DATABASE MANAGEMENT SYSTEM MINI PROJECT

Title: Voice-Based Transport Enquiry System

Group Members: -

Roll No	Name	Semester
B-62	Vedant Ghodmare	6TH
B-63	Yash Soni	6TH
B-64	Atharva Rathi	6TH
B-65	Shrinit Jichkar	6TH
B-66	Vivek Ghosh	6TH

Problem Statement: The increasing demand for quick, hands-free access to real-time transport information highlights the need for a user-friendly solution beyond traditional web or mobile interfaces. This project aims to develop a Voice-Based Transport Enquiry System that allows users to retrieve schedules, routes, and availability for buses, trains, or flights using voice commands. By integrating speech recognition, text-to-speech, and a relational database, the system ensures accurate, real-time responses, making transport information more accessible, especially for users with disabilities or in hands-busy situations.

Objectives

- To provide real-time transport information using voice commands.
- To create an interactive and accessible interface for all users.
- To allow admin-based management of transport routes and data.
- To implement a relational database for reliable, structured storage.
- To reduce dependency on manual systems and support digital transport services

Introduction:

The Voice-Based Automated Transport Enquiry System offers a practical and efficient way to assist users in travel planning by providing real-time schedule updates through voice interaction. It eliminates the need for human-operated help desks and operates without constant monitoring, making it a cost-effective solution. Public transport services can benefit from improved efficiency and user engagement. Especially for visitors unfamiliar with local routes or language, the system offers an accessible and user-friendly interface to find transport options and plan their journeys with ease.

Functionality:

The system provides two distinct login modes: one for general users and another for administrators. Administrators have full control over the transport database, including the ability to add, modify, or delete bus route information. General users, on the other hand, have read-only access to view schedules and routes.

Users can interact with the system using voice commands by specifying the source and destination. These voice inputs are converted into database queries, and relevant transport details are retrieved and provided both audibly and visually on the screen.

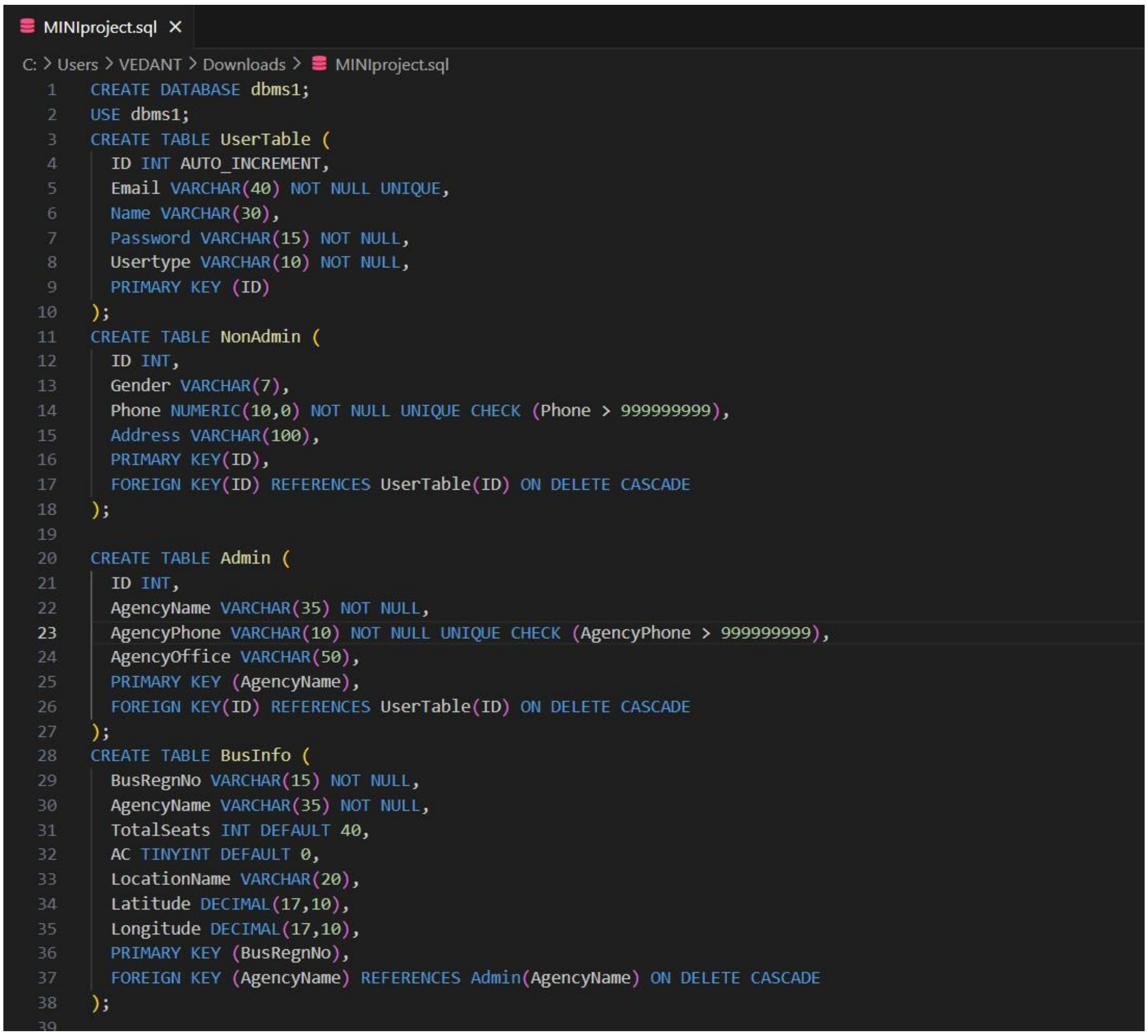
The system also allows users to reserve or cancel tickets based on availability. Additionally, users can access details such as available routes, fares, estimated time of arrival (ETA), and distance. Real-time bus location tracking is also supported for better journey planning.

