HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

EcoBike Rental

Subject: ITSS Software Development

Group 6

Vũ Hồng Hải - 20205150

Bùi Duy Hiển - 20200208

Nguyễn Tuấn Hiệp - 20205151

Hà Huy Hoàng - 20205152

*Hanoi,* *August 2023*

Table of Contents

Table of Contents 1

1 Introduction 5

1.1 Objective 5

1.2 Scope 5

1.3 Glossary 5

1.4 References 5

2 Overall Description 7

2.1 General Overview 7

2.2 Assumptions/Constraints/Risks 7

2.2.1 Assumptions 7

2.2.2 Constraints 7

2.2.3 Risks 8

3 System Architecture and Architecture Design 9

3.1 Architectural Patterns 9

3.2 Interaction Diagrams 9

3.3 Analysis Class Diagrams 9

3.4 Unified Analysis Class Diagram 9

3.5 Security Software Architecture 9

4 Detailed Design 10

4.1 User Interface Design 10

4.1.1 Screen Configuration Standardization 10

4.1.2 Screen Transition Diagrams 10

4.1.3 Screen Specifications 10

4.2 Data Modeling 10

4.2.1 Conceptual Data Modeling 10

4.2.2 Database Design 10

4.3 Non-Database Management System Files 11

4.4 Class Design 11

4.4.1 General Class Diagram 11

4.4.2 Class Diagrams 11

4.4.3 Class Design 11

5 Design Considerations 13

5.1 Goals and Guidelines 13

5.2 Architectural Strategies 13

5.3 Coupling and Cohesion 14

5.4 Design Principles 14

5.5 Design Patterns 14

**List of Figures**

**No table of figures entries found.**

**List of Tables**

No table of figures entries found.

# Introduction

The task involves developing an EcoBike Rental software intended for use in the Ecopark township. The Software Design Document serves as a comprehensive guide for the software development process, outlining the specific instructions for constructing the software. The proposed designs align closely with the stipulated requirements outlined in the Software Requirements Specifications document created for the project. This document encompasses both descriptive and visual representations of the software design, encompassing interaction diagrams, system and subsystem architecture, user interface layout, database structure, class diagrams, and supplementary details derived from the requirements.

## Objective

The purpose of this document is to present a detailed description of the designs of the EcoBike Rental Software. This document is intended for the programming group in Group 2, to use the designs as guidelines to implement the project.

## Scope

Our application will provide managers with automatic bike rental and return services in dock lots. But in this problem, we will skip the login part and focus on the function related to renting and returning the bike. The goal is to create a fully automated management application that provides customers with the most basic functionalit A simple way the user after accessing, will see a list of dock lots. After that, users can view information about dock lots and perform bike rental functions through bike codes. After the initial payment process, customers can use the registered bike. Before or during use, customers can also view information about their rented bike such as bike status, battery status. At the end of the usage period, the user will be able to perform the function of returning the bike and making the final payment.

## Glossary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***No*** | ***Term*** | ***Explanation*** | ***Example*** | ***Note*** |
| 1 | token | A piece of data created by server, and contains the user's information, as well as a special token code that user can pass to the server with every method that supports authentication, instead of passing a username and password directly. | JSON Web Token (JWT) | Compact, URL-safe and usable especially in web browser single sign-on (SSO) context. |
| 2 | Use case | Show the interaction between the user and the system, thereby expressing the user's requirements | Return Bike | Use case performs the bike return function |
| 3 | Actor | Only users or external objects interact with the system | Customer | As a system user |
| 4 | Use case diagrams | Demonstrate the different ways that a user might interact with a system. |  |  |
| 5 | Activity Diagram | Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system |  |  |
|  |  |  |  |  |

## References

|  |  |
| --- | --- |
| [1] | Centers for Medicare & Medicaid Services, "System Design Document Template," [Online]. Available: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx. |

# Overall Description

## General Overview

The software has 2 actors: Customer, Interbank. Customer will be the main user of the system, interacting with basic functions such as renting and returning bikes. The Interbank actor will represent the interaction between the system and the bank to perform payment-related work.

