Hanoi University of Science and Technology
School of Information and Communication Technology

Software Design Document

Version 1.0

EcoBike Application

Subject: ITSS Software Development

Supervisor: Asst. Prof. Nguyen Thi Thu Trang

Group 11

Group members

Name	Student ID
Pham Tuan Long	20205161
Dang Trong Luat	20205187
Dang Hoang Long	20200357
Duong Duy Manh	20205162

1. Introduction

1.1. Objective

The objective of the EcoBike software is to provide a comprehensive and user-friendly platform for efficient and convenient bike rental services within the Ecopark township. This software aims to offer residents and visitors a seamless experience when renting, using, and returning bicycles through an intuitive mobile application.

By integrating user-friendly features, transparent pricing models, and real-time information about available bikes and docking stations, the software intends to encourage eco-friendly transportation alternatives and promote healthy lifestyle choices.

1.2. Scope

The scope of the software encompasses the design, development, and implementation of a user-friendly mobile application that facilitates efficient bike rental and return processes within the Ecopark township. The software is focused on providing a seamless and convenient experience for users seeking alternative and eco-friendly modes of transportation.

1.3. Glossary

Term	Definition
Administrator	The person who uses the system for the purposes of monitoring list of bicycles in the system
Admin	as “administrator”
Bicycle	The transportation mean to be rent in this application system
Bike	as “bicycle”
Cardholder name	The name of the owner of the credit card, printed on the credit card
Credit card	A card connected to the interbank, used for performing transaction
Customer	The person who uses EcoBike application system for the purposes of renting bike
Database	Collection of all information monitored by this system

Deposit	An amount of money customer has to pay at first in order to rent a bike
Dock	A place where bicycles are put
Interbank	The organization in charges of performing payment and return deposit transactions in the system
Payment	An amount of money customer has to pay to rent a bike, including deposit and rental fee
Rent a bike	The action of using a bike in a period of time, with paying deposit and rental fee
Rental fee	An amount of money customer has to pay, outside of the deposit, which depends on the rental time
Rental time	The time period when the bike is being rented
Return a bike	The action of stopping using a bike after having rented
Software	Requirement Specification A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
Station	as “dock”
Transaction	The action of paying for bike deposit, bike rental or returning deposit
User	Customer or Administrator

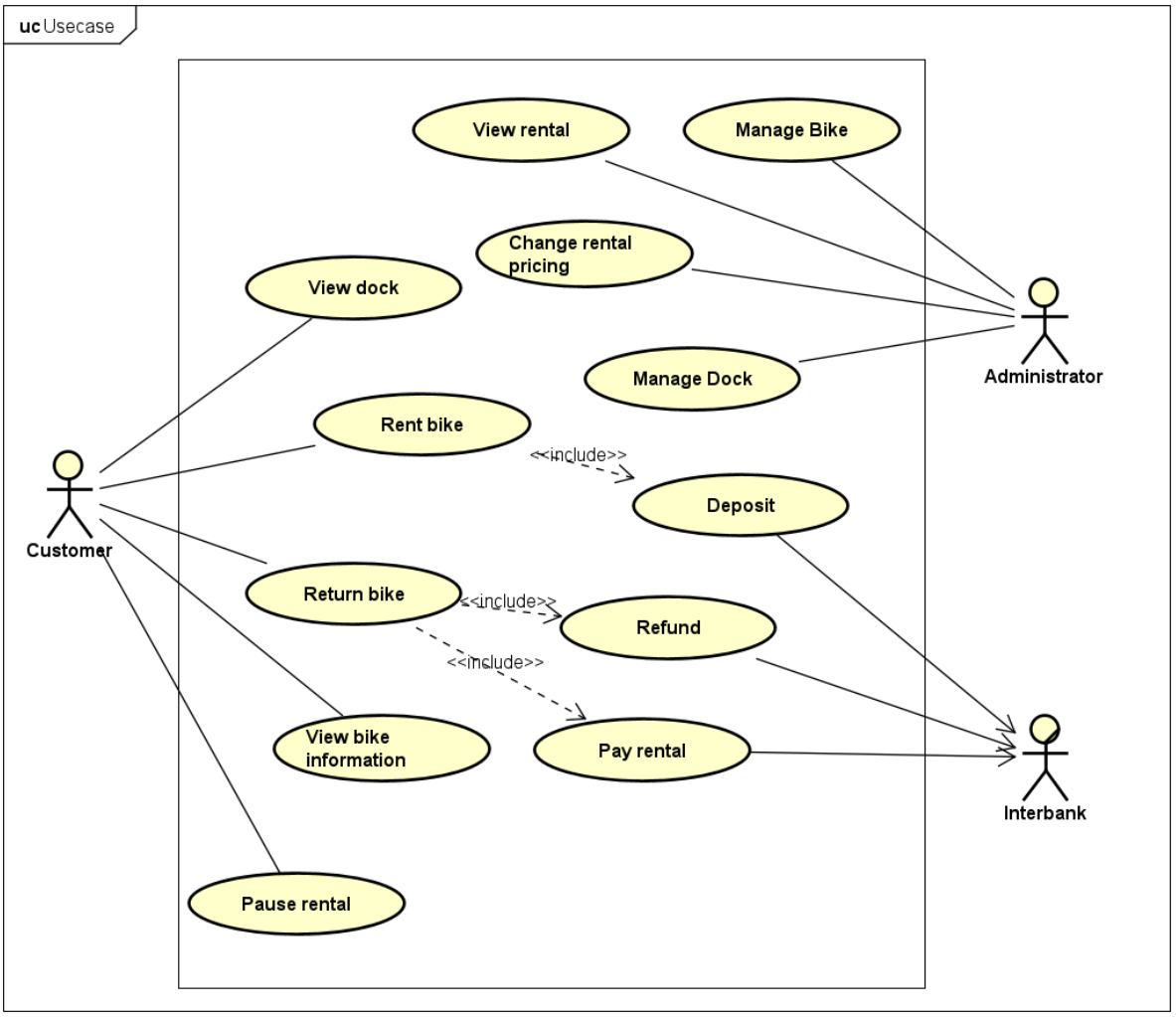
2. Overall Description

2.1. General Overview

The EcoBikeRent software is a cutting-edge application designed to revolutionize the bike rental experience within the Ecopark township. With a focus on sustainability, convenience, and user-friendliness, the software aims to offer residents and visitors an efficient and eco-friendly alternative for transportation.

By providing a seamless platform for renting, using, and returning bicycles, the EcoBikeRental application seeks to enhance the quality of life within the community while contributing to environmentally conscious practices.

The below figure is the general use-case diagram for our design:



2.2. Assumptions/Constraints/Risks

2.2.1. Assumptions

In order to use the software, users must have an internet connection as well as a personal computer to run the app. We would also require the latest version of NodeJS in order to ensure the software's stability.

2.2.2. Constraints

- 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

2.2.3. Risks

3. System Architecture and Architecture Design

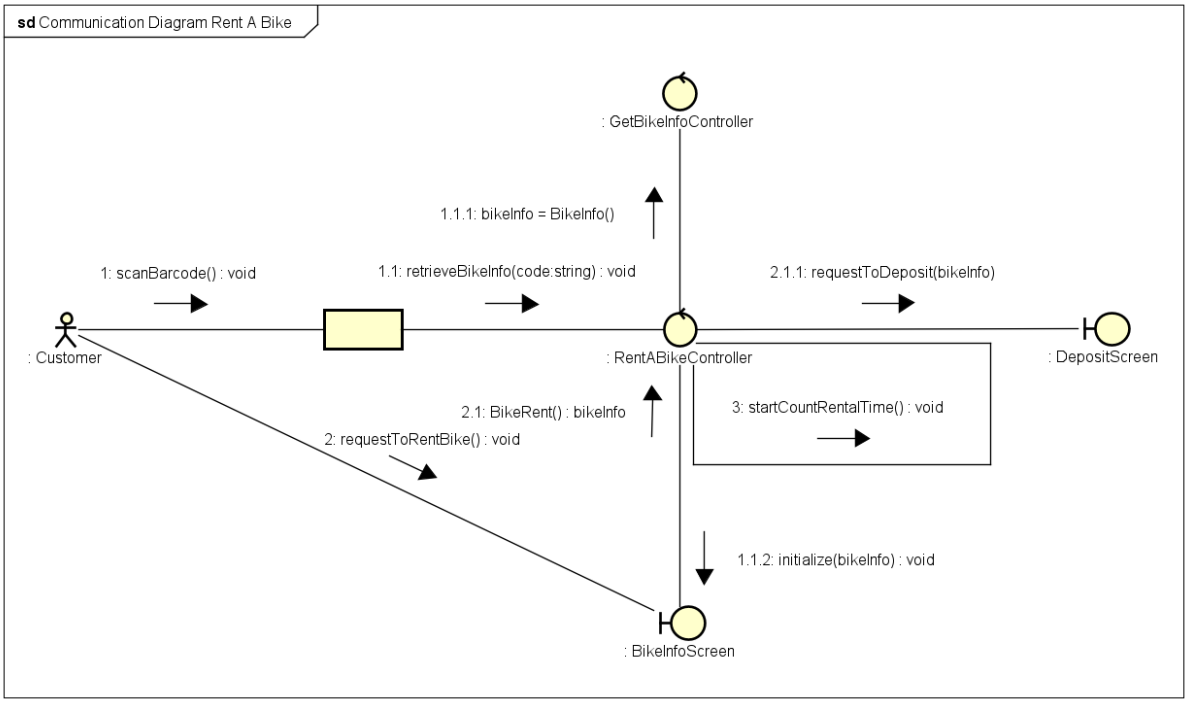
3.1. Architectural Patterns

The EcoBikeRent software is designed using a combination of architectural patterns to ensure modularity, scalability, and maintainability. The primary architectural patterns employed are:

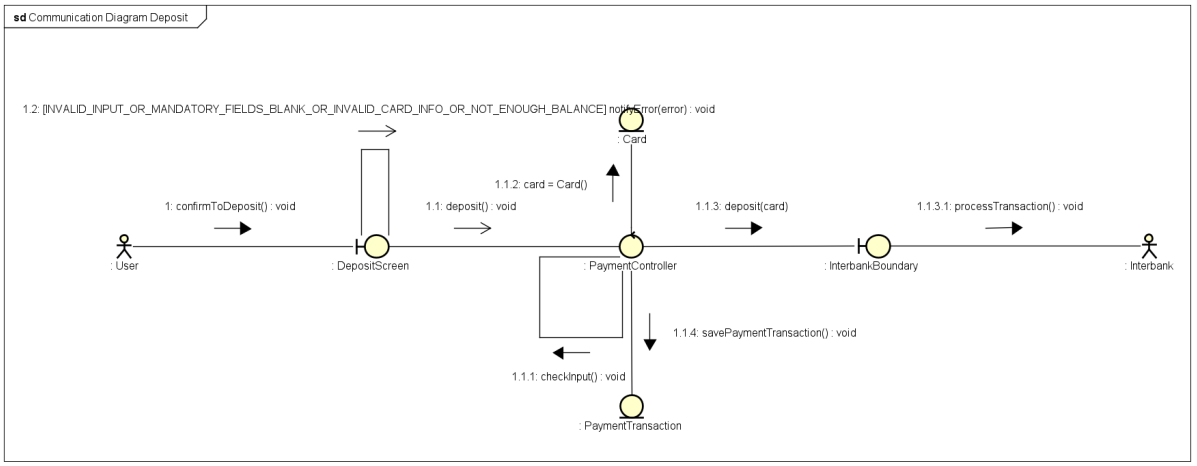
1. **Model-View-Controller (MVC):**
    - The MVC pattern separates the application into three components: Model (data and business logic), View (user interface), and Controller (handles user input and manages communication between Model and View).
    - This pattern enhances maintainability by isolating different concerns and enabling independent development and testing of each component.
2. **Layered Architecture:**
    - The software utilizes a layered architecture, consisting of presentation, business logic, and data access layers.
    - The presentation layer handles the user interface, the business logic layer manages application-specific processes, and the data access layer interacts with the database.
3. **Client-Server Architecture:**
    - The application follows a client-server architecture, where the mobile application (client) interacts with the server-side components for data retrieval, processing, and storage.
    - This pattern supports scalability and enhances the separation of concerns between the client and server.
4. **Microservices Architecture:**
    - The software employs a microservices architecture to break down complex functionalities into smaller, loosely coupled services.
    - This approach allows for independent development, deployment, and scalability of individual services, enhancing system agility.

