

Report Final Project

Nguyen Manh Tien 20184312

ITSSS | 25/12/2021

Appendix

1. Use case specification
2. Use case analysis
3. GUI design
4. Apply principle SOLID and pattern
5. Detail design
6. Use case specification

Use case “Rent bike”

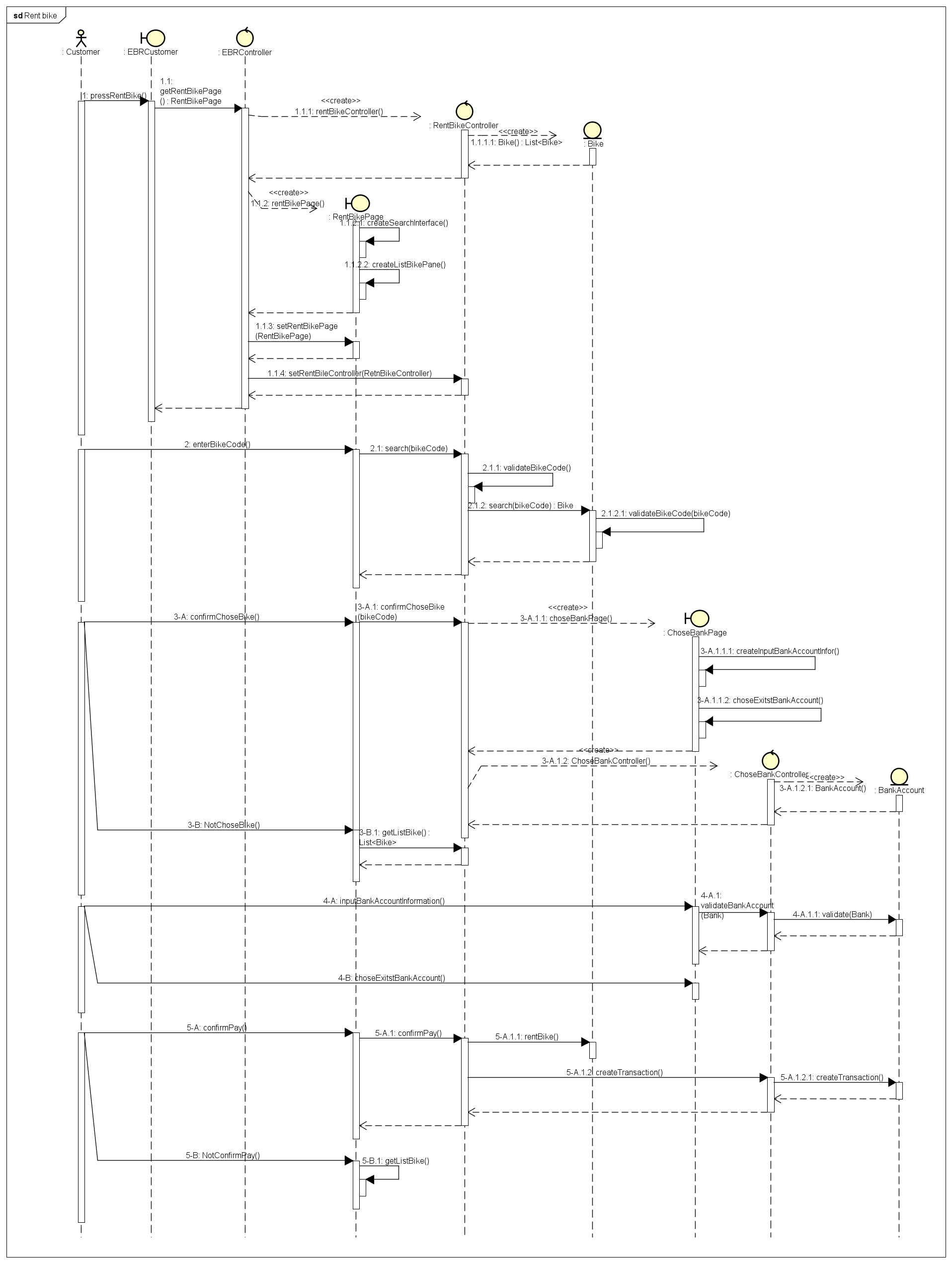
|  |  |  |  |
| --- | --- | --- | --- |
| **Use case Code** | UC001 | **Use case name** | rent bike |
| **Actor** | customer | | |
| **Precondition** | Login success | | |
| **Main flow of event**  **(success)** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | |  | Customer | Choose rent bike function | |  | System | Display rent bike function interface | |  | Customer | Input Bike Code | |  | Customer | Choose bike request | |  | System | Check bike code valid or not ? | |  | System | Display the bike information | |  | Customer | Confirm to choose the bike or not ? | |  | System | Display bill in detail | |  | Customer | Confirm to rent bike or not ? | |  | System | Display chose bank account interface | |  | Customer | Input bank account or choose existed account in the system | |  | System | Validate bank account | |  | Customer | Confirm pay or not ? | |  | System | Pay for service | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | | 5a | System | Error: Invalid bike code | | 7a | System | Display rent bike screen | | 9a | System | Display rent bike screen | | 12a | System | Error: Invalid bank account | | 13a | System | Display rent bike screen | |  | | | | | |
| **Post condition** | Save transaction, send email to customer | | |

