November 9, 2020

Beatrice Antoniu

w1688177

Enterprise Application development

coursework 1

Part A – Requirements

# functional requirements

1. The software should display a login page.
   1. The login page should have one input email/username text field for the user to fill.
   2. The login page should have one input password text field for the user to fill.
   3. The login page should have one login button that redirects the user to the financial app.
   4. The login page should have a register link.
      1. The register page should have an input email/username text field for the user to fill.
      2. The register page should have an input password text field for the user to fill.
      3. The register page should have a drop-down list of questions for the user to pick from.
      4. The register page should have an input text field for the user to answer the question they picked.
   5. The login page should have a password recovery link.
      1. The password recovery page should display a pop-up that asks the user for his secret question.
2. The software should allow the user to enter expenses for a certain date.
   1. The software must allow for the setting up of a payer or payee.
   2. The software will provide a form for the entry of expenses.
   3. An option will be given to enter recurring or one-off expenses.
3. The software should allow the user to produce a financial report.
4. The software should allow the user to get a prediction for a certain date.
   1. The software must allow for the selection of a payer or payee.
5. The software should have a pre-set list of contacts.
   1. The software should allow the user to modify payees and payers.
      1. The software must allow the user to update a payee’s or a payer’s information.
      2. The software must allow the user to remove a payee or a payer.
6. The software should display a “Logout” button.
7. The software should display an “Exit” button.

## Non-functional requirements

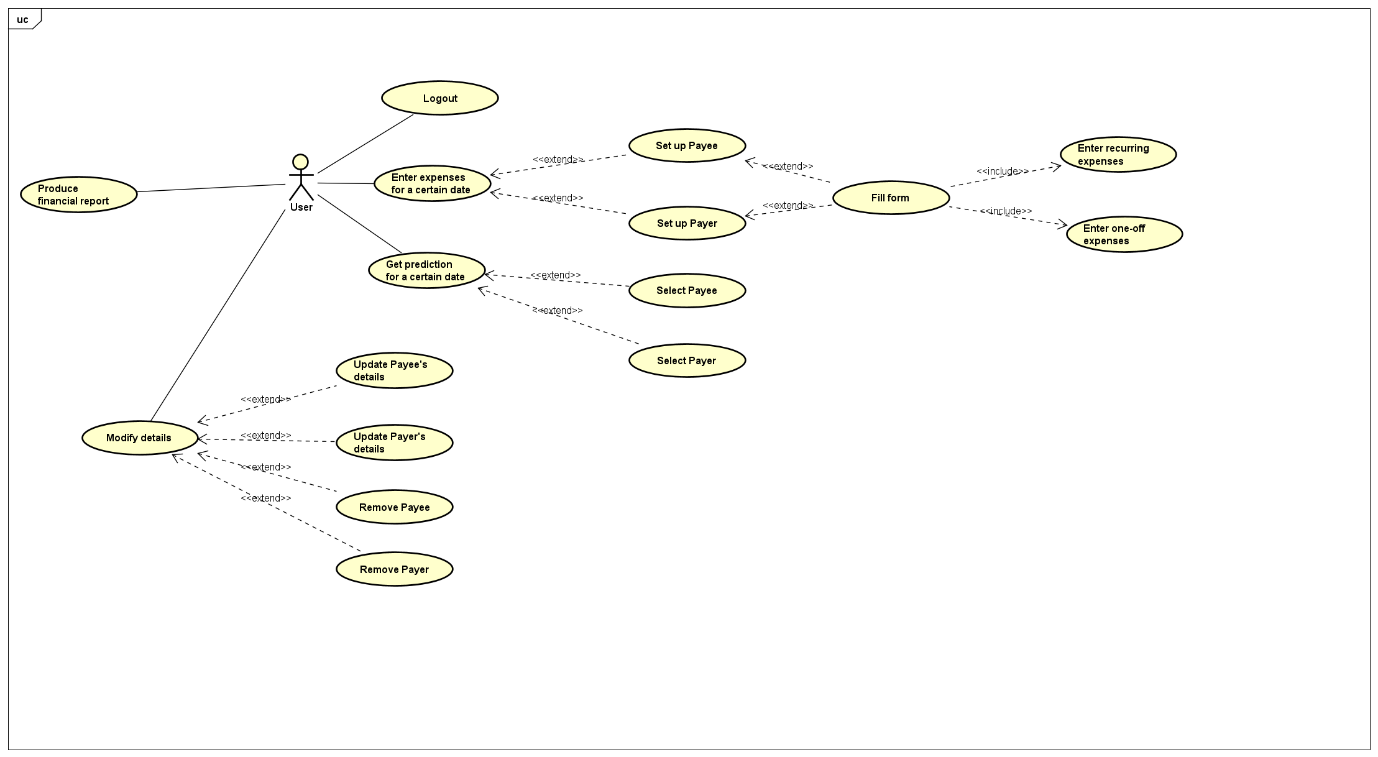
1. The software must run on Windows.
2. The software must be implemented using C# and Windows Forms.
3. The software cannot be used unless the user is registered.
4. The software requires the user to login in order to use the financial application.