3.2. Interaction Diagrams

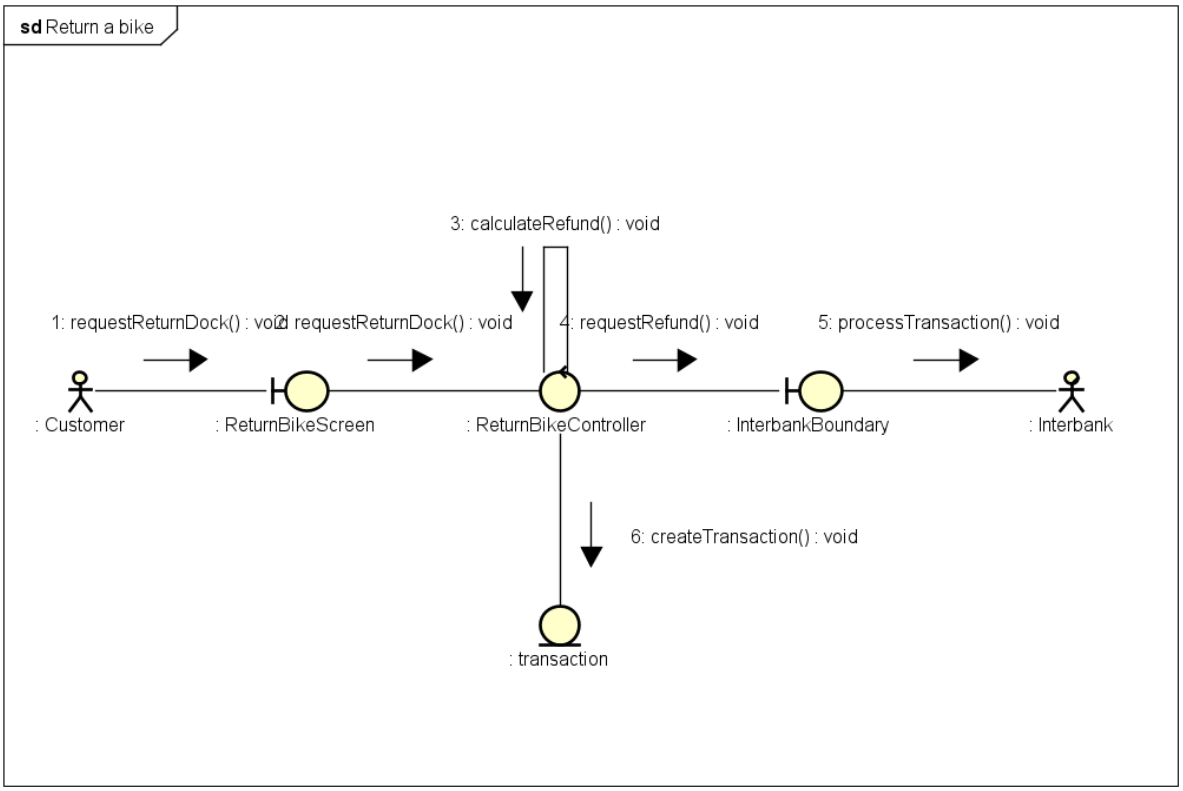
3.2.1. Communication Diagrams



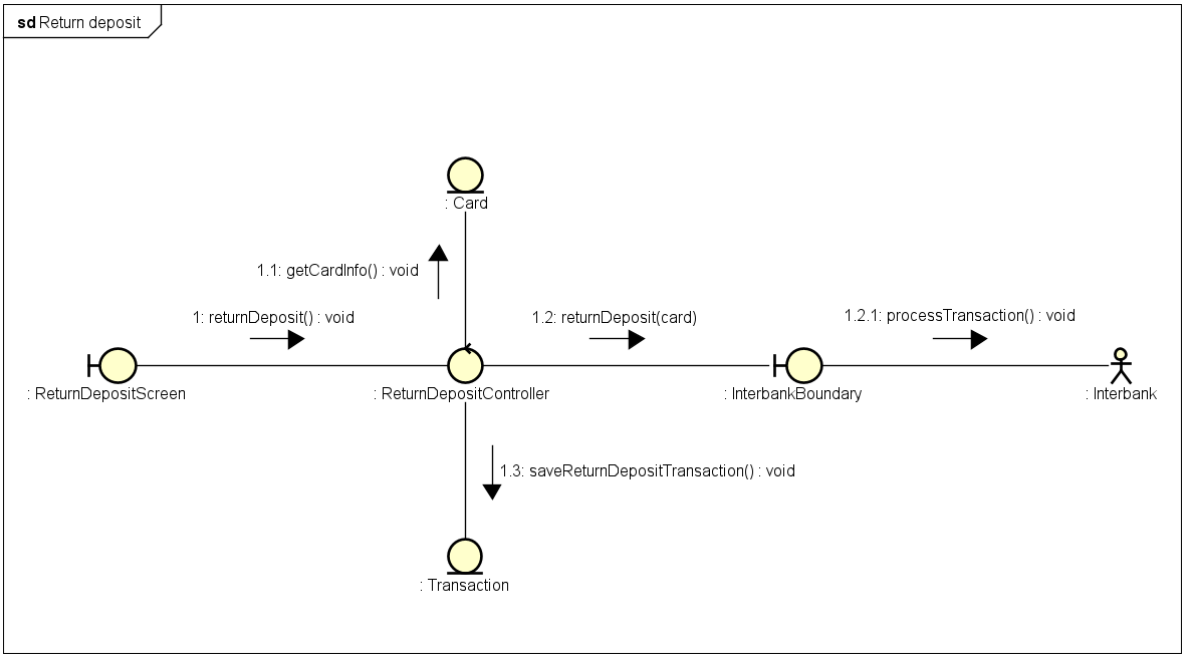
Communication Diagram for Rent Bike Use Case



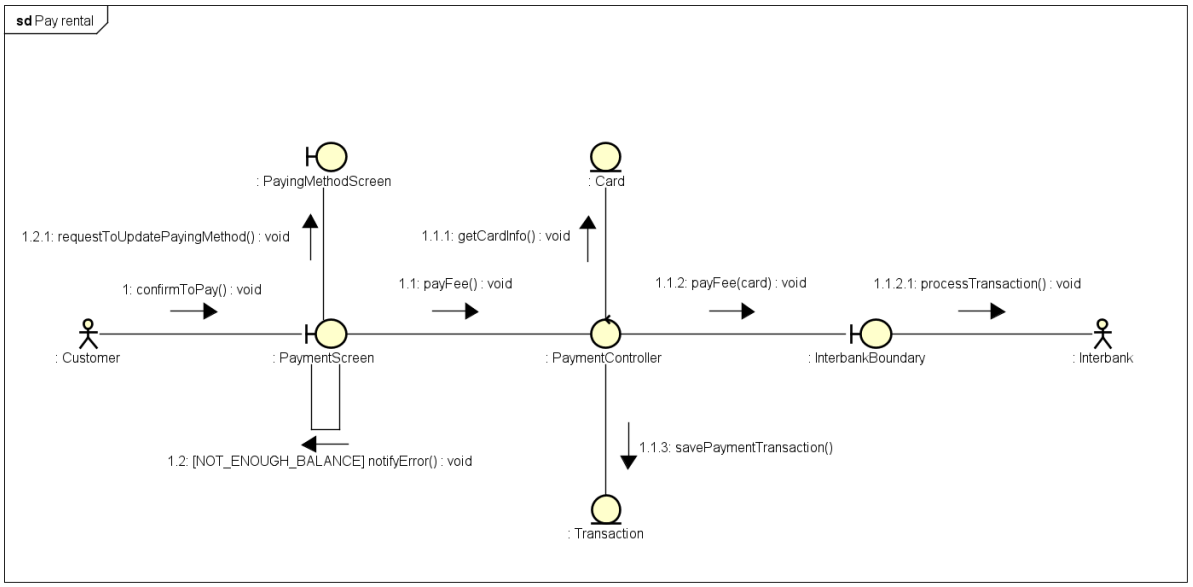
Communication Diagram for Deposit Use Case



