Even Semester (2019)



**BINUS UNIVERSITY**



**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Individual Work)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  |  |  |
| **Student Information**: | **Surname** | | |  |  | **Given Names**  **Rasyid** | **Student ID Number**  **2201798295** |
| 1. | **Alifio** | |
|  |  |  |  |  |  |  |  |
| **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**    **Submission Pattern** | **: Final Project** | |  |  |  |  |  |
| **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 3**
2. **Solution Design 3**
   1. **Program Structure Description 3**
   2. **UML Diagram 4**
3. **Code Implementation Discussion 5**
   1. **Implementation 5**
   2. **How it works 5**
4. **Evidence of Working Programme 8**
   1. **Login Page 8**
   2. **Sign Up Page 9**
   3. **Main Page 10**
   4. **Add Item Page 11**
   5. **Edit Item Page 12**
   6. **Account Settings Page 12**
5. **References 13**
6. **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.

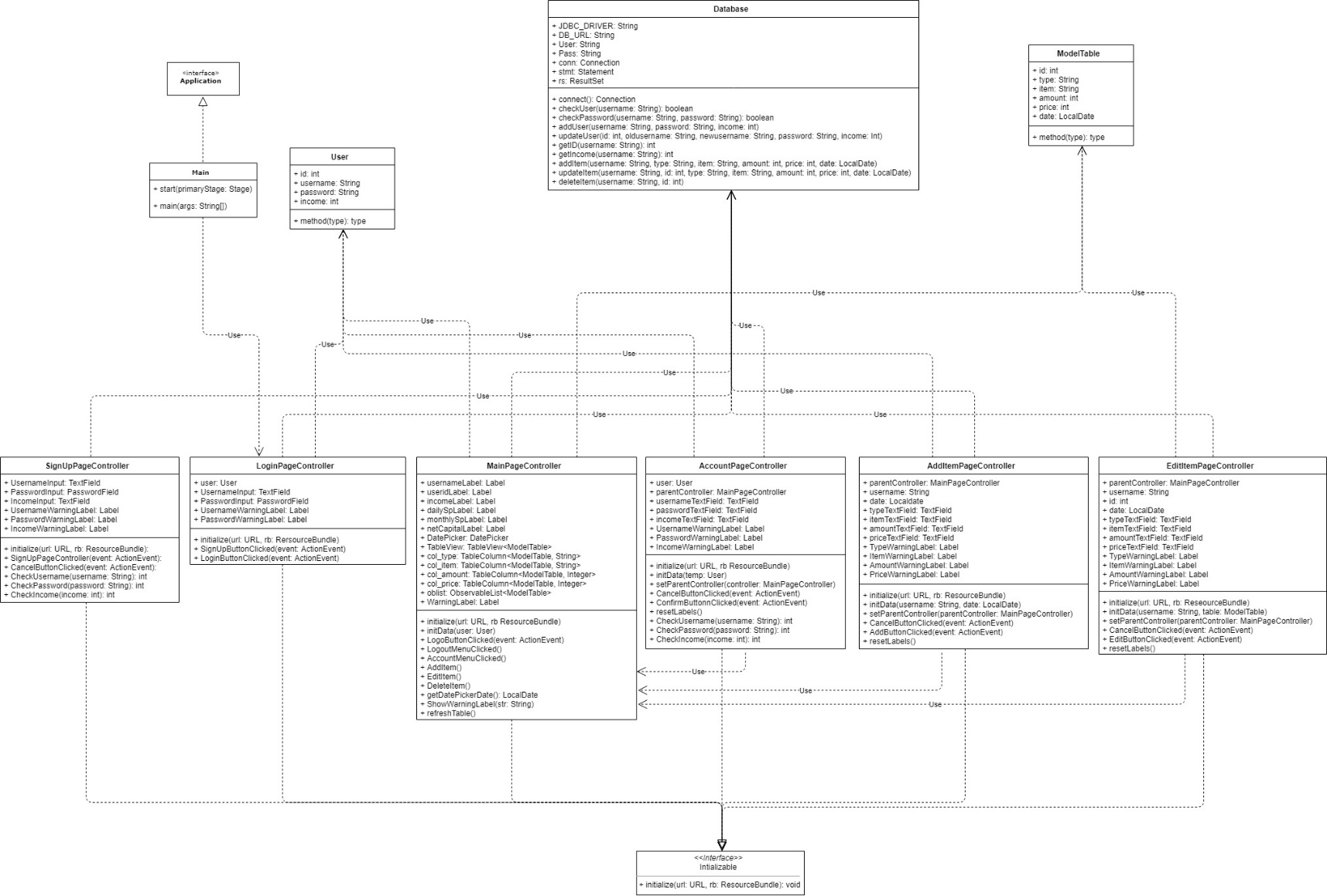
1. **Solution Design**
   1. **Program Structure Description**

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).

