## Java Programming Report #3: Graphical User Interface

# ARM

Class : 18CLC2-KTPM

|  |  |
| --- | --- |
| **Doi07**: | **Đinh Hoàng Dương – 18127084**  **Dương Trần Mẫn Duy –18127087**  **Huỳnh Đức Lê – 18127126** |

**Table of content**

[Java Programming Report #3: Graphical User Interface 1](#_Toc59895923)

[**Revision History** 3](#_Toc59895924)

[**Individual Contributions Breakdown** 4](#_Toc59895925)

[**Introduction** 5](#_Toc59895926)

[**Analysis and design** 6](#_Toc59895927)

[**Implementation** 8](#_Toc59895928)

[**Demonstration** 15](#_Toc59895929)

[**Result** 17](#_Toc59895930)

[**Plan** 18](#_Toc59895931)

[**References** 18](#_Toc59895932)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 26/12/2020 | 1.0 | Finish the report | Duong, Duy, Le |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Individual Contributions Breakdown

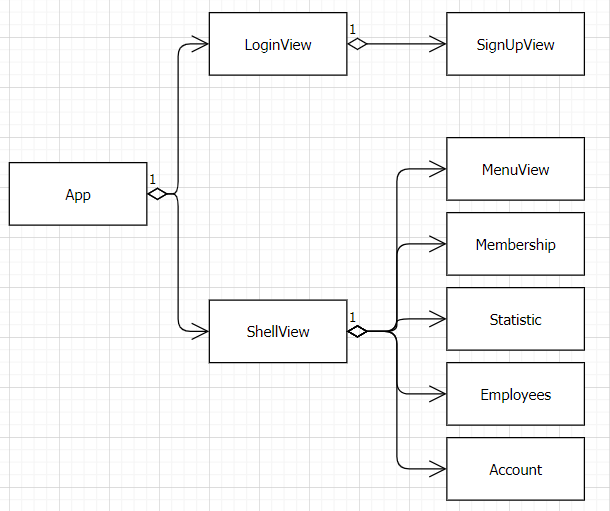
All team members contributed equally

# Introduction

Our app is expected to be used by employees and manager from a restaurant. Therefore, the app will have some restricted screens or functions which are only used when logining with manager account.

Instead of designing screens for normal employee and manager, we put them altogether. Base on account type, irrelevant UI components are grayed out or hidden.

# Analysis and design



Design:

Overall, the first screen user gonna see when open the app is LoginView. This screen contains sub-screens for login and sign up. It also provides button for navigating between them.

Only after login successfully, user is redirected to ShellView which is a tab view containing 5 sub-screens with MenuView as the default screen. StatisticView and EmployeeView are used by the manager so it would be gray out if user login with normal employee account

Analysis:

We have done 3 screens so far (LoginView, SignUpView, MenuView)

* LoginView:

There 3 things to notice in this screen which are user name, password field and the login button. It would take user approximately 5-6 actions to finish the login session:

1. Mouse click to user name field
2. Typing in
3. Move to password field by mouse or tab
4. Typing in
5. Enter/Click login button
6. Receiving login result

* SignUpView:

There’re many fields that needed to be filled in this screen. User will need about 10-11 actions:

1. Click SignUp button in login screen
2. Fill all the field
3. Click confirm button and receive sign up result

* MenuView:

This screen is saw by restaurant employees and the customer. Customers give their desired dishes, employee will order for them. Depending on how many dishes ordered, it would take from 3 to infinite actions.

