



---

# **A Case Study on Designing and Developing a Database with a User Interface for a Community Rental Bike System**

In Partial Fulfillment  
of the Requirement of the Subject  
**Database Design and Development Laboratory**

*Submitted by:*

**Guy, Lawrence Adrian B.**

**Surio, Rovil Jr. M.**

**Sustento, Myke Alvin E.**

**Vallarta, Troy Joaquin G.**

*Schedule:*

**THURSDAY / 10:30 A.M to 1:30 P.M / ROOM or Blackboard Ultra**

*Submitted to:*

**Engr. Jordan Vhane D. Sardalla**

Instructor

---

## **Abstract**

The case study that the group works on is about designing and developing a database for a bike rental system. This bike rental will be available for those who are part of the community. The program was created using C# Programming and the Graphical User Interface was developed using the windows form that is available on Visual Studio Code. In addition, the group used the agile methodology since this method offers flexibility and adaptability due to iterations created for the development of the project. In this way, we can optimize the program efficiently.

The objective of the study was achieved by successfully developing a program with a Graphical User Interface that is connected to the database and allows direct access for modification of data. The program was developed using concepts from Programming Logic & Design, Object-Oriented Programming, and Database and Design. The database is created using Microsoft's SQL Server. The database has the following table names: Staff; Benefactors; Beneficiaries; Bikes; and Rental Records. The program can manage, store, and modify information of each table in the database.

## **Acknowledgment**

First and foremost, praise and thank God, the Almighty, for His showers of blessings throughout the case study to complete it successfully.

The members would like to express their deep and sincere gratitude to their database design and development lab professor, Engr. Jordan Vhane Sardalla, for teaching us the required knowledge necessary to do this case study. The members would also like to express their appreciation and gratitude to their family and friends for their undying love and support on the member's endeavors in their computer engineering courses.

May this case study serve as the fruition of all the support and encouragement provided by the people in the researcher's lives.

**Table of Contents**

Abstract ..... 2

Acknowledgment ..... 3

The Problem and Its Background ..... 6

    Background of the Study ..... 6

    Objectives of the Study ..... 6

    Significance of the Study ..... 7

    Scope and Delimitation ..... 7

    Operational Definition of Terms ..... 7

Review of Related Literature and Studies..... 8

Methodology..... 9

    General Method Used ..... 9

    Procedure ..... 9

Results and Discussions..... 14

    The Developed System ..... 14

    Verification and Testing Result ..... 20

Summary of Findings, Conclusions, and Recommendations..... 23

    Summary of Findings ..... 23

    Conclusions ..... 23

    Recommendations ..... 23

Appendices ..... 25

    References ..... 26

    User Manual ..... 28

    Source Code ..... 32

    Researcher’s Profile ..... 59

## List of Figures

Figure 1 – Agile method .....	9
Figure 2 – UML Diagram.....	10
Figure 3 – ERD Diagram.....	11
Figure 4 – Storyboard.....	13
Figure 5 – Main Form.....	14
Figure 6 – Staffs Form.....	14
Figure 7 – Add Button.....	15
Figure 8 – Refreshed Button.....	15
Figure 9 – Update Button.....	16
Figure 10 – Delete Button .....	16
Figure 11 – Active Button.....	16
Figure 12 – Inactive Button.....	17
Figure 13 – Beneficiaries.....	17
Figure 14 – Benefactors.....	18
Figure 15 – Bikes.....	18
Figure 16 – Rental Records Add.....	19
Figure 17 – Rental Records Update.....	
Figure 18 – Testing and Verification GetTime().....	20
Figure 19 – Testing and Verification GetTime() Solution.....	20
Figure 20 – Testing and Verification Constraint.....	20
Figure 21 – Testing and Verification Constraint Solution.....	21
Figure 22 – Testing and Verification Constraint Solution Checking.....	22
Figure 23 – Testing and Verification for BIT Data Type and User-Generated Column.....	22

## Chapter I

### The Problem and Its Background

The case study is about creating a bike rental system or program that is connected to the database. The database will hold all the data such as records of staff, benefactors, beneficiaries, bikes, and rental records. The program with a graphical user interface will be the primary mode of accessing the records from the database.

#### A. Background of the Study

The COVID-19 case in the Philippines was first confirmed by the Department of Health (DOH) on the 30th of January 2020 [1], and the first COVID-19 death was recorded two days after the first confirmation or on the 2nd of February 2020 [2]. The transmission of the said disease was announced on the 7th of March and the current confirmed cases in the country by the 27th of May 2021 were reported at 1,193,976 total cases with 46,037 active cases, 1,127,770 recoveries, and 20,169 deaths [3]. Moreover, the World Health Organization (WHO) declared COVID-19 as a global pandemic on the 11th of March 2020 [4].

The uncontrollable acceleration of COVID-19 cases resulted in lockdown measures. On the 12th of March 2020, President Rodrigo Duterte announced a travel restriction that will last until the 14th of April 2020 in Metro Manila after the health authorities raised the alert level of COVID-19 to the highest concerning the declaration of the World Health Organization last 11th of March [5]. It was not until the 16th of March 2020 when the Luzon was placed under ‘enhanced’ community quarantine in which mass public transportation services were suspended [6]. According to the National Economic and Development Authority [7], “The COVID-19 pandemic has hurt the economy, especially the transport sector.” This is due to the fact that the community quarantines and physical distance regulations must be strictly followed to protect lives which resulted in reduced transport supply and public transport shortages. With that being said, the National Economic and Development Authority promotes cycling and building protected bike lanes because they believe that this can improve public transport and help people safely get to work during the COVID-19 pandemic. Therefore, a lot of people nowadays consider bicycles as a main mode of transportation.

#### B. Objectives of the Study

*General Objective:* To design and develop a database that is connected to a program with the graphical user interface.

*Specific Objective:*

1. To design and create a community rental bike database with tables for staff, beneficiaries, benefactors, bikes, and rental records.
2. To create a graphical user interface for the database using C# Programming.

3. To create a bike rental system that manages and stores information about the staff, beneficiaries, benefactors, bikes, and rental records.

### C. Significance of the Study

This study aims to create a rental bike program that would be controlled by staff. The program should let a customer manage the records through the user interface that is connected to the database. Furthermore, the study could be of importance to the following.

**Staff.** The program created from this study should help staff have more efficient storage and management of rental records.

**Beneficiaries.** The program created for this study should help beneficiaries have a more efficient arrangement in borrowing bikes with bike rental staff.

**Benefactors.** The program created for this study should help the benefactors have a faster and smoother arrangement in donating bikes with bike rental staff.

**Future Researchers.** The ideas presented may be used by future researchers as a reference. This may help them create their own programs or improve the program created by the researchers

### D. Scope and Delimitation

The rental bike program allows insertion of data, updating, and deleting a specific row. There are two types of deleting that can be done in the program and these are the soft delete or by updating the value of the active state or the validity column to False. And the second one is the hard delete which executes a DELETE statement from SQL. Furthermore, the rental bike program is created using a local database hence the database cannot be updated or accessed online. In addition, the program doesn't deal with transactions involving money as it is a free community rental bike.

### E. Operational Definition of Terms

***Bike rental system*** - The application created was about managing the records of the rental, staff, benefactor, and beneficiary.

***C#*** - The programming language that was used in the creation of the bike rental program.

***Database*** - contains an organized collection of data that can be accessed using a computer system.

***SQL*** - A language being used for accessing and manipulating the database.

***Staff*** - The people that handle the bike rental system.

***Beneficiaries*** - The people that borrow bikes from the community rental bike.

***Benefactors*** - The people that donates bikes to the community rental bike.

## **Chapter II**

### **Review of Related Literature and Studies**

This chapter is designed to enumerate and identify some research related to the current study being developed.

The database design and development paradigm introduce the process of creating a data model of a database [8]. It is a collection of processes that implements design, development, implementation, and maintenance of data management systems [9]. The objectives of designing a database are to produce logical and actual models of the proposed database system [9]. A good database design will help a database designer reduce data storage, increase performance, and data indexing which will help a designer revise and redesign the database if necessary, during development [10].

#### **SQL Commands & Queries**

Structured Query Language or SQL for short is a programming language specifically designed for managing data stored in databases [11]. These commands let a designer keep data accurate and secure [11].

One of the main features of using SSMS is in its queries. It produces many features that programmers can operate at their disposal [12]. For example, using a query to create a database is to provide the programmer with an option to create a faster time. Using a drop query [13] can easily remove a database. The last most common query is the create table [14], wherein the programmer can create a table inside the database and create the attributes needed in the program. The given queries are one of the commonly used queries in this research.

Database design and development used are what the programmer called the “keys” used to create a relationship between different tables. A primary key is one of the most powerful and essential keys in a database [15]. This key is used in every table on the database because a table needs one and only one primary key that provides the focal point data on the table. Foreign keys are used to create a key that correlates to the primary key of the other table [16]. The foreign key is used when the programmer wants to have the same data on the table, and the data should be dependent on the primary key of the other table. The keys mentioned above are essential in creating a database because it provides easy data collection and security of a database.



## Chapter III

### Methodology

The methodology and procedures used in designing and developing the project are chosen to consider the factors such as adaptability and flexibility. This chapter will primarily discuss the reason why the group decided to make use of the chosen methodology and procedures.

#### A. General Method Used

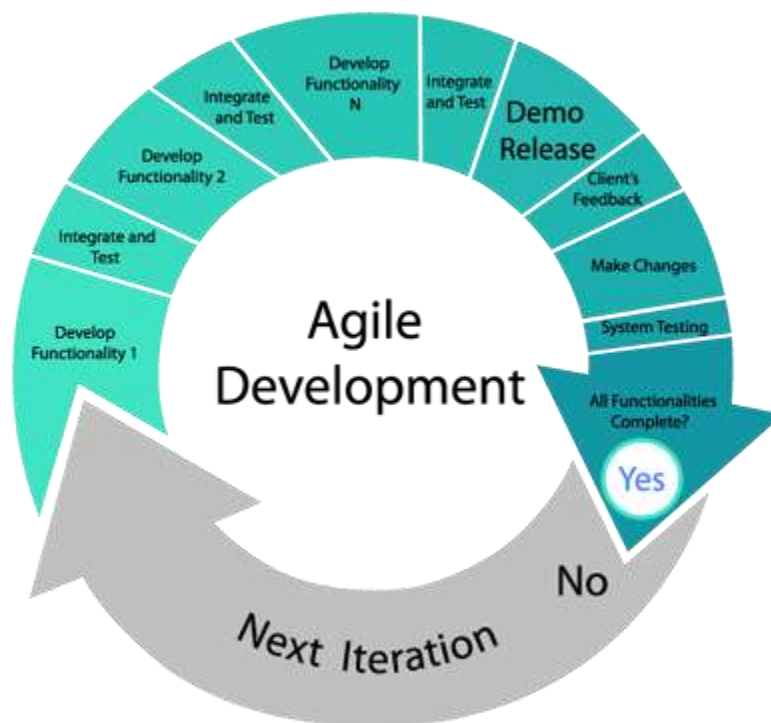


Figure 1 Agile method

The agile methodology was used since this method offers flexibility and adaptability. The group utilized making iterations for the development of the project to accommodate change and this makes the project more optimized since all files were being checked again per iterations. Overall, the program created using MS SSMS and Visual Studio goes to multiple testing until the desired output is achieved.

#### B. Procedure

##### 1. Design Phase

The researchers planned before creating the program by creating diagrams that may guide them on creating the program. This is to provide a direction, foundation, and blueprint for the flow of production of the program.