Part B – Use Case Diagrams

# diagrams

Use Case Diagram - Login Page


Use Case Diagram - Login Page



Use Case Diagram - Financial App

## description

|  |  |
| --- | --- |
| Use Case ID | UC: Login |
| Functional requirement associated ID | R1: Login |
| Description | The user wishes to access to the system’s functionalities. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | Database |
| Stakeholders and Interests | User: be able to access to the system’s functionalities.  System: grant access to legitimate users. |
| Pre-conditions | No user was logged in in the system. |
| Post-conditions | The user is allowed to access to the software’s functionality. |
| Trigger | The user wishes to login to his/her account. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Starts the software. |  |
|  | 2. Opens a view displaying fields to fill  including username and password |
| 3. Writes a username |  |
| 4. Writes a password |  |
| 5. Selects the option to login. |  |
|  | 6. Checks the combination of username and  password. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Login2 |
| Functional requirement associated ID | R1.IV: Register |
| Description | The user wishes to create an account to access the system’s functionalities. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | Database |
| Stakeholders and Interests | User: be able to create an account to access the system’s functionalities.  System: record new account to database |
| Pre-conditions | No user was registered with the same details |
| Post-conditions | The user creates an account successfully. |
| Trigger | The user uses the register link. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the register link. |  |
|  | 2. Opens a view displaying fields to fill  including username, password, and secret question |
| 3. Writes a username |  |
| 4. Writes a password |  |
| 5. Picks a question |  |
| 6. Writes an answer |  |
| 7. Selects the option to register. |  |
|  | 8. Checks the details with the database and goes back to the login page. |
|  |  |
|  | End of use case. |

### Alternate scenario 1

|  |  |
| --- | --- |
| **User** | **System** |
| 3. Writes a username |  |
|  | 4. Checks that username is not empty |
| 5. Username textbox filled |  |
|  | 6. Checks the details with the database and goes back to the login page. |
|  |  |
|  | End of use case. |

### Alternate scenario 2

|  |  |
| --- | --- |
| **User** | **System** |
| 3. Writes a username |  |
|  | 4. Checks that username is not taken |
| 5. Username is unique |  |
|  | 6. Checks the details with the database and goes back to the login page. |
|  |  |
|  | End of use case. |

### Alternate scenario 3

|  |  |
| --- | --- |
| **User** | **System** |
| 4. Writes a password |  |
|  | 5. Checks that password is not empty |
| 6. Password textbox filled |  |
|  | 7. Checks the details with the database and goes back to the login page. |
|  |  |
|  | End of use case. |

### alternate scenario 4

|  |  |
| --- | --- |
| **User** | **System** |
| 4. Writes a password |  |
|  | 5. Checks that password’s length is at least equal to 6 |
| 6. Password’s length is equal or bigger than 6 |  |
|  | 7. Checks the details with the database and goes back to the login page. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Login3 |
| Functional requirement associated ID | R1.V: Forgot Password |
| Description | The user wishes to recover their account to access the system’s functionalities. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | Database |
| Stakeholders and Interests | User: be able to create a new password to access their account.  System: retrieve account and create new password to database |
| Pre-conditions | User exists. |
| Post-conditions | The user creates a new password and gets access to their account successfully. |
| Trigger | The user uses the recovery link. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the recovery link. |  |
|  | 2. Opens a view displaying a field for the user to fill his secret question. |
| 3. Writes secret question answer |  |
|  | 4. Checks the details with the database. |
| 5. Accepts the answer. |  |
|  | 6. Prompts for new password. |
| 7. Writes new password. |  |
|  | 8.Records new password and goes back to the login page. |
|  |  |
|  | End of use case. |

