HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

<Project Name>

Subject: <Name of subject>

<Group Number>

<List of participants>

*Hanoi,* *<month, year>*

*<All notations inside the angle bracket are not part of this document, for its purpose is for extra instruction. When using this document, please erase all these notations and/or replace them with corresponding content as instructed>*

*<This document, written by Asst. Prof. NGUYEN Thi Thu Trang, is used as a case study for student with related courses. Any modifications and/or utilization without the consent of the author is strictly forbidden>*

Table of Contents

Table of Contents 1

1 Introduction 3

1.1 Objective 3

1.2 Scope 3

1.3 Glossary 3

1.4 References 3

2 Overall Description 4

2.1 General Overview 4

2.2 Assumptions/Constraints/Risks 4

2.2.1 Assumptions 4

2.2.2 Constraints 4

2.2.3 Risks 5

3 System Architecture and Architecture Design 6

3.1 Architectural Patterns 6

3.2 Interaction Diagrams 6

3.3 Analysis Class Diagrams 6

3.4 Unified Analysis Class Diagram 6

3.5 Security Software Architecture 6

4 Detailed Design 7

4.1 User Interface Design 7

4.1.1 Screen Configuration Standardization 7

4.1.2 Screen Transition Diagrams 7

4.1.3 Screen Specifications 7

4.2 Data Modeling 7

4.2.1 Conceptual Data Modeling 7

4.2.2 Database Design 7

4.3 Non-Database Management System Files 8

4.4 Class Design 8

4.4.1 General Class Diagram 8

4.4.2 Class Diagrams 8

4.4.3 Class Design 8

5 Design Considerations 10

5.1 Goals and Guidelines 10

5.2 Architectural Strategies 10

5.3 Coupling and Cohesion 11

5.4 Design Principles 11

5.5 Design Patterns 11

**List of Figures**

**List of Tables**

No table of figures entries found.

# Introduction

*<The following subsections of the Software Design Document (SDD) document should provide an overview of the entire SDD.>*

## Objective

<*Identify the purpose of this SDD and its intended audience. In this subsection, describe the purpose of the SDD and specify the intended audience for the SDD*>

## Scope

<*In this subsection:*

1. *Identify the software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here*

*Note that this will be similar to what was written in the SRS.*

>

## Glossary

*<Listing and explaining the terms appearing in the software’s profession and this document. Any assumption of the reader’s prior knowledge or experience on the subject is ill advised>*

## References

Centers for Medicare & Medicaid Services. (n.d.). *System Design Document Template.* Retrieved from Centers for Medicare & Medicaid Services: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx

*<Listing the referenced material used in this document, including the one related to the project>*

# Overall Description

<*This section describes the principles and strategies to be used as guidelines when designing and implementing the system.>*

## General Overview

*<Briefly introduce the system context and the basic design approach or organization. Provide a brief overview of the system and software architectures and the design goals. Include the high-level context diagram(s) for the system and subsystems provided in previous documents like SRS (e.g., general use case diagram, lower-level use case diagrams, activity diagrams), updated as necessary to reflect any changes that have been made based on more current information or understanding. If the high-level context diagram has been updated, identify the changes that were made and why>*

## Assumptions/Constraints/Risks

### Assumptions

*<Describe any assumptions or dependencies regarding the system, software and its use. These may concern such issues as: related software or hardware, operating systems, end-user characteristics, and possible and/or probable changes in functionality>*

### Constraints

*<Describe any global limitations or constraints that have a significant impact on the design of the system’s hardware, software and/or communications, and describe the associated impact. Such constraints may be imposed by any of the following (the list is not exhaustive):*

* *Hardware or software environment*
* *End-user environment*
* *Availability or volatility of resources*
* *Standards compliance*
* *Interoperability requirements*
* *Interface/protocol requirements*
* *Licensing requirements*
* *Data repository and distribution requirements*
* *Security requirements (or other such regulations)*
* *Memory or other capacity limitations*
* *Performance requirements*
* *Network communications*
* *Verification and validation requirements (testing)*
* *Other means of addressing quality goals*
* *Other requirements described in the Requirements Document*

*>*

### Risks

*<Describe any risks associated with the system design and proposed mitigation strategies.>*

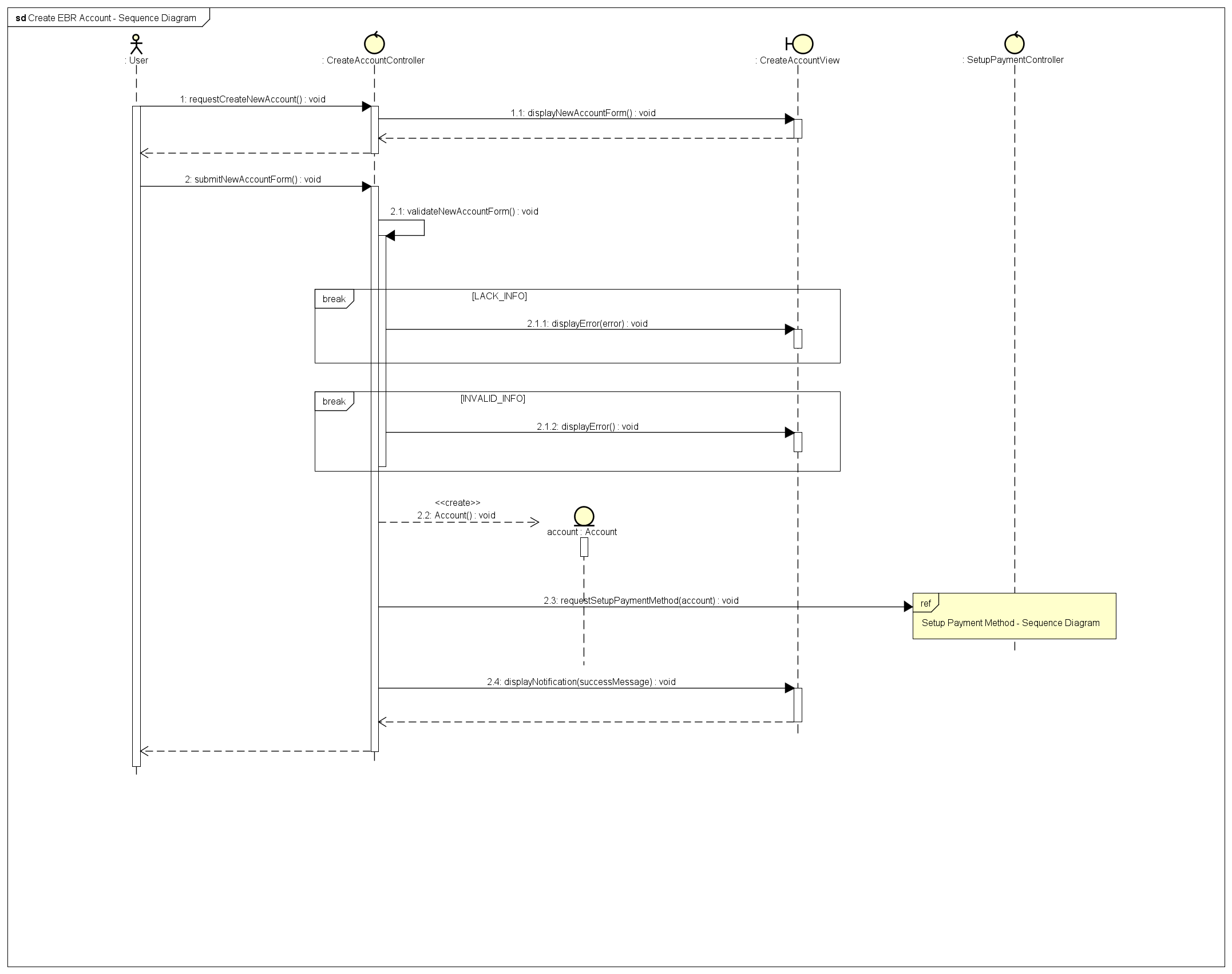
# System Architecture and Architecture Design

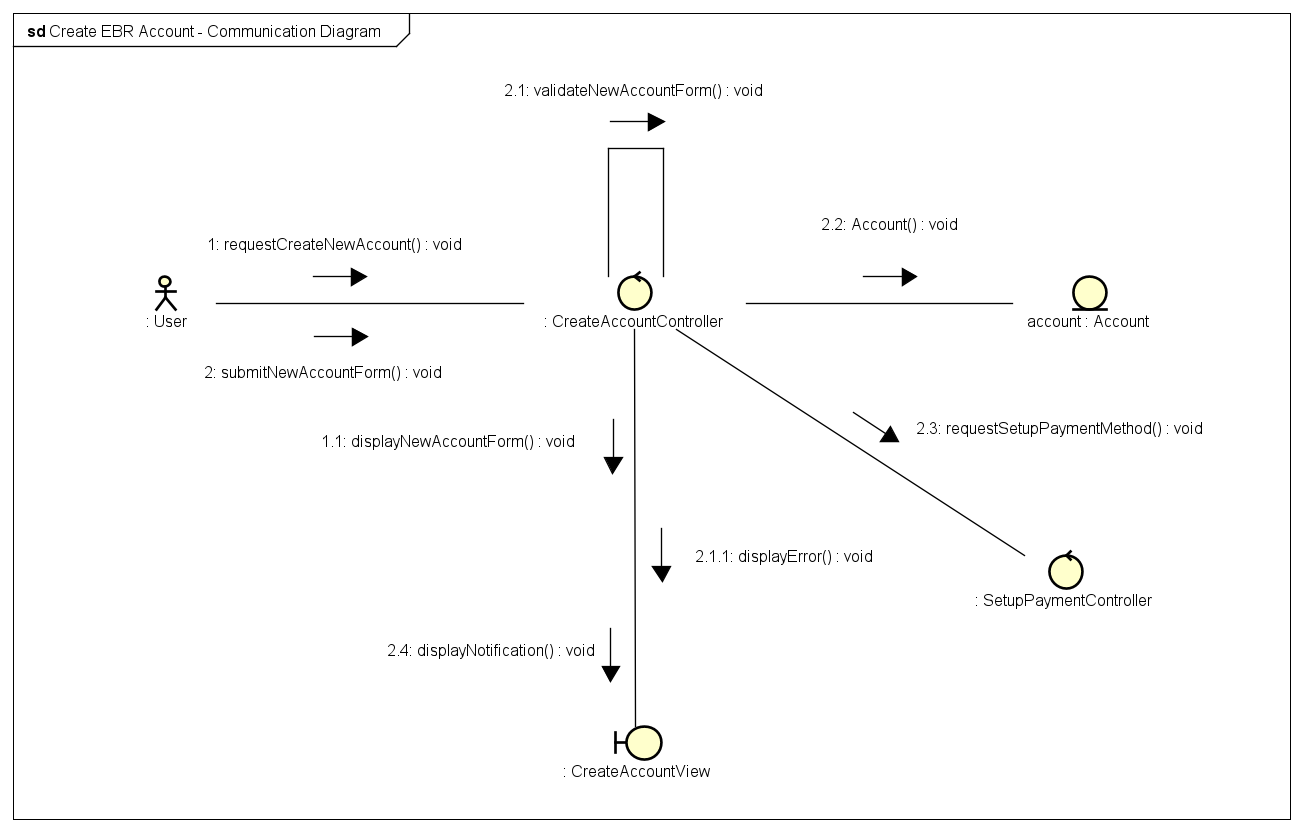
<*Briefly describe the architectural design steps*>

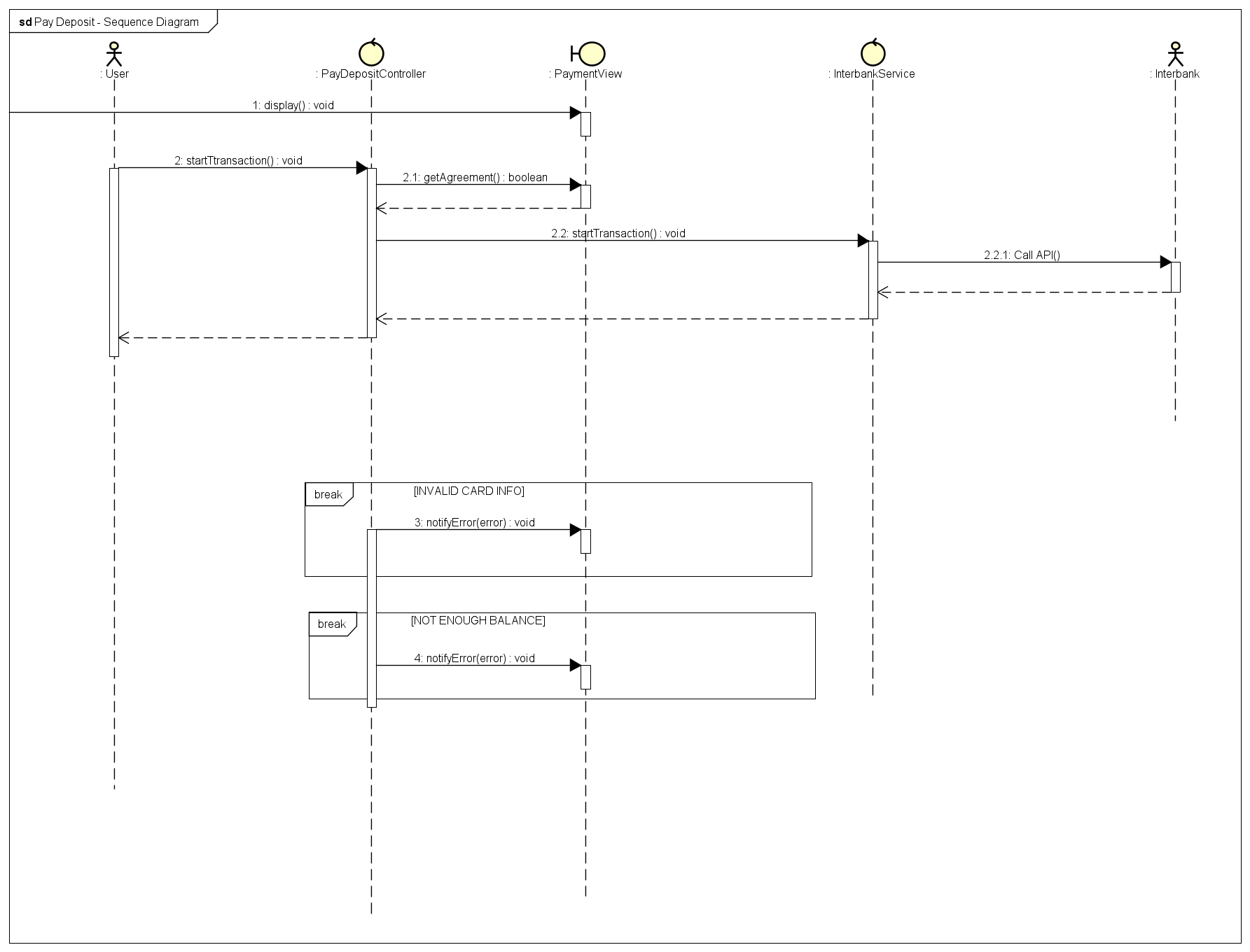
## Architectural Patterns

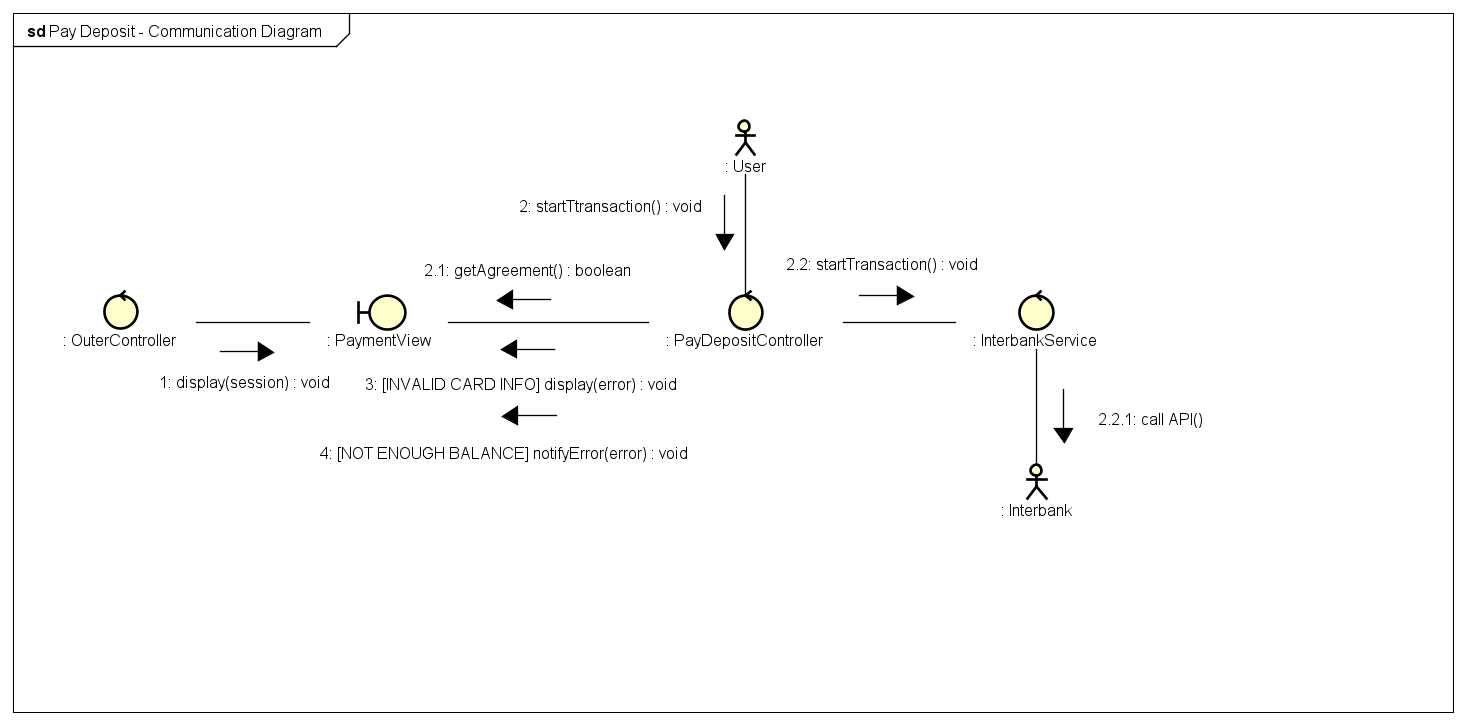
*<Specify and briefly describe the chosen architectural patterns and the reasons why they were chosen>*