Modules

- Administrator Login
- Transport Info Entry
- Transaction Report
- Customer Registration
- Customer Login
- Result in Text-to-Speech and UI Display

Development Platform: - MySQL Workbench

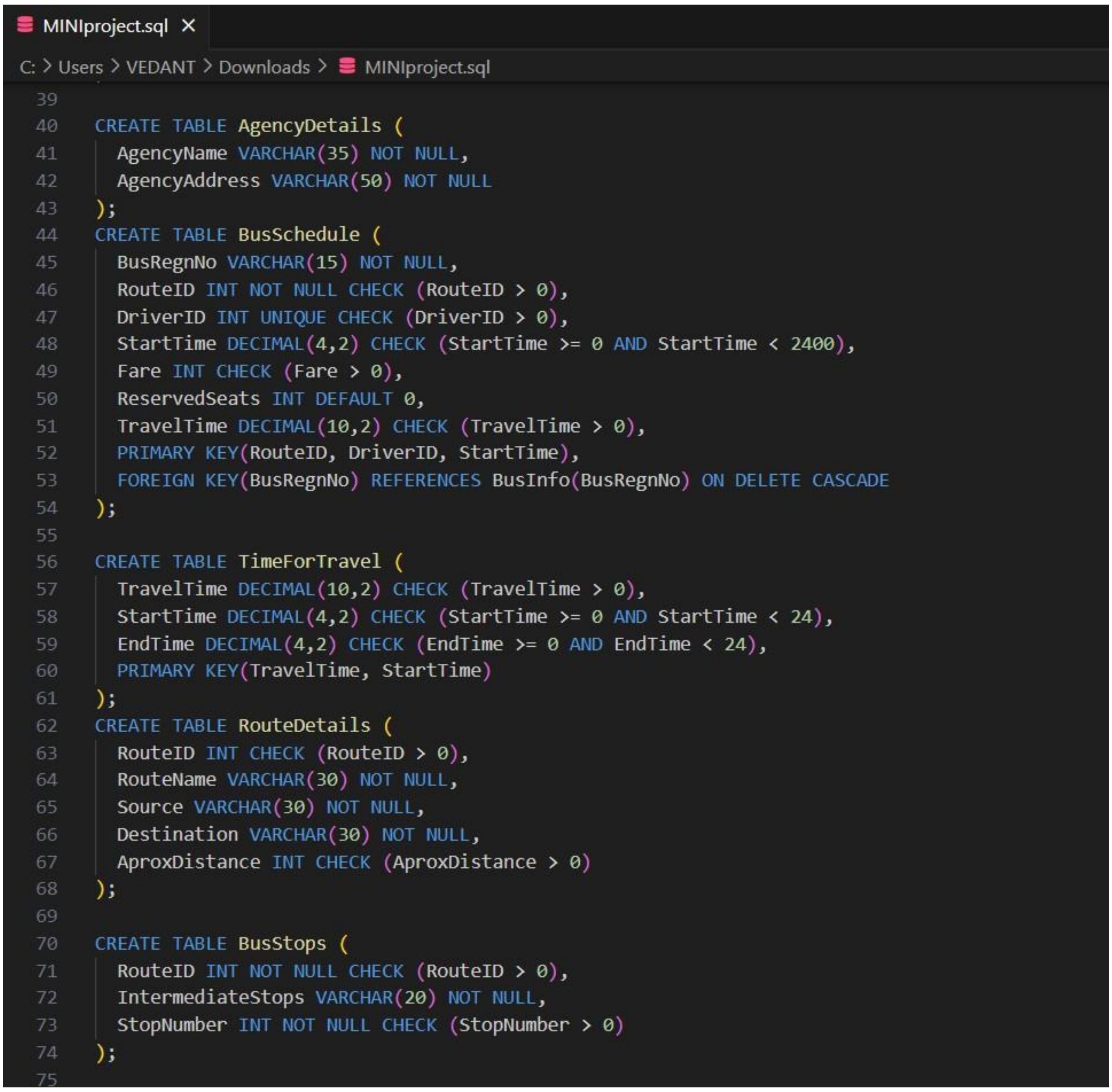
SQL Input Codes:



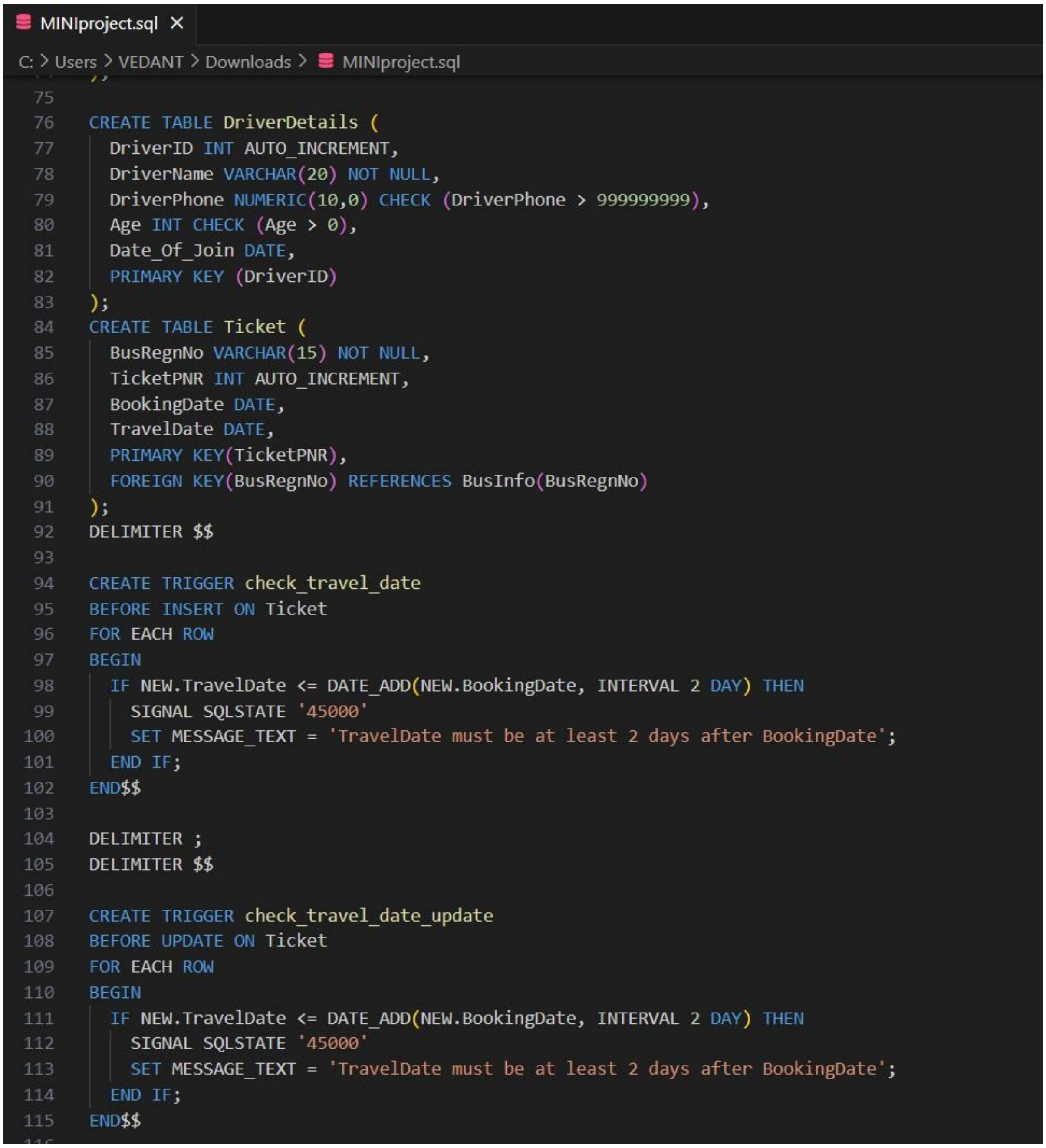
MINIproject.sql ×

C: > Users > VEDANT > Downloads > MINIproject.sql

```sql
CREATE DATABASE dbms1;
USE dbms1;
CREATE TABLE UserTable (
  ID INT AUTO_INCREMENT,
  Email VARCHAR(40) NOT NULL UNIQUE,
  Name VARCHAR(30),
  Password VARCHAR(15) NOT NULL,
  Usertype VARCHAR(10) NOT NULL,
  PRIMARY KEY (ID)
);
CREATE TABLE NonAdmin (
  ID INT,
  Gender VARCHAR(7),
  Phone NUMERIC(10,0) NOT NULL UNIQUE CHECK (Phone > 999999999),
  Address VARCHAR(100),
  PRIMARY KEY(ID),
  FOREIGN KEY(ID) REFERENCES UserTable(ID) ON DELETE CASCADE
);

CREATE TABLE Admin (
  ID INT,
  AgencyName VARCHAR(35) NOT NULL,
  AgencyPhone VARCHAR(10) NOT NULL UNIQUE CHECK (AgencyPhone > 999999999),
  AgencyOffice VARCHAR(50),
  PRIMARY KEY (AgencyName),
  FOREIGN KEY(ID) REFERENCES UserTable(ID) ON DELETE CASCADE
);
CREATE TABLE BusInfo (
  BusRegnNo VARCHAR(15) NOT NULL,
  AgencyName VARCHAR(35) NOT NULL,
  TotalSeats INT DEFAULT 40,
  AC TINYINT DEFAULT 0,
  LocationName VARCHAR(20),
  Latitude DECIMAL(17,10),
  Longitude DECIMAL(17,10),
  PRIMARY KEY (BusRegnNo),
  FOREIGN KEY (AgencyName) REFERENCES Admin(AgencyName) ON DELETE CASCADE
);
```

