HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

PERSONAL REPORT

EcoBikeRental

ITSS Software Development

Group Number: 6

Member name: Lai Tien Duc

Member ID: 20176722

Assistant Lecturer: PhD. Trinh Tuan Dat

Hanoi, December, 2020

Table of Content

1. Introduction

1.1 Objective

The purpose of this document is to provide a description of the design of a usecase which i work with in projecta and fully enough to understand what is to be built and how it is expected to built.

1.2 Scope

This document provides a part of Software Design in the project.

2. Overall Description

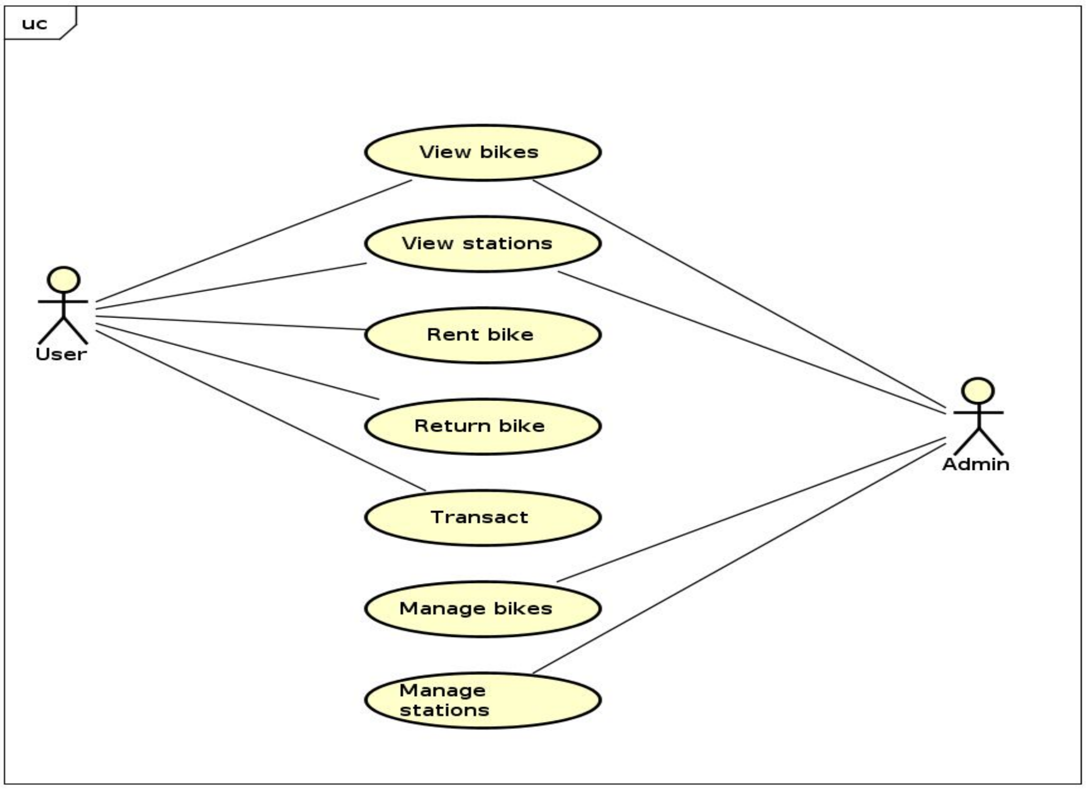
In the EcoBikeRental project, I am assigned use case “Rent/Return bike”.

I’m also responsible for these tasks:

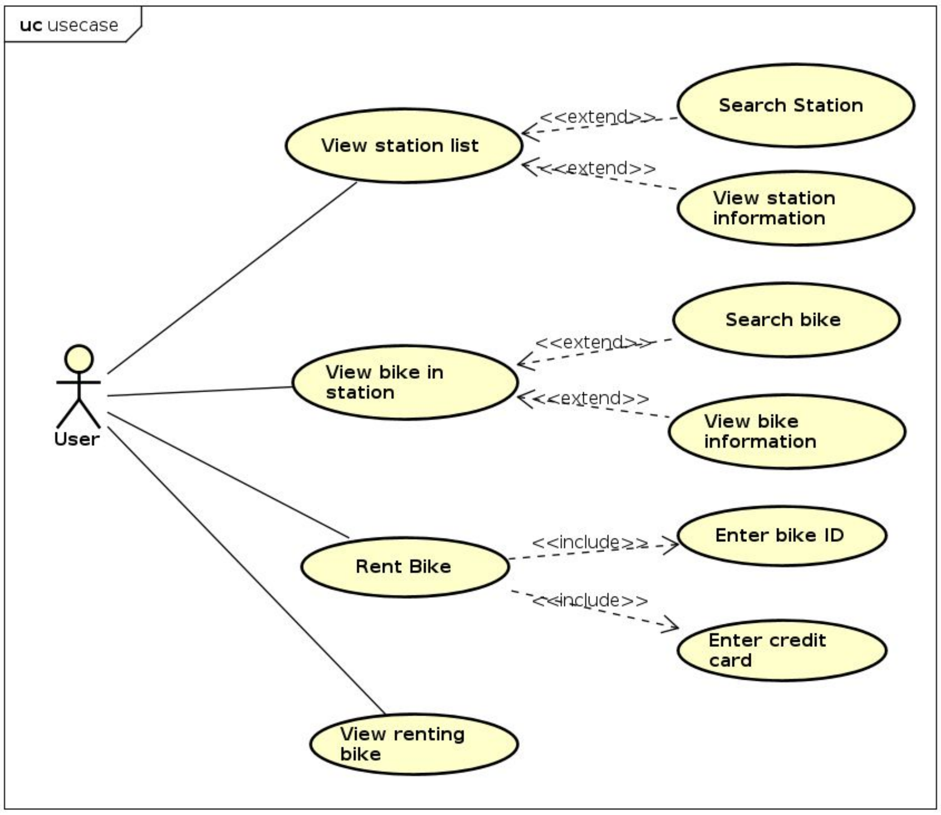
- User view all Station, Bike and detail of those
- Develop method for renting bike
- Develop method for returning bike

2.1. User-case diagram

2.1.1. General use-case diagram



2.1.2 Use-case diagram for "Rent Bike"



- Use-case specification

Use case ID	UC003	**Use case name**	Rent Bike
Actors	User, System		
Pre-Condition(s)	- User login the app - User has bank account		
Basic Path (Success)			

No	Proceeded by	Actions
1.	User	Login in the app
2.	System	Display station list screen
3.	User	Click rent button
4.	System	Check whether bike is renting or not
5.	System	Display rent bike screen
6.	User	Input bike id
7.	System	Chekc bike id
8.	User	Input credit card
9.	System	Check credit card
10.	User	Click rent button
11.	System	Record information of user, bike, account

<table>
<tr><td><b>Alternative Paths</b></td><td>

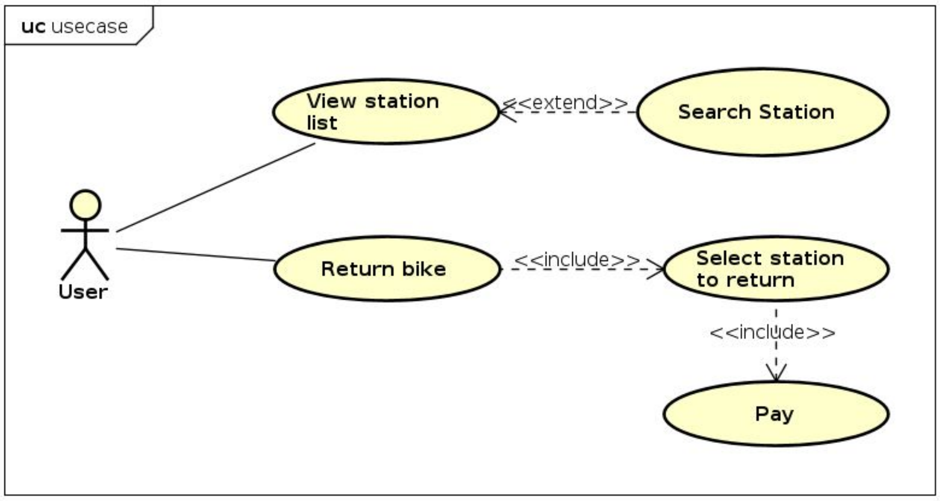
No	Proceeded by	Actions
3a	User	Click view station detatil
3a1	System	Display station detail screen
3a2	User	Click view bike list
3a3	System	Display list bike in station screen
3a4	User	Click view bike detatil
3a5	System	Display bike detail screen
3a6	User	Click rent bike
7a	System	Display error : invalid bike id
9a	System	- IF card_number is not exist , display error - IF balance is not enough, display error

</td></tr>
<tr><td><b>Post-Condition(s)</b></td><td>- User rented bike successfully
- System records the information of user, bike, bank account</td></tr>
</table>

* Input data of renting information includes these following fields:

ID	Data field	Description	Mandatory	Valid condition	Example
1	bike id		Yes	bike id exist	12
2	credit card		Yes	card exists, has enough money	

2.2.3 Use-case diagram for “Return bike”



- Use-case specification

Use case ID	UC004	Use case name	Return Bike
Actors	User, System		
Pre-Condition(s)	- User rented a bike		
Basic Path (Success)			

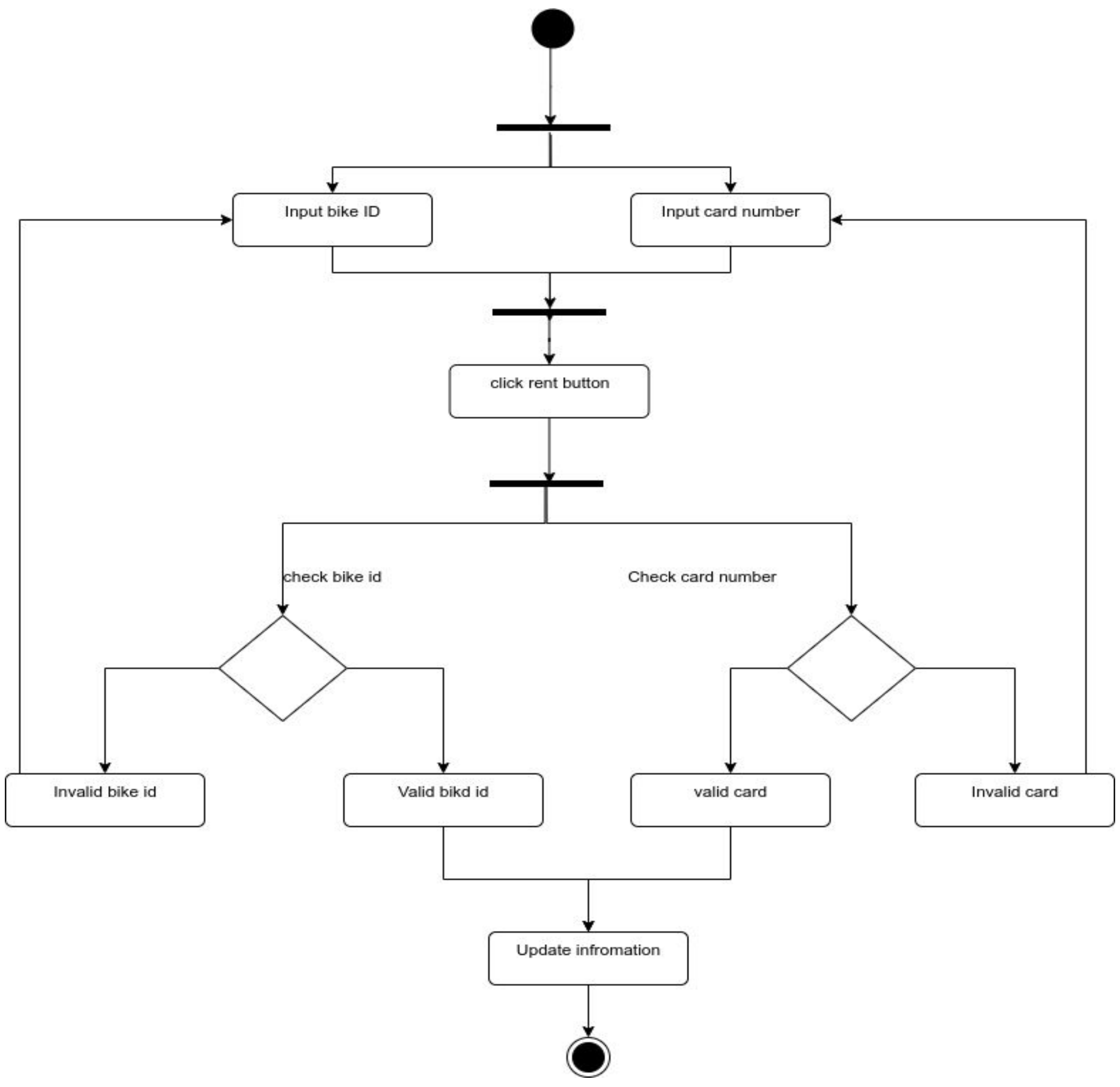
No	Proceeded by	Actions
1.	User	Click return button
2.	System	Display station list screen
3.	User	Select station to return
4.	System	Check empty dock in station
5.	User	Click payment button
6.	System	Display payment screen
7.	User	Click pay button
8.	System	Update record
9.	System	Refund money
10.	System	Show success payment

