**Project Report**

**On**

**“Flight Reservation System”**

Submitted in the Partial fulfillment of the requirement for the Award of Degree of

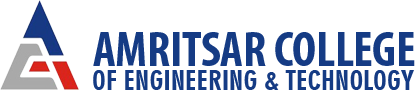
**Bachelor of Technology**

in

**COMPUTER SCIENCE & ENGINEERING**

Batch

(2021-2025)



|  |  |
| --- | --- |
| **Submitted to:**  Er. Ajay Sharma (Associate Professor) | **Submitted by:**  Anjar Hasan – 2131816  Aadil Hasan - 2131795 Ankit Kr. Singh - 2131818  Ankit kr. Mishra - 2131819 |

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Amritsar Group of Colleges , Amritsar**

**(Autonomous college under UGC Act – 1956[2(f) and 12(B)] )**

# ACKNOWLEDGEMENT

It is our proud privilege to release the feelings of our gratitude to several persons who helped us directly or indirectly to conduct this Analysis Project Work. We express our heart full thanks and owe a deep sense of gratitude to our teacher and my faculty guide **Er. Ajay Sharma,** for their guidance and inspiration in completing this project.

We are extremely thankful to the **Dr. Sandeep Kad, Head of Department** and all faculty members of CSE Department at **Amritsar Group of Colleges, Amritsar** for their co-operation, kind guidance and encouragement.

We also thank all our friends who have more or less contributed to the preparation of this Project Report, we will be always indebted to them.

This project completion has indeed helped us explore more knowledge avenues related to RDBMS/Python and we are sure it will help us in future too.

# DECLARATION

We hereby as a team declare that the project work entitled **“FLIGHT RESERVATION SYSTEM”** is an authentic record of our own work carried out as per the requirements of RELATIONAL DATABASE MANAGEMENT SYSTEMS (ACCS-21402)/PROGRAMMING

IN PYTHON (ACCS-21403) Labs for the award of degree of **B.Tech (CSE)**, **Amritsar Group of Colleges, Amritsar**, under the guidance of **Er. Ajay Sharma (Associate Professor)**

Anjar Hasan - 2131816

Aadil Hasan - 2131795

Ankit Kr. Singh – 2131818

Ankit kr. Mishra - 2131819

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Content** | **Page No.** |
| 1. | Introduction to RDBMS | 5-9 |
| 2. | Introduction to Python | 10-16 |
| 3. | Introduction to Project | 17 |
| 4. | Abstract | 18 |
| 5. | Objectives of the Project | 19-22 |
| 6. | Tables in Database | 20-22 |
| 7. | Introduction to Oracle | 23-26 |
| 8. | Introduction to tkinter | 27-28 |
| 9. | Source Code | 29-49 |
| 10. | Snapshots | 50-54 |
| 11. | Refrences | 55 |
| 12. | Conclusion | 56 |

**INTRODUCTION TO RDBMS**

**RDBMS** stands for *Relational Database Management System.*

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS.

It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

# How it works

Data is represented in terms of tuples (rows) in RDBMS.

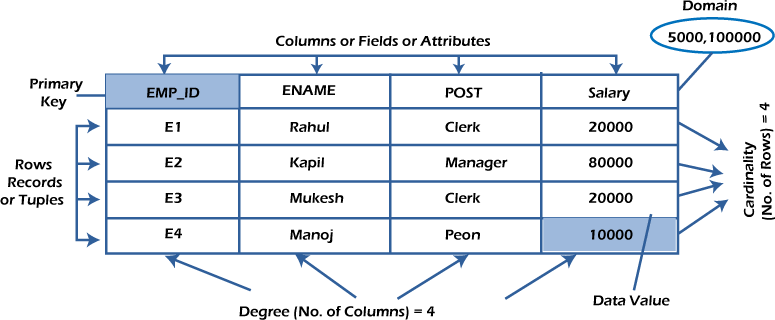
A relational database is the most commonly used database. It contains several tables, and each table has its primary key.

Due to a collection of an organized set of tables, data can be accessed easily in RDBMS.

# Brief History of RDBMS

From 1970 to 1972, E.F. Codd published a paper to propose using a relational database model. RDBMS is originally based on E.F. Codd's relational model invention.

**Following are the various terminologies of RDBMS:**



# What is table/Relation?

Everything in a relational database is stored in the form of relations. The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data. Each table represents some real-world objects such as person, place, or event about which information is collected.

The organized collection of data into a relational table is known as the logical view of the database.

# Properties of a Relation:

* Each relation has a unique name by which it is identified in the database. o

Relation does not contain duplicate tuples.