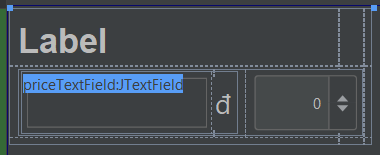
The basic flow is:

1. Look for item in the menu
2. Click order button in specific item
3. See the price and click confirm button to finish order

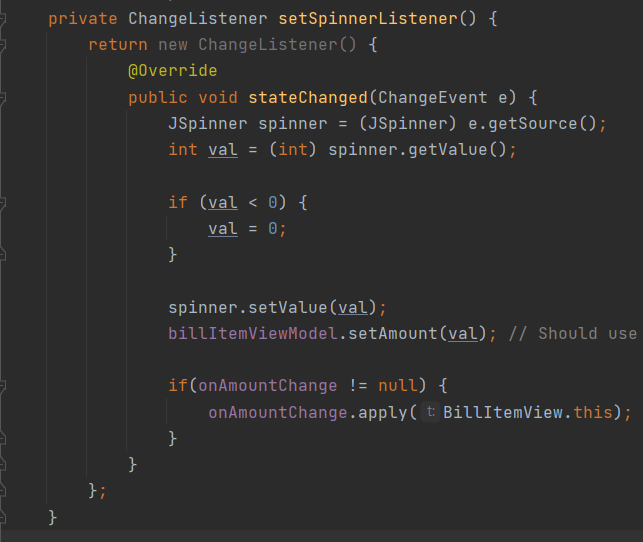
# Implementation

Using IntelliJ’s GUI Form to make User Interface for the Application

* Single Item in a Bill
  + Form:

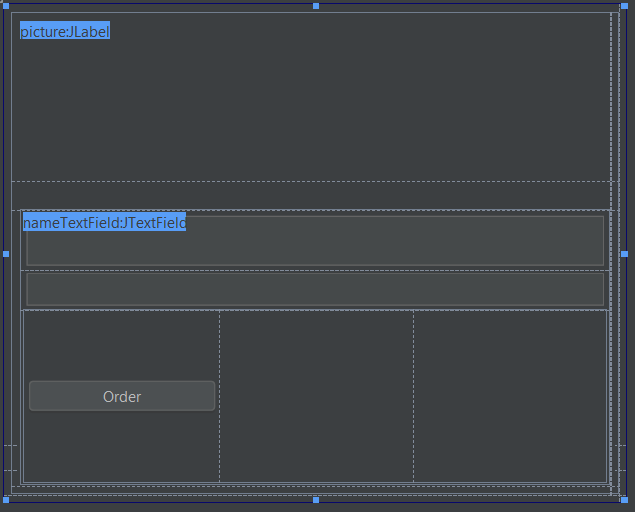


* + Main attributes are nameTextField: Item’s name; priceTextField: Item’s price - change according to amountSpinner; amountSpinner: show number of dishes has been chosen.
  + amountSpinner changes when clicking up arrow or down arrow and makes the price in priceTextField change too.