Alternative Paths	

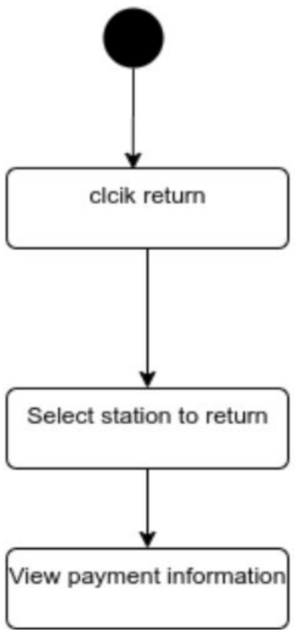
No	**Proceeded by**	**Actions**
4a	System	IF station doesn's have empty dock, display error: can not returnn to this station

Post-Condition(s)	- User return bike successfully - System records the information of user, bike, bank account - User's account bank debited

2.2. Activity diagram for "Rent bike"



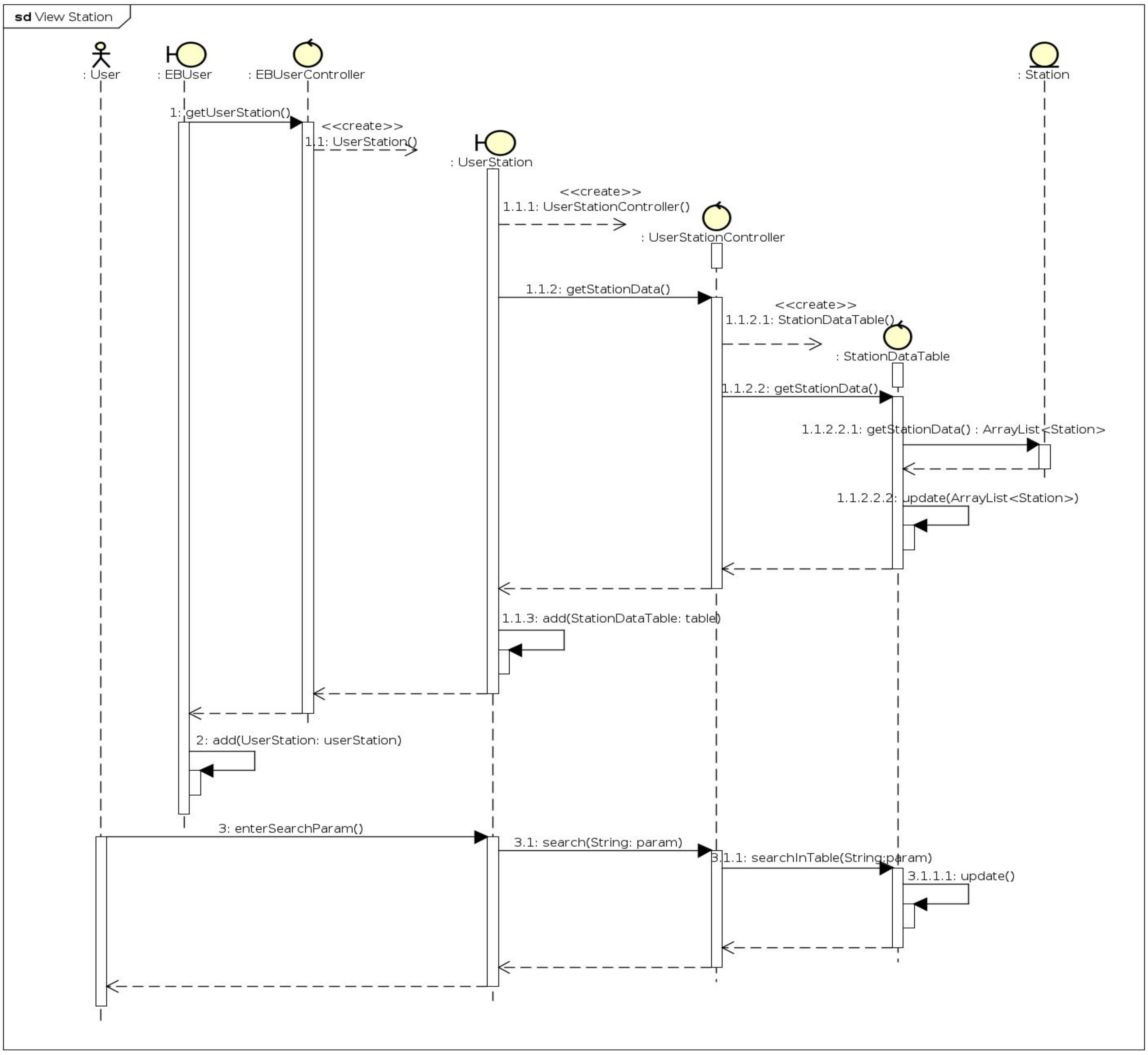
2.3. Activity diagram for “Return bike”



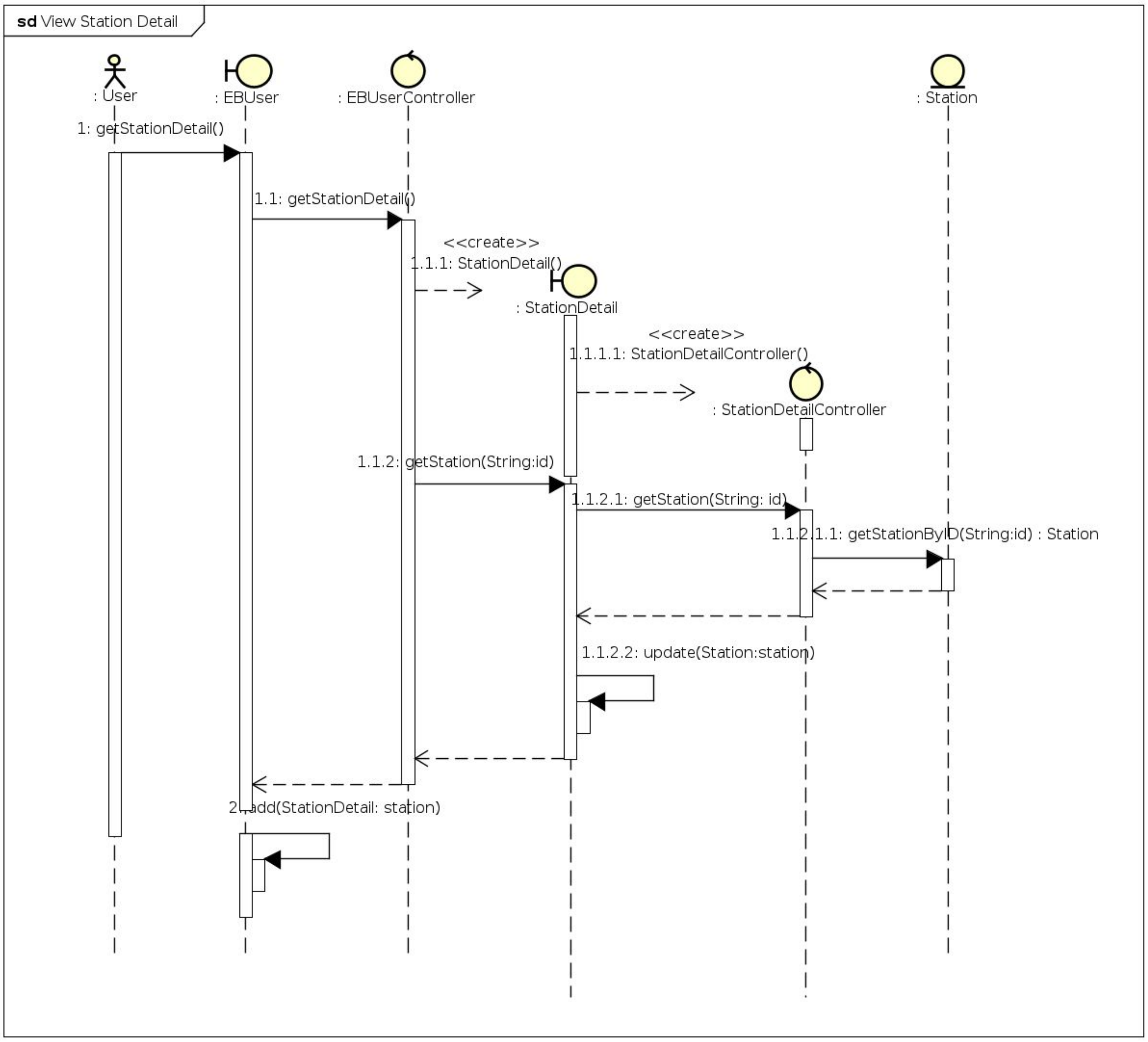
3. Detail design

3.1 Sequence diagram

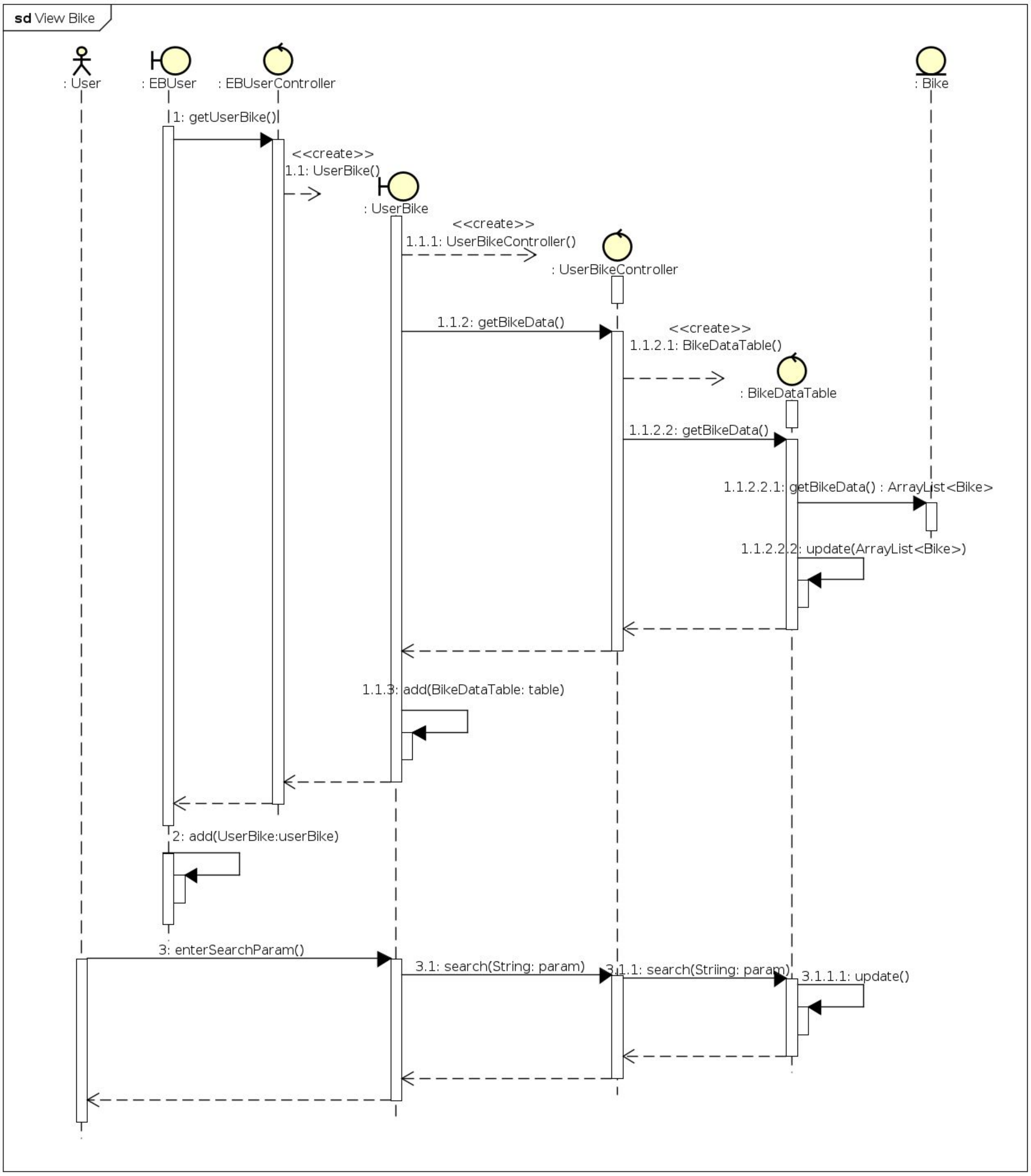
3.1.1 Sequence diagram for “View Station”