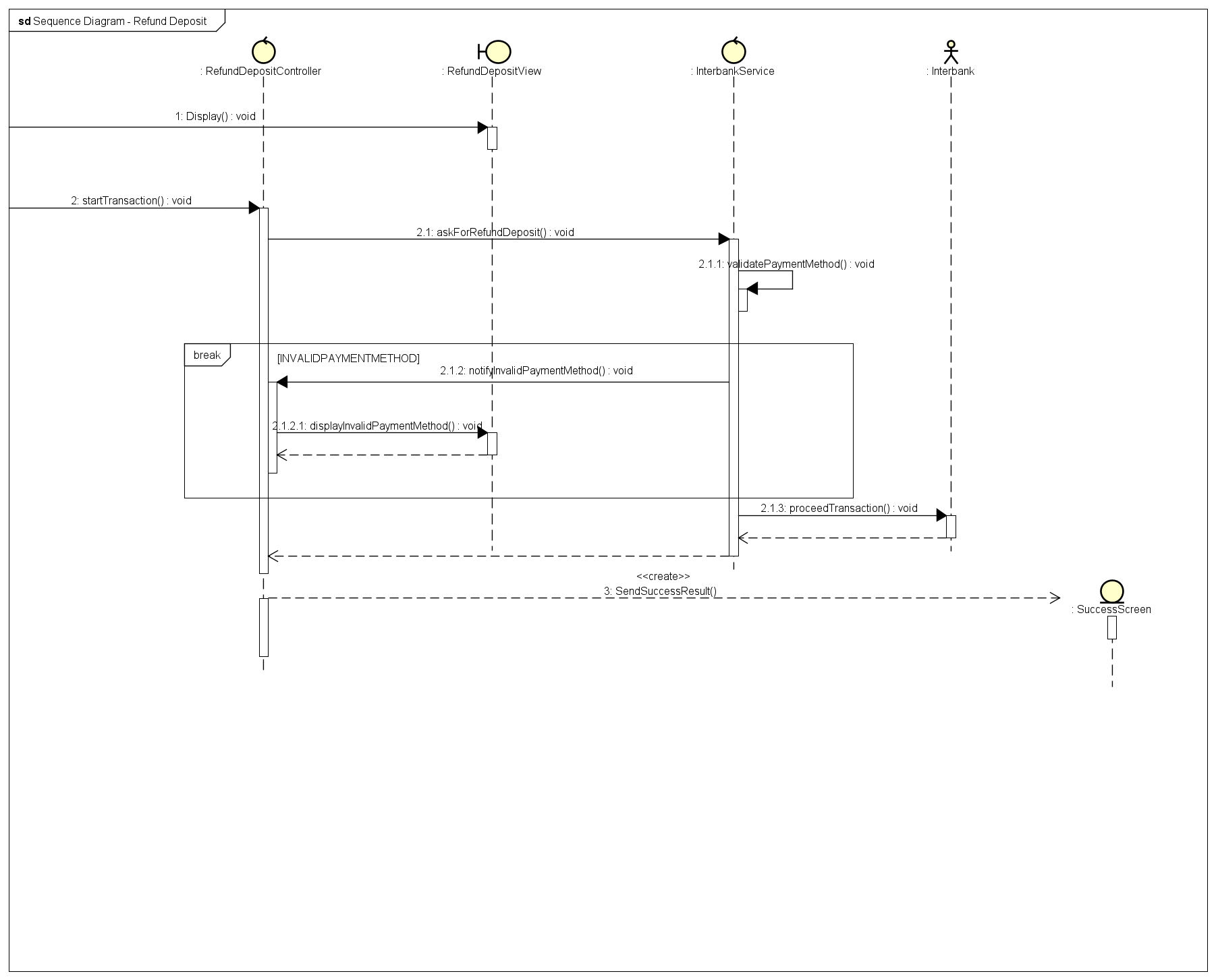
## Interaction Diagrams

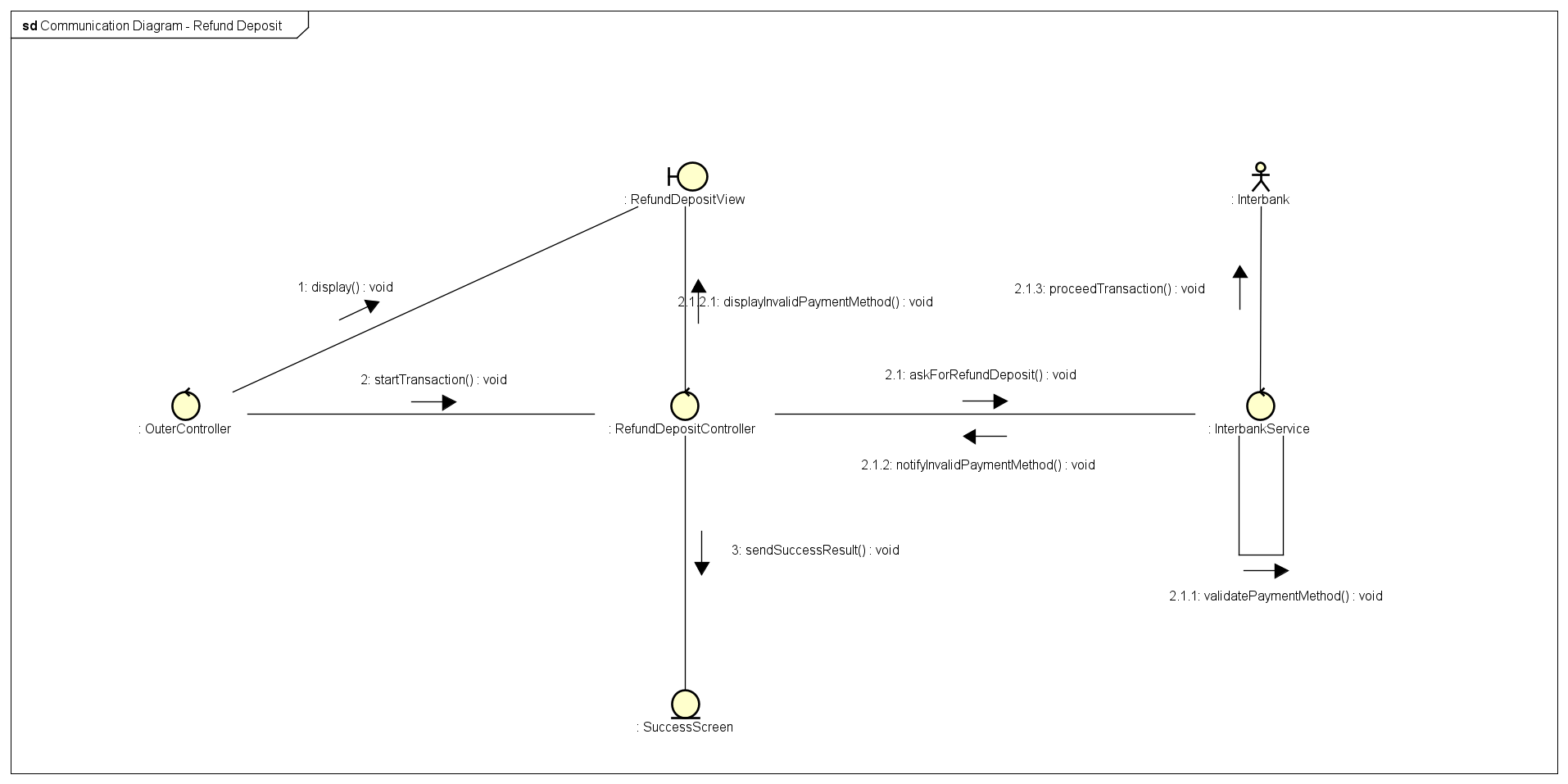


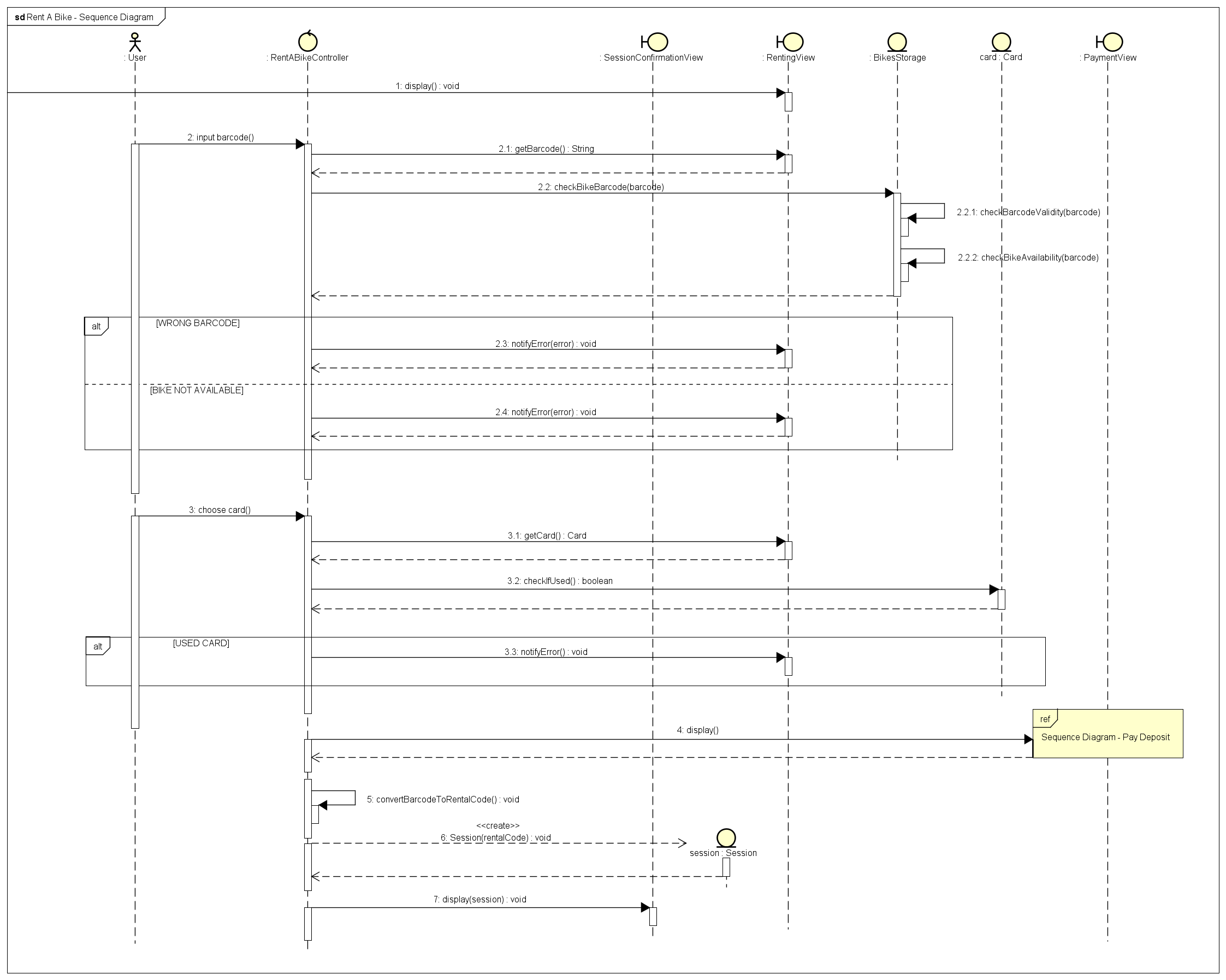


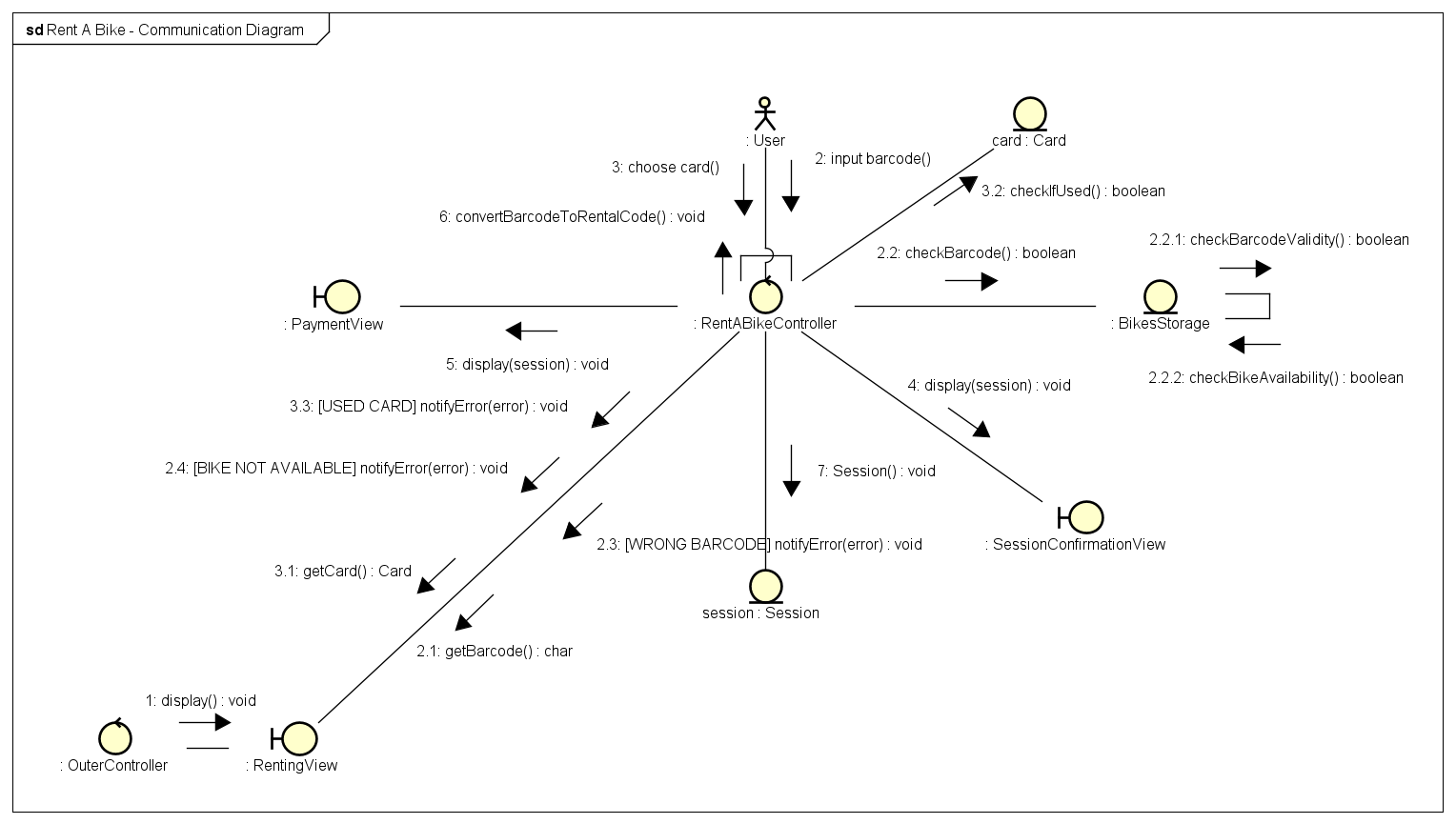


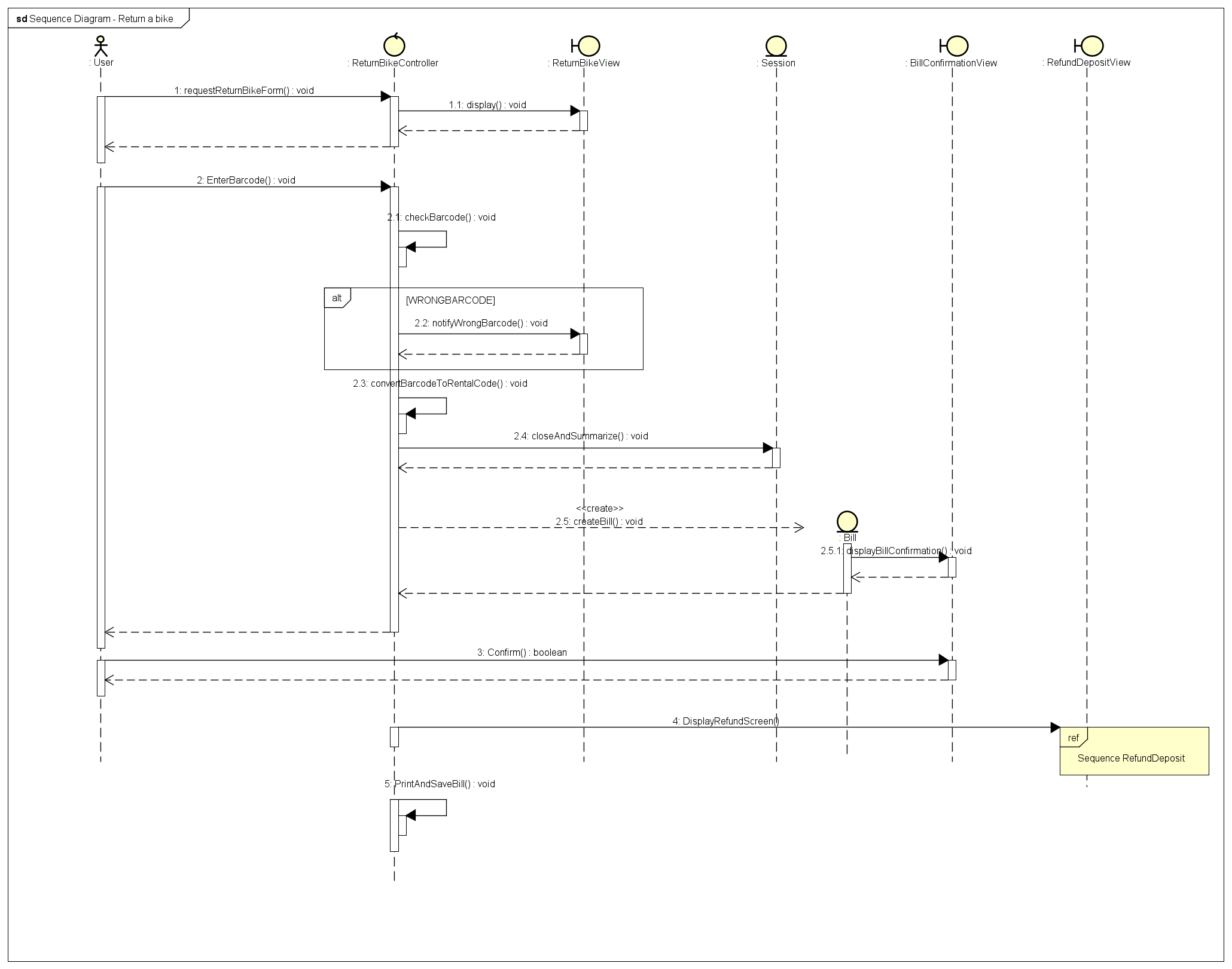


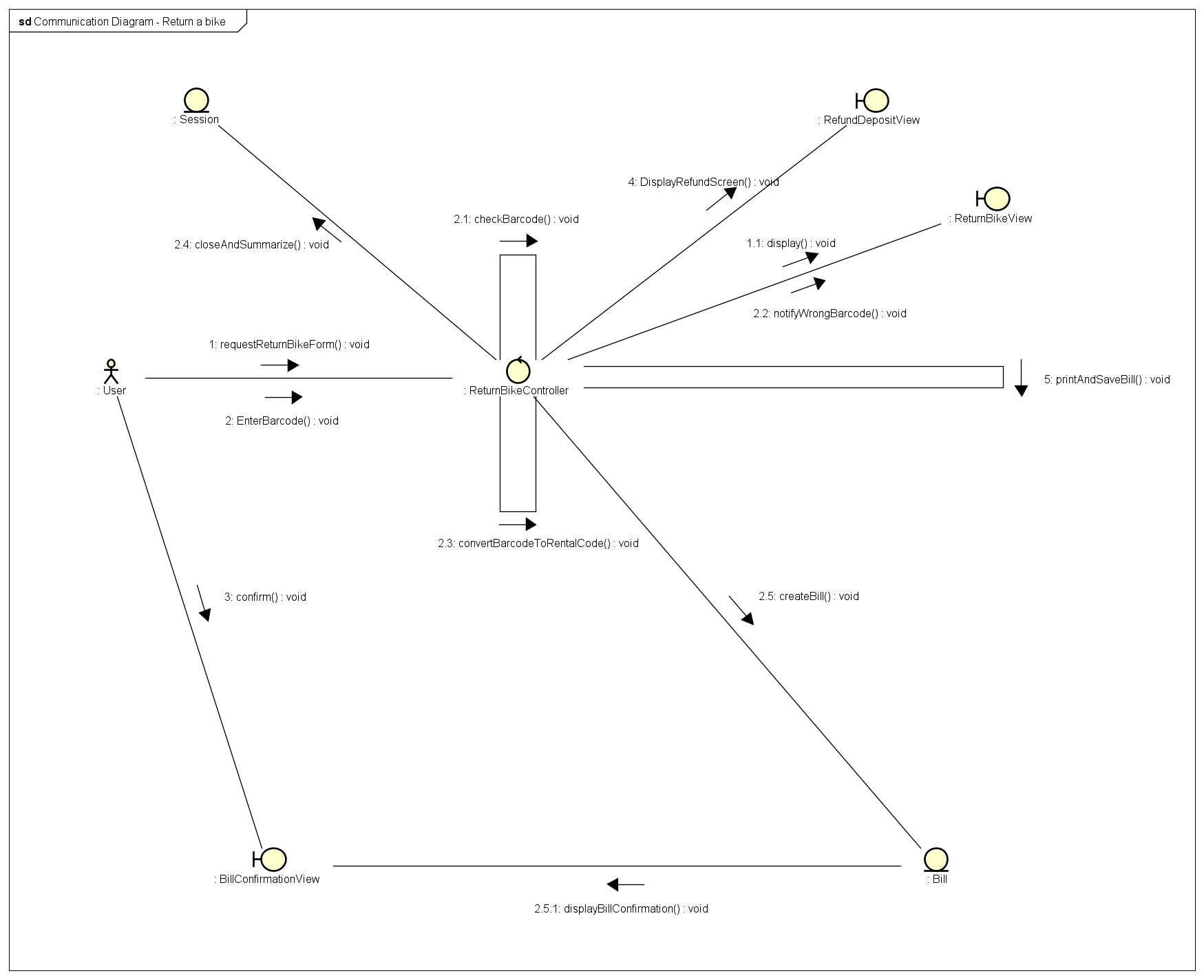


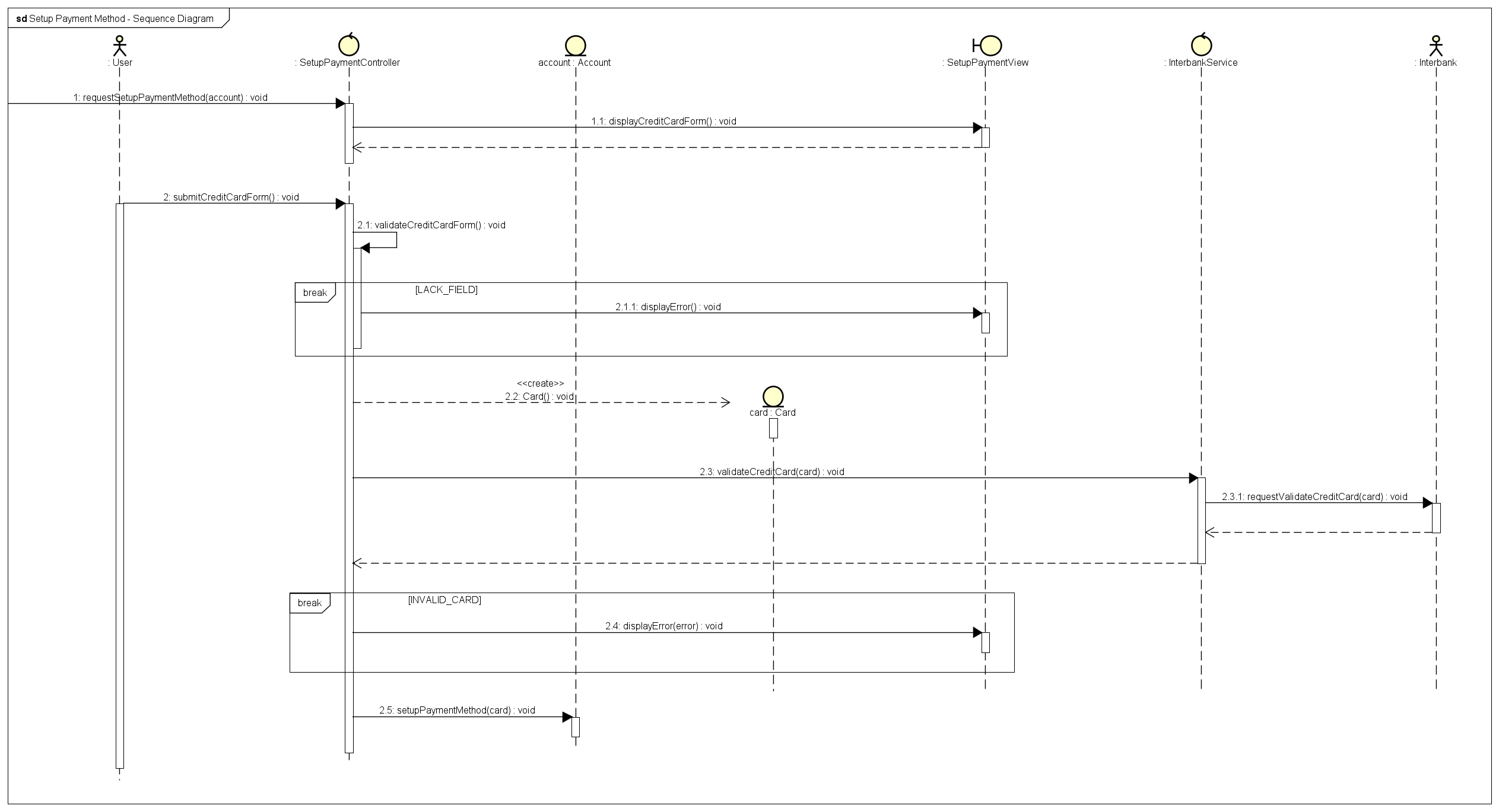




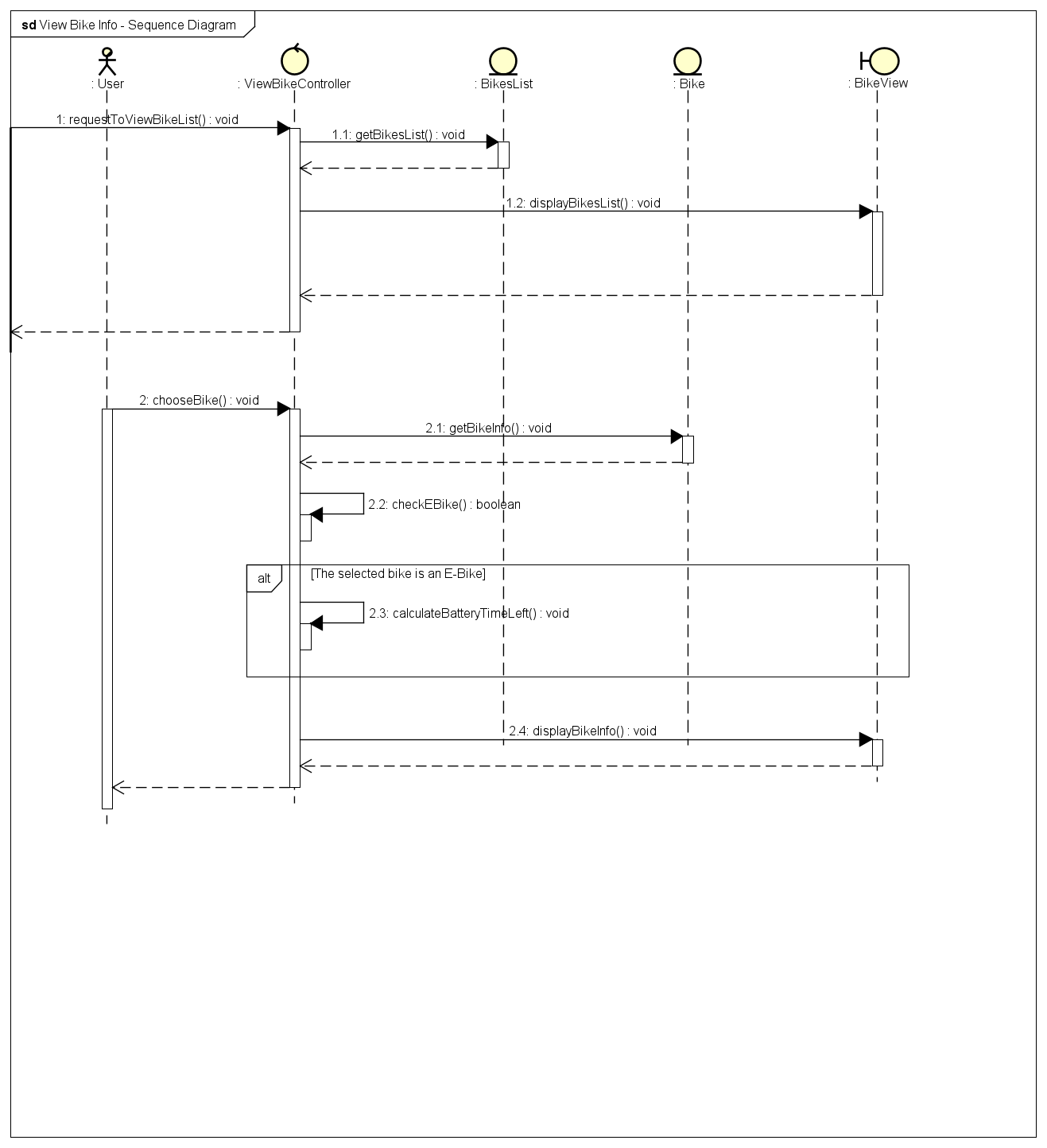


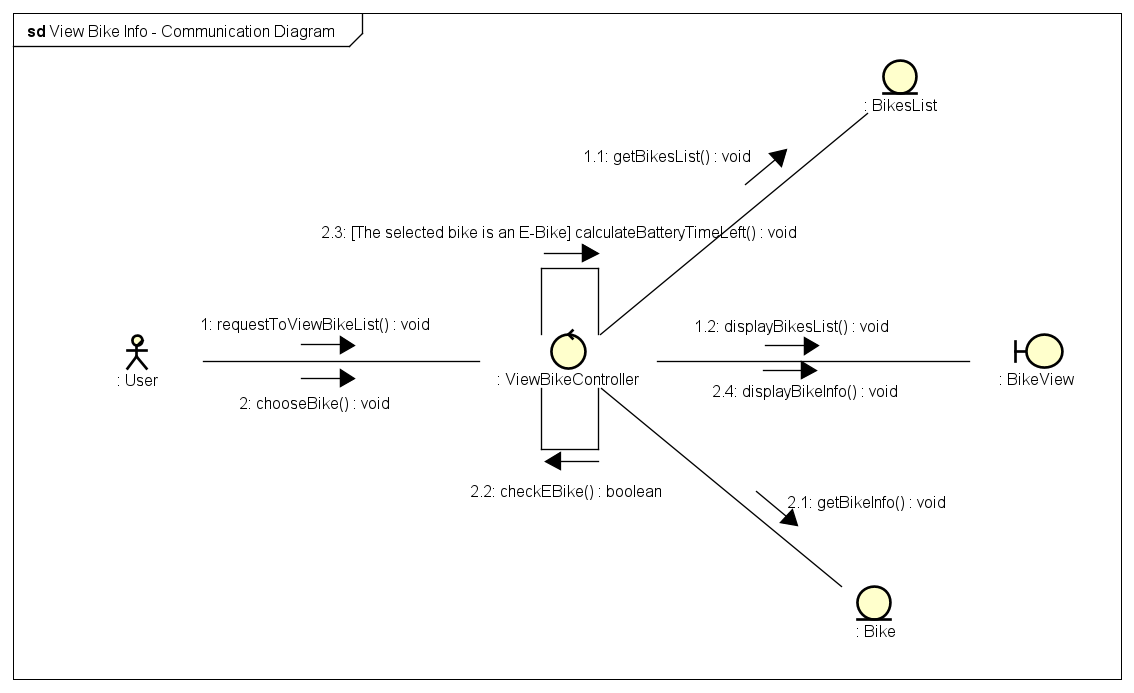


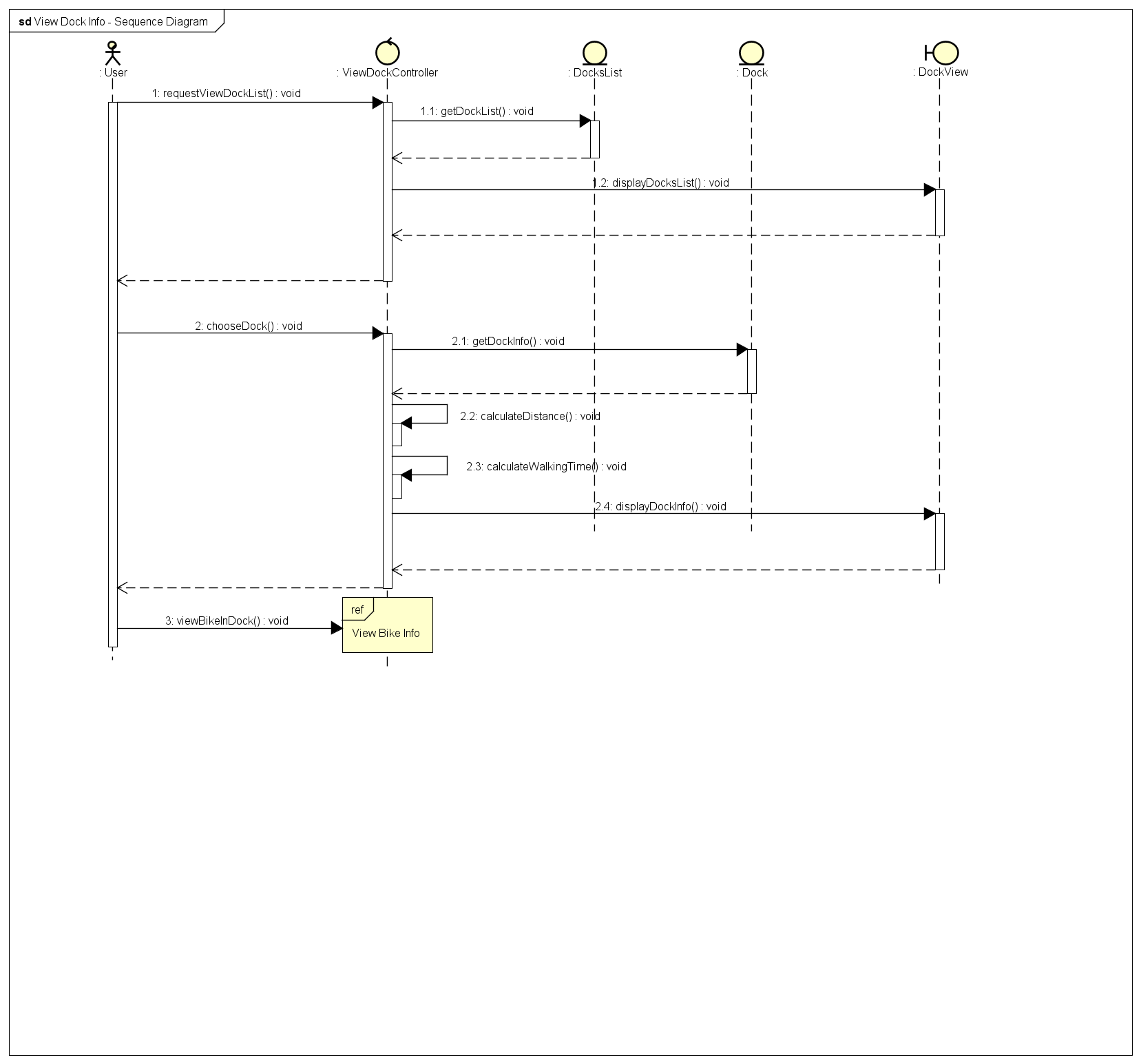


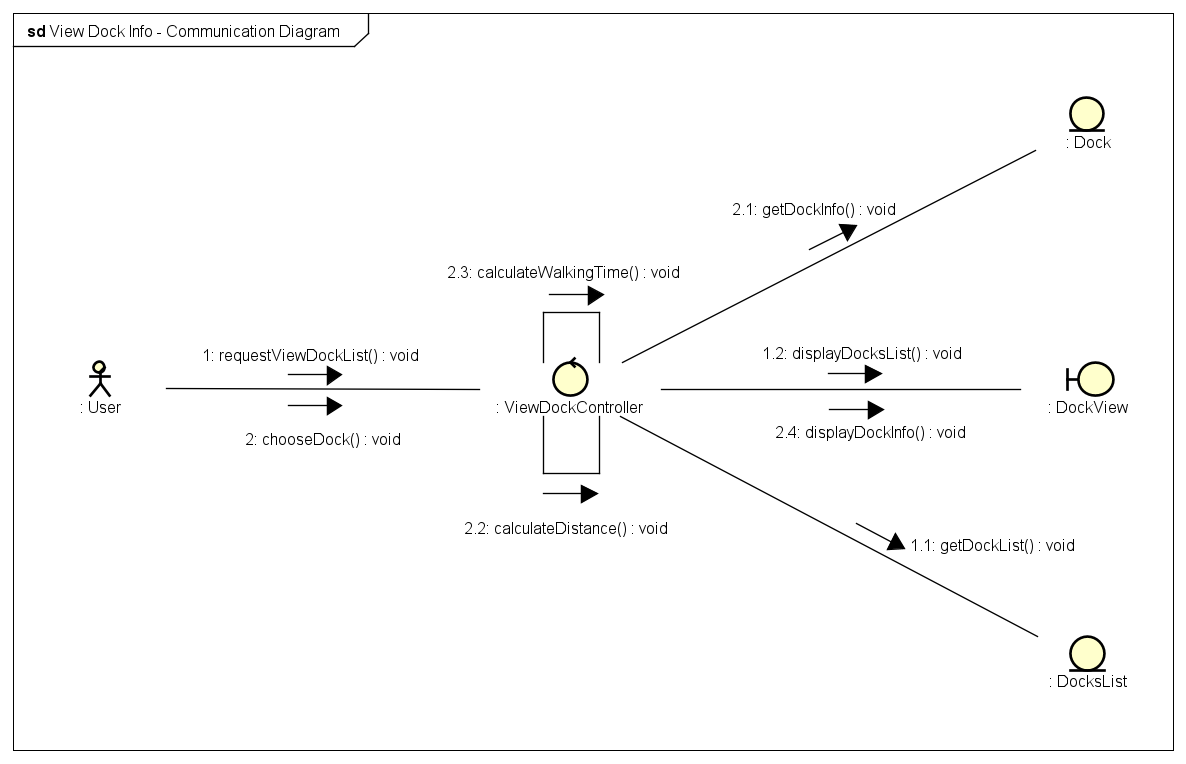






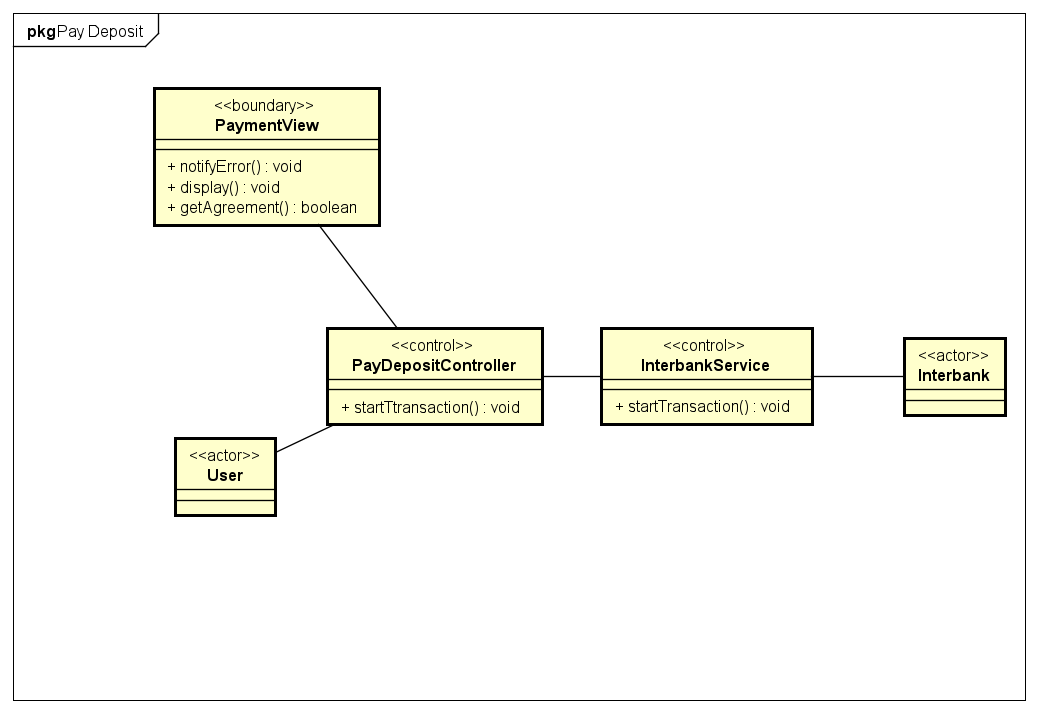


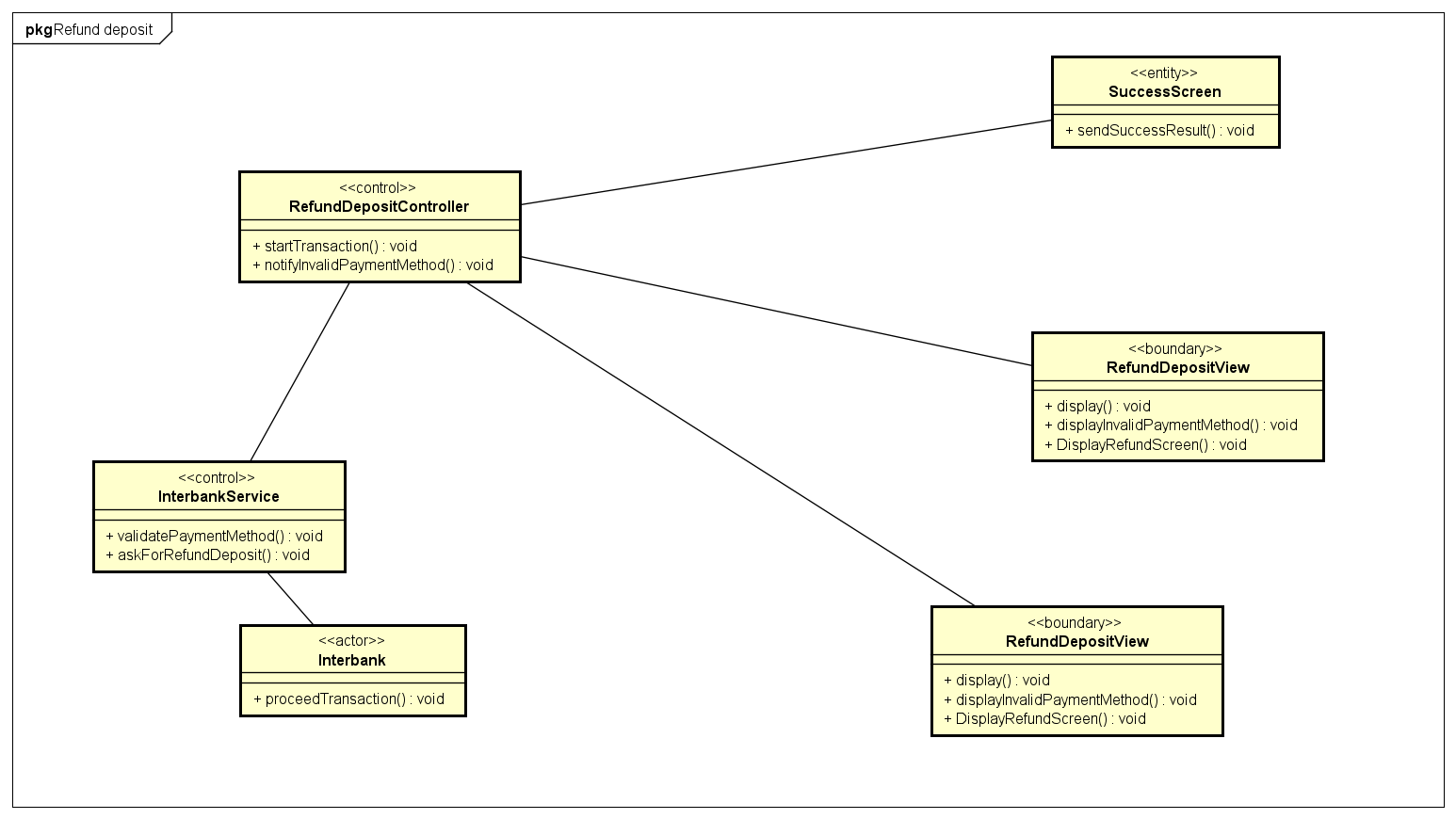


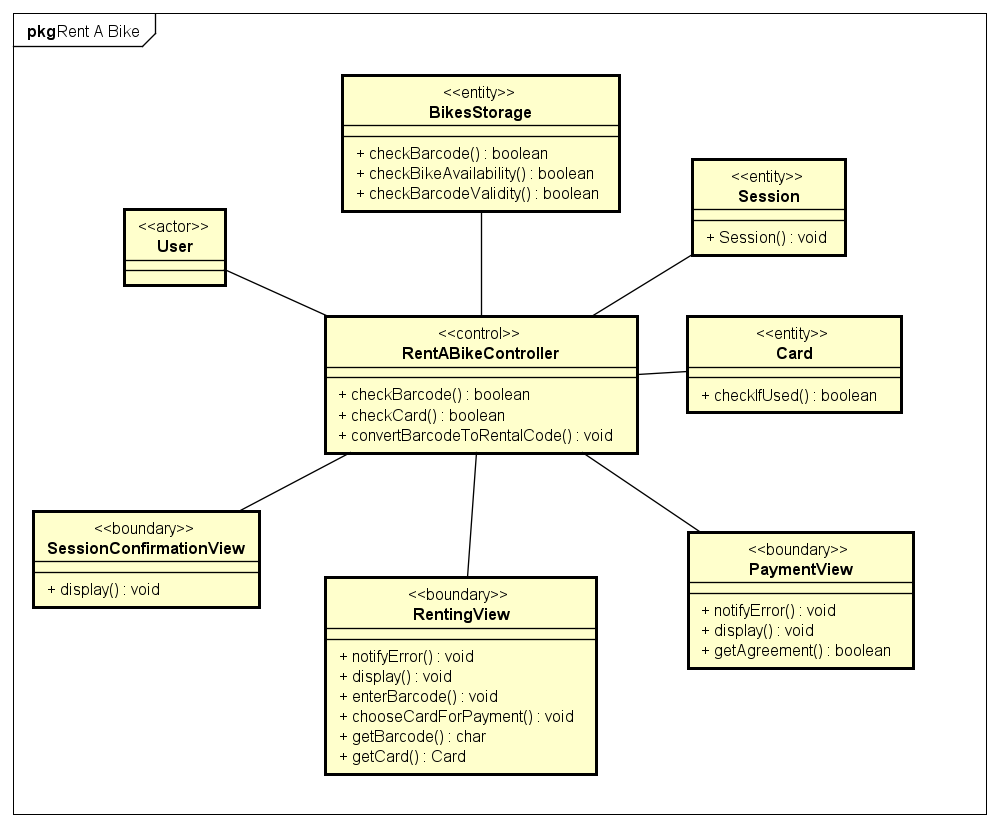


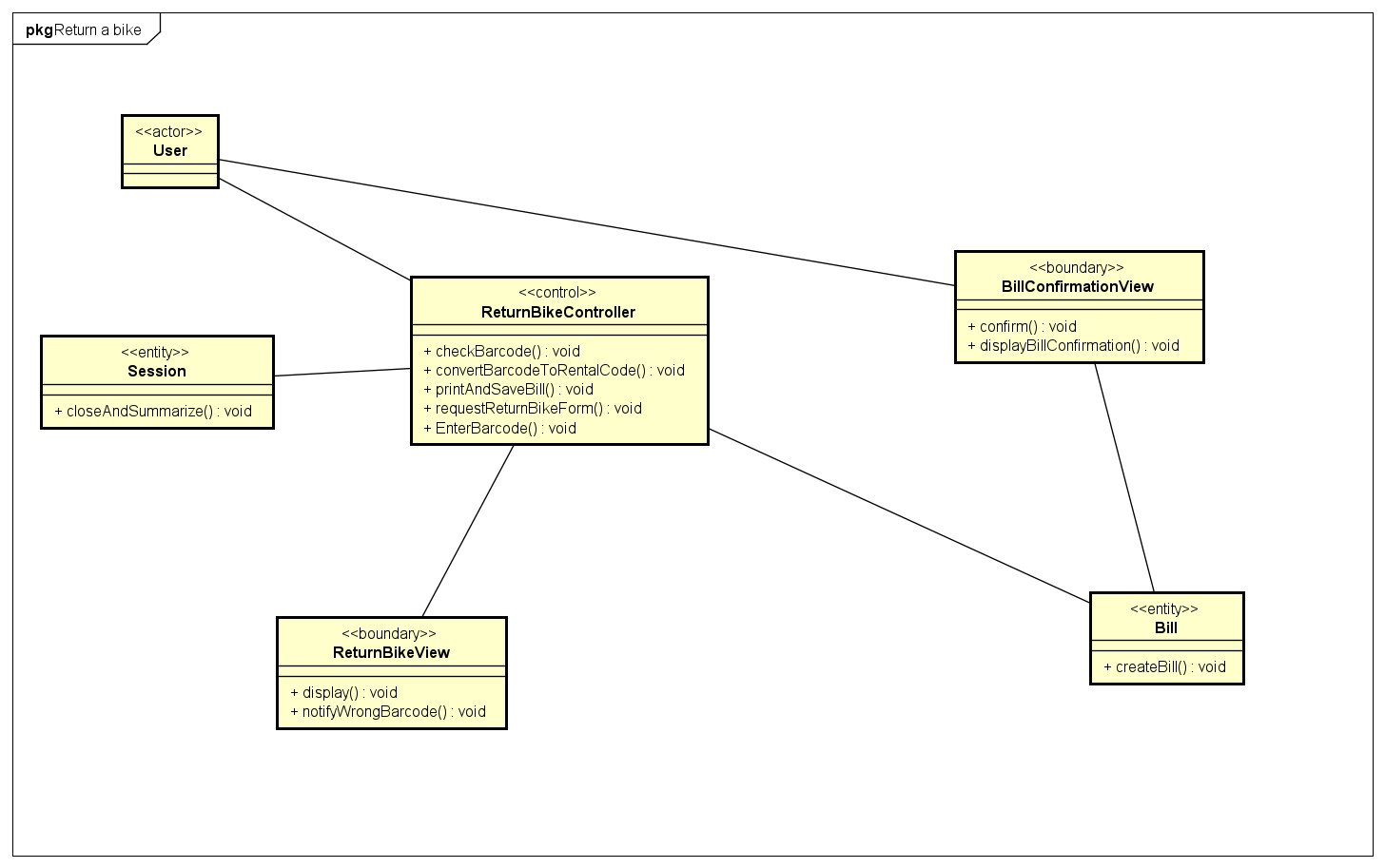
## Analysis Class Diagrams

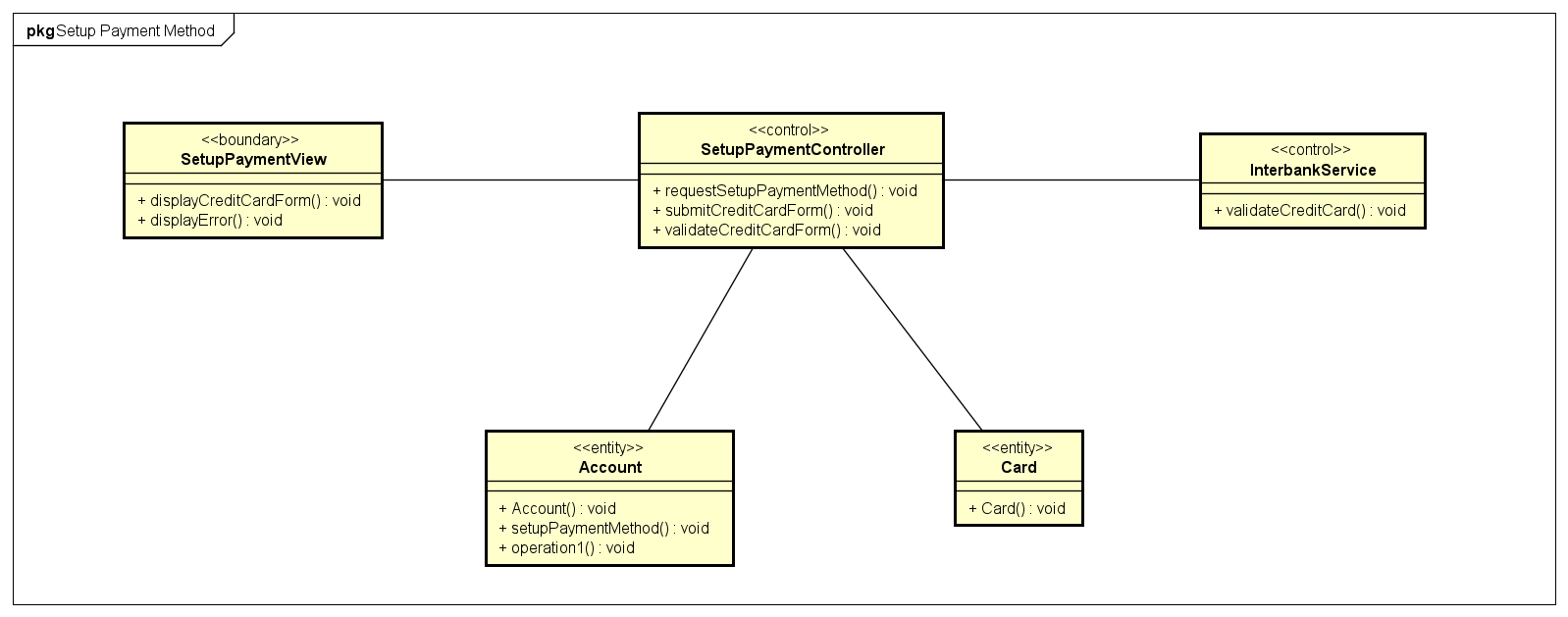


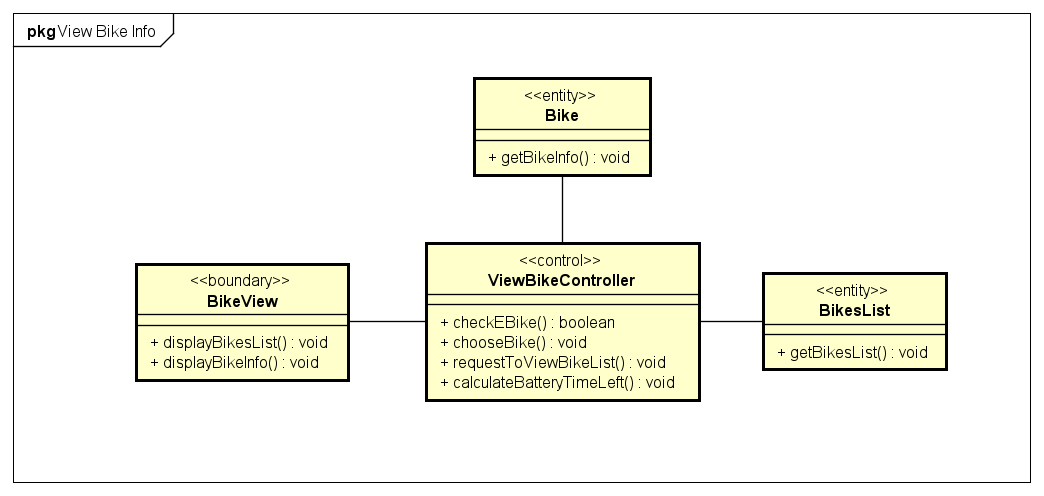


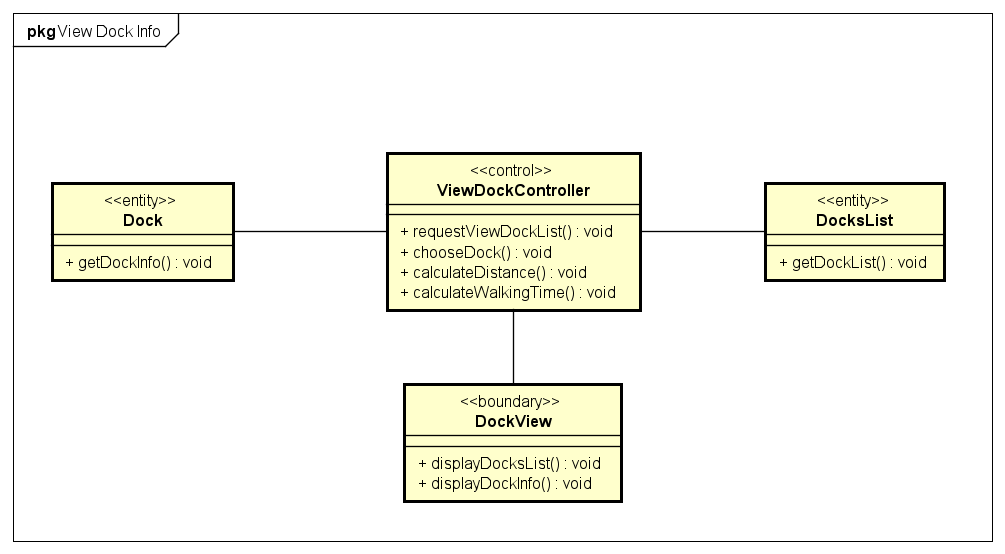




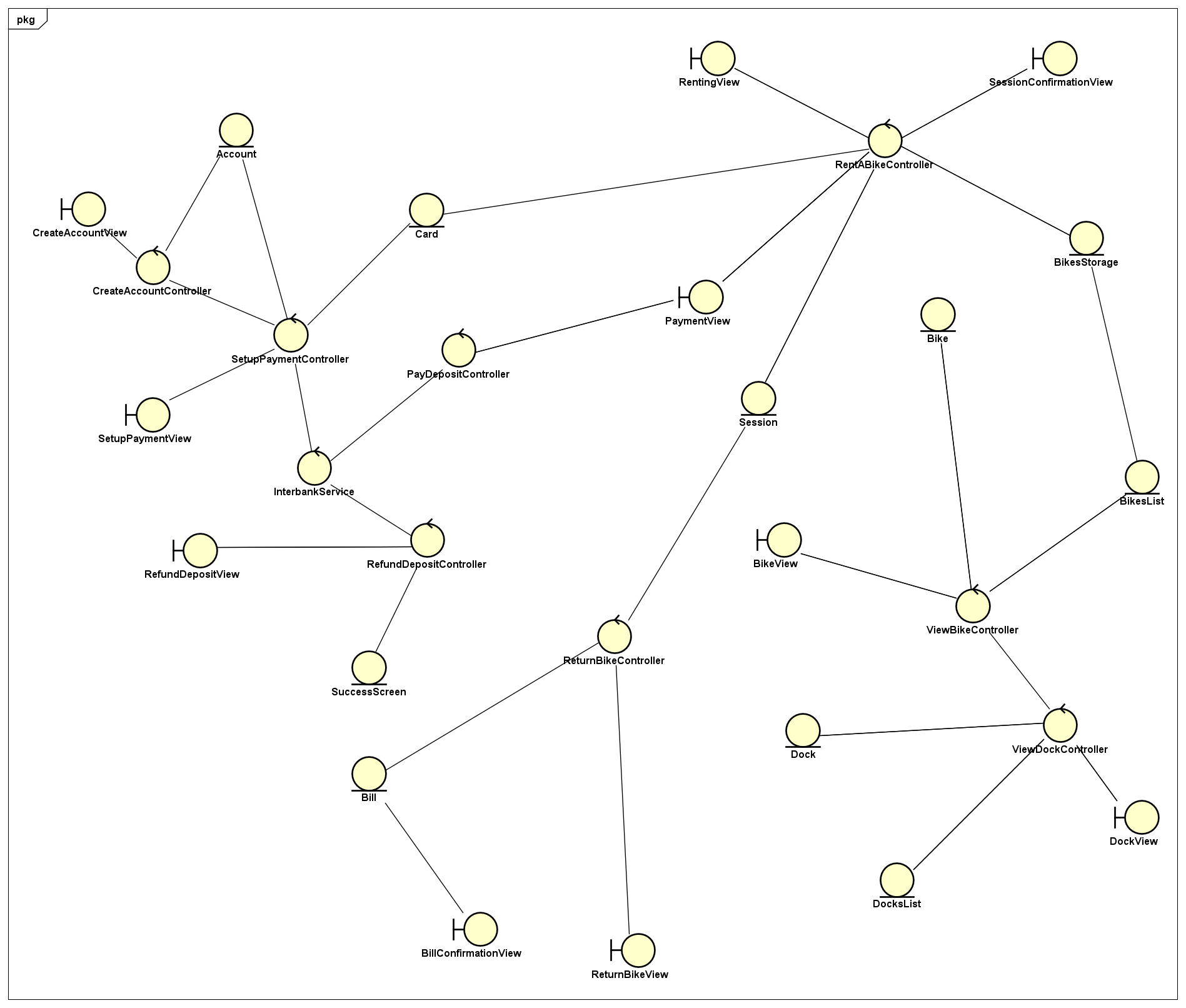








## Unified Analysis Class Diagram



## Security Software Architecture

*<Describe the software components and configuration supporting the security and privacy of the system. Specify the architecture for (1) authentication to validate user identity before allowing access to the system;(2) authorization of users to perform functional activity once logged into the system, (3) encryption protocol to support the business risks and the nature of information, and (4) logging and auditing design, if required.>*

# Detailed Design

## User Interface Design

*<Suppose that you design a Graphical User Interface (GUI)>*

### Screen Configuration Standardization

***Display***

*Number of colors supported:* 16,777,216 colors

*Resolution:* 1366 × 768 𝑝𝑖𝑥𝑒𝑙𝑠

***Screen***

*Location of standard buttons:* At the bottom (vertically) and in the middle

(horizontally) of the frame.

* Affirmative button to the right, background-color: #8C36C0, text-color: #FFFFFF
* Negative button to the left, background-color: #FFFFFF, text-color: #8C36C0

*Location of the messages:* Starting from the top vertically and in the middle

horizontally of the frame down to the bottom.

*Display of the screen title:* The title is located at the center-top of the screen and also on the top-left of the window bar.

*Consistency in expression of alphanumeric numbers:* comma for separator of

thousand while strings only consist of characters, digits, commas, dots, spaces,

underscores, and hyphen symbol.

***Control***

*Size of the text:*

* Title and header text: font: Roboto, style: bold, size: 48px, color: #8C36C0

(medium-dark purple)

* Text in button: font: Roboto, style: black, size: 24px, color: #FFFFFF
* Other text: font: Roboto, style: regular, size: 14px, color: #000000