* The tuples of a relation have no specific order.
* All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

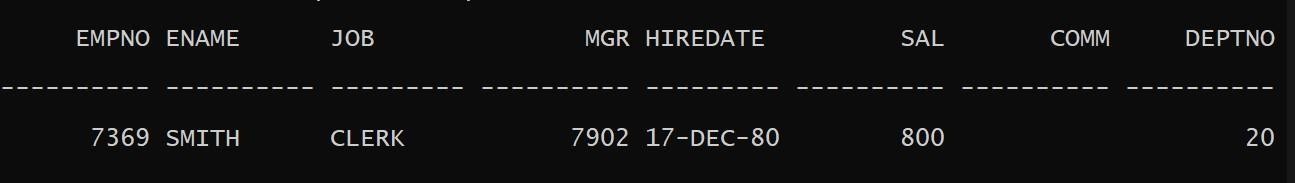
A table is the simplest example of data stored in RDBMS.

## Let's see the example of the Emp table.



**What is a row or record?**

A row of a table is also called a record or tuple. It contains the specific information of each entry in the table. It is a horizontal entity in the table. For example, The above emp table contains 14 records.



## Properties of a row:

o No two tuples are identical to each other in all their entries.

* All tuples of the relation have the same format and the same number of entries.
* The order of the tuple is irrelevant. They are identified by their content, not by their position.

# What is a column/attribute?

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example, "name" is a column in the above table which contains all information about a student's name.

## Properties of an Attribute:

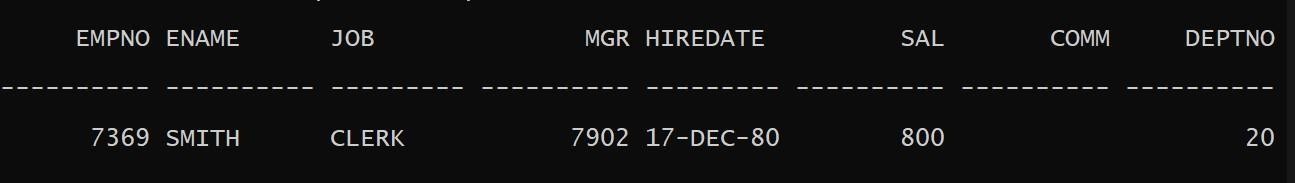
* Every attribute of a relation must have a name. o Null values are permitted for the attributes.
* Default values can be specified for an attribute automatically inserted if no other valueis specified for an attribute.
* Attributes that uniquely identify each tuple of a relation are the primary key.

# What is data item/Cells?\

The smallest unit of data in the table is the individual data item. It is stored at the intersection of tuples and attributes.

**Properties of data items:**

* Data items are atomic.
* The data items for an attribute should be drawn from the same domain.



## Degree:

The total number of attributes that comprise a relation is known as the degree of the table.

**For example, the above emp table has 8 attributes, and its degree is 8.**

## Cardinality:

The total number of tuples at any one time in a relation is known as the table's cardinality. The relation whose cardinality is 0 is called an empty table.

**For example, the above emp table has 14 rows, and its cardinality is 14.**

## Domain:

The domain refers to the possible values each attribute can contain. It can be specified using standard data types such as integers, floating numbers, etc. **For example**, An attribute entitled Marital\_Status may be limited to married or unmarried values.

## NULL Values

The NULL value of the table specifies that the field has been left blank during record creation. It is different from the value filled with zero or a field that contains space.

## Data Integrity

There are the following categories of data integrity exist with each RDBMS:

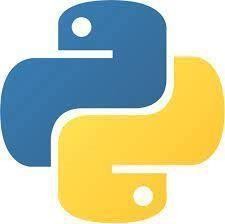
* **Entity integrity**: It specifies that there should be no duplicate rows in a table.
* **Domain integrity**: It enforces valid entries for a given column by restricting the type, the format, or the range of values.
* **Referential integrity** specifies that rows cannot be deleted, which are used by other records.
* **User-defined integrity**: It enforces some specific business rules defined by users. These rules are different from the entity, domain, or referential integrity.

# INTRODUCTION TO PYTHON PROGRAMMING

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high level programming language.

It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected.](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) It supports multiple [programming paradigms,](https://en.wikipedia.org/wiki/Programming_paradigms) including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural),](https://en.wikipedia.org/wiki/Procedural_programming) [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [functional programming.](https://en.wikipedia.org/wiki/Functional_programming)



# Origin of python

It is invented in Netherlands, in early 90‟s and its implementation was started in December 1989. Guido [van](https://en.wikipedia.org/wiki/Guido_van_Rossum) Rossum began working on Python in the late 1980s, as a successor to the [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)) [programming language,](https://en.wikipedia.org/wiki/ABC_(programming_language)) and first released it in 1991 as Python 0.9.0.