sd View Station
: User
: EBUser
: EBUserController
: UserStation
: UserStationController
: StationDataTable
: Station
1: getUserStation()
<<create>>
1.1: UserStation()
<<create>>
1.1.1: UserStationController()
1.1.2: getStationData()
<<create>>
1.1.2.1: StationDataTable()
1.1.2.2: getStationData()
1.1.2.2.1: getStationData() : ArrayList<Station>
1.1.2.2.2: update(ArrayList<Station>)
1.1.3: add(StationDataTable: table)
2: add(UserStation: userStation)
3: enterSearchParam()
3.1: search(String: param)
3.1.1: searchInTable(String:param)
3.1.1.1: update()

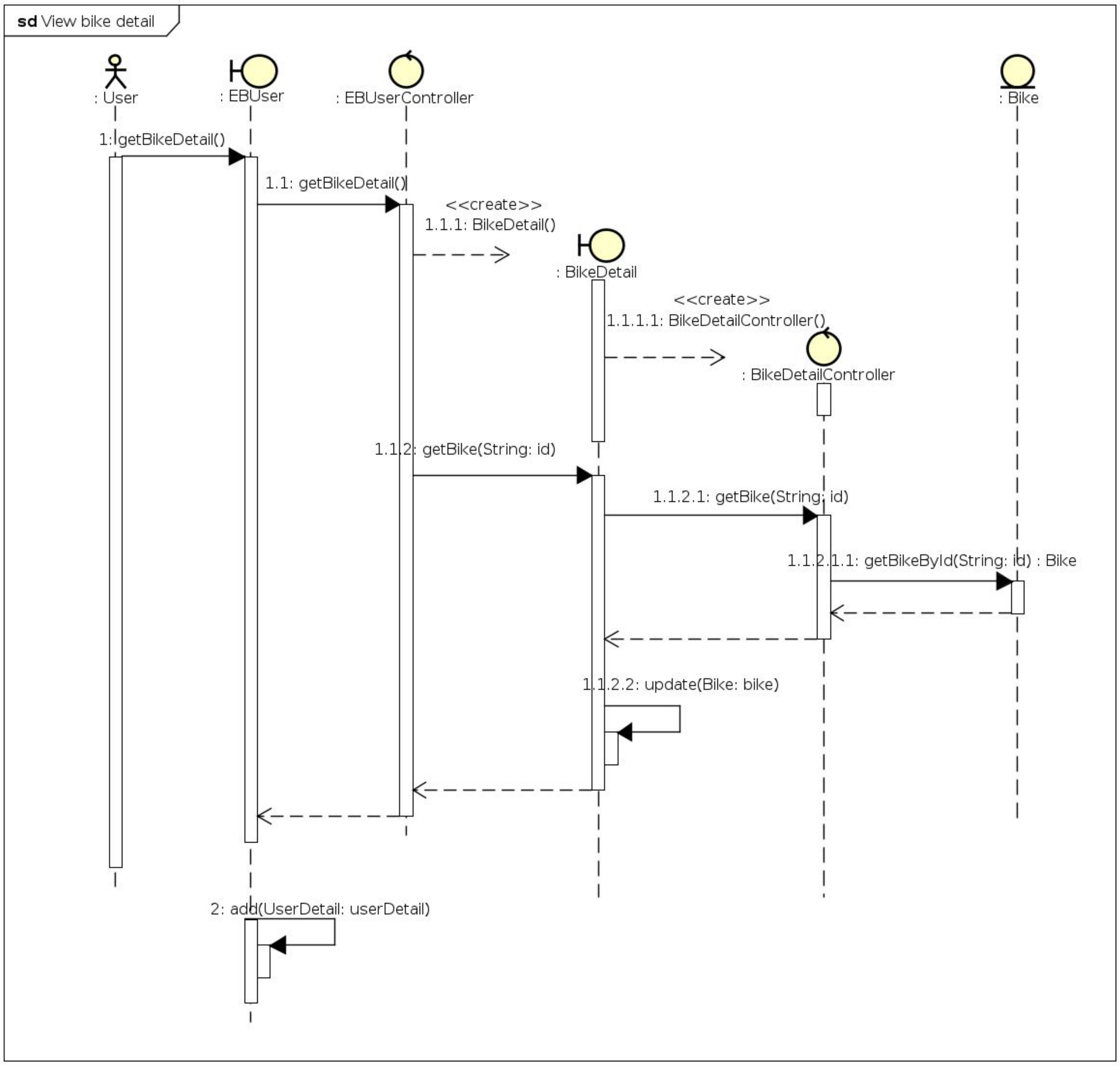
3.1.2. Sequence diagram for “view Station detail”


sd View Station Detail
: User
: EBUser
: EBUserController
: Station
1: getStationDetail()
1.1: getStationDetail()
<<create>>
1.1.1: StationDetail()
: StationDetail
<<create>>
1.1.1.1: StationDetailController()
: StationDetailController
1.1.2: getStation(String:id)
1.1.2.1: getStation(String: id)
1.1.2.1.1: getStationByID(String:id) : Station
1.1.2.2: update(Station:station)
2: add(StationDetail: station)

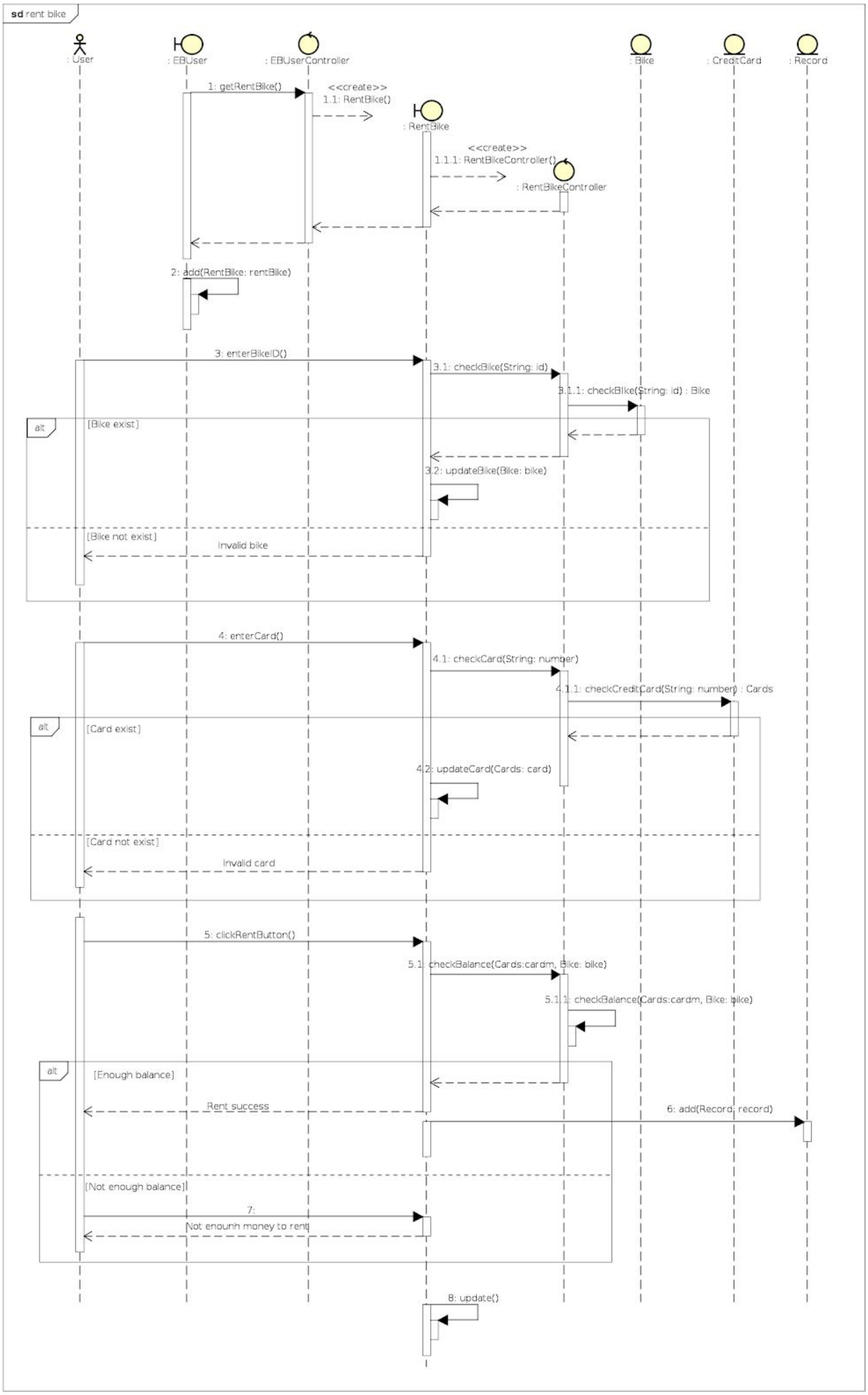
3.1.3. Sequence diagram “View Bike in Station”