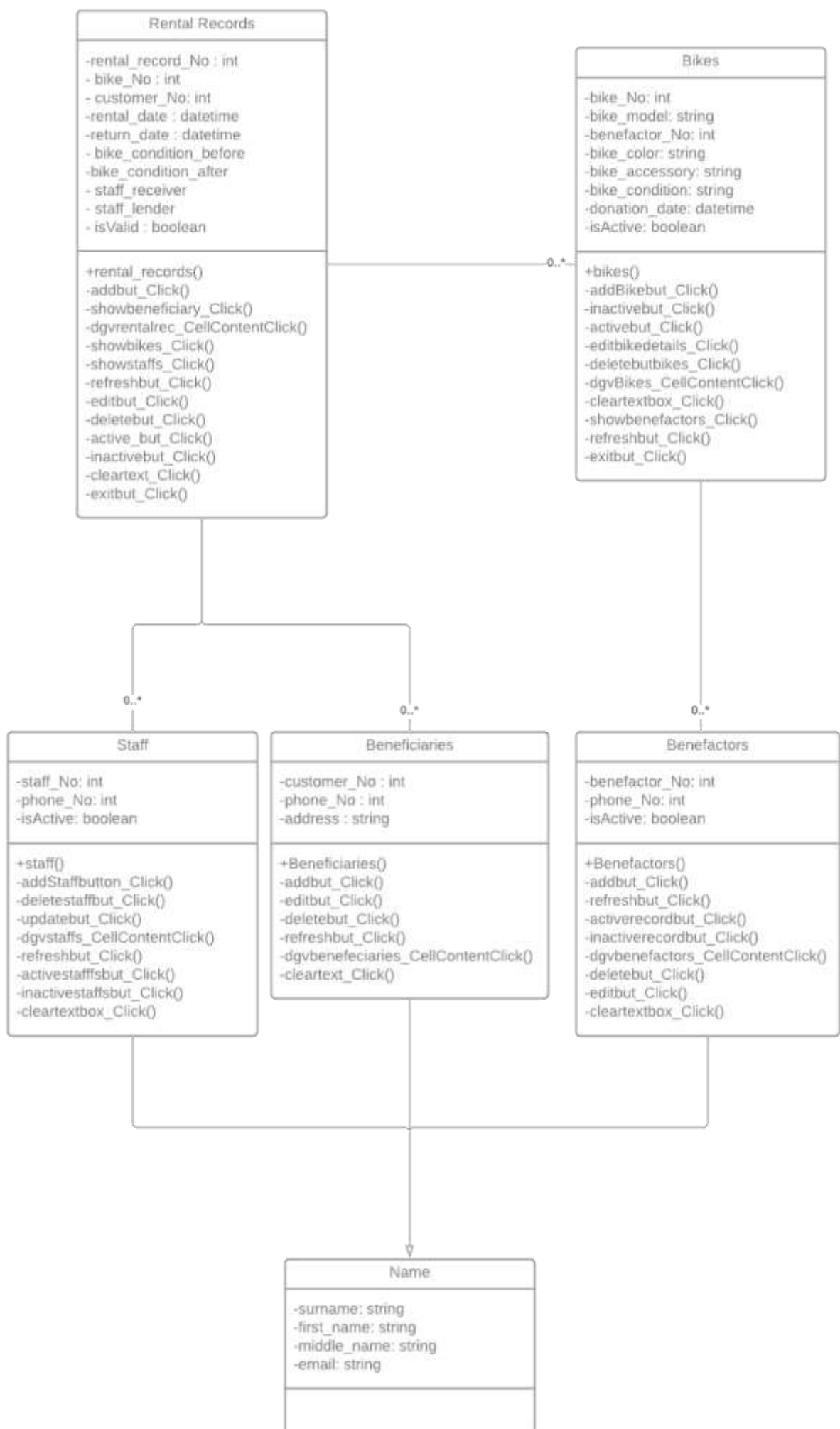


Figure 2 UML Diagram

The UML created can be seen in figure 2. First, the researchers created the UML diagram to give the team unity in implementing the program. The UML diagram shows the flow of the database system that will be implemented in a clear and concise manner. It also includes the methods that were used to clarify the function of each class.

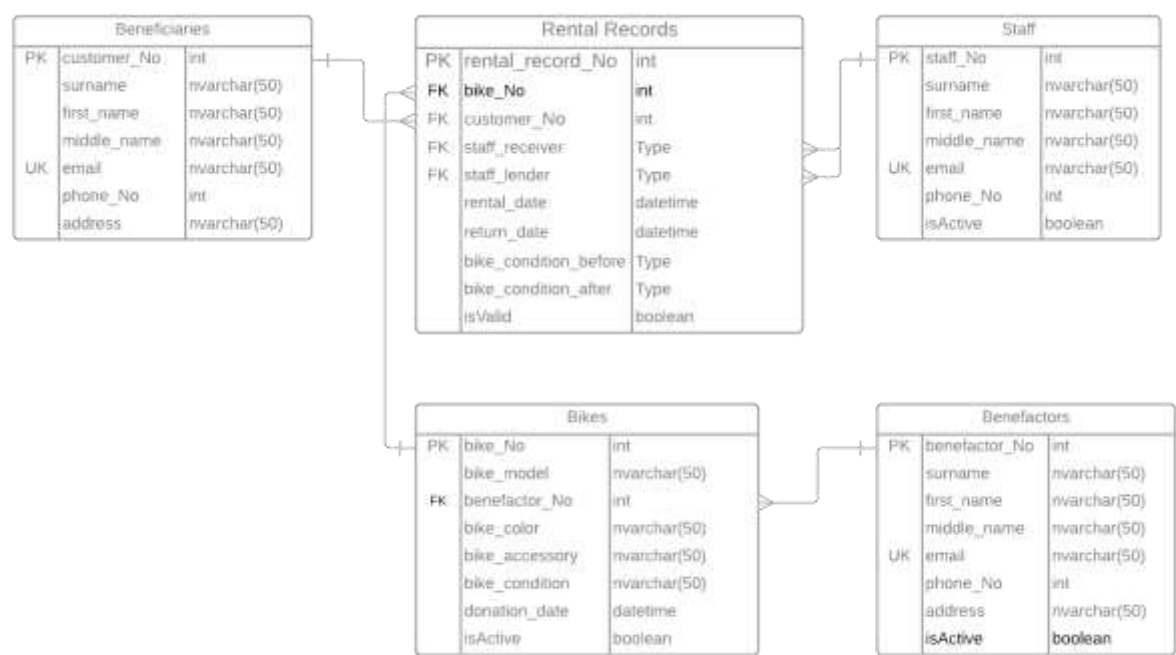
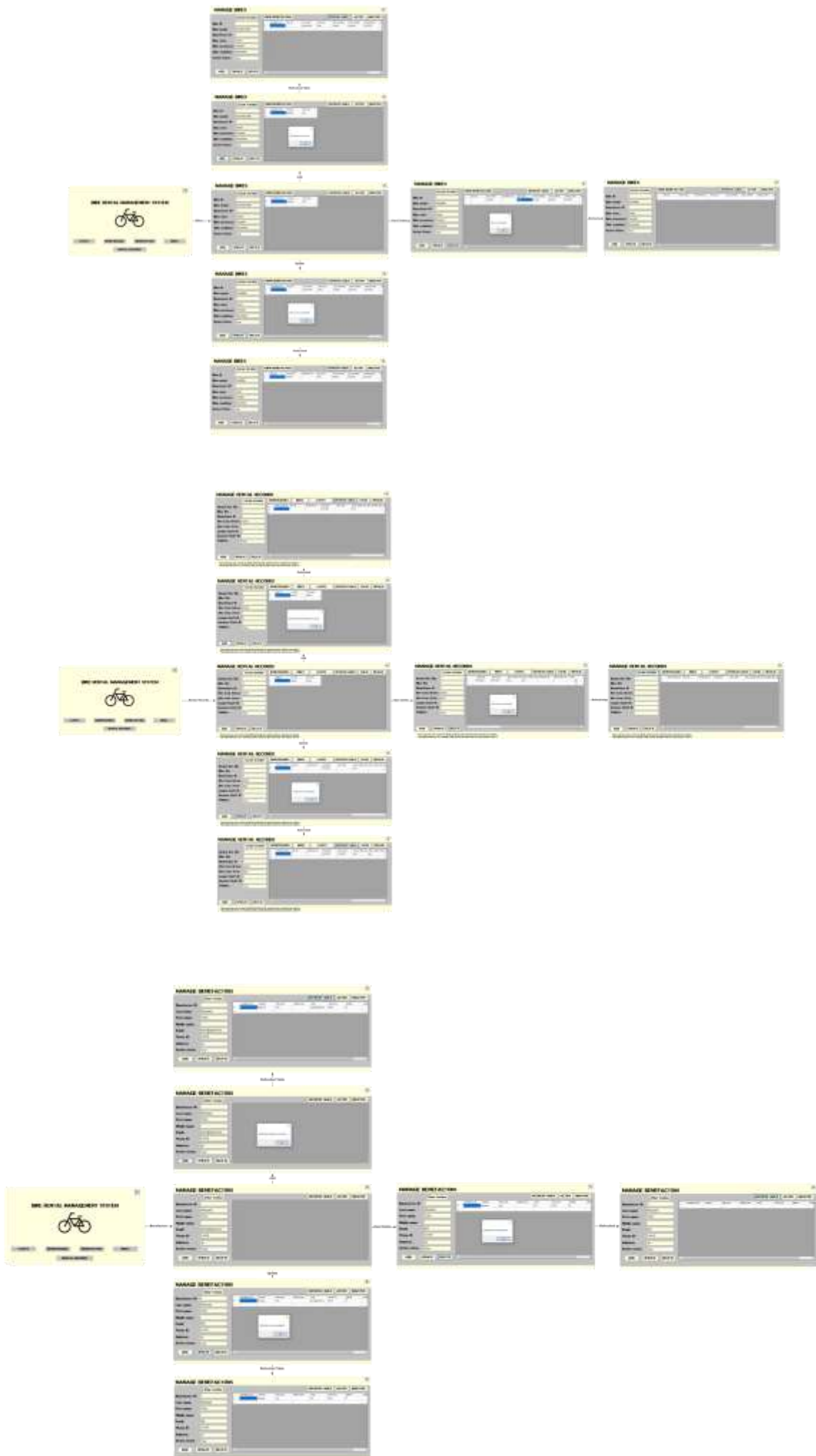


Figure 3 ERD Diagram

Figure 3 shows the ERD diagram created by the researchers. The ERD diagram was used to properly show the relationship between the different entities that will be used in the bike rental system. It also shows the different attributes of different entities as well as their keys and data types. This will set the guidelines used for the MS SSMS system.



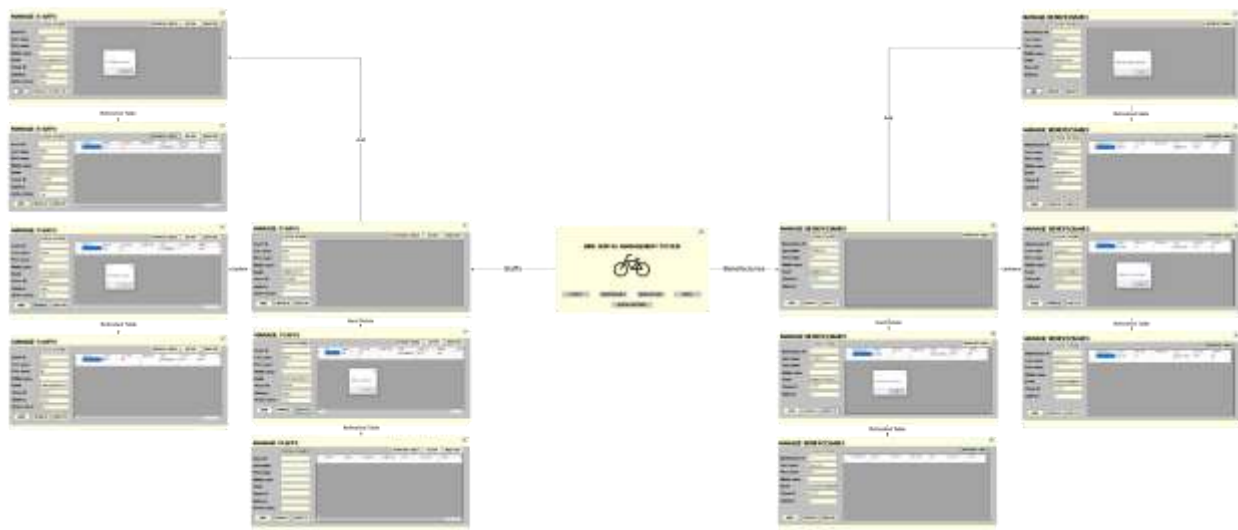


Figure 4 Storyboard

Figure 4 shows the storyboard created by the researchers. The storyboard helped visualize the actual user interface that will be used in the bike rental management system. These will guide the researchers on how they will implement the system in Visual Studio.