## 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

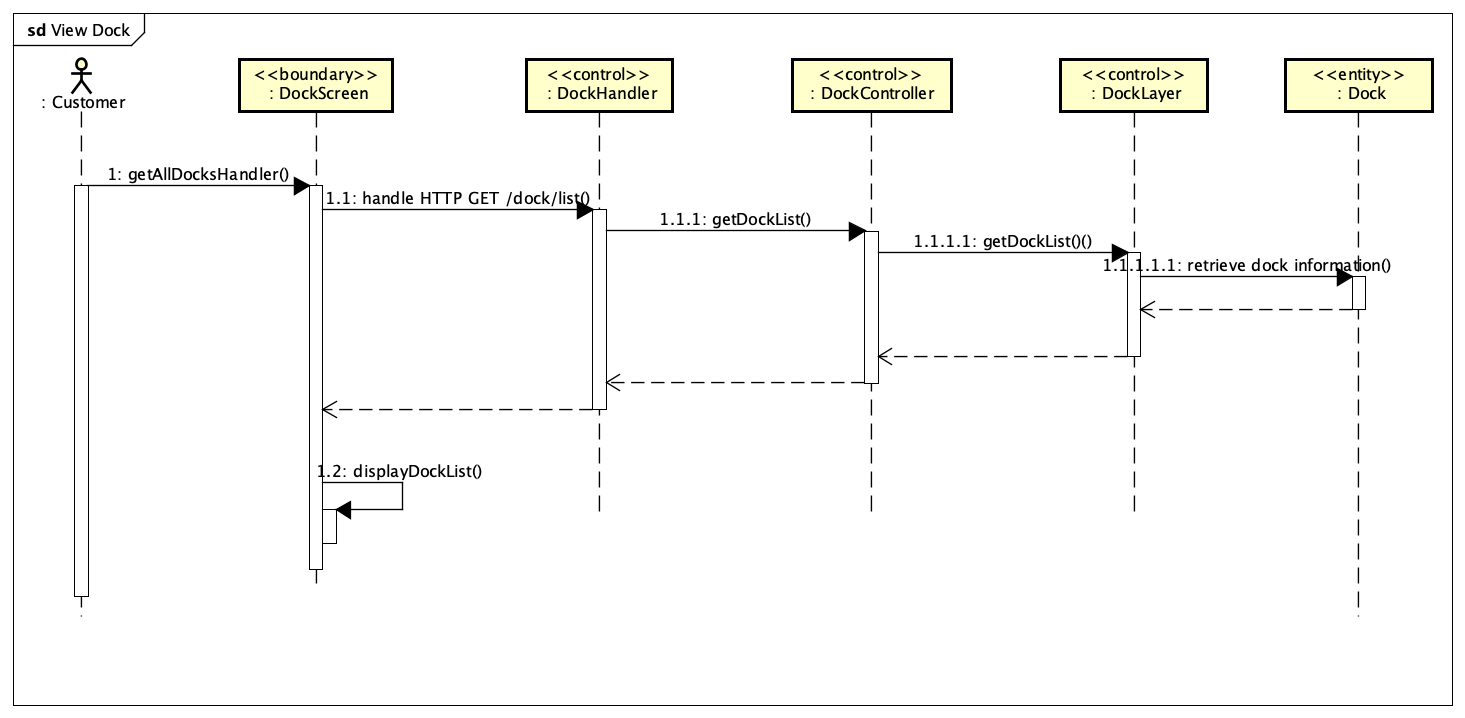
*<Describe any risks associated with the system design and proposed mitigation strategies.>*