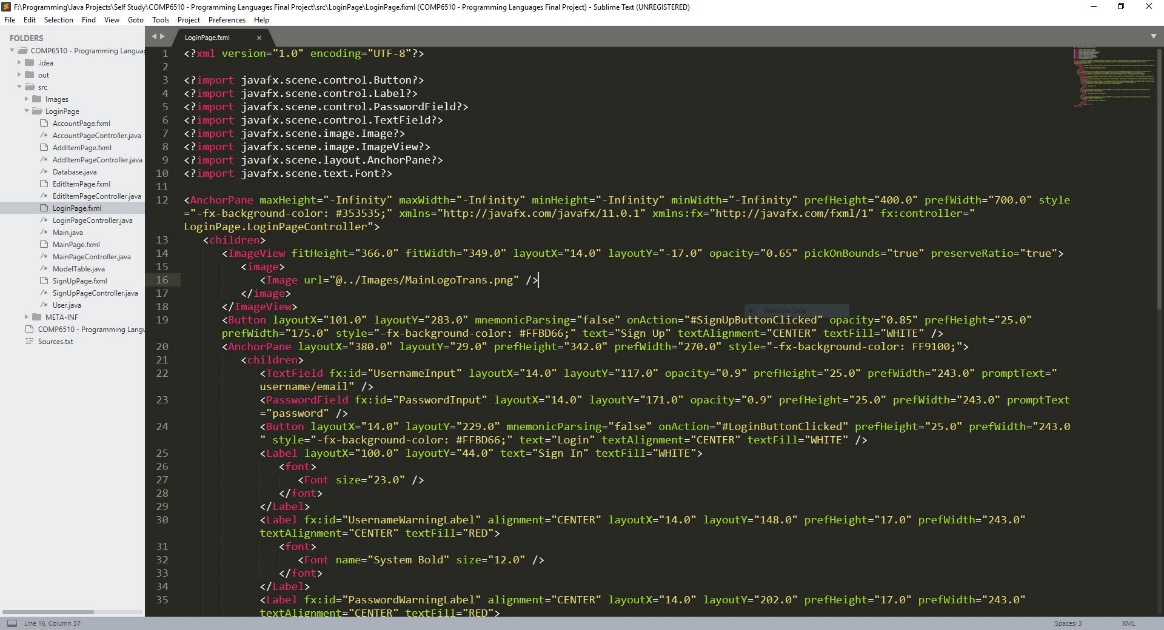
AccountPage, AddItemPage and EditItemPage controllers all stores a MainPageController object inside them. This enables the three classes to control the Main Page from outside of its controller, in this case it’s trying to refresh the table inside the Main Page.

