BUS MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

MUTTHESH M **220701176**
GODLYLASON M **220701520**

In partial fulfillment for the award of the

degree ofBACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report " **BUS MANAGEMENT SYSTEM**" is the bonafide work of " **MUTTHESH M(220701176), GODLYLASON M(220701520)**" who carried out the project under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Dr.R.SABITHA

Professor and II Year Academic Head
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai - 602 105

SIGNATURE

Ms.D.KALPANA

Assistant Professor (SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

This study introduces an automated system for bus ticket booking,

aiming to enhance efficiency and convenience for both customers and

service providers. Traditional bus ticket booking processes often

involve long queues, manual transactions, and limited accessibility.To

address these challenges, our system leverages modern technology,

including web and mobile applications, to facilitate seamless booking

experiences. Key features include real-time seat availability updates,

secure online payments, and user-friendly interfaces. Additionally, the

system integrates with existing bus operator databases to ensure

accurate scheduling and timely updates for passengers.

Through a combination of user feedback, system analysis, and

performance evaluations, we demonstrate theeffectiveness of our

approach in optimizing the bus ticket booking process. Our findings

suggest that the adoption of automated systems holds significant

promise in revolutionizing the transportation industry, enhancing

customer satisfaction, and driving operational efficiency.

1.TABLE OF CONTENTS INTRODUCTION

1.1 INTRODUCTION TO BUS TICKET BOOKING SYSTEM

1.2 OBJECTIVES

1.3 MODULES

2.SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES USED IN BUS TICKET BOOKING SYSTEM

2.3 DATABASE MANAGEMENT SYSTEM: MONGODB

2.4 PROGRAMMING LANGUAGE: PYTHON

3. REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION FOR BUS TICKET BOOKING SYSTEM

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM

4.PROGRAM

4.1 PROGRAM CODE

5.RESULTS AND DISCUSSION

5.1 USER DOCUMENTATION

6. CONCLUSION

6.1 CONCLUSION

7. REFERENCES

7.1 REFERENCES

CHAPTER 1

1.1 INTRODUCTION

Efficient management of bus bookings is essential for transportation

companies and travelers alike. Traditional booking methods often lead to

errors and operational challenges. The implementation of an Automated

Bus Booking System (BBS) provides a viable solution. By harnessing

contemporary technologies, the BBS automates the booking process,

including reservation recording, data storage, and analysis.

With user friendly interfaces and robust data management capabilities,

it ensures seamless booking procedures, enhanced accuracy, and valuable

insights into booking trends. This introduction sets the stage for an

exploration of the BBS's design and deployment, highlighting its

significance in optimizing efficiency and reliability in the

transportation industry.

The way we manage bus bookings is on the cusp of a transformation.

Traditional methods, often prone to errors and inefficiencies, are being

eclipsed by a new era of automation. Enter the Automated Bus Booking

System (BBS), a powerful software solution designed to streamline the

booking process for both transportation companies and travelers.

This introduction serves as a springboard for delving deeper into the

transformative power of the BBS. By harnessing the latest technologies,

the BBS automates every step of the booking journey, from reservation

recording to data storage and analysis. This translates to

1.2 OBJECTIVES

The Bus Booking System (BBS) seeks to revolutionize the booking experience by automating reservation processes, improving accuracy, and

reducing time consumption. It places a premium on secure user

authentication to uphold data confidentiality and integrity. Utilizing

centralized data storage facilitates convenient retrieval and analysis,

empowering decision-makers with valuable insights. User-friendly

interfaces cater to passengers, operators, and administrators, **ensuring a**

smooth booking journey. Robust reporting tools identify booking patterns,

facilitating data-driven decisions. The system's scalability and adaptability

accommodate a variety of operational requirements. Leveraging cutting-

edge technologies, the BBS ensures reliability, performance, and cross-

platform compatibility. Ultimately, its goal is to enhance efficiency and

reliability in the transportation sector.

The BBS is designed with user-friendliness in mind. Intuitive

interfaces cater to passengers, bus operators, and administrators alike,

ensuring a smooth and seamless experience for everyone involved.

Passengers can book their seats with ease, operators can manage bookings

efficiently, and administrators have the tools they need to optimize the

entire system.

