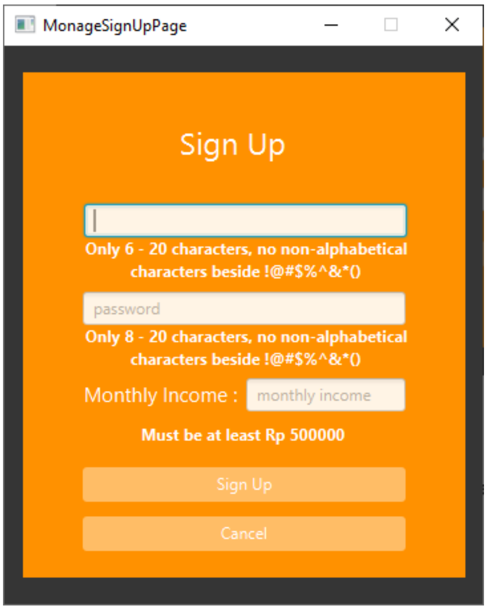
•••••••

Wrong Password..

Image 4.1.3 LoginPage Warnings

4.2. Sign Up Page

The Sign Up Page has a few instructions/warning by default below each text view. Although, if the user manages to insert the wrong format there will be a warning message shown in red; on the other hand, there will be a message in green if the user input the right format. Once the user has sign up, they will be redirected to the Login Page; their account information will be added to the database and a new table will be added with their username.



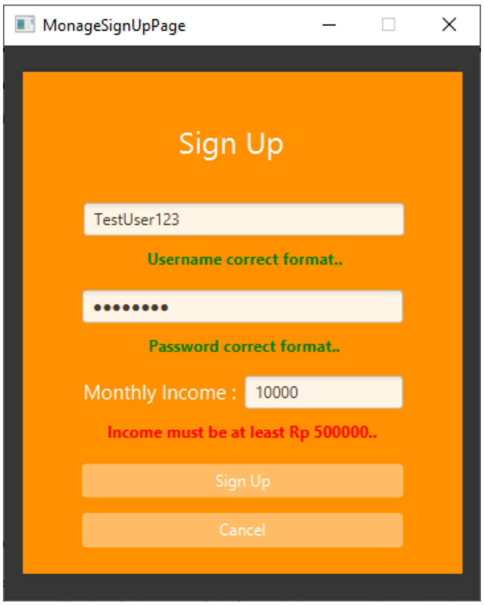
Only 6 - 20 characters, no non-alphabetical characters beside !@#$%^&*()

Only 8 - 20 characters, no non-alphabetical characters beside !@#$%^&*()

Monthly Income :

Must be at least Rp 500000

Image 4.2.1 SignUpPage



TestUser123

Username correct format..