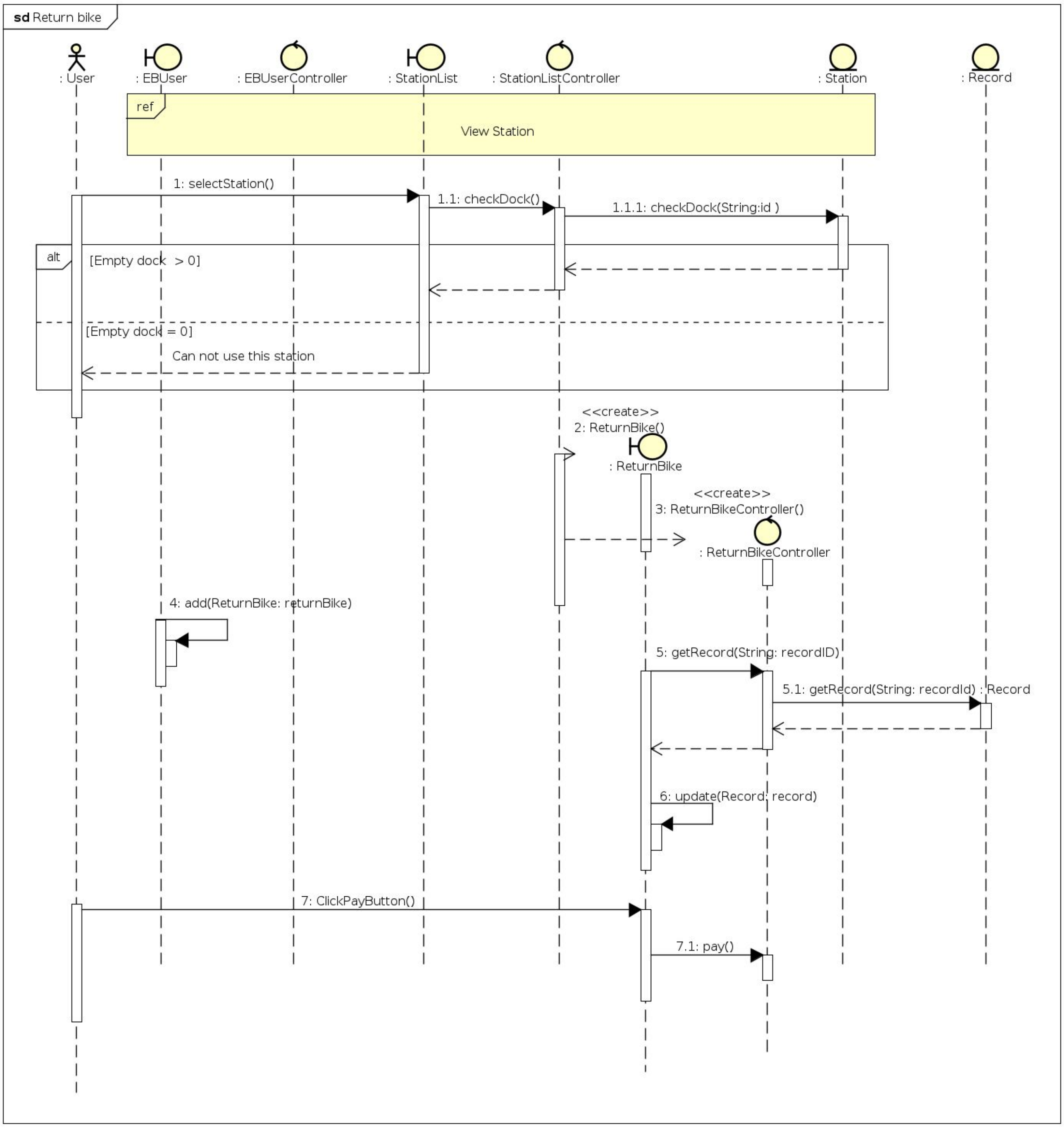
3.1.4. Sequence diagram for “View bike detail”


sd View bike detail
: User
: EBUser
: EBUserController
: Bike
1: getBikeDetail()
1.1: getBikeDetail()
<<create>>
1.1.1: BikeDetail()
: BikeDetail
<<create>>
1.1.1.1: BikeDetailController()
: BikeDetailController
1.1.2: getBike(String: id)
1.1.2.1: getBike(String: id)
1.1.2.1.1: getBikeById(String: id) : Bike
1.1.2.2: update(Bike: bike)
2: add(UserDetail: userDetail)

3.1.5. Sequence diagram for “Rent Bike”

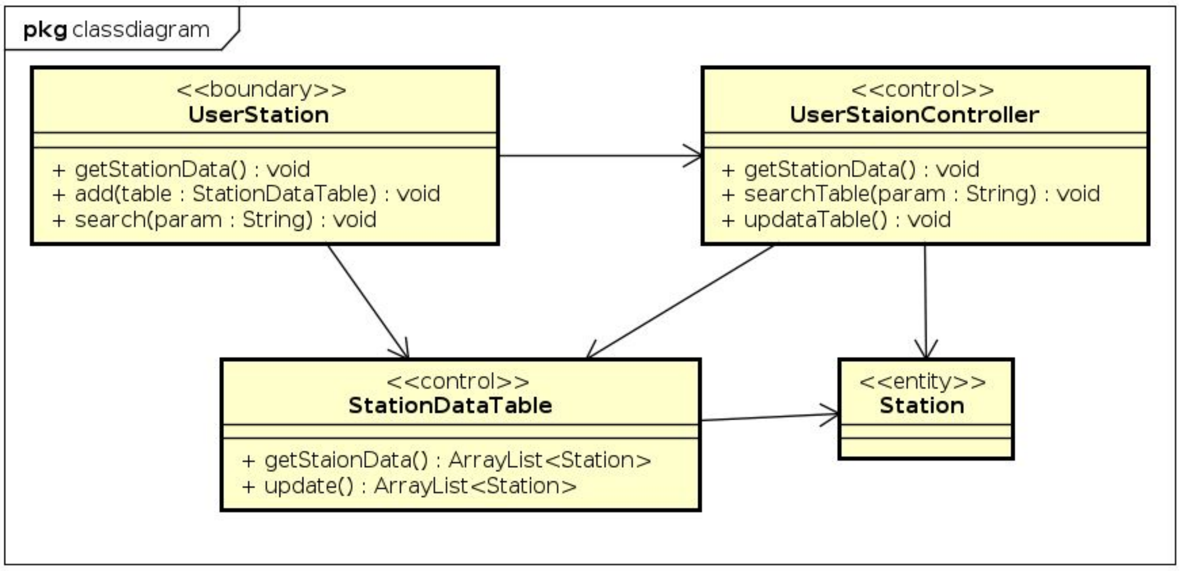

sd rent bike
: User
: EBUser
: EBUserController
: Bike
: CreditCard
: Record
1: getRentBike()
<<create>>
1.1: RentBike()
: RentBike
<<create>>
1.1.1: RentBikeController()
: RentBikeController
2: add(RentBike: rentBike)
3: enterBikeID()
3.1: checkBike(String: id)
3.1.1: checkBIke(String: id) : Bike
alt
[Bike exist]
3.2: updateBike(Bike: bike)
[Bike not exist]
Invalid bike
4: enterCard()
4.1: checkCard(String: number)
4.1.1: checkCreditCard(String: number) : Cards
alt
[Card exist]
4.2: updateCard(Cards: card)
[Card not exist]
Invalid card
5: clickRentButton()
5.1: checkBalance(Cards:cardm, Bike: bike)
5.1.1: checkBalance(Cards:cardm, Bike: bike)
alt
[Enough balance]
Rent success
6: add(Record: record)
[Not enough balance]
7:
Not enounh money to rent
8: update()

3.1.6. Sequence diagram for “Return Bike”

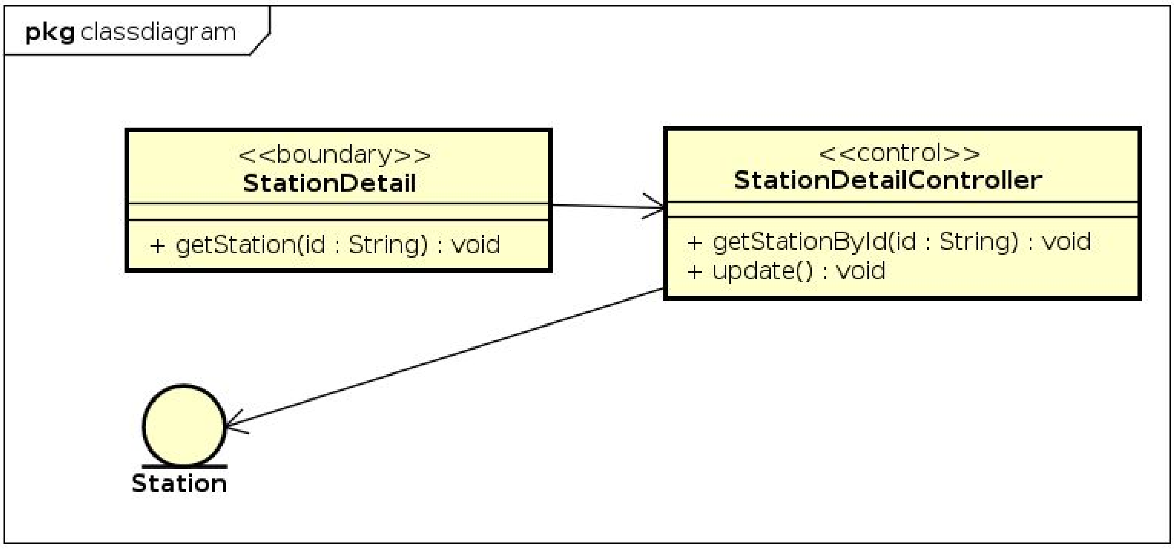


3.2. Analysis Class diagram

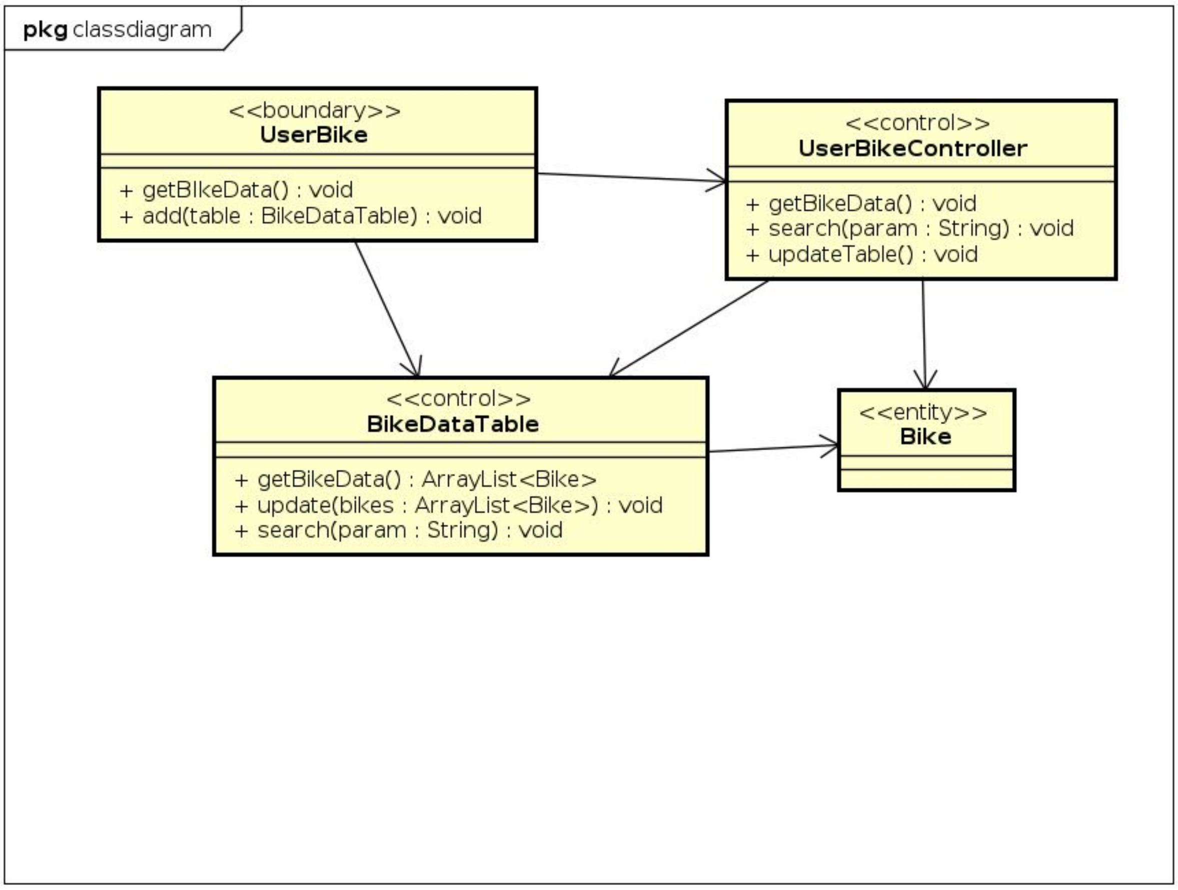
3.2.1. Analysis Class diagram for “View Station”


pkg classdiagram
<<boundary>>
UserStation
+ getStationData() : void
+ add(table : StationDataTable) : void
+ search(param : String) : void
<<control>>
UserStaionController
+ getStationData() : void
+ searchTable(param : String) : void
+ updataTable() : void
<<control>>
StationDataTable
+ getStaionData() : ArrayList<Station>
+ update() : ArrayList<Station>
<<entity>>
Station