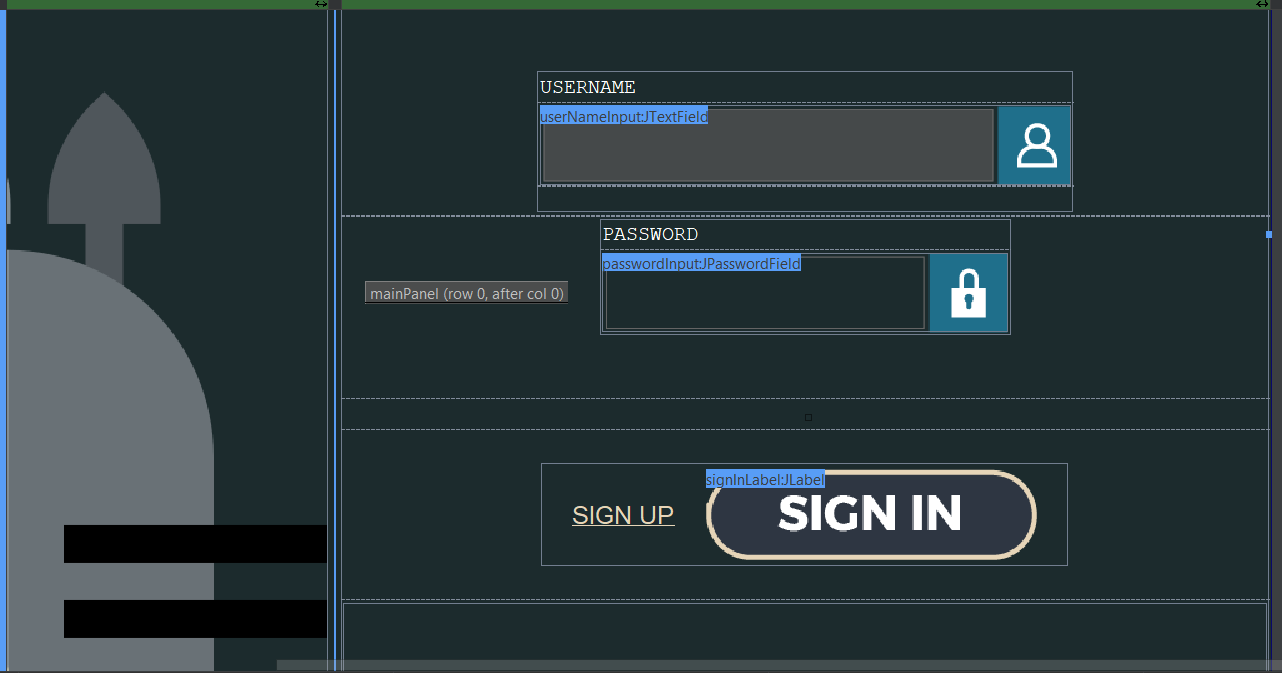


amountSpinner Listener

* Employees View (Update later)
* Single Item in the Menu
  + Form:



* + Main attributes are picture: Item’s illustration; nameTextField: show Item’s name; priceTextField: show Item’s price; orderButton: clicking this button, Item will be added to that Bill.
* Log-in View
  + Form:



* + userNameInput and passwordInput received input and checked with database when click signIn, if true switch to Shell, else throw Error message: “Incorrect Username or Password”; clicking signUp will switch to SignUp UI.
  + SignUp function:

private void onSignInButtonClick() {

try {

CompletableFuture<User> completableFuture = loginViewModel.loginAsync(userNameInput.getText(),

String.*copyValueOf*(passwordInput.getPassword()));

completableFuture.thenAccept(new Consumer<User>() {

@Override

public void accept(User user) {

if (user == null) {

infoLabel.setText("Incorrect Username or Password!");

try {

Thread.*sleep*(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

infoLabel.setText("");

}

else { //Login Success

DialogView.*showInfoDialog*("Login", "Success!");

onLoginSuccess.apply(user);

}

}

});

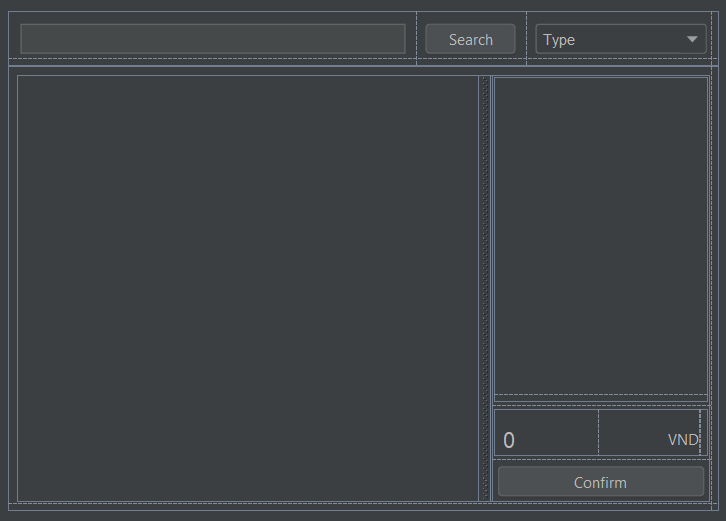
} catch (Exception exception) {

System.*out*.println(exception.getMessage());