Python 2.0 was released in 2000 and introduced new features, such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a garbage collection system using [reference counting.](https://en.wikipedia.org/wiki/Reference_counting) Python 3.0 was released in 2008 and was a major revision of the language that is not completely [backward-compatible.](https://en.wikipedia.org/wiki/Backward_compatibility)

# Why the language is named Python ?

Despite all the reptiles made on Python books and icons,the truth is that Python is named after British comedy group Monty Python,makers of the 1970s BBC comedy series Monty Python‟s

Flying Circus and a handful of later full-length films,including Monty Python and The Holy Grail,that are still widely popular today.Python‟s original creator,Guido Van Rossum was a fan of Monty Python as of many software developers.

# Features of Python

There are many features in Python, some of which are discussed below –

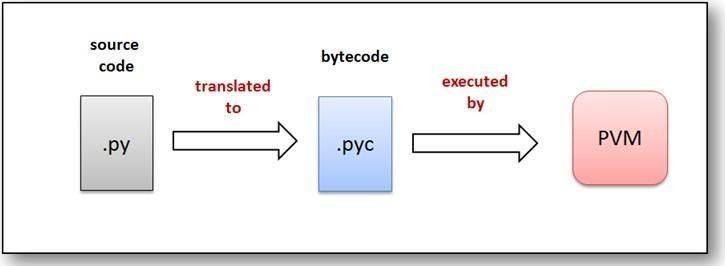
* **Easy to code** - Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc.
* **Free and Open Source** - Python language is freely available at the official website and you can download it.

**Introduction to Python and its features**

**Object-Oriented Language** - Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

* **GUI Programming Support** - Creating GUI based desktop applets.Using libraries like Tkinter, pyQt and pyGame developers can develop desktop applets supporting GUI.
* **Data Science and Machine Learning** - By far the most used application of python is in data science and machine learning. With its strong community and vast libraries the data processing has become quite easy in python. Be it data manipulation, data visualization, data cleaning python has its own libraries for the same.
* **High-Level Language**
* **Extensible feature** - We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.
* **Portable language** - For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
* **Integrated language** - Integration libraries like Cython and Jython makes python integrate with c/c++ and java for cross platform development.
* **Interpreted Language** - Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc..The source code of python is converted into an immediate form called **bytecode**.
* **Large Standard Library**- There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

## Internal Working of Python

The standard implementation of python is called “cpython”. It is the default and widely used implementation of Python.Python doesn‟t convert its code into machine code, something that hardware can understand. It actually converts it into something called **byte code**. So within python, compilation happens, but it‟s just not into a machine language. It is into byte code (.pyc ) and this byte code can‟t be understood by the CPU. So we need an interpreter called the **Python Virtual Machine** to execute the byte code.

## Real world Applications of Python

1. **Web and Internet Development**
   * Frameworks such as [Django](http://www.djangoproject.com/) and [Pyramid.](http://www.pylonsproject.org/)
   * Micro-frameworks such as [Flask](http://flask.pocoo.org/) and [Bottle.](http://bottlepy.org/)
   * [HTML and XML](http://docs.python.org/library/markup)
   * [JSON](http://docs.python.org/library/json.html)
   * [E-mail processing.](http://docs.python.org/library/email)
   * Support for [FTP,](http://docs.python.org/library/ftplib.html) [IMAP,](http://docs.python.org/2/library/imaplib.html) and other [Internet protocols.](http://docs.python.org/library/internet)
   * Easy-to-use [socket interface.](http://docs.python.org/howto/sockets.html)
   * [Requests,](https://pypi.org/project/requests/) a powerful HTTP client library.1

**Machine Learning and Artificial Intelligence -** [Machine Learning](https://www.edureka.co/blog/machine-learning-career/) and [Artificial Intelligence a](https://www.edureka.co/blog/pros-and-cons-of-ai/)re the talks of the town as they yield the most promising careers for the future. We make the computer learn based on past experiences through the data stored or better yet, create algorithms which makes the computer learn by itself.

**Audio and Video Applications -** We use Python to develop applications that can multi-task and also output media. Video and audio applications such as TimPlayer, Cplay have been developed using Python libraries. They provide better stability and performance in comparison to other media players.

## Python Flavors

* CPython(Python)
* JPython(Jython)
* IronPython: (Python for.NET)
* Stackless(Python for concurrency)
* PyPy (Python for speed)

## Disadvantages of Python

1. **Slow Speed:**

Python is an interpreted language and dynamically-typed language.The line by line execution of code often leads to slow execution.

1. **Not Memory Efficient :**

To provide simplicity to the developer, Python has to do a little tradeoff.The Python programming language uses a large amount of memory

.

1. **Weak in Mobile Computing:**

Python is generally used in server-side programming.We don‟t get to see Python on the client-side or mobile applications because of the reason that Python is not memory efficient and it has slow processing power as compared to other languages.