Communication Diagram for Return Bike Use Case

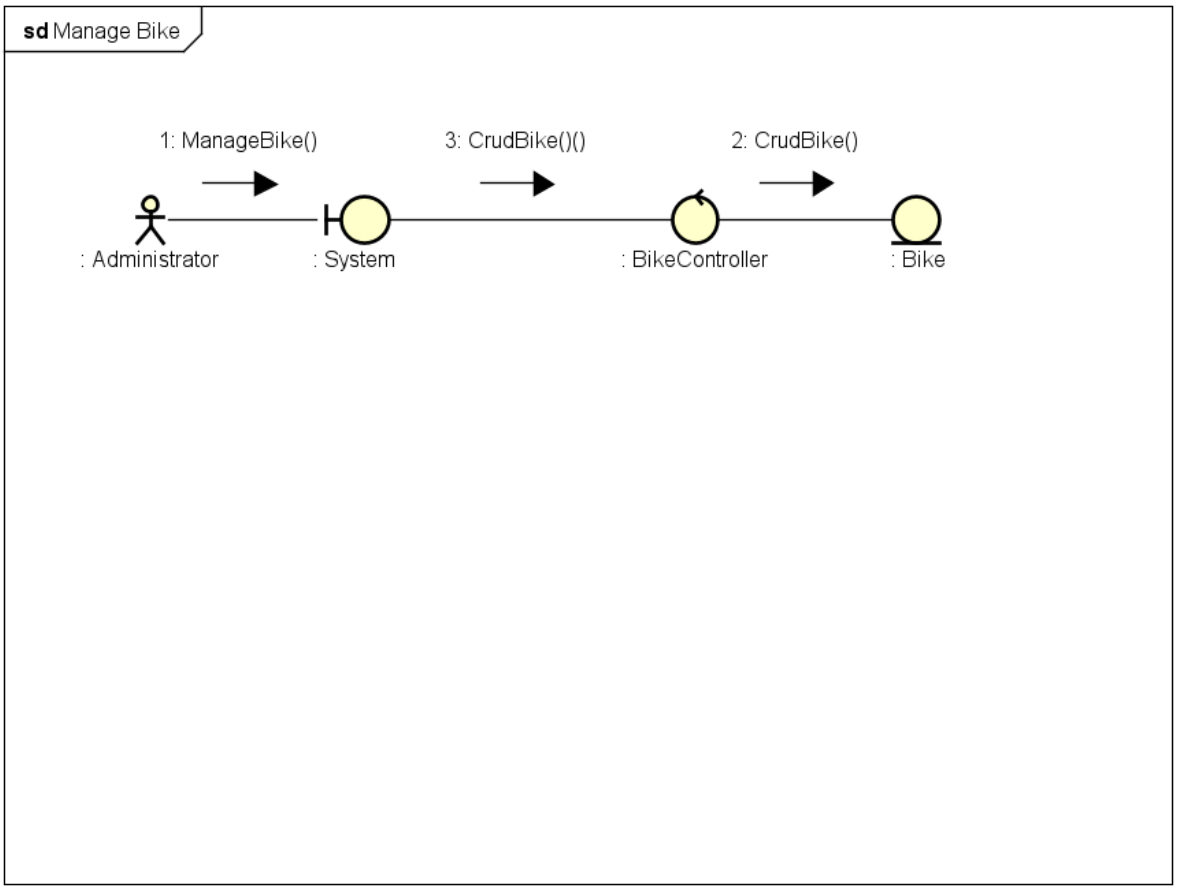


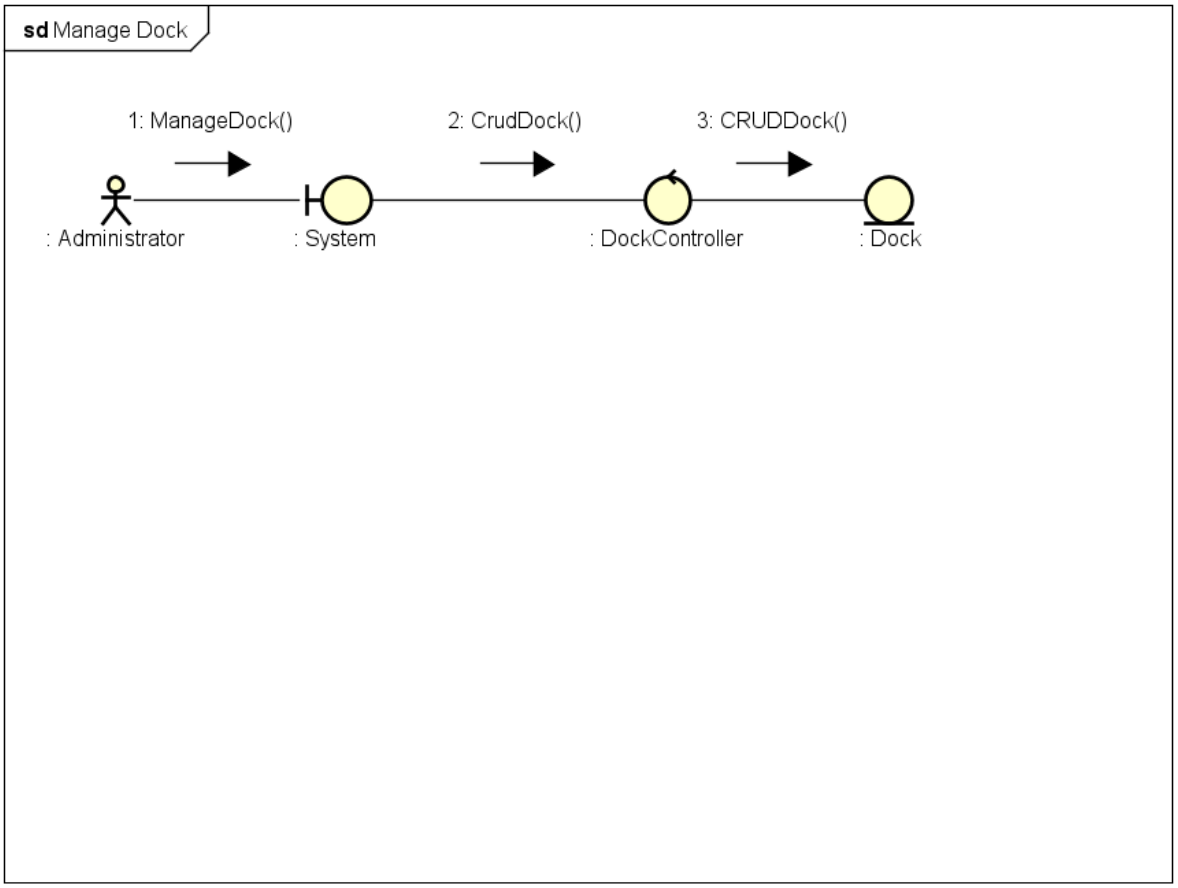
Communication Diagram for Return Deposit Use Case

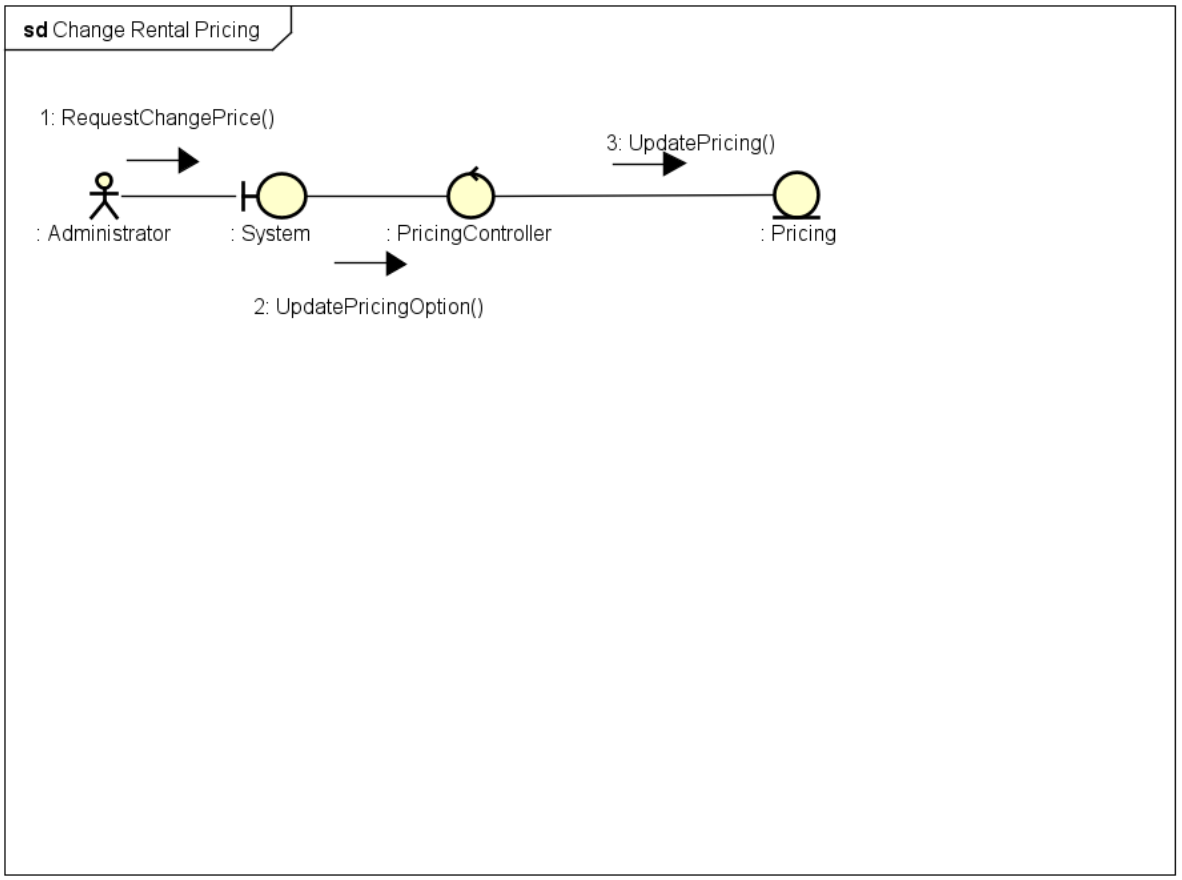


Communication Diagram for Pay For Rental Use Case

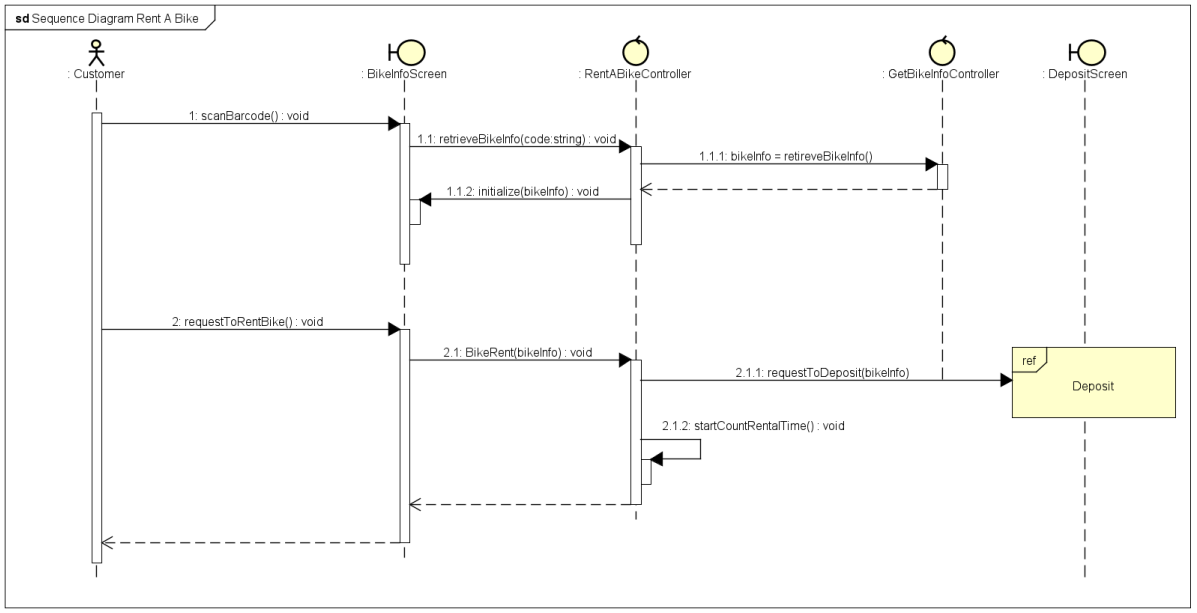
Communication Diagram For Admin


sd Manage Bike
1: ManageBike()
3: CrudBike()()
2: CrudBike()
: Administrator
: System
: BikeController
: Bike

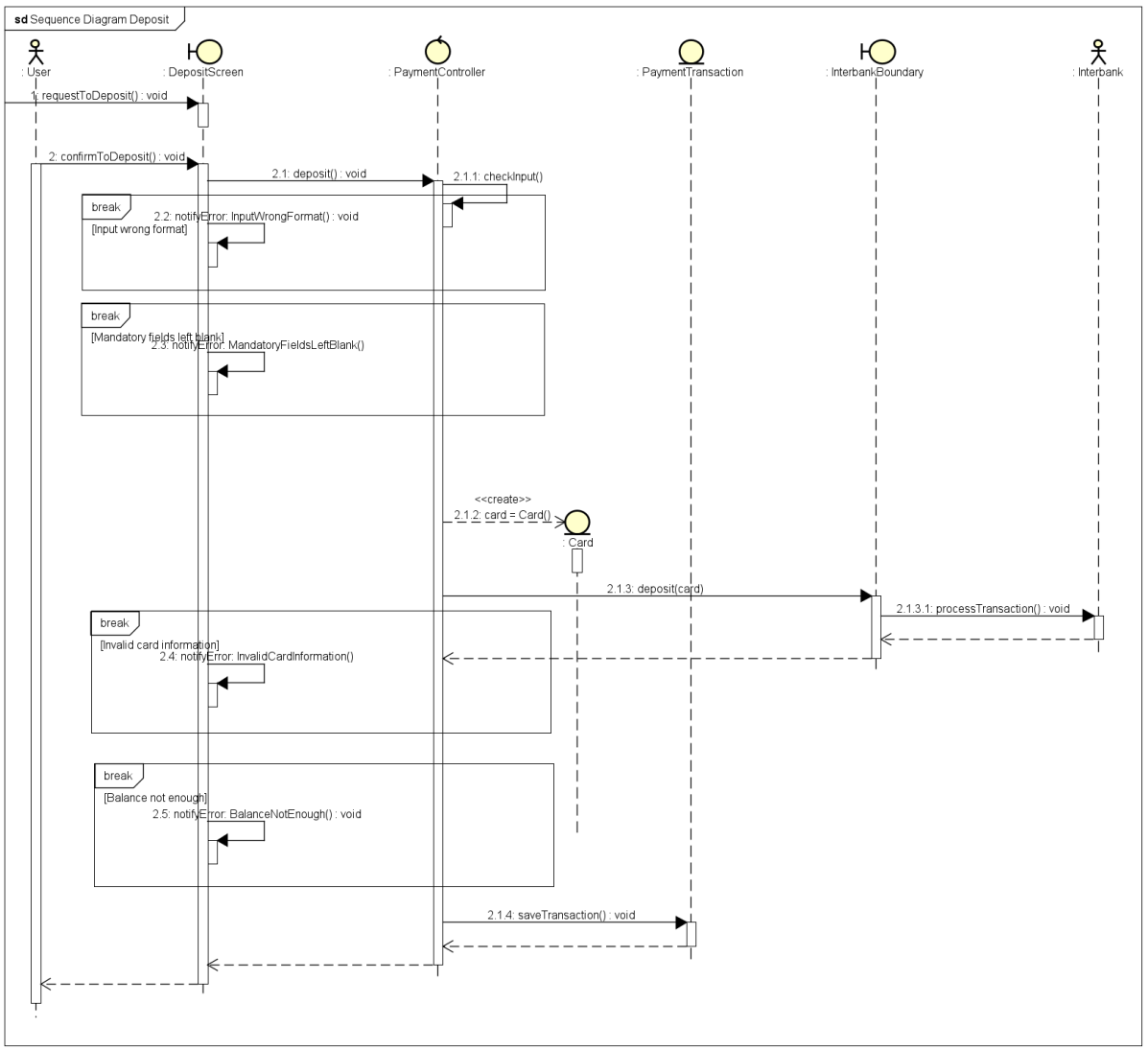

sd Manage Dock
1: ManageDock()
2: CrudDock()
3: CRUDDock()
: Administrator
: System
: DockController
: Dock