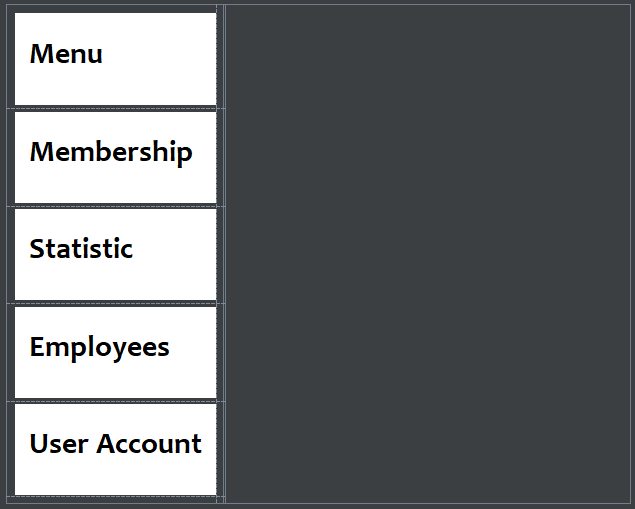
}

}

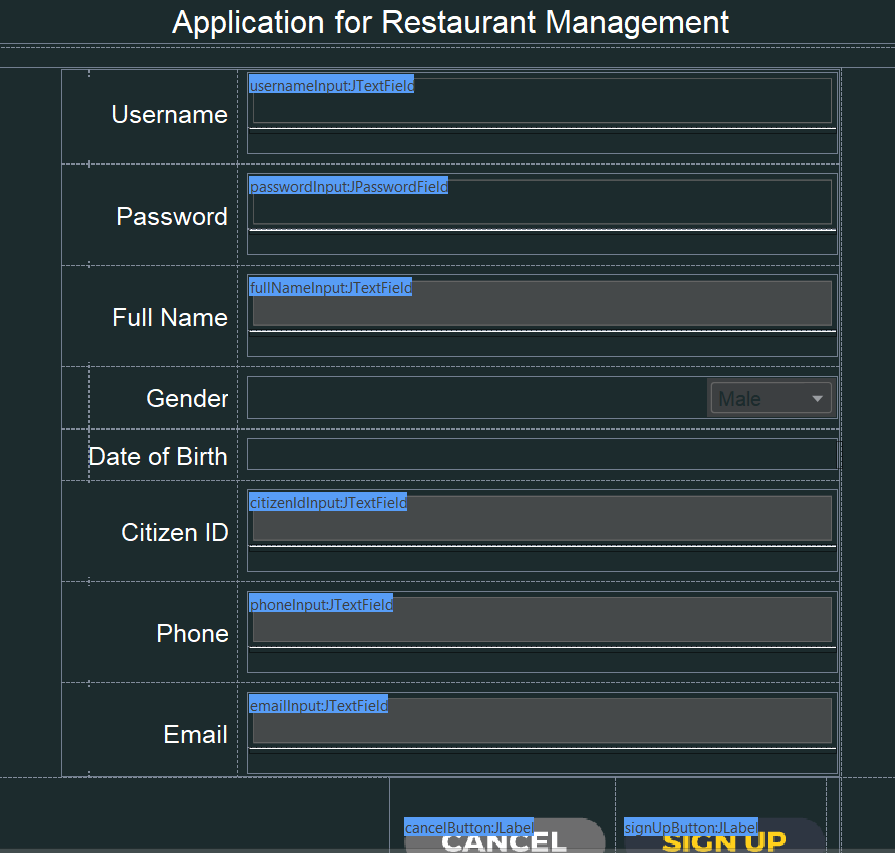
* Membership Scene (Update later)
* Menu
  + Form:



* + SearchBar and confirmButton will be updated later, SearchBar will find Item by typing in the TextArea, confirmButton will save Bill information and upload to MongoBD.
  + itemsScrollPanel and itemsPane will show the list of items (include all information about that item).
  + billPane will show a list of items have been chosen and their total price.
* Shell (main scene of the Application)
  + Form:



* + Each Label, when clicking them, the App will switch to the corresponding scene.
* Sign-up View
  + Form:



* + When filled all inputTextField with the right form, clicking the button will trigger the corresponding function.
  + Cancel button: return to Login scene
  + SignUp button:

private User registerUser() {

User user = new User();

String username = usernameInput.getText();

if (username == null || username.length() < 1) {

throw new NullPointerException("Username must not be empty");

}

if (!username.matches("[a**-**zA**-**Z0**-**9]\*")) {

throw new InputMismatchException("Username must only contain letter or number");

}

String password = String.*valueOf*(passwordInput.getPassword());

if (password.length() < 1) {

throw new NullPointerException("Password must not be empty");

}

if (!password.matches("[a**-**zA**-**Z0**-**9]\*")) {

throw new InputMismatchException("Password must only contain letter or number");

}

String citizenID = citizenIdInput.getText();

if (citizenID == null || citizenID.length() < 1) {

throw new NullPointerException("Citizen ID mustn't be empty!");

}

if (!citizenID.matches("[0**-**9]+")) {

throw new InputMismatchException("Citizen ID must be number");

}

String phone = phoneInput.getText();

if (phone == null || phone.length() < 1) {

throw new NullPointerException("Phone number mustn't be empty!");

}

if (!phone.matches("[0**-**9]+")) {

throw new InputMismatchException("Phone number must only be number");

}

user.setUsername(username);

user.setPassword(password);

user.setName(fullNameInput.getText());

user.setCitizenID(citizenID);

user.setGender(Objects.*requireNonNull*(genderConboBox.getSelectedItem()).toString());

user.setDoB(datePicker.getDateStringOrEmptyString());

user.setEmail(emailInput.getText());

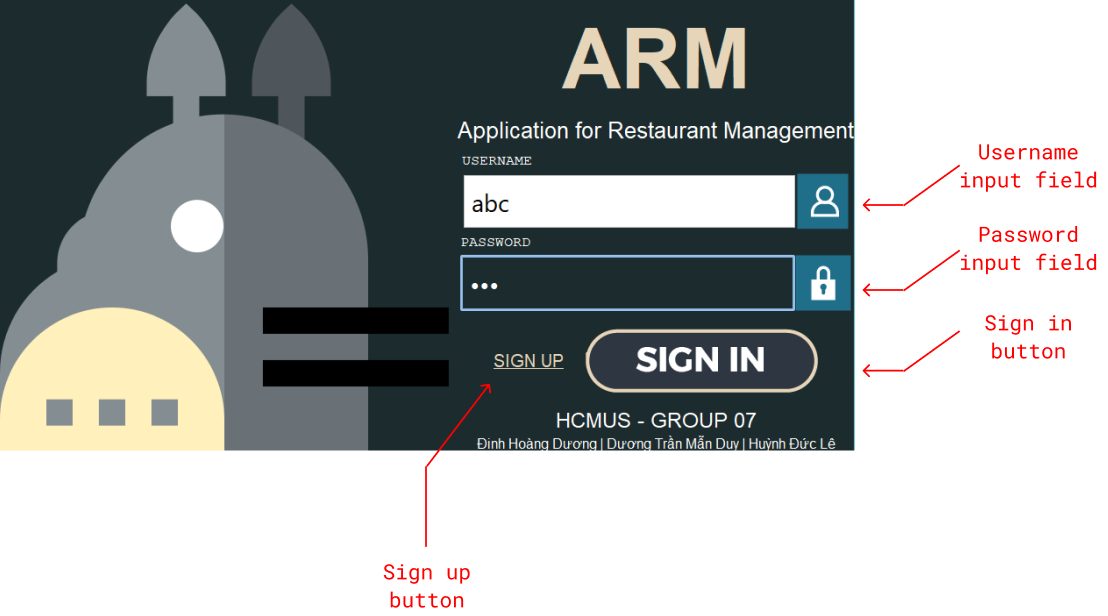
user.setPhoneNumber(phone);

