Software Requirements and Design Document

for

Bike Borrow Rental System

**Prepared by:**

**Muzzamil Hissan 22i-0881**

**Saad Awais 22i-1303**

**Nouman Nawaz 22i-0752**

**Organization:**

**Department of Computer Science FAST NUCES**

**Date created:**

**26/11/24**

**Table of Contents**

**Table of Contents ii**

**1.** **Introduction 1**

1.1 Purpose 1

1.2 Product Scope 1

1.3 Title 1

1.4 Objectives 1

1.5 Problem Statement 1

**2.** **Overall Description 1**

2.1 Product Perspective 1

2.2 Product Functions 2

2.3 List of Use Cases 2

2.4 Extended Use Cases 2

2.5 Use Case Diagram 2

**3.** **Other Nonfunctional Requirements 2**

3.1 Performance Requirements 2

3.2 Safety Requirements 2

3.3 Security Requirements 2

3.4 Software Quality Attributes 2

3.5 Business Rules 3

3.6 Operating Environment 3

3.7 User Interfaces 3

**4.** **Domain Model 3**

**5.** **System Sequence Diagram 3**

**6.** **Sequence Diagram 3**

**7.** **Class Diagram 4**

**8.** **Package Diagram 4**

**9.** **Deployment Diagram 4**

# Introduction

## Purpose

*The purpose of the Bike Borrow Rental System is to create a robust and efficient solution for managing bike rentals. It aims to streamline the rental process for users while providing features for administrators to manage inventory and maintenance efficiently. The system includes bike renting, returning, damage reporting, and administrative functionalities.*

## Product Scope

*The Bike Borrow Rental System is a web-based application designed for:*

* *Managing bike rentals for users.*
* *Allowing users to report bike damage.*
* *Enabling administrators to manage bike inventory, approve damage reports, and track system activities. The system is aimed at simplifying operations for bike-sharing services and ensuring a seamless experience for users and administrators.*

## Title

***Bike Borrow Rental System*** *– Streamlining Bike Rentals and Maintenance*

## Objectives

* *Simplify the bike rental and return processes.*
* *Enable efficient inventory and damage report management for administrators.*
* *Provide users with an intuitive platform for renting bikes and submitting damage reports.*
* *Ensure data integrity and secure handling of sensitive information.*

## Problem Statement

*Bike-sharing services often face issues with manual processes, such as tracking inventory, managing rentals, and handling damage reports. These inefficiencies can result in poor user experiences, loss of revenue, and logistical challenges. The Bike Borrow Rental System addresses these challenges by digitizing and automating the rental and maintenance workflows. The system ensures accurate record-keeping, reduces manual intervention, and provides detailed insights into the system's usage and performance.*

# Overall Description

## Product Perspective

*The Bike Borrow Rental System is a new, self-contained web application. It integrates user and admin functionalities, allowing seamless coordination between bike renters and the administrative team. It replaces traditional manual systems with a digital platform for enhanced accuracy and efficiency.*

## Product Functions

***User Features:***

* *Register and log in to the platform.*
* *View available bikes.*
* *Rent and return bikes.*
* *Submit damage reports.*

***Admin Features:***

* *Add or remove bikes from the inventory.*
* *View and manage damage reports.*
* *Approve or reject damage reports.*
* *Monitor active rentals and system activities.*

## List of Use Cases

***1. Register User***

***2. Log In User***

***3. Browse Available Bikes***

***4. View Bike Details***

***5. Confirm Bike Rental***

***6. Process Payment***

***7. Track Rental Status***

***8. View Rental History***

***9. Submit Damage Report***

***10. Return Bike***

***11. Monitor Bike Inventory***

***12. View Active Rentals***

***13. Issue Maintenance Requests***

## Extended Use Cases

**1. Register User**

**Component Details Saad Use Case Name** Register User

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders**:

● **User**: Wants to create an

account to rent bikes.

● **System**: Must validate

and store the user's

registration details

securely.

● **Mobile Network**

**Provider**: Provides SMS

services to deliver

verification codes to the

user (only applicable if

SMS verification is part of

the registration process).

**Preconditions** 1. The user has access to the internet and

navigates to the

registration page.

2. The user has a valid

email address and mobile

number (optional, if SMS

verification is needed).

3. The system is available

and responsive.

**Postconditions** ● **Success**: The user is successfully registered,

and their details (e.g.,

name, contact, payment

information) are securely

stored.

● **Failure**: The system does

not store any details if the

registration is

unsuccessful (due to

invalid data or system

errors).

**Main Success Scenario User**

1.User enters personal

and payment details.