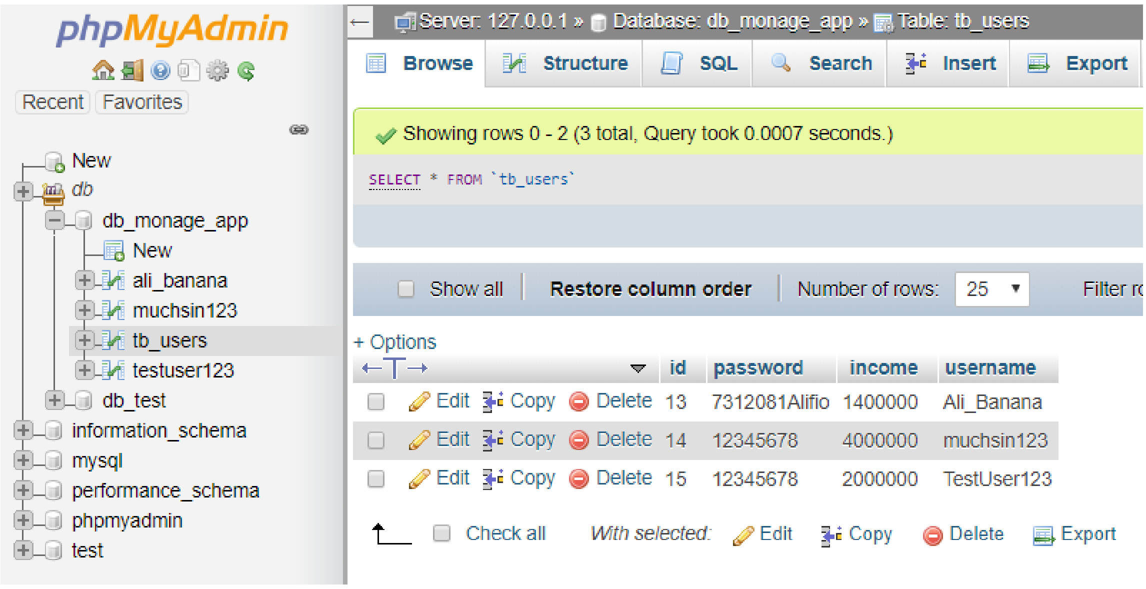
••••••••

Password correct format..

Monthly Income : 10000

Income must be at least Rp 500000..

Image 4.2.2 SignUpPage Warnings



Server: 127.0.0.1 » Database: db_monage_app » Table: tb_users

Showing rows 0 - 2 (3 total, Query took 0.0007 seconds.)

```
SELECT * FROM `tb_users`
```

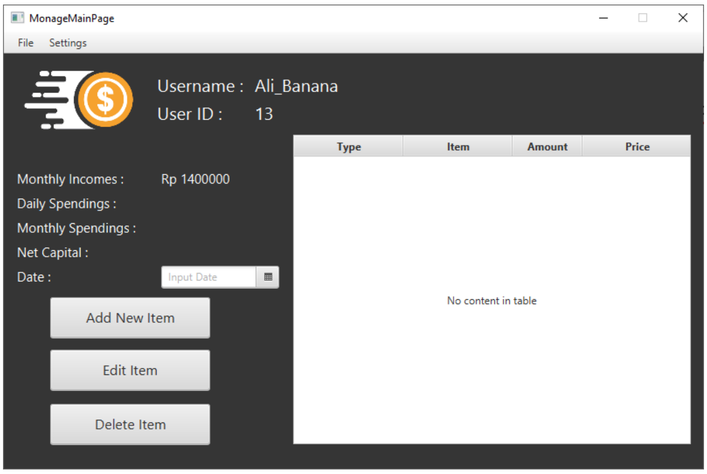
				id	password	income	username
□	Edit	Copy	Delete	13	7312081Alifio	1400000	Ali_Banana
□	Edit	Copy	Delete	14	12345678	4000000	muchsin123
□	Edit	Copy	Delete	15	12345678	2000000	TestUser123

Image 4.2.3 Table & Database

4.3. Main Page

Users will be directed to the Main Page after logging in with their account; this is where most of the program's features are located at. The top part of the page shows the program's logo as well as user's account information, the password is not shown for security purposes.

A large portion of the page is taken by the table which shows user's expenses for the chosen date (by default no date is chosen yet), for ease of use the table will automatically refresh itself when a date is chosen; there's however an option to refresh the table from "File" tab above. The top right empty space are used by warning labels that will warned users when doing something unwanted; for example, trying to add an item before choosing the date from the DatePicker.



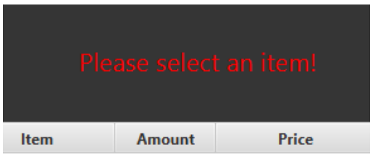
Username : Ali_Banana
User ID : 13

Monthly Incomes : Rp 1400000
Daily Spendings :
Monthly Spendings :
Net Capital :
Date : Input Date

Type	Item	Amount	Price

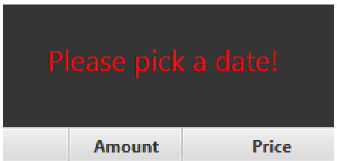
No content in table

Image 4.3.1 MainPage Default


Please select an item!