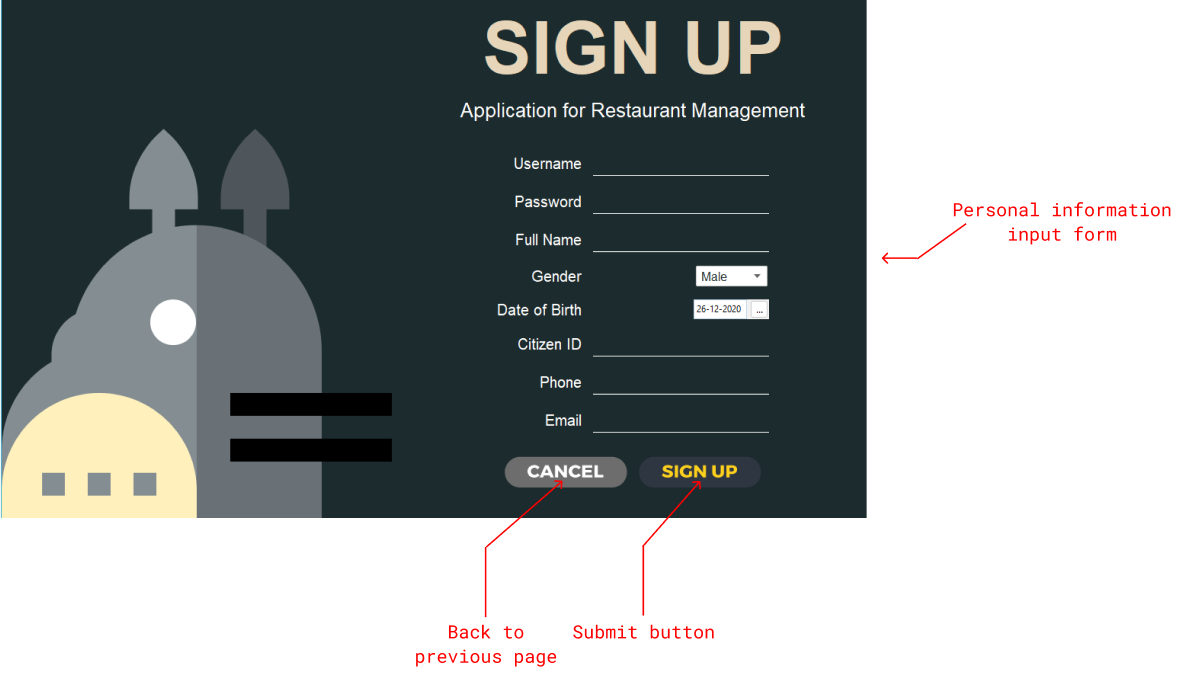
return user;

