**VOICE BASED AUTOMATED TRANSPORT ENQUIRY SYSTEM**

**GROUP 16**

Adarsh T – CB.EN.U4CSE19404

Balamurugan B – CB.EN.U4CSE19414

M.Bhavya Sree – CB.EN.U4CSE19438

Oviya.H – CB.EN.U4CSE19445

**ABSTRACT:**

Voice Based Automated Transport Enquiry System is the enquiry system which works based on the voice input given by the user. This system takes voice commands as input and gives information in form of voice and displays relevant information on the screen. This system can be installed in any transport terminals like bus stop, railway terminals or airports.

We have experienced waiting in transport terminal to get tickets or to get some information about the transport facility but most of the time we realize that there will be no staff for providing these information significantly.

Here is a solution for this problem which does not need human intervention and less maintenance required in the transport terminals. Voice based automated transport enquiry system is developed for providing such information in the transport terminals.

The commands are given to the system in voice commands and they are processed using speech recognition and then the system shall search the appropriate result according to the command asked. The system will display the result on the screen and the user can perform some operations like changing the query or moving from previous to next results for a query or user can stop the current query from continuing and can also ask multiple queries.

**MODULES:**

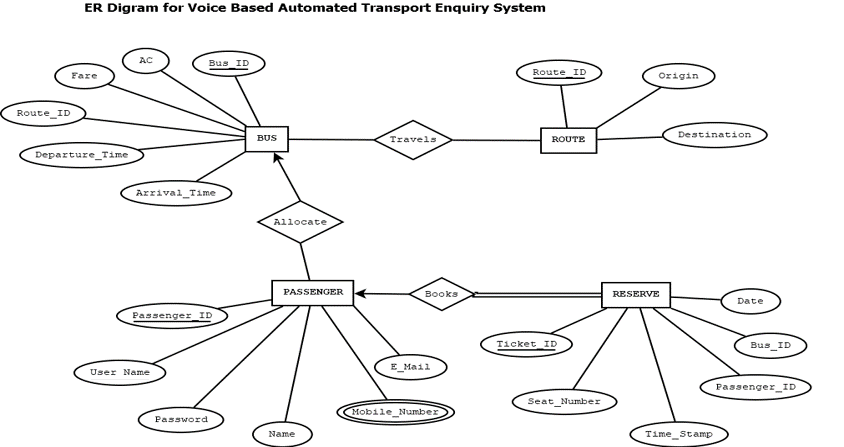
* Billing module for ticket booking
* Bus details and bus route details
* Voice to text conversion and text to voice conversion
* Passenger details
* Admin module
* User module

This system helps in travel planning and saves time. Only little human intervention is needed.