## 2. Implementation Phase

After the design phase of the program, the researchers implemented the bike rental system using MS SSMS and Visual Studio. The MS SSMS was used to create the database of the different entities. The Visual Studio was used to program the database using the C# programming language to have different functionalities.

### Lucidchart Links:

[Staffs and Bikes: Lucidchart](#)

[benefactors, bikes, rent\\_records: Lucidchart](#)

[UML Diagram: Lucidchart](#)

[Lemon Group Bike Rental Shop: Lucidchart](#)

Chapter IV

Results and Discussions

This chapter will discuss the developed output and the results of the tests discussed in the previous chapter, describing how the code was created and the methods that were used in order to make the application with regard to the objective of the case study. The functions were shown as well as its output to ensure the reliability and consistency of its output.

A. The Developed System

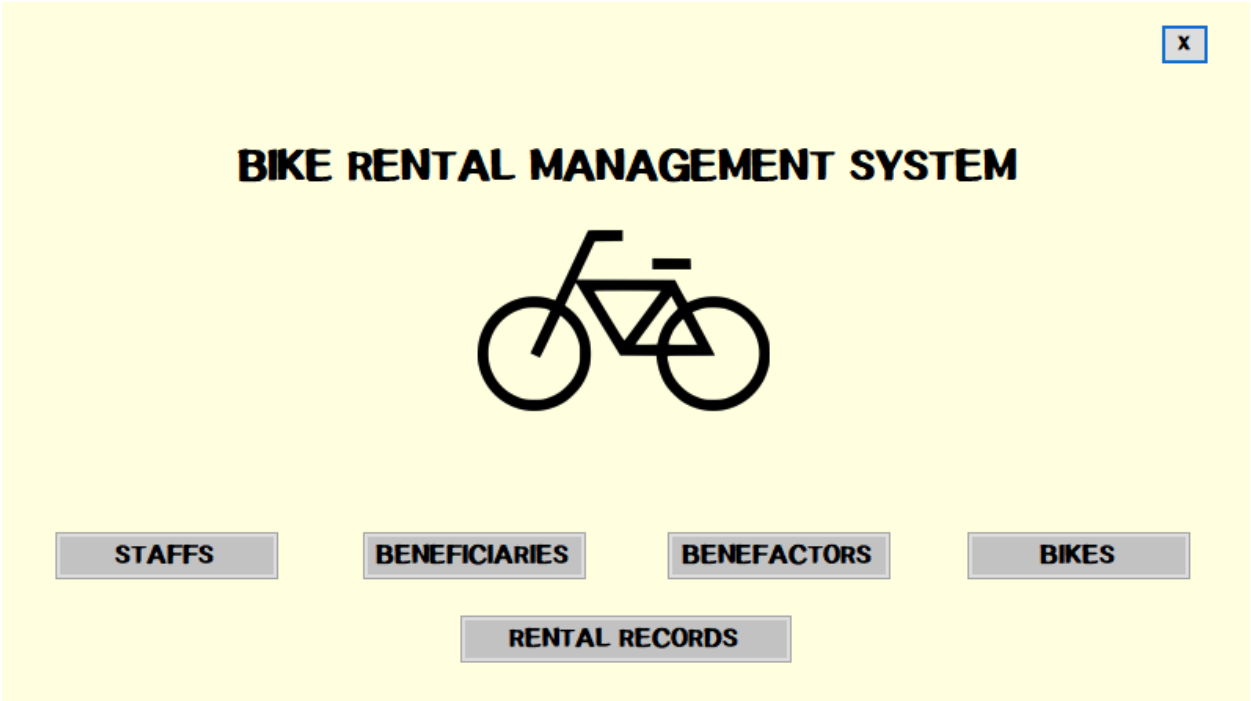


Figure 5 Main Form

Figure 5 shows the main form of the created bike rental system that contains all the buttons for managing specific tables from the database.

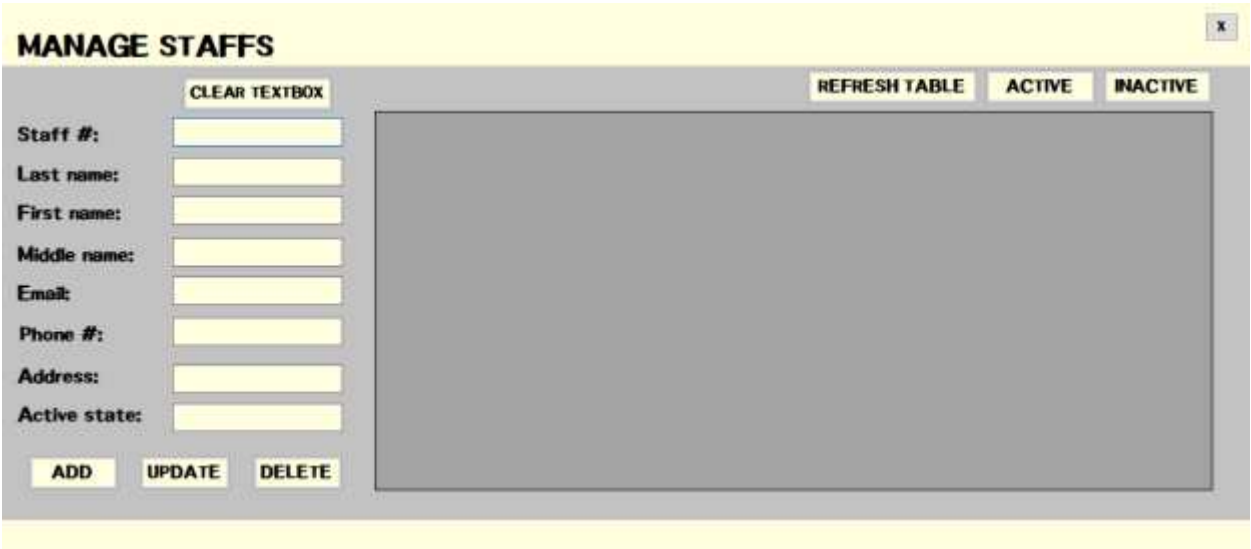


Figure 6 Staffs Form

Figure 6 shows the user interface for adding, updating, and deleting a staff record from the bike rental database. This also has a button for refreshing the table and show only the active or inactive records.



Figure 7 Add Button

Figure 7 shows the process on how to insert data from the user interface directly to the database of the bike rental system. It can be observed that the staff # is empty in the insertion because it was auto-generated or in auto-seeding in the database. The staff # text box, therefore, is used for updating specific data because it is referencing the staff # of a specific row of data.

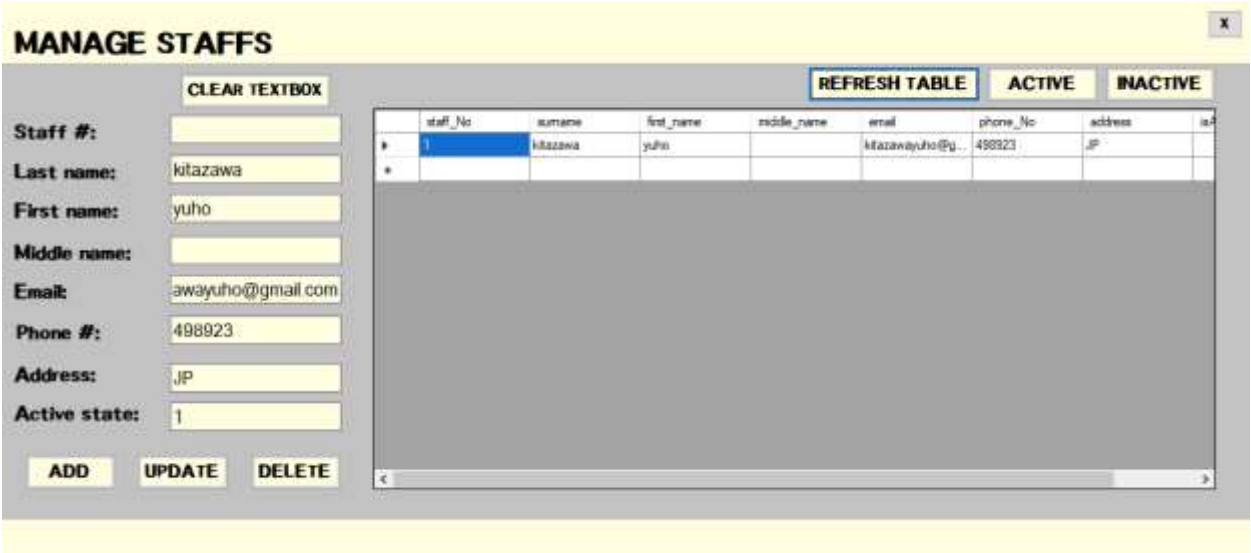


Figure 8 Refreshed Button

Figure 8 shows the result of clicking the refresh button on the user interface. This selects everything from the staff table in the bike rental database and displays it on the data grid view as shown in Figure 8.



Figure 9 Update Button

Figure 9 shows the functionality of the update button. It can be observed that the staff # of the row of data must be specified in order to update it.

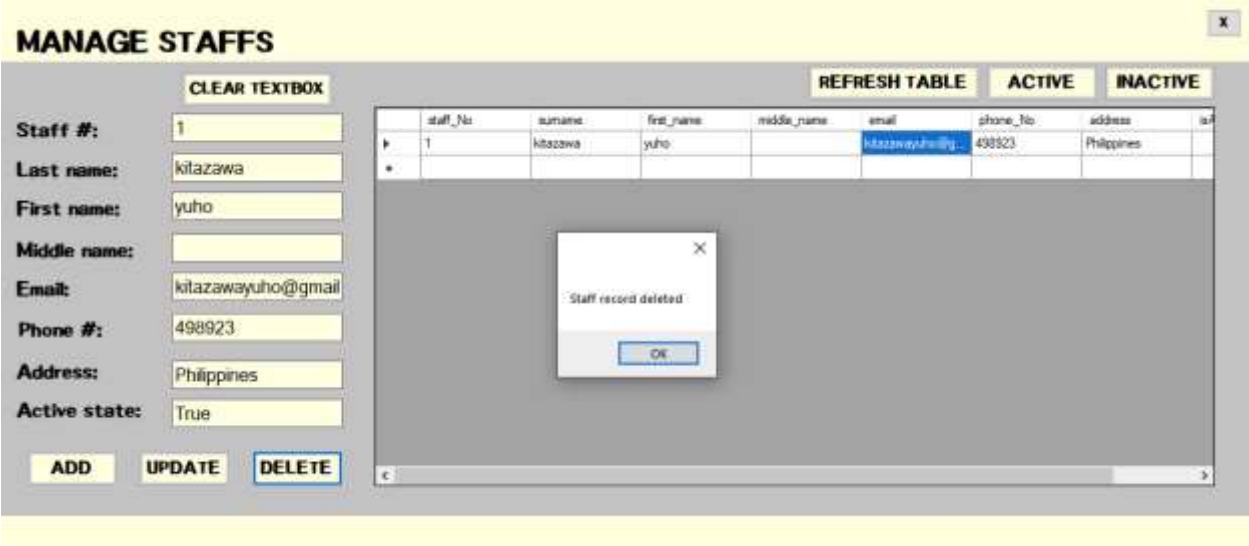


Figure 10 Delete button

Figure 10 shows the deletion of the row specified through its staff # on the staff # text box. This button is the hard delete while updating the active state of the specific row.