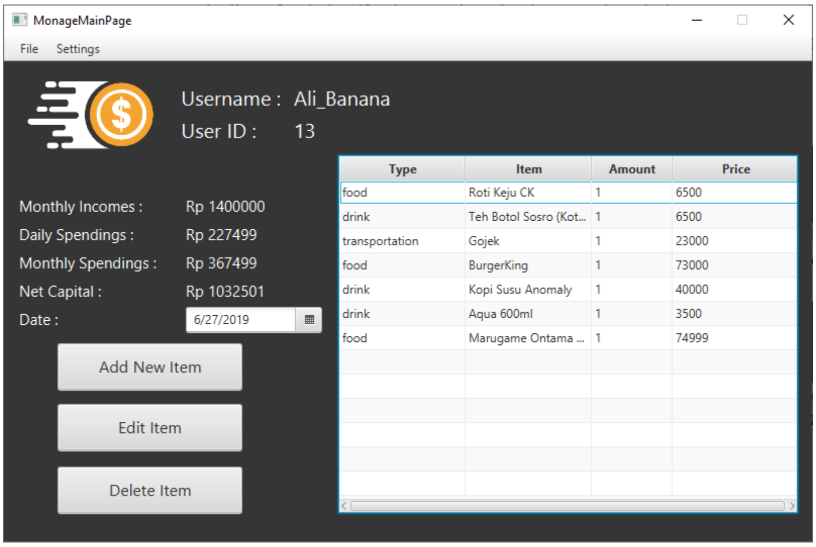
Item	Amount	Price

Image 4.3.2 Warning


Please pick a date!

	Amount	Price

Image 4.3.2 Warning2



Username : Ali_Banana
User ID : 13

Monthly Incomes : Rp 1400000
Daily Spendings : Rp 227499
Monthly Spendings : Rp 367499
Net Capital : Rp 1032501
Date : 6/27/2019

Type	Item	Amount	Price
food	Roti Keju CK	1	6500
drink	Teh Botol Sosro (Kot...	1	6500
transportation	Gojek	1	23000
food	BurgerKing	1	73000
drink	Kopi Susu Anomaly	1	40000
drink	Aqua 600ml	1	3500
food	Marugame Ontama ...	1	74999

Image 4.3.4 MainPage Filled

4.4. Add Item Page

Users will be directed to a new window “Add Item Page” when clicking on the “Add New Item” button on the main page, the program was set up so that users won’t be able to do anything on the Main page before closing on the current window. Just like the Login and Sign Up pages, this pages also have its own warning labels below every TextField to remind users to input data in the correct format. Clicking on either the “Cancel” or “Add Item” (with correct inputs) button will close the window and redirect user to the Main page.

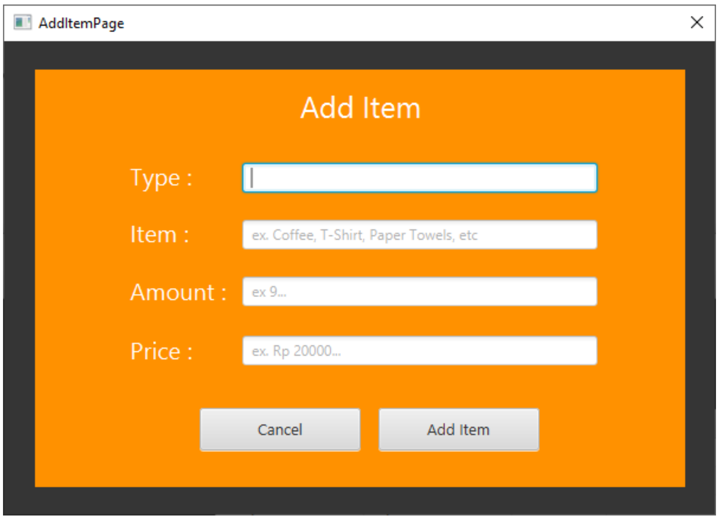
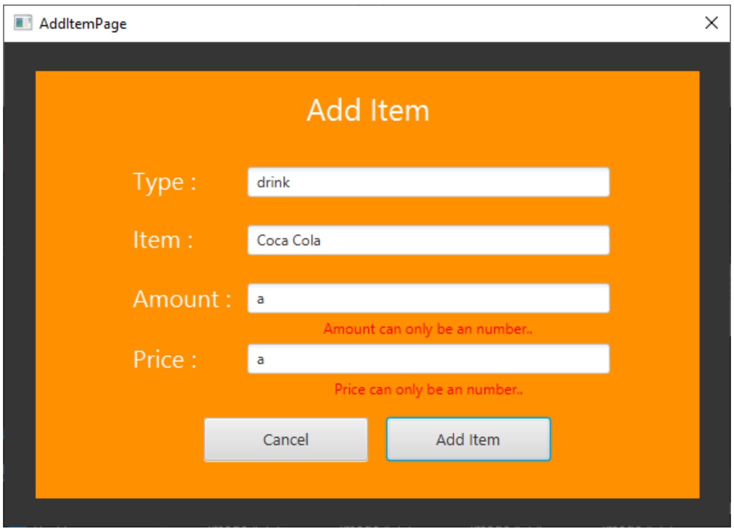


Image 4.4.1 AddItemPage Empty



Type : drink

Item : Coca Cola

Amount : a

Amount can only be an number..