### Alternate scenario 1

|  |  |
| --- | --- |
| **User** | **System** |
| 3. Writes secret question answer |  |
|  | 4. Checks that field is not empty |
| 5. Accepts the answer |  |
|  | 6. Checks the details with the database and goes back to the login page. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Login |
| Functional requirement associated ID | R7: Exit |
| Description | The user wishes to close the software. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to close the software.  System: turn off. |
| Pre-conditions | User has to be on login page |
| Post-conditions | The user is able to close the software. |
| Trigger | The user wishes to close the application. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the exit button. |  |
|  | 2. Closes the application. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Financial App |
| Functional requirement associated ID | R2: Enter expenses |
| Description | The user wishes to enter new expenses. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to enter new expenses.  System: records new expenses. |
| Pre-conditions | The expenses are not recorded yet. |
| Post-conditions | The user is able to record new expenses. |
| Trigger | The user wishes to enter new expenses to his/her account. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the enter expenses link. |  |
|  | 2. Opens a view displaying a date picker. |
| 3. Picks date. |  |
|  | 4. Opens a view displaying a form for the contact details. |
| 5. Enters details. |  |
|  | 6. Gives option for the expenses type. |
| 7. Picks the expenses type. |  |
|  | 8. Records the contact details and the expenses. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Financial App |
| Functional requirement associated ID | R3: Produce financial report |
| Description | The user wishes to produce a financial report of their expenses. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to produce a report.  System: creates a report for the user. |
| Pre-conditions | There is existent data for the report do be produced. |
| Post-conditions | The user is shown a report of their expenses. |
| Trigger | The user wishes to produce a report to his/her expenses. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the financial report link. |  |
|  | 2. Opens a view displaying a date picker. |
| 3. Selects date. |  |
|  | 4. Opens a view displaying the contacts on the picked date. |
| 5. Selects contact. |  |
|  | 6. Displays report. |
|  |  |
|  | End of use case. |

### alternate scenario 1

|  |  |
| --- | --- |
| **User** | **System** |
| 3. Selects date. |  |
|  | 4. There is no recorded data on the selected date. |
|  | 5. Returns to the dashboard. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Financial App |
| Functional requirement associated ID | R4: Get prediction of a certain date |
| Description | The user wishes to have his expenses predicted on a date of their choice. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to get a prediction on a certain date.  System: create a prediction of the user’s expenses. |
| Pre-conditions | There is existent data for the prediction to be created. |
| Post-conditions | The user is shown a prediction of their expenses. |
| Trigger | The user wishes to get a prediction of his/her expenses. |

### MAin success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the expenses prediction link. |  |
|  | 2. Opens a view displaying a date picker. |
| 3. Selects date. |  |
|  | 4. Displays prediction. |
|  |  |
|  | End of use case. |

### alternate scenario 1

|  |  |
| --- | --- |
| **User** | **System** |
| 3. Selects date. |  |
|  | 4. There is no recorded data on the selected date. |
|  | 5. Returns to the dashboard. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Financial App |
| Functional requirement associated ID | R5.I.i: Update details |
| Description | The user wishes to modify their contacts. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to modify existent details of the payees and payers.  System: record the modifications made by the user. |
| Pre-conditions | There are contacts in the user’s list. |
| Post-conditions | The user can modify their contacts. |
| Trigger | The user wishes to modify a payee’s or payer’s information. |

### Main success Scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the contacts’ link. |  |
|  | 2. Opens a view displaying the contacts’ list. |
| 3. Selects the payee/payer. |  |
|  | 4. Opens a view displaying the contact’s details. |
| 5. Modifies details. |  |
|  | 6. Saves changes. |
|  |  |
|  | End of use case. |

