Sri Eshwar
College of Engineering
An Autonomous Institution
Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Academic Year

2023 – 2024 (Odd Semester)

MINI PROJECT REPORT

R19CS202 Database Management Systems

R19CS203 Object-Oriented Programming

R19IT251 Software Engineering

------ BIKE TRACKING SYSTEM -----

Submitted by,

[22AD078] SAM MESHACH A

TABLE OF CONTENTS

ABSTRACT

The Bike Tracking System aims to enhance the management and tracking of bikes within a fleet. This comprehensive system enables real-time monitoring of bike locations, maintenance history, and user data. It also provides administrative tools to streamline bike usage and maintenance operations. The system's core features include a Bikes Table for storing essential bike information, User Table for managing user details securely, a Rides Table for tracking ride details, a Maintenance Schedule Table for scheduling and recording maintenance activities, and a Locations Table for managing bike locations. Additionally, the system incorporates a Payment Transactions Table to record user payments for rides and a Feedback Table to store user feedback and ratings. The system utilizes JDBC connectivity to facilitate Bike Tracking, User Management, Ride Operations, Maintenance Tracking, Admin Tools, and Reporting/Analytics.

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVES

The primary objectives of the Bike Tracking System are to enhance bike fleet management by providing real-time tracking, streamlining maintenance activities, ensuring secure user management, and offering administrative tools for system control.

1.2 SCOPE OF THE PROJECT

The scope of the project encompasses the development of a comprehensive Bike Tracking System. It includes functionalities for tracking bike locations, managing user data securely, recording ride details, scheduling maintenance, providing administrative tools, and generating reports for analysis.

CHAPTER 2

SYSTEM ANALYSIS AND SPECIFICATION

2.1 PROBLEM DESCRIPTION

The Bike Tracking System addresses the need for efficient tracking and management of bikes within a fleet. Common challenges such as real-time monitoring, secure user management, and streamlined maintenance operations are tackled through the system's features

2.2 FUNCTIONAL REQUIREMENT –

SOFTWARE AND HARDWARE REQUIREMENT

The system requires standard hardware components (servers, databases) and software tools (programming languages, database management systems) to ensure seamless functionality. Detailed specifications for hardware and software requirements are outlined in this section.

2.3 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements, such as system reliability, performance, and security, are critical for the successful implementation and operation of the Bike Tracking System. These aspects are thoroughly examined and specified in this section.

CHAPTER 3

SYSTEM DESIGN

3.1 SCHEMA DIAGRAM

Bikes Table:

- BikeID (Primary Key)
- BikeNumber
- Status (Available, In Use, Under Maintenance)
- LocationID (Foreign Key)
- MaintenanceHistoryID (Foreign Key)

Users Table:

- UserID (Primary Key)
- Username
- Password
- Email
- PhoneNumber

Rides Table:

- RideID (Primary Key)
- BikeID (Foreign Key)
- UserID (Foreign Key)
- StartTime
- EndTime
- DistanceCovered
- RideFare

Maintenance Schedule Table:

- MaintenanceID (Primary Key)
- BikeID (Foreign Key)
- ScheduledDate
- MaintenanceType

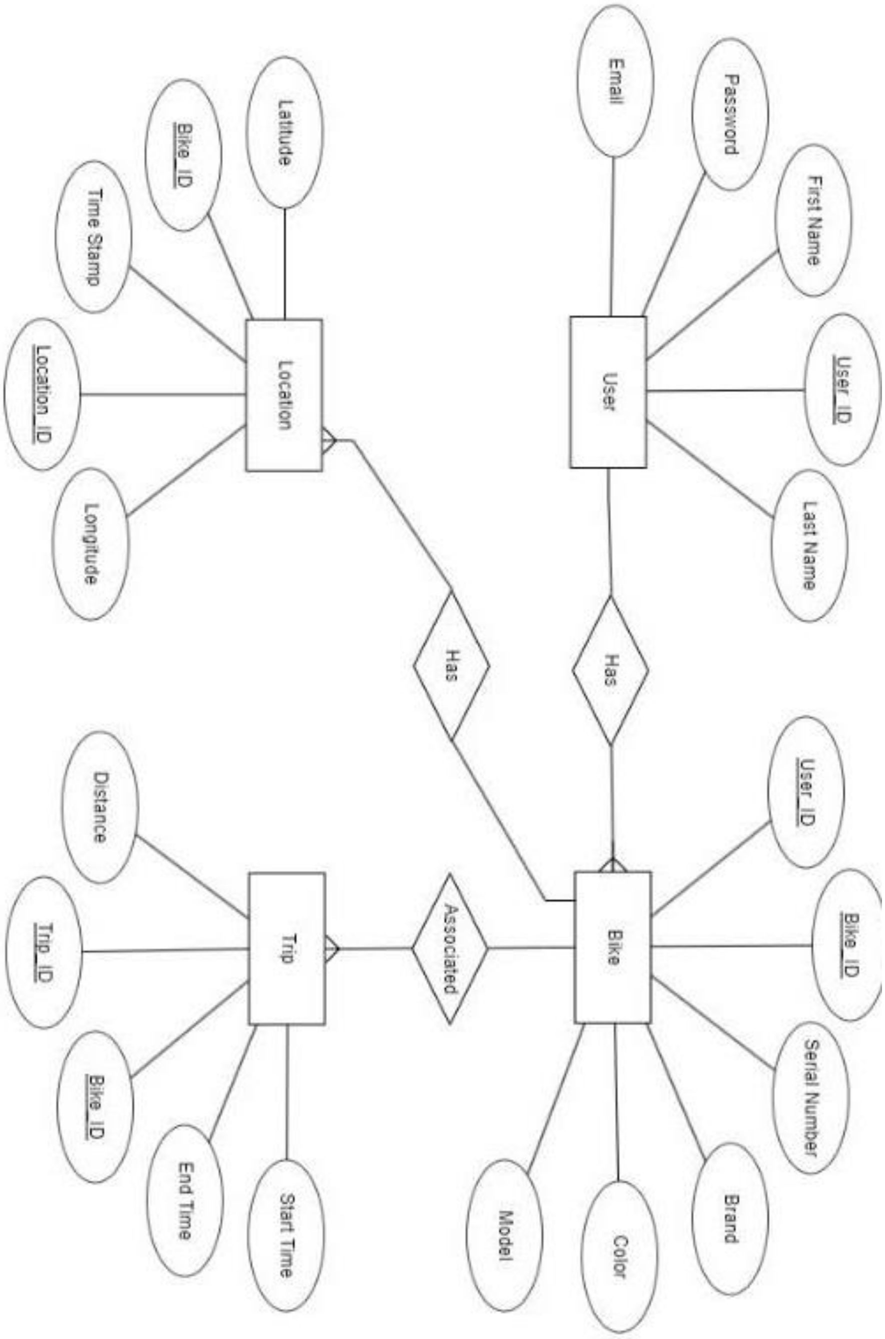
Locations Table:

- LocationID (Primary Key)
- Name
- Address
- Latitude
- Longitude

Payment Transactions Table:

- TransactionID (Primary Key)
- UserID (Foreign Key)
- RideID (Foreign Key)
- TransactionAmount
- Timestamp

3.2 ER DIAGRAM



CHAPTER 4

PROPOSED SOLUTION

4.1 USER INTERFACE DESIGN

The user interface design aims to provide an intuitive and efficient experience for users interacting with the Bike Tracking System. It will feature sections for bike tracking, user management, ride operations, maintenance tracking, admin tools, and reporting/analytics. Each section will have a user-friendly layout, ensuring easy navigation and accessibility.