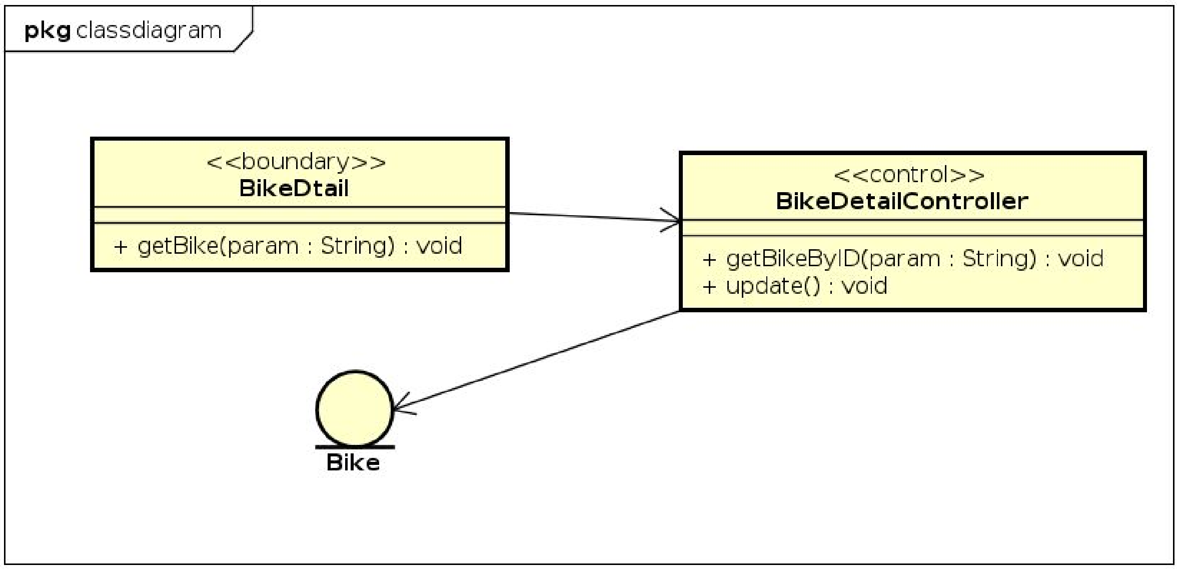
3.2.2. Analysis Class diagram for “View Station Detail”



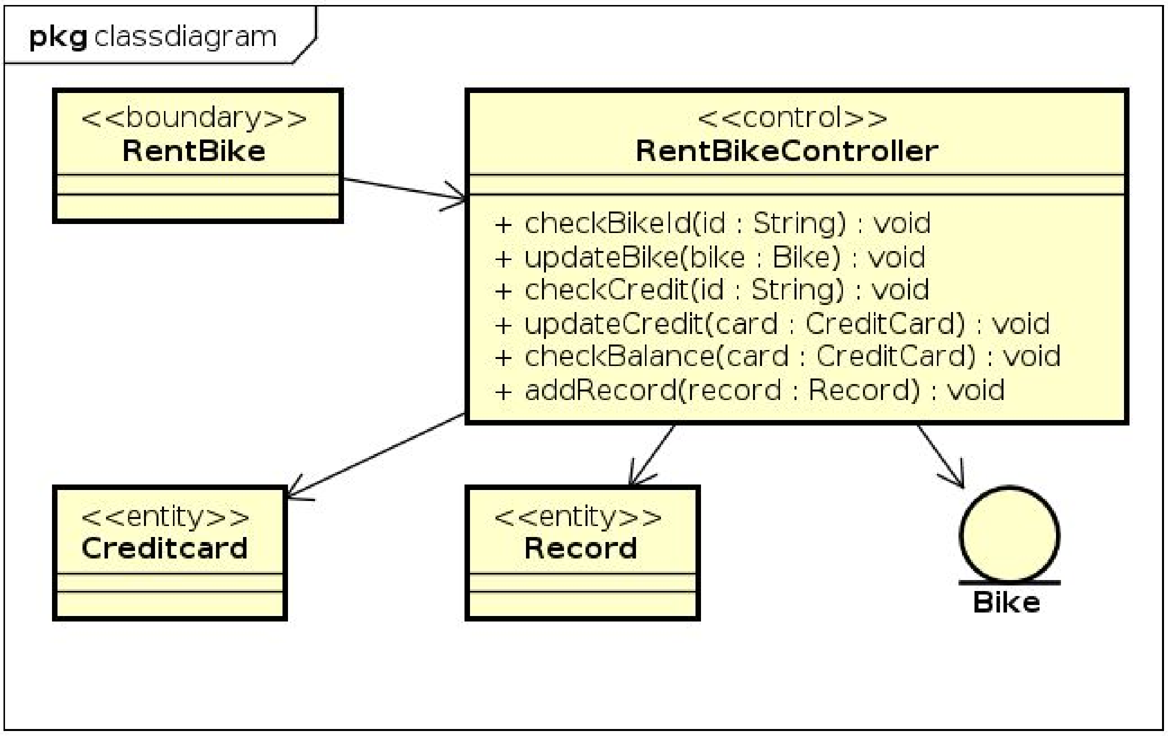
3.2.3. Analysis Class diagram for “View Bike”


pkg classdiagram
<<boundary>>
UserBike
+ getBIkeData() : void
+ add(table : BikeDataTable) : void
<<control>>
UserBikeController
+ getBikeData() : void
+ search(param : String) : void
+ updateTable() : void
<<control>>
BikeDataTable
+ getBikeData() : ArrayList<Bike>
+ update(bikes : ArrayList<Bike>) : void
+ search(param : String) : void
<<entity>>
Bike

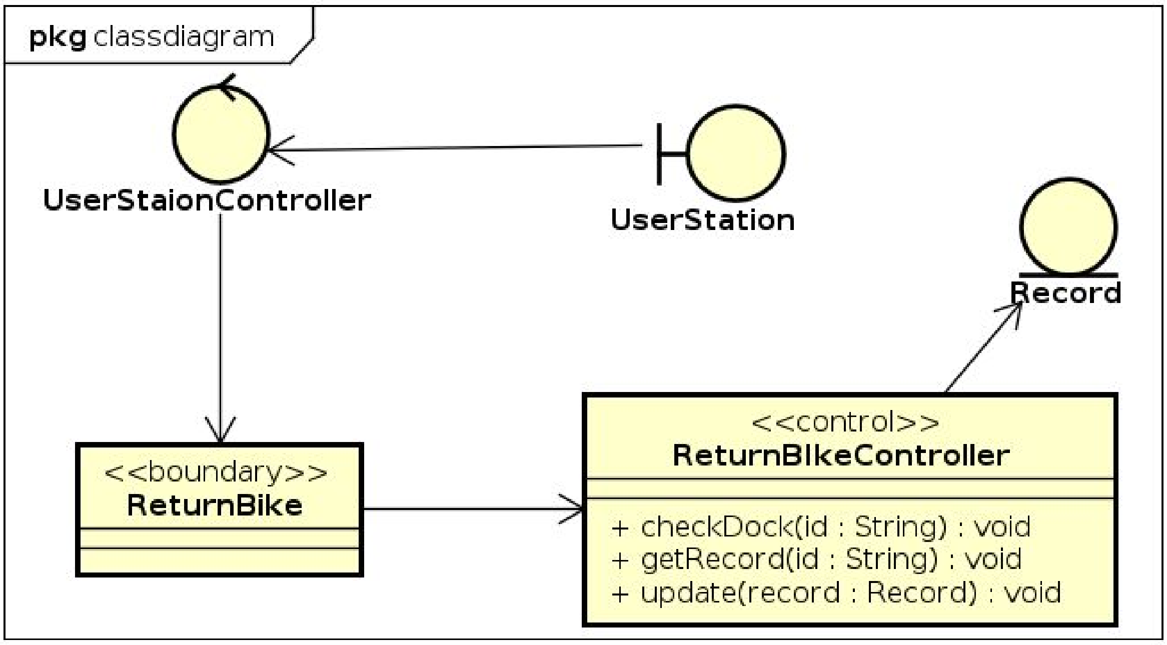
3.2.4. Analysis Class diagram for “View Bike Detail”



3.2.5. Analysis Class diagram for “Rent bike”

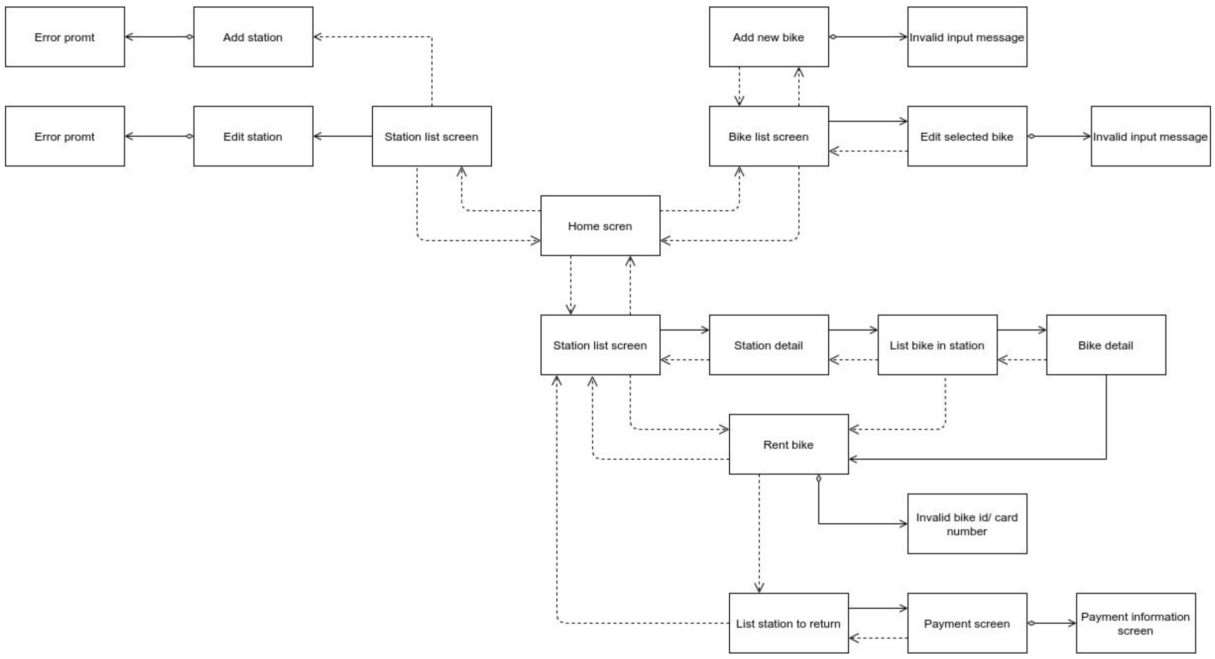

pkg classdiagram
<<boundary>>
RentBike
<<control>>
RentBikeController
+ checkBikeId(id : String) : void
+ updateBike(bike : Bike) : void
+ checkCredit(id : String) : void
+ updateCredit(card : CreditCard) : void
+ checkBalance(card : CreditCard) : void
+ addRecord(record : Record) : void
<<entity>>
Creditcard
<<entity>>
Record
Bike