* 1. **UML Diagram**

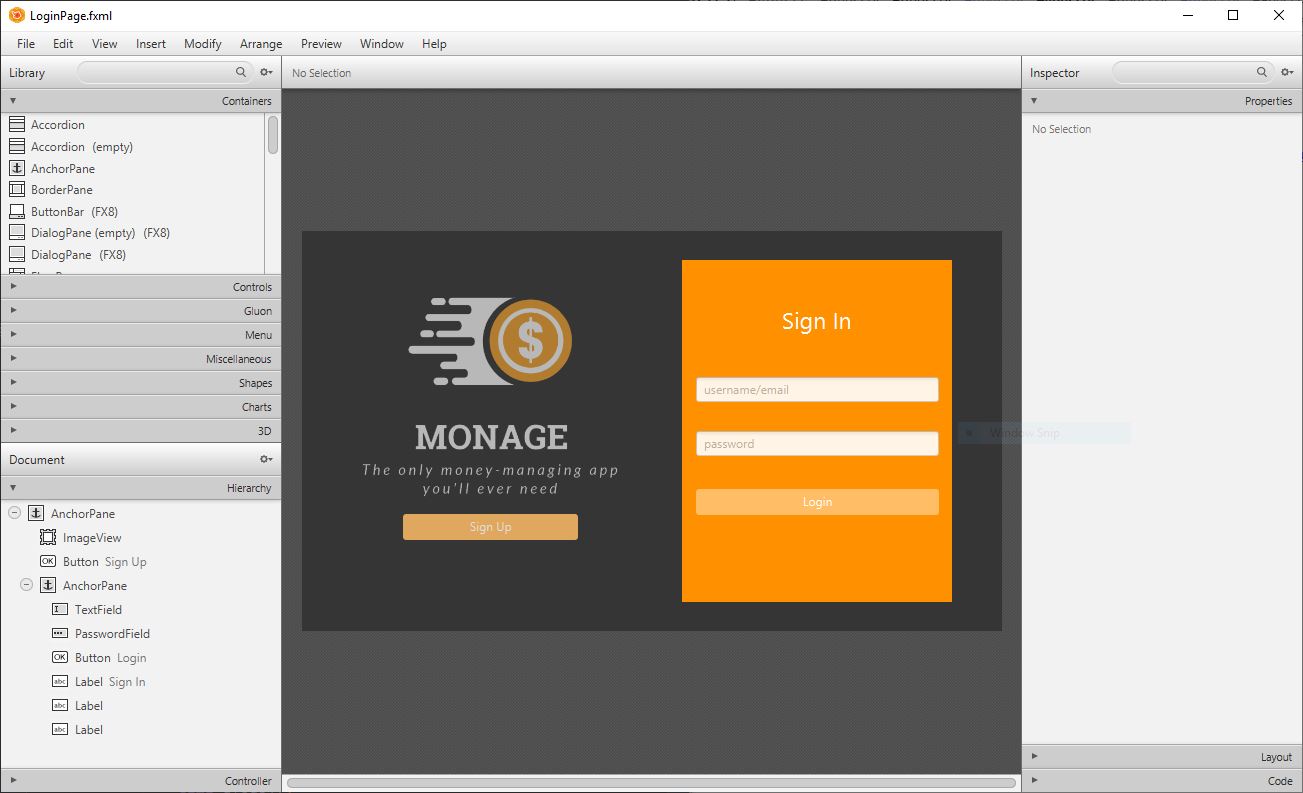
1. **Code Implementation Discussion**
   1. **Implementation**

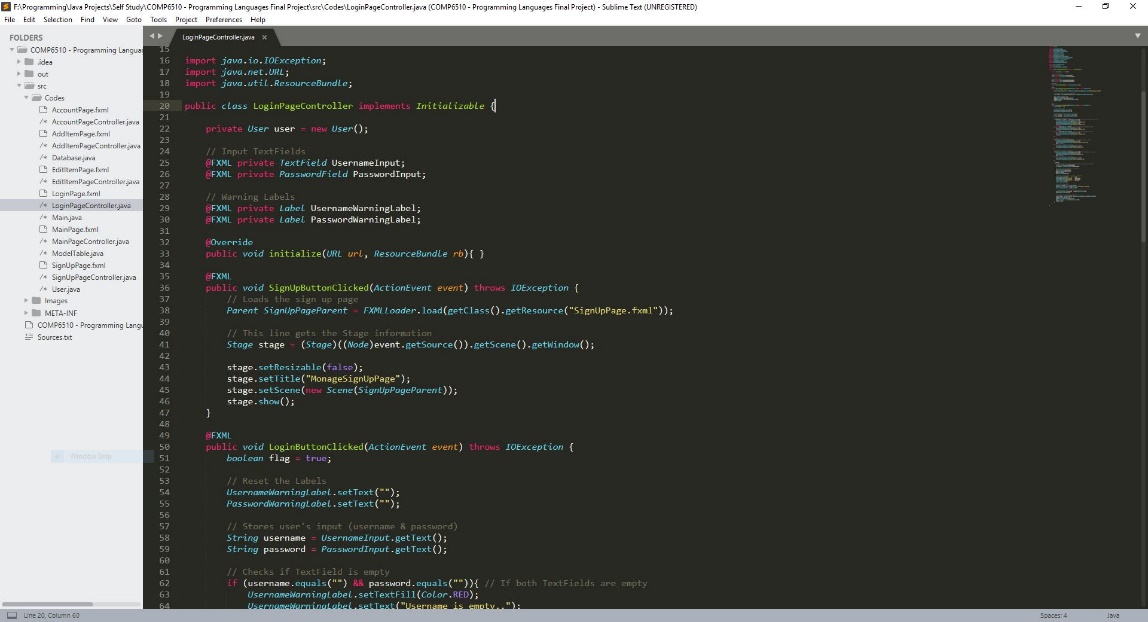
* **javafx**, library used to develop desktop application as well as Rich Internet Application (RIA). It is intended to replace swing in Java app as a GUI framework.
* **java.sql**, a package that contains all the JDBC API; enables java to send SQL statements to the database and retrieves the results.
* **java.time**, API for date and time.
  1. **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.