### Alternate scenario 1

|  |  |
| --- | --- |
| **User** | **System** |
| 5. Modifies details. |  |
|  | 6. Check if details exist already. |
| 7. Accepts new details. |  |
|  | 7. Saves changes. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Financial App |
| Functional requirement associated ID | R5.I.ii: Delete contact |
| Description | The user wishes to delete one of their contacts. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to delete an existent payee or payer.  System: delete the contact from the list. |
| Pre-conditions | There are contacts in the user’s list. |
| Post-conditions | The user can delete a contact. |
| Trigger | The user wishes to delete one of their contacts. |

### Main success scenario

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the contacts’ link. |  |
|  | 2. Opens a view displaying the contacts’ list. |
| 3. Selects the payee/payer. |  |
|  | 4. Opens a view displaying the contact’s details. |
| 5. Clicks the delete button. |  |
|  | 6. Deletes contact and saves changes. |
|  |  |
|  | End of use case. |

|  |  |
| --- | --- |
| Use Case ID | UC: Financial App |
| Functional requirement associated ID | R6: Logout |
| Description | The user wishes to logout from the software. |
| Level | Low Level Summary |
| Primary actor | User, System |
| Supporting Actors | - |
| Stakeholders and Interests | User: be able to logout from the software.  System: logout the user. |
| Pre-conditions | - |
| Post-conditions | The user is able to logout and go back to the login page. |
| Trigger | The user wishes to logout from his/her account. |

### Main success scenario

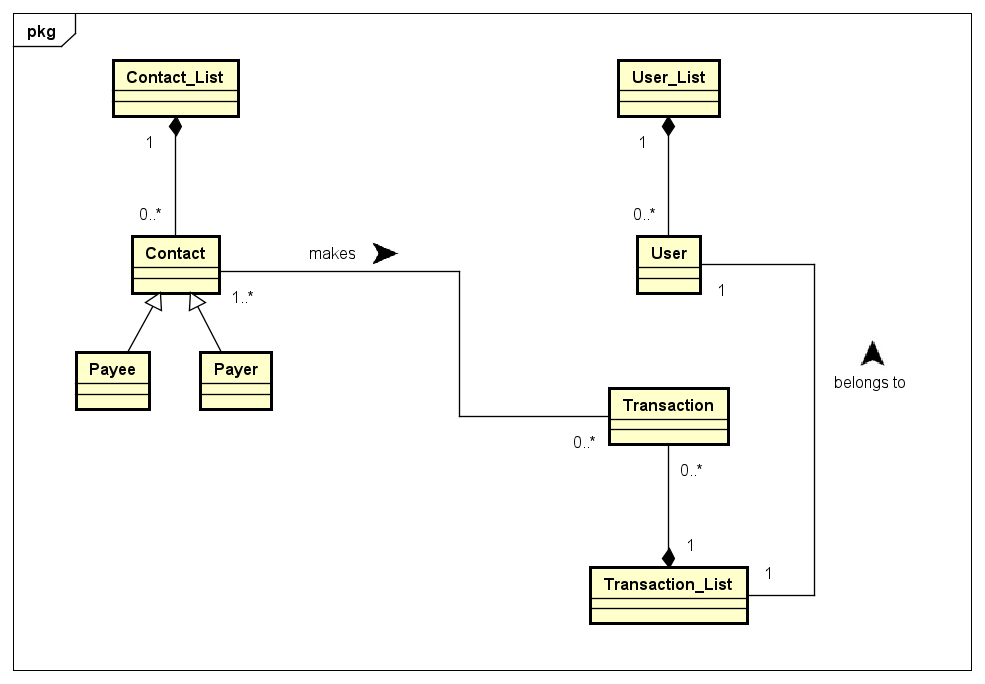
|  |  |
| --- | --- |
| **User** | **System** |
| 1. Clicks the logout link. |  |
|  | 2. Goes back to the login page. |
|  |  |
|  | End of use case. |