Input check process: Should check if it is empty or not. Next, check if the input is in the correct format or not

*Sequence of moving the focus:* There will be some popup screen. The popup screen will overlay the current parent screen. Cannot interact with parent screen while the popup screen overlaying. Click outside of the popup screen will close it, return active screen to the parent screen.

After the opening screen, the app will start with splash screen, and then the first screen (homescreen) will appear.

*Sequences of the system screens:*

1. Splash screen (first screen)
2. Home screen – show list of docks
3. Barcode Popup – for inputting the barcode (rent/return bike)
4. Dock View Screen – view all available bikes and dock’s info
5. Bike View Screen – detailed bike’s info
6. Payment Screen – fill in payment method (credit card)
7. Renting Session – all session’s info (start time, current rental fee, bike’s info…)
8. Invoice screen – view order details

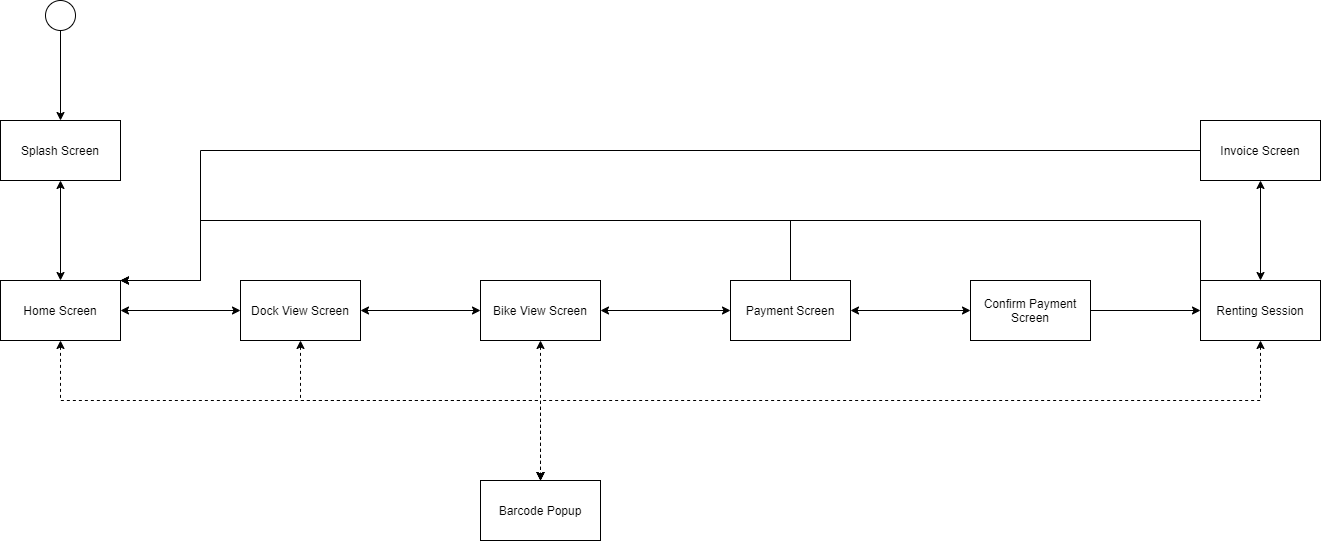
***Direct input from the keyboard***

There will be no shortcuts. There are back buttons to move back to the previous screen located at the top-right corner of each screen. All input fields take input directly from keyboard.

***Error***

A message will be given to notify the users what is the problem.

### Screen Transition Diagrams



### Screen Specifications

**Home Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Home Screen | 6/11/2020 |  |  | Nguyễn Thái An |
|  | | Control | Operation | Function | |
| Enter Barcode Button | Click | Display Barcode Popup Screen | |
| Back button | Click | Back to Splash Screen | |
| Area for entering search | Initial | Display “search” faded word | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Home Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Search field | 50 | String | Black | Left justified |
| Dock’s name | 30 | String | White | Left justified |
| Dock’s address | 50 | String | White | Left justified |
| Bike Number | 4 | Int | White | Left justified |

**Payment Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Payment Screen | 6/11/2020 |  |  | Nguyễn Thái An |
|  | | Control | Operation | Function | |
| Place holder text in input fields | Initial | Display what kind of information need for the field | |
| Back button | Click | Navigate to the previous screen. | |
| Confirm button | Click | Process payment, show Renting Session Screen | |
| Cancel button | Click | Navigate back to Home Screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Payment Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Card Owner | 30 | String | Black | Left justified |
| Card Number | 16 | Numeral | Black | Left justified |