\* Input data:

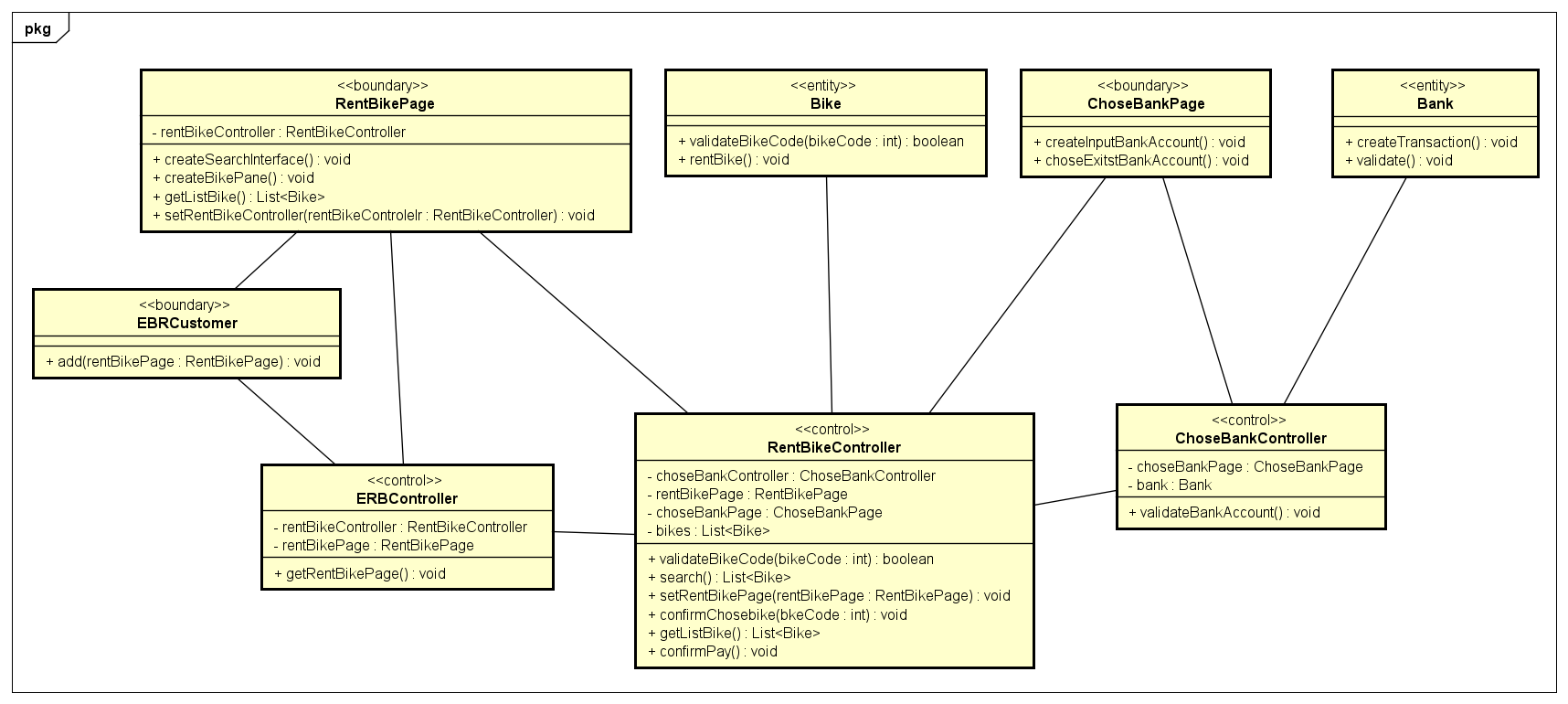
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field** | **Description** | **Required?** | **Valid condition** | **Example** |
|  | Bike code |  | Yes | Not null or empty | B12345 |
|  | Card number |  | yes | * Not null or empty * Contain 9 digit | 123456789 |
|  | Card holder name |  | yes | Not null or empty | Nguyen Van A |
|  | Issues Bank |  | yes | Not null or empty | Viettin Bank |
|  | Expiration date |  | yes | Not null or empty, date format | 20/02/2030 |
|  | Security code |  | yes | Not null or empty | 123456 |
|  | Transaction description |  | no |  |  |