Part C – Classes

# crc

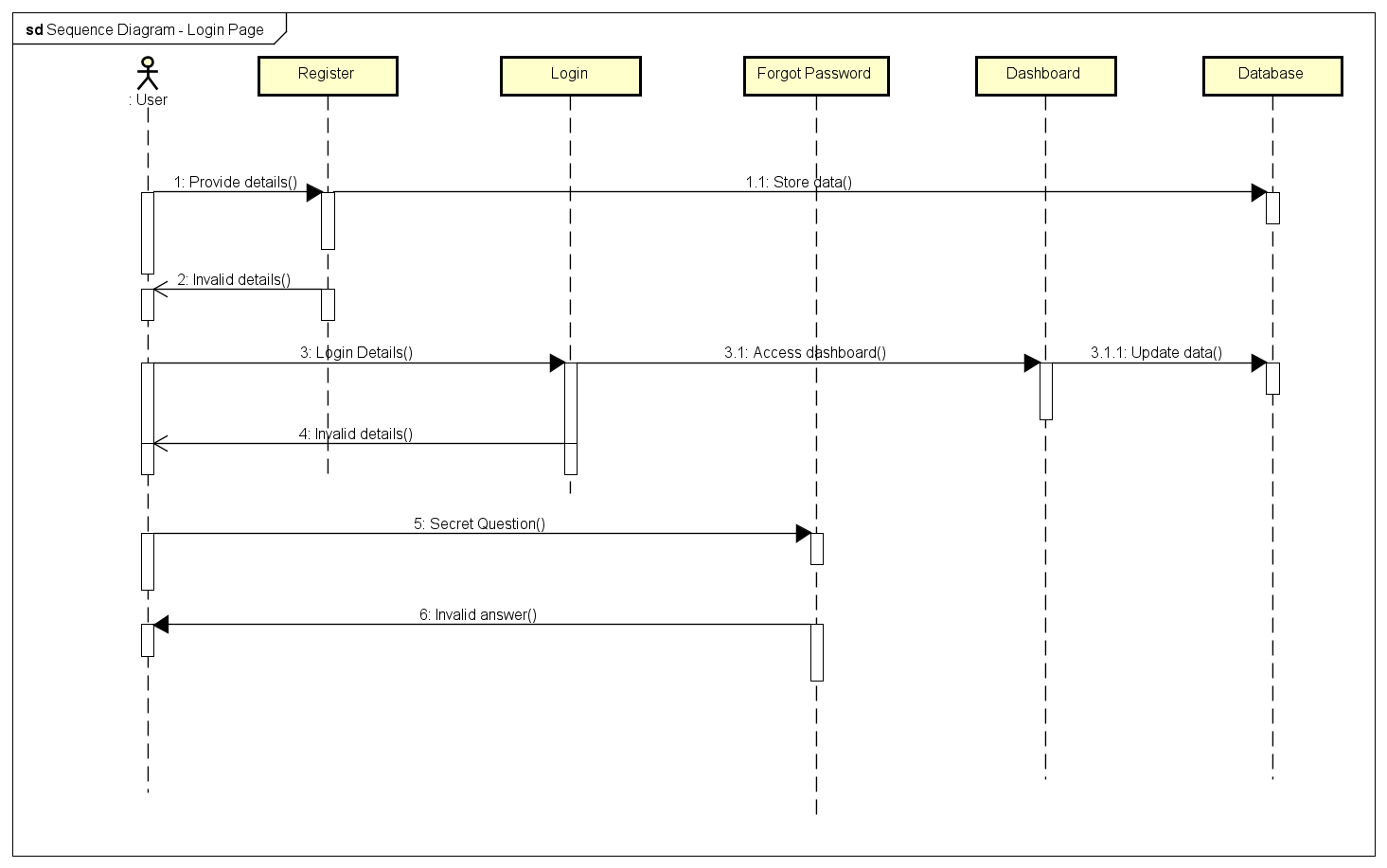
|  |  |  |  |
| --- | --- | --- | --- |
| Class Name | Type | Responsibility | Collaborations |
| Contact | Model | Holds details of contacts such as name, address, contact number, bank details |  |
| User | Model | Holds details of the user such as name, username, password, and secret question |  |
| Transaction | Model | Holds details of the transactions made by the user |  |
| Contact\_List | Model | Contains all the contacts in the user’s contact list | ContactView |
| User\_List | Model | Contains all the user registered with the software |  |
| Transaction\_List | Model | Contains all the transactions made by the user | TransactionView |
| Payee | Model | Separates the contact list into payees and payers | ModifyContactView |
| Payer | Model | Separates the contact list into payees and payers | ModifyContactView |
| ContactView | View | View for displaying the contact list | ContactViewController |
| TransactionView | View | View for displaying the transaction list | TransactionViewController |
| ModifyContactView | View [Form] | View contact details and modify | ModifyContactController |