4.2 CLASS CONSTRUCTION

The system's class construction involves organizing and structuring the code into classes to achieve modularity and maintainability. Key classes will include those for bikes, users, rides, maintenance schedules, locations, payment transactions, and feedback. These classes will encapsulate the relevant data and functionalities associated with each entity.

4.3 DATABASE CREATION

The database will play a crucial role in storing and managing data for the Bike Tracking System. Tables corresponding to bikes, users, rides, maintenance schedules, locations, payment transactions, and feedback will be created. The relationships between these tables will be defined to maintain data integrity. Additionally, the database will support efficient retrieval and updating of information.

CHAPTER 5

PROJECT DESCRIPTION

5.1 MODULE DESCRIPTION

5.1 Bike Tracking Module

The Bike Tracking module focuses on real-time monitoring of bike locations and statuses. Users can quickly identify available bikes, making the system responsive to user demands. This functionality involves leveraging JDBC connectivity for seamless communication with the database.

5.1.1 User Management Module

User Management ensures secure registration and management of user details. It involves the creation of a Users Table in the database, where encrypted passwords and personal information are stored. JDBC connectivity is utilized for effective communication between the application and the database.

5.1.2 Ride Operations Module

Ride Operations initiate and conclude rides, capturing essential ride details like duration, distance, and fare calculation. The module utilizes JDBC connectivity to record and retrieve ride information from the Rides Table in the database.

5.1.3 Maintenance Tracking Module

Maintenance Tracking schedules bike maintenance, tracks service history, and updates maintenance schedules. JDBC connectivity facilitates seamless interaction with the Maintenance Schedule Table in the database, ensuring accurate and timely maintenance records.

5.1.4 Admin Tools Module

Admin Tools provide administrative functionalities, such as adding new bikes, updating bike statuses, and managing system data. These operations involve JDBC connectivity to modify the Bikes Table and other relevant tables in the database.

5.1.5 Reporting and Analytics Module

The Reporting and Analytics module generates reports on user ride history, bike usage patterns, and maintenance schedules. JDBC connectivity is crucial for extracting and analyzing data from different tables in the database, supporting informed decision-making.

5.2 JDBC CONNECTIVITY

JDBC (Java Database Connectivity) connectivity is a crucial component in the Bike Tracking System, facilitating seamless communication between the application and the underlying database. This technology allows Java applications to interact with relational databases, enabling operations such as data retrieval, insertion, update, and deletion. In the context of the Bike Tracking System, JDBC connectivity is instrumental in implementing various functionalities:

Bike Tracking:

JDBC is utilized to retrieve real-time data on bike locations and statuses. The system can efficiently query the database to provide users with up-to-date information on the availability and current status of bikes within the fleet.

User Management:

For secure user registration and login processes, JDBC ensures a secure interaction between the system and the Users Table in the database. Encrypted passwords and user details are retrieved and stored using JDBC connections.

Ride Operations:

JDBC connectivity is employed to record and retrieve ride details from the Rides Table. When a user initiates or concludes a ride, the system utilizes JDBC to update the relevant information in the database, including start and end times, distance covered, and ride fares.

Maintenance Tracking:

Scheduling and tracking bike maintenance involve frequent interactions with the Maintenance Schedule Table. JDBC enables the system to efficiently manage maintenance records, update schedules, and retrieve historical data regarding bike servicing.

Admin Tools:

Administrative functionalities, such as adding new bikes or updating bike statuses, heavily rely on JDBC connectivity. The system interacts with the Bikes Table and other relevant tables to make real-time updates and modifications.

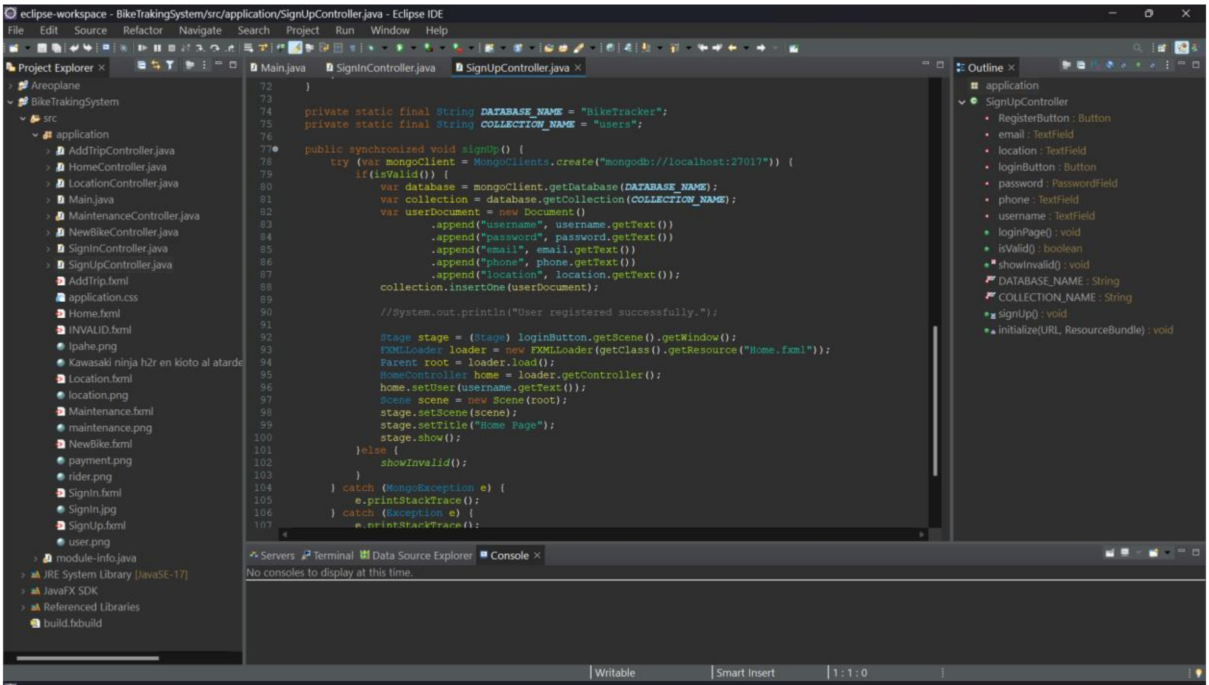
Reporting and Analytics:

Generating reports on user ride history, bike usage patterns, and maintenance schedules necessitates extracting and analyzing data from various tables. JDBC connectivity ensures a smooth flow of data, allowing the system to present valuable insights for decision-making.