*Image 3.2.1 LoginPage.fxml*

**

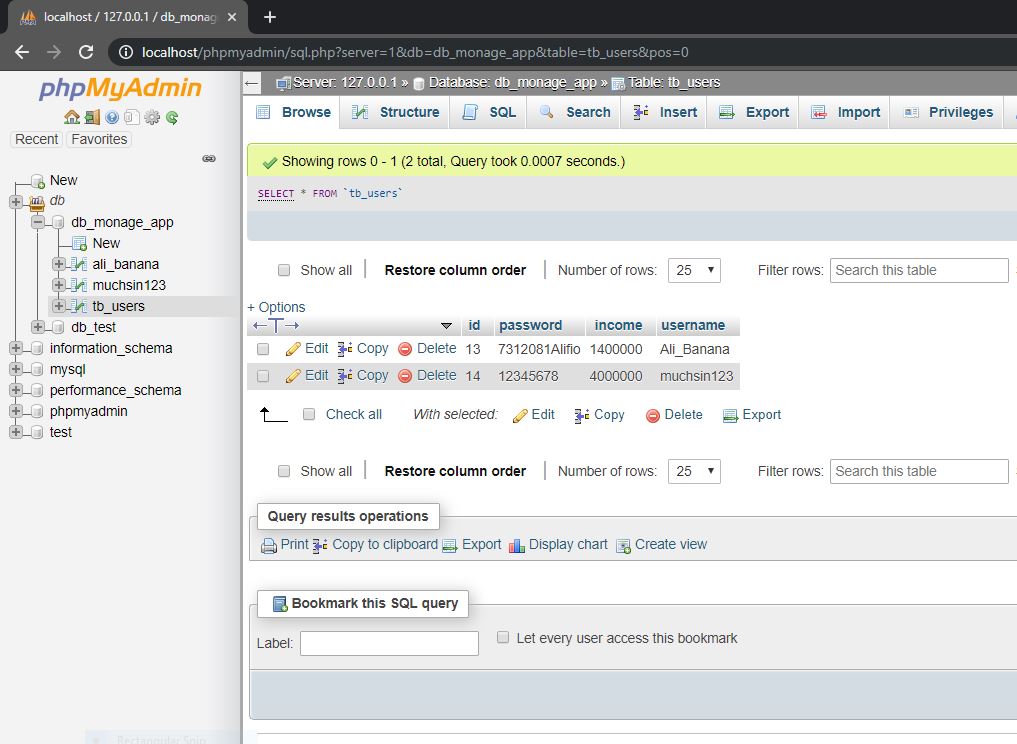
*Image 3.2.2 Scene Builder*

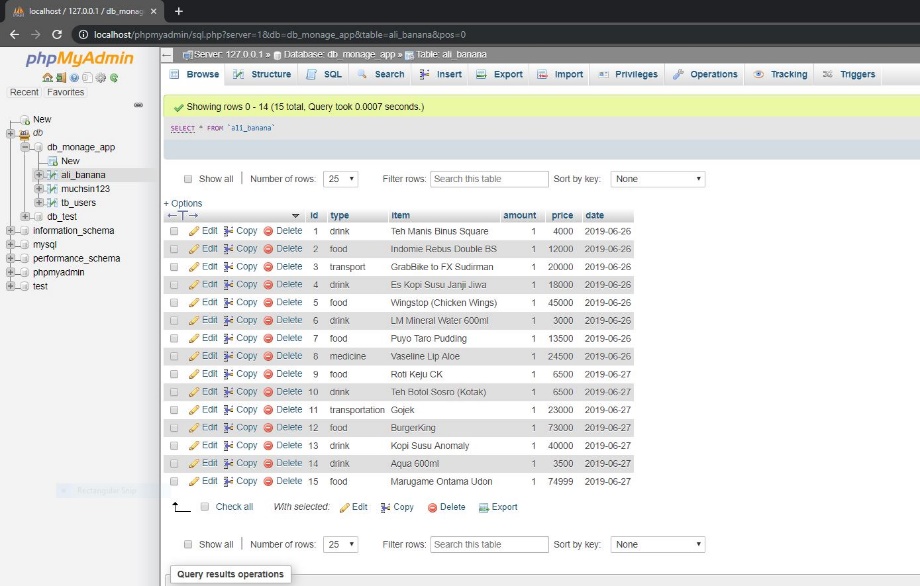
*Image 3.2.3 LoginPageController*

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.

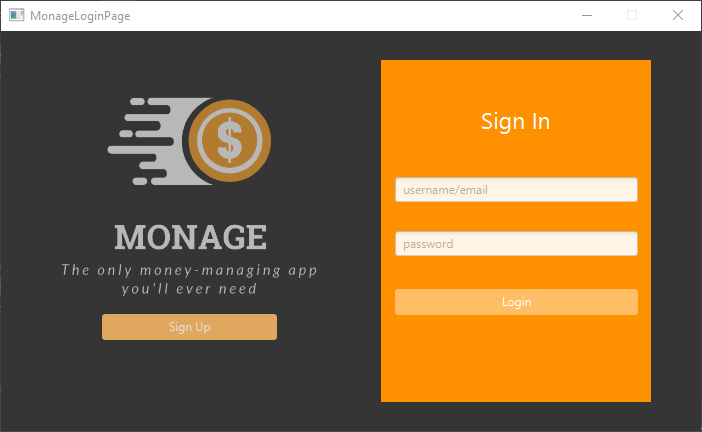
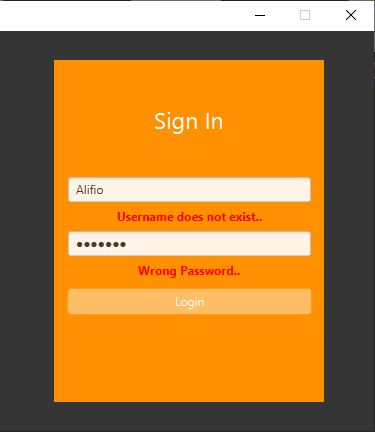
Using the user’s username as the name of each respective table also makes it easy to access and update them based on the user’s data.

*Image 3.2.4 tb\_users*