1. **Runtime Errors:**

As we know Python is a dynamically typed language so the data type of a variable can change anytime.A variable containing integer number may hold a string in the future, which can lead to Runtime Errors.

1. **Weak in mobile devices:**

We have seen python in web servers and desktop applications ,but it is not considered favorable for mobile devices because it uses more memory and slow processing compared to other languages. **Built-in Data Types**

## Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).10 Here are naming conventions for Python identifiers −

* Class names start with an uppercase letter. All other identifiers start with a lowercaseletter.
* Starting an identifier with a single leading underscore indicates that the identifier isprivate.
* Starting an identifier with two leading underscores indicates a strongly private identifier.
* If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## Indentation

Indentation is a very important concept of Python because without proper indenting the Python code, you will end up seeing IndentationError and the code will not get compiled.In simple terms indentation refers toadding white space before a statement.

Python indentation is a way of telling a Python interpreter that the group of statements belongs to a particular block of code. A block is a combination of all these statements. Block can be regarded as the grouping of statements for a specific purpose. Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation to highlight the blocks of code.

# INTRODUCTION TO PROJECT

Flight Management System is a python based project. We have developed this System using Python and oracle. The main modules available in this project are Booking Enquiry module which manages the functionality of Booking Enquiry, Flight Enquiry is normally used for managing Airline Enquiry, Flights Booking contains all the functionality related to Airlines Booking, Passenger , Reservation manages the Passenger Reservation functionality, Passenger has all the features of Passenger and Ticket Booking module manages the functionality of Ticket Booking. In this project we developed features for Booking Enquiry, Airline Enquiry, Passenger etc, which reduces the human efforts and increase the efficiency. Scope of python is growing day by day. This Python and oracle project on Flight Management System is a user based

**ABSTRACT**

Flight Management System is a python based project. We have developed this System using Python and oracle. The main modules available in this project are Booking Enquiry module which manages the functionality of Booking Enquiry, Flight Enquiry is normally used for managing Airline Enquiry, Flights Booking contains all the functionality related to Airlines Booking, Passenger , Reservation manages the Passenger Reservation functionality, Passenger has all the features of Passenger and Ticket Booking module manages the functionality of Ticket Booking. In this project we developed features for Booking Enquiry, Airline Enquiry, Passenger etc, which reduces the human efforts and increase the efficiency. Scope of python is growing day by day. This Python and oracle project on Flight Management System is a user based.

**OBJECTIVE**

The objectives of the Filght Management system are :-

* To reduce paperwork
* Fast processing
* Increased operational efficiency
* Data security
* Most importantly user friendly and easy to use.

This Project Contains Two Modes :

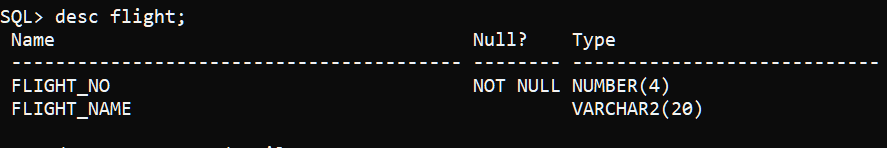
* + Administration
  + User

The administrator can access and modify all the details regarding the project in the database. This includes the train details , user details, adding or updating any column. It is the main mode through which the whole project is managed and being functioned.

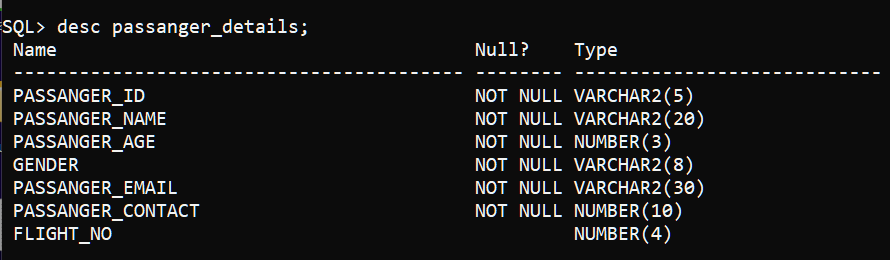
ADMINISTRATION mode contains the following functions:-

* + - ∙ Insert data:-use to insert the data into the table
    - ∙ Update data:-use to update the table
    - Delete data:-use to delete any entry from the table The user mode contain all the user related interface. Through this the user can check their all details.