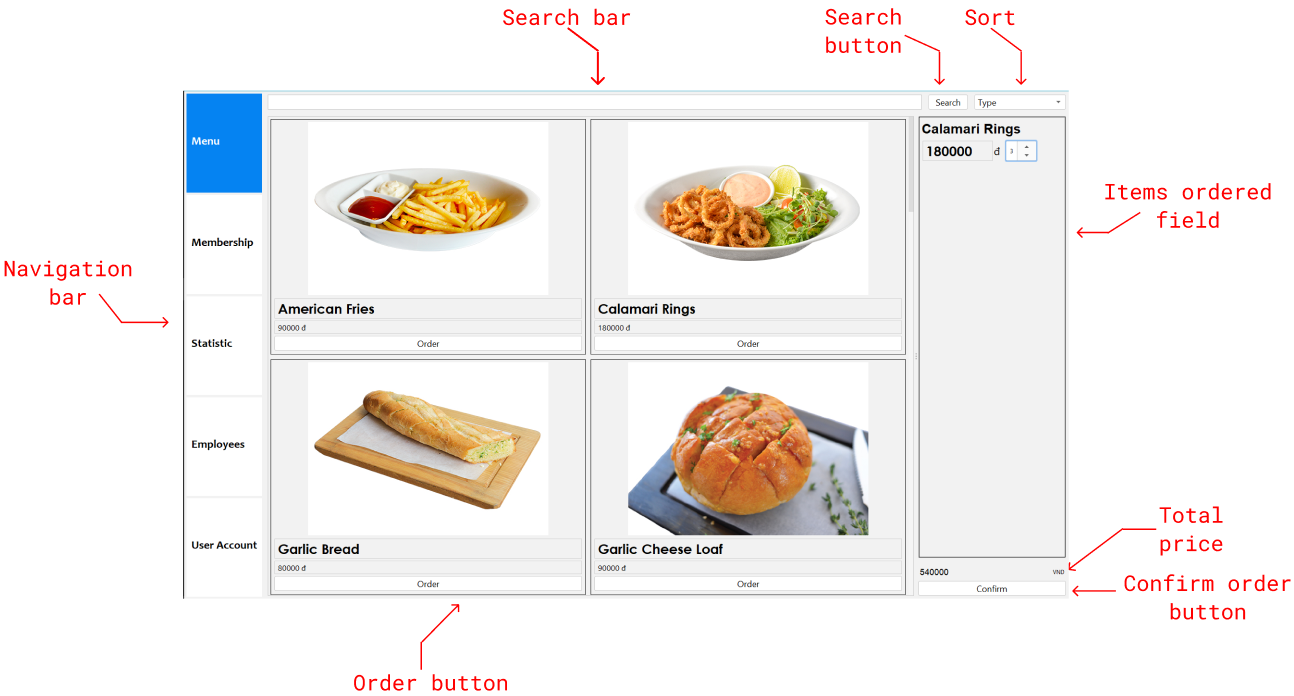
}

* Statistic (Update later)
* User’s account (Update later)

# Demonstration

****

****

****

# Result

Until to the present moment, we have done 3 major screens which is Login, SignUp and Menu. Beside, we also made some place-holders for remaining screens.

Some of the screens lie in ShellView are not actually drawn when user switches tab. The only solution we’ve found is minimizing/maximizing window programmatically. As a result, user would see a small glitching. This side effect is so short and it does the job so we will live with it.

# Plan

|  |  |  |
| --- | --- | --- |
| **Duration** | **Task** | **Author** |
| 27/12/2020 - 2/1/2021 | Exam week | Dương |
| Duy, Lê |
| 3/1/2021 - 9/1/2021 | Design Membership screen | Dương |
| Implement ViewModel for Membership screen | Duy, Lê |
| 10/1/2021 - Present day | Design UserAccount screen | Dương |
| Implement ViewModel for UserAccount screen | Duy, Lê |

# References

[Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/) - Solving all kinds of problem

<https://commons.apache.org/proper/commons-codec/download_codec.cgi> - For encryption

[CompletableFuture (Java Platform SE 8 )](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html) - Asynchronous Code

<https://www.flaticon.com/> - For UI elements

[Free Vectors, Stock Photos & PSD Downloads | Freepik](https://www.freepik.com/) - For UI elements

<https://github.com/LGoodDatePicker/LGoodDatePicker.git> - For date picker UI

[Maven Repository: com.google.code.gson » gson » 2.8.6 (mvnrepository.com)](https://mvnrepository.com/artifact/com.google.code.gson/gson/2.8.6) - For working with JSON

[https://www.mongodb.com](https://www.mongodb.com/) - For database