## domain model

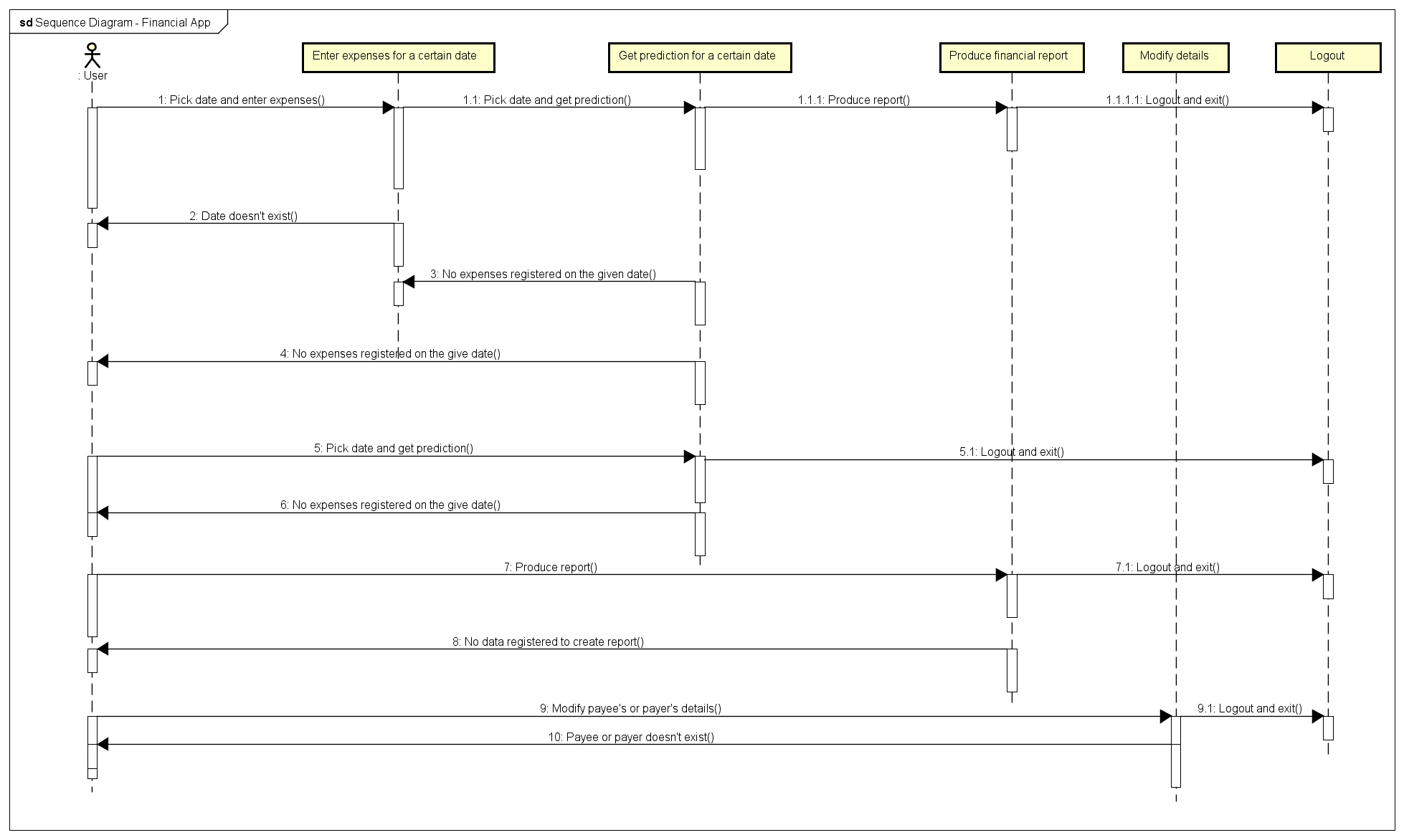


Class Diagram

Part D – Collaboration

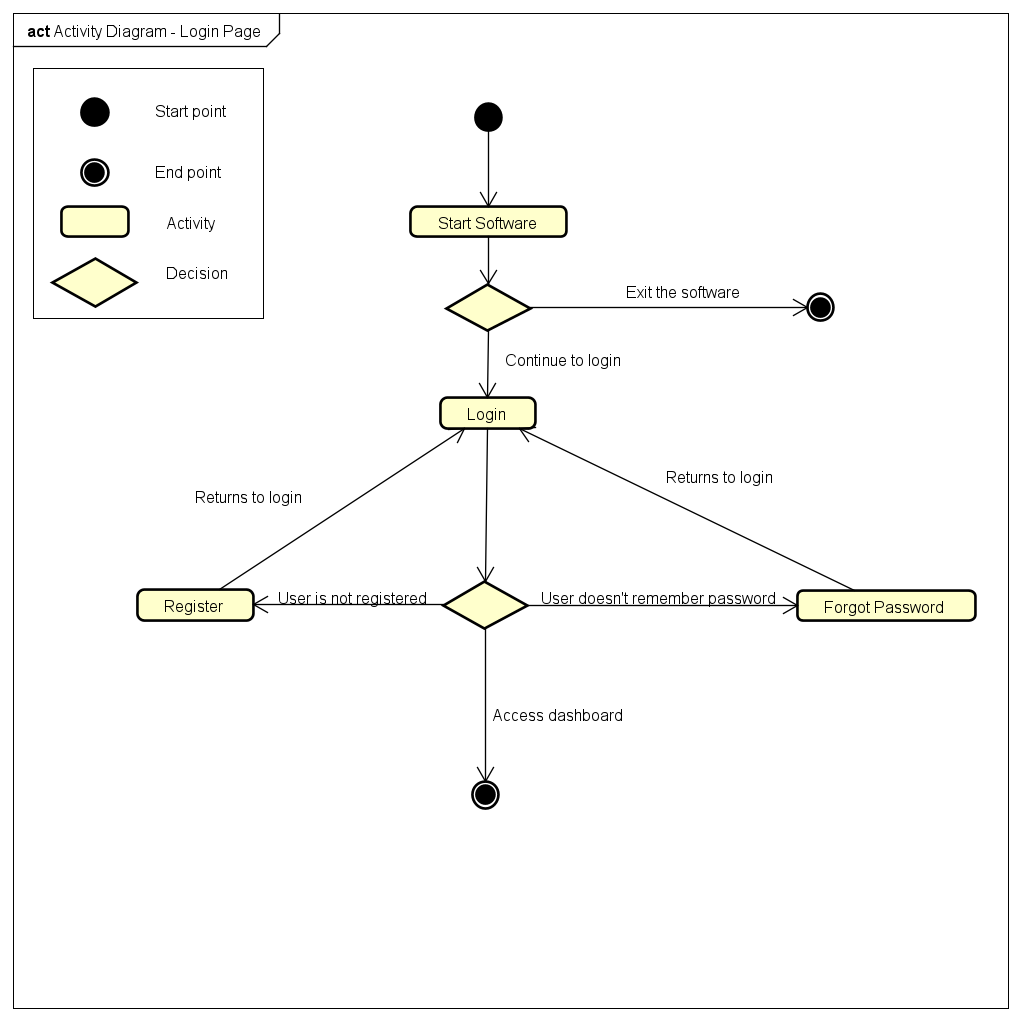


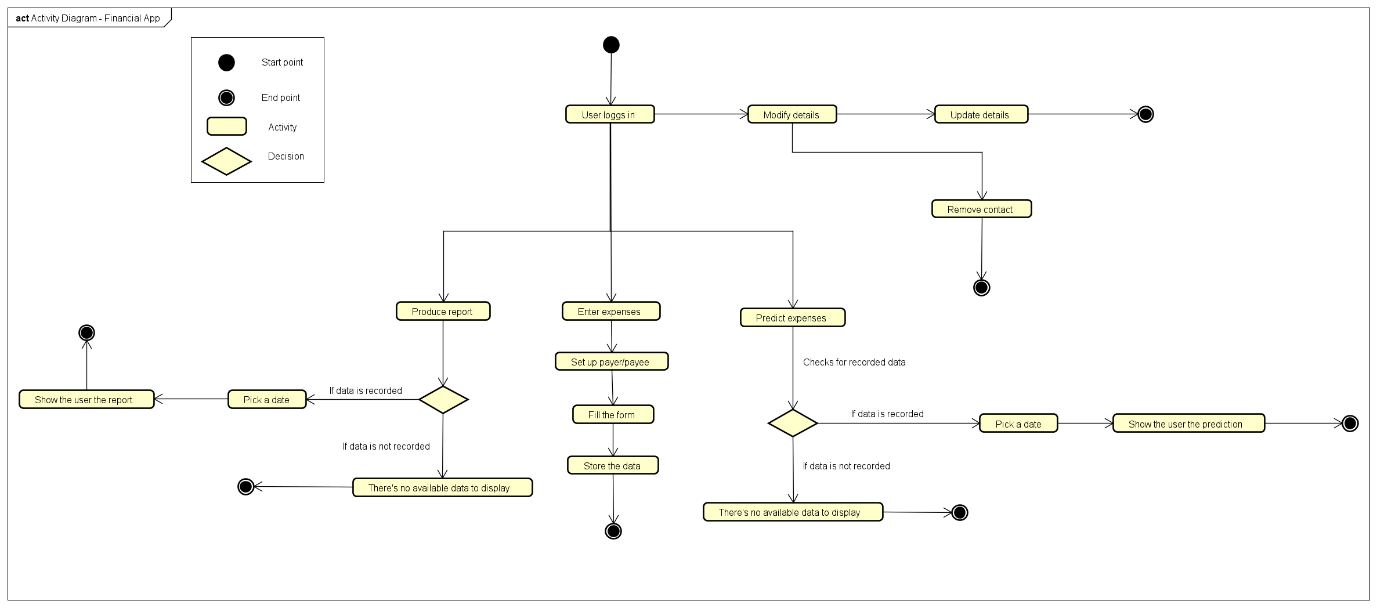
Sequence Diagram - Login Page



Sequence Diagram - Financial App

Part E – Activity



Activity Diagram - Login Page 

Activity Diagram - Financial App

Part F – Self assessment form and Report

# Self Assessment form



## Suitability

The design I have created for this software is meant to explain in an easy manner what the final product should and should not do. I have created Use Case, Sequence and Activity diagrams, one of each for the Login System of the software and one of each for the software itself: creating reports, prediction of expenses, modifying contact details and more. Although the diagrams could go in much more detail, I have chosen a simpler way that catches the essential of the software. The diagrams take in consideration errors and exceptions for a better implementation: if the user wants to obtain a financial report and there is no data recorded of any expenses or transactions