Informed choices lead to better outcomes. The BBS features robust

reporting tools that identify booking patterns and travel trends. This

empowers stakeholders to make data-driven decisions that enhance the

overall travel experience.

1.3 MODULES

- **Passenger Registration Module.**
- **Booking Management Module.**
- **Occupancy Management Module**

CHAPTER-2

2.1 SOFTWARE DESCRIPTION

Visual studio Code:

Visual Studio Code combines the simplicity of a source code editor withpowerful developer tooling, like IntelliSense code completion and debugging.

2.2 LANGUAGES

1. Python:

- It is used for scripting the application's logic, managing database operations, and integrating different modules.

2.Streamlit:

- A powerful Python library for creating interactive web applications with simple Python scripts, enabling rapid prototyping

and deployment of data-driven applications with ease..

3.MongoDB:

- A scalable NoSQL database solution, offering high performance, flexibility, and seamless integration with modern applications..

CHAPTER-3

REQUIREMENT AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION:

Bus Booking System

The Bus Booking System must meet stringent functional and non-functional criteria to ensure optimal performance and user satisfaction.

It must offer secure authentication for passengers and administrators, facilitate hassle-free booking processes with real-time seat availability updates, maintain a centralized database for secure data storage and retrieval, generate detailed reports on bookings and revenue, send timely notifications to passengers regarding booking confirmations and updates, and provide user-friendly interfaces for both passengers and administrators.

Non-functionally, the system must prioritize security measures such as data encryption and access controls, guarantee high performance to handle peak booking periods, scale effectively to accommodate increasing demand, ensure reliability through backup and recovery mechanisms, prioritize usability with intuitive design and navigation, and maintain compatibility across multiple platforms and devices to cater to diverse user needs. These requirements collectively aim to deliver a reliable, secure, and efficient bus booking solution for transportation companies and passengers.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