3.System creates a new

account and stores data.

**Extensions 2a. Invalid Data**:

**System**

2. System validates inputs.

● 2a.1: System

detects invalid or

missing data (e.g., email format,

password strength). ● 2a.2: System

highlights errors

and prompts the

user to re-enter the

data

**3a. Payment Information**

**Not Entered (if optional)**:

● 3a.1: System

proceeds without

payment

information if it’s not

mandatory for

registration.

● 3a.2: If payment

info is required,

system prompts

user to enter the

necessary details.

**4a. User Already**

**Registered**:

● 4a.1: System

detects an account

associated with the

entered email or

phone number.

● 4a.2: System

displays an error

message and

prompts the user to

log in or recover

their account..

**Alternative Flow**

**Unverified User Flow: If the**

**system requires email or SMS**

**verification:**

**1.1:** After

registering, the

system sends a

verification link or

code to the user.

**1.2:** User clicks the link or

enters the code to verify

their account.

**1.3:** System verifies

the user’s account

and grants access

to the app's full

functionality.

**2. Log In User**

**Component Details Saad**

**Use Case Name** Log In User

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders** User: Wants to access their account.

System: Responsible for validating login credentials and managing user sessions.

**Preconditions** User has a valid account.  The user is on the login page.

**Postconditions** User is logged in, and session is active. 

**Main Success Scenario**

**User**

1. User enters login credentials.

3. User is logged in and directed to the dashboard.

**System**

2. System validates credentials. 

**Extensions** 2a. Invalid credentials: System shows error and  prompts for re-entry.

**3. Browse Available Bikes**

**Component Details Saad**

**Use Case Name** Browse Available Bikes

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders** ● User: Wants to browse

and rent available bikes.

● System: Responsible for

managing the inventory

and availability of bikes

in real-time.

**Preconditions** ● User is logged in.

● The system has

up-to-date information

on bike inventory and

availability.

**Postconditions** ● A list of available bikes

is displayed for the user

to browse.

● The user can view

detailed information on

each bike, such as

location, rental price,

and specifications. 

**Main Success Scenario**

**User**

1.user navigates to the "Browse Bikes" section

4.user selects a bike to view detailed specifications

**System**

2.system retrieves bikes that are currently available for rental 3.system shows a list of available bikes

the system displays a detailed page with comprehensive information. 

**Extensions** 2a. No bikes available: System shows a notification of unavailability.

3a. Retrieval error:System

encounters an error when retrieving

bike listings.

5a. Limited availability

notification:The selected bike has

limited availability.

**Special**

● Real-time availability: 

**Requirements**

The system must update bike availability promptly as rentals are

processed.

● Location-based filtering: Users should be able to filter bikes by proximity to their current or

selected location. 

**4. View Bike Details**

**Component Details Saad**

**Use Case Name** View Bike Details

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders** User: Wants to view details

before renting.

Bike-Borrow Rental System:

Responsible for managing the

inventory and availability of

bikes in real-time.

**Preconditions** The system has up-to-date

information on bike inventory

and availability.

**Postconditions** User views bike specifications

and condition. 

**Main Success Scenario**

**User**

1. User selects a bike from the list.

**System**

2. System displays bike

specifications (e.g., type, condition).

**Extensions** None.

**5. Rent Bike**

**Component Details Saad**

**Use Case Name** Rent Bike

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders** User: Wants to rent a bike.

Bike-Borrow Rental System:

Facilitates the rental process

and ensures accurate billing.

**Preconditions** User has selected a bike and

bike is ready to rent.

**Postconditions** Bike rental is confirmed and

logged in the system. 

**Main Success Scenario**

**User**

1. User selects rental duration.

3. User confirms rental.

**System**

2. System calculates rental fees. 4.System processes rental. 

**Extensions** 2a. Insufficient funds: System

displays payment error.

**6. Process Payment**

**Component Details Saad**

**Use Case Name** Process Payment

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** Third-Party Payment Gateway

**Stakeholders** User: Wants to pay rental fees securely.

Bike-Borrow Rental System:

Ensures secure payment

processing and accurate

transaction records.

Third-Party Payment Gateway

(e.g., Bank): Verifies,

authorizes, and processes the

payment transaction on behalf

of the user.

**Preconditions** Rental details are confirmed, and payment details are

provided.

**Postconditions** Payment is processed and receipt is generated

successfully. 

**Main Success Scenario**

**User**

1. User enters payment information.

**System**

2. System processes payment via payment gateway.

3. System confirms successful payment. 

**Extensions** 2a. Payment failure: System prompts for alternative payment

method.