# System Architecture and Architecture Design

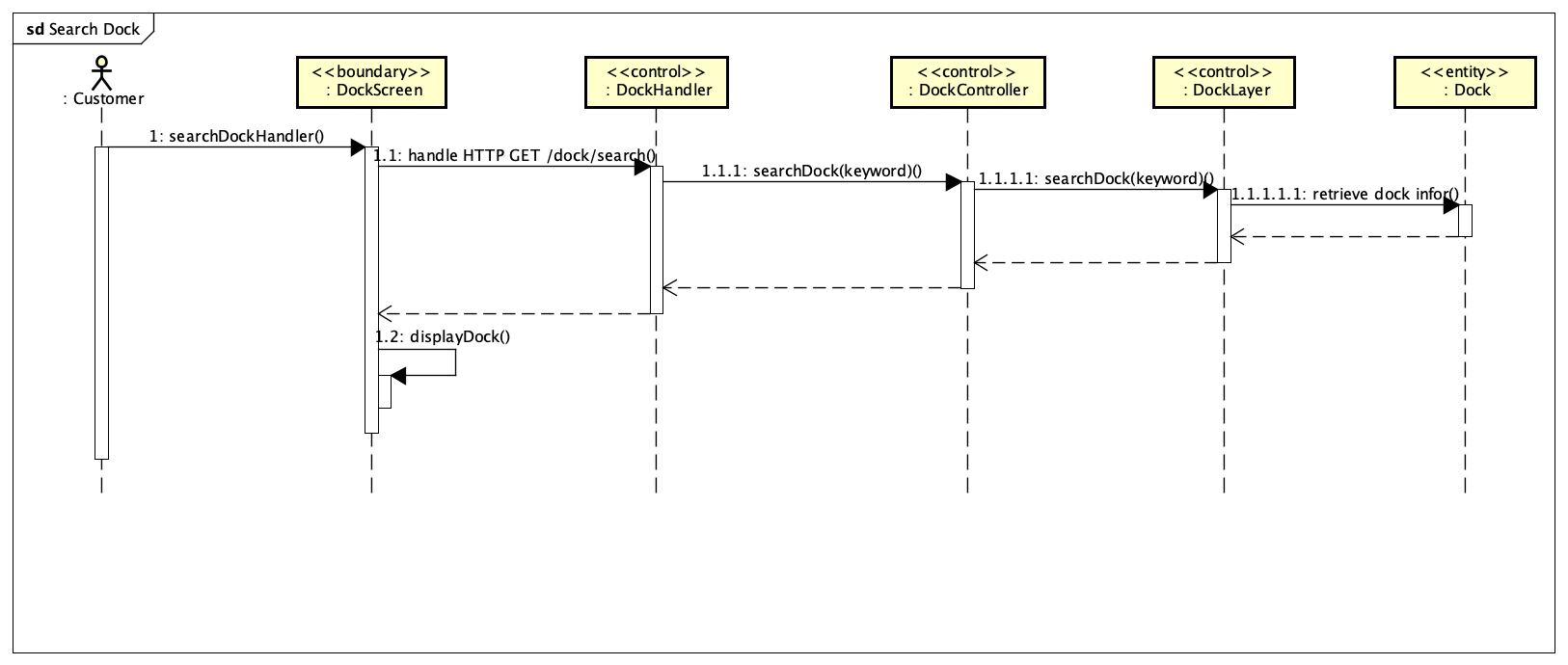
## Architectural Patterns

## Interaction Diagrams

3.2.1 Use case “View Dock list”



3.2.2 Use case “Search Dock”

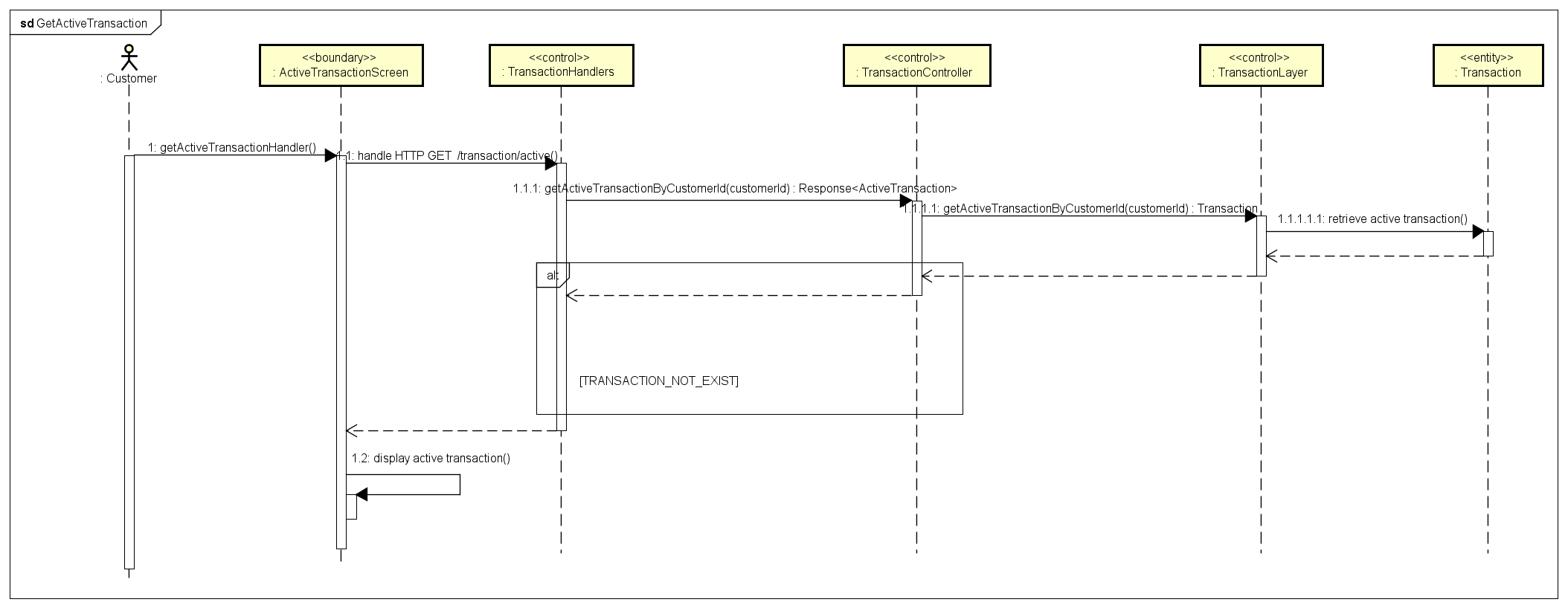


3.2.3 Use case “View bike list in dock”