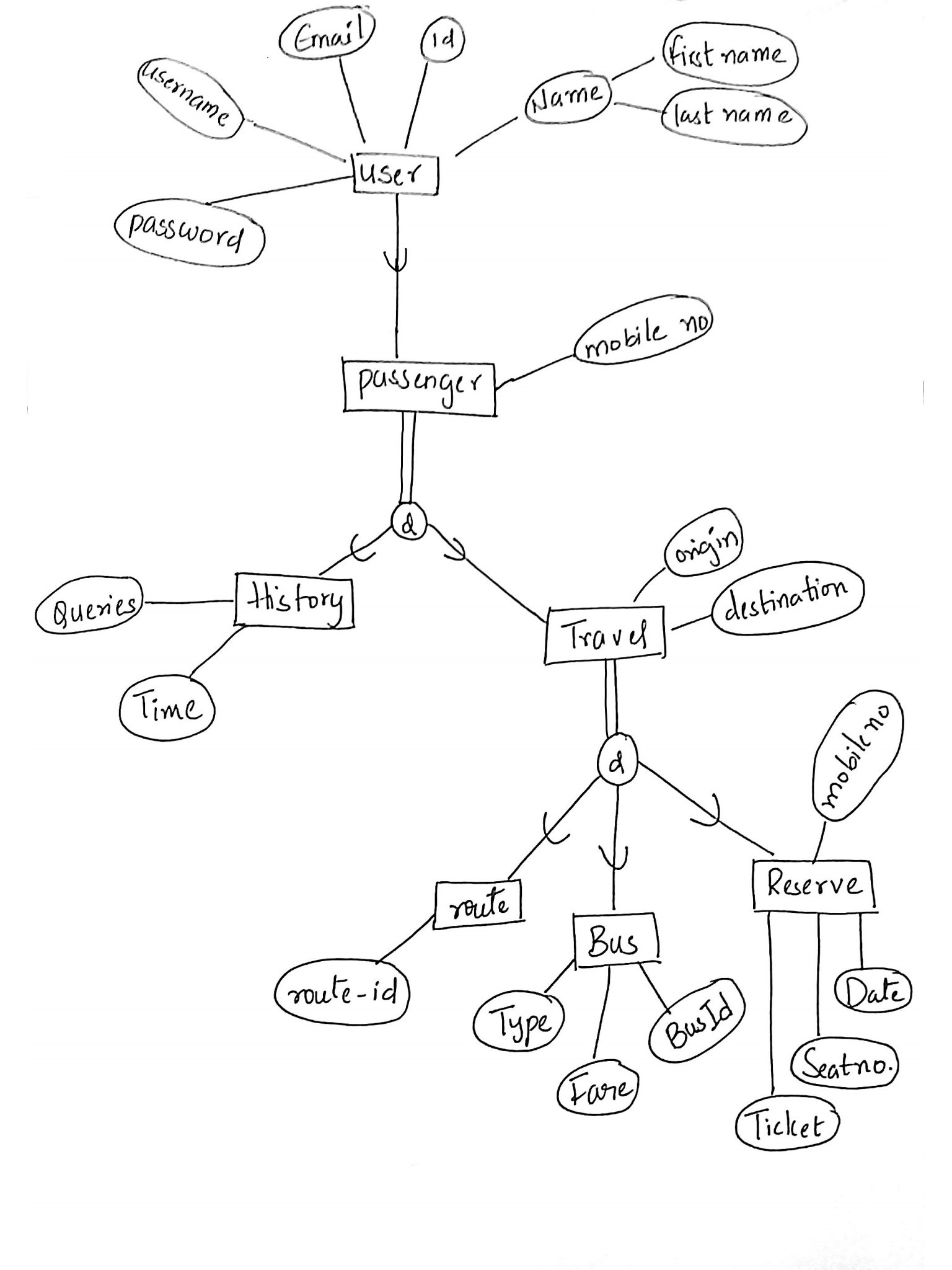
**Tools Used:**

* tkinter ( Python library ) - Front End
* MySQL
* gtts ( Python library ) Back End
* vlc ( Python library )

**ENITITY RELATIONSHIP DIAGRAM:**

****

**EXTENDED ER DIAGRAM:**



**NORMALIZED SCHEMA:**

**BUS Table** (Bus\_id, Type, Fare)

**Route Table** (Route\_id, Origin, Destination)

**BusRoute Table** (Bus\_id, Route\_id, Dept\_time, Arrival\_time)

**Ticket Table** (Ticket\_id, Passenger\_id, Seat\_no, Bus\_id, Date\_of\_travel)

**Passenger Table** (Passenger\_id, Name, Mobile\_no, Email)

**NORMALIZATION:**

**MASTER TABLE:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bus\_id | Type | Fare | Route\_id | Dept\_time | Arrival\_time | | Origin | Destination | Ticket\_Id |
| Passenger\_id | Seat\_no | Date\_of\_travel | Name | Mobile\_no | Email |

**Functional Dependencies:**

Bus\_id 🡪Type, Fare

Bus\_id, Route\_id🡪Dept\_time, Arrival\_time

Route\_id🡪Origin, Destination

Ticket\_id🡪Seat\_no, Passenger\_id, Bus\_id, Date\_of\_travel, Name, Mobile\_no, Email

Passenger\_id🡪Name, Mobile\_no, Email

**Closure:**

(Bus\_id) + = {Bus\_id, Type, Fare}

(Route\_id)+= {Origin, Destination, Route\_id}

(Bus\_id, Route\_id) += {Bus\_id, Type, Fare, Origin, Destination, Route\_id, Dept\_time, Arrival\_time}