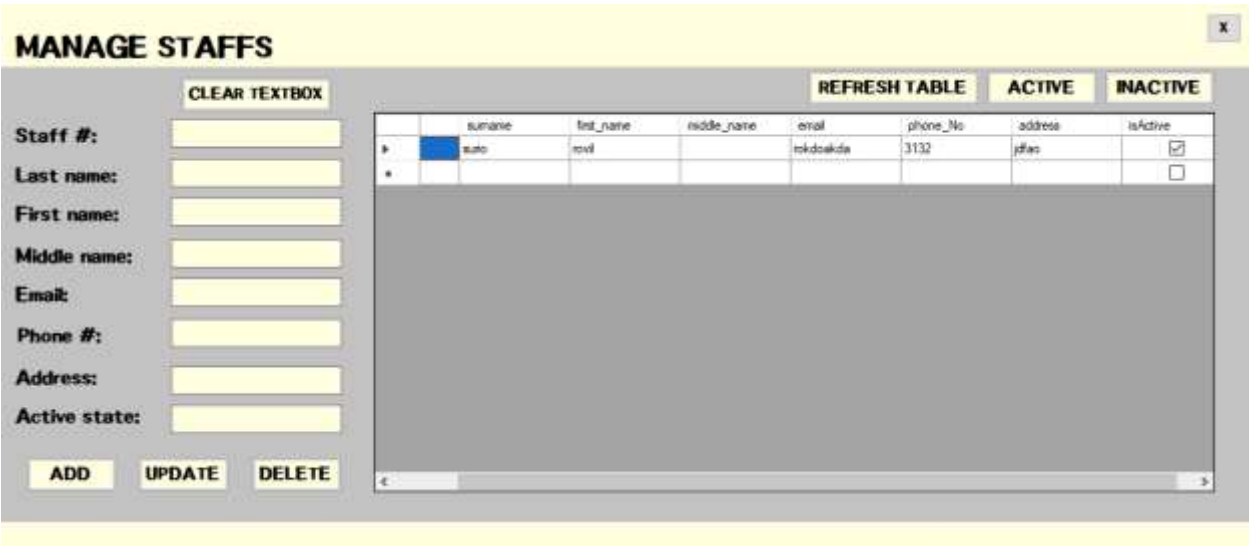


Figure 11 Active button



Figure 11 shows the result of clicking the active button on the user interface. It can be observed that it displays the records with the active state on the data grid view.



Figure 12 Inactive button

Figure 12 shows the result of clicking the inactive button from the user interface. This shows the row of data with the inactive state on the data grid view.



Figure 13 Beneficiaries

Figure 13 shows the user interface for adding, updating, and deleting a beneficiary record from the bike rental database. Like the staff table, this also has a refresh table button, add button, update button, delete button, and clear textbox button. These buttons have the functionalities as the staff table.

MANAGE BENEFACTORS

Clear Textbox

Benefactor #:

Last name: hatdog

First name: cheesedog

Middle name: palaman

Email: kwekkwek@gmail.co

Phone #: 0923423049

Address: ermita hehe

Active state: true

ADD

UPDATE

DELETE

REFRESH TABLE

ACTIVE

INACTIVE

	benefactor_no	surname	first_name	middle_name	email	phone_no	address	isAct
➤	1	Klaszawa	Yuh		yuh@gmail	4151	JP	
	2	hatdog	cheesedog	palaman	kwekkwek@gmail	923423049	ermita hehe	
*								

Figure 14 Benefactors

Figure 14 shows the user interface for adding, updating, and deleting a benefactors record from the bike rental database. Like the staff table, this also has a refresh table button, add button, update button, delete button, and clear textbox button. These buttons have the functionalities as the staff table.

MANAGE BIKES

CLEAR TEXTBOX

Bike #:

Bike model: carbonlodi

Benefactor #: 2

Bike color: blue

Bike accessory: keychain

Bike condition: broken hearted

Active State: 1

ADD

UPDATE

DELETE

SHOW BENEFACTORS

REFRESH TABLE

ACTIVE

INACTIVE

	benefactor_no	surname	first_name
➤	1	Klaszawa	Yuh
	2	hatdog	cheesedog
*			

Figure 15 Bikes

Figure 15 shows the user interface for adding, updating, and deleting a bike record from the bike rental database. Like the staff table, this also has a refresh table button, add button, update button, delete button, and clear textbox button. These buttons have the functionalities as the staff table. However, it can be seen that there is another button called the show benefactors button which shows the user the list of all the benefactors which can guide them to know what to input in the benefactor # textbox.

MANAGE RENTAL RECORDS

CLEAR TEXTBOX

Rental Rec. No:

Bike No:

1

Beneficiary #:

1

Bike Cond. Before:

good

Bike Cond. After:

Lender Staff #:

2

Receiver Staff #:

Validity:

1

ADD

UPDATE

DELETE

BENEFICIARIES

BIKES

STAFFS

REFRESH TABLE

VALID

INVALID

	rental_record_No	bike_No	beneficiary_No	rental_date	return_date	bike_condition_bef	bike_condition_aft	sta
+								

Bike Rental Record Added Successfully

OK

\*upon inserting, only provide the bike#, beneficiary#, condition before, lender#, and validity

\*the update button is for returning a bike, provide the bike cond. after and receiver staff #

Figure 16 Rental Records Add

Figure 16 shows the user interface for adding a record on the rental records table from the database. The bike No. should be existing from the bikes table same with the beneficiary No. because these attributes are foreign keys on this rental records table, so it is referencing from the other table. Also, the bike condition after and receiver staff No. is empty during the insertion of the rental record since this specific transaction is mainly for lending a bike.

MANAGE RENTAL RECORDS

CLEAR TEXTBOX

Rental Rec. No:

1

Bike No:

1

Beneficiary #:

1

Bike Cond. Before:

good

Bike Cond. After:

very bad

Lender Staff #:

2

Receiver Staff #:

2

Validity:

True

ADD

UPDATE

DELETE

BENEFICIARIES

BIKES

STAFFS

REFRESH TABLE

VALID

INVALID

	rental_record_No	bike_No	beneficiary_No	rental_date	return_date	bike_condition_bef	bike_condition_aft	sta
+	1	1	1	27/5/2021	27/5/2021	good	very bad	2

Rental record was updated.

OK

\*upon inserting, only provide the bike#, beneficiary#, condition before, lender#, and validity

\*the update button is for returning a bike, provide the bike cond. after and receiver staff #

Figure 17 Rental Records Update

The bike condition after and receiver staff No. can be updated using the Update button when someone will return a borrowed bike. Take note that the rental record number of a specific row must be specified on the rental rec. No. text box to specify what data you will edit.

19

B. Verification and Testing Result

	bike_No	bike_model	benefactor_No	bike_color	bike_accessory	bike_condition	donation_date	isActive
1	13	3123	1	weqwe	qweqwe	ege	2021-05-25 20:12:46.363	1
2	14	qweqwe	1	eqweq	ewqe	qweq	2021-05-25 20:12:46.363	1
3	15	32131	1	eqwe	eqwe	qeqwe	2021-05-25 20:12:46.363	1
4	17	32131	1	eqwe	eqwe	qeqwe	2021-05-25 20:12:46.363	1
5	18	33333	1	eqweqwe	eqweq	1qweqwe	2021-05-25 20:12:46.363	1
6	19	mountain	1	green	ewan	ewan	2021-05-25 20:12:46.363	1
7	20	ROAD	1	ASDASD	EQWEQ	SDFASD	2021-05-25 20:12:46.363	0

Figure 18 Testing and Verification GetTime()

Upon testing the system, the group encountered an error in using the GetDate() function for the retrieval of the current database system time in SQL Server. What happened is that when selecting the data from a table that uses getdate() function for the date and time specifically for the rental records table and bikes table, it rewrites the date and time value of the table by the current date and time. To fix this problem, the group implements the getdate() function on the user interface side instead rather than the database side.

```
else
{
    var Date = DateTime.Now.ToString("MM/dd/yyyy");
    string query = "INSERT INTO Bikes (bike_model, benefactor_No,bike_color,bike_accessory,bike_condition,isActive,donation_date) VALUES ('" + bike_model.Text + "','" +
    benefactor_No.Text + "','" + bike_color.Text + "','" + bike_accessory.Text + "','" + bike_condition.Text + "','" + isActiveState.Text + "','" + Date + "')";
    SqlCommand cmd = new SqlCommand(query, Con);
    cmd.ExecuteNonQuery();
    MessageBox.Show("Bike added successfully");
    Con.Close();
}
```

Figure 19 Testing and Verification GetTime() Solution

Figure 19 shows the solution for the problem in getting the date and time of the current system time. The DateTime() function of the C# programming language was used in order to get the current system time when someone adds bike detail on the bikes table or when someone's adding or updating a record from the rental records table.

	benefactor_No	surname	first_name	middle_name	email	phone_No	address	isActive
	1	eqweqwe	fwqw	321	eqwe12e	41241	effqqwr	True
...	NULL		rovil	m	rsurio@gmail.c	3213	djashndkajd	True
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 20 Testing and Verification Constraint

The other problem encountered by the group is that even a specific column from one table does not allow nulls when the user does not put anything on the textbox in the GUI or just inputted a space, the database still accepts it because it doesn't see that column as NULL hence it allows it. To solve the problem, the group added constraints per table and per column that check the length of the value and make sure that it is greater than or equal to 1.

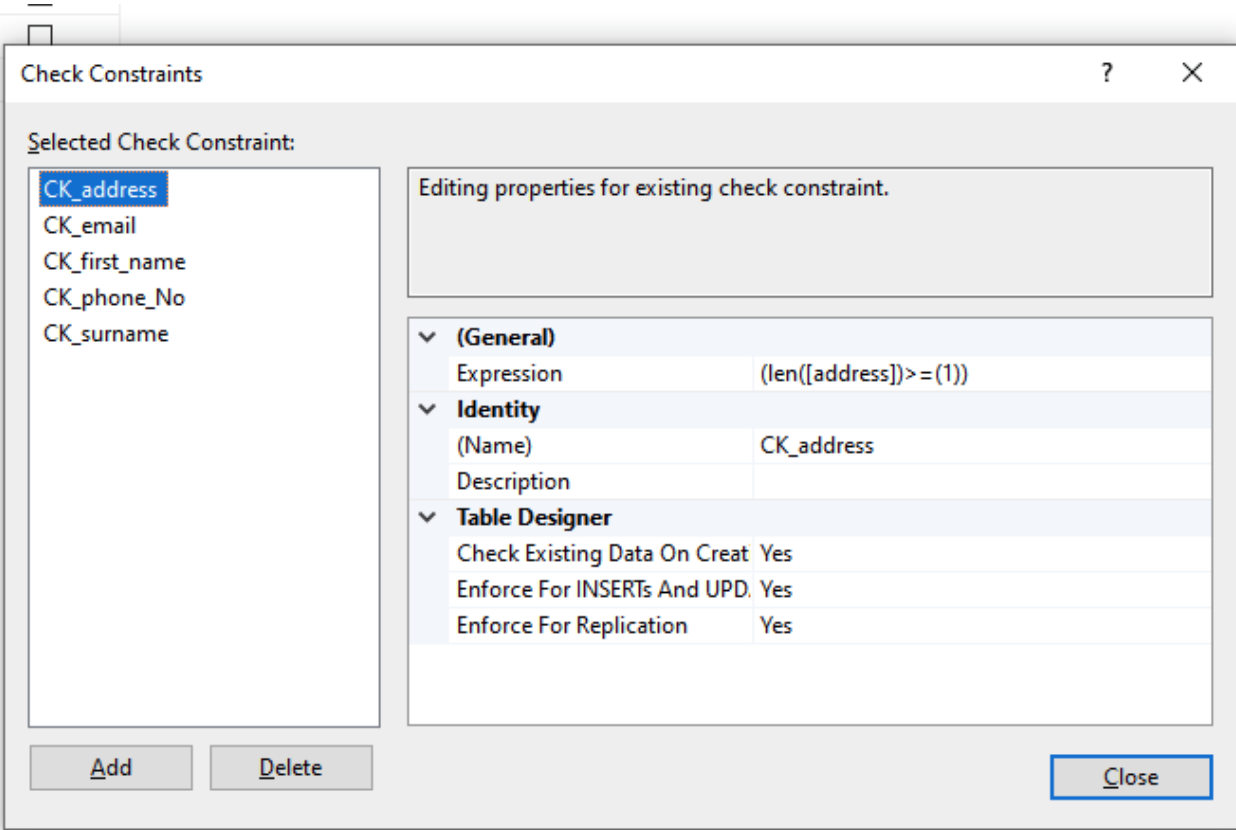


Figure 21 Testing and Verification Constraint Solution

Figure 21 shows the solution for the problem that the group encountered in the insertion of a blank textbox from GUI or with an input of spaces. It can be observed that each column of the table is being checked if the length of the data is greater than or equal to 1.