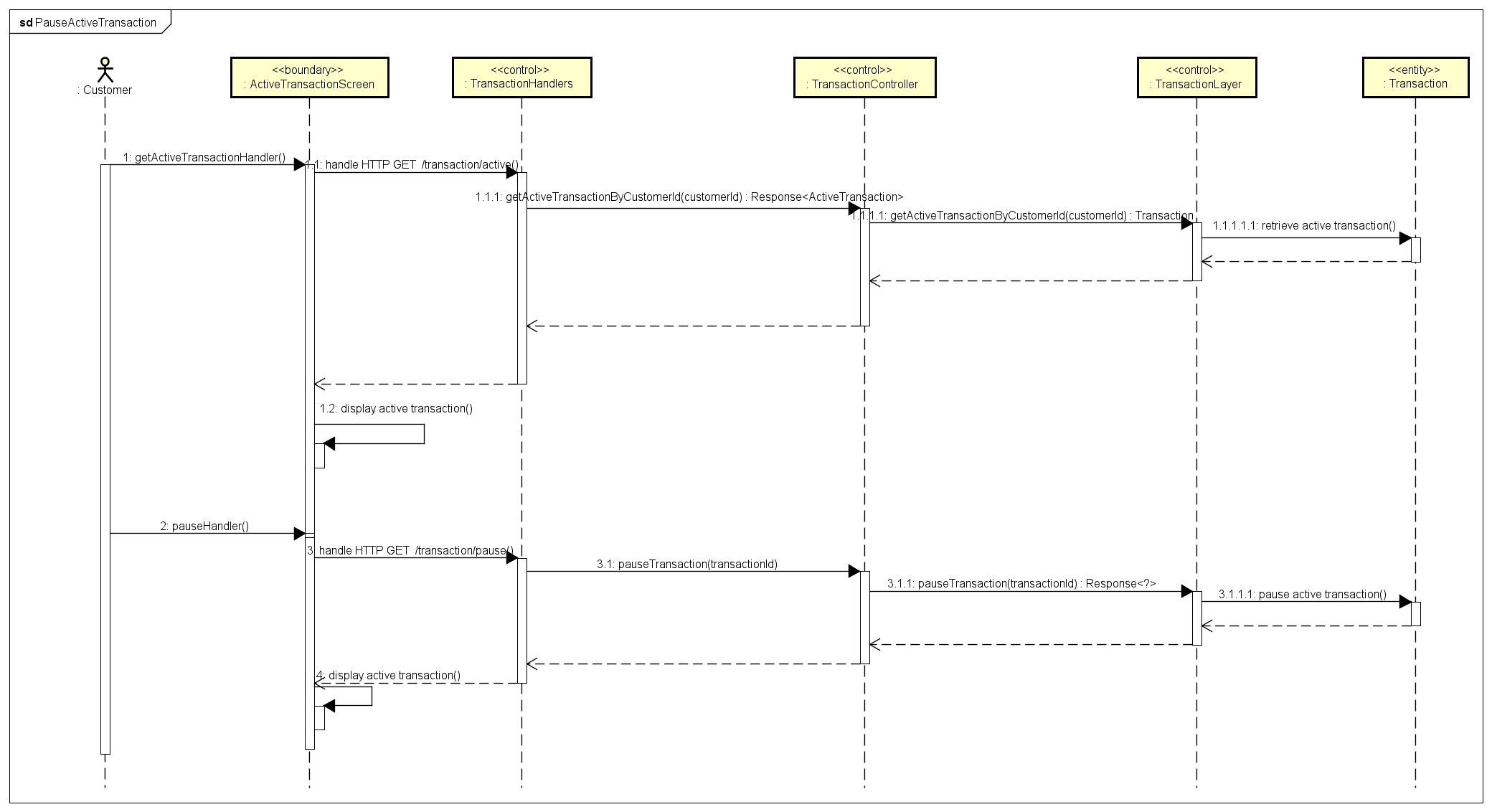
A diagram of a diagram

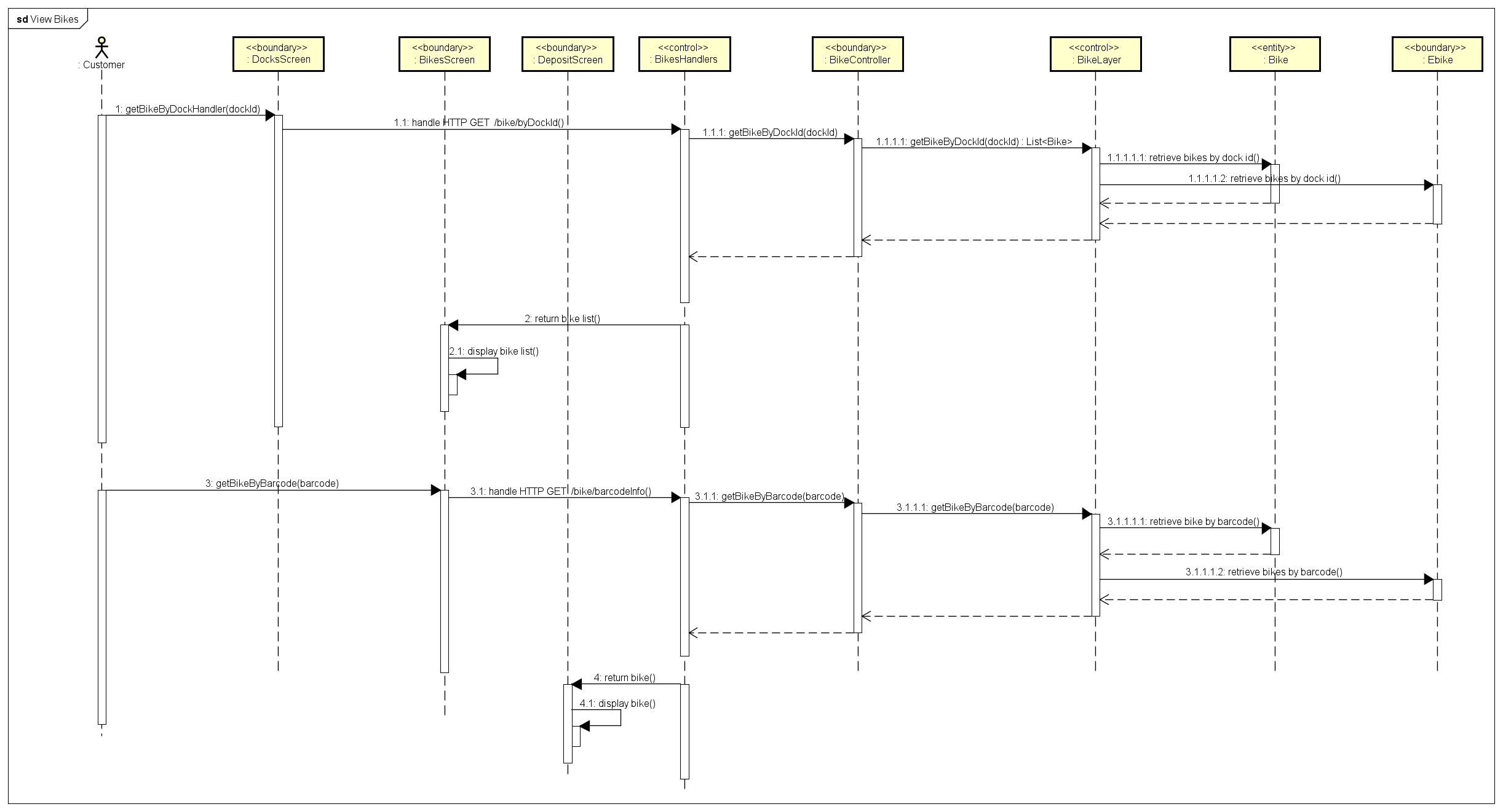
Description automatically generated

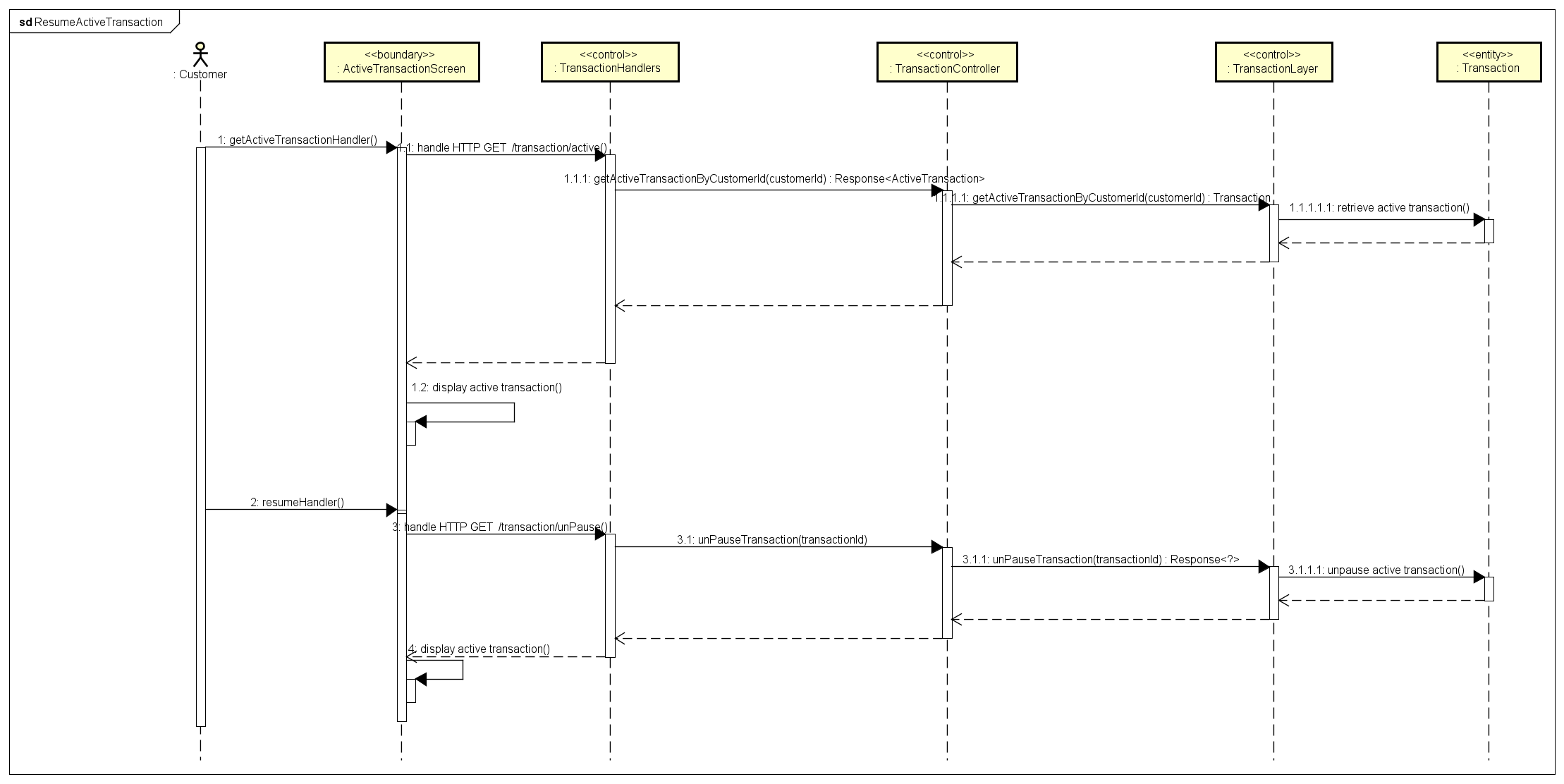
3.2.5. Use case “View Active Transaction”



3.2.6. Use case “Pause Active Transaction”



3.2.7.3.2.8 Use case "Resume Active Transaction”



3.2.9 Use case Rent Bike

A diagram with lines and dots

Description automatically generated

3.2.10 Use case Create Transaction

A diagram of a project

Description automatically generated

3.2.11 Use case Return Bike

A diagram of a project

Description automatically generated with medium confidence

3.2.12 Use case Create Invoice

A screenshot of a computer screen

Description automatically generated

3.2.13 Use case Pay Money

Divide Interbank actor into CreditCard… classes. Ảnh có chứa văn bản, biểu đồ, Song song, hàng