3.2.2. Sequence Diagrams

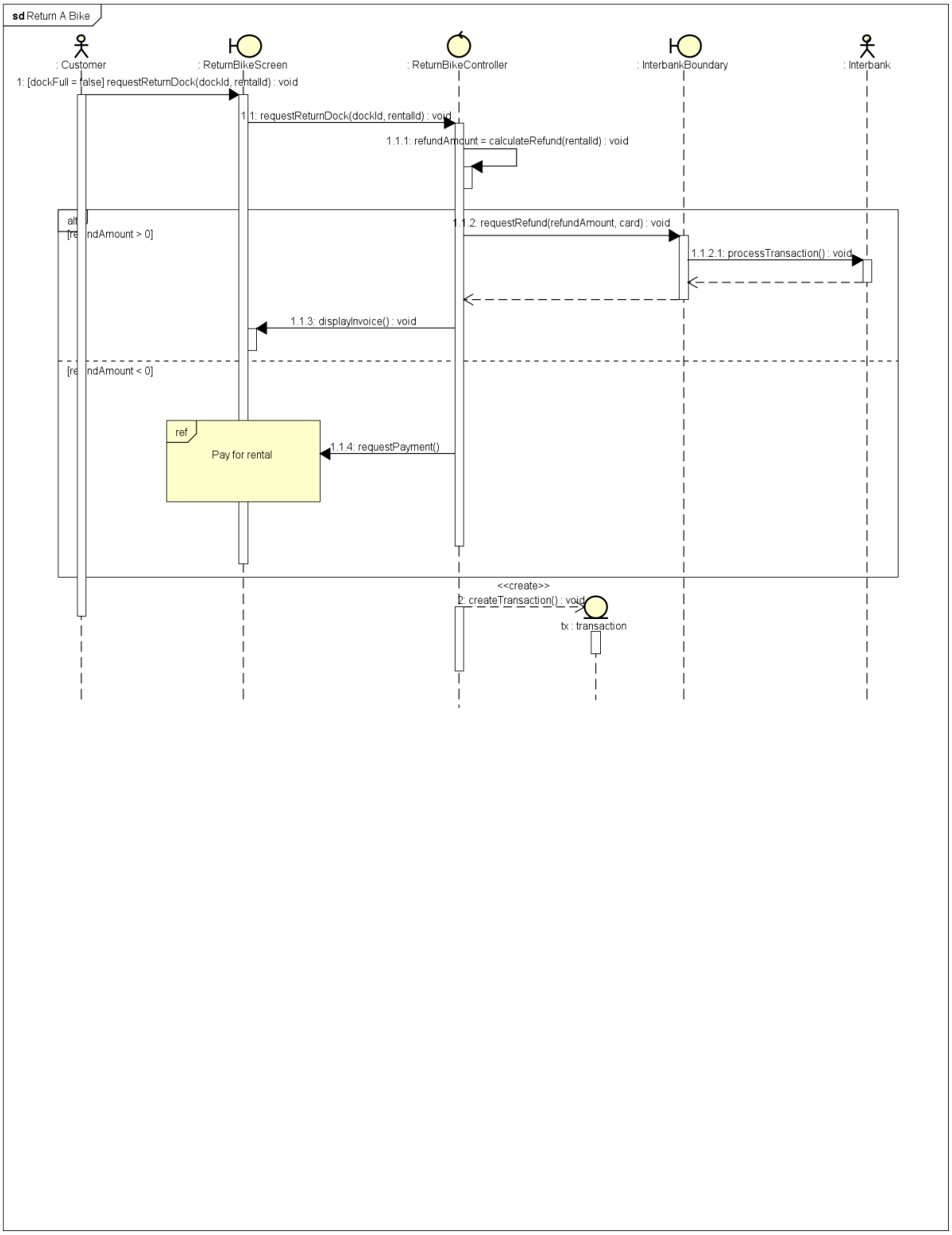


Sequence Diagram for Rent Bike Use Case

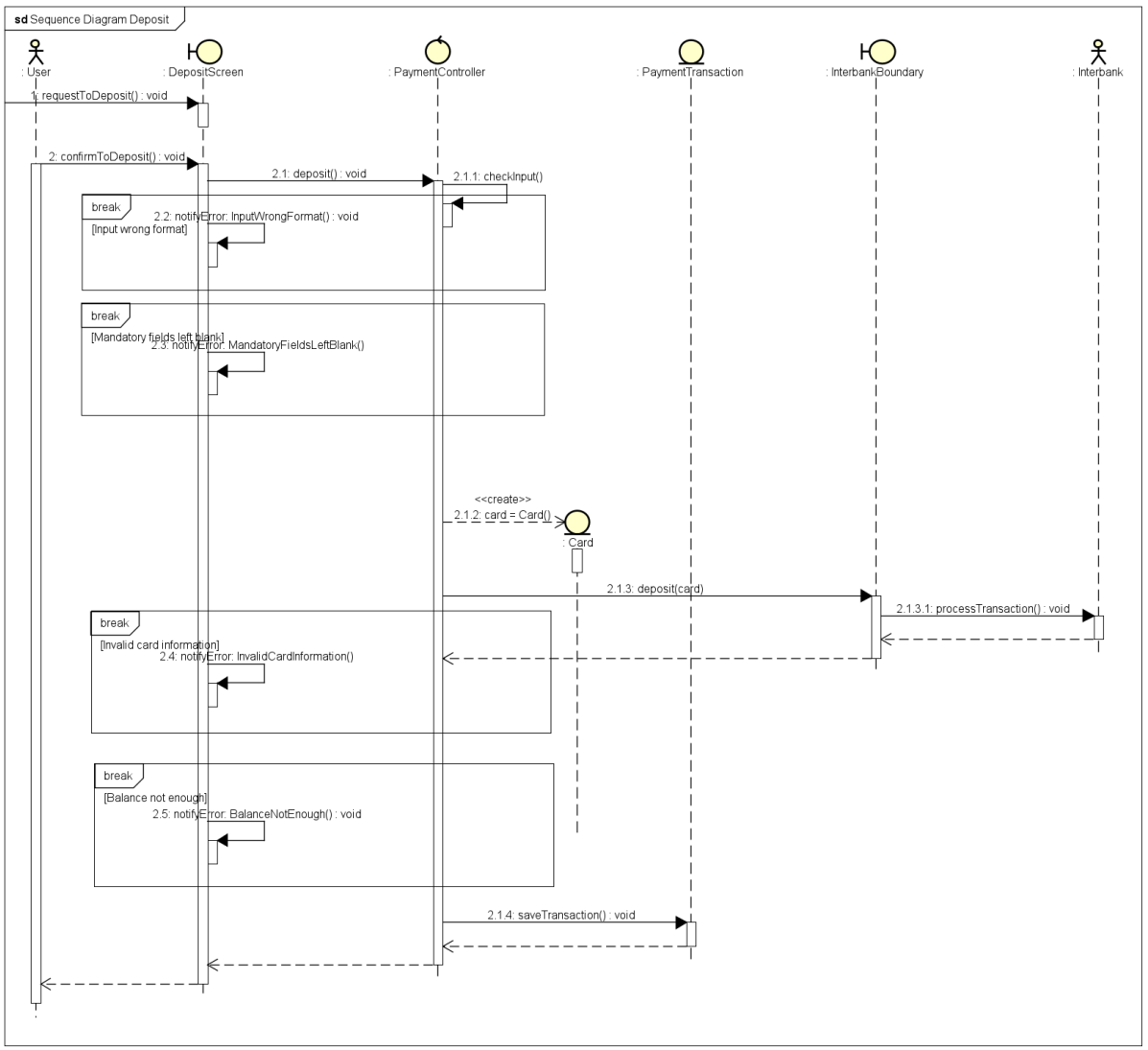


Sequence Diagram for Deposit Use Case

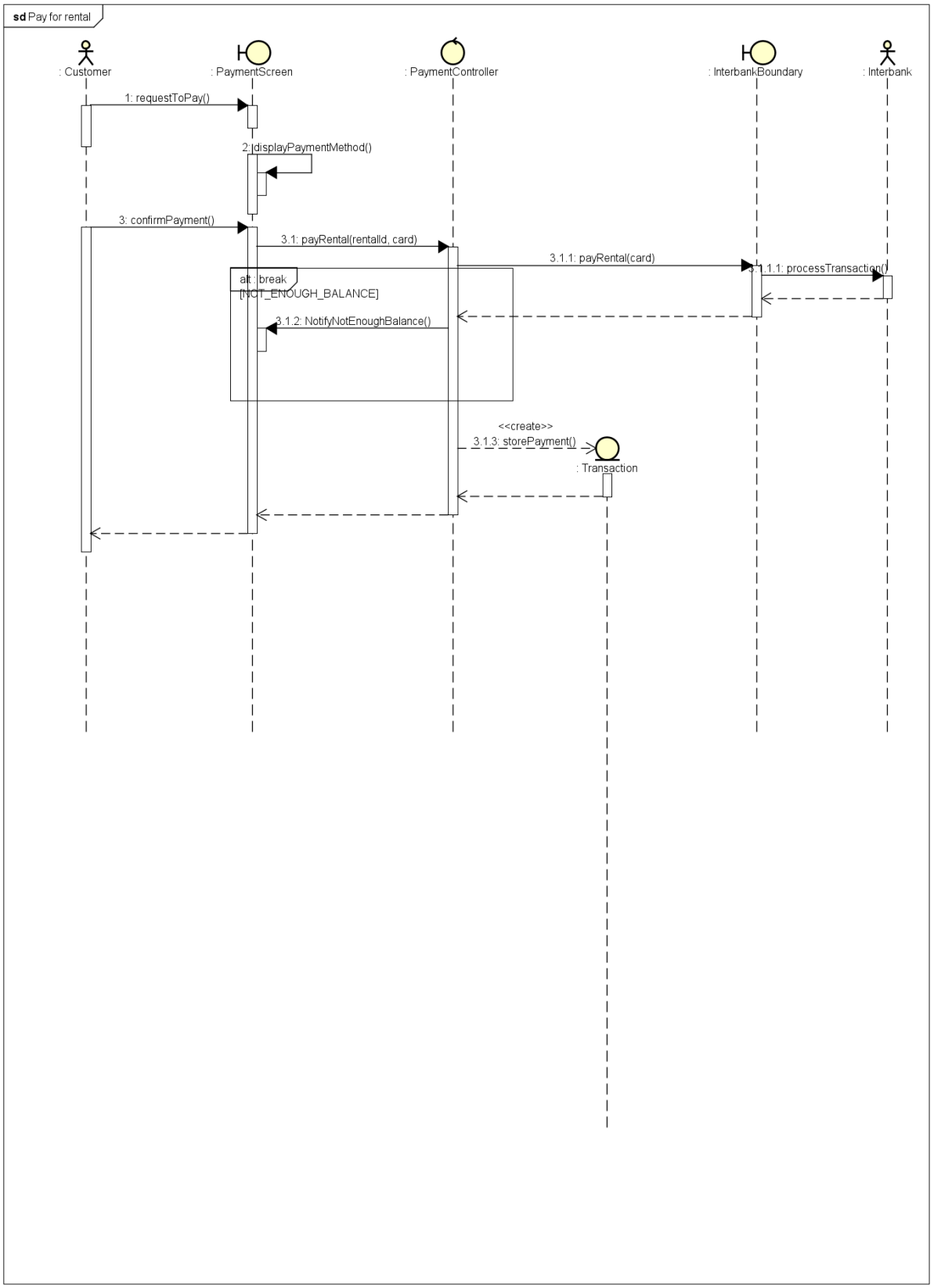
Sequence Diagram for Update Payment Method Use Case