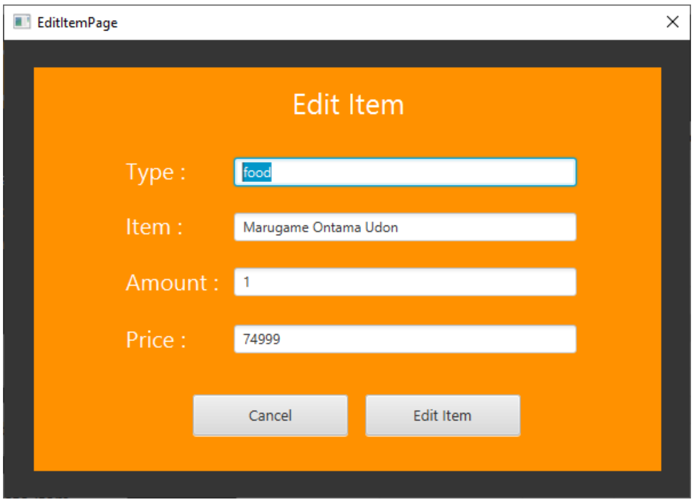
Price : a

Price can only be an number..

Image 4.4.2 Warnings

4.5. Edit Item Page

The “Edit Item Page” is very much similar in terms of layout with the “Add Item Page”, the main difference being that the user would need to select an item on the table in the Main Page before being able to click on the “Edit Item” button. Another main difference would be that once the new window is up, users will find that all of the TextFields has been occupied with data from the selected item. Just like the “Add Item Page” this page also has its own warning labels and validations.



EditItemPage

Edit Item

Type : food

Item : Marugame Ontama Udon

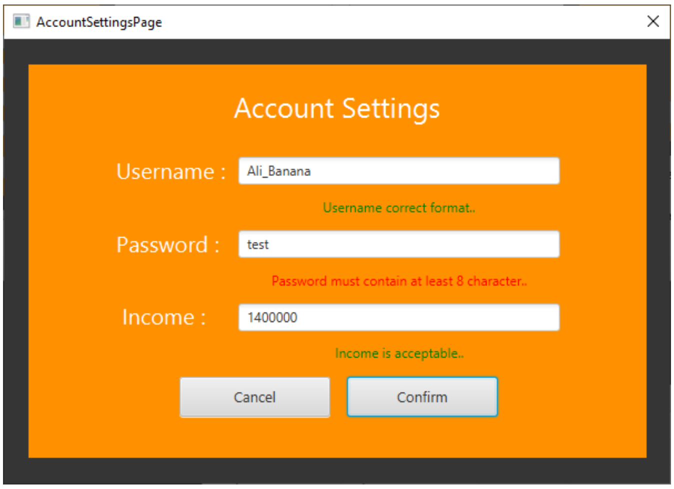
Amount : 1

Price : 74999

Image 4.5.1 EditItemPage

4.6. Account Settings Page

The ‘Account Settings” page are opened when the user clicks on the account settings option below the settings tab on the menu bar, it is similar to the “Edit Item” page in a way that it will automatically shows the user’s account information when its opened. The password is also shown to ease the users when changing their password. This page also has its own validations labels below each of its TextFields.



AccountSettingsPage

Account Settings

Username : Ali_Banana

Username correct format..

Password : test

Password must contain at least 8 character..

Income : 1400000

Income is acceptable..

Image 4.6.1 Account Settings

5. References

- https://www.javatpoint.com/javafx-overview
- https://docstore.mik.ua/orelly/java-ent/jenut/ch18_01.htm
- https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html
- https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG
- http://g2pc1.bu.edu/~qzpeng/manual/MySQL%20Commands.htm
- https://www.youtube.com/watch?v=XCgcQTQCfJQ

GitHub Link: https://github.com/alibanana/COMP6510-Final-Project