email	phone_No	address	isActive
eqwe12e	41241	effqqwr	True
rsurio@gmail.c...	3213		True
NULL	NULL	NULL	NULL

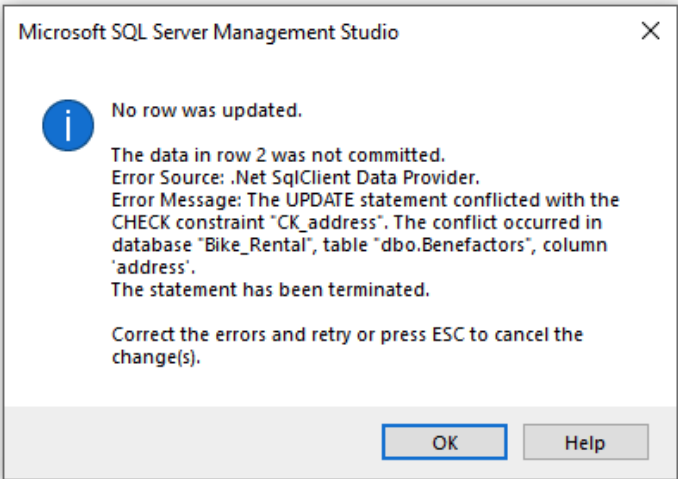


Figure 22 Testing and Verification Constraint Solution Checking

Figure 22 shows that a specific column does not accept space inputs now or an empty textbox insertion from GUI. A notice for the constraint created will pop up when tried to input space as input data.

```
using (SqlConnection Con = new SqlConnection(connectionString))
try
{
    Con.Open();

    if (isActiveState.Text == "")
    {
        MessageBox.Show("Active state is empty");
        Con.Close();
    }
    else if (bike_No.Text != "")
    {
        MessageBox.Show("The bike number is auto-generated!");
        Con.Close();
    }

    else
    {
        var Date = DateTime.Now.ToString("M/d/yyyy");
```

Figure 23 Testing and Verification for BIT Data Type and User-Generated Column

Figure 23 shows the algorithm created for the BIT data type and a PK from a table that is auto-generated. The constraints used for another column such as the constraints from Figure 21 cannot be applied into the active state checking column because this accepts a string and an integer hence checking the length cannot be implemented alone. To fix this problem and for making sure that it does not insert a blank value into the table just like what happened in Figure 20, the checking of input was done through the GUI or the textbox rather than the database.

Next is the PK of the table, in the insertion part, the textbox for the PK should not contain any value because it is auto-generated, hence it is being checked also through the GUI. The text box for the PK exists in the design because it is needed for the query in the updating part of the values from the table.

## Chapter V

### Summary of Findings, Conclusions, and Recommendations

This chapter will discuss the findings, conclusions, and recommendations of the created program Game Shop.

#### A. Summary of Findings

**Planning by creating an Entity Relationship Diagram, Unified Modeling Language Diagram, and Storyboarding made the creation of the program easier.**

Creating diagrams before creating the program made the creation of the system easier. The diagram is used as a guide to be followed while creating the program. Especially important is the Entity Relationship Diagram, which can be exported into SQL codes.

**A background in Programming Logic & Design and Object-Oriented Programming will make the development of the Bike Rental System easier.** The use of Programming Logic & Design and Object-Oriented Programming made the creation of the program easier. It also made the program shorter and functional. It also helped in creating the graphical user interface as the code was made using the C# language.

**Knowledge in SQL and SQL Server is necessary.** The study's main purpose is to test the knowledge of the programmers on SQL and SQL Server. The study made the programmers create a Bike Rental System that manages and stores information about the staff, beneficiaries, benefactors, and bikes. Using the knowledge gained from this course the programmers were able to create a functional system.

#### B. Conclusions

The Bike Rental System was accomplished using the knowledge gained from past and present courses. The Graphical User Interface was created using Programming Logic & Design and Object-Oriented Programming. The researchers were able to create the system efficiently by creating diagrams such as Entity Relationship, Unified Modelling Diagrams, and Storyboard. Lastly, the lessons that were learned in the current course made the creation of the database possible.

The Bike Rental System Database tables are created using Microsoft's SQL Server. The database has the following table names: Staff; Benefactors; Beneficiaries; Bikes; and Rental Records. The database also uses a Graphical User Interface using C# and the form creator that is included in Visual Studio. Lastly, The database can manage and store information of each table with proper relationship and attributes.

#### C. Recommendations

Future researchers may add a function which the database can be accessed online to so that it can be accessed anywhere with an internet connection. Another function to add

is the computations of transactions since the created database doesn't include one as it is a free community rental.



# Appendices

## Appendix A

### References

- [1]Gregorio, "Philippines confirms first case of novel coronavirus", *cnn*, 2021. [Online]. Available: <https://cnnphilippines.com/news/2020/1/30/Philippines-coronavirus-case.html>. [Accessed: 26- May- 2021].
- [2]Ramzy, "Philippines Reports First Coronavirus Death Outside China (Published 2020)", *Nytimes.com*, 2021. [Online]. Available: <https://www.nytimes.com/2020/02/02/world/asia/philippines-coronavirus-china.html>. [Accessed: 26- May- 2021].
- [3]DOH, "COVID-19 Tracker | Department of Health website", *Doh.gov.ph*, 2021. [Online]. Available: <https://doh.gov.ph/covid19tracker>. [Accessed: 26- May- 2021].
- [4]M. Domenico Cucinotta, "WHO Declares COVID-19 a Pandemic", *PubMed Central (PMC)*, 2021. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7569573/>. [Accessed: 26- May- 2021].
- [5]CNN Staff, "Metro Manila to be placed on 'lockdown' due to COVID-19", *cnn*, 2021. [Online]. Available: <https://cnnphilippines.com/news/2020/3/12/COVID-19-Metro-Manila-restrictions-Philippines.html>. [Accessed: 26- May- 2021].
- [6]Gregorio, "Philippines confirms first case of novel coronavirus", *cnn*, 2021. [Online]. Available: <https://cnnphilippines.com/news/2020/1/30/Philippines-coronavirus-case.html>. [Accessed: 26- May- 2021].
- [7]NEDA, "CYCLING, PROTECTED BIKE LANES CAN AUGMENT PUBLIC TRANSPORT SHORTAGE DURING PANDEMIC – NEDA - The National Economic and Development Authority", *The National Economic and Development Authority*, 2021. [Online]. Available: <https://www.neda.gov.ph/cycling-protected-bike-lanes-can-augment-public-transport-shortage-during-pandemic-neda/>. [Accessed: 26- May- 2021].
- [8]"Basics of Database Design & Development", Udemy, 2021. [Online]. Available: <https://www.udemy.com/course/database-design-development/>. [Accessed: 27- May- 2021].
- [9]"Database Design Tutorial: Learn Data Modeling", Guru99.com, 2021. [Online]. Available: <https://www.guru99.com/database-design.html>. [Accessed: 27- May- 2021].
- [10]"TechNet Wiki", Social.technet.microsoft.com, 2021. [Online]. Available: <https://social.technet.microsoft.com/wiki/contents/articles/930.sql-server-how-to-design-create->

and-maintain-a-database.aspx. [Accessed: 27- May- 2021].

[11]"SQL Commands | Codecademy", Codecademy, 2021. [Online]. Available:

<https://www.codecademy.com/articles/sql-commands>. [Accessed: 27- May- 2021].

[12] SQL CREATE DATABASE Statement. [Online]. Available:

[https://www.w3schools.com/sql/sql\\_create\\_db.asp](https://www.w3schools.com/sql/sql_create_db.asp). [Accessed: 27-May-2021].

[13] SQL DROP DATABASE Statement. [Online]. Available:

[https://www.w3schools.com/sql/sql\\_drop\\_db.asp](https://www.w3schools.com/sql/sql_drop_db.asp). [Accessed: 27-May-2021].

[14] SQL CREATE TABLE Statement. [Online]. Available:

[https://www.w3schools.com/sql/sql\\_create\\_table.asp](https://www.w3schools.com/sql/sql_create_table.asp). [Accessed: 27-May-2021].

[15] SQL PRIMARY KEY Constraint. [Online]. Available:

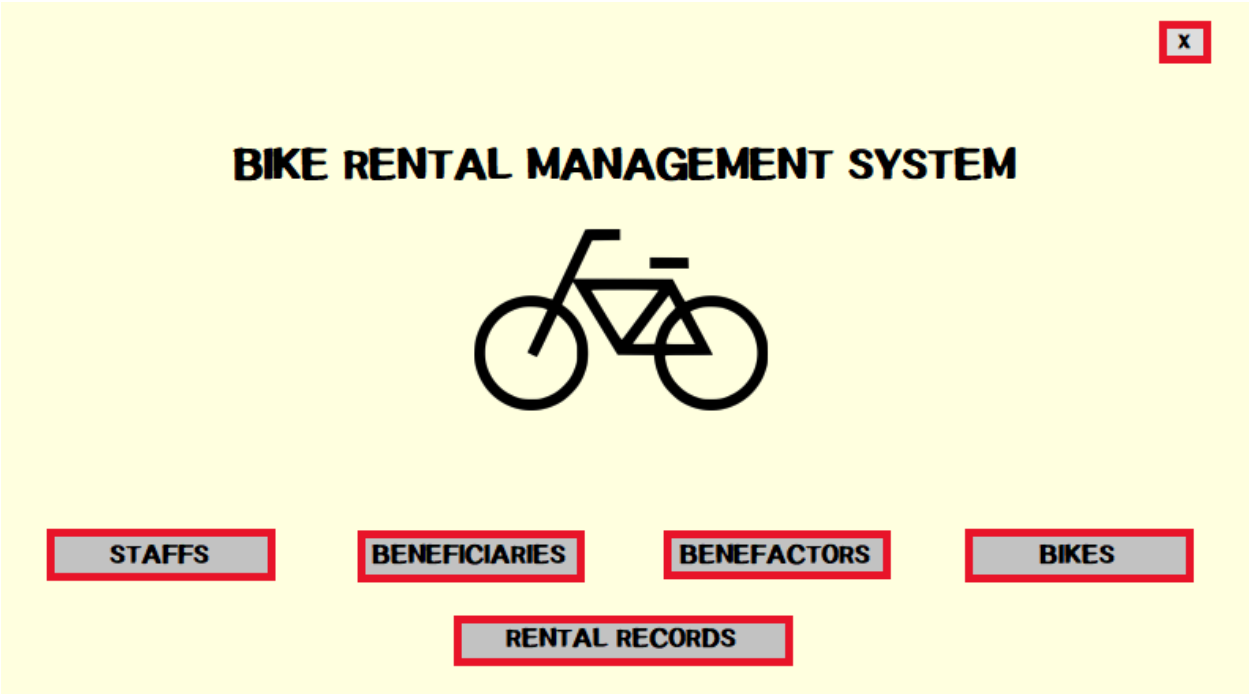
[https://www.w3schools.com/sql/sql\\_primarykey.asp](https://www.w3schools.com/sql/sql_primarykey.asp). [Accessed: 27-May-2021].

[16] SQL FOREIGN KEY Constraint. [Online]. Available:

[https://www.w3schools.com/sql/sql\\_foreignkey.asp](https://www.w3schools.com/sql/sql_foreignkey.asp). [Accessed: 27-May-2021].