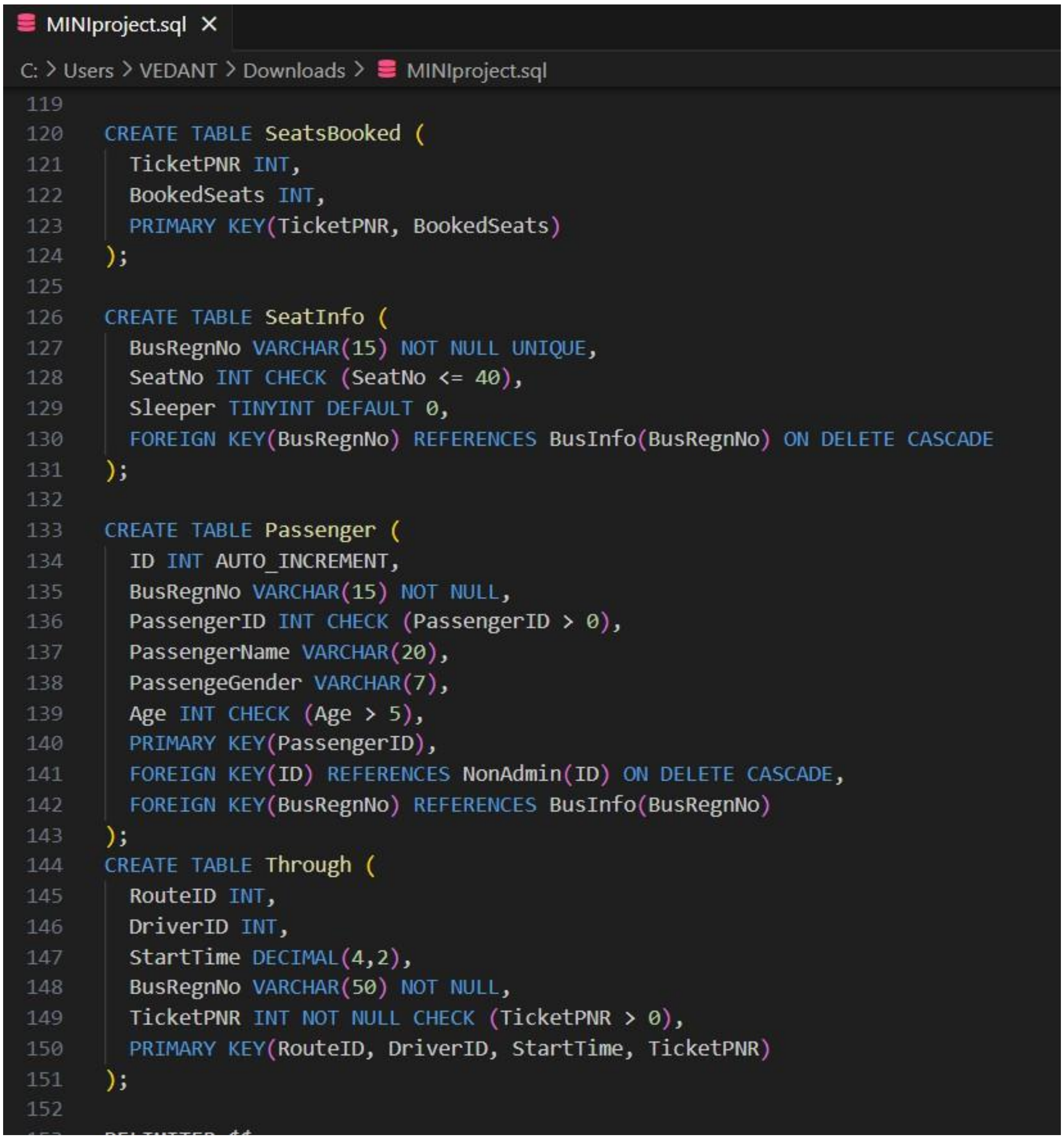


```sql

CREATE TABLE AgencyDetails (
  AgencyName VARCHAR(35) NOT NULL,
  AgencyAddress VARCHAR(50) NOT NULL
);
CREATE TABLE BusSchedule (
  BusRegnNo VARCHAR(15) NOT NULL,
  RouteID INT NOT NULL CHECK (RouteID > 0),
  DriverID INT UNIQUE CHECK (DriverID > 0),
  StartTime DECIMAL(4,2) CHECK (StartTime >= 0 AND StartTime < 2400),
  Fare INT CHECK (Fare > 0),
  ReservedSeats INT DEFAULT 0,
  TravelTime DECIMAL(10,2) CHECK (TravelTime > 0),
  PRIMARY KEY(RouteID, DriverID, StartTime),
  FOREIGN KEY(BusRegnNo) REFERENCES BusInfo(BusRegnNo) ON DELETE CASCADE
);

CREATE TABLE TimeForTravel (
  TravelTime DECIMAL(10,2) CHECK (TravelTime > 0),
  StartTime DECIMAL(4,2) CHECK (StartTime >= 0 AND StartTime < 24),
  EndTime DECIMAL(4,2) CHECK (EndTime >= 0 AND EndTime < 24),
  PRIMARY KEY(TravelTime, StartTime)
);
CREATE TABLE RouteDetails (
  RouteID INT CHECK (RouteID > 0),
  RouteName VARCHAR(30) NOT NULL,
  Source VARCHAR(30) NOT NULL,
  Destination VARCHAR(30) NOT NULL,
  AproxDistance INT CHECK (AproxDistance > 0)
);

CREATE TABLE BusStops (
  RouteID INT NOT NULL CHECK (RouteID > 0),
  IntermediateStops VARCHAR(20) NOT NULL,
  StopNumber INT NOT NULL CHECK (StopNumber > 0)
);