Sequence Diagram for Return Bike Use Case

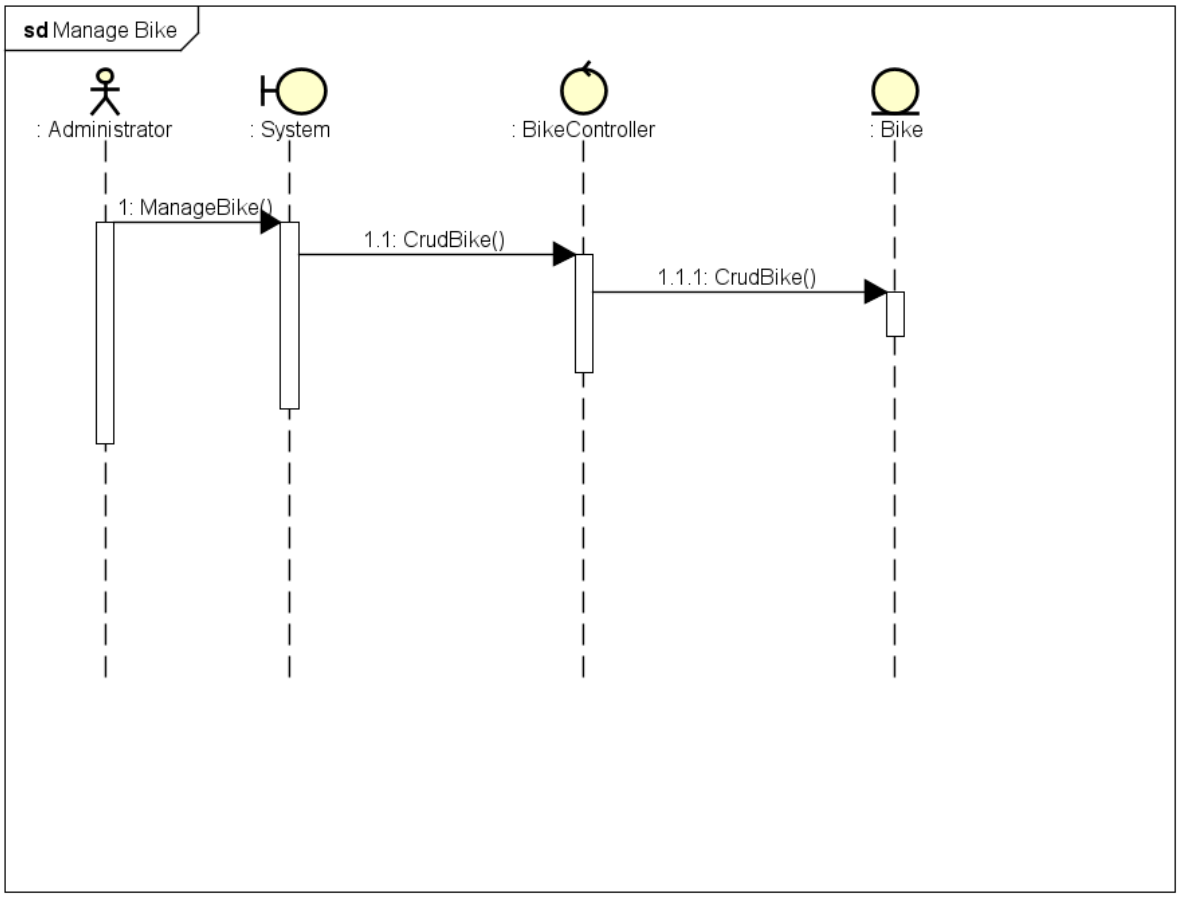


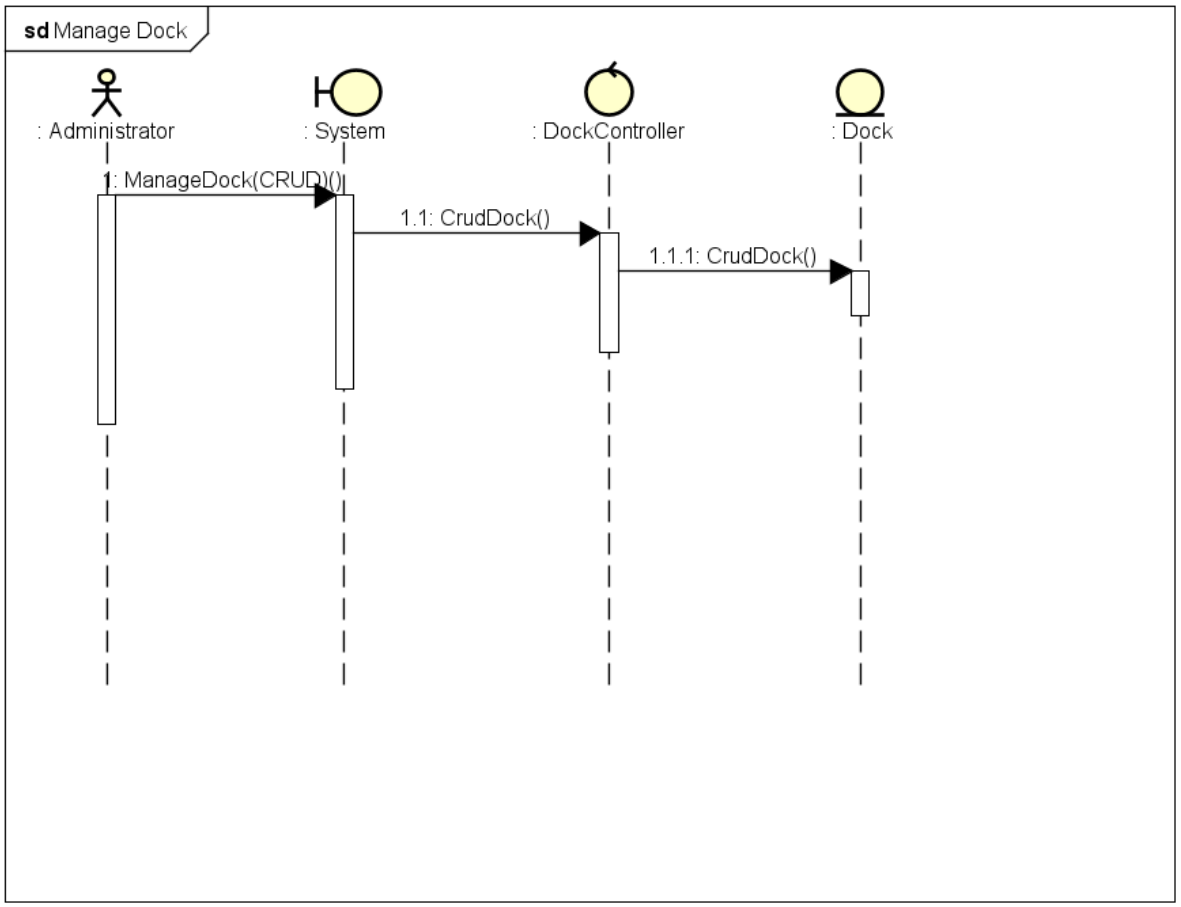
Sequence Diagram for Return Deposit Use Case