CHAPTER 6

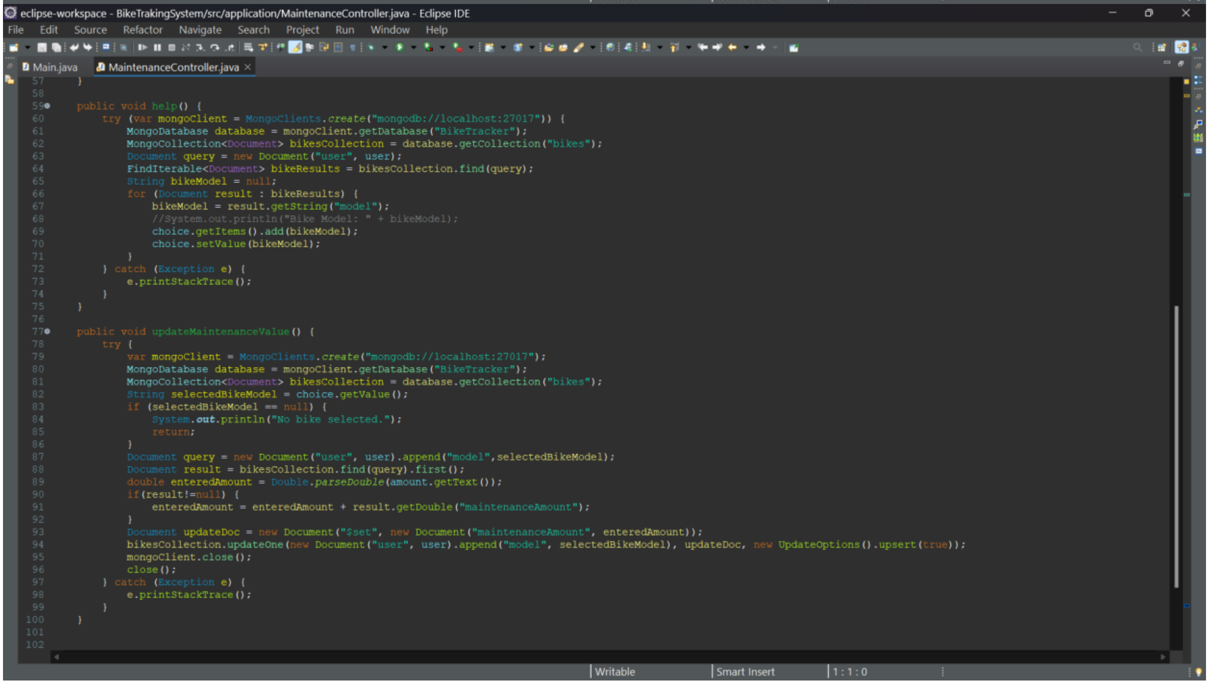
IMPLEMENTATION

The implementation phase involves translating the proposed solution into a functional Bike Tracking System. This includes coding the user interface, creating classes, establishing database connections using JDBC, and ensuring proper integration of all modules.

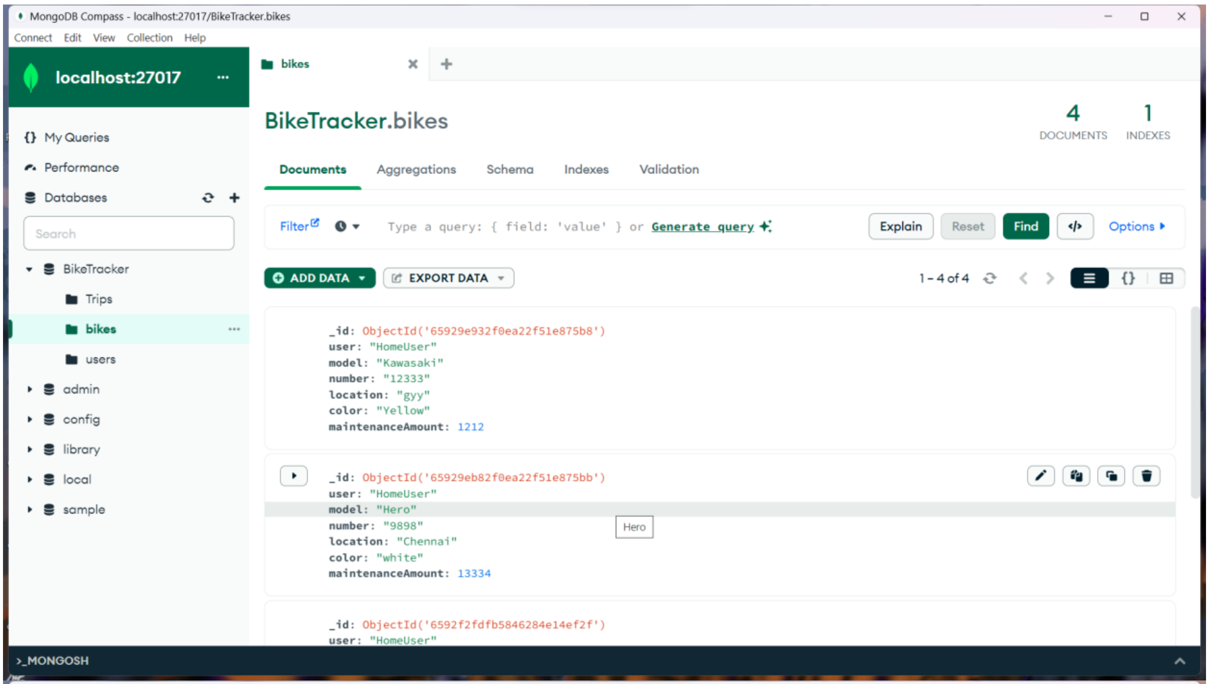


```java
    }

    private static final String DATABASE_NAME = "BikeTracker";
    private static final String COLLECTION_NAME = "users";

    public synchronized void signUp() {
        try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) {
            if(isValid()) {
                var database = mongoClient.getDatabase(DATABASE_NAME);
                var collection = database.getCollection(COLLECTION_NAME);
                var userDocument = new Document()
                        .append("username", username.getText())
                        .append("password", password.getText())
                        .append("email", email.getText())
                        .append("phone", phone.getText())
                        .append("location", location.getText());
                collection.insertOne(userDocument);

                //System.out.println("User registered successfully.");

                Stage stage = (Stage) loginButton.getScene().getWindow();
                FXMLLoader loader = new FXMLLoader(getClass().getResource("Home.fxml"));
                Parent root = loader.load();
                HomeController home = loader.getController();
                home.setUser(username.getText());
                Scene scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("Home Page");
                stage.show();
            }else {
                showInvalid();
            }
        } catch (MongoException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
```

No consoles to display at this time.