| EXP date | 5 | String | Black | Left justified, MM/YY |
| Security Code | 3 | Numeral | Black | Left justified |

**Dock View Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Dock View Screen | 7/11/2020 |  |  | Bùi Tú Hoàng |
|  | | Control | Operation | Function | |
| Enter Barcode Button | Click | Display Barcode Popup Screen | |
| Back Button | Click | Back to Home Screen | |
| Rent Bike Button | Click | Display Bike View Screen | |
| EBR Home Screen Button | Click | Display Home Screen | |
| Area for Displaying Number of Spots and Bikes | Initial | Displaying Number of Spots and Bikes | |
| Area for Displaying Bike in the Dock | Initial | Displaying Bike in the Dock | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Dock View Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Number of Spot | 3 | Numeral | Black | Right justified |
| Number of Standard Bike | 3 | Numeral | Black | Right justified |
| Number of Twin Bike | 3 | Numeral | Black | Right justified |
| Number of E-Bike | 3 | Numeral | Black | Right justified |
| Name of Bike | 10 | String | Black | Left justified |
| Number of Saddle | 1 | Numeral | Black | Left justified |
| Number of Rear Seat | 1 | Numeral | Black | Left justified |
| Number of Pedal | 1 | Numeral | Black | Left justified |
| Battery Left | 4 | String | Black | Right justified |
| Battery Time Left | 6 | String | Black | Right justified |

**Bike View Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Bike View Screen | 7/11/2020 |  |  | Bùi Tú Hoàng |
|  | | Control | Operation | Function | |
| Enter Barcode Button | Click | Display Barcode Popup Screen | |
| Back Button | Click | Back to the Dock View screen. | |
| Rent Now Button | Click | Display Payment Screen | |
| EBR Home Screen Button | Click | Display Home Screen | |
| Area for displaying the Bike Information | Initial | Displaying the Bike Information and Deposit | |
| Area for displaying Renting Charge | Initial | Displaying Renting Charge | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Bike View Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Bike Code | 8 | String | Black | Left justified |
| Location | 30 | String | Black | Left justified |
| Battery | 4 | String | Black | Left justified |
| Usage | 25 | String | Black | Left justified |
| Deposit | 15 | String | Green | Left justified |
| Charge | 15 | String | Green | Right justified |

**Renting Session Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Renting Session Screen | 7/11/2020 |  |  | Nguyễn Mạnh Khang |
|  | | Control | Operation | Function | |
| Enter Barcode Button | Click | Display Barcode Popup Screen | |