```


MINIproject.sql ×

C: > Users > VEDANT > Downloads > MINIproject.sql

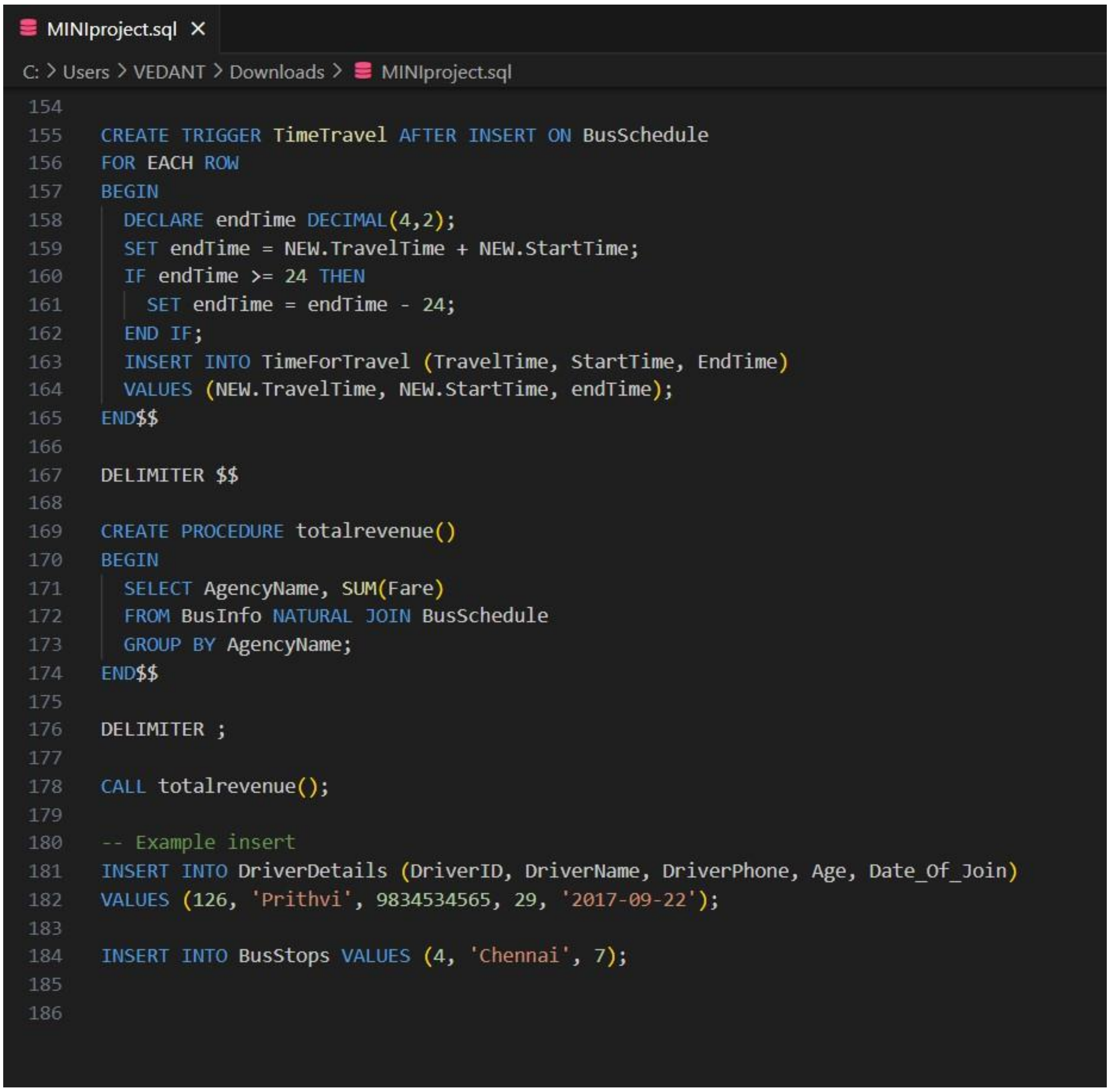
```sql
CREATE TABLE DriverDetails (
  DriverID INT AUTO_INCREMENT,
  DriverName VARCHAR(20) NOT NULL,
  DriverPhone NUMERIC(10,0) CHECK (DriverPhone > 999999999),
  Age INT CHECK (Age > 0),
  Date_Of_Join DATE,
  PRIMARY KEY (DriverID)
);
CREATE TABLE Ticket (
  BusRegnNo VARCHAR(15) NOT NULL,
  TicketPNR INT AUTO_INCREMENT,
  BookingDate DATE,
  TravelDate DATE,
  PRIMARY KEY(TicketPNR),
  FOREIGN KEY(BusRegnNo) REFERENCES BusInfo(BusRegnNo)
);
DELIMITER $$

CREATE TRIGGER check_travel_date
BEFORE INSERT ON Ticket
FOR EACH ROW
BEGIN
  IF NEW.TravelDate <= DATE_ADD(NEW.BookingDate, INTERVAL 2 DAY) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'TravelDate must be at least 2 days after BookingDate';
  END IF;
END$$

DELIMITER ;
DELIMITER $$

CREATE TRIGGER check_travel_date_update
BEFORE UPDATE ON Ticket
FOR EACH ROW
BEGIN
  IF NEW.TravelDate <= DATE_ADD(NEW.BookingDate, INTERVAL 2 DAY) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'TravelDate must be at least 2 days after BookingDate';
  END IF;
END$$
```