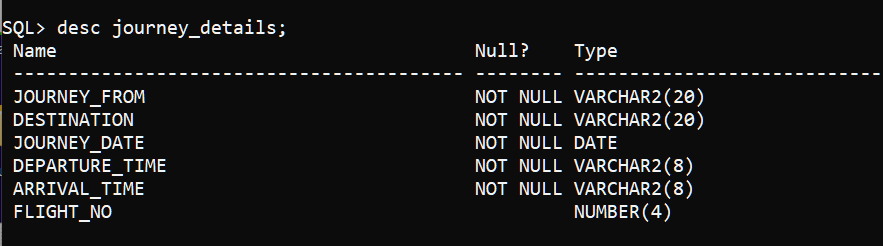
**Tables used in Flight Management System**



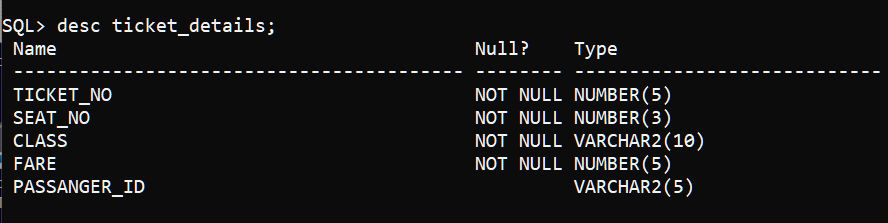
**Table 1**



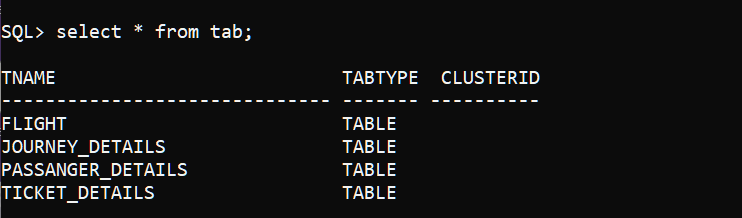
**Table 2**



**Table 3**



**Table 4**



**Table 5**

# INTRODUCTION TO ORACLE DATABASE

An Oracle **database** is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a [**server** r](https://docs.oracle.com/cd/B19306_01/server.102/b14220/glossary.htm#i432724)eliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has **logical structures** and **physical structures**. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

## Overview of Application Architecture

There are two common ways to architect a database: client/server or multitier. As internet computing becomes more prevalent in computing environments, many database management systems are moving to a multitier environment.

## Client/Server Architecture

* **Multiprocessing** uses more than one processor for a set of related jobs. Distributed processing reduces the load on a single processor by allowing different processors to concentrate on a subset of related tasks, thus improving the performance and capabilities of the system as a whole.
* An Oracle database system can easily take advantage of distributed processing by using its **client/server architecture**. In this architecture, the database system is divided into two parts: a front-end or a **client**, and a back-end or a **server**.

# The Client

The client is a database application that initiates a request for an operation to be performed on the database server. It requests, processes, and presents data managed by the server. The client workstation can be optimized for its job. For example, it might not need large disk capacity, or it might benefit from graphic capabilities.

Often, the client runs on a different computer than the database server, generally on a PC. Many clients can simultaneously run against one server.

## Multitier Architecture: Application Servers

A **multitier architecture** has the following components:

A client or initiator process that starts an operation

One or more application servers that perform parts of the operation. An **application server** provides access to the data for the client and performs some of the query processing, thus removing some of the load from the database server. It can serve as an interface between clients and multiple database servers, including providing an additional level of security.

An end or database server that stores most of the data used in the operation This architecture enables use of an application server to do the following:

Validate the credentials of a client, such as a Web browser

Connect to an Oracle database server

Perform the requested operation on behalf of the client

If proxy authentication is being used, then the identity of the client is maintained throughout all tiers of the connection.

## Overview of Physical Database Structures\

The following sections explain the physical database structures of an Oracle database, including datafiles, redo log files, and control files.

## Datafiles

Every Oracle database has one or more physical **datafiles**. The datafiles contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the datafiles allocated for a database.

## The characteristics of datafiles are:

* A datafile can be associated with only one database.
* Datafiles can have certain characteristics set to let them automatically extend when the database runs out of space.
* One or more datafiles form a logical unit of database storage called a tablespace.
* Data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested information is not already in the memory cache for the database, then it is read from the appropriate datafiles and stored in memory.
* Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and to increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the [**database writer process (DBW*n*)**](https://docs.oracle.com/cd/B19306_01/server.102/b14220/glossary.htm#i996724)[b](https://docs.oracle.com/cd/B19306_01/server.102/b14220/glossary.htm#i996724)ackground process.

# INTRODUCTION TO Tkinter

Tkinter is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don‟t need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit.Installation :

# What are Widgets?

Widgets in Tkinter are the elements of GUI application which provides various controls (such as Labels, Buttons, ComboBoxes, CheckBoxes, MenuBars, RadioButtons and many more) to users to interact with the application.