(Ticket\_id) + = {Seat\_no, Passenger\_ID, Bus\_id, Date\_of\_travel, Ticket\_id, Type, Fare, Name, Mobile\_no, email}

(Passenger\_id) += {Passenger\_id, Name, Mobile\_no, Email}

**Candidate Key:**

{Bus\_id, Route\_id, Ticket\_id}

**1NF:**

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values. Hence the MASTER TABLE is in **1NF.**

**2NF:**

Table is said to be in 2NF if both the following conditions hold:

* Table is in 1NF (First normal form)
* No Partial Dependency

For the master table candidate key is a composite key:

{Bus\_id, Route\_id, Ticket\_id}

Bus\_id 🡪 Type, Fare 🡪 partial dependency

Bus\_id, Route\_id🡪 Dept\_time, Arrival\_time 🡪 partial dependency

Route\_id🡪 Origin, Destination 🡪 partial dependency

Ticket\_id🡪 Seat\_no, Passenger\_id, Bus\_id, Date\_of\_travel, Name, Mobile\_no, Email 🡪 partial dependency

**MASTER TABLE:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bus\_id | Type | Fare | Route\_id | Dept\_time | Arrival\_time | | Origin | Destination | Ticket\_Id |
| Passenger\_id | Seat\_no | Date\_of\_travel | Name | Mobile\_no | Email |

**Bus table:**

|  |  |  |
| --- | --- | --- |
| Bus\_id | Type | Fare |

**FD:** Bus\_id 🡪 Type, Fare

**Route table:**

|  |  |  |
| --- | --- | --- |
| Route\_id | Origin | Destination |

**FD:** Route\_id🡪 Origin, Destination

**Bus-Route table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Bus\_id | Route\_id | Dept\_time | Arrival\_time |

**FD:** Bus\_id, Route\_id🡪 Dept\_time, Arrival\_time

**Ticket table:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ticket\_id | Passenger\_id | Seat\_no | Bus\_id | Date\_of\_travel | Name | Mobile\_no | Email |

**FD:** Ticket\_id🡪Seat\_no, Passenger\_id, Bus\_id, Date\_of\_travel, Name, Mobile\_no, Email

**3NF:**

A table design is said to be in 3NF if both the following conditions hold:

* Table must be in 2NF
* Transitive functional dependency of non-prime attribute on any super key should be removed.

Ticket\_id🡪Seat\_no, Passenger\_id, Bus\_id, Date\_of\_travel, Name, Mobile\_no, Email

Passenger\_id🡪Name, Mobile\_no, Email

This is transitive in nature.

**Ticket table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ticket\_id | Passenger\_id | Seat\_no | Bus\_id | Date\_of\_travel |

**FD:** Ticket\_id🡪Seat\_no, Passenger\_id, Bus\_id, Date\_of\_travel

**Passenger table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Passenger\_id | Name | Mobile\_no | Email |

**FD:** Passenger\_id🡪Name, Mobile\_no, Email

Hence the there is no transitive dependencies and the tables are in 3NF.

The tables after 3NF normalization are:

**Bus table:**

|  |  |  |
| --- | --- | --- |
| Bus\_id | Type | Fare |

**Route table:**

|  |  |  |
| --- | --- | --- |
| Route\_id | Origin | Destination |

**Bus-Route table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Bus\_id | Route\_id | Dept\_time | Arrival\_time |

**Ticket table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ticket\_id | Passenger\_id | Seat\_no | Bus\_id | Date\_of\_travel |

**Passenger table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Passenger\_id | Name | Mobile\_no | Email |

**TABLE CREATION:**

CREATE DATABASE test;

Create table test.bus(

Bus\_id numeric(4),

Type varchar(10),

Fare numeric(4),

Primary key (Bus\_id));

Create table test.route(

Route\_id varchar(3),

Origin varchar(30),

Destination varchar(30),

Primary key(route\_id));

Create table test.busroute(

Bus\_id numeric(4),

Route\_id varchar(3),

Dept\_time varchar(5),

arrival\_time varchar(5),

Primary key(Bus\_id,Route\_id),

Foreign key(Bus\_id) references bus(Bus\_id),

Foreign key(Route\_id) references route(Route\_id));