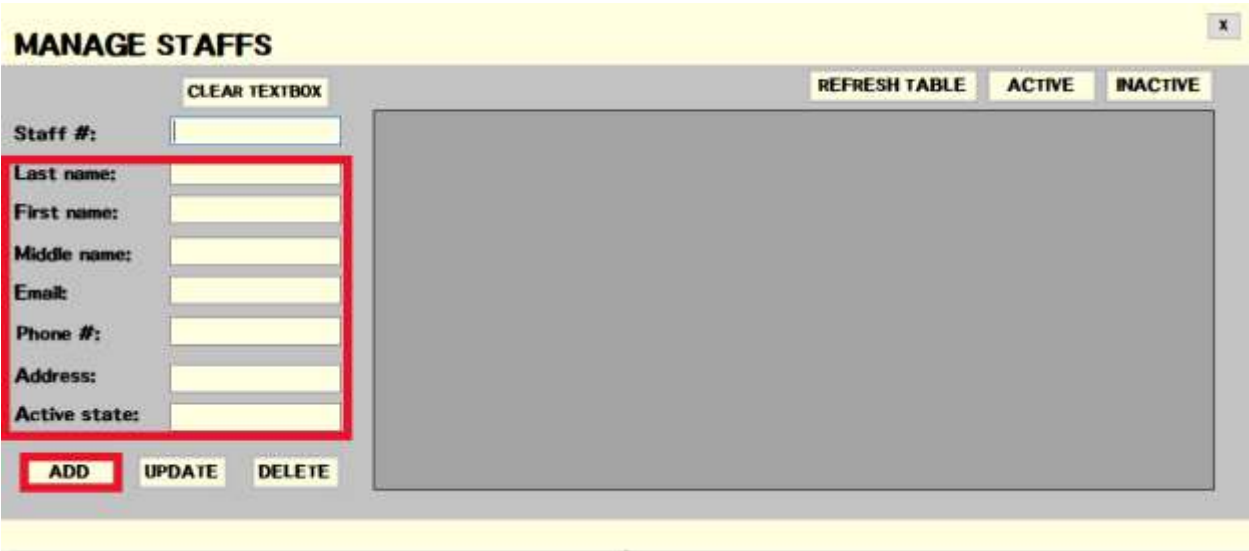
Appendix B

User Manual



1. Click the Staffs Button for opening the window for managing the staff records.
2. Click the Beneficiaries Button for opening the window for managing the beneficiary records.
3. Click the Benefactors Button for opening the window for managing the benefactor records.
4. Click the Bikes Button for opening the window for managing the bike records.
5. Click the Rental Records Button for opening the window for managing the rental records.
6. Click the “X” Button at the top right side of the window to close the application.

*(NOTE: Most of the procedure that will be shown below are identical to the other window hence this can serve as a reference since most of the windows have the same buttons available.)*



1. Input the details for the staff and click the “ADD” button if you want to add it to the database. *(Note: Leave the Staff # textbox blank when adding because it is auto-generated. It will be use for updating a row from table.)*

MANAGE STAFFS

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

Last name:

dwqe

First name:

ewqeqw

Middle name:

Email:

eqwqweqw

Phone #:

123123

Address:

eqeqw

Active state:

1

ADD

UPDATE

DELETE

2. Click the Clear Textbox Button if you want to delete everything from the textbox.

MANAGE STAFFS

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

1

Last name:

dwqe

First name:

ewqeqw

Middle name:

Email:

update

Phone #:

123123

Address:

eqeqw

Active state:

1

ADD

UPDATE

DELETE

3. Click the Update Button for updating a data or row from the database. Make sure to include the Staff # of the data or row.

MANAGE STAFFS

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

1

Last name:

dwqe

First name:

ewqeqw

Middle name:

Email:

update

Phone #:

123123

Address:

eqeqw

Active state:

1

ADD

UPDATE

DELETE

4. Click the Delete button for deleting a data or row from the table in the database. Make sure to include the Staff # of the data or row.

MANAGE STAFFS

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

Last name:

First name:

Middle name:

Email:

Phone #:

Address:

Active state:

ADD

UPDATE

DELETE

staff_no	surname	first_name	middle_name	email	phone_no	address	is
1	sufo	nvvl		dqjhw	3143	jp	

5. Click the Refresh Table Button to refresh the data grid view or the table shown.

MANAGE STAFFS

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

Last name:

First name:

Middle name:

Email:

Phone #:

Address:

Active state:

ADD

UPDATE

DELETE

	surname	first_name	middle_name	email	phone_no	address	isActive
1	sufo	nvvl		dqjhw	3143	jp	<input checked="" type="checkbox"/>

6. Click the Active Button if you wish to display only the active records.

MANAGE STAFFS

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

Last name:

First name:

Middle name:

Email:

Phone #:

Address:

Active state:

ADD

UPDATE

DELETE

	surname	first_name	middle_name	email	phone_no	address	isActive
1	ktazawa	yuho		yuho@	3213	3123	<input type="checkbox"/>

7. Click the Inactive Button if you wish to display only the inactive records.

MANAGE STAFFS

X

CLEAR TEXTBOX

REFRESH TABLE

ACTIVE

INACTIVE

Staff #:

Last name:

First name:

Middle name:

Email:

Phone #:

Address:

Active state:

ADD

UPDATE

DELETE

	surname	first_name	middle_name	email	phone_no	address	isActive
▶	kitazawa	yuho		yuho@	3213	3123	<input type="checkbox"/>
*							<input type="checkbox"/>

8. Click the “X” Button at the top right side of the window to go back to the main menu.

## Appendix C

### Source Code

**Github Repository:** <https://github.com/RovilSurioJr/Database>

#### **Program.cs**

```
using System;
using System.Windows.Forms;

namespace Bike_Rental_System
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new main());
        }
    }
}
```

#### **main.cs**

```
using System;
using System.Windows.Forms;

namespace Bike_Rental_System
{
    public partial class main : Form
    {
        public main()
        {
            InitializeComponent();
        }

        private void exitbut_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void staffsbut_Click(object sender, EventArgs e)
        {
            staff s = new staff();
            s.ShowDialog();
        }

        private void beneficiariesbut_Click(object sender, EventArgs e)
        {
            beneficiaries bfs = new beneficiaries();
            bfs.ShowDialog();
        }
    }
}
```



```

    }

    private void benefactorbut_Click(object sender, EventArgs e)
    {
        benefactors bns = new benefactors();
        bns.ShowDialog();
    }

    private void bikes_Click(object sender, EventArgs e)
    {
        bikes bi = new bikes();
        bi.ShowDialog();
    }

    private void rentalrecordbut_Click(object sender, EventArgs e)
    {
        rental_records rr = new rental_records();
        rr.ShowDialog();
    }
}

```

#### **staff.cs**

```

using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace Bike_Rental_System
{
    public partial class staff : Form
    {
        public staff()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            main s = new main();
            this.Close();
        }
        private void addStaffbutton_Click(object sender, EventArgs e)
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
            using (SqlConnection Con = new SqlConnection(connectionString))
            try
            {
                Con.Open();

                if (isActivestate.Text == "")
                {

```

```

        MessageBox.Show("Active state is empty");
        Con.Close();
    }
    else if (staff_No.Text != "")
    {
        MessageBox.Show("The staff number is auto-generated!");
        Con.Close();
    }
    else
    {
        string query = "INSERT INTO Staffs (surname, first_name,
middle_name,email,phone_No,address,isActive) VALUES ('" + surname.Text + "','" +
        first_name.Text + "','" + middle_name.Text + "','" + email.Text + "','" +
phone_No.Text + "','" + address.Text + "','" + isActivestate.Text + "')";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Staff added successfully");
        Con.Close();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
    Con.Close();
}
}
private void deletestaffbut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    try
    {
        if (staff_No.Text == "")
        {
            MessageBox.Show("Select the staff you want to delete");
        }
        else
        {
            Con.Open();
            string query = "DELETE FROM Staffs WHERE staff_No=" + staff_No.Text +
"";

            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Staff record deleted");
            Con.Close();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
}

```

```

private void updatebut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    try
    {
        if (staff_No.Text == "" || surname.Text == "" || first_name.Text == "" || email.Text
== "" || phone_No.Text == "" || address.Text == "" || isActivestate.Text == "")
        {
            MessageBox.Show("There is missing field! Only Middle_name allow NULLS");
            Con.Close();
        }
        else
        {
            Con.Open();
            string query = "UPDATE Staffs SET surname = " + surname.Text + ",
first_name = " + first_name.Text + ", " +
            "middle_name = " + middle_name.Text + ", email = " + email.Text +
            ",phone_No = " + phone_No.Text + ", " +
            "address = " + address.Text + ", isActive = " + isActivestate.Text + " WHERE
staff_No =" + staff_No.Text + """;
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Staff record updated");
            Con.Close();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        Con.Close();
    }
}

private void dgvstaffs_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = this.dgvstaffs.Rows[e.RowIndex];
        staff_No.Text = row.Cells["staff_No"].Value.ToString();
        surname.Text = row.Cells["surname"].Value.ToString();
        first_name.Text = row.Cells["first_name"].Value.ToString();
        middle_name.Text = row.Cells["middle_name"].Value.ToString();
        email.Text = row.Cells["email"].Value.ToString();
        phone_No.Text = row.Cells["phone_No"].Value.ToString();
        address.Text = row.Cells["address"].Value.ToString();
        isActivestate.Text = row.Cells["isActive"].Value.ToString();
    }
}

private void refreshbut_Click(object sender, EventArgs e)
{

```

```

        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))
        {
            Con.Open();
            string query = "SELECT * FROM Staffs";
            SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
            System.Data.DataTable dtbl = new System.Data.DataTable();
            sqldata.Fill(dtbl);
            dgvstaffs.DataSource = dtbl;
            Con.Close();
        }
    }

    private void activestaffsbut_Click(object sender, EventArgs e)
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))
        {
            Con.Open();
            string query = "SELECT staff_No, surname, first_name, middle_name, email,
phone_No, address, isActive from Staffs WHERE Staffs.isActive = 'TRUE'";
            SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
            System.Data.DataTable dtbl = new System.Data.DataTable();
            sqldata.Fill(dtbl);
            dgvstaffs.DataSource = dtbl;
            Con.Close();
        }
    }

    private void inactivestaffsbut_Click(object sender, EventArgs e)
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))
        {
            Con.Open();
            string query = "SELECT staff_No, surname, first_name, middle_name, email,
phone_No, address, isActive from Staffs WHERE Staffs.isActive = 'FALSE'";
            SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
            System.Data.DataTable dtbl = new System.Data.DataTable();
            sqldata.Fill(dtbl);
            dgvstaffs.DataSource = dtbl;
            Con.Close();
        }
    }

    private void cleartextbox_Click(object sender, EventArgs e)
    {
        staff_No.Text = "";
        surname.Text = "";
        first_name.Text = "";
        email.Text = "";
    }

```

```

        phone_No.Text = "";
        address.Text = "";
        isActiveState.Text = "";
        middle_name.Text = "";
    }
}
}

```

## **bikes.cs**

```

using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace Bike_Rental_System
{
    public partial class bikes : Form
    {
        public bikes()
        {
            InitializeComponent();
        }
        private void addBikebut_Click(object sender, EventArgs e)
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
            using (SqlConnection Con = new SqlConnection(connectionString))
            try
            {
                Con.Open();

                if (isActiveState.Text == "")
                {
                    MessageBox.Show("Active state is empty");
                    Con.Close();
                }
                else if (bike_No.Text != "")
                {
                    MessageBox.Show("The bike number is auto-generated!");
                    Con.Close();
                }

                else
                {
                    var Date = DateTime.Now.ToString("M/d/yyyy");
                    /* string query = "INSERT INTO Bikes (bike_model,
benefactor_No,bike_color,bike_accessory,bike_condition,donation_date,isActive) VALUES ("
+ bike_model.Text + "," +
                    benefactor_No.Text + "," + bike_color.Text + "," + bike_accessory.Text + "," +
                    bike_condition.Text + "," + donation_date.Text + "," + isActiveState.Text + ")"; */