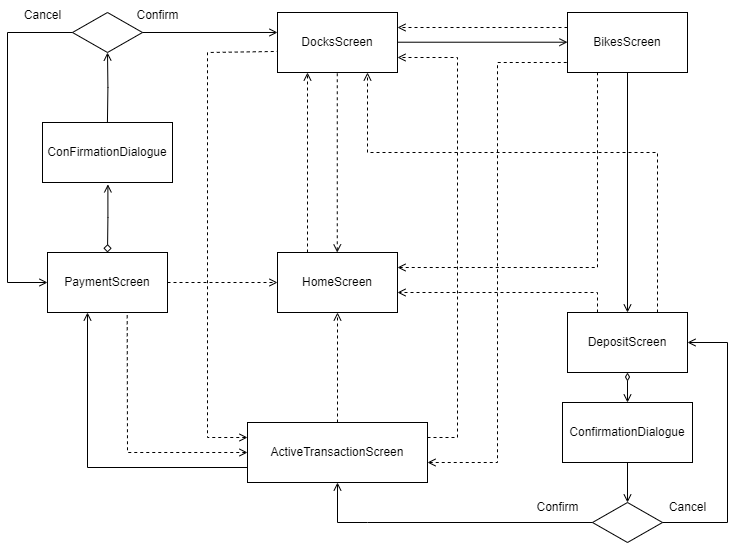
Mô tả được tạo tự động

# Detailed Design

## User Interface Design

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

### Screen Transition Diagrams



### Screen Specifications

*<Screen images should be included in the screen specifications>*

#### Home Screen

**Specification:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcoBike Rental System | | Date of creation | Approved by | Reviewed by | Persion in charge |
| Screen specification | Home screen | 16/8/2023 | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp |
|  | | Control | Operation | Function | |
| Area | Initial | Display header, footer | |
| Navigation Button | Click | Navigate page to home screen, dock screen. active transaction screen | |
| Button | Click | Switch to docks screen | |

#### Docks Screen

**Specification:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcoBike Rental System | | Date of creation | Approved by | Reviewed by | Persion in charge |
| Screen specification | Docks screen | 16/8/2023 | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp |
|  | | Control | Operation | Function | |
| Area | Initial | Display header, footer | |
| Navigation Button | Click | Navigate page to home screen, cart screen, user screen, category list screen, item screen | |
| View Button | Click | View bike list of a dock | |
| Search | Input field | Search docks | |

#### Deposit Screen

**Specification:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcoBike Rental System | | Date of creation | Approved by | Reviewed by | Persion in charge |
| Screen specification | Deposit screen | 16/8/2023 | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp |
|  | | Control | Operation | Function | |
| Area | Initial | Display header, footer | |
| Navigation Button | Click | Navigate page to home screen, dock screen, active transaction screen | |
| Button | Click | Cancel or deposit bike | |
| Select Option | Select | Choose transaction type | |

#### Active Transaction Screen

**Specification:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcoBike Rental System | | Date of creation | Approved by | Reviewed by | Persion in charge |
| Screen specification | Active transaction screen | 16/8/2023 | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp |
|  | | Control | Operation | Function | |
| Area | Initial | Display header, footer | |
| Navigation Button | Click | Navigate page to home screen, dock screen, active transaction screen | |
| Button | Click | Pause, resume active transaction or go to payment page | |

#### Payment Screen

**Specification:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcoBike Rental System | | Date of creation | Approved by | Reviewed by | Persion in charge |
| Screen specification | Payment screen | 16/8/2023 | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp | Nguyễn Tuấn Hiệp |
|  | | Control | Operation | Function | |
| Area | Initial | Display header, footer | |
| Navigation Button | Click | Navigate page to home screen, active transaction screen | |
| Button | Click | Reload time rent and price of make payment of the transaction | |
| Text field | Fill | Fill credit card information | |

## Data Modeling

### Conceptual Data Modeling

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

### Database Design

#### Database Management System

*<Specify what is the decision of Database Management System (DBMS) and give some description of the DBMS>*

PostgreSQL is an open-source relational database management system (RDBMS) known for its robust features, reliability, and performance. Here are some key points about PostgreSQL:

1. **Open Source:** PostgreSQL is open-source software, which means it's freely available, and you have full access to its source code. This makes it cost-effective and gives you the flexibility to customize and extend it to meet your specific needs.
2. **Relational Database:** PostgreSQL is a powerful relational database system, which excels at managing structured data and complex relationships between different data tables, making it well-suited for your bike rental management system.
3. **Extensibility:** PostgreSQL supports various data types, including custom data types and functions, which can be handy for modeling complex data scenarios or adding specific functionality to your application.
4. **Advanced Querying:** PostgreSQL offers advanced querying capabilities, including support for complex joins, subqueries, and window functions, allowing you to retrieve and manipulate data in sophisticated ways.
5. **Scalability:** PostgreSQL can handle large datasets and is designed to scale both vertically and horizontally. This is essential as your bike rental system grows and accumulates more data.
6. **Security:** PostgreSQL has robust security features, including role-based access control, SSL encryption, and support for authentication mechanisms. You can control who has access to your data and how they can interact with it.
7. **Community Support:** PostgreSQL has a large and active open-source community. This means you can find plenty of documentation, tutorials, and community support to help you with any issues or questions you might encounter.