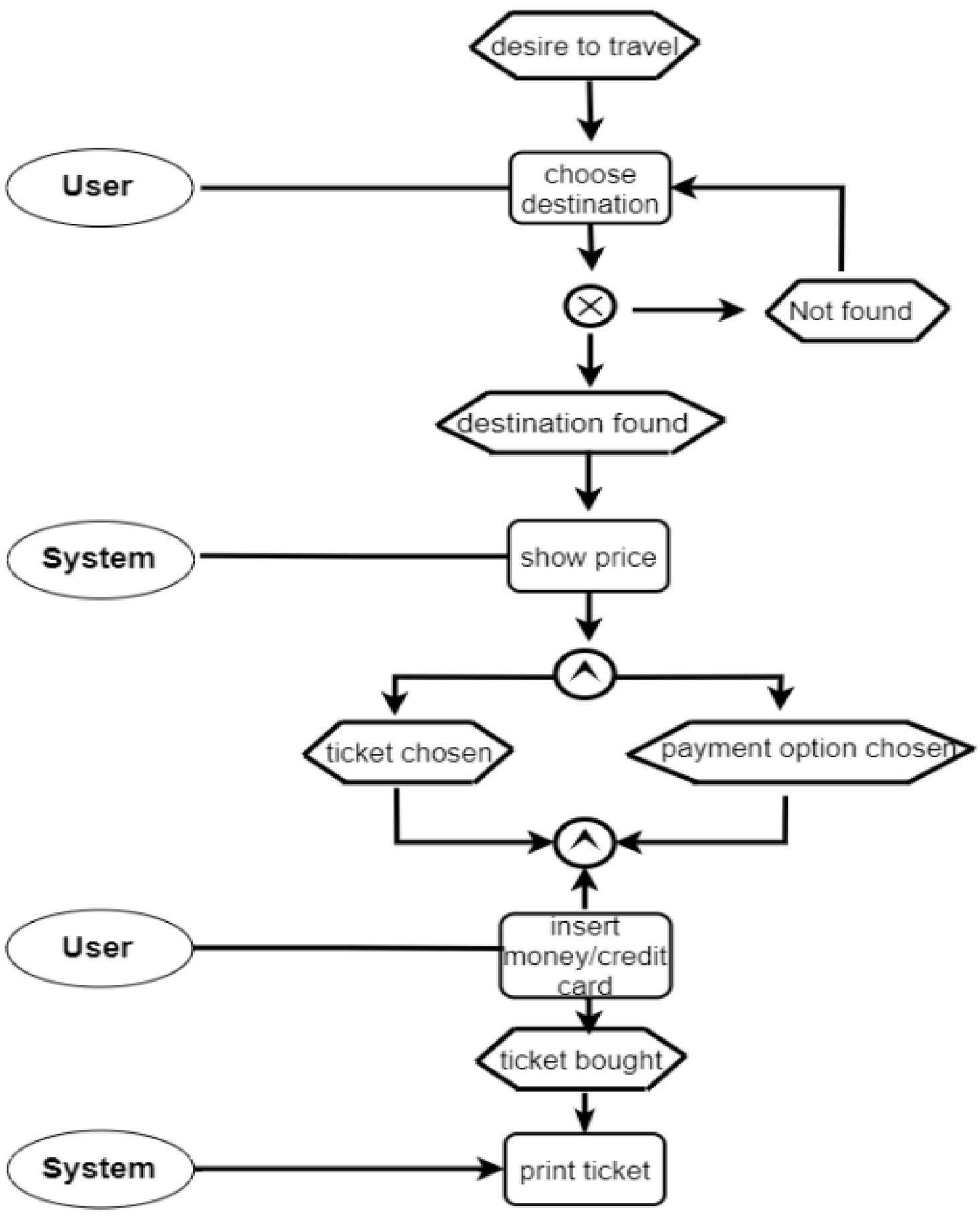
Hardware Requirements

- Processor: 1 GHz or faster processor
- RAM: 2 GB or more
- Storage: At least 500 MB of available disk space
- Display: Minimum resolution of 1024x768
- Input Devices: Keyboard and mouse