**Basic Tkinter Widgets:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Widgets | Description |  |
|  | Label | It is used to display text or image on the screen |  |
|  | Button | It is used to add buttons to your application |  |
|  | ComboBox | It contains a down arrow to select from list of available options |  |
|  | RadiButton | It is used to implement one-of-many selection as it allows only optio be selected |  |
|  | Entry | It is used to input single line text entry from user |  |
|  | Frame | It is used as container to hold and organize the widgets |  |
|  | Message | It works same as that of label and refers to multi-line and non- | itabl  t |

# SOURCE CODE :

import random

from tkinter import messagebox

import uuid

from tkinter import \*

from tkinter.ttk import Combobox

import tkinter as tk

from PIL import Image, ImageTk

import cx\_Oracle

import pandas as pd

con = cx\_Oracle.connect("C##Anjar/123456")

cur = con.cursor()

def paymentdetails(searchscreen):

if (f.get() != '' and c.get() != ''):

searchscreen.destroy()

con = cx\_Oracle.connect("C##Anjar/123456")

cur = con.cursor()

personaldetails = Tk()

personaldetails.title("Personal details and Payment")

personaldetails.resizable(0, 0)

personaldetails.geometry("850x600+200+100")

personaldetails.configure(bg="lavender")

frame\_personal = Frame(personaldetails).pack(side="top")

gender = StringVar()

pname = StringVar()

page = StringVar()

pnumber = StringVar()

pemail = StringVar()

pnameoncard = StringVar()

pexpmon = StringVar()

pcardno = StringVar()

def price():

if f.get() == '601':

if c.get() == "Business":

rate = 4200

elif c.get() == "Economy":

rate = 2670

return rate

elif f.get() == '602':

if c.get() == "Business":

rate = 9560

elif c.get() == "Economy":

rate = 6390

return rate

elif f.get() == '603':

if c.get() == "Business":

rate = 11940

elif c.get() == "Economy":

rate = 7340

return rate

elif f.get() == '604':

if c.get() == "Business":

rate = 9780

elif c.get() == "Economy":

rate = 6500

return rate

elif f.get() == '605':

if c.get() == "Business":

rate = 13650

elif c.get() == "Economy":

rate = 9300

return rate

elif f.get() == '606':

if c.get() == "Business":

rate = 8220

elif c.get() == "Economy":

rate = 4800

return rate

fno = f.get()

stat1 = ''' select journey\_from,destination from journey\_details where flight\_no = :y'''

cur.execute(stat1, y=fno)

var1 = cur.fetchone()

stat2 = '''select flight\_name from flight where flight\_no = :z'''

cur.execute(stat2, z=fno)

var2 = cur.fetchone()

from\_label = Label(personaldetails, text="FROM", font=(

"Consolas", 10), bg="lavender", fg="black", width=10).place(x=660, y=49)

todestination\_label = Label(personaldetails, text="TO", font=(

"Consolas", 10), bg="lavender", fg="black", width=10).place(x=750, y=49)

from\_entrylabel = Label(personaldetails, text=var1[0], font=(

"Consolas", 12), bg="#F6E7D8", fg="black", width=10).place(x=640, y=69)

todestinatiion\_entrylabel = Label(personaldetails, text=var1[1], font=(

"Consolas", 12), bg="#F6E7D8", fg="black", width=10).place(x=740, y=69)

Flightno\_label = Label(personaldetails, text="FLIGHT ", font=(

"Consolas", 10), bg="lavender", fg="black", width=10).place(x=700, y=115)

flighno\_entrylabel = Label(personaldetails, text=f.get(), font=(

"Consolas", 12), bg="#F6E7D8", fg="black", width=10).place(x=640, y=140)

flightname\_entrylabel = Label(personaldetails, text=var2[0], font=(

"Consolas", 12), bg="#F6E7D8", fg="black", width=10).place(x=740, y=140)

Class\_label = Label(personaldetails, text="CLASS", font=(

"Consolas", 10), bg="lavender", fg="black", width=10).place(x=660, y=186)

Classname\_entrylabel = Label(personaldetails, text=c.get(), font=(

"Consolas", 12), bg="#F6E7D8", fg="black", width=10).place(x=640, y=210)

Price\_label = Label(personaldetails, text="PRICE(Rs.)", font=(

"Consolas", 10), bg="lavender", fg="black", width=10).place(x=750, y=186)