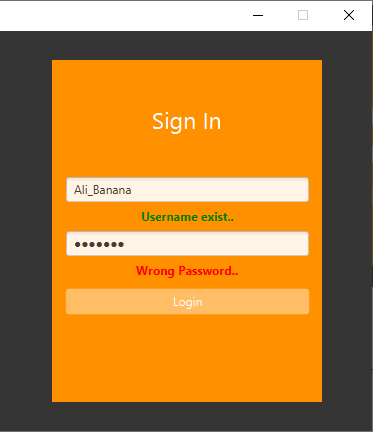


*Image 3.2.5 table ali\_banana*

1. **Evidence of Working Program**
   1. **Login Page**

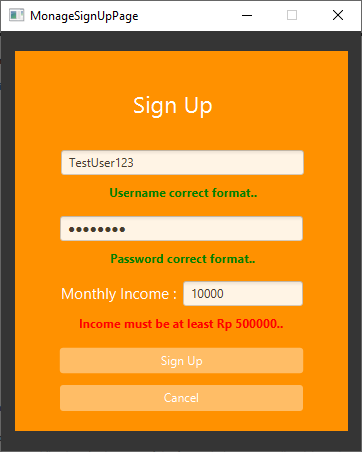
****When the program is run, it will direct the user to the Login Page, here the user will have a few options, if they don’t have an account yet they can click the Sign Up button; this will direct them to the Sign Up Page. Otherwise, they could log in with their existing account and password.