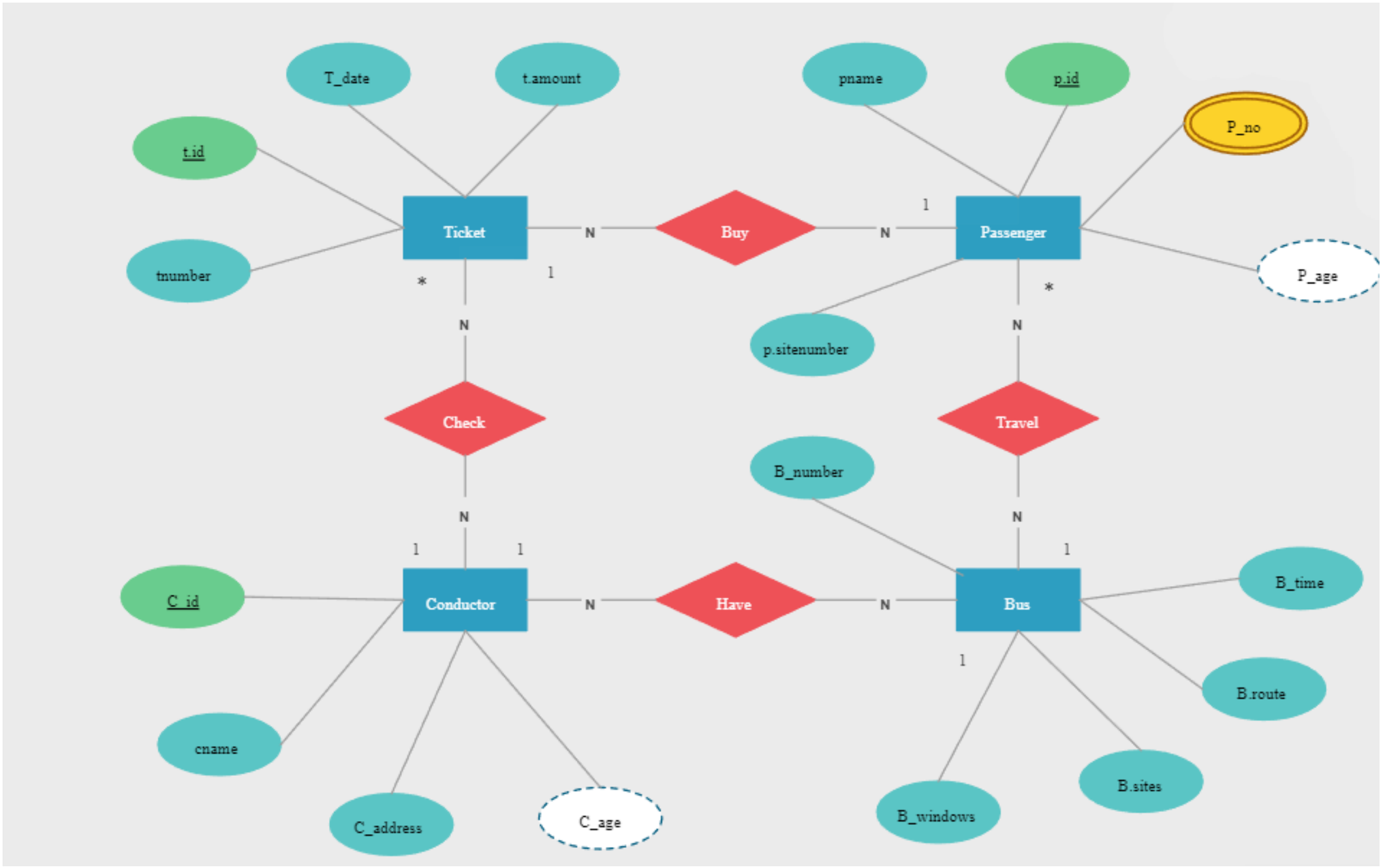
Software Requirements

- Operating System: Windows 7 or later, macOS, or Linux
- Python: Version 3.6 or higher
- SQLite: Version 3 or higher
- Python Libraries:
  - ‘ pymongo’ connecting to and interacting with MongoDB (use command pip install pymongo)
  - ‘ Streamlit’ building the web application interface (use command pip install streamlit)

3.1 ARCHITECTURE DIAGRAM:



3.2 ER DIAGRAM:



CHAPTER-4

PROGRAM CODE

```
import streamlit as st

from pymongo import MongoClient

from datetime import datetime



# MongoDB connection details

MONGO_URI = "mongodb://localhost:27017"

DB_NAME = "bus_ticket_booking_db"



# Backend: MongoDB Operations

class BusTicketDB:

    def _init_(self, uri, db_name):

        self.client = MongoClient(uri)

        self.db = self.client[db_name]
```

```
        self.bookings_collection = self.db['bookings']
        self.buses_collection = self.db['buses']


    def book_ticket(self, name, bus_number, departure, destination,
date):
        existing_booking =
self.bookings_collection.find_one({"bus_number": bus_number,
"date": date})
        if existing_booking:
            st.error("Booking already exists for this bus on the selected
date.")
        else:
            booking = {
                "name": name,
                "bus_number": bus_number,
                "departure": departure,
                "destination": destination,
                "date": date
            }
```

```
            self.bookings_collection.insert_one(booking)

            st.success("Ticket booked successfully")



def add_bus(self, bus_number, departure, destination, date):

        existing_bus = self.buses_collection.find_one({"bus_number":
bus_number, "date": date})



if existing_bus:

            st.error("Bus already exists with the same number and date.")

        else:

            bus = {

                "bus_number": bus_number,

                "departure": departure,

                "destination": destination,

                "date": date

            }