1. Use case analysis

Sequence analysis diagram



Class analysis diagram



1. GUI design

# Standardizing the screen configuration

## Display

Resolution: 900 x 600 px

## Screen

Position of button: bottom (vertical) and center (horizontal) of frame.

Position of message: center of frame

Position of screen title: Title top-left of frame.

Numeric display consistency: commas to separate thousands, and strings consisting of only characters, digits, commas, periods, spaces, underscores, and hyphen symbols.

## Control

Size text: medium size (15px). Font: system UI. Color: #000000

Check input: check empty and format.

## Enter input from keyboard

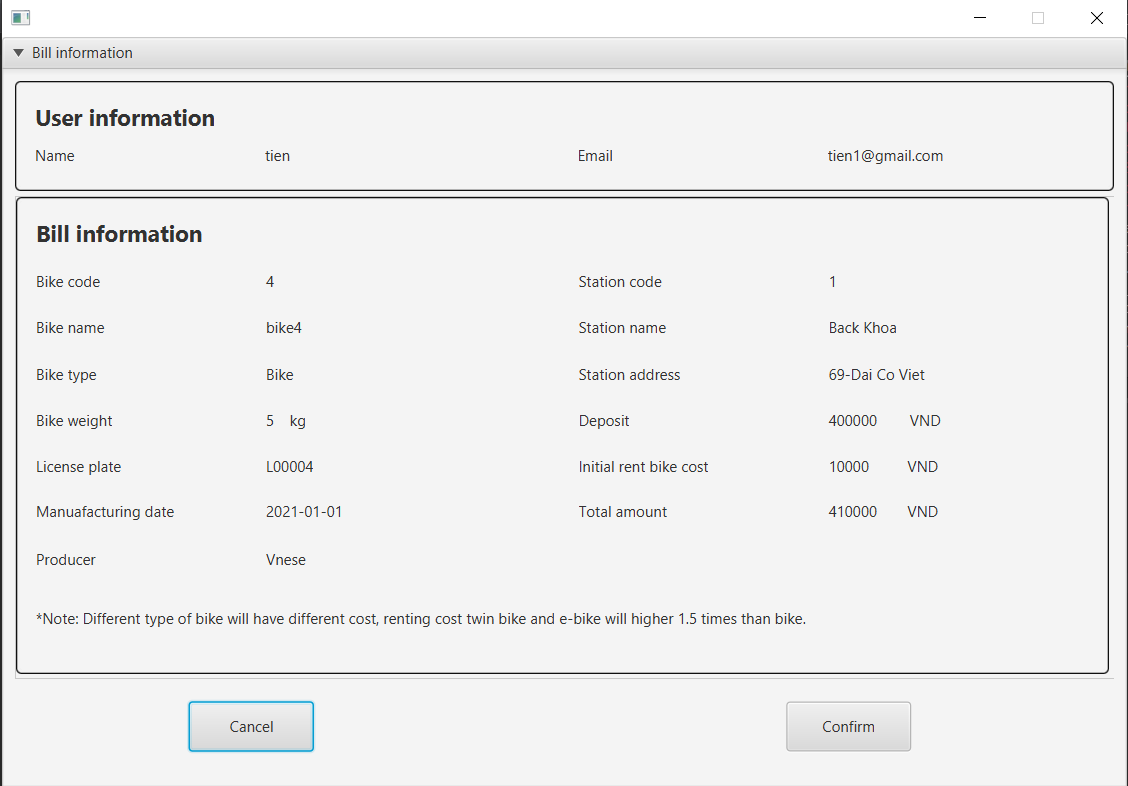
No keyboard shortcuts. Using button to return previous screen. Otherwise button “X” in top-right of window to close screen.