3.2.6. Analysis Class diagram for “Return bike”

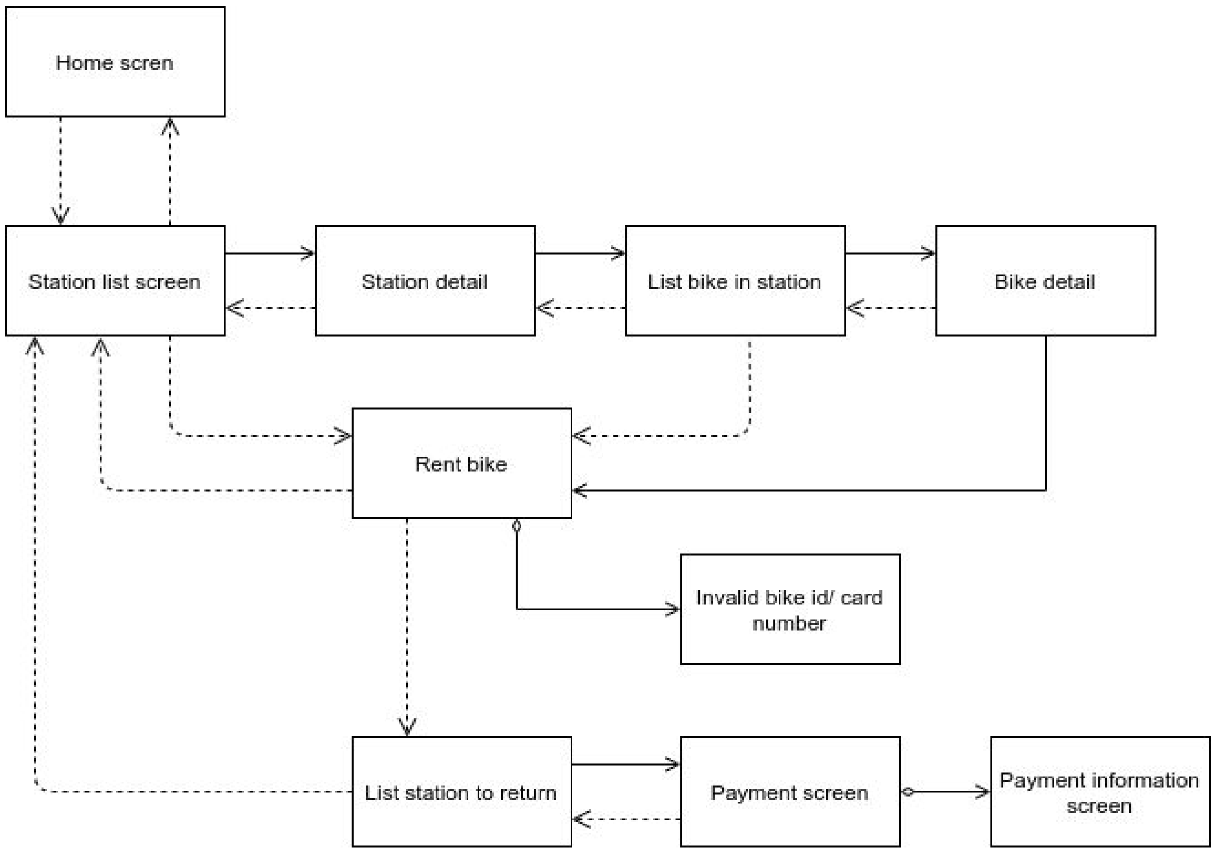


3.3. Screen Transition

- Overall transition

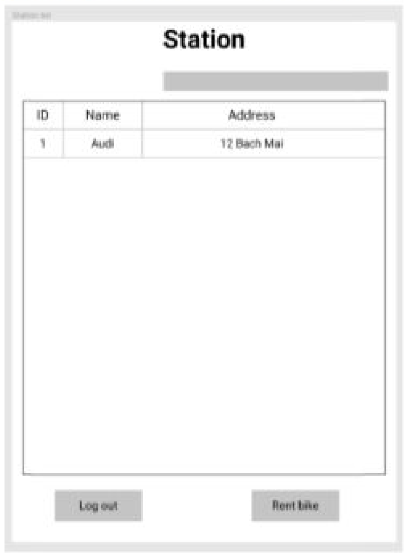


- Rent/Return transition

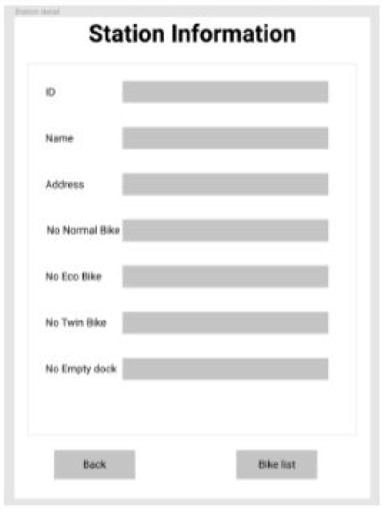


3.4. Screen specification

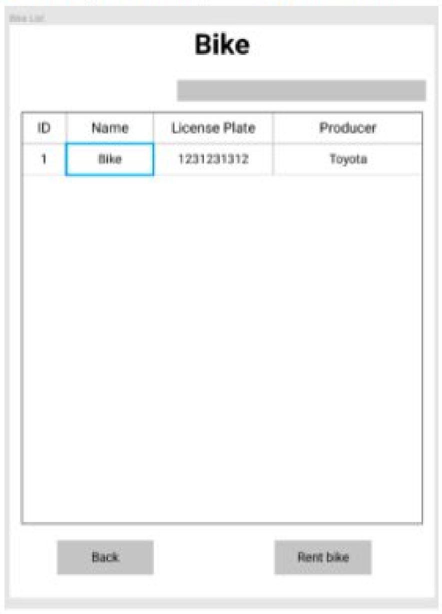
3.4.1.Station list

EcoBike Rental		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen sprcification	Station screen	10/12/2020			Lại Tiến Đức
 Station ID Name Address 1 Audi 12 Bach Mai Log out Rent bike		**Control**	**Operation**	**Function**	
		search text field	initial	search station	
		Area for displaying station	initial	display all station	
		row in table	click	display station detail	
		Logout button	click	logout of app	
		Rent bike button	click	Display rent bike screen	

3.4.2. Station detail GUI

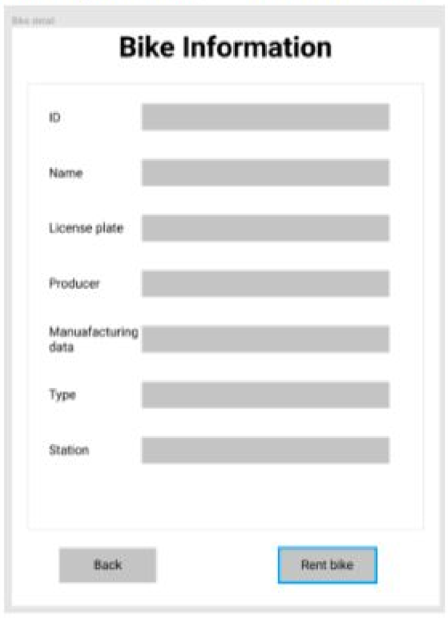
EcoBike Rental		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen sprcification	station detail screen	10/12/2020			Lại Tiến Đức
 Station Information ID Name Address No Normal Bike No Eco Bike No Twin Bike No Empty dock Back Bike list		**Control**	**Operation**	**Function**	
		Area for displaying station information	initial	display all information of station, and disable;	
		back button	click	back to station list screen	
		bike list button	click	Display all bike in station screen	

3.4.3. Bike list GUI

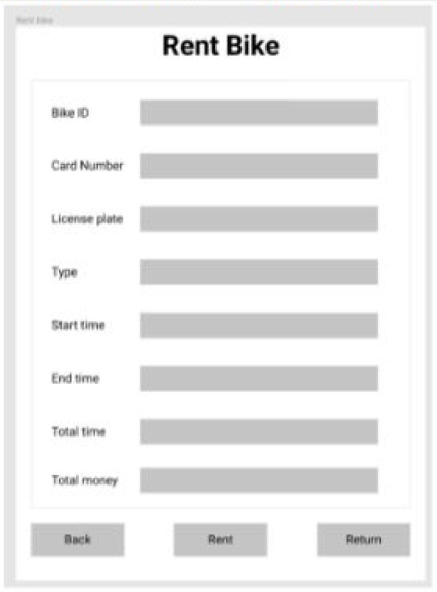
EcoBike Rental		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen sprcification	bike screen	10/12/2020			Lại Tiến Đức
		**Control**	**Operation**	**Function**	
		search text field	initial	search bike	
		Area for displaying bike	initial	display all bike	
		back button	click	back to station detail screen	
		Rent bike button	click	Display rent bike screen	
		Row in table	click	Display bike detail screen	

ID	Name	License Plate	Producer
1	Bike	1231231312	Toyota

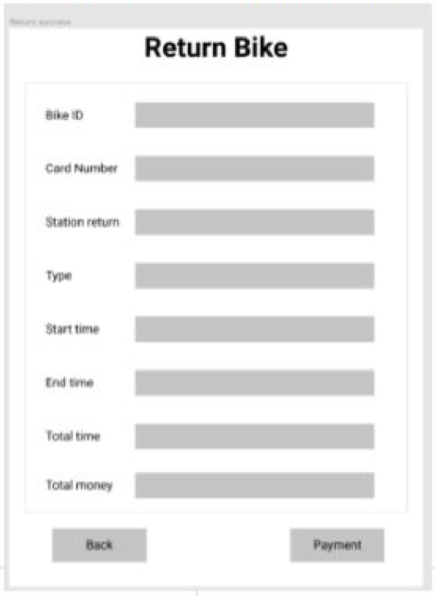
3.4.4. Bike Detail GUI

EcoBike Rental		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen sprcification	bike detail screen	10/12/2020			Lại Tiến Đức
		**Control**	**Operation**	**Function**	
		Area for displaying bike information	initial	display all information of bike, and disable;	
		back button	click	back to station list screen	
		Rent bike button	click	Display rent bike screen	

3.4.5. Rent Bike GUI

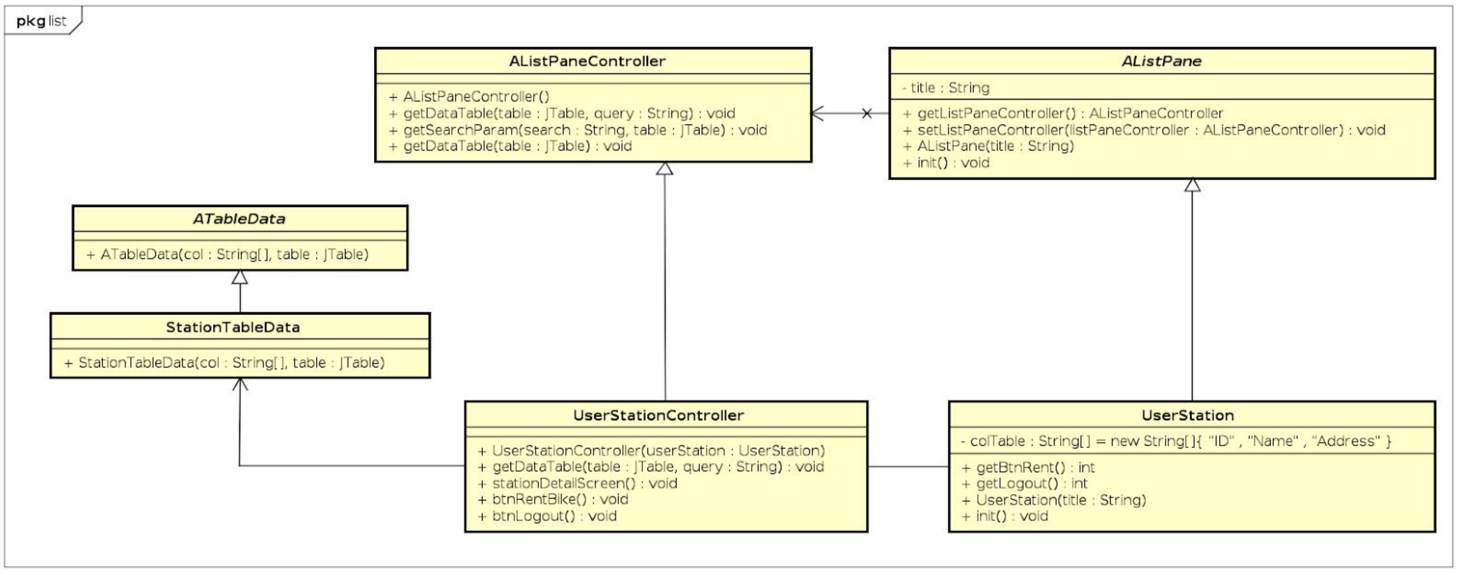
EcoBike Rental		Date of creation	Approved by	Reviewed by	Persion in charge
Screen sprcification	rent bike screen	10/12/2020			Lại Tiến Đức
 Rent Bike Bike ID Card Number License plate Type Start time End time Total time Total money Back Rent Return		**Control**	**Operation**	**Function**	
		bike id input	initial	Enter bike id to rent	
		card number	initial	Enter credit card to rent	
		back button	click	back to station list screen	
		area for displaying renting bike information		display information of renting bike after user rent	
		Rent button	click	rent bike	
		return bike	click	return bike	