| Back Button | Click | Navigate to the previous screen. | |
| Return Bike button | Click | Display Invoice Screen | |
| EBR Home Screen Button | Click | Display Home Screen | |
| Area for displaying the Session Information | Initial | Displaying the Session Information and Deposit | |
| Area for displaying Renting Fees | Initial | Displaying Renting Fees | |
| Close button | Click | Display Home Screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Bike View Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Bike Code | 8 | String | Black | Left justified |
| Start time | 30 | String | Black | Left justified |
| Battery | 4 | String | Black | Left justified |
| Usage | 25 | String | Black | Left justified |
| Session length | 8 | String | Black | Left justified |
| Charge | 15 | String | Green | Right justified |
| Renting fees | 15 | String | Green | Right justified |

**Invoice Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Invoice Screen | 7/11/2020 |  |  | Nguyễn Mạnh Khang |
|  | | Control | Operation | Function | |
| Return Home Button | Click | Display Payment Screen | |
| Area for displaying the Session Information | Initial | Displaying the Session Information and Fees | |
| Area for displaying Returned | Initial | Displaying Returned money | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Invoice Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Card number | 20 | String | Black | Left justified |
| Start time | 30 | String | Black | Left justified |
| End time | 30 | String | Black | Left justified |
| Total fees | 30 | String | Green | Left justified |
| Deposit | 30 | String | Green | Left justified |
| Returned | 30 | String | Green | Right justified |

**Splash Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Splash Screen | 7/11/2020 |  |  | Vũ Minh Hoàng |
|  | | Control | Operation | Function | |
| Home Screen Button | Click | Display Home Screen | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Dock View Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |

**Barcode Popup**

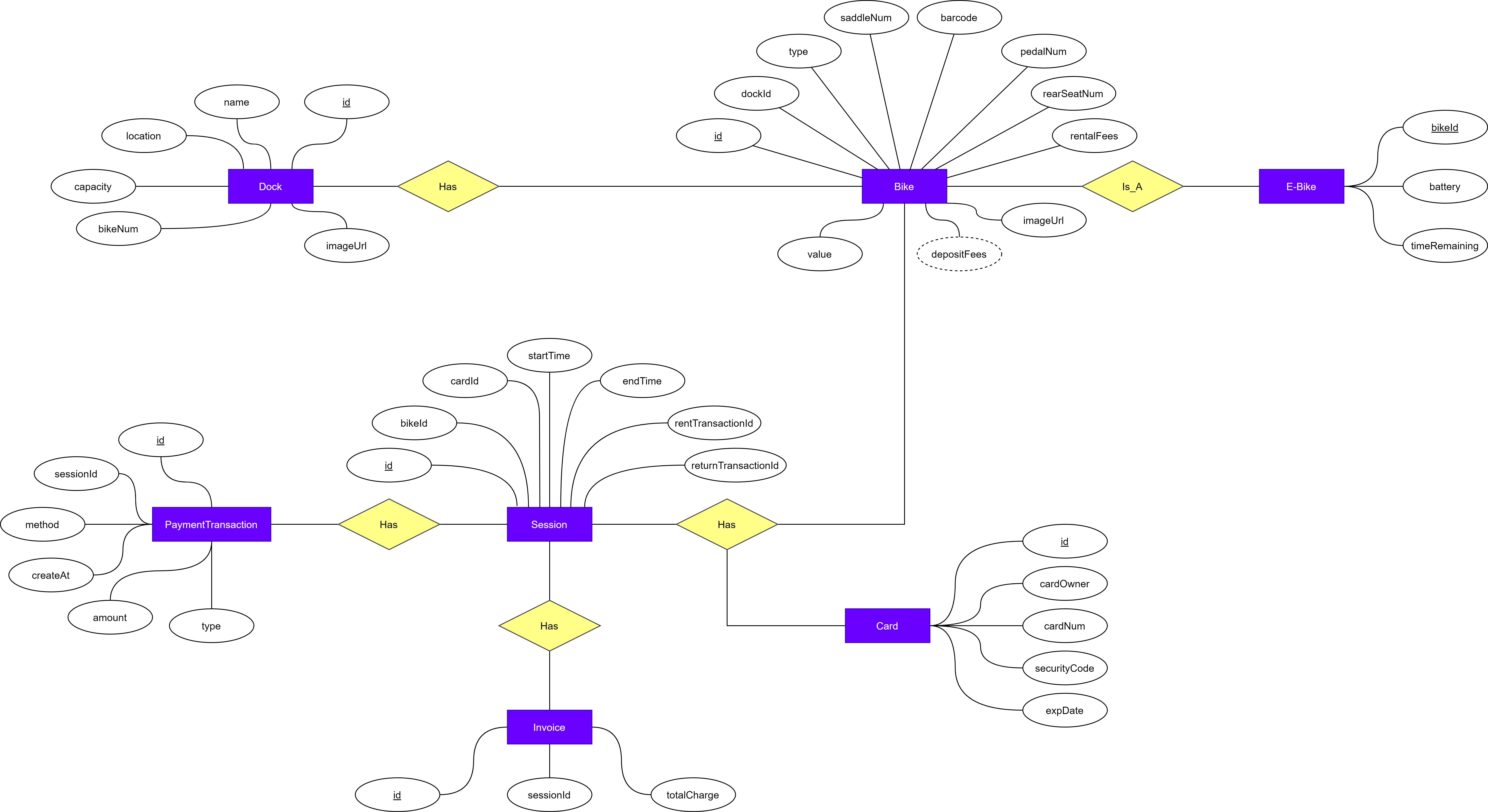
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EBR Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen specification | Barcode Popup | 7/11/2020 |  |  | Vũ Minh Hoàng |
|  | | Control | Operation | Function | |
| Area for Entering Barcode | Initial | Display a text holder for entering barcode for renting bike | |
| Confirm Button | Click | Navigate to the previous screen. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Screen name | Bike View Screen | | | |
| Item name | Number of digits (bytes) | Type | Field attribute | Remarks |
| Barcode | 8 | String | Black | Left justified |