```

```

        string query = "INSERT INTO Bikes (bike_model,
benefactor_No,bike_color,bike_accessory,bike_condition,isActive,donation_date) VALUES ("
+ bike_model.Text + "," +
        benefactor_No.Text + "," + bike_color.Text + "," + bike_accessory.Text + "," +
bike_condition.Text + "," + isActiveState.Text + "," + Date + ")";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Bike added successfully");
        Con.Close();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
    Con.Close();
}
}
private void inactivebut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();
        string query = "SELECT bike_No, bike_model, benefactor_No, bike_color,
bike_accessory, bike_condition, donation_date, isActive from Bikes WHERE Bikes.isActive =
'FALSE'";
        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
        System.Data.DataTable dtbl = new System.Data.DataTable();
        sqldata.Fill(dtbl);

        dgvBikes.DataSource = dtbl;
        Con.Close();
    }
}
private void activebut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();
        string query = "SELECT bike_No, bike_model, benefactor_No, bike_color,
bike_accessory, bike_condition, donation_date, isActive from Bikes WHERE Bikes.isActive =
'TRUE'";
        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
        System.Data.DataTable dtbl = new System.Data.DataTable();
        sqldata.Fill(dtbl);
        dgvBikes.DataSource = dtbl;
        Con.Close();
    }
}
private void editbikedetails_Click(object sender, EventArgs e)
{

```

```

        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))
        try
        {
            if (isActiveState.Text == "")

                {
                    MessageBox.Show("Active state is empty");
                    Con.Close();
                }
            else if (bike_No.Text == "")
            {
                MessageBox.Show("Please select the bike_No of bike you want to edit");
                Con.Close();
            }
            else
            {
                Con.Open();
                string query = "UPDATE Bikes SET bike_model = " + bike_model.Text + ",
benefactor_No = " + benefactor_No.Text + ", " +
                "bike_color = " + bike_color.Text + ", bike_accessory = " + bike_accessory.Text
+ ",bike_condition = " + bike_condition.Text + ", " +
                "isActive = " + isActiveState.Text + " WHERE bike_No =" + bike_No.Text +
""";

                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Bike record was updated");
                Con.Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            Con.Close();
        }
    }

    private void deletebutbikes_Click(object sender, EventArgs e)
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))
        try
        {
            if (bike_No.Text == "")
            {
                MessageBox.Show("Select the bike you want to delete");
            }
            else
            {
                Con.Open();
                string query = "DELETE FROM Bikes WHERE bike_No=" + bike_No.Text + ";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            Con.Close();
        }
    }
}

```

```

        MessageBox.Show("Bike record deleted");
        Con.Close();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
private void dgvBikes_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = this.dgvBikes.Rows[e.RowIndex];
        bike_No.Text = row.Cells["bike_No"].Value.ToString();
        bike_model.Text = row.Cells["bike_model"].Value.ToString();
        benefactor_No.Text = row.Cells["benefactor_No"].Value.ToString();
        bike_color.Text = row.Cells["bike_color"].Value.ToString();
        bike_accessory.Text = row.Cells["bike_accessory"].Value.ToString();
        bike_condition.Text = row.Cells["bike_condition"].Value.ToString();
        isActiveState.Text = row.Cells["isActive"].Value.ToString();

    }
}
private void cleartextbox_Click(object sender, EventArgs e)
{
    bike_model.Text = "";
    benefactor_No.Text = "";
    bike_color.Text = "";
    bike_accessory.Text = "";
    bike_condition.Text = "";
    isActiveState.Text = "";
    bike_No.Text = "";
}
private void showbenefactors_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();
        string query = "SELECT benefactor_No, surname, first_name from Benefactors
WHERE Benefactors.isActive = 'TRUE'";
        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
        System.Data.DataTable dtbl = new System.Data.DataTable();
        sqldata.Fill(dtbl);
        dgvBikes.DataSource = dtbl;
        Con.Close();
    }
}
private void refreshbut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

```



```

using (SqlConnection Con = new SqlConnection(connectionString))
{
    Con.Open();
    string query = "SELECT * from Bikes";
    SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
    System.Data.DataTable dtbl = new System.Data.DataTable();
    sqldata.Fill(dtbl);
    dgvBikes.DataSource = dtbl;
    Con.Close();
}
}
private void exitbut_Click(object sender, EventArgs e)
{
    main s = new main();
    this.Close();
}
}
}

```

### **beneficiaries.cs**

```

using System;
using System.Data.SqlClient;
using System.Windows.Forms;
using System.Configuration;

namespace Bike_Rental_System
{
    public partial class beneficiaries : Form
    {
        public beneficiaries()
        {
            InitializeComponent();
        }
        private void exitbut_Click(object sender, EventArgs e)
        {
            main s = new main();
            this.Close();
        }
        private void addbut_Click(object sender, EventArgs e)
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
            using (SqlConnection Con = new SqlConnection(connectionString))
            try
            {
                Con.Open();
                if (beneficiary_No.Text != "")
                {
                    MessageBox.Show("The beneficiary number is auto-generated!");
                    Con.Close();
                }
                else
                {

```

```

        string query = "INSERT INTO Beneficiaries (surname, first_name,
middle_name,email,phone_No,address) VALUES ('"+ surname.Text + "','" +
        first_name.Text + "','" + middle_name.Text + "','" + email.Text + "','" +
phone_No.Text + "','" + address.Text + "')";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Beneficiary added successfully");
        Con.Close();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
    Con.Close();
}
}

private void editbut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    try
    {
        if (beneficiary_No.Text == "")
        {
            MessageBox.Show("Please select the beneficiary number");
            Con.Close();
        }
        else
        {
            Con.Open();
            string query = "UPDATE Beneficiaries SET surname = '" + surname.Text + "',
first_name = '" + first_name.Text + "', " +
            "middle_name = '" + middle_name.Text + "', email = '" + email.Text +
            "',phone_No = '" + phone_No.Text + "', " +
            "address = '" + address.Text + "' WHERE beneficiary_No ='" +
beneficiary_No.Text + "'";
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Beneficiary record updated");
            Con.Close();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        Con.Close();
    }
}

private void deletebut_Click(object sender, EventArgs e)
{

```

```

        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))
        try
        {
            if (beneficiary_No.Text == "")
            {
                MessageBox.Show("Select the beneficiary to delete");
            }
            else
            {
                Con.Open();
                string query = "DELETE FROM Beneficiaries WHERE beneficiary_No=" +
beneficiary_No.Text + "";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Beneficiary record deleted");
                Con.Close();
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void refreshbut_Click(object sender, EventArgs e)
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
        using (SqlConnection Con = new SqlConnection(connectionString))

        {
            Con.Open();
            string query = "SELECT * FROM Beneficiaries";
            SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);
            System.Data.DataTable dtbl = new System.Data.DataTable();
            sqldata.Fill(dtbl);
            dgvbenefeciaries.DataSource = dtbl;
            Con.Close();
        }
    }

    private void dgvbenefeciaries_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            DataGridViewRow row = this.dgvbenefeciaries.Rows[e.RowIndex];
            beneficiary_No.Text = row.Cells["beneficiary_No"].Value.ToString();
            surname.Text = row.Cells["surname"].Value.ToString();
            first_name.Text = row.Cells["first_name"].Value.ToString();
            middle_name.Text = row.Cells["middle_name"].Value.ToString();
        }
    }

```

```

        email.Text = row.Cells["email"].Value.ToString();
        phone_No.Text = row.Cells["phone_No"].Value.ToString();
        address.Text = row.Cells["address"].Value.ToString();
    }
}

private void cleartext_Click(object sender, EventArgs e)
{
    beneficiary_No.Text = "";
    surname.Text = "";
    first_name.Text = "";
    middle_name.Text = "";
    email.Text = "";
    phone_No.Text = "";
    address.Text = "";
}

}
}

```

### **benefactors.cs**

```

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace Bike_Rental_System
{
    public partial class benefactors : Form
    {
        public benefactors()
        {
            InitializeComponent();
        }

        private void addbut_Click(object sender, EventArgs e)
        {
            string connectionString =
            ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

            using (SqlConnection Con = new SqlConnection(connectionString))

```

```

try
{
    Con.Open();

    if(isActiveState.Text == "")
    {
        MessageBox.Show("Active state is empty");

        Con.Close();
    }

    else if (benefactor_No.Text != "")
    {
        MessageBox.Show("The benefactor number is auto-generated!");

        Con.Close();
    }

    else
    {
        string query = "INSERT INTO Benefactors (surname, first_name,
middle_name,email,phone_No,address,isActive) VALUES ('" + last_name.Text + "','" +
        first_name.Text + "','" + middle_name.Text + "','" + email.Text + "','" +
phone_No.Text + "','" + address.Text + "','" + isActiveState.Text+"")";

        SqlCommand cmd = new SqlCommand(query, Con);

        cmd.ExecuteNonQuery();

        MessageBox.Show("Benefactor added successfully");

        Con.Close();
    }
}

catch (Exception ex)
{
    MessageBox.Show(ex.Message);

    Con.Close();
}
}

```

```

private void refreshbut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();

        string query = "SELECT * FROM Benefactors";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvbenefactors.DataSource = dtbl;

        Con.Close();
    }
}

private void activerecordbut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();

        string query = "SELECT benefactor_No, surname, first_name, middle_name,
email, phone_No, address, isActive from Benefactors WHERE Benefactors.isActive = 'TRUE'";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvbenefactors.DataSource = dtbl;

        Con.Close();
    }
}

```

```

    }

    private void inactiverecordbut_Click(object sender, EventArgs e)

    {

        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

        using (SqlConnection Con = new SqlConnection(connectionString))

        {

            Con.Open();

            string query = "SELECT benefactor_No, surname, first_name, middle_name,
email, phone_No, address, isActive from Benefactors WHERE Benefactors.isActive =
'FALSE'";

            SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

            System.Data.DataTable dtbl = new System.Data.DataTable();

            sqldata.Fill(dtbl);

            dgvbenefactors.DataSource = dtbl;

            Con.Close();

        }

    }

    private void dgvbenefactors_CellContentClick(object sender,
DataGridViewCellEventArgs e) // Clicking the Data Grid View will put in the text box the
content

    {

        if (e.RowIndex >= 0)

        {

            DataGridViewRow row = this.dgvbenefactors.Rows[e.RowIndex];

            benefactor_No.Text = row.Cells["benefactor_No"].Value.ToString();

            last_name.Text = row.Cells["surname"].Value.ToString();

            first_name.Text = row.Cells["first_name"].Value.ToString();

            middle_name.Text = row.Cells["middle_name"].Value.ToString();

            email.Text = row.Cells["email"].Value.ToString();

            phone_No.Text = row.Cells["phone_No"].Value.ToString();

            address.Text = row.Cells["address"].Value.ToString();

```

```

        isActiveState.Text = row.Cells["isActive"].Value.ToString();
    }
}

private void deletebut_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
    using (SqlConnection Con = new SqlConnection(connectionString))
    try
    {
        if(benefactor_No.Text == "")
        {
            MessageBox.Show("Select the benefactor to delete");
        }
        else
        {
            Con.Open();

            string query = "DELETE FROM Benefactors WHERE benefactor_No=" +
benefactor_No.Text + "";

            SqlCommand cmd = new SqlCommand(query, Con);

            cmd.ExecuteNonQuery();

            MessageBox.Show("Benefactor record deleted");

            Con.Close();
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);

        Con.Close();
    }
}

private void editbut_Click(object sender, EventArgs e)