on the date he/she picked, they will be redirected to the main dashboard of the software. I proposed as a Functional Requirement the idea of the user having a pre-set contact list so they would be able to start adding payees and payers the moment they login into their account if the account is newly created. I have also given the user the option to “Exit” the application and to “Logout” of the application, creating at the same time a nice touch to the software features. I have followed the MVC pattern to offer a more detailed explanation of the classes that are going to be used and how they interact with each view. The design lacks on the non-functional requirements part as I have only listed four of them. Unless the user is not registered or logged in, anyone can easily use the software if they are Windows users. The password recovery process is not really secure either, as long as you have the answer to the secret question you’ve created with the account you can reset your password easily without an email confirmation or two-step authentication of any sort. Creating an account is also not requesting for a name, date of birth or any personal details, except your email/username, password and a security question, which means the users are going to be anonymous and their accounts will be only collect transactions details. From a manager’s view this would be a straightforward and simple design with limitless ways of being improved and with a lot of growing potential. It would definitely make for a great Beta version of the software that with the right amount of time and skills could be built into an amazing product. The users would be properly registered to the database with crypted information and more difficult ways of account recovery, more features such as: report of set direct debits, display of yearly expenses and how the user could improve their saving methods, regular payments and adding more than one bank account and or cards. All together the design is suitable for the stage the application is expected to be delivered with plenty room for extra features and changes to create an even better and well organised software with as little as possible errors.