## Data Modeling

### Conceptual Data Modeling

*<E-R Diagram image and description of entities and relationships>*



### Database Design

#### Database Management Systems

*Database Management System: PostgreSQL*

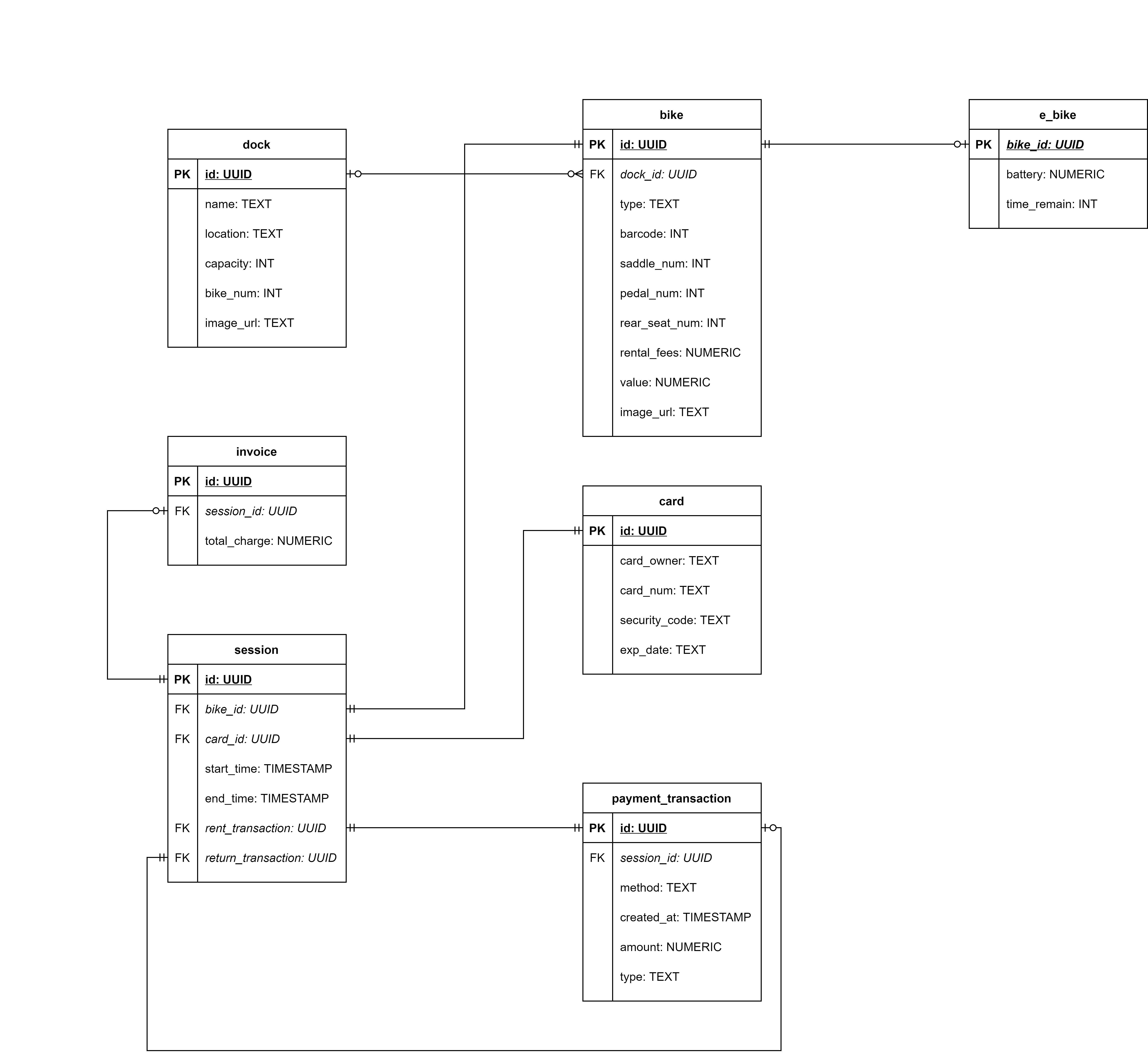
*PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.*

#### Logical Data Model

<

* *Show the process to design database from E-R diagram*
* *Show the diagram of DB design*

*>*



#### Physical Data Model

<

*Give a detail design of each element in the DB diagram. For instance, in a Relational DBMS, give a detail design for each Table and their constraints, illustrated in below table (PK: Primary Key, FK: Foreign Key).*

bike

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | UUID | Yes | UUID |
| 2 |  |  | type | TEXT | Yes | Bike type, e.g, Standard Bike, E-Bike,… |
| 3 |  |  | barcode | INT | Yes | Barcode for renting |
| 4 |  |  | saddle\_num | INT | Yes | Number of saddle |
| 5 |  |  | pedal\_num | INT | Yes | Number of pedal |
| 6 |  |  | rear\_seat\_num | INT | Yes | Number of rear seat |
| 7 |  |  | value | NUMERIC | Yes | Value of the bike |
| 8 |  |  | rental\_fees | NUMERIC | Yes | Rental fees |
| 9 |  | X | dock\_id | UUID | No | Dock ID |
| 10 |  |  | image\_url | TEXT | Yes | URL of image of bike |

e\_bike

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 |  | X | bike\_id | UUID | Yes | ID, same as ID of Bike of which type is E-Bike |
| 2 |  |  | battery | NUMERIC | Yes | Battery left percentage |
| 3 |  |  | time\_remain | INT | Yes | Time remaining |

dock

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | UUID | Yes | UUID |
| 2 |  |  | name | TEXT | Yes | Name of the dock |
| 3 |  |  | location | TEXT | Yes | Address of the dock |
| 4 |  |  | capacity | INT | Yes | Total parking slot |
| 5 |  |  | bike\_num | INT | Yes | Number of bike in the dock |
| 6 |  |  | image\_url | TEXT | Yes | URL of image of dock |

session

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | UUID | Yes | UUID |
| 2 |  | X | card\_id | UUID | Yes | Card Id |
| 3 |  | X | bike\_id | UUID | Yes | Bike Id |
| 4 |  |  | start\_time | TIMESTAMP | Yes | Start time of session |
| 5 |  |  | end\_time | TIMESTAMP | No | End time of session |
| 6 |  | X | rent\_transaction\_id | UUID | Yes | Renting transaction ID |
| 7 |  | X | return\_transaction\_id | UUID | No | Returning transaction ID |

payment\_transaction

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | UUID | Yes | UUID |
| 2 |  | X | session\_id | UUID | Yes | Session ID |
| 3 |  |  | type | TEXT | Yes | Type of the transaction |
| 4 |  |  | create\_at | TIMESTAMP | Yes | Time when the transaction is created |
| 5 |  |  | amount | NUMERIC | Yes | Transaction amount |
| 6 |  |  | method | TEXT | Yes | Method of payment |

card

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | UUID | Yes | UUID |
| 2 |  |  | card\_owner | TEXT | Yes | Full name of the card owner |
| 3 |  |  | card\_num | TEXT | Yes | Card number, hashed |
| 4 |  |  | security\_code | TEXT | Yes | Security code, hashed |
| 5 |  |  | exp\_date | TEXT | Yes | Expiration date |

invoice

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data Type | Mandatory | Description |
| 1 | X |  | id | UUID | Yes | UUID |
| 2 |  | X | session\_id | UUID | Yes | Session ID |
| 3 |  |  | total\_charge | NUMERIC | Yes | Total charge money |

**SQL Script**

--CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

Create table dock(

id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

name text NOT NULL,

location text not null,

capacity integer not null,

bike\_num integer not null,

image\_url text DEFAULT ''::text,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

constraint dock\_pkey PRIMARY KEY (id)

);

CREATE TABLE bike (

id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

type text NOT NULL,

barcode integer not null,

saddle\_num integer not null,

pedal\_num integer not null,

rear\_seat\_num integer not null,

value integer not null,

rental\_fees integer not null,

deposit\_fees integer not null,

image\_url text DEFAULT ''::text,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

description text,

dock\_id uuid not null,

CONSTRAINT bike\_pkey PRIMARY KEY (id),

constraint bike\_fk foreign key (dock\_id)

references dock (id) on delete cascade on update no action

);

create table e\_bike(

bike\_id uuid not null,

battery numeric not null,

time\_remain int not null,

image\_url text DEFAULT ''::text,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

constraint e\_bike\_fk foreign key (bike\_id)

references bike (id) on delete cascade

);

Create table card(

id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

card\_owner text not null,

card\_num text not null,

security\_code text not null,

exp\_date text not null,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

constraint card\_pkey PRIMARY KEY (id)

);

create table payment\_transaction(

id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

session\_id uuid not null,

type text not null,

amount numeric not null,

method text not null,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

constraint payment\_transaction\_pkey PRIMARY KEY (id)

--constraint payment\_transaction\_fk foreign key(session\_id) references session (id)

);

create table session(

id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

card\_id uuid NOT NULL,

bike\_id uuid not null,

rent\_transactionId uuid not null,

return\_transactionId uuid not null,

start\_time timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

end\_time timestamp with time zone,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

constraint session\_pkey PRIMARY KEY (id),

constraint session\_fk1 foreign key(card\_id) references Card (id),

constraint session\_fk2 foreign key(bike\_id) references Bike (id),

constraint session\_fk3 foreign key(rent\_transactionId) references payment\_transaction (id),

constraint session\_fk4 foreign key(return\_transactionId) references payment\_transaction (id)

);

Create table invoice(

id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

session\_id uuid not null,

total\_charde numeric not null,

created\_at timestamp with time zone DEFAULT CURRENT\_TIMESTAMP,

constraint invoice\_pkey PRIMARY KEY (id),

constraint invoice\_fk foreign key (session\_id) references session (id)

);

## Non-Database Management System Files

*<Provide the detailed description of all non-DBMS files if any and include a narrative description of the usage of each file that identifies if the file is used for input, output, or both, and if the file is a temporary file. Also provide an indication of which modules read and write the file and include file structures (refer to the data dictionary). As appropriate, the file structure information should include the following:*

*• Record structures, record keys or indexes, and data elements referenced within the records*

*• Record length (fixed or maximum variable length) and blocking factors*

*• Access method (e.g., index sequential, virtual sequential, random access, etc.)*

*• Estimate of the file size or volume of data within the file, including overhead resulting from file access methods*

*• Definition of the update frequency of the file (If the file is part of an online transaction-based system, provide the estimated number of transactions per unit of time, and the statistical mean, mode, and distribution of those transactions.)*

*• Backup and recovery specifications>*

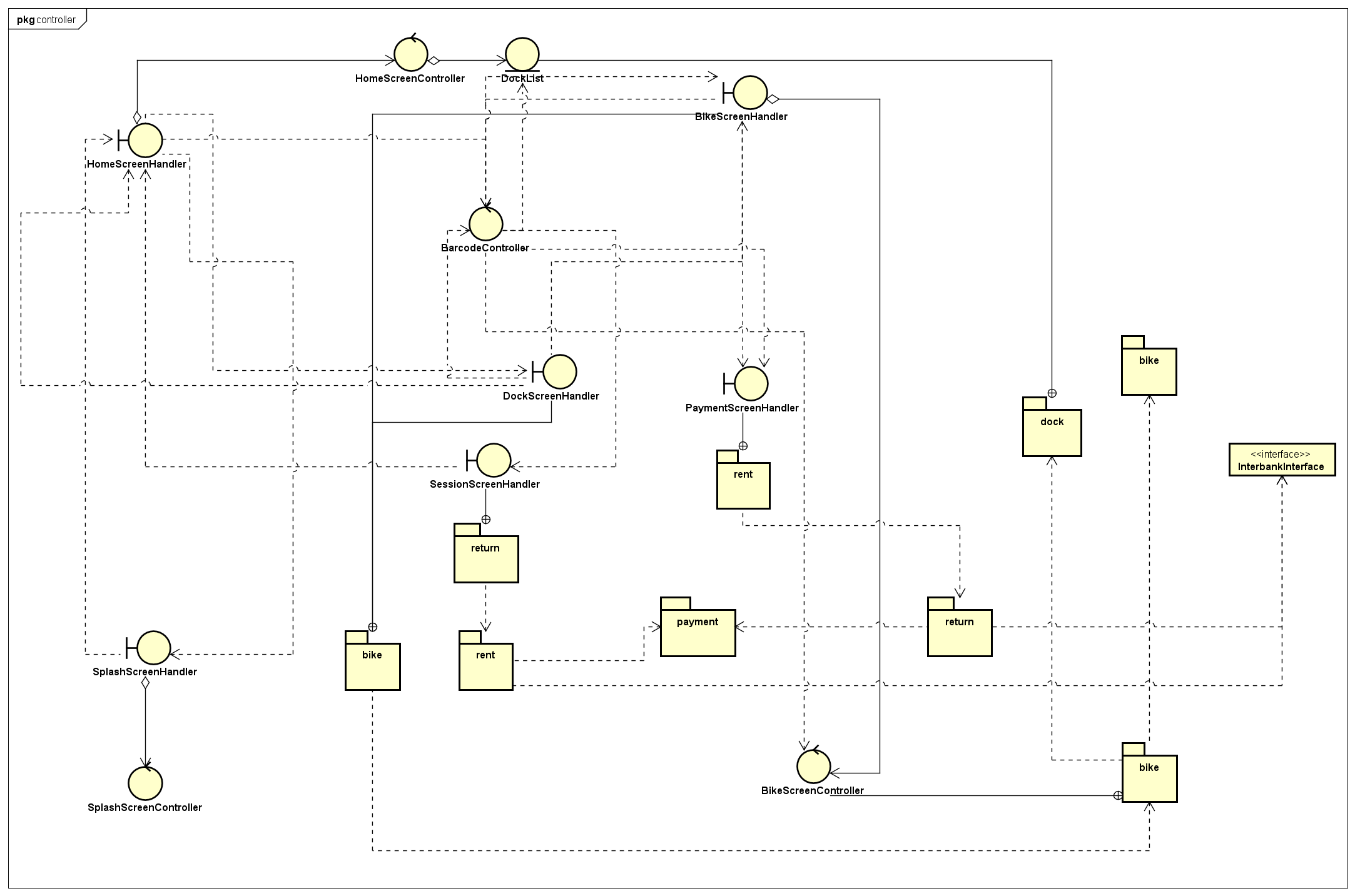
## Class Design

### General Class Diagram

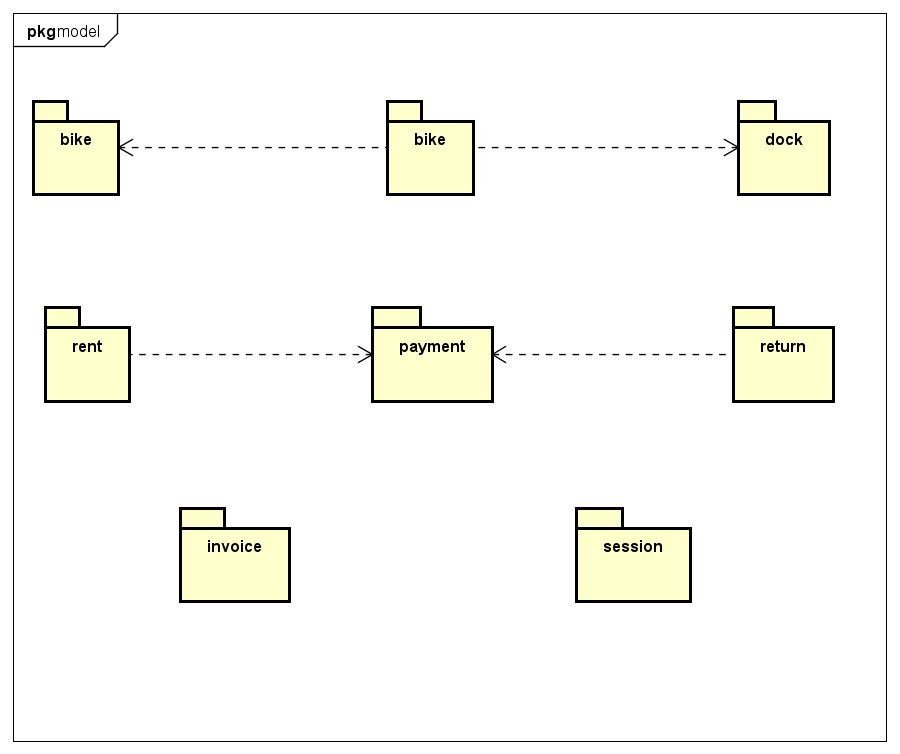


### Class Diagrams

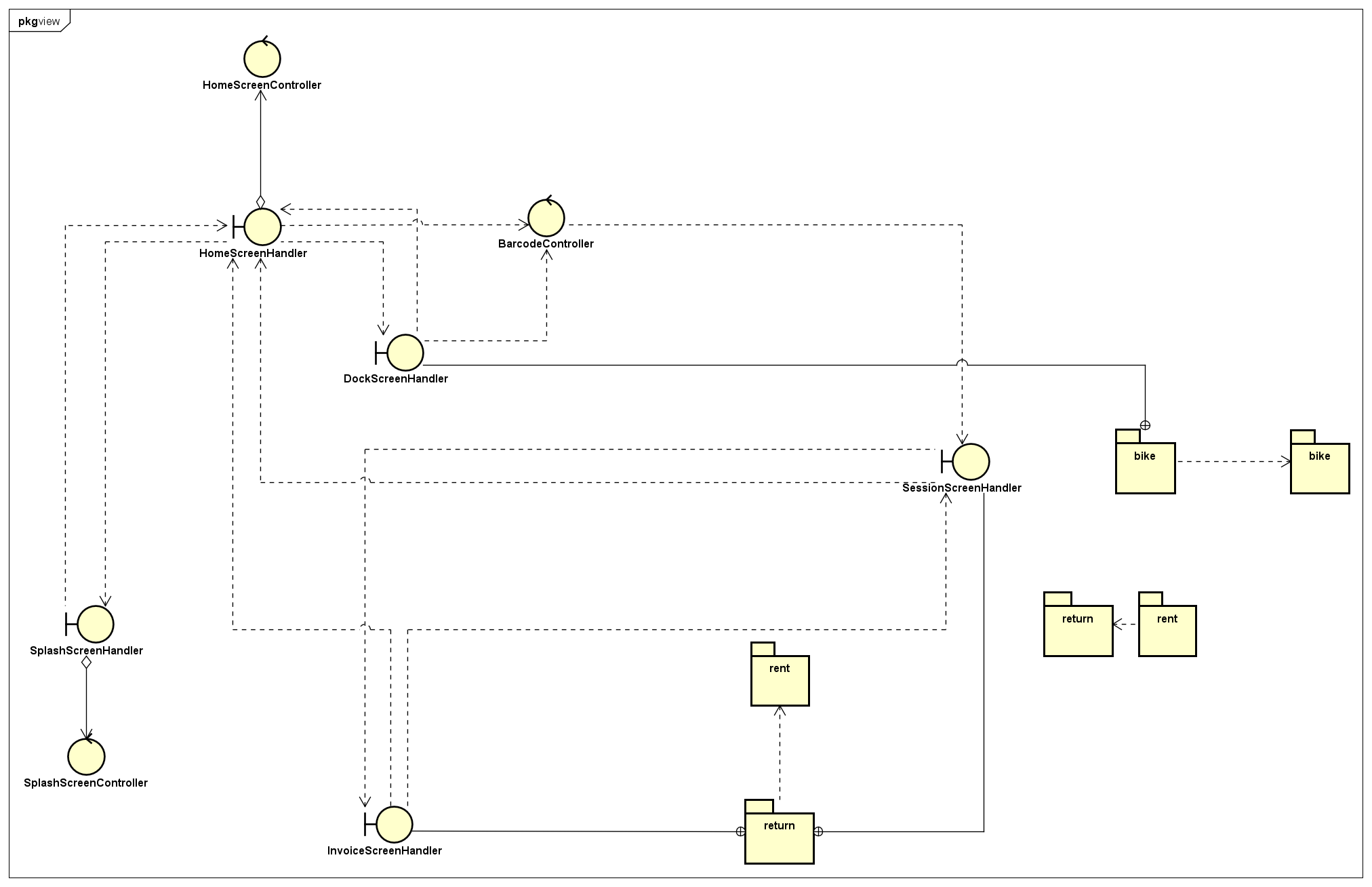
#### Class Diagram for Package Controller