MINIproject.sql ×

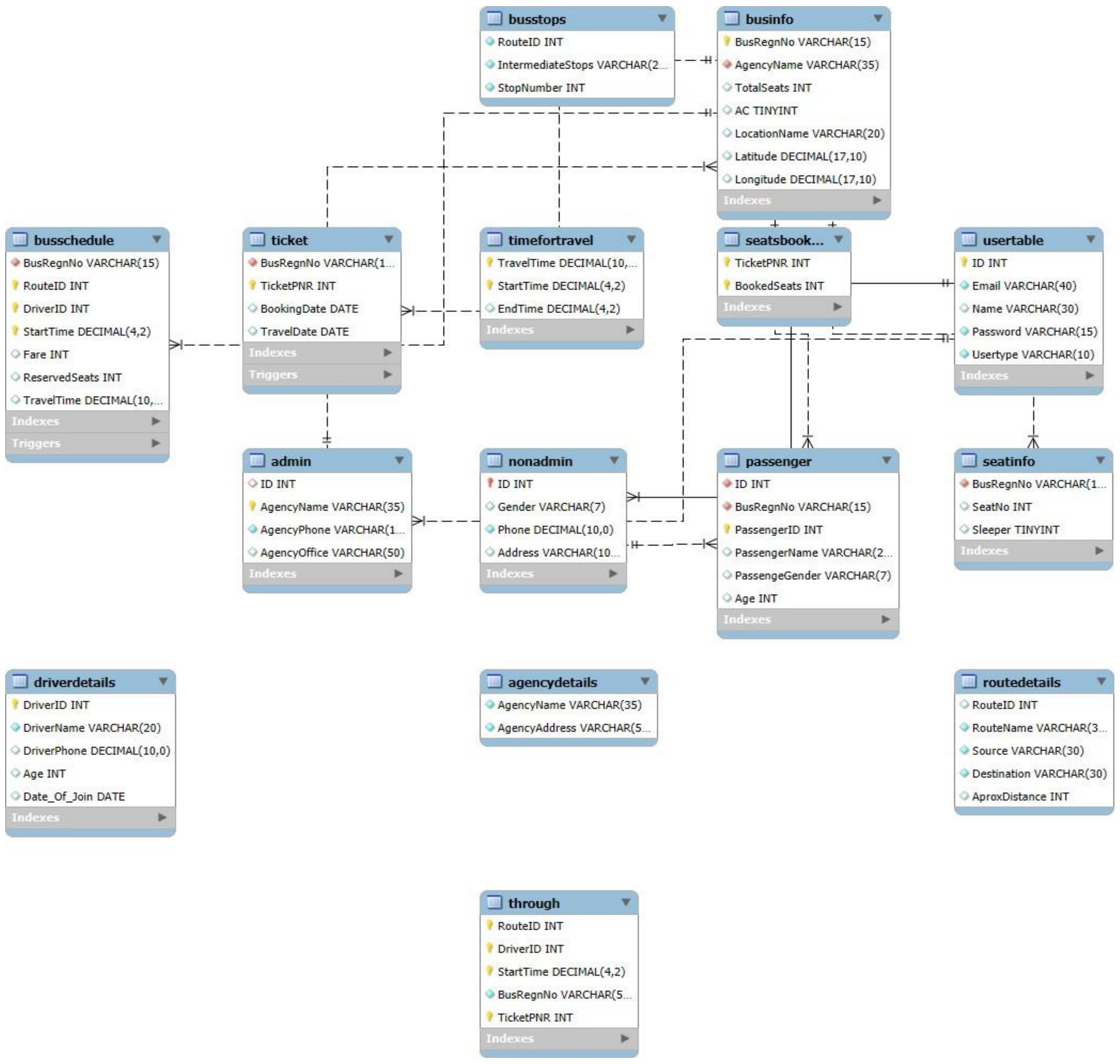
C: > Users > VEDANT > Downloads > MINIproject.sql

```sql

CREATE TABLE SeatsBooked (
  TicketPNR INT,
  BookedSeats INT,
  PRIMARY KEY(TicketPNR, BookedSeats)
);

CREATE TABLE SeatInfo (
  BusRegnNo VARCHAR(15) NOT NULL UNIQUE,
  SeatNo INT CHECK (SeatNo <= 40),
  Sleeper TINYINT DEFAULT 0,
  FOREIGN KEY(BusRegnNo) REFERENCES BusInfo(BusRegnNo) ON DELETE CASCADE
);

CREATE TABLE Passenger (
  ID INT AUTO_INCREMENT,
  BusRegnNo VARCHAR(15) NOT NULL,
  PassengerID INT CHECK (PassengerID > 0),
  PassengerName VARCHAR(20),
  PassengeGender VARCHAR(7),
  Age INT CHECK (Age > 5),
  PRIMARY KEY(PassengerID),
  FOREIGN KEY(ID) REFERENCES NonAdmin(ID) ON DELETE CASCADE,
  FOREIGN KEY(BusRegnNo) REFERENCES BusInfo(BusRegnNo)
);
CREATE TABLE Through (
  RouteID INT,
  DriverID INT,
  StartTime DECIMAL(4,2),
  BusRegnNo VARCHAR(50) NOT NULL,
  TicketPNR INT NOT NULL CHECK (TicketPNR > 0),
  PRIMARY KEY(RouteID, DriverID, StartTime, TicketPNR)
);