3.4.6. Return bike GUI

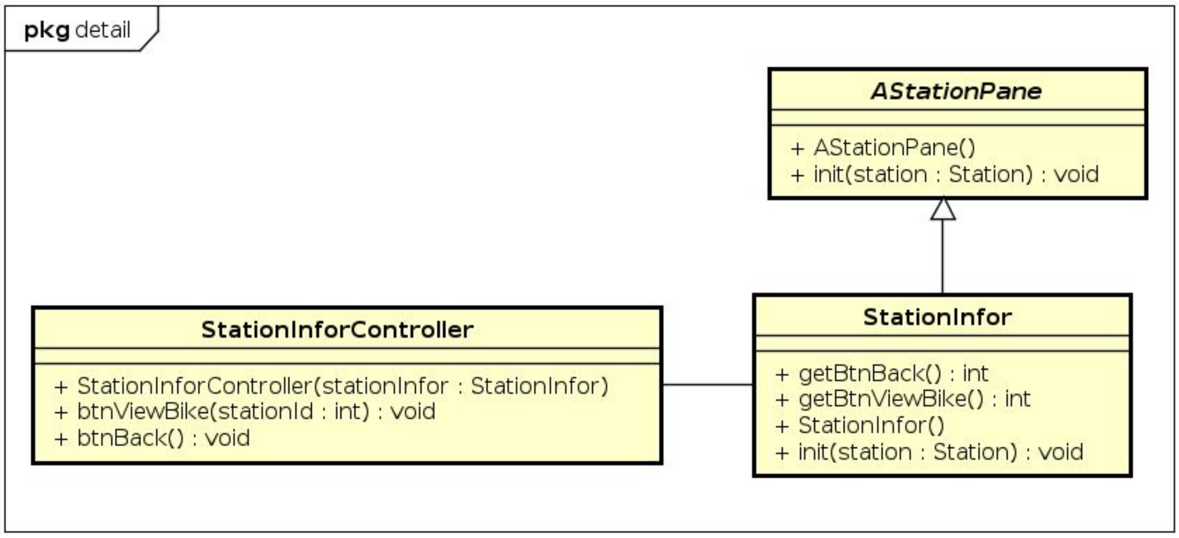
EcoBike Rental		Date of creation	Approved by	Reviewed by	Persion in charge
Screen sprcification	return bike screen	10/12/2020			Lại Tiến Đức
 Return Bike Bike ID Card Number Station return Type Start time End time Total time Total money Back Payment		**Control**	**Operation**	**Function**	
		back button	click	back to station list screen	
		area for displaying renting bike information	initial	display information of renting bike	
		Payment button	click	Display payment screen	

3.5. Class Diagram

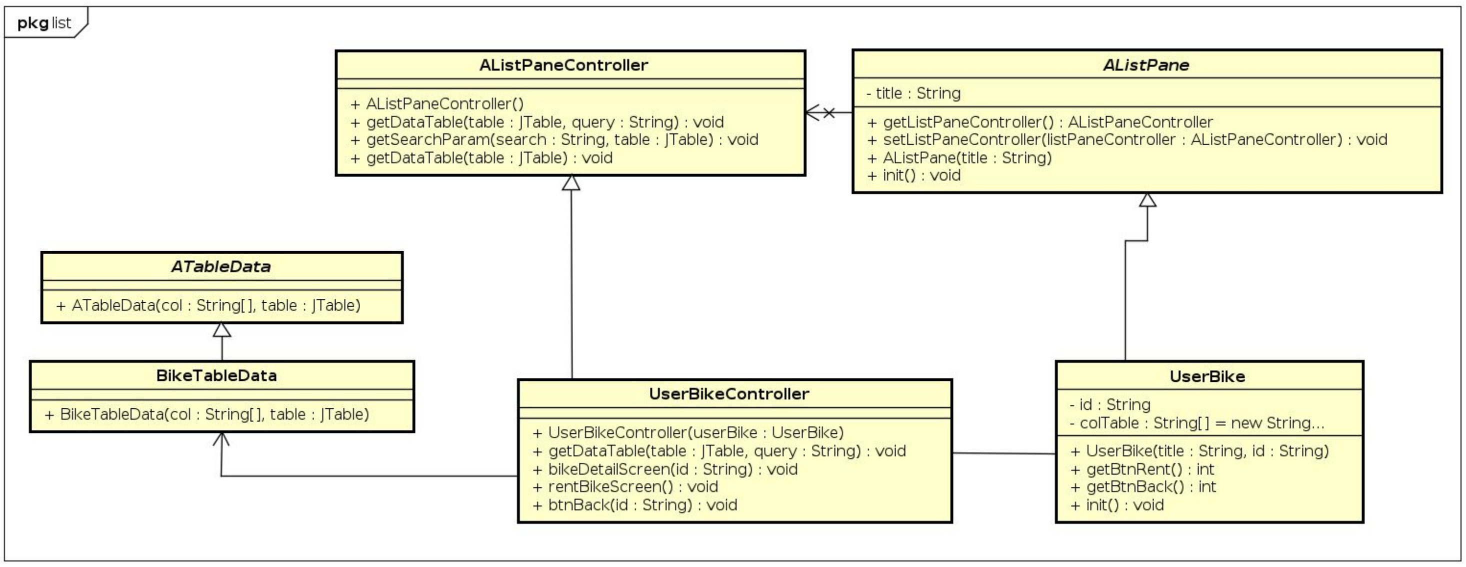
3.5.1. Class Diagram for “view Station”


pkg list
AListPaneController
+ AListPaneController()
+ getDataTable(table : JTable, query : String) : void
+ getSearchParam(search : String, table : JTable) : void
+ getDataTable(table : JTable) : void
AListPane
- title : String
+ getListPaneController() : AListPaneController
+ setListPaneController(listPaneController : AListPaneController) : void
+ AListPane(title : String)
+ init() : void
ATableData
+ ATableData(col : String[ ], table : JTable)
StationTableData
+ StationTableData(col : String[ ], table : JTable)
UserStationController
+ UserStationController(userStation : UserStation)
+ getDataTable(table : JTable, query : String) : void
+ stationDetailScreen() : void
+ btnRentBike() : void
+ btnLogout() : void
UserStation
- colTable : String[ ] = new String[ ]{ "ID" , "Name" , "Address" }
+ getBtnRent() : int
+ getLogout() : int
+ UserStation(title : String)
+ init() : void

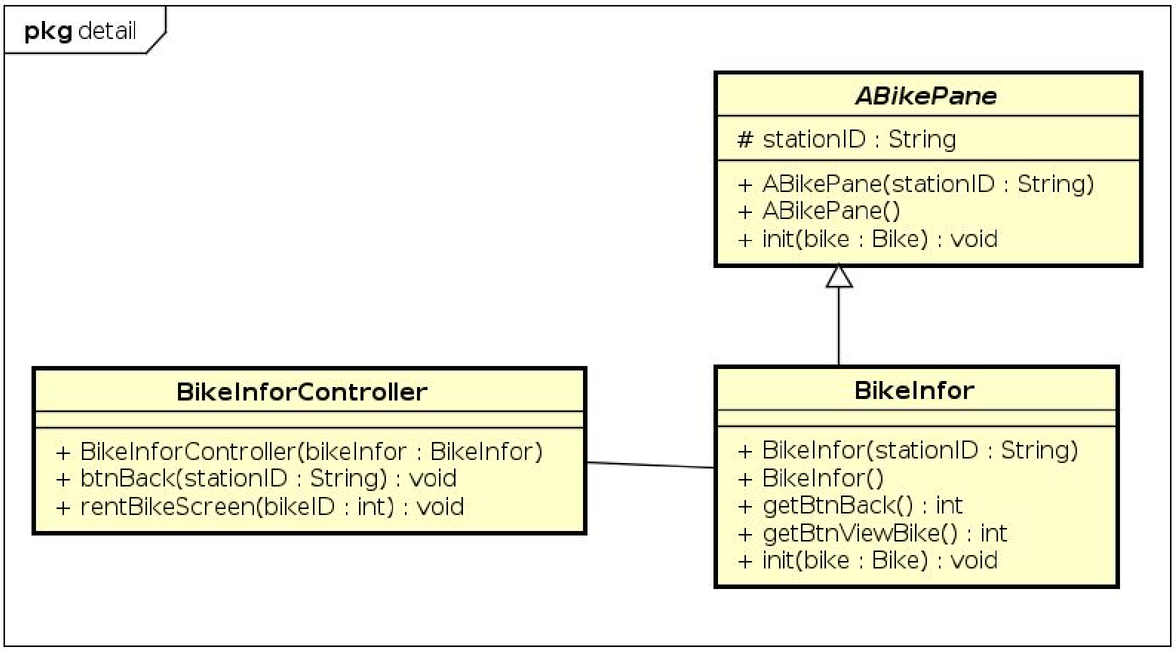
3.5.2. Class Diagram for “View Station Detail”


pkg detail
AStationPane
+ AStationPane()
+ init(station : Station) : void
StationInforController
+ StationInforController(stationInfor : StationInfor)
+ btnViewBike(stationId : int) : void
+ btnBack() : void
StationInfor
+ getBtnBack() : int
+ getBtnViewBike() : int
+ StationInfor()
+ init(station : Station) : void

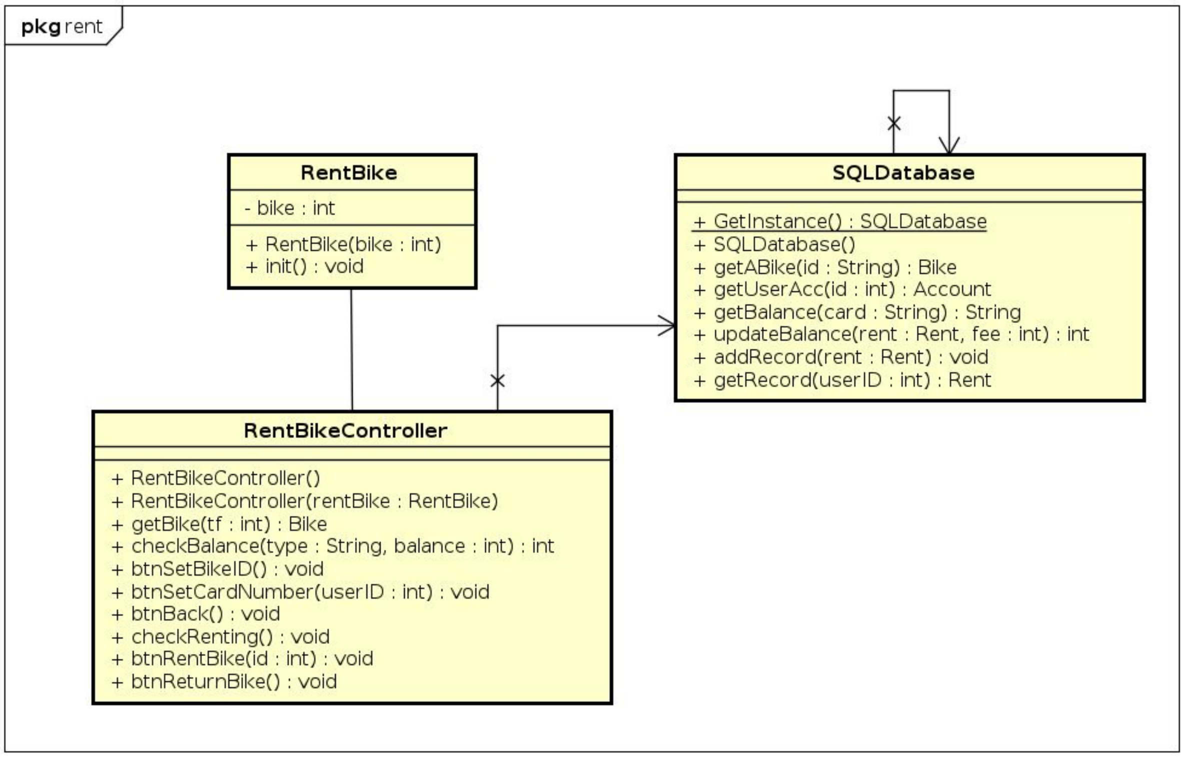
3.5.3. Class Diagram for “View Bike in station”


pkg list
AListPaneController
+ AListPaneController()
+ getDataTable(table : JTable, query : String) : void
+ getSearchParam(search : String, table : JTable) : void
+ getDataTable(table : JTable) : void
AListPane
- title : String
+ getListPaneController() : AListPaneController
+ setListPaneController(listPaneController : AListPaneController) : void
+ AListPane(title : String)
+ init() : void
ATableData
+ ATableData(col : String[], table : JTable)
BikeTableData
+ BikeTableData(col : String[], table : JTable)
UserBikeController
+ UserBikeController(userBike : UserBike)
+ getDataTable(table : JTable, query : String) : void
+ bikeDetailScreen(id : String) : void
+ rentBikeScreen() : void
+ btnBack(id : String) : void
UserBike
- id : String
- colTable : String[] = new String...
+ UserBike(title : String, id : String)
+ getBtnRent() : int
+ getBtnBack() : int
+ init() : void