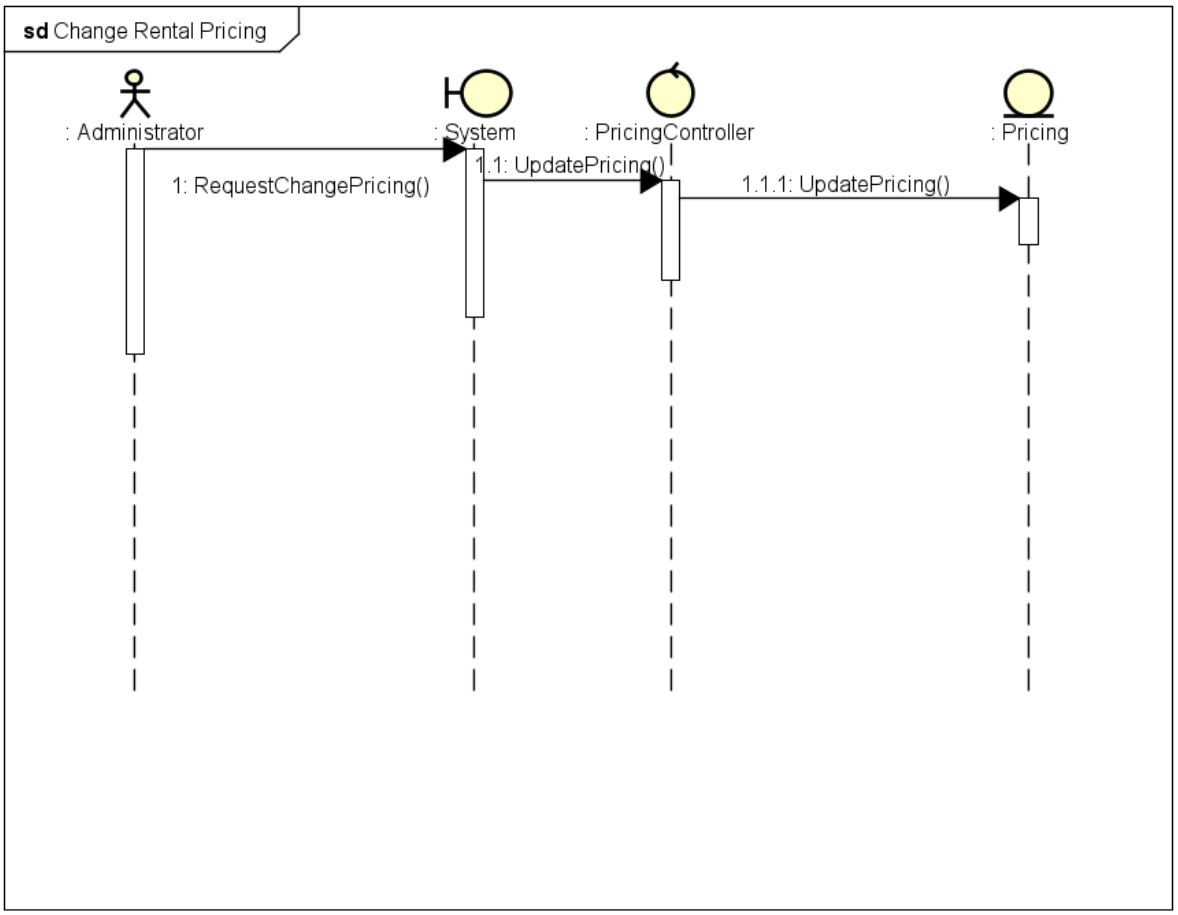


Sequence Diagram for Pay For Rental Use Case

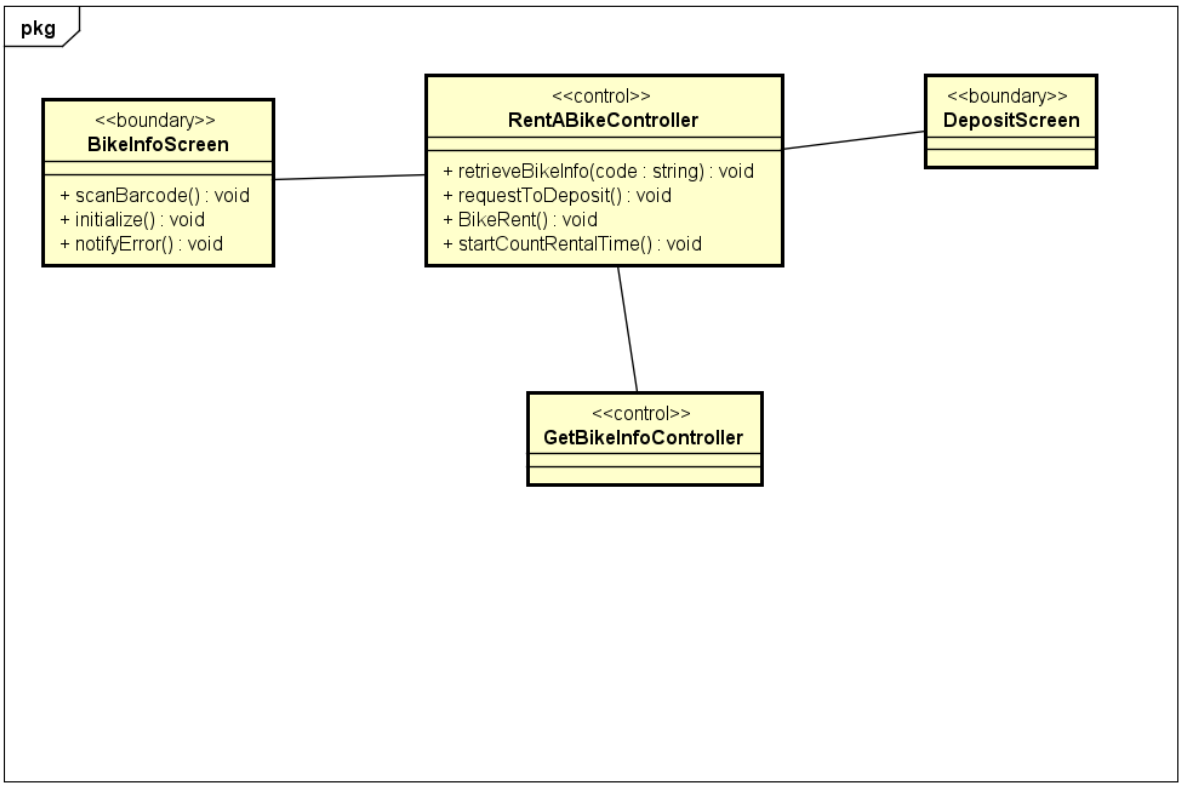
Sequence Diagram for Admin


sd Manage Bike
: Administrator
: System
: BikeController
: Bike
1: ManageBike()
1.1: CrudBike()
1.1.1: CrudBike()

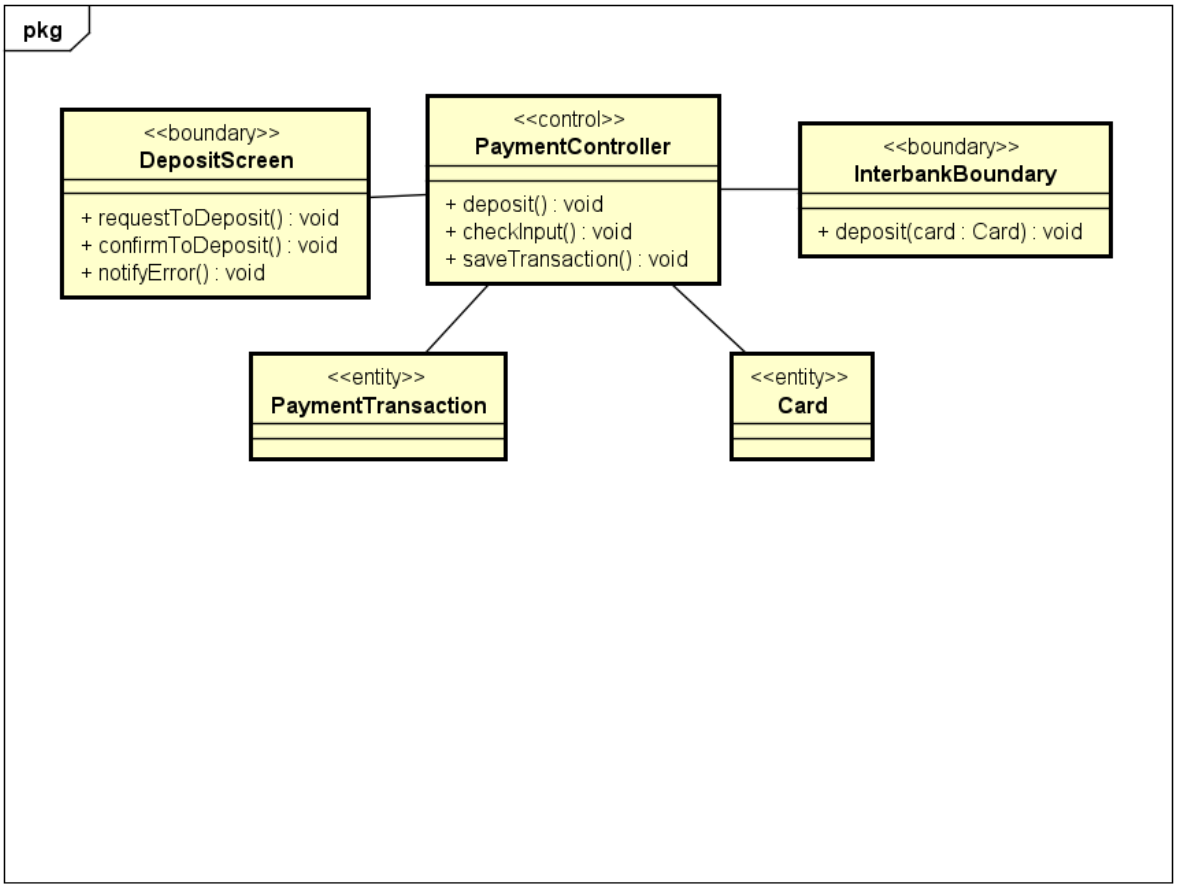

sd Manage Dock
: Administrator
: System
: DockController
: Dock
1: ManageDock(CRUD)()
1.1: CrudDock()
1.1.1: CrudDock()