**7. Track Rental Status**

**Component Details Nouman**

**Use Case Name** Track Rental Status

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders** User: Wants to know the status of their rental.

Bike-Borrow Rental

System: Manages and

updates rental status

information in real-time to

provide accurate data to the

user.

**Preconditions** Bike is currently rented.

**Postconditions** User can see the status and return date of the bike. 

**Main Success Scenario**

**User**

1. User requests rental status.

3. User views rental status.

**System**

2. System retrieves rental details. 

**Extensions** None.

**8. View Rental History**

**Component Details Nouman Use Case Name** View Rental History

**Scope** Bike-Borrow Rental System

**Level** User-goal

**Primary Actor** User

**Stakeholders** User: Wants to see a log of their

past rentals.

Bike-Borrow Rental System:

**Preconditions** User is logged in.

**Postconditions** User views past rentals and

their details. 

**Main Success Scenario**

**User**

1. User selects to view rental history.

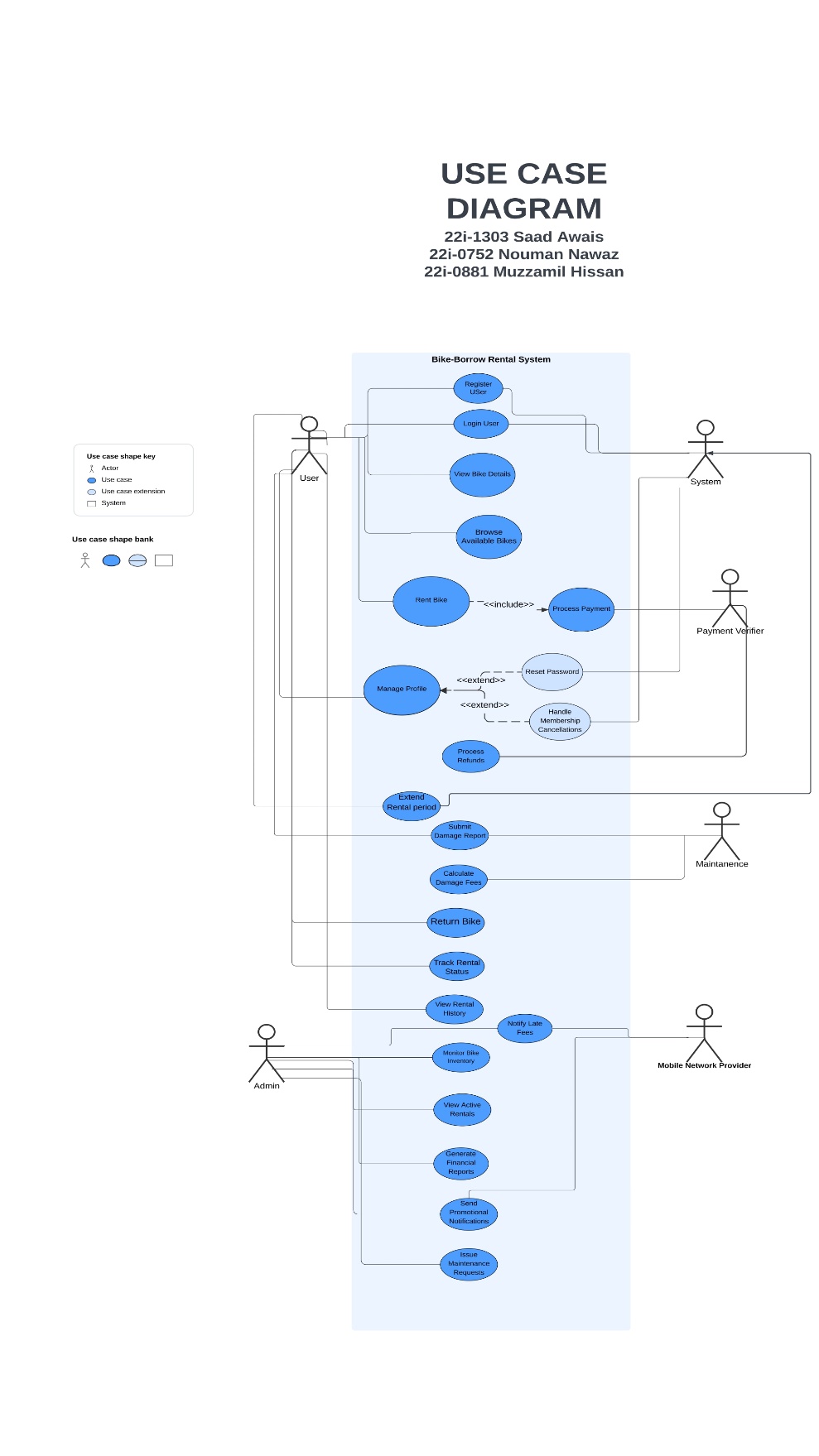
3. User views details of past rentals.