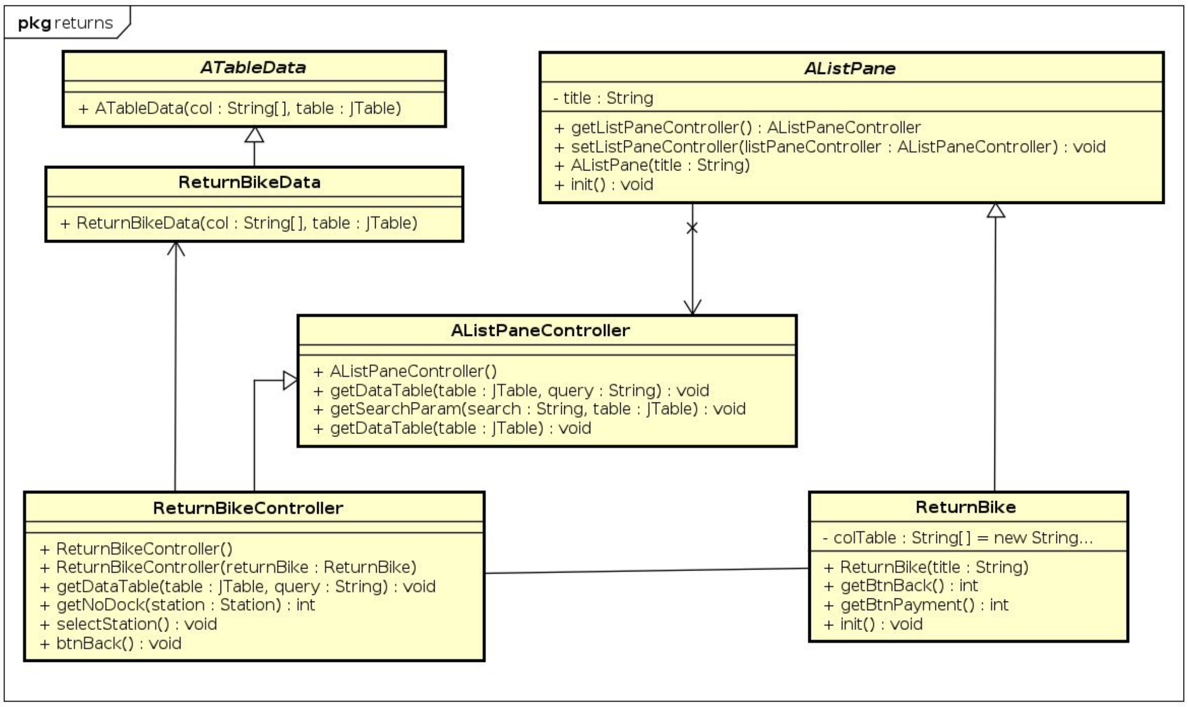
3.5.4. Class Diagram for “View bike detail”


pkg detail
ABikePane
# stationID : String
+ ABikePane(stationID : String)
+ ABikePane()
+ init(bike : Bike) : void
BikeInforController
+ BikeInforController(bikeInfor : BikeInfor)
+ btnBack(stationID : String) : void
+ rentBikeScreen(bikeID : int) : void
BikeInfor
+ BikeInfor(stationID : String)
+ BikeInfor()
+ getBtnBack() : int
+ getBtnViewBike() : int
+ init(bike : Bike) : void

3.5.5. Class Diagram for “Rent bike”

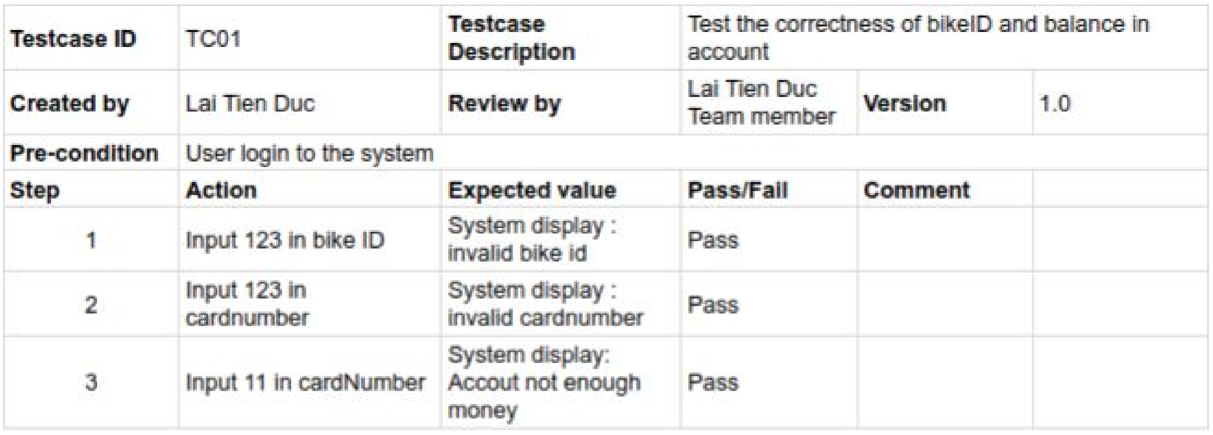

pkg rent
RentBike
- bike : int
+ RentBike(bike : int)
+ init() : void
SQLDatabase
+ GetInstance() : SQLDatabase
+ SQLDatabase()
+ getABike(id : String) : Bike
+ getUserAcc(id : int) : Account
+ getBalance(card : String) : String
+ updateBalance(rent : Rent, fee : int) : int
+ addRecord(rent : Rent) : void
+ getRecord(userID : int) : Rent
RentBikeController
+ RentBikeController()
+ RentBikeController(rentBike : RentBike)
+ getBike(tf : int) : Bike
+ checkBalance(type : String, balance : int) : int
+ btnSetBikeID() : void
+ btnSetCardNumber(userID : int) : void
+ btnBack() : void
+ checkRenting() : void
+ btnRentBike(id : int) : void
+ btnReturnBike() : void

3.5.6. Class Diagram for “Return Bike”


pkg returns
ATableData
+ ATableData(col : String[], table : JTable)
ReturnBikeData
+ ReturnBikeData(col : String[], table : JTable)
AListPane
- title : String
+ getListPaneController() : AListPaneController
+ setListPaneController(listPaneController : AListPaneController) : void
+ AListPane(title : String)
+ init() : void
AListPaneController
+ AListPaneController()
+ getDataTable(table : JTable, query : String) : void
+ getSearchParam(search : String, table : JTable) : void
+ getDataTable(table : JTable) : void
ReturnBikeController
+ ReturnBikeController()
+ ReturnBikeController(returnBike : ReturnBike)
+ getDataTable(table : JTable, query : String) : void
+ getNoDock(station : Station) : int
+ selectStation() : void
+ btnBack() : void
ReturnBike
- colTable : String[] = new String...
+ ReturnBike(title : String)
+ getBtnBack() : int
+ getBtnPayment() : int
+ init() : void

4. Test plan

4.1. Testcase for “Rent bike”

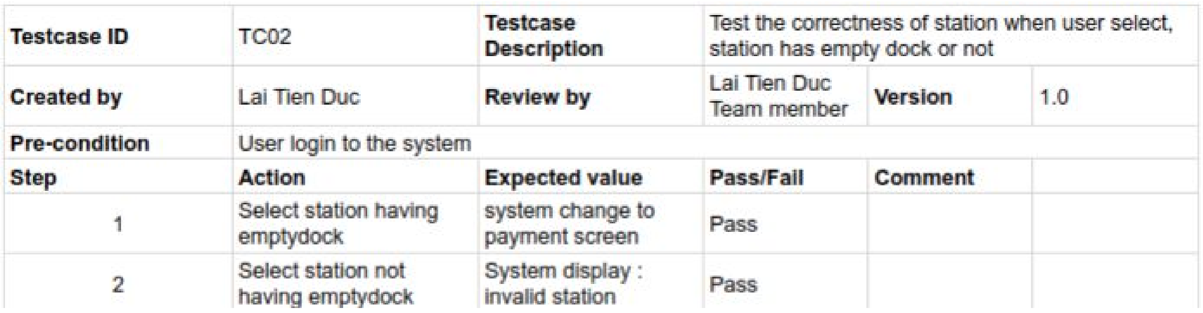


Testcase ID	TC01	**Testcase Description**	Test the correctness of bikeID and balance in account		
Created by	Lai Tien Duc	**Review by**	Lai Tien Duc Team member	**Version**	1.0
Pre-condition	User login to the system				
Step	**Action**	**Expected value**	**Pass/Fail**	**Comment**	
1	Input 123 in bike ID	System display : invalid bike id	Pass		
2	Input 123 in cardnumber	System display : invalid cardnumber	Pass		
3	Input 11 in cardNumber	System display: Accout not enough money	Pass		

I use both black-box and white-box unittest

- Black-box:
  - when input 123 in bikeID text field, expect function getBike() return null
  - when input 11 in bikeID text field, expect function checkBalace() return 0
- White-box:
  - Expect system display error box when get invalid value

4.2. Testcase for “Return bike”



Testcase ID	TC02	**Testcase Description**	Test the correctness of station when user select, station has empty dock or not		
Created by	Lai Tien Duc	**Review by**	Lai Tien Duc Team member	**Version**	1.0
Pre-condition	User login to the system				
Step	**Action**	**Expected value**	**Pass/Fail**	**Comment**	
1	Select station having emptydock	system change to payment screen	Pass		
2	Select station not having emptydock	System display : invalid station	Pass		

I use white-box unittest

- White-box:
  - Expect system display error box when select station which doesn’t have empty dock

5. Design pattern

- In this project, I used Single design. The reason why i chose this design pattern is that i have class MainController() and SQLDatabase() which are used entry in the project and the instance of those should be initialized one time.
-

```
private static SQLDatabase sql;
   public static SQLDatabase GetInstance() {
      if (sql == null) {
         sql = new SQLDatabase();
      }
      return sql;
   }
```

-

```
private static MainController mainController;
public static MainController GetInstance() {
      if (mainController == null) {
         mainController = new MainController();
      }
      return mainController;
   }
```

- I also use GRASP and SOLID design principles
    - GRASP - coupling: each class associate with class in same package and database class
    - GRASP - cohension: each class has specific fuction

        Example: returnBike class display information of renting bike and total time, total money for user to pay
    - SOLID
- All design principle and design pattern I applied make maintain, debug more easily and can develop more requirement like: add new bike type or change payment method