## Error

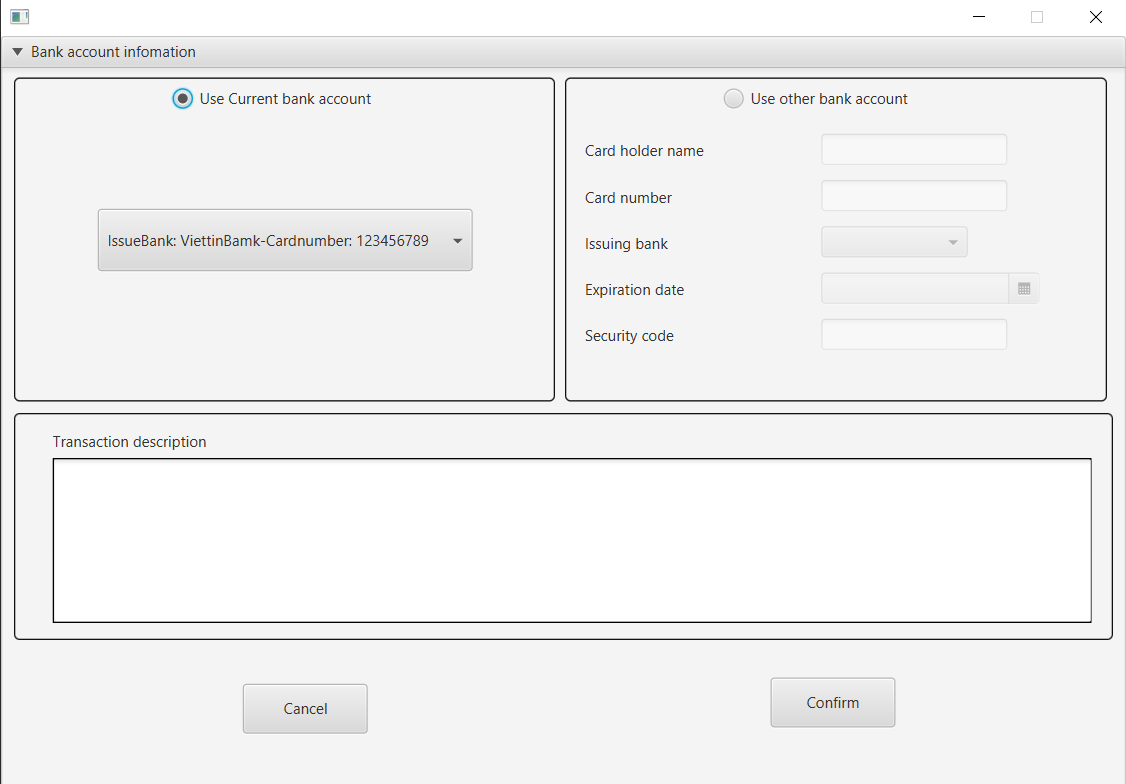
Show message by label warnning.

# Create screen images

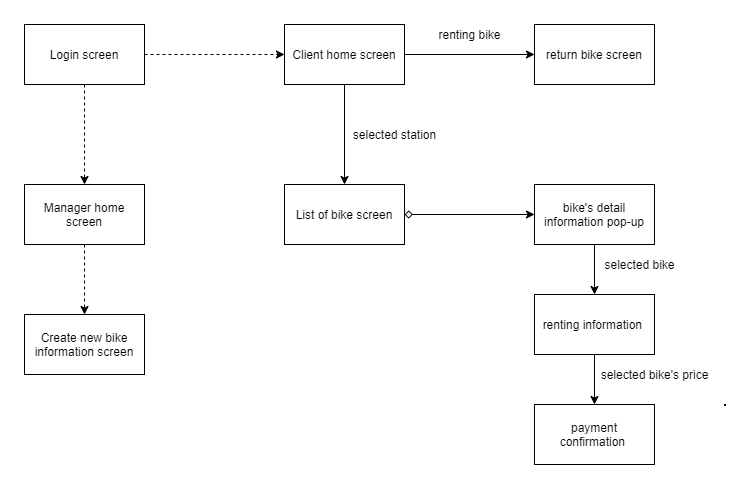
* Bill information screen



* Bank account information



# Create a screen transition diagram



# Creating screen specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software |  | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | View cart screen | 07/11/2021 |  |  | Nguyễn Mạnh Tiến |
|  |  | Control | Operation | Function |  |
| BillScreen | | Label | Initial | Display the bill information | |
| Cancel button | Click | Cancel rent bike and return list of bike screen | |
| Confirm button | Click | Confirm the bill and transition to next screen | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Label | Initial | Display the account information |
| Area text | Initial | Display the transaction description |
| Cancel button | Click | Cancel rent bike and return list of bike screen |
| Confirm button | Click | Confirm the account and pay |
| Radio Button | Click | Chose current bank account or new ones |
| ChoiceBox | Click | Chose bank account |