eclipse-workspace - BikeTrakingSystem/src/application/MaintenanceController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main.java
MaintenanceController.java
Writable
Smart Insert
1 : 1 : 0

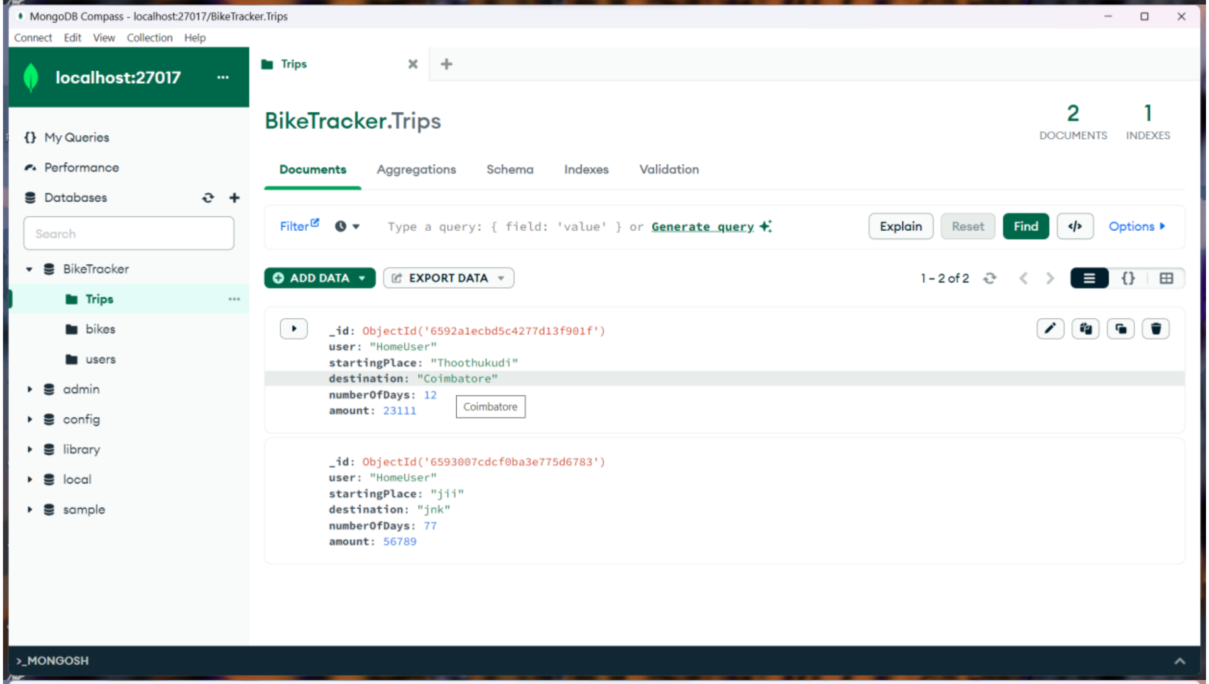
```java
    }

    public void help() {
        try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) {
            MongoDatabase database = mongoClient.getDatabase("BikeTracker");
            MongoCollection<Document> bikesCollection = database.getCollection("bikes");
            Document query = new Document("user", user);
            FindIterable<Document> bikeResults = bikesCollection.find(query);
            String bikeModel = null;
            for (Document result : bikeResults) {
                bikeModel = result.getString("model");
                //System.out.println("Bike Model: " + bikeModel);
                choice.getItems().add(bikeModel);
                choice.setValue(bikeModel);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateMaintenanceValue() {
        try {
            var mongoClient = MongoClients.create("mongodb://localhost:27017");
            MongoDatabase database = mongoClient.getDatabase("BikeTracker");
            MongoCollection<Document> bikesCollection = database.getCollection("bikes");
            String selectedBikeModel = choice.getValue();
            if (selectedBikeModel == null) {
                System.out.println("No bike selected.");
                return;
            }
            Document query = new Document("user", user).append("model",selectedBikeModel);
            Document result = bikesCollection.find(query).first();
            double enteredAmount = Double.parseDouble(amount.getText());
            if(result!=null) {
                enteredAmount = enteredAmount + result.getDouble("maintenanceAmount");
            }
            Document updateDoc = new Document("$set", new Document("maintenanceAmount", enteredAmount));
            bikesCollection.updateOne(new Document("user", user).append("model", selectedBikeModel), updateDoc, new UpdateOptions().upsert(true));
            mongoClient.close();
            close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```



```
_id: ObjectId('65929e932f0ea22f51e875b8')
user: "HomeUser"
model: "Kawasaki"
number: "12333"
location: "gyy"
color: "Yellow"
maintenanceAmount: 1212
```

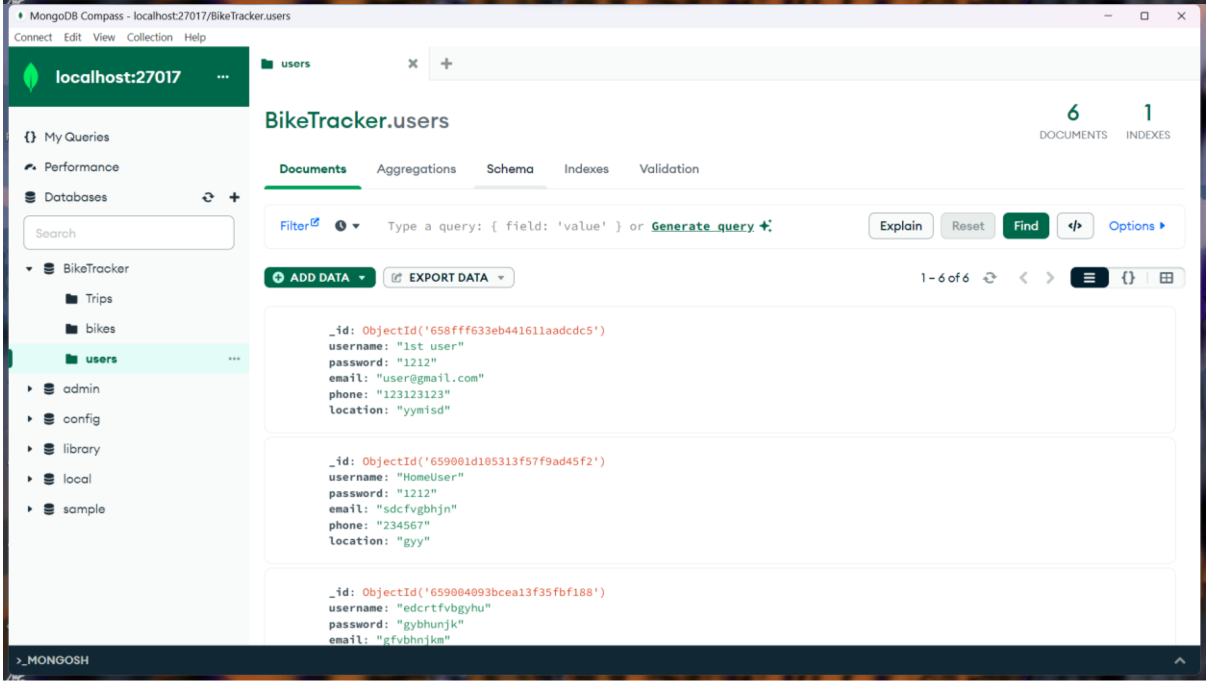
```
_id: ObjectId('65929eb82f0ea22f51e875bb')
user: "HomeUser"
model: "Hero"
number: "9898"
location: "Chennai"
color: "white"
maintenanceAmount: 13334
```