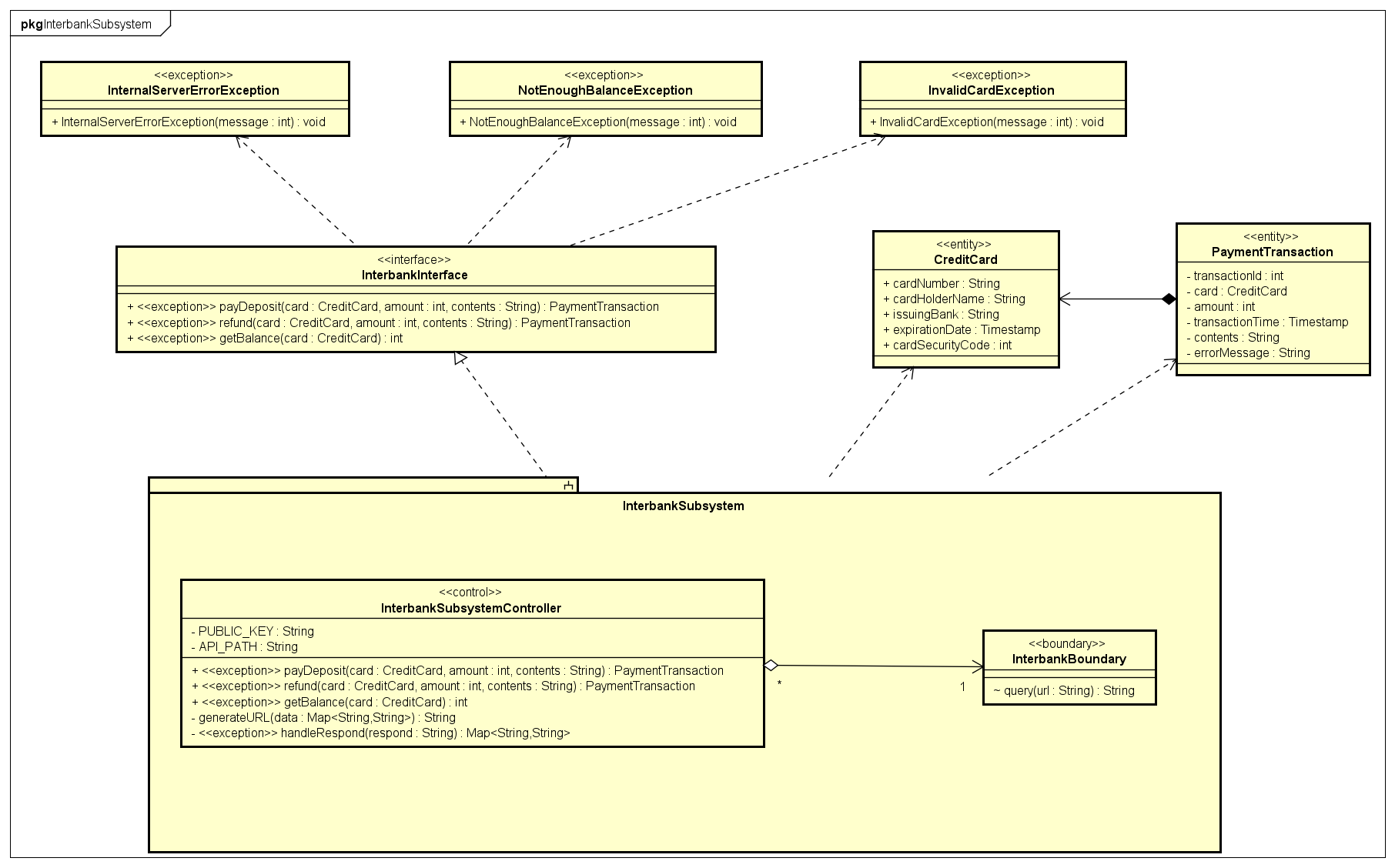
#### Class Diagram for Package Model



#### Class Diagram for Package View

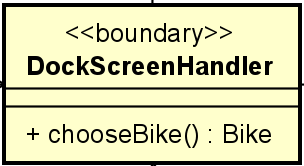


#### Class Diagram for Subsystem Interbank



### Class Design

#### Class “DockScreenHandler”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return type** | **Description** |
| 1 | chooseBike | Bike | forward bike to the BikeScreenHandler |

Parameter:

None

Exception:

None

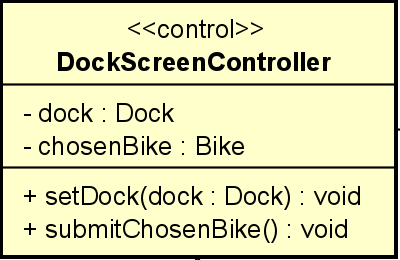
**Method**

None

**State**

None

#### Class “DockScreenController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Name** | **Data type** | **Default value** | **Description** |
| 1 | dock | Dock | NULL | store the dock it works with |
| 2 | chosenBike | Bike | NULL | the bike user chooses from UI |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Description |
| 1 | setDock | void | set value for dock |
| 2 | submitChosenBike | void | send the chosen bike to other object |

Parameter:

dock – the dock instance

Exception:

None

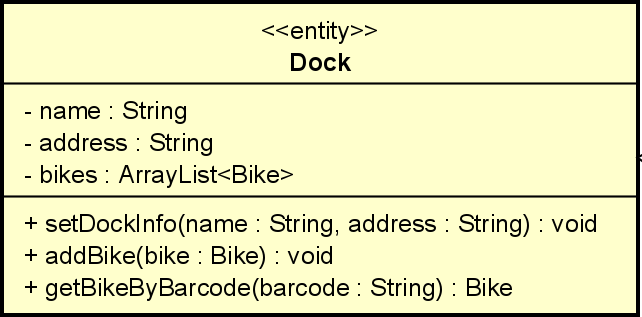
**Method**

None

**State**

None

#### Class “Dock”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | name | String | NULL | name of the dock |
| 2 | address | String | NULL | location of the dock |
| 3 | bikes | ArrayList<Bike> | empty ArrayList | all available bikes |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Description |
| 1 | setDockInfo | void | update dock |
| 2 | addBike | void | add new bike to dock |
| 3 | getBikeByBarcode | Bike | search bike by barcode |

Parameter:

name – name of the dock

address – dock’s address

bike – the bike instance

barcode – bike’s barcode

Exception:

None

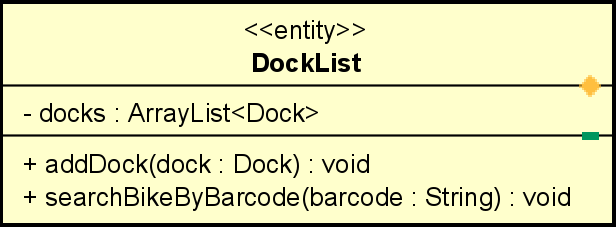
**Method**

None

**State**

None

#### Class “DockList”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | name | String | NULL | name of the dock |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Description |
| 1 | addDock | void | add new Dock |
| 2 | searchBikeByBarcode | Bike | search and return bike with matching barcode |

Parameter:

dock – dock’s instance

barcode – the string barcode of a bike

Exception:

None

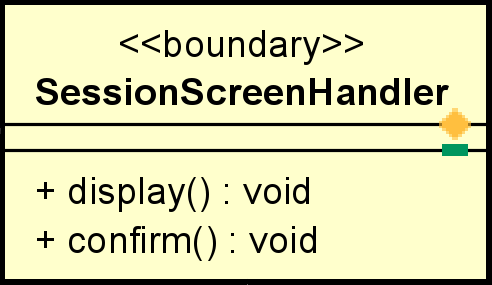
**Method**

None

**State**

None

#### Class “SessionScreenHandler”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Description |
| 1 | display | void | Display renting session |
| 2 | confirm | void | Handler for user to move to invoice screen |

Parameter:

None

Exception:

None

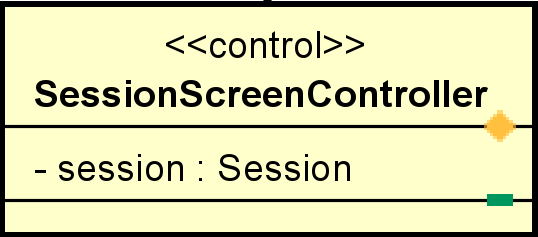
**Method**

None

**State**

None

#### Class “SessionScreenController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | session | Session | NULL | Renting Session |

**Operation**

None

Parameter:

None

Exception:

None

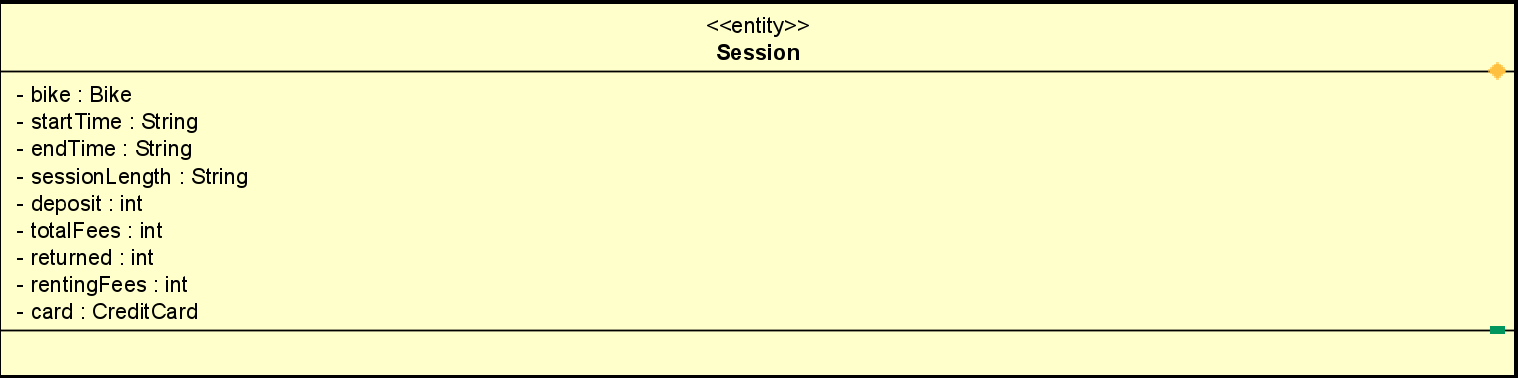
**Method**

None

**State**

None

#### Class “Session”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | bike | Bike | NULL | Bike info |
| 2 | startTime | String | NULL | Renting start time |
| 3 | endTime | String | NULL | Renting end time |
| 4 | sessionLength | String | NULL | Renting length time |
| 5 | totalFees | int | 0 | total price |
| 6 | deposit | int | 0 | deposit |
| 7 | returned | Int | 0 | Returned money |
| 8 | rentingFees | int | 0 | Renting Fee |
| 9 | card | CreditCard | NULL | Credit card info |

**Operation**

none

Parameter:

none

Exception:

None

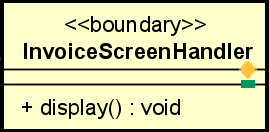
**Method**

None

**State**

None

#### Class “InvoiceScreenHandler”



**Attribute**

None

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | display | void | Display invoice |

Parameter:

None

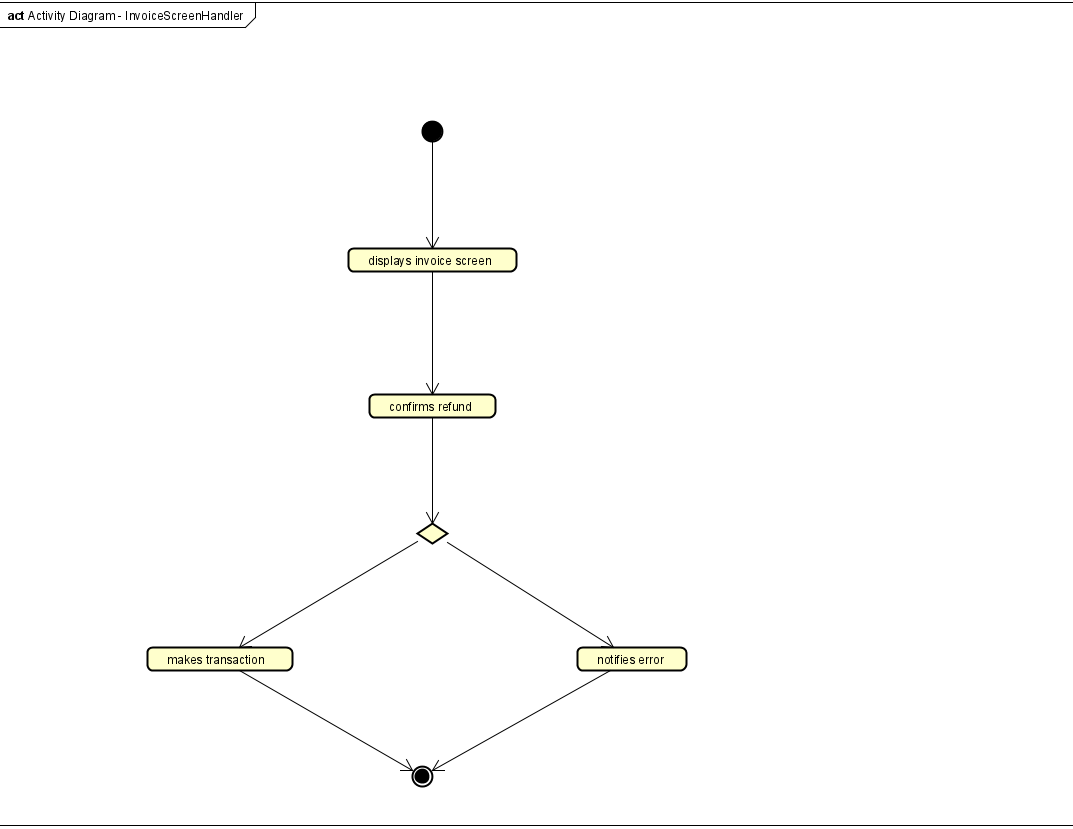
Exception:

InternalServerErrorException

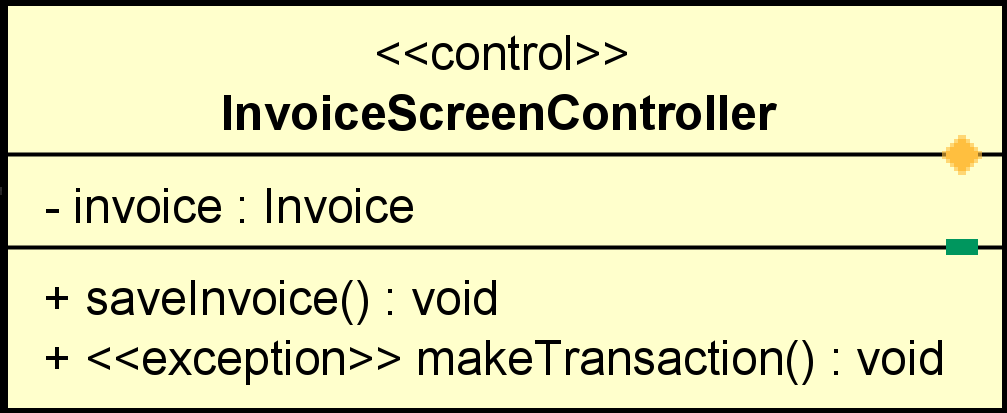
**Method**

None

**State**



#### Class “InvoiceScreenController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | invoice | Invoice | NULL | Invoice info |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | saveInvoice | void | Save invoice history |
| 2 | makeTransaction | void | Make transaction |

Parameter:

None

Exception:

None

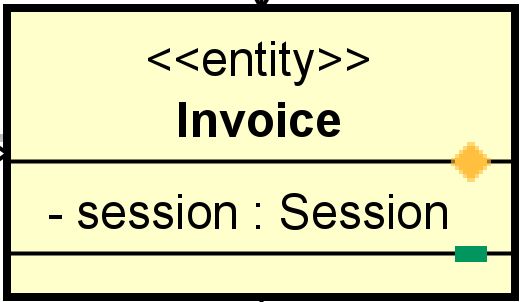
**Method**

None

**State**

None

#### Class “Invoice”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | session | Session | NULL | Invoice info |

**Operation**

None

Parameter:

None

Exception:

None

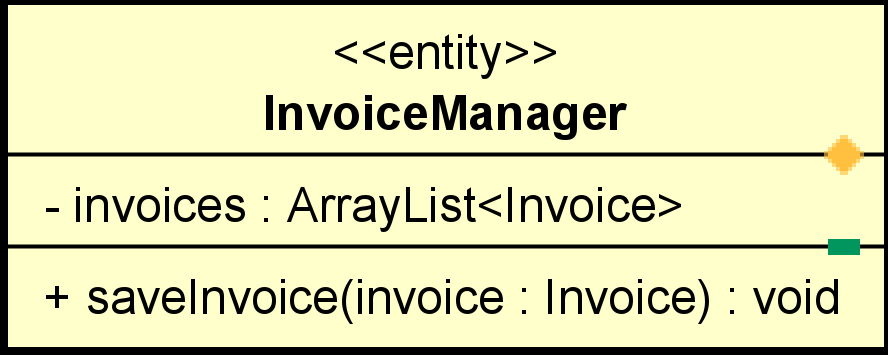
**Method**

None

**State**

None

#### Class “InvoiceManager”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | invoices | ArrayList | Empty array | List invoices |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | saveInvoice | void | Save invoice history |

Parameter:

None

Exception:

None

**Method**

None

**State**

None

#### Class “HomeScreenHandler”

**Attribute**: None

**Operation**:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return type** | **Description** |
| 1 | chooseDock | Dock | handle dock tab that chosen by user |

**Parameter**:

None

**Exception**:

None

**Method**:

None

**State**:

None

#### Class “HomeScreenController”

**Attribute**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Default Value** | **Description** |
| 1 | list | DockList | empty ArrayList | List of docks |
| 2 | chosenDock | Dock | NULL | Particular dock that chosen by user |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return type** | **Description** |
| 1 | setDockList | void | Set the list of docks |
| 2 | submitChosenDock | void | Submit the dock chosen by user |
| 3 | searchDock | void | Search dock by name or address |

**Parameter**:

list: list of docks

value: the string that user want to search

**Exception**:

None

**Method**:

None

**State**:

None

#### Class “BikeScreenHandler”

**Attribute:** None

**Operation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return type** | **Description** |
| 1 | rentBike | void | Rent bike |

**Parameter:**

None

**Exception:**

None

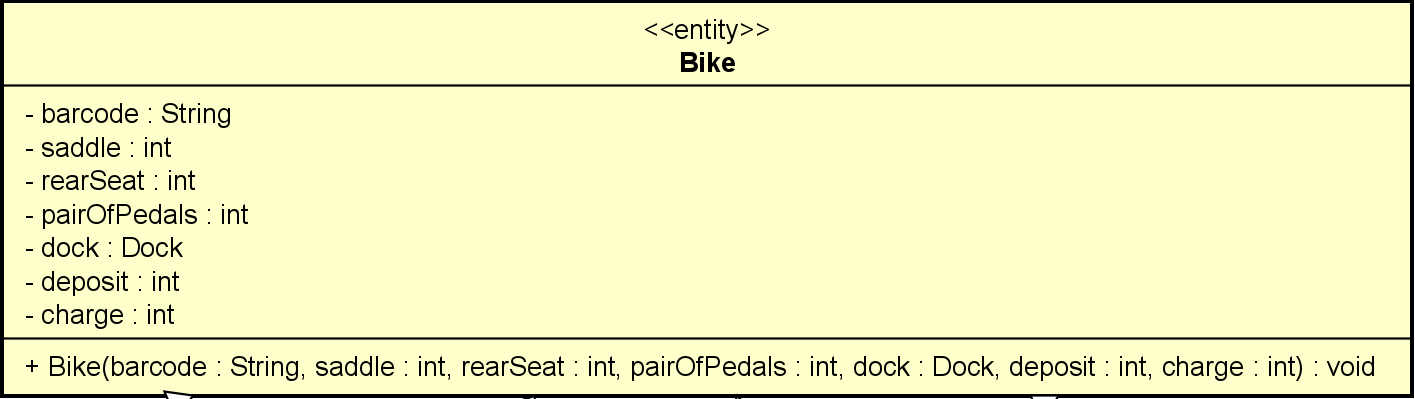
**Method:**

None

**State:**

None

#### Class “Bike”

 **Attribute**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Default Value** | **Description** |
| 1 | barcode | String | NULL | Barcode of the bike |
| 2 | saddle | int | 1 | Number of saddle |
| 3 | rearSeat | int | 1 | Number of rear seat |
| 4 | pairOfPedals | int | 1 | Number of pair of pedals |
| 5 | dock | Dock | NULL | Dock where bike is locating |
| 6 | deposit | int | NULL | Deposit money of bike |
| 7 | charge | int | NULL | Charge of renting |

**Operation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return type** | **Description** |
| 1 | bike | void | Initialize the info of the bike |

**Parameter**:

bike: Barcode of the bike

saddle: Number of saddle

rearSeat: Number of rear seat

pairOfPedals: Number of pair of pedals

dock: Dock where bike is locating

deposit: Deposit money of bike

charge: Charge of renting

**Exception:**

None

**Method**:

None

**State**:

None

#### Class “StandardElectricalBike”

**Attribute**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Default Value** | **Description** |
| 1 | battery | int | 100% | The percentage of the battery |
| 2 | time | String | NULL | Time left before running out of battery |

**Operation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Return type** | **Description** |
| 1 | setBattery | void | Set the percentage of the battery of E-Bike |
| 2 | setTimeLeft | void | Set the time left before running out of battery of E-Bike |

**Parameter**:

battery: The percentage of the battery

time: Time left before running out of battery

**Exception:**

None

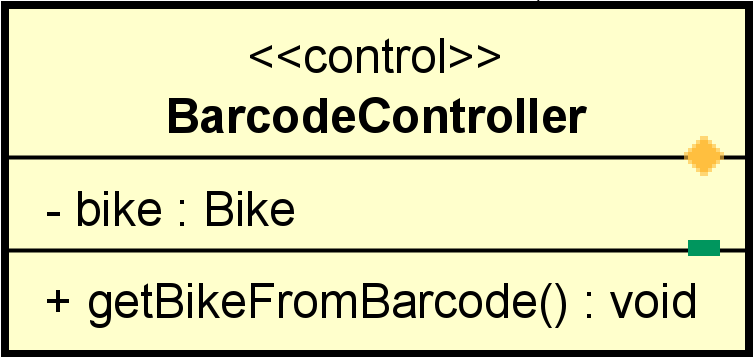
**Method**:

None

**State**:

None

#### Class “BarcodeController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | bike | Bike | NULL | The result of barcode searching |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | getBikeFromBarcode | void | Get bike using barcode from DockList |

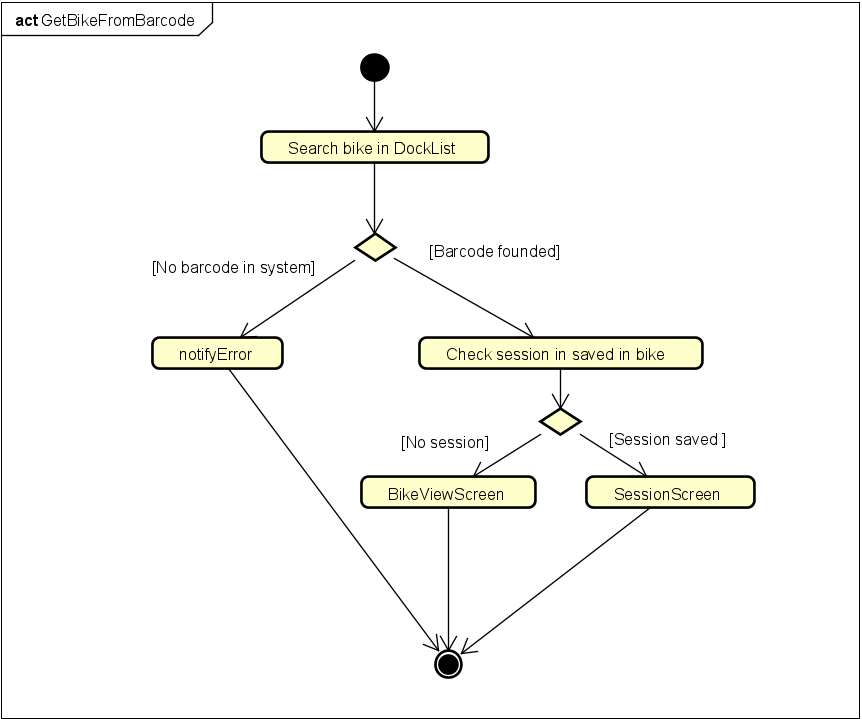
Parameter:

None

Exception:

None

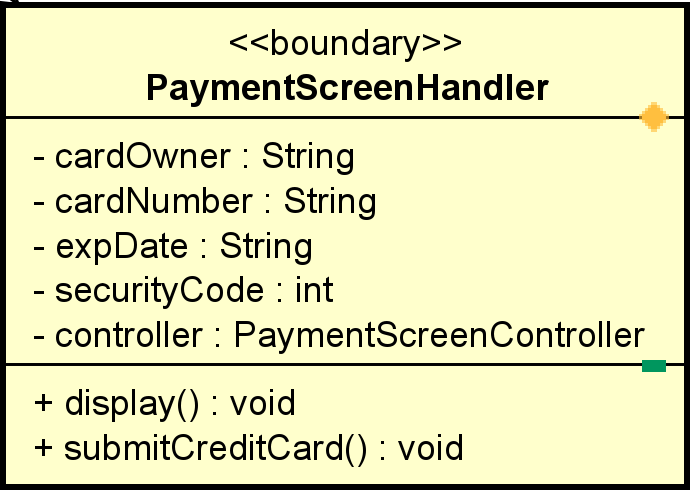
**Method**

****

**State**

None

#### Class “PaymentScreenHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | controller | PaymentScreenController | NULL | Controller for this screen |
| 2 | cardOwner | String | Empty | Card owner |
| 3 | cardNumber | String | Empty | Card number |
| 4 | expDate | String | Empty | Expired Date |
| 5 | securityCode | Int | Empty | Security Date |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 3 | display | void | Display screen |
| 4 | submitCreditCard | void | Send card to next handler |

Parameter:

None

Exception:

None

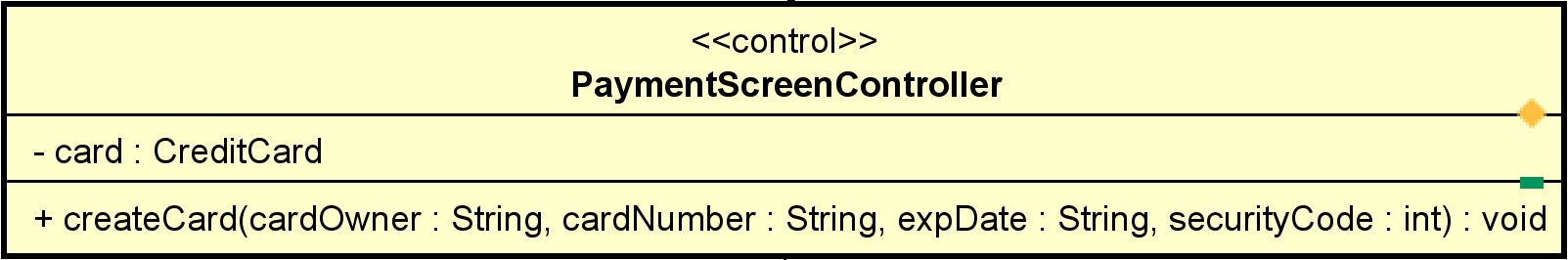
**Method**

None

**State**

None

#### Class “PaymentScreenController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | card | CreditCard | NULL | Card for payment |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | createCard | void | Create an instance of CreditCard |

Parameter:

cardOwner – card owner

cardNumber – card number

expDate – expired date

securityCode – security code

Exception:

None

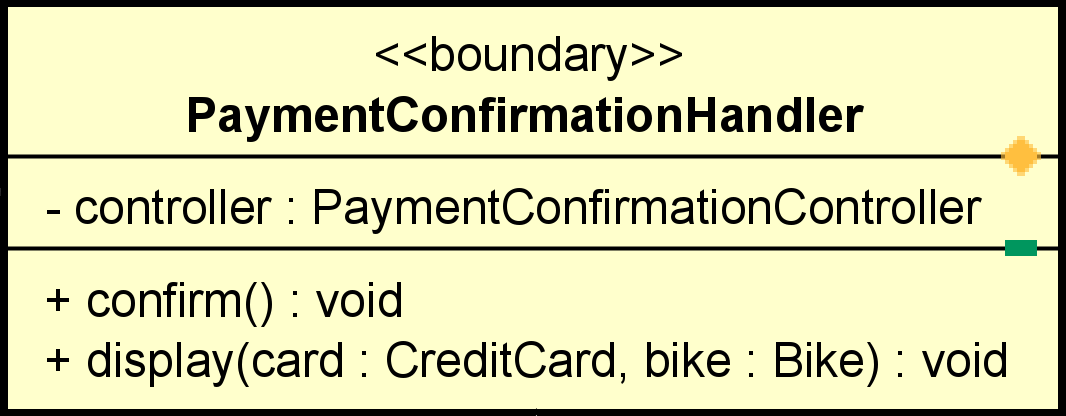
**Method**

None

**State**

None

#### Class “PaymentConfirmationHandler”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | controller | PaymentConfirmationController | NULL | Controller for this screen |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | display | void | Display screen |
| 2 | confirm | void | Move to next screen |

Parameter:

None

Exception:

None

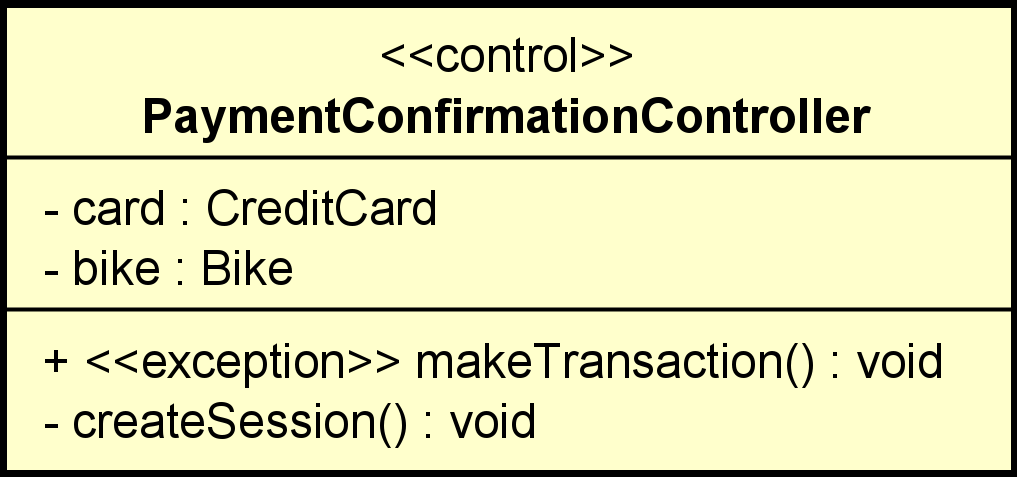
**Method**

None

**State**

None

#### Class “PaymentConfirmationController”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | card | CreditCard | NULL | Card for payment |
| 2 | bike | Bike | NULL | Bike for renting |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | makeTransaction | void | Make transaction |
| 2 | createSession | void | Create new Session |

Parameter:

None

Exception:

InvalidCardException

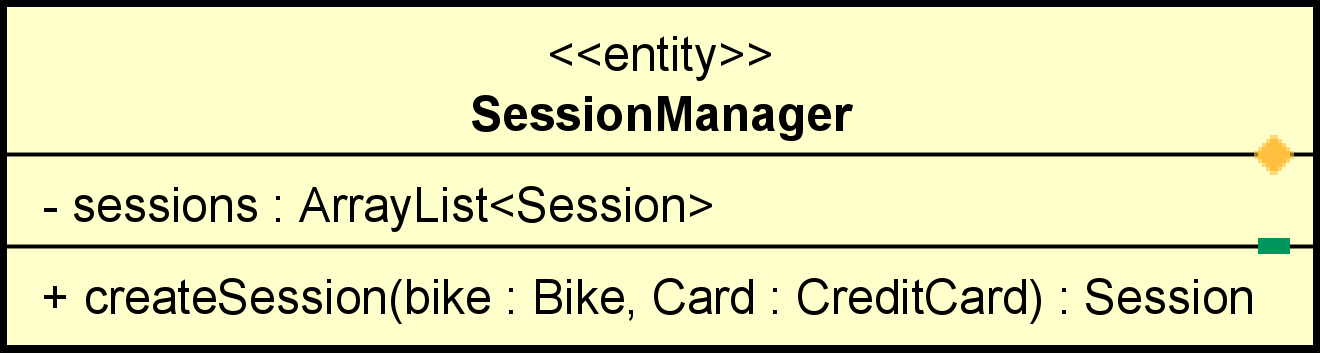
NotEnoughBalanceException

InternalServerErrorException

**Method**

None

#### Class “SessionManager”



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Data type | Default value | Description |
| 1 | sessions | ArrayList<Session> | Empty ArrayList | List of sessions |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Return type | Desciption |
| 1 | createSession | Session | Create new session and add to list |

Parameter:

bike – the rented bike

card – the used card

Exception:

None

**Method**

None

# Design Considerations

***<Describe issues which need to be addressed or resolved before attempting to devise a complete design solution>***

## Goals and Guidelines

*<Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system and its software.*

*Examples of such goals might be: an emphasis on speed versus memory use; or working, looking, or “feeling” like an existing product.*

*Guidelines include coding guidelines and conventions.*

*For each such goal or guideline, describe the reason for its desirability unless it is implicitly obvious.*

*Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system (e.g., choice of which specific product to use)*>

## Architectural Strategies

*<Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off.*

*Examples of design decisions might concern (but are not limited to) things like the following:*

*• Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)*

*• Reuse of existing software components to implement various parts/features of the system*

*• Future plans for extending or enhancing the software*

*• User interface paradigms (or system input and output models)*

*• Hardware and/or software interface paradigms*

*• Error detection and recovery*

*• Memory management policies*

*• External databases and/or data storage management and persistence*

*• Distributed data or control over a network*

*• Generalized approaches to control*

*• Concurrency and synchronization*

*• Communication mechanisms*

*• Management of other resources*

>

## Coupling and Cohesion

*<Evaluate your design and describe which levels of coupling and cohesion that your design is at. Give proofs for your assumptions. Explain if there is any special design or exceptions>*

## Design Principles

*<Does your design follow the SOLID principles for the new requirements/changing requirements? Give proofs for your assumptions. Explain if there is any special design or exceptions>*

## Design Patterns

*<Do you use any design patterns for your design? If yes, describe detailly why you use those design patterns? Describe in detail on the solutions and how to implement each design pattern>*