# Define field

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | View cart |  |  |  |
| Item name | Number of digits (bytes) | Type | Required | Remarks |
| Cardholder name | 100 | Char | true | Left-justified |
| Card number | 20 | Digit | true | Left- justified |
| Issue Bank |  | Char | true | Left-justified |
| Expiration Date |  | Date | true |  |
| Security Code |  | Char | true | Left-justified |

1. Apply principle SOLID and pattern
2. Principle SOLID

* Single-responsiblity principle: each class has only one responsibility, such as: billApi to connect, interact with database(insert, update, get data, delete), Bill model for hold data.
* Open-closed principle: we have class BaseApi, if in the future, there is new object want to connect database, it will create a new class inherit BaseApi. Therefor, it will be open to add new feature in new class and close for modify BaseApi.
* Liskov substitution principle
* Interface segregation principle
* Dependency Inversion Principle

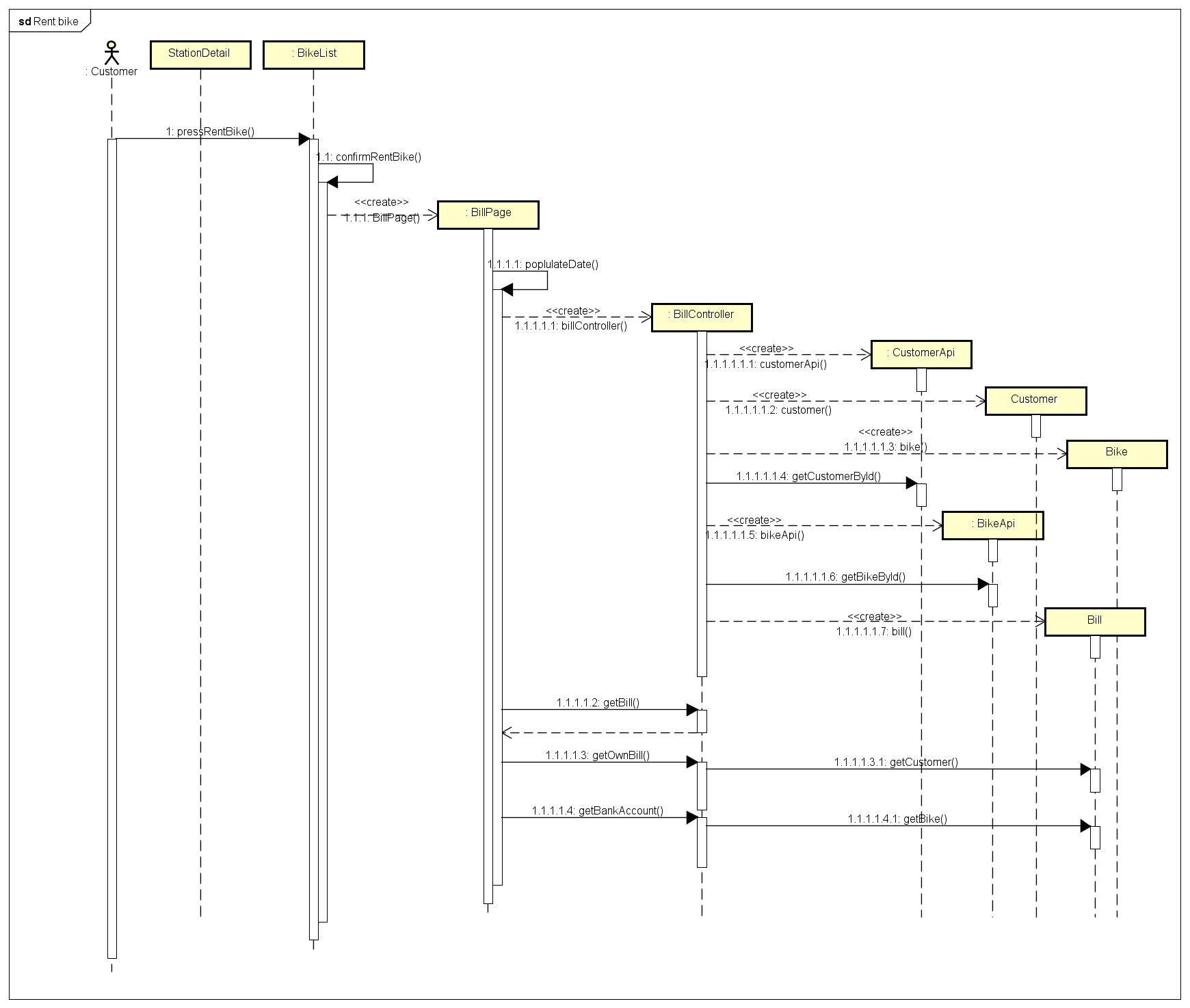
We using interface between subsystem and application. Each subsystem has own interface corresponding.