```
_id: ObjectId('6592f2fdfb5846284e14ef2f')
user: "HomeUser"
```



```
_id: ObjectId('6592a1ecbd5c4277d13f901f')
user: "HomeUser"
startingPlace: "Thoothukudi"
destination: "Coimbatore"
numberOfDays: 12
amount: 23111
```

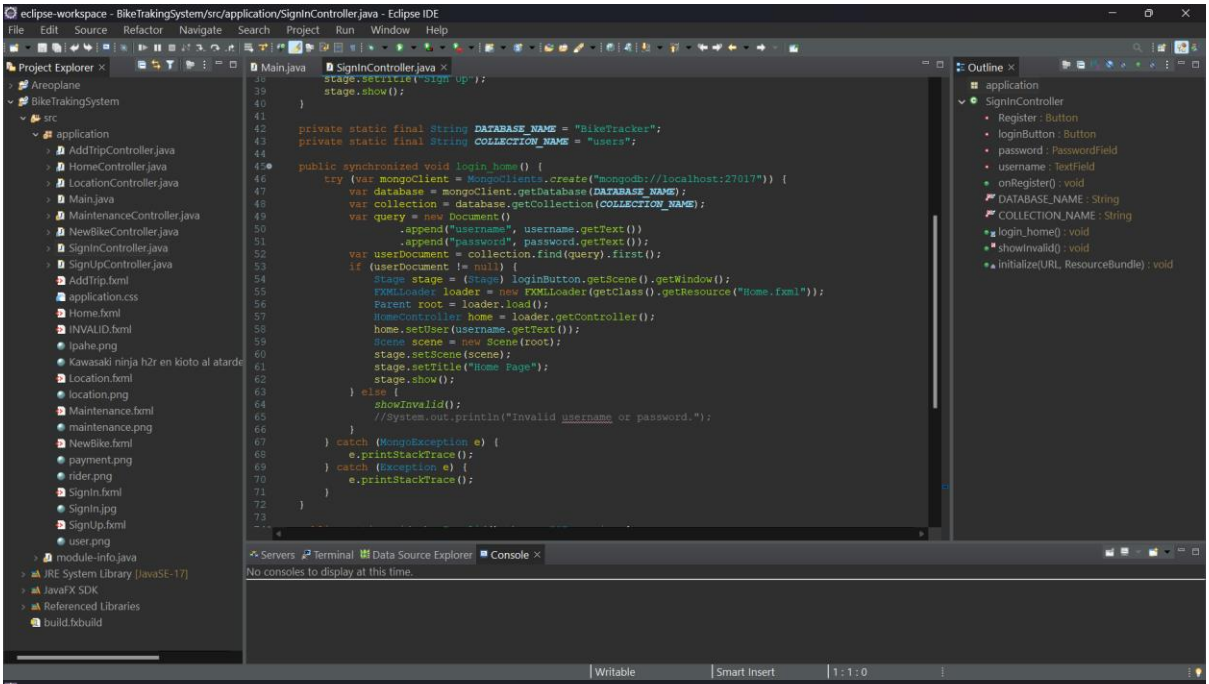
```
_id: ObjectId('6593007cdcf0ba3e775d6783')
user: "HomeUser"
startingPlace: "jii"
destination: "jnk"
numberOfDays: 77
amount: 56789
```



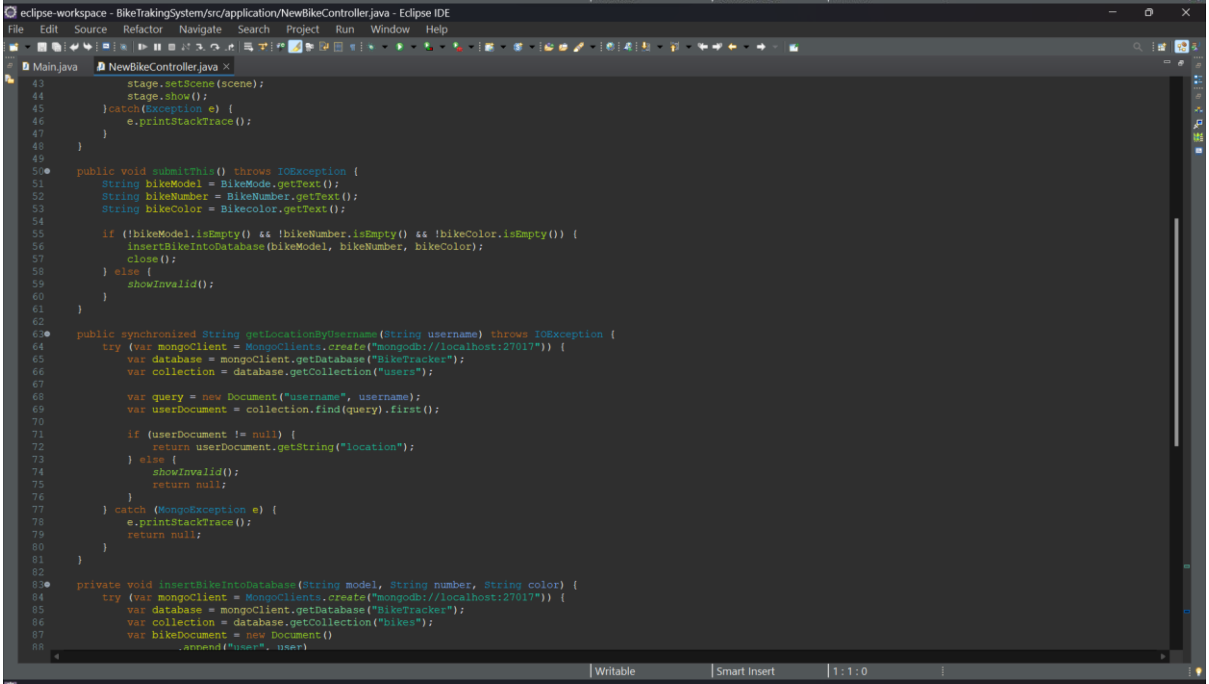
```
_id: ObjectId('658fff633eb441611aadcdc5')
username: "1st user"
password: "1212"
email: "user@gmail.com"
phone: "123123123"
location: "yymisd"
```