Create table test.passenger(

Passenger\_id numeric(5),

Name varchar(30),

Mobile\_no varchar(10),

Email varchar(50),

Primary key(passenger\_id));

Create table test.ticket(

Ticket\_id varchar(5),

Seat\_no varchar (3),

Passenger\_id numeric(5),

Bus\_id numeric(4),

Date\_of\_travel date,

Primary key(ticket\_id),

Foreign key (passenger\_id) references passenger(passenger\_id),

Foreign key(Bus\_id) references bus(Bus\_id));

Insert into test.bus (Bus\_id,Type,Fare)

values

(1025,'AC',10),

(1432,'non-AC',10),

(1524,'AC',20),

(1743, 'non- AC',30),

(1824,'AC',50),

(1924,'non-AC',15),

(1372,'AC',20),

(1963,'AC',40);

Insert into test.route (Route\_id,Origin,Destination)

values

('4BE','Moscow','omen'),

('9F2','Nairobi','Toledo'),

('3Z5','Oslo','Penza'),

('4M3','Helsinki','Zimbabwe'),

('5O7','Tokyo','nartkala');

select \* from test.busroute;

Insert into test.busroute (Bus\_id,Route\_id,Dept\_time,arrival\_time)

values

(1025,'4BE','12:40','14:30'),

(1432,'9F2','13:40','16:50'),

(1524,'3Z5','9:00','12:00'),

(1743,'4BE','6:00','13:00'),

(1824,'9F2','8:00','14:00'),

(1924,'3Z5','18:00','19:00'),

(1372,'4M3','20:00','2:00'),

(1963,'5O7','23:00','5:00');

Insert into test.passenger (Passenger\_id,Name,Mobile\_no,Email)

values

(19432,'Justin','9876543210','Justin@gmail.com'),

(17245,'Hannah','7325197210','Hannah@gmail.com'),

(25735,'Mille','6532107210','Millie@gmail.com'),

(72590,'Arthuro','8332969286','arthuro@gmail.com'),

(83792,'Raquel','9998472206','Raquel@gmail.com'),

(94132,'Sergio','9432172341','seirgio@gmail.com'),

(65431,'Berlin','7397210812','Berlin@gmail.com'),

(57321,'Monica','9345810873','Monica@gmail.com');

select \* from test.route;

select \* from test.bus;

select \* from test.busroute;

select \* from test.passenger;

select \* from test.ticket;

Insert into test.ticket (Ticket\_id,Seat\_no,Passenger\_id,Bus\_id,Date\_of\_travel)

values

('A0001', '1A', 19432, 1025, '12-09-21'),

('A0002', '1B', 17245, 1432, '22-11-21'),

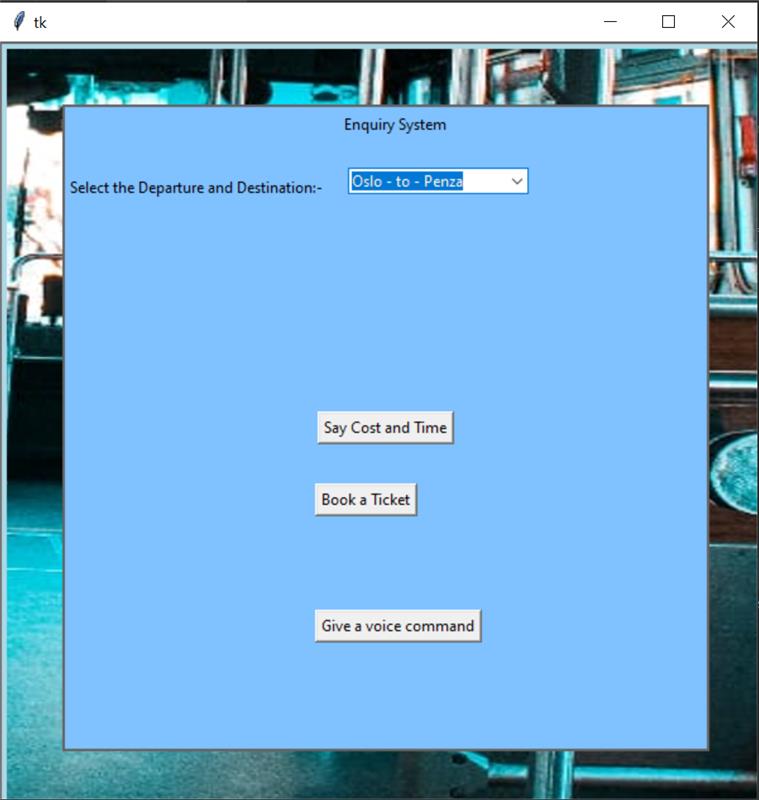
('A0003', '1C', 25735, 1524, '04-11-21'),