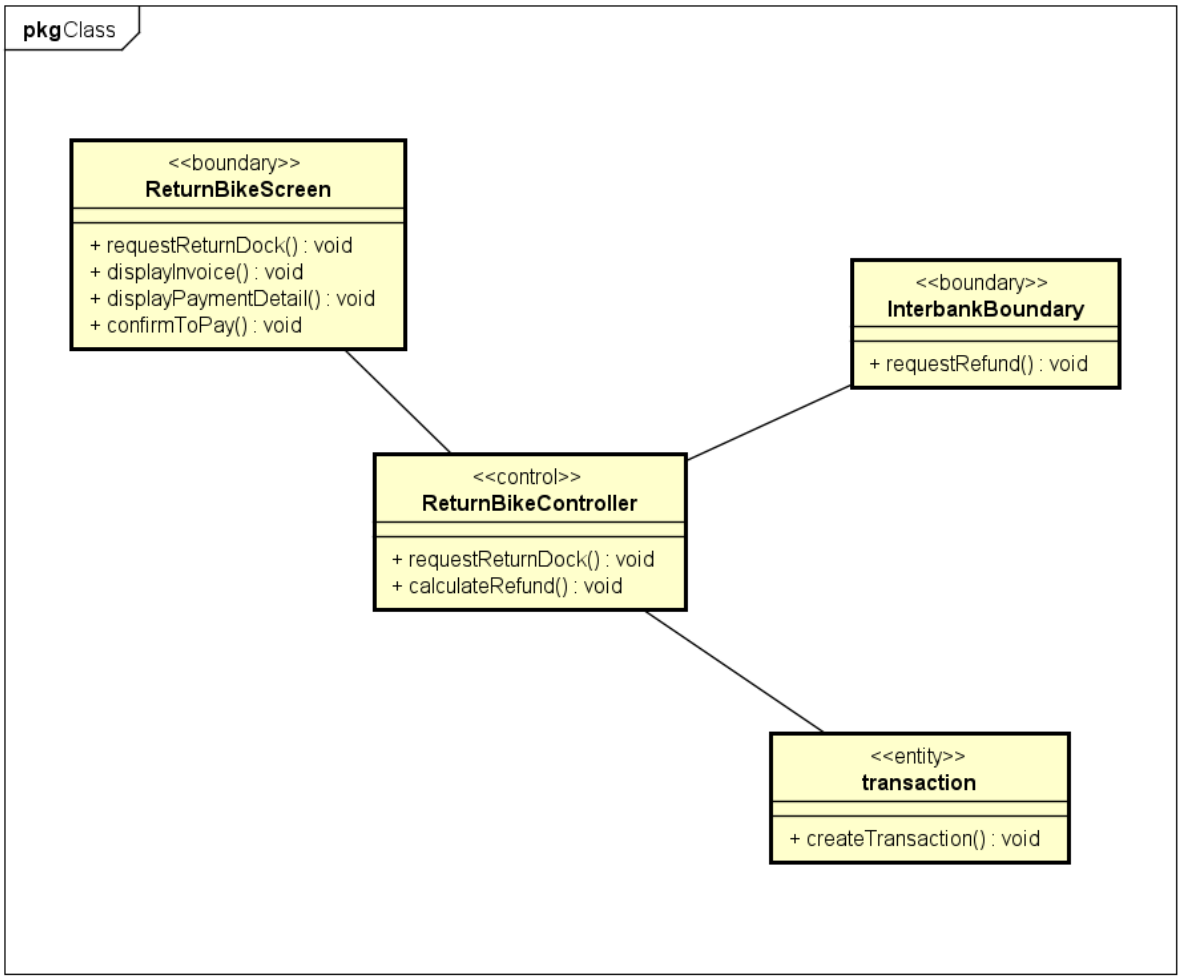
3.3. Analysis Class Diagrams



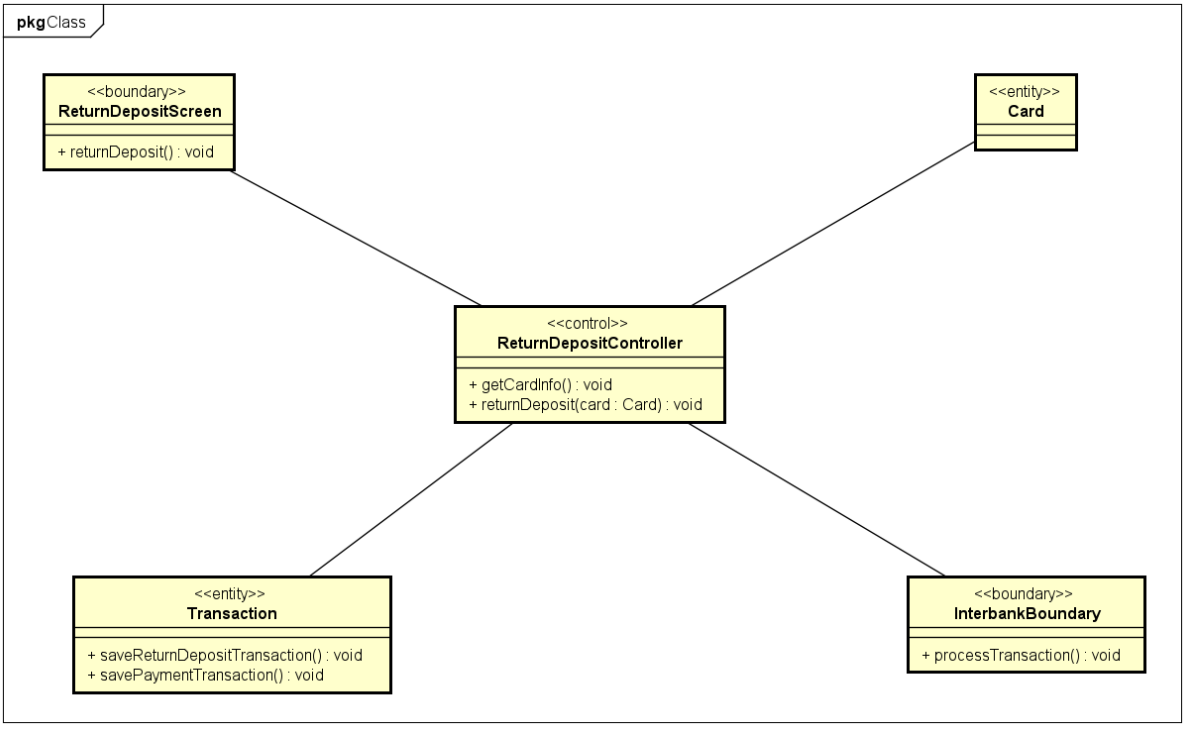
Class Diagram for Rent Bike Use Case



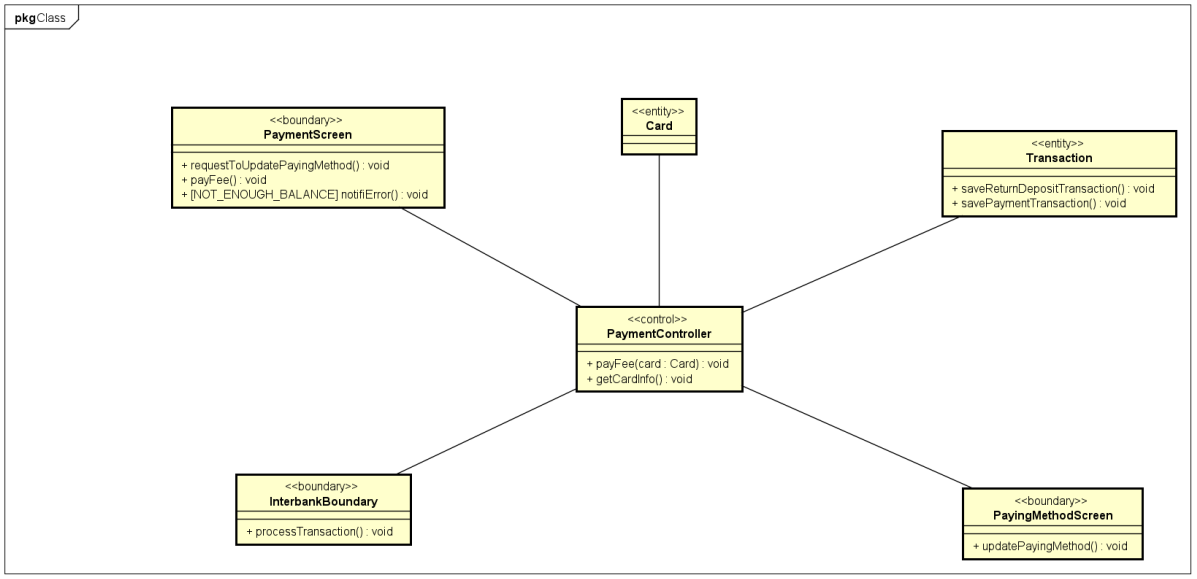
Class Diagram for Deposit Use Case



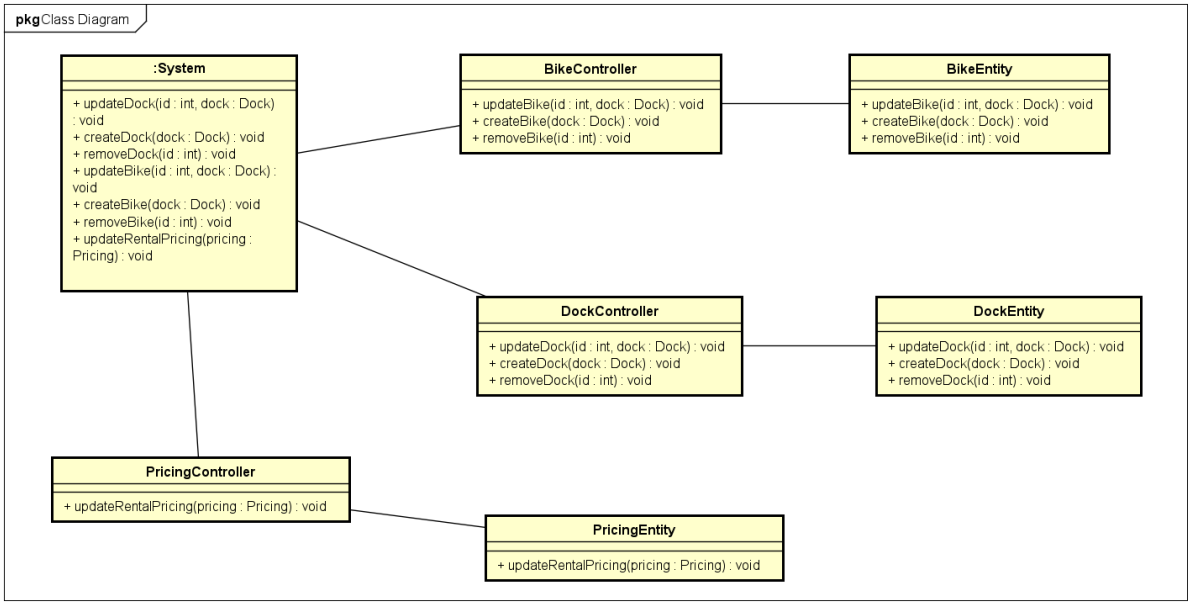
Class Diagram for Return Bike Use Case



Class Diagram for Return Deposit Use Case



Class Diagram for Pay Rental Use Case



Class Diagram For Admin

3.4. Security Software Architecture

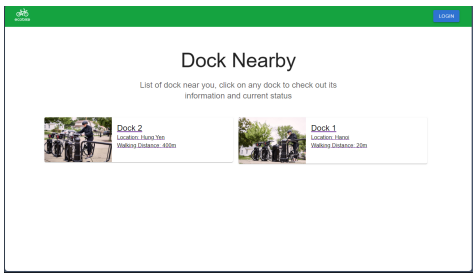
In this project, we will not consider features such as user authentication (e.g., sign up, sign in, sign out), we only focus on features related to rent and return bikes.

4. Detailed Design

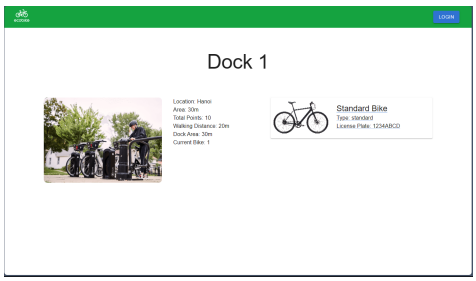
4.1. User Interface Design

4.1.1.1. Main Screen

Ecobike Software	Date of Creation	Approved by	Reviewed by	Person in charge

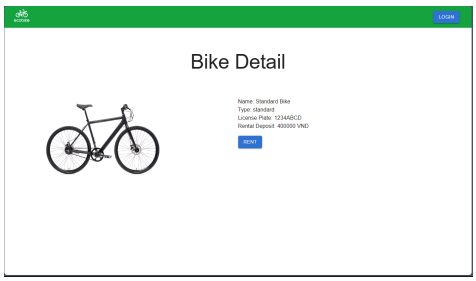
Screen specification	Main Screen	25/7/2023			Pham Tuan Long
		**Control**	**Operation**	**Function**	
		Logo	Click	Redirect to Dock List screen	
		Dock	Click	Navigate to Dock detail screen	
		Dock List	Initial	Show all docks nearby with several basic information	

4.1.1.2. Dock Screen

Ecobike Software		Date of Creation	Approved by	Reviewed by	Person in charge
Screen specification	Dock Screen	25/7/2023			Pham Tuan Long
		**Control**	**Operation**	**Function**	
		Logo	Click	Redirect to Dock List screen	
		Dock detail	Initial	DIsplay the information of the dock, follow by available bike	
		Bike list	Click	Redirect to Bike detailed screen	

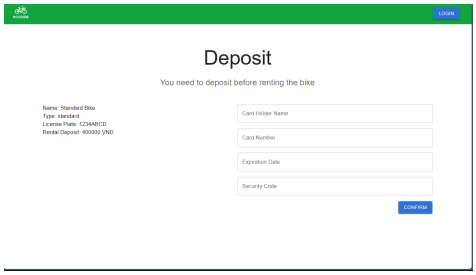
4.1.1.3. Bike Screen

Ecobike Software		Date of Creation	Approved by	Reviewed by	Person in charge
Screen specification	Bike Detail Screen	25/7/2023			Pham Tuan Long

	Control	Operation	Function
	Logo	Click	Redirect to Dock List screen
	Bike Detail	Initial	Show the detail information about the bike
	Rent button	Click	Redirect to deposit screen with the information of the bike

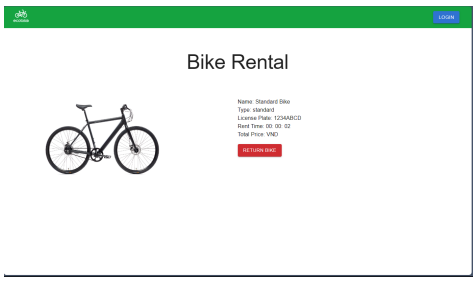
4.1.1.4. Deposit Screen

Ecobike Software		Date of Creation	Approved by	Reviewed by	Person in charge
Screen specification	**Deposit Screen**	25/7/2023			Pham Tuan Long

	Control	Operation	Function
	Logo	Click	Redirect to Dock List screen
	Card Information	Type	Display the input information
	Confirm button	Click	Confirm deposit, redirect to rental screen if success
	Bike information	Inital	Display the information of the bike in the rental

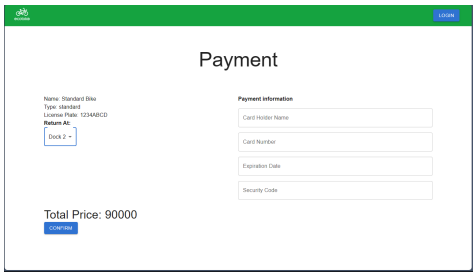
4.1.1.5. Rental Screen

Ecobike Software		Date of Creation	Approved by	Reviewed by	Person in charge
Screen specification	**Rental Screen**	25/7/2023			Pham Tuan Long

	Control	Operation	Function
	Logo	Click	Redirect to Dock List screen
	Return bike button	Click	Redirect to Payment screen
	Rental information	Initial	Realtime update the information and fee calculation based on time rented

4.1.1.6 Payment Screen

Ecobike Software		Date of Creation	Approved by	Reviewed by	Person in charge
Screen specification	**Payment Screen**	25/7/2023			Pham Tuan Long

	Control	Operation	Function
	Logo	Click	Redirect to Dock List screen
	Card Information	Type	Display the input information
	Confirm button	Click	Confirm deposit, redirect to main screen if success
	Rental information	Initial	Display the rental information in detail