```
_id: ObjectId('659001d105313f57f9ad45f2')
username: "HomeUser"
password: "1212"
email: "sdcfvgbhjn"
phone: "234567"
location: "gyy"
```

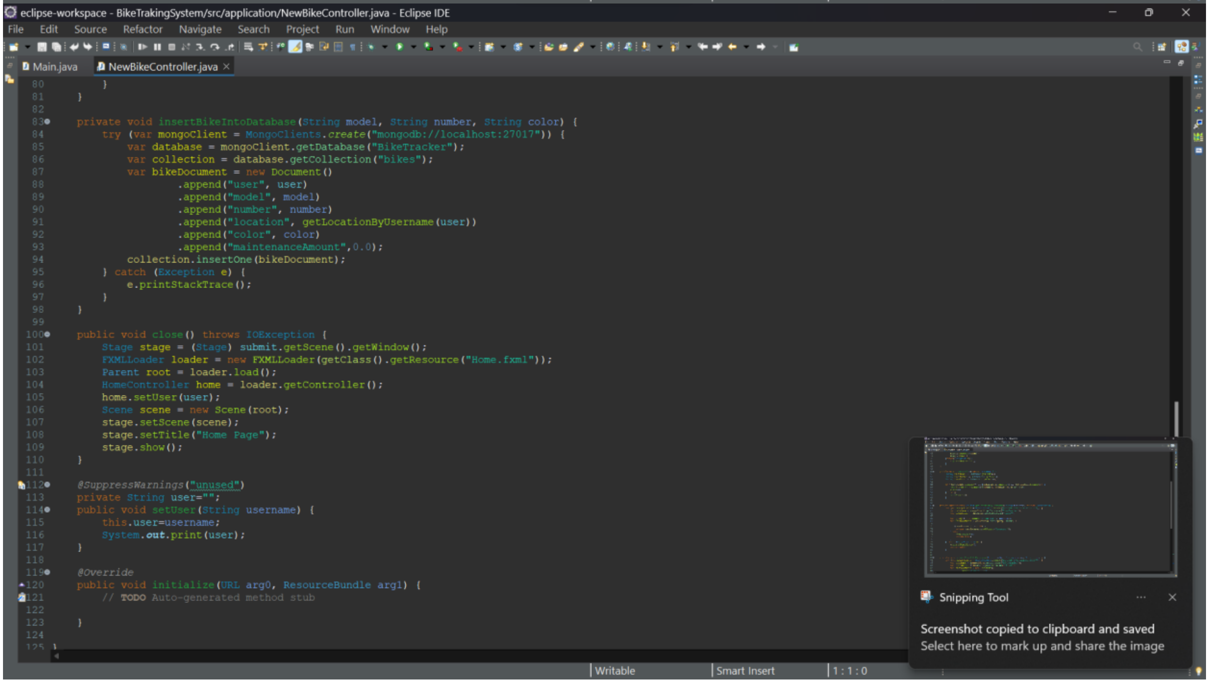
```
_id: ObjectId('659004093bcea13f35fbf188')
username: "edcrtfvbgyhu"
password: "gybhunjk"
email: "gfvbhnjkm"
```



```java
        stage.setTitle("Sign Up");
        stage.show();
    }

    private static final String DATABASE_NAME = "BikeTracker";
    private static final String COLLECTION_NAME = "users";

    public synchronized void login_home() {
        try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) {
            var database = mongoClient.getDatabase(DATABASE_NAME);
            var collection = database.getCollection(COLLECTION_NAME);
            var query = new Document()
                    .append("username", username.getText())
                    .append("password", password.getText());
            var userDocument = collection.find(query).first();
            if (userDocument != null) {
                Stage stage = (Stage) loginButton.getScene().getWindow();
                FXMLLoader loader = new FXMLLoader(getClass().getResource("Home.fxml"));
                Parent root = loader.load();
                HomeController home = loader.getController();
                home.setUser(username.getText());
                Scene scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("Home Page");
                stage.show();
            } else {
                showInvalid();
                //System.out.println("Invalid username or password.");
            }
        } catch (MongoException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

```


eclipse-workspace - BikeTrakingSystem/src/application/NewBikeController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main.java
NewBikeController.java
Writable
Smart Insert
1 : 1 : 0

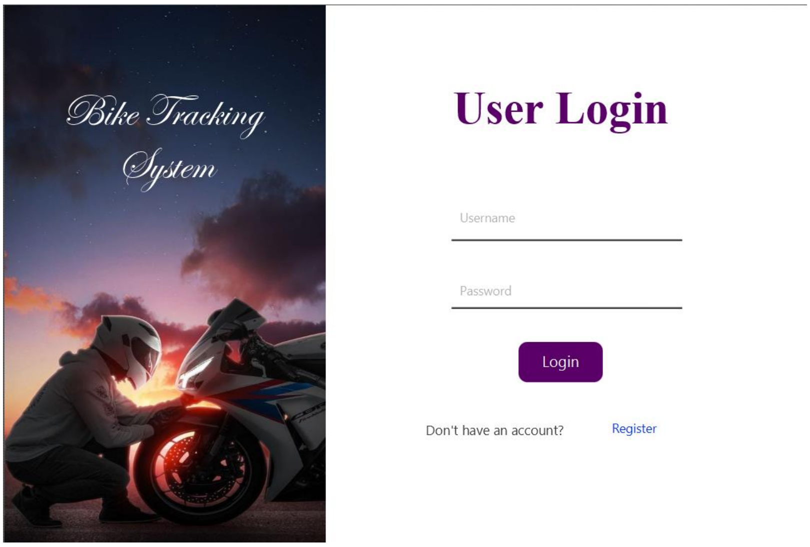
```java
                stage.setScene(scene);
                stage.show();
            }catch(Exception e) {
                e.printStackTrace();
            }
    }

    public void submitThis() throws IOException {
        String bikeModel = BikeMode.getText();
        String bikeNumber = BikeNumber.getText();
        String bikeColor = Bikecolor.getText();

        if (!bikeModel.isEmpty() && !bikeNumber.isEmpty() && !bikeColor.isEmpty()) {
            insertBikeIntoDatabase(bikeModel, bikeNumber, bikeColor);
            close();
        } else {
            showInvalid();
        }
    }

    public synchronized String getLocationByUsername(String username) throws IOException {
        try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) {
            var database = mongoClient.getDatabase("BikeTracker");
            var collection = database.getCollection("users");

            var query = new Document("username", username);
            var userDocument = collection.find(query).first();

            if (userDocument != null) {
                return userDocument.getString("location");
            } else {
                showInvalid();
                return null;
            }
        } catch (MongoException e) {
            e.printStackTrace();
            return null;
        }
    }