1. Pattern:

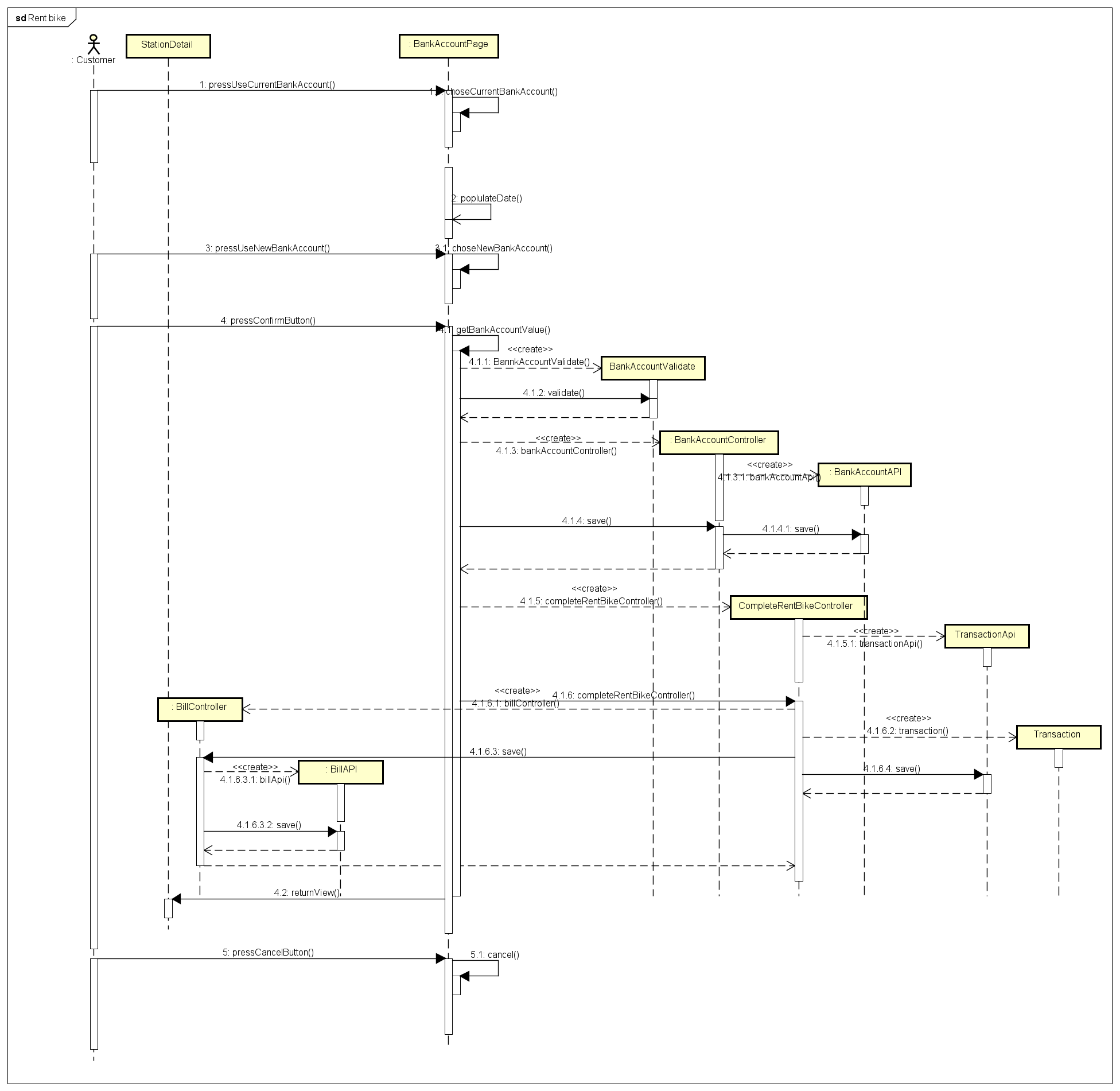
* MVC: For example: we have bill screen, billController, and billModel.
* Singleton pattern: init api
* Strategy pattern