4.2. Data Modeling

4.2.1. Conceptual Data Modeling

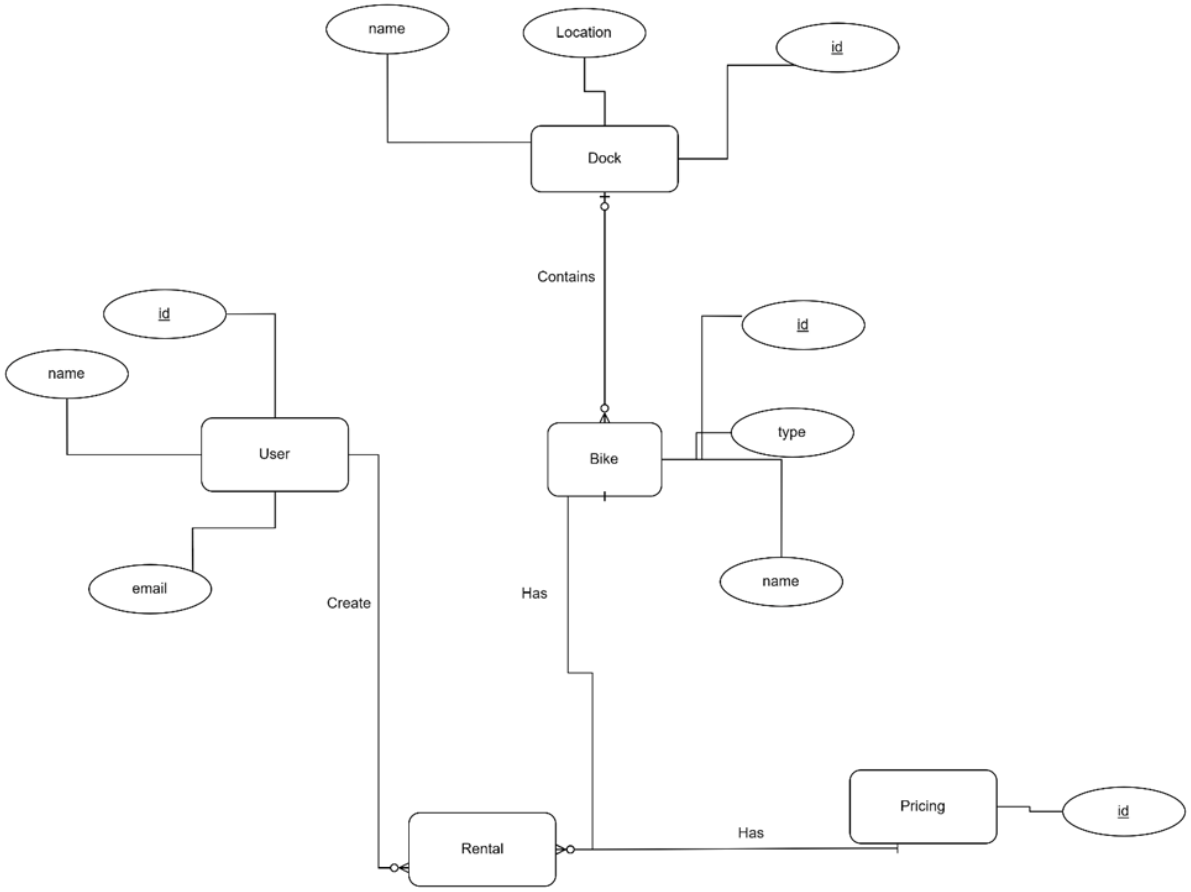
4.2.2. Database Design

4.2.2.1. Database Management System

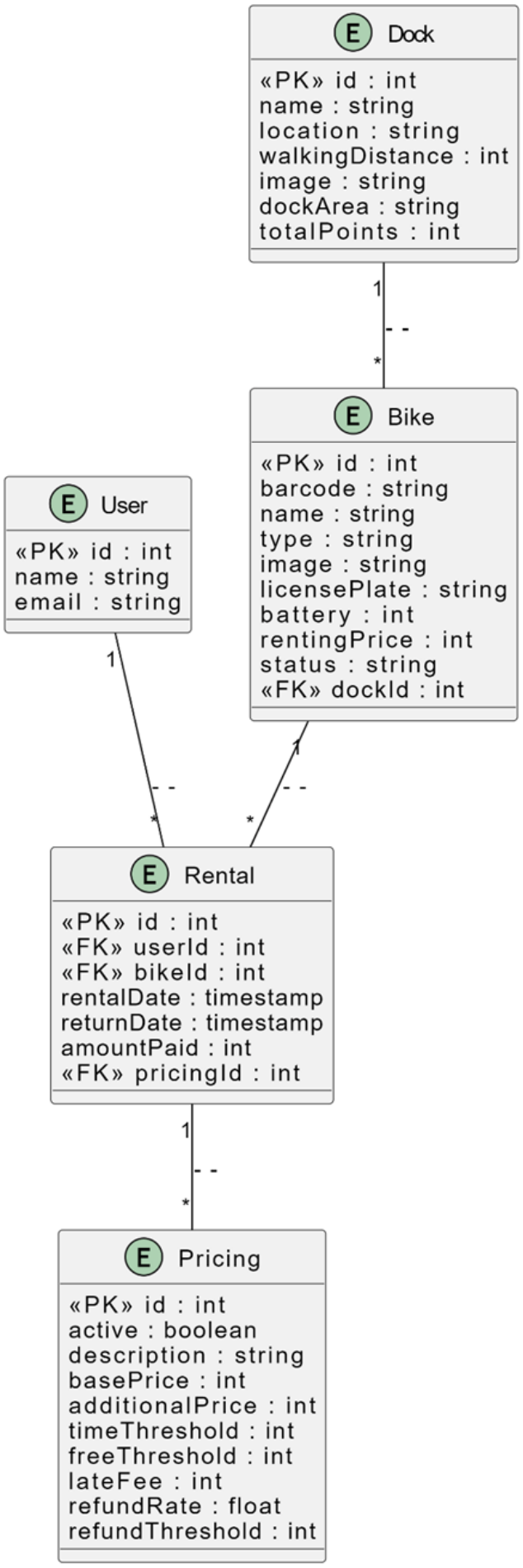
Database Management System: PostgreSql

4.2.2.2. Database Diagram

4.2.2.2.1 ERD



4.2.2.2.2 Database Schema


Dock
«PK» id : int
name : string
location : string
walkingDistance : int
image : string
dockArea : string
totalPoints : int
1
*
Bike
«PK» id : int
barcode : string
name : string
type : string
image : string
licensePlate : string
battery : int
rentingPrice : int
status : string
«FK» dockId : int
User
«PK» id : int
name : string
email : string
1
*
1
*
Rental
«PK» id : int
«FK» userId : int
«FK» bikeId : int
rentalDate : timestamp
returnDate : timestamp
amountPaid : int
«FK» pricingId : int
1
*
Pricing
«PK» id : int
active : boolean
description : string
basePrice : int
additionalPrice : int
timeThreshold : int
freeThreshold : int
lateFee : int
refundRate : float
refundThreshold : int

4.2.2.3. Database Detail Design

- **Bike**

No.	PK	FK	Name	Data type	Mandatory	Description
1	Yes	No	id	INT	Yes	Primary key - Auto-generated unique identifier.
2	No	No	barcode	VARCHAR(255)	Yes	Unique barcode associated with the bike.
3	No	No	name	VARCHAR(255)	Yes	Name of the bike.
4	No	No	type	ENUM	No	Type of the bike (e.g., STANDARD, etc.).
5	No	No	image	VARCHAR(255)	No	URL or path to the image of the bike.
6	No	No	licensePlate	VARCHAR(255)	Yes	License plate number of the bike.

7	No	No	battery	NUMERIC	Yes	Battery level of the bike.
8	No	No	rentingPrice	NUMERIC	Yes	Renting price for the bike.
9	No	No	status	ENUM	No	Current status of the bike (e.g., FREE, etc.).
10	No	Yes	dockId	INT	No	Foreign key referencing the Dock table.

- **Dock**

No.	PK	FK	Name	Data type	Mandatory	Description
1	**Yes**	**No**	**id**	**INT**	**Yes**	**Primary key - Auto-generated unique identifier.**
2	**No**	**No**	**name**	**VARCHAR(255)**	**No**	**Name of the dock.**

3	No	No	location	VARCHAR(255)	No	Location of the dock.
4	No	No	walkingDistance	NUMERIC	No	Walking distance to the dock.
5	No	No	image	VARCHAR(255)	No	URL or path to the image of the dock.
6	No	No	dockArea	VARCHAR(255)	No	Area of the dock.
7	No	No	totalPoints	NUMERIC	No	Total points associated with the dock.

- **Pricing**

No.	PK	FK	Name	Data type	Mandatory	Description
1	Yes	No	id	INT	Yes	Primary key - Auto-generated unique identifier.

2	No	No	active	BOOLEAN	No	Indicates if the pricing is currently active.
3	No	No	description	VARCHAR(255)	Yes	Description of the pricing.
4	No	No	basePrice	NUMERIC	Yes	Base price in VND.
5	No	No	additionalPrice	NUMERIC	Yes	Additional price in VND.
6	No	No	timeThreshold	NUMERIC	Yes	Time threshold in minutes for regular pricing.
7	No	No	freeThreshold	NUMERIC	No	Time threshold in minutes for free period.
8	No	No	lateFee	NUMERIC	Yes	Late fee in VND.

9	No	No	refundRate	NUMERIC	Yes	Refund rate as a decimal (e.g.15000VNĐ for each 15 min).
10	No	No	refundThreshold	NUMERIC	Yes	Refund threshold in hours.

- **Rental**

No.	PK	FK	Name	Data type	Mandatory	Description
1	Yes	No	id	INT	Yes	Primary key - Auto-generated unique identifier.
2	No	Yes	userId	INT	Yes	Foreign key referencing the User table.
3	No	Yes	bikeId	INT	Yes	Foreign key referencing the Bike table.
4	No	No	rentalDate	TIMESTAMP	Yes	Timestamp when

						the rental started.
5	**No**	**No**	**returnDate**	**TIMESTAMP**	**No**	**Timestamp when the rental was returned.**
6	**No**	**No**	**amountPaid**	**NUMERIC**	**Yes**	**Amount paid for the rental in VND.**
7	**No**	**Yes**	**pricingId**	**INT**	**Yes**	**Foreign key referencing the Pricing table.**

5. Design Considerations

5.1. Goals and Guidelines

Goals:

- Provide a user-friendly application
- Provide an eye-catching interface and convenient experience for users
- The response time for the system is 1 second at normal and 2 seconds during a peak load

Guidelines:

- Avoid hard-coding
- Write comments for codes
- Structure the doc for maintenance

5.2. Architectural Strategies

Our intention is to reuse components

- Frontend: ReactJS
- Database: Postgresql
- UML: Astah

- Backend: NestJS

We're always looking toward minimizing the memory and space usage; reduce the complexity to speed up the response time, and improve the performance. We're also concerned about the maintenance. For the future, we're looking forward to updating the system, integrating new features such as admin to manage the crud, the statistics, the profit.

5.3. Design Principles

We design simple classes follow SOLID principles that means:

- A class should have only one job, one responsibility.
- Software entities are open for extension but close for modification.
- We also use interfaces, abstract classes. So, subclasses should be substitutable for their base classes.
- Use specific interfaces if necessary instead of using general purpose interfaces which do not use.
- We put all classes with the same properties into one package to manage easily. Therefore, we can reuse source code, adapt to any changing requirements.

5.4. Design Patterns

Facade pattern:

- We use InterbankInterface for communication between software and interbank subsystem. It decrease the overall complexity of our application and provides an easier interface for communication

Singleton pattern:

- We use singleton pattern for screen handler so that we do not need to create new instance of screen handler each time we change the screen

Factory pattern:

- Use this pattern to create instances of bike objects based on their types. It allows you to encapsulate the instantiation logic and delegate the object creation to subclasses or factory methods.

Observer pattern:

- Apply this pattern to establish a one-to-many relationship between objects. When one object changes state, all its dependents are notified and updated automatically.
- Use the observer pattern to notify users about changes in bike availability, status, or rental transactions.