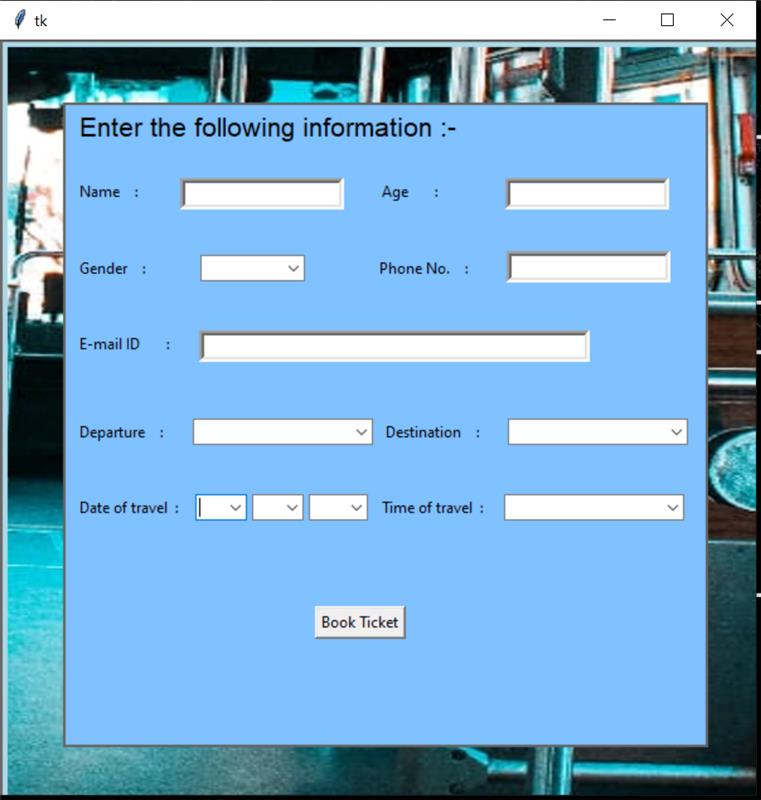
('A0004', '1E', 72590, 1743, '08-10-21'),

('A0005', '2D', 83792, 1824, '09-06-21'),

('A0006', '2A', 94132, 1924, '15-01-21');

**UI SCREENSHOTS:**

****

****

**Connectivity Code:**

import MySQLdb

db = MySQLdb.connect(host='localhost',

user='root',

passwd='balas11c1',

db='test')

cur = db.cursor()

cur.execute('select \* from test.route')

**Sample UI Code:**

def OnCreate(self):

from tkinter import Canvas, Label, StringVar

from tkinter import ttk, Button

import MySQLdb

from gtts import gTTS

from gtts.tokenizer import pre\_processors

import vlc

bg = "#80c1ff"

db = MySQLdb.connect(host='localhost',

user='root',

passwd='balas11c1',

db='test')

cur = db.cursor()

cur2 = db.cursor()

cur3 = db.cursor()

cur.execute('select \* from test.route')

cur2.execute('select \* from test.busroute')

cur3.execute('select \* from test.bus')

origin = []

route\_id = []

dest = {}

arr = {}

for i in cur.fetchall():

route\_id.append(i[0])

arr[i[2]] = i[0]

dest[i[1]] = i[0]

origin.append(i[1] + " - to - " + i[2])