1. Detail design

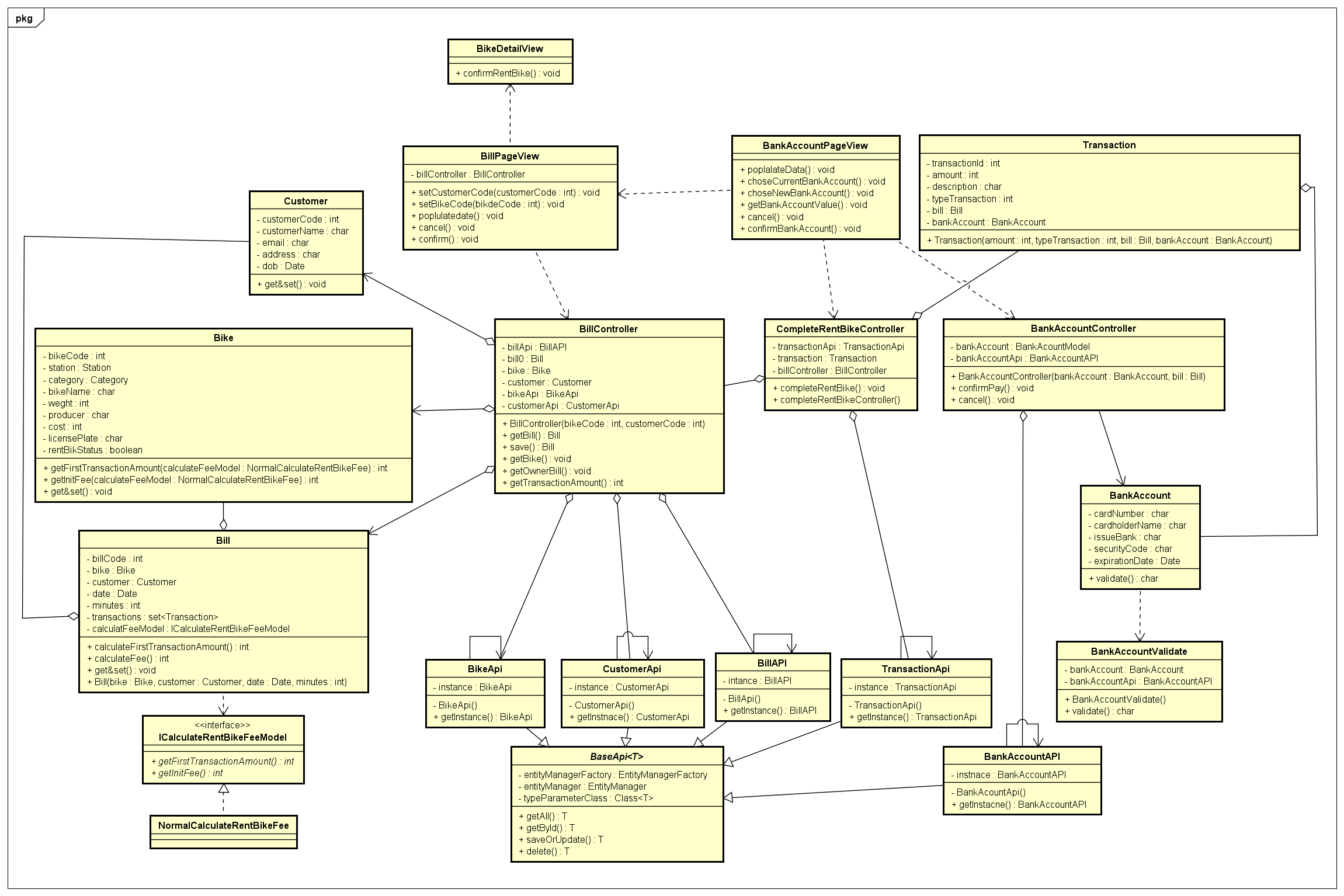
Sequence diagram: billScreen



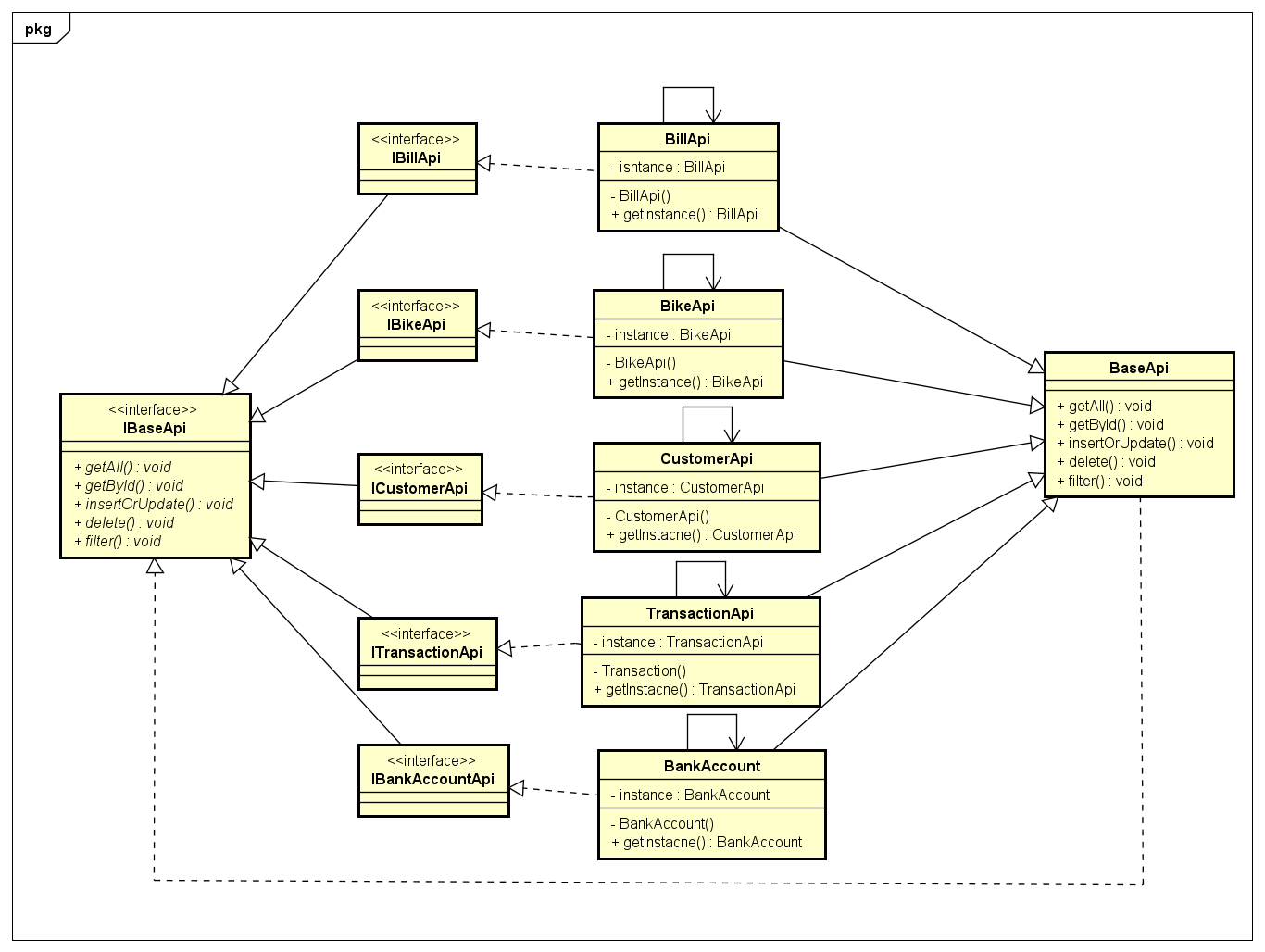
Sequence diagram: bankAccountScreen & complete rent bike



Class diagram:



Detail design subsystem



1. Unit test

In Test\_Usecase\_rentbike.doc and TestCase\_ValidateNewBankAcount.doc file