    private void insertBikeIntoDatabase(String model, String number, String color) {
        try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) {
            var database = mongoClient.getDatabase("BikeTracker");
            var collection = database.getCollection("bikes");
            var bikeDocument = new Document()
                    .append("user", user)
```

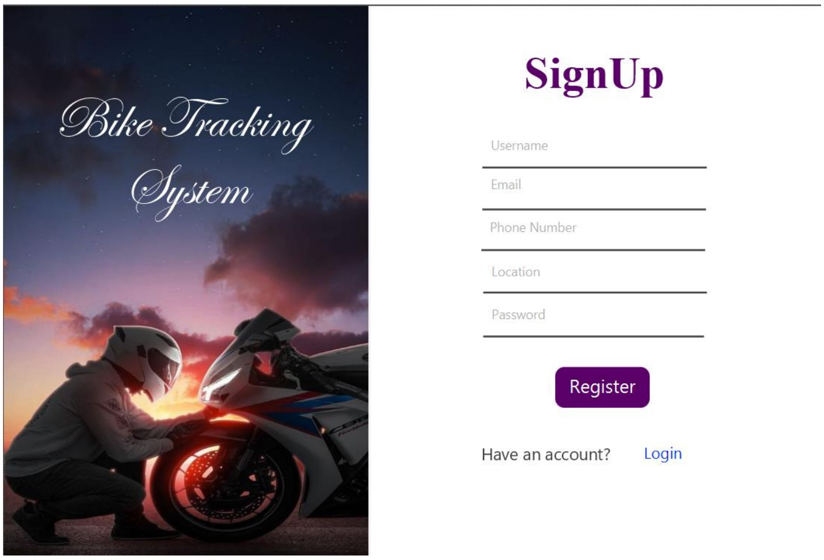

eclipse-workspace - BikeTrakingSystem/src/application/NewBikeController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main.java
NewBikeController.java
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image
Writable
Smart Insert
1 : 1 : 0

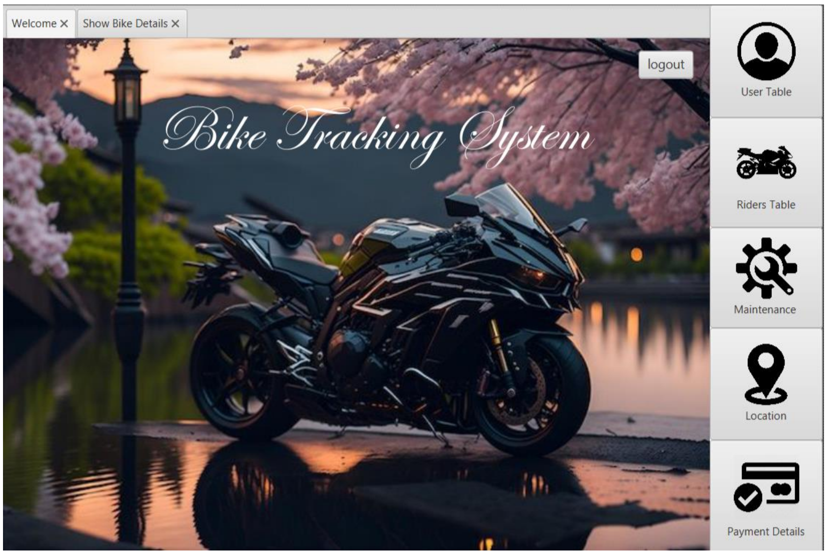
```java
        }
    }

    private void insertBikeIntoDatabase(String model, String number, String color) {
        try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) {
            var database = mongoClient.getDatabase("BikeTracker");
            var collection = database.getCollection("bikes");
            var bikeDocument = new Document()
                    .append("user", user)
                    .append("model", model)
                    .append("number", number)
                    .append("location", getLocationByUsername(user))
                    .append("color", color)
                    .append("maintenanceAmount",0.0);
            collection.insertOne(bikeDocument);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void close() throws IOException {
        Stage stage = (Stage) submit.getScene().getWindow();
        FXMLLoader loader = new FXMLLoader(getClass().getResource("Home.fxml"));
        Parent root = loader.load();
        HomeController home = loader.getController();
        home.setUser(user);
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("Home Page");
        stage.show();
    }

    @SuppressWarnings("unused")
    private String user="";
    public void setUser(String username) {
        this.user=username;
        System.out.print(user);
    }

    @Override
    public void initialize(URL arg0, ResourceBundle arg1) {
        // TODO Auto-generated method stub