bus\_id = {}

r\_time = {}

route\_id2 = []

time = []

for j in cur2.fetchall():

route\_id2.append(j[1])

bus\_id[j[1]] = j[0]

time.append(j[2])

r\_time[j[0]] = j[2]

fare = {}

b\_id2 = []

for k in cur3.fetchall():

b\_id2.append(k[0])

fare[k[0]] = k[2]

def speech(text, x):

try:

output = gTTS(text, lang='en', slow=False)

try:

output.save("speech.mp3")

try:

self.output.destroy()

except AttributeError:

pass

self.output = Label(self.Frame, bg=bg, text=x)

self.output.place(x=dept\_label.winfo\_reqwidth() + 20, y=departure.winfo\_reqheight() + 70)

media = vlc.MediaPlayer("speech.mp3")

media.play()

except ValueError:

speech(text, x)

except AssertionError:

print('Enter a text to get the speech')

w.Welcome(self.welcomeNote)

self.Parent.configure(height=600, width=600)

from PIL import Image, ImageTk

img = Image.open("C:/Users/Hp/Downloads/bus.jpeg")

img = img.resize((1500, 1500))

self.Image = ImageTk.PhotoImage(img)

self.view = Canvas(self.Parent, width=600, height=600, bd=4, bg='lightblue')

self.view.create\_image(0, 0, image=self.Image)

self.view.place(x=self.Pos["X"], y=self.Pos["Y"])

self.Frame = Canvas(self.view, width=500, height=500, cursor="circle", bd=4, bg=bg)

self.Frame.place(x=self.C\_pos["X"], y=self.C\_pos["Y"])

heading = Label(self.Frame, bg=bg, text="Enquiry System")

heading.place(x=220, y=5)

dept\_label = Label(self.Frame, bg=bg, text="Select the Departure and Destination:-")

dept\_label.place(x=3, y=55)

dept\_var = StringVar()

departure = ttk.Combobox(self.Frame, width=20, textvariable=dept\_var)

departure.place(x=dept\_label.winfo\_reqwidth() + 20, y=50)

departure['values'] = tuple(origin)

def on\_click():

r\_id = None

b\_id = None

b\_t = None

self.place = dept\_var.get()

val = self.place.split()

if val[0] in dest.keys():

r\_id = dest.get(val[0])

if r\_id in bus\_id.keys():

b\_id = bus\_id.get(r\_id)

if b\_id in fare.keys():

cost = fare.get(b\_id)

if b\_id in r\_time.keys():

out2 = r\_time.get(b\_id)

out = f'''Total fair from {self.place}: {cost} $

Bus arrival time {out2}

'''

last = self.place.split()

self.arrival\_out = last[0]

self.destination\_out = last[4]

self.timing = out2

input\_text = out

text = pre\_processors.abbreviations(input\_text)

text = pre\_processors.word\_sub(input\_text)

from \_thread import start\_new\_thread

start\_new\_thread(speech, (text, out))

submit = Button(self.Frame, text='Say Cost and Time', command=on\_click)

submit.place(x=self.Frame.winfo\_reqwidth() / 2, y=self.Frame.winfo\_reqheight() / 2, anchor="center")

def Next():

w.Destroy()

w.SecondWindow()

book\_ticket = Button(self.Frame, text='Book a Ticket', command=Next)

book\_ticket.place(x=200, y=300)

**CONCLUSION:**

Our project "Voice based automatic transportation system enquiry system" is very helpful for the public to know the details and to book tickets. We have designed our app in such a way that it takes the input in voice and gives the required information in both voice and text. We have done voice input using listener functionality and speechrecogonition library which converts input voice into text. we have used gtts library to convert text to voice. People will feel very comfortable to use this kind of enquiry system.

**REFEERENCES:**

* <https://www.tutorialspoint.com/python/python_gui_programming.htm>
* <https://pypi.org/>
* <https://www.geeksforgeeks.org/python-convert-speech-to-text-and-text-to-speech/>
* <https://www.geeksforgeeks.org/convert-text-speech-python/#:~:text=There%20are%20several%20APIs%20available,saved%20as%20a%20mp3%20file.>