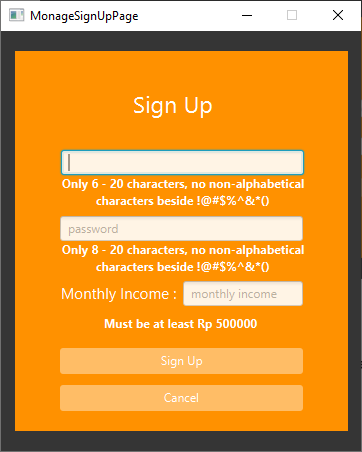
*Image 4.1.1 LoginPage*

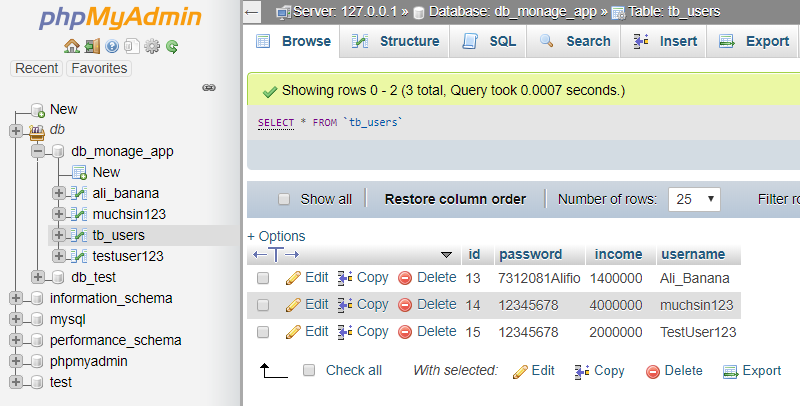


*Image 4.1.2 LoginPage Warnings Image 4.1.3 LoginPage Warnings*

* 1. **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.

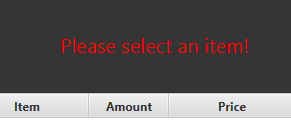
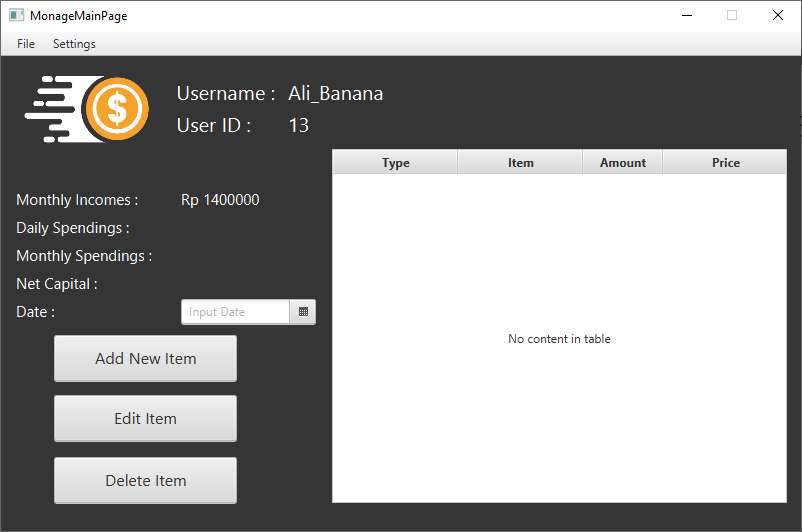
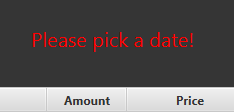


* Image 4.2.1 SignUpPage Image 4.2.2 SignUpPage Warnings*

*Image 4.2.3 Table & Database*

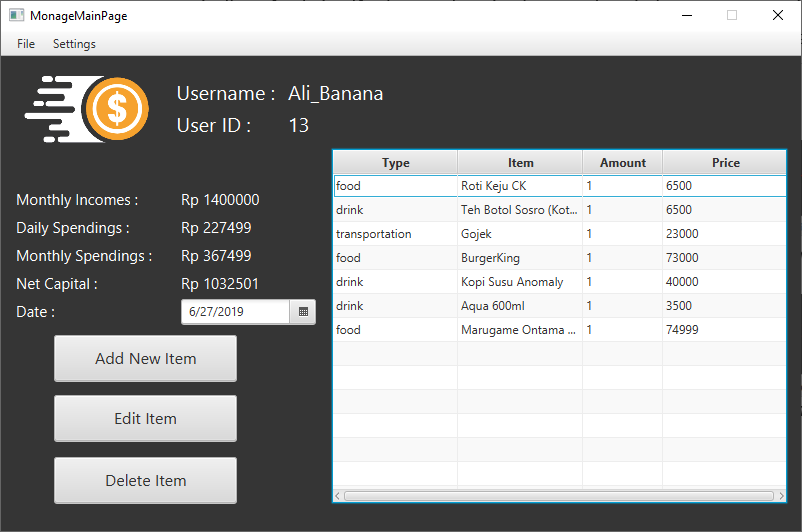
* 1. **Main Page**

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.