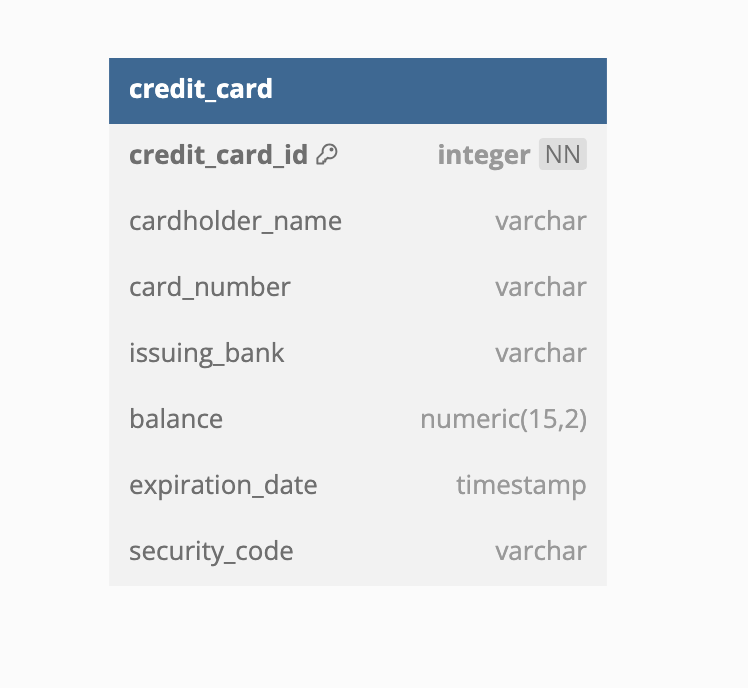
#### Database Diagram

1. EcoBikeSystem

A computer screen shot of a computer

Description automatically generated

1. Interbank



#### Database Detail Design

<

*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).*

Table 1. Example of table design

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | ProductID |  |  |  |  |
| 2 |  | x | CategoryID |  |  |  |  |

*You may add indexing, trigger, view, etc.*

*Give the database script*>

## 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

<General class diagram which shows the whole class diagram of the software. This diagram may have packages, subsystems and classes. Classes in this diagram may not have all attributes and operations>

### Class Diagrams

<Detail class diagram with full attributes and operations>

A yellow folders with black lines

Description automatically generated

### Class Design

#### Class “Bike”

<SampleClass1 class image in UML>

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, thực đơn

Mô tả được tạo tự động

Table 1. Example of attribute design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | bikeName | String |  | Name of the bike |
| 2 | image | String |  | Link to the image |
| 3 | bikeId | Integer |  | Id of the bike |
| 4 | category | Category |  | Category of the bike |
| 5 | barcode | UUID |  | Barcode of the bike |
| 6 | dock | Dock |  | Current dock that the bike is in it. |

Table 1. Example of operation design

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | setImage | void | Set data |
| 2 | getBarcode | UUID | Get data |
| 3 | setDock | void | Set data |
| 4 | setBarcode | Void | Set data |
| 5 | setBikeId | Void | Set data |
| 6 | setBikeName | Void | Set data |
| 7 | setCategory | void | Set data |
| 8 | getDock | Dock | Get data |
| 9 | getImage | String | Get data |
| 10 | getBikeId | Integer | Get data |
| 11 | getBikeName | String | Get data |
| 12 | getCategory | Category | Get data |

#### Class “EBike”

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Table 2. Example of attribute design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | bikeName | String |  | Name of the bike |
| 2 | image | String |  | Link to the image |
| 3 | bikeId | Integer |  | Id of the bike |
| 4 | category | Category |  | Category of the bike |
| 5 | barcode | UUID |  | Barcode of the bike |
| 6 | dock | Dock |  | Current dock that the bike is in it. |
| 7 | battery | Integer |  | Current battery of Ebike |

Table 2. Example of operation design

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | setImage | void | Set data |
| 2 | getBarcode | UUID | Get data |
| 3 | setDock | void | Set data |
| 4 | setBarcode | Void | Set data |
| 5 | setBikeId | Void | Set data |
| 6 | setBikeName | Void | Set data |
| 7 | setCategory | void | Set data |
| 8 | setBattery | void | Set data |
| 9 | getDock | Dock | Get data |
| 10 | getImage | String | Get data |
| 11 | getBikeId | Integer | Get data |
| 12 | getBikeName | String | Get data |
| 13 | getCategory | Category | Get data |
| 14 | getBattery | Integer | Get data |

# Design Considerations

***<Describe issues which need to be addressed or resolved before attempting to devise a complete design solution. Remember that, you have to refactor your source code to strictly follow the final design>***

## 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>*

*<You may show the previous design from which you made improvements to get better levels of coupling and cohesion. You should clarify how and why you did these improvements>*

## Design Principles

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

*<You may show the previous design from which you made improvements to get a better design, which follows SOLID principles in spite of additional requirements. You should clarify how and why you did these improvements>*

## 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>*