```



```sql

CREATE TRIGGER TimeTravel AFTER INSERT ON BusSchedule
FOR EACH ROW
BEGIN
  DECLARE endTime DECIMAL(4,2);
  SET endTime = NEW.TravelTime + NEW.StartTime;
  IF endTime >= 24 THEN
    SET endTime = endTime - 24;
  END IF;
  INSERT INTO TimeForTravel (TravelTime, StartTime, EndTime)
  VALUES (NEW.TravelTime, NEW.StartTime, endTime);
END$$

DELIMITER $$

CREATE PROCEDURE totalrevenue()
BEGIN
  SELECT AgencyName, SUM(Fare)
  FROM BusInfo NATURAL JOIN BusSchedule
  GROUP BY AgencyName;
END$$

DELIMITER ;

CALL totalrevenue();

-- Example insert
INSERT INTO DriverDetails (DriverID, DriverName, DriverPhone, Age, Date_Of_Join)
VALUES (126, 'Prithvi', 9834534565, 29, '2017-09-22');

INSERT INTO BusStops VALUES (4, 'Chennai', 7);
```

Entity-Relationship Diagram (ERD):



Database Design (Schema)

Key DBMS Concepts Used:

- Normalization (up to 3NF to avoid redundancy)
- Primary and Foreign Keys
- 1-to-Many and Many-to-One Relationships
- Views (for filtered transport listings)
- Triggers (for logging updates)
- Stored Procedures (for booking & cancellation)

- Joins (for combining route, schedule, and booking info)

Entities:

- User (UserID, Name, Email, Password, Type)
- Admin (AdminID, Name, Email, Password)
- Transport (TransportID, Type, Name)
- Route (RouteID, Source, Destination, Distance)
- Schedule (ScheduleID, TransportID, RouteID, DepartureTime, ArrivalTime, Fare)
- Booking (BookingID, UserID, ScheduleID, BookingDate, Status)

Relationships:

- One Route → Many Schedules (1:M)
- One User → Many Bookings (1:M)
- One Transport → Many Schedules (1:M)
- One Schedule → Many Bookings (1:M)

Voice Integration Flow

1. User speaks: "Show buses from Pune to Mumbai."
2. Speech is converted to text via **SpeechRecognition**.
3. Query is parsed and executed on MySQL.
4. Result is converted to speech using **pyttsx3** and also shown on screen.
5. Booking/cancellation can be triggered by voice: "Book ticket from Pune to Mumbai at 9 AM."

Benefits

- Simple and easy to use interface
- Voice enabled database updation and retrieval
- No necessity of a human resource
- Software availability round the clock
- Ability to select the best possible route to optimize cost and time.

- Supports hands-free interaction.
- Reduces staffing needs at enquiry counters.
- Works 24x7 without supervision.
- Can be extended for multilingual support.
- Improves accessibility for differently abled users.

Future Enhancements

- Integration with Google Maps for route visualization.
- Multi-language voice support.
- OTP verification for booking confirmation.
- Notification via SMS or email.
- Mobile app version for broader access.

Conclusion

The Voice-Based Transport Enquiry System simplifies access to public transport information by using voice commands and a robust relational database. It enhances user experience by eliminating the need for manual inputs, thus providing a smarter, faster, and more accessible way of travel planning. This DBMS project demonstrates how database principles can be combined with modern voice technology to build real-time, user-friendly solutions.