    }

}
```

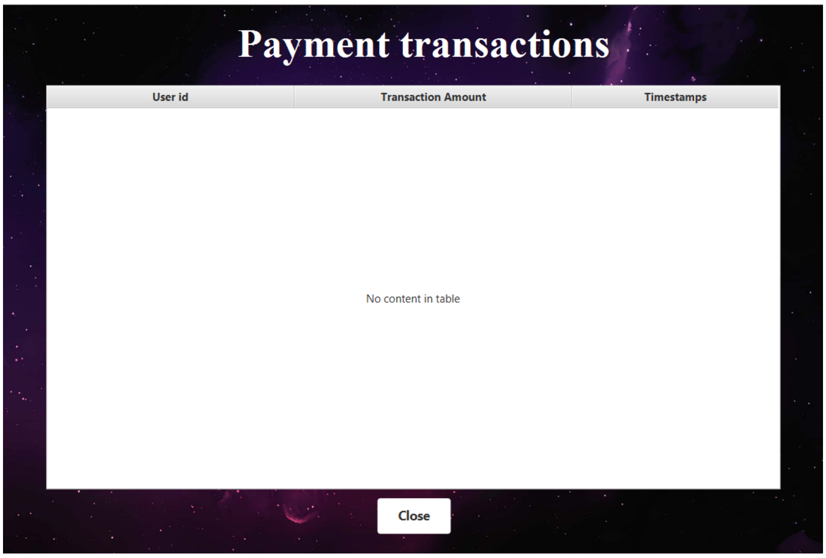
CHAPTER 7

RESULTS AND DISCUSSIONS





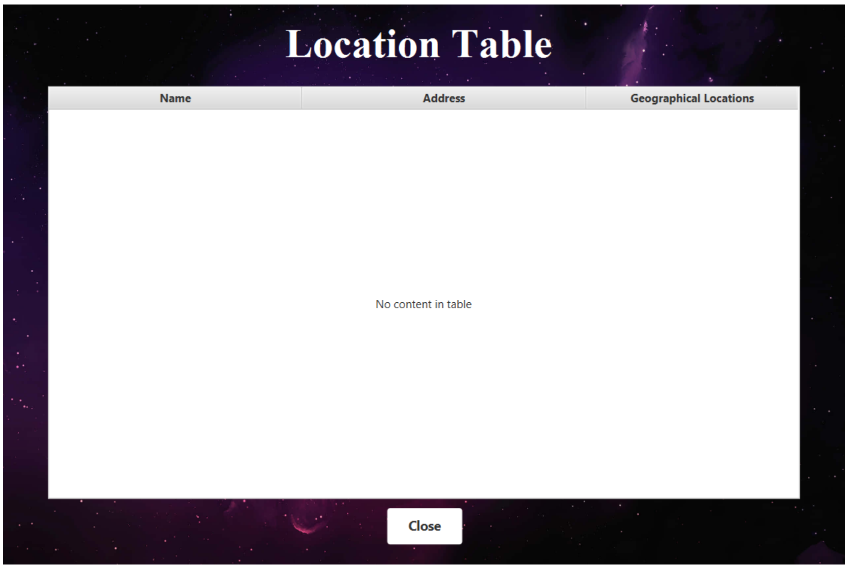




Payment transactions

User id	Transaction Amount	Timestamps

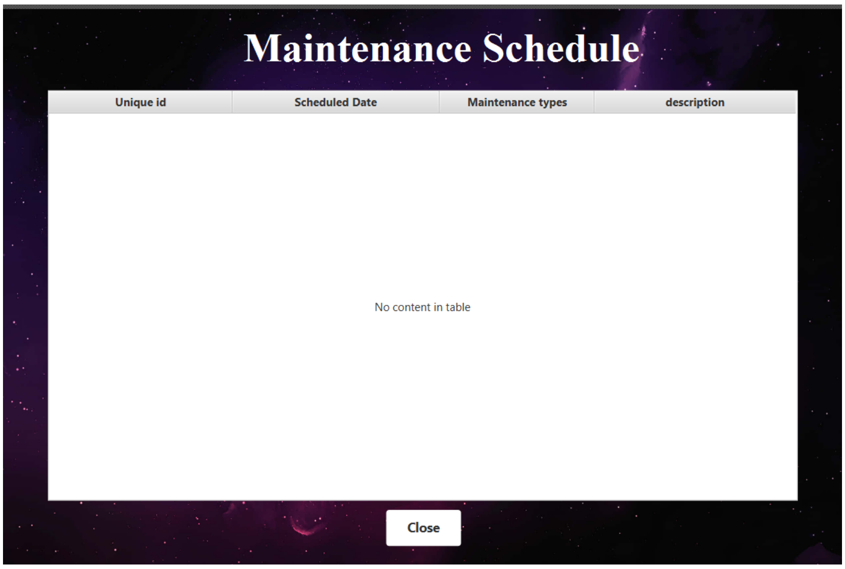
No content in table



Location Table

Name	Address	Geographical Locations

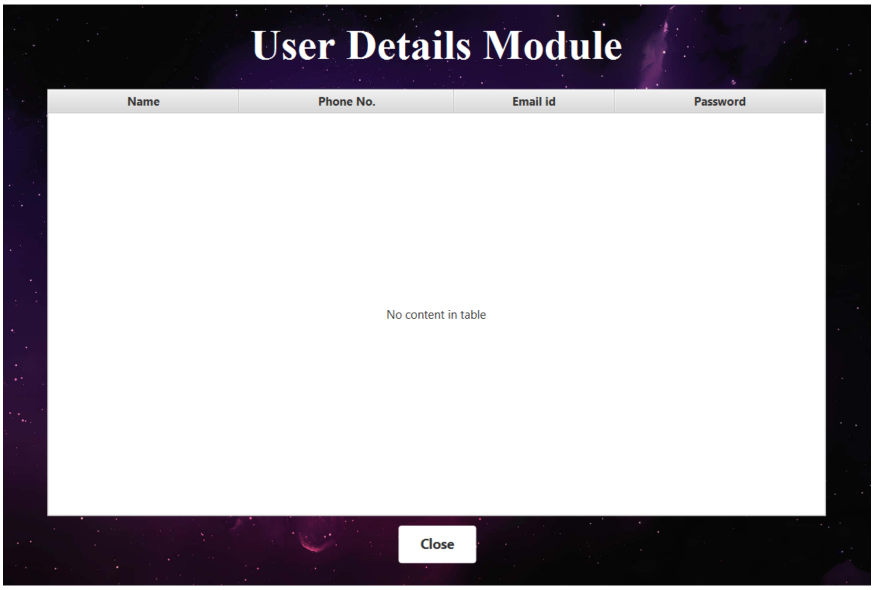
No content in table



Maintenance Schedule

Unique id	Scheduled Date	Maintenance types	description

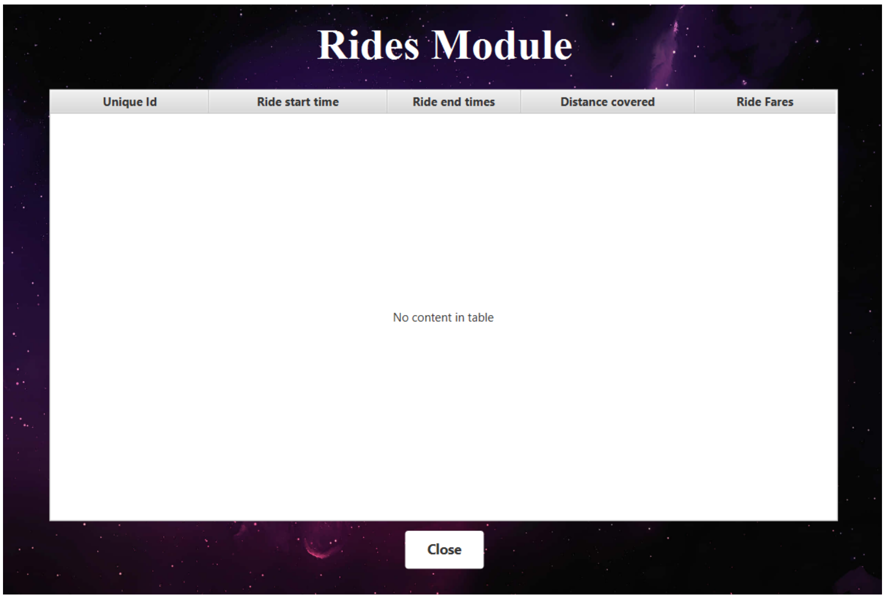
No content in table



User Details Module

Name	Phone No.	Email id	Password

No content in table



Rides Module

Unique Id	Ride start time	Ride end times	Distance covered	Ride Fares

No content in table

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

The development and implementation of the Bike Tracking System mark a significant advancement in bike fleet management. By addressing challenges related to real-time tracking, user management, ride operations, maintenance scheduling, and administrative control, the system contributes to the efficiency and reliability of bike usage and maintenance operations.

The integration of JDBC connectivity ensures a robust communication link between the application and the database, enabling seamless data retrieval, updates, and management. The system's modular design, as reflected in the class construction and database creation, lays the foundation for a scalable and maintainable solution.

Throughout the implementation phase, the system has demonstrated its ability to track bike locations, manage user data securely, record ride details, schedule maintenance, and generate insightful reports for analysis. The user interface design prioritizes user-friendliness, enhancing the overall experience for both end-users and administrators.

REFERENCES

[1] Oracle Corporation. (2022). JDBC™ Database Access. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/19/jjdbc/index.html

[2] Ambler, S., & Sadalage, P. J. (2006). Refactoring Databases: Evolutionary Database Design. Addison-Wesley.

[3] Date, C. J. (2003). An Introduction to Database Systems. Addison-Wesley.

[4] GeeksforGeeks. (2022). Introduction to JDBC. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-jdbc/

[5] Microsoft Docs. (2022). JDBC Driver for SQL Server. [Online]. Available: https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15