            self.buses_collection.insert_one(bus)

            st.success("Bus added successfully")
```

```
def get_buses(self):
        return list(self.buses_collection.find())

 def get_bookings(self):
        return list(self.bookings_collection.find())



def delete_booking(self, bus_number):
        self.bookings_collection.delete_one({"bus_number":
bus_number})


# Initialize database connection
db = BusTicketDB(MONGO_URI, DB_NAME)



# Page navigation
```

```
page = st.sidebar.selectbox("Select Page", ["Admin", "User"])


if page == "Admin":
    st.header("Admin Page")

    # Form to add a bus
    st.subheader("Add a Bus")
    bus_number = st.text_input("Bus Number")
    departure = st.text_input("Departure")
    destination = st.text_input("Destination")
    date = st.date_input("Date")


if st.button("Add Bus"):
        db.add_bus(bus_number, departure, destination,
date.strftime("%Y-%m-%d"))
```

```
# Display all buses
    st.subheader("View Buses")
    if st.button("Show Buses"):
        buses = db.get_buses()
        if buses:
            for bus in buses:
                st.write(f"Bus Number: {bus['bus_number']}, Departure:
{bus['departure']}, Destination: {bus['destination']}, Date: {bus['date']}")
        else:
            st.write("No buses found.")

elif page == "User":
    st.header("User Page")

# Form to book a ticket
    st.subheader("Book a Ticket")

    name = st.text_input("Name")
```

```
buses = db.get_buses()

bus_numbers = [bus['bus_number'] for bus in buses]

bus_number = st.selectbox("Bus Number", bus_numbers)

departure = st.text_input("Departure")

destination = st.text_input("Destination")

date = st.date_input("Date")
```

```
 if st.button("Book Ticket"):

      db.book_ticket(name, bus_number, departure, destination,
date.strftime("%Y-%m-%d"))
```

```
# Display all bookings

st.subheader("View Bookings")

if st.button("Show Bookings"):

   bookings = db.get_bookings()

   if bookings:

      for booking in bookings:

         st.write(f"ID: {booking['_id']} - Name: {booking['name']}, Bus
```