```



```

    {
        string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

        using (SqlConnection Con = new SqlConnection(connectionString))

        try
        {

            //if (benefactor_No.Text == "" || last_name.Text == "" || first_name.Text == "" ||
email.Text == "" || phone_No.Text == "" || address.Text == "")

            if (isActiveState.Text == "")

            {

                MessageBox.Show("Active state is empty");

                Con.Close();

            }

            else

            {

                Con.Open();

                string query = "UPDATE Benefactors SET surname = " + last_name.Text + ",
first_name = " + first_name.Text + ", " +

                "middle_name = " + middle_name.Text + ", email = " + email.Text +

                ",phone_No = " + phone_No.Text + ", " +

                "address = " + address.Text + ", isActive = " + isActiveState.Text + " WHERE
benefactor_No = " + benefactor_No.Text + """;

                SqlCommand cmd = new SqlCommand(query, Con);

                cmd.ExecuteNonQuery();

                MessageBox.Show("Benefactor record updated");

                Con.Close();

            }

        }

        catch (Exception ex)

        {

            MessageBox.Show(ex.Message);

            Con.Close();

        }

    }

```

```

    }

    private void cleartextbox_Click(object sender, EventArgs e)
    {
        benefactor_No.Text = "";
        last_name.Text = "";
        first_name.Text = "";
        middle_name.Text = "";
        email.Text = "";
        phone_No.Text = "";
        address.Text = "";
        isActiveState.Text = "";

    }

    private void exitbut_Click(object sender, EventArgs e)
    {
        main s = new main();
        this.Close();
    }
}

```

### **rental\_records.cs**

```

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System;

namespace Bike_Rental_System
{
    public partial class rental_records : Form

```

```

{
public rental_records()
{
InitializeComponent();
}

private void exitbut_Click(object sender, EventArgs e)
{
main s = new main();
this.Close();
}

private void addbut_Click(object sender, EventArgs e)
{
string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;
using (SqlConnection Con = new SqlConnection(connectionString))
try
{
Con.Open();

if (rental_rec_No.Text != "")
{
MessageBox.Show("The rental record number is auto-generated!");
Con.Close();
}

else if (validity.Text == "")
{
MessageBox.Show("Active state is empty");
Con.Close();
}

else

```

```

        {
            var Date = DateTime.Now.ToString("M/d/yyyy");

            string query = "INSERT INTO Rental_records (bike_No,
beneficiary_No,rental_date,bike_condition_before,staff_lender_No,isValid) VALUES ('" +
bike_No.Text + "','" +

            beneficiary_No.Text + "','" + Date + "','" + cond_before.Text + "','" +
lender_staff_No.Text + "','" + validity.Text + "' )";

            SqlCommand cmd = new SqlCommand(query, Con);

            cmd.ExecuteNonQuery();

            MessageBox.Show("Bike Rental Record Added Successfully");

            Con.Close();

        }
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);

        Con.Close();

    }
}

private void showbeneficiary_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();

        string query = "SELECT beneficiary_No, surname, first_name FROM
Beneficiaries";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvrentalrec.DataSource = dtbl;
    }
}

```

```

        Con.Close();
    }
}

private void dgvrentalrec_CellContentClick(object sender, DataGridViewCellEventArgs
e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = this.dgvrentalrec.Rows[e.RowIndex];

        rental_rec_No.Text = row.Cells["rental_record_No"].Value.ToString();

        bike_No.Text = row.Cells["bike_No"].Value.ToString();

        beneficiary_No.Text = row.Cells["beneficiary_No"].Value.ToString();

        cond_before.Text = row.Cells["bike_condition_before"].Value.ToString();

        cond_after.Text = row.Cells["bike_condition_after"].Value.ToString();

        lender_staff_No.Text = row.Cells["staff_lender_No"].Value.ToString();

        receiver_staff_No.Text = row.Cells["staff_receiver_No"].Value.ToString();

        validity.Text = row.Cells["isValid"].Value.ToString();

    }
}

private void showbikes_Click(object sender, EventArgs e)
{
    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))
    {
        Con.Open();

        string query = "SELECT bike_No, bike_model, bike_color,bike_condition from
Bikes WHERE Bikes.isActive = 'TRUE'";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

```

```

        sqldata.Fill(dtbl);

        dgvrentalrec.DataSource = dtbl;

        Con.Close();

    }

}

private void showstaffs_Click(object sender, EventArgs e)

{

    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))

    {

        Con.Open();

        string query = "SELECT staff_No, surname, first_name FROM Staffs WHERE
Staffs.isActive = 'TRUE'";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvrentalrec.DataSource = dtbl;

        Con.Close();

    }

}

private void refreshbut_Click(object sender, EventArgs e)

{

    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))

    {

        Con.Open();

        string query = "SELECT * from Rental_records";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

```

```

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvrentalrec.DataSource = dtbl;

        Con.Close();

    }

}

private void editbut_Click(object sender, EventArgs e)

{

    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))

    try

    {

        if (rental_rec_No.Text == "")

        {

            MessageBox.Show("Specify the rental record number you want to edit");

            Con.Close();

        }

        else if (cond_after.Text == "" || receiver_staff_No.Text == "")

        {

            MessageBox.Show("Please input the condition after and the staff receiver
number");

            Con.Close();

        }

        else

        {

            var Date = DateTime.Now.ToString("M/d/yyyy");

            Con.Open();

            string query = "UPDATE Rental_records SET bike_No = " + bike_No.Text + ",
beneficiary_No = " + beneficiary_No.Text + ", bike_condition_after = " + cond_after.Text + ",
staff_receiver_No = " + receiver_staff_No.Text + ", " +

```

```
        "isValid = '" + validity.Text + "', return_date = '" + Date + "' WHERE  
rental_record_No ='" + rental_rec_No.Text + "'";
```

```
        SqlCommand cmd = new SqlCommand(query, Con);  
  
        cmd.ExecuteNonQuery();  
  
        MessageBox.Show("Rental record was updated");  
  
        Con.Close();  
    }  
}  
  
catch (Exception ex)  
{  
  
    MessageBox.Show(ex.Message);  
  
    Con.Close();  
}  
}  
  
private void deletebut_Click(object sender, EventArgs e)  
{  
  
    string connectionString =  
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;  
  
    using (SqlConnection Con = new SqlConnection(connectionString))  
  
    try  
    {  
  
        if (rental_rec_No.Text == "")  
        {  
  
            MessageBox.Show("Select the record you want to delete");  
  
        }  
  
        else  
        {  
  
            Con.Open();  
  
            string query = "DELETE FROM Rental_records WHERE rental_record_No=" +  
rental_rec_No.Text + "";  
  
            SqlCommand cmd = new SqlCommand(query, Con);
```



```

        cmd.ExecuteNonQuery();

        MessageBox.Show("Bike rental record deleted");

        Con.Close();

    }

}

catch (Exception ex)

{

    MessageBox.Show(ex.Message);

}

}

private void active_but_Click(object sender, EventArgs e)

{

    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))

    {

        Con.Open();

        string query = "SELECT rental_record_No, bike_No, beneficiary_No,
rental_date, return_date, bike_condition_before, bike_condition_after, staff_lender_No,
staff_receiver_No,isValid FROM Rental_records WHERE Rental_records.isValid = 'TRUE'";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvrentalrec.DataSource = dtbl;

        Con.Close();

    }

}

private void inactivebut_Click(object sender, EventArgs e)

{

    string connectionString =
ConfigurationManager.ConnectionStrings["dbx"].ConnectionString;

    using (SqlConnection Con = new SqlConnection(connectionString))

    {

```

```

        Con.Open();

        string query = "SELECT rental_record_No, bike_No, beneficiary_No,
rental_date, return_date, bike_condition_before, bike_condition_after, staff_lender_No,
staff_receiver_No,isValid FROM Rental_records WHERE Rental_records.isValid = 'FALSE'";

        SqlDataAdapter sqldata = new SqlDataAdapter(query, Con);

        System.Data.DataTable dtbl = new System.Data.DataTable();

        sqldata.Fill(dtbl);

        dgvrentalrec.DataSource = dtbl;

        Con.Close();

    }

}

private void cleartext_Click(object sender, EventArgs e)
{
    bike_No.Text = "";
    beneficiary_No.Text = "";
    cond_after.Text = "";
    cond_before.Text = "";
    rental_rec_No.Text = "";
    lender_staff_No.Text = "";
    receiver_staff_No.Text = "";
    validity.Text = "";
}

}

}

```

Appendix C

Researcher’s Profile



# GUY, LAWRENCE ADRIAN B.

Computer Engineer

**PROFILE**

I am a hardworking student currently studying in Adamson University. I currently major in Bachelor of Science in Computer Engineering.

**CONTACT**

PHONE:  
09613081274

EMAIL:  
lawrenceadrianguy@gmail.com

**HOBBIES**

Watching Anime/Kdrama  
Playing Computer Games  
Listening to Pop punk/Indie  
OPM/Kpop

**EDUCATION**

**Adamson University**  
High School  
2013 - 2019

**Adamson University**  
College  
2019 - 2023

**EXPERIENCE**

**Achievements**  
**With Honors**  
2019  
Senior High School

**Webinars**  
2020  
Eskwelabs: Aral Aral Python

**SKILLS**

Programming	90%
MS Office	90%
Leadership	60%
Teamwork	60%
Communication	30%



# SURIO, ROVIL JR., M.

Computer Engineer

## PROFILE

I am hardworking, ambitious, and creative, with broad knowledge and experience in coding with different languages.

A key strength is communication, building a good relationship with others to have good teamwork and deliver the best results.

## CONTACT

PHONE:  
09454022651

WEBSITE:  
<https://github.com/RovilSurioJr>

EMAIL:  
rovilsuriojr@gmail.com

## HOBBIES

Watching Anime/Kdrama  
Playing Basketball/Table tennis  
Listening to Music  
Collecting Figurines  
Custom PC Building

## EDUCATION

**Melanoiah Academy**  
Elementary and Junior High School  
2004 - 2017

**Statefields School**  
Senior High School  
2017 - 2019

**Adamson University**  
College  
2019 - 2023

## EXPERIENCE

### Achievements

2017  
With honors

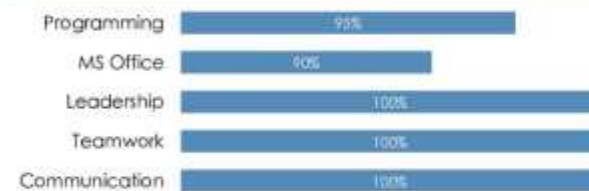
### Seminars

2019  
Environmental Awareness

### Webinars

2020  
Arise: Technological Matrix

## SKILLS





# MYKE ALVIN E. SUSTENTO

Computer Engineer

## EDUCATIONAL ATTAINMENT



High School  
2013 – 2019



BS Computer Engineering  
2019 – 2023

## ORGANIZATIONS

- Institute of Computer Engineers of the Philippines, Inc Student Edition (ICpEP.Se)
- Adamson University's Robotics and Programming Guild (RPG)

## ACHIEVEMENT / EXPERIENCES

- Graduated with Honors (Senior High School)
- Eskwelabs: Aral Aral Webinar Python
- Arise: Technological Matrix Webinar

## SKILLS



Microsoft Office



Programming



Editing

Email Address

mykesustento@gmail.com

Contact Number

092-308-59333

Location

Ilustre St. Sta. Cruz, Manila

Birthdate

September 16, 2000

Nationality

Filipino

Sex

Male

## LANGUAGES

English



Filipino






**VALLARTA**  
**Troy Joaquin G.**

COMPUTER ENGINEER

 Troy Vallarta

 @TroyVallarta

 Lot 2 Block 5 VSJ Subd.  
Brgy. Masalukot 1,  
Candelaria, Quezon

 0995-995-9229

ABOUT ME

I am a fresh graduate from Adamson University as a Computer Engineer as of S.Y. 2022-2023. The four years of learning in college gave me a lot of learning and a lot of experience in my life.

EDUCATION

Highschool



MANUEL S. ENVERGA  
UNIVERSITY FOUNDATION  
CANDELARIA INC.  
2013-2019

College



ADAMSON University  
2019-2023

EXPERIENCE

ACHIVEMENTS:

WITH HONORS SENIOR HIGH SCHOOL  
SERU RESPONDENT MSEUF

ADAMSON SEMINARS:

ENVIRONMENTAL AWERENESS

SKILLS