Price\_entrylabel = Label(personaldetails, text=price(), font=(

"Consolas", 12), bg="#F6E7D8", fg="black", width=10).place(x=740, y=210)

heading\_color = Label(frame\_personal, text="a", width=200, height=2,

bg="#120c6e", fg="#120c6e", relief='flat').pack(side="top")

heading\_label = Label(frame\_personal, text="PERSONAL DETAILS", font=(

"Sitka Small Semibold", 14), width=20, height=0, bg="#120c6e", fg="white").place(x=5, y=1)

name\_label = Label(personaldetails, text="Name:", font=(

"Sitka Small Semibold", 12), bg="lavender", fg="black", width=10).place(x=20, y=57)

age\_label = Label(personaldetails, text="Age:", font=(

"Sitka Small Semibold", 12), bg="lavender", fg="black", width=9).place(x=20, y=107)

number\_label = Label(personaldetails, text="Phone no:", font=(

"Sitka Small Semibold", 12), bg="lavender", fg="black", width=10).place(x=20, y=157)

email\_label = Label(personaldetails, text="Email Id:", font=(

"Sitka Small Semibold", 12), bg="lavender", fg="black", width=10).place(x=20, y=207)

gender\_label = Label(personaldetails, text="Gender:", font=(

"Sitka Small Semibold", 13), bg="lavender", fg="black", width=10).place(x=19, y=257)

name\_entry = Entry(personaldetails, width=20, bg="#EEEEEE", font=(

"Microsoft PhagsPa", 14), textvariable=pname).place(x=140, y=60)

age\_entry = Entry(personaldetails, width=20, bg="#EEEEEE", font=(

"Microsoft PhagsPa", 14), textvariable=page).place(x=140, y=110)

number\_entry = Entry(personaldetails, width=17, bg="#EEEEEE", font=(

"Microsoft PhagsPa", 14), textvariable=pnumber).place(x=170, y=160)

numbernineone\_label = Label(personaldetails, text="+91 ", bg="lavender",

fg="black", font=("arial", 14), width=3).place(x=130, y=160)

email\_entry = Entry(personaldetails, width=23, bg="#EEEEEE", font=(

"Consoals", 12), textvariable=pemail).place(x=140, y=210)

male\_radiobutton = Radiobutton(personaldetails, text="Male", font=(

"Consoals", 13), width=10, bg="lavender", variable=gender, value=0, activebackground="lavender").place(x=120, y=260)

female\_radiobutton = Radiobutton(personaldetails, text="Female", font=(

"Consoals", 13), width=10, bg="lavender", variable=gender, value=1, activebackground="lavender").place(x=220, y=260)

heading\_color = Label(frame\_personal, text="a", width=200, height=2,

bg="#120c6e", fg="#120c6e", relief='flat').place(x=0, y=300)

paymentdetails\_label = Label(frame\_personal, text="PAYMENT DETAILS", font=(

"Sitka Small Semibold", 14), width=20, height=0, bg="#120c6e", fg="white").place(x=0, y=300)

cardheading\_label = Label(frame\_personal, text="Credit/Debit/ATM Card", font=(

"Microsoft YaHei UI", 11), bg="lavender").place(x=15, y=350)

cardnumber\_label = Label(personaldetails, text="Card Number:", font=(

"Sitka Small Semibold", 11), bg="lavender", fg="black", width=12).place(x=15, y=390)

cardnumber\_entry = Entry(personaldetails, width=20, bg="#EEEEEE", textvariable=pcardno, font=(

"Consoals", 14)).place(x=155, y=390)

nameoncard\_label = Label(personaldetails, text="Name on Card:", font=(

"Sitka Small Semibold", 11), bg="lavender", fg="black", width=12).place(x=16, y=440)

nameoncard\_entry = Entry(personaldetails, width=20, textvariable=pnameoncard, bg="#EEEEEE", font=(

"Consoals", 14)).place(x=155, y=440)

expirymonth\_label = Label(personaldetails, text="Exp.Month &\nYear(MM/YY)", font=(

"Sitka Small Semibold", 11), bg="lavender", fg="black", width=12).place(x=15, y=490)

expirymonth\_entry = Entry(personaldetails, width=20, bg="#EEEEEE", textvariable=pexpmon, font=(

"Consoals", 14)).place(x=155, y=500)

def back():

bookticket(personaldetails)

def bookit():

if (pname.get() == "" or page.get() == "" or gender.get() == "" or pemail.get() == "" or pnumber.get() == "" or pnameoncard.get() == ""

or pexpmon.get() == "" or pcardno.get() == ""):

messagebox.showerror(

"Error", "PLEASE FILL ALL THE REQUIRED DETAILS !!")