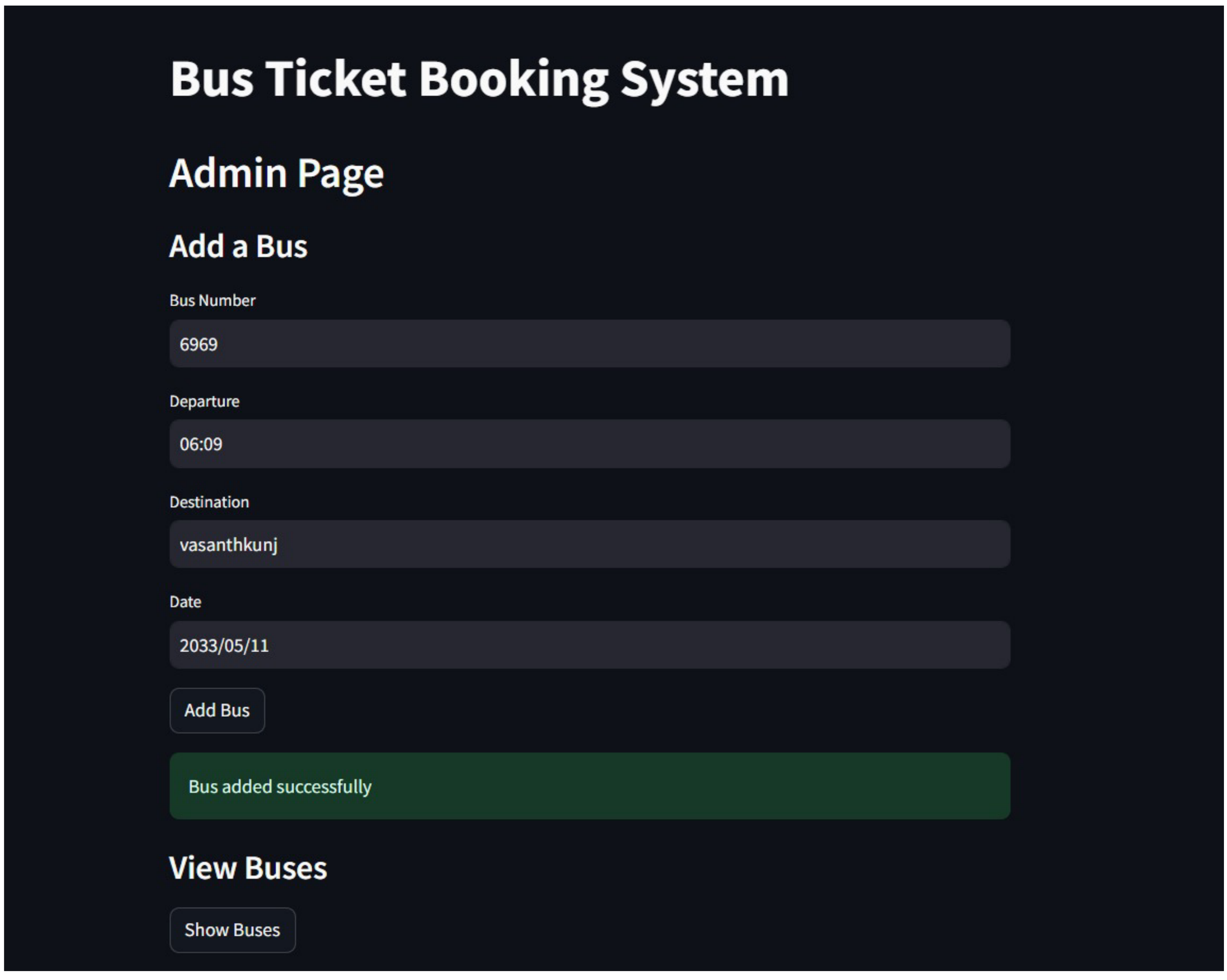
```
Number: {booking['bus_number']}, Departure: {booking['departure']},
Destination: {booking['destination']}, Date: {booking['date']}")
        else:
            st.write("No bookings found.")

    # Delete a booking
    st.subheader("Delete Booking")
    delete_bus_number = st.text_input("Bus Number to Delete
Booking")

    if st.button("Delete Booking"):
        try:
            db.delete_booking(delete_bus_number)
            st.success("Booking deleted successfully")
        except Exception as e:
            st.error(f"Error: {e}")
```