**System**

2. System retrieves history.

**Extensions** None.

## Use Case Diagram



# Other Nonfunctional Requirements

## Performance Requirements

*The system should handle concurrent users without significant performance degradation.*

*All operations, such as renting a bike or submitting a damage report, should complete within 2 seconds.*

## Safety Requirements

* *All user and transaction data should be backed up regularly to prevent data loss.*
* *Damage reports and critical system changes should log details for auditing purposes.*

## Security Requirements

* *User authentication is required to access rental and reporting functionalities.*
* *Admin functionalities should only be accessible to authenticated admins.*
* *Data transfer must use HTTPS to ensure secure communication.*

## Software Quality Attributes

* ***Reliability:*** *Ensure 99.9% uptime for uninterrupted user access.*
* ***Usability:*** *Provide an intuitive interface for both users and administrators.*
* ***Scalability:*** *Support increased user load and additional features as needed.*

## Business Rules

* *Users can only rent one bike at a time.*
* *Damage reports must be approved or rejected.*
* *Admin actions are restricted to authenticated users with the role of "admin."*

## Operating Environment

***Frontend:*** *React.js for user and admin interfaces.*

***Backend:*** *Spring Boot for REST API services.*

***Database:*** *MySQL for data persistence.*

***Hosting:*** *Deployed on a local host for now.*

## User Interfaces

* *User Dashboard: Displays bike inventory, rentals, and report submission options.*
* *Admin Dashboard: Displays inventory, damage reports, and system logs.*

# Domain Model

# System Sequence Diagram

A diagram of a login user

Description automatically generatedA diagram of a bike list

Description automatically generatedA diagram of a bike list

Description automatically generatedA diagram of a diagram

Description automatically generatedA diagram of a bike rental

Description automatically generatedA diagram of a payment method

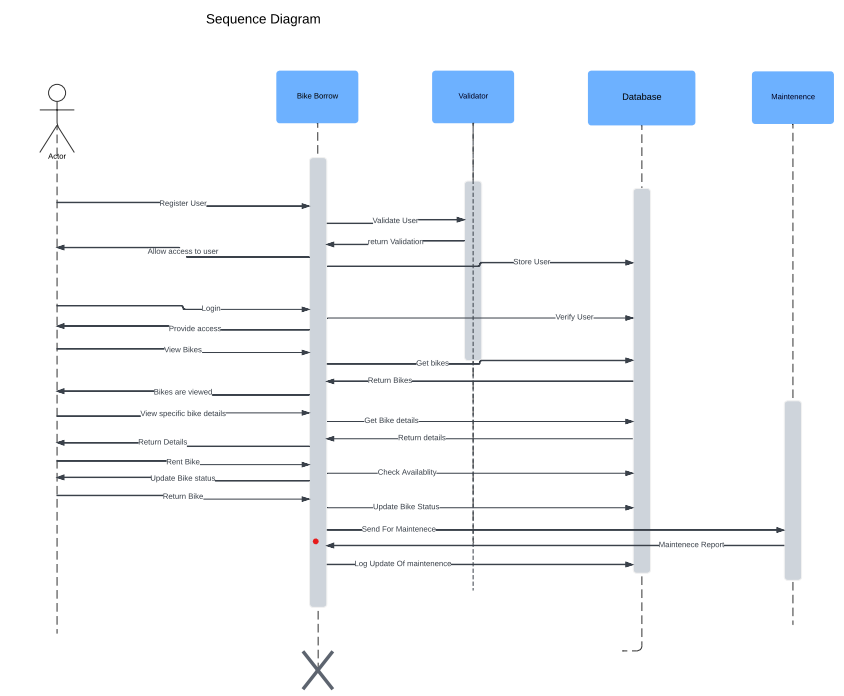
Description automatically generatedA diagram of a problem

Description automatically generatedA diagram of a bike

Description automatically generatedA diagram of a computer error

Description automatically generated

# Sequence Diagram



# Class Diagram

# Component Diagram

A screenshot of a computer

Description automatically generated

# Package Diagram

A diagram of a software system

Description automatically generated with medium confidence

# Deployment Diagram

A diagram of a diagram

Description automatically generated