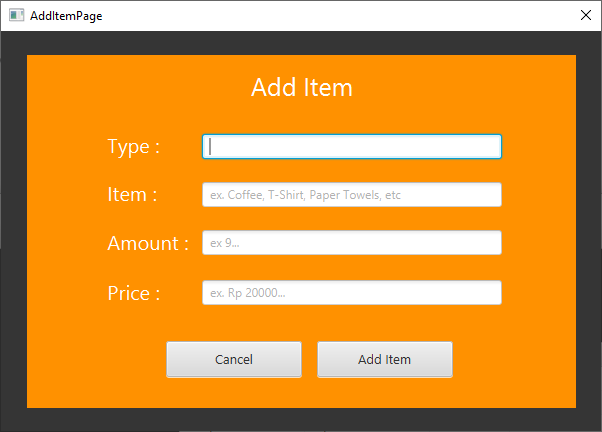
*Image 4.3.2 Warning*

*Image 4.3.2 Warning2*

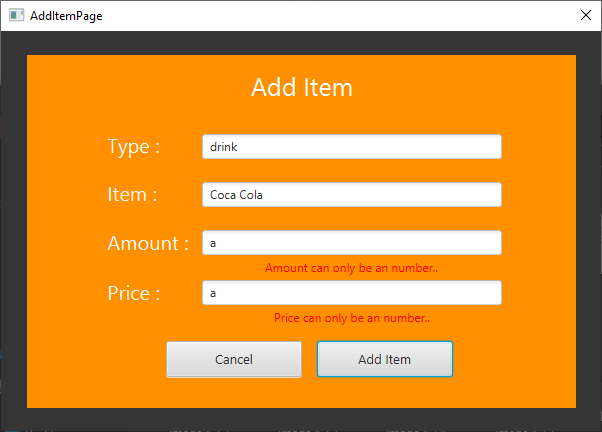
 *Image 4.3.1 MainPage Default*

*Image 4.3.4 MainPage Filled*

* 1. **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.

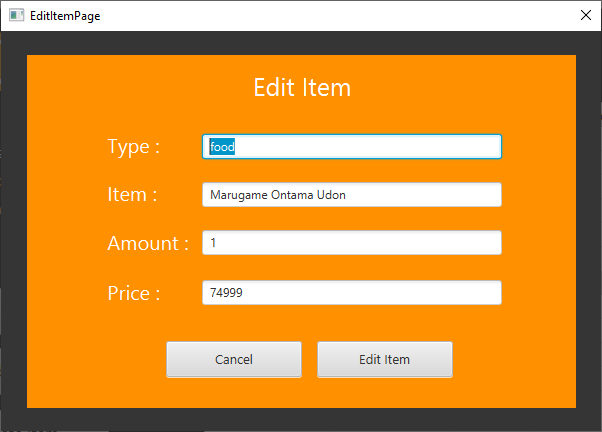
*Image 4.4.1 AddItemPage Empty*

****

*Image 4.4.2 Warnings*

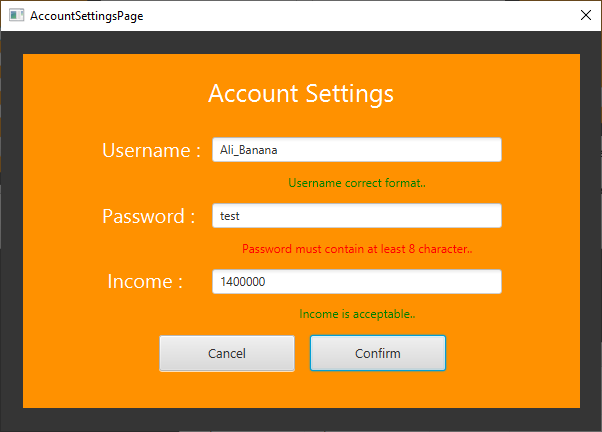
* 1. **Edit Item Page**

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.



*Image 4.5.1 EditItemPage*

* 1. **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.

*Image 4.6.1 Account Settings*

1. **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>