CHAPTER-5

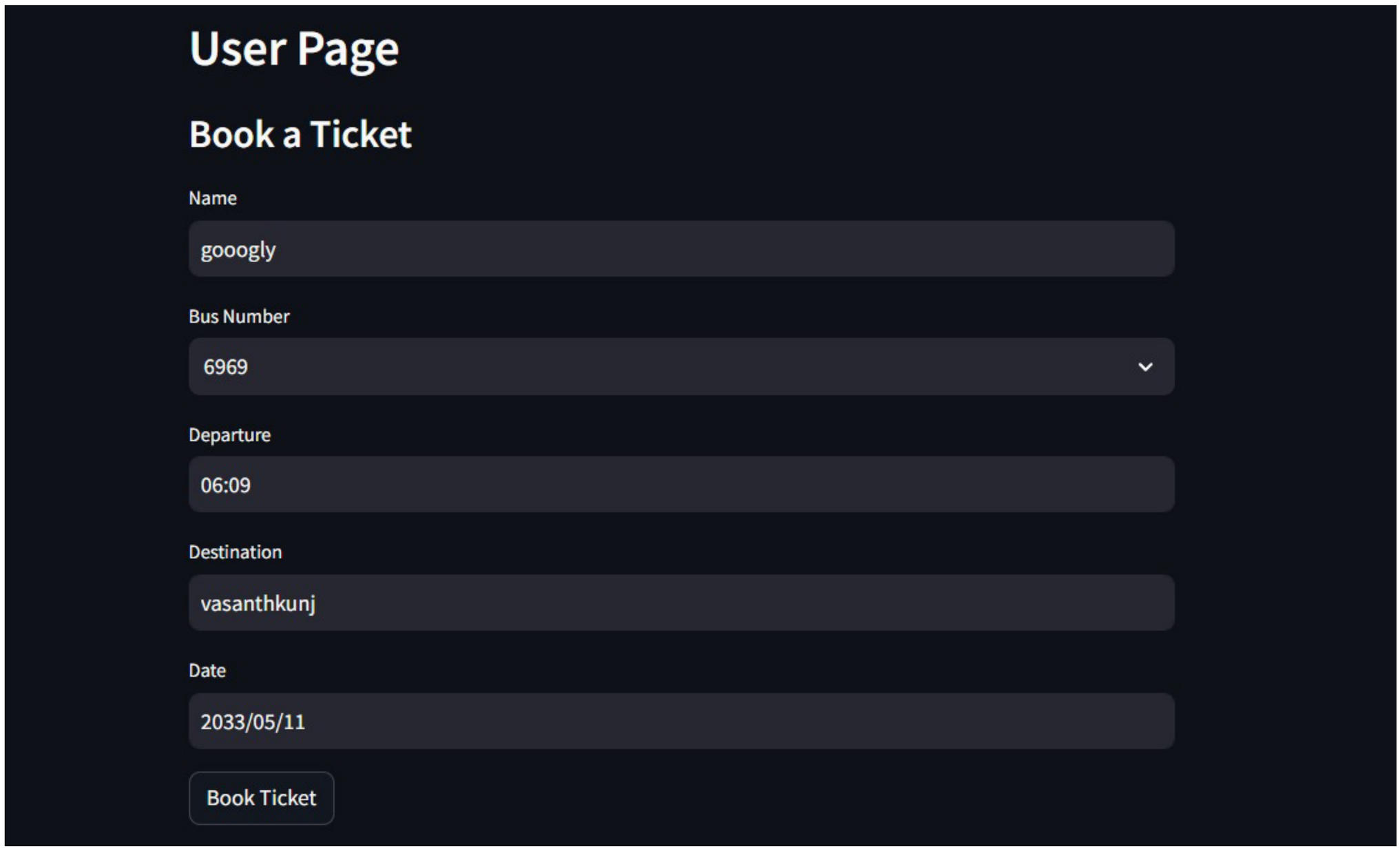
RESULTS AND DISCUSSION

5.1 USER DOCUMENTATION:

ADMIN PAGE MODULE:



BUS BOOKING MODULE:



SALES MANAGEMENT MODULE:

View Bookings

Show Bookings

ID: 66559644bd19124065106d2c - Name: gooogly, Bus Number: 6969, Departure: 06:09, Destination: vasanthkunj, Date: 2033-05-11

Delete Booking

Bus Number to Delete Booking

6969

Delete Booking

CHAPTER-6

6.1 CONCLUSION:

Having concluded the development of the Bus Ticket Booking System,

we are confident in its ability to effectively tackle the pertinent challenges

in managing bus ticket reservations. This sophisticated software solution

is meticulously crafted to minimize errors while significantly boosting

operational efficiency. The core objective of this project was to alleviate

the burden on human resources by automating repetitive tasks and

optimizing the booking process.

For instance, users can effortlessly search for available tickets by

inputting specific criteria, and the system facilitates seamless

modification of reservations through intuitive update functions.

In summary, the Bus Ticket Booking System successfully fulfills its

primary aim of facilitating accurate and efficient management of ticket

reservations. By automating data handling and simplifying navigation and

modification processes, the system empowers staff to dedicate their time

to more strategic endeavors. This project serves as a tangible example of

the practical application of database management systems (DBMS) in

addressing real-world challenges, particularly in the domain of bus ticket

reservation management.

The BBS is more than just a feature-rich platform; it's a foundation for

future advancements. The system's architecture allows for integration with

external services, such as loyalty programs, ride-sharing options for first

and last-mile connectivity, and real-time traffic updates to enhance the

overall travel experience.

By leveraging the BBS, bus companies can streamline operations,

improve customer satisfaction, and gain a competitive edge in the ever-

evolving transportation landscape.

· **User Management System:** Create different user roles with varying access

levels, ensuring data security and role-based functionality.

· **Real-Time Inventory Management:** Monitor seat availability across

the entire bus network, allowing for dynamic pricing strategies and

maximizing revenue opportunities.

- **Automated Fare Management:** Set ticket prices, manage discounts

and promotions, and automate fare calculations based on pre-

defined rules.

- **Advanced Reporting and Analytics:** Gain insights into overall system

performance, identify areas for improvement, and make strategic

decisions to optimize the entire bus booking ecosystem.

CHAPTER-7

7.1 REFERENCES:

1. Python Documentation: Python Software Foundation. (n.d.).

Python Documentation. Retrieved from

https://docs.python.org/3/

2.Tkinter Documentation: TkDocs. (n.d.). Tkinter Tutorial.

Retrieved from

https://tkdocs.com/tutorial/

3.SQLite Documentation: SQLite. (n.d.). SQLite Documentation.

Retrieved from

https://www.sqlite.org/docs.html

4.Automate the Boring Stuff with Python: Sweigart, A. (2015).

Automate the Boring Stuff with Python: Practical

Programming

for Total Beginners. No Starch Press.

5.Python GUI Programming with Tkinter: Grayson, J. E. (2000).

Python and Tkinter Programming. Manning Publications.

6.SQLite Database Programming: Owens, M. (2006). The

Definitive Guide to SQLite. Apress.

7.Database System Concepts: Silberschatz, A., Korth, H. F., &

Sudarshan, S. (2010). Database System Concepts (6th ed.).

McGraw-Hill.

8.GeeksforGeeks: Various authors. (n.d.). GeeksforGeeks.

Retrieved from

https://www.geeksforgeeks.org/

9.W3Schools: W3Schools. (n.d.). SQL Tutorial. Retrieved from https://www.w3schools.com/sql/