else:

con = cx\_Oracle.connect("C##Anjar/123456")

cur = con.cursor()

if gender.get() == "0":

gen = "male"

elif gender.get() == "1":

gen = "female"

ppname = pname.get()

ppage = int(page.get())

ppflight = int(f.get())

ppemail = pemail.get()

ppnumber = int(pnumber.get())

pid = str(uuid.uuid4())

pid = pid[:5]

ticno = random.randint(10000, 99999)

seatno = random.randint(100, 500)

clas = c.get()

fare = price()

stat1 = '''insert into passanger\_details values(:a,:b,:c,:d,:e,:f,:g)'''

stat2 = '''insert into ticket\_details values(:z,:y,:x,:w,:v)'''

cur.execute(stat1, [pid, ppname, ppage,

gen, ppemail, ppnumber, ppflight])

cur.execute(stat2, [ticno, seatno, clas, fare, pid])

con.commit()

response = messagebox.showinfo(

"Confirmation", "You have successfully booked the ticket.\n\nYour passangerid has been generated!\n\nclick ok to view!")

if response == "ok":

messagebox.showinfo("Passanger id", pid)

if response == "ok":

ticketdetails(personaldetails)

paynow\_button = Button(personaldetails, text="PAY NOW", width=10, bd=3, font=(

"Sitka Small Semibold", 10), bg="#035397", fg="white", activebackground="#B7CADB", command=bookit).place(x=730, y=550)

cancel\_button = Button(personaldetails, text="BACK", width=10, bd=3, font=(

"Sitka Small Semibold", 10), bg="#035397", fg="white", activebackground="#B7CADB", command=back).place(x=620, y=550)

con.close()

personaldetails.mainloop()

else:

messagebox.showerror(

"Error", "PLEASE SELECT BOTH FLIGHT NO. AND CLASS !")

def bookticket(root):

root.destroy()

searchscreen = Tk()

searchscreen.title("Book Flight")

searchscreen.geometry("970x600+200+100")

searchscreen.resizable(0, 0)

searchscreen.configure(bg="lavender")

frame\_search = Frame(searchscreen).pack(side="top")

global f

global c

f = StringVar()

c = StringVar()

heading\_color = Label(frame\_search, text="a", width=200, height=2,

bg="#120c6e", fg="#120c6e", relief='flat').pack(side="top")

heading\_label = Label(frame\_search, text="AVAILABLE FLIGHTS", font=(

"Sitka Small Semibold", 14), width=20, height=0, bg="#120c6e", fg="white").place(x=5, y=1)

flightno\_label = Label(frame\_search, text="Flight no.", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=11).place(x=30, y=60)

flightname\_label = Label(frame\_search, text="Flight name", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=11).place(x=160, y=60)

flightfrom\_label = Label(frame\_search, text="From", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=11).place(x=290, y=60)

flightto\_label = Label(frame\_search, text="to", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=11).place(x=420, y=60)

flightdepttime\_label = Label(frame\_search, text="Dept. time", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=11).place(x=550, y=60)

flightarrtime\_label = Label(frame\_search, text="Arrival time", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=13).place(x=680, y=60)

fligthdate\_label = Label(frame\_search, text="Date", bg="#4d77ff", fg="white", font=(

"Consolas", 12), width=11).place(x=830, y=60)

flightno1 = Label(frame\_search, text="601", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=30, y=100)

flightname1 = Label(frame\_search, text="Air India", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=160, y=100)

flightfrom1 = Label(frame\_search, text="Amritsar", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=290, y=100)

flighto1 = Label(frame\_search, text="New Delhi", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=420, y=100)

flightdepttime1 = Label(frame\_search, text="16:10", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=550, y=100)

flightarrtime1 = Label(frame\_search, text="18:05", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=13).place(x=680, y=100)

flight1date = Label(frame\_search, text="17-may-2022", bg="#F6E7D8",

fg="black", font=("Consolas", 12), width=12).place(x=825, y=100)

flightno2 = Label(frame\_search, text="602", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=30, y=140)

flightname2 = Label(frame\_search, text="Indigo", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=160, y=140)

flightfrom1 = Label(frame\_search, text="New Delhi", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=290, y=140)

flighto2 = Label(frame\_search, text="Bengaluru", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=420, y=140)

flightdepttime2 = Label(frame\_search, text="20:10", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=550, y=140)

flightarrtime2 = Label(frame\_search, text="23:15", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=13).place(x=680, y=140)

flight2date = Label(frame\_search, text="19-may-2022", bg="#F6E7D8",

fg="black", font=("Consolas", 12), width=12).place(x=825, y=140)

flightno3 = Label(frame\_search, text="603", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=30, y=180)

flightname3 = Label(frame\_search, text="GoAir", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=160, y=180)

flightfrom3 = Label(frame\_search, text="New Delhi", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=290, y=180)

flighto3 = Label(frame\_search, text="Mumbai", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=420, y=180)

flightdepttime3 = Label(frame\_search, text="08:10", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=550, y=180)

flightarrtime3 = Label(frame\_search, text="10:30", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=13).place(x=680, y=180)

flight3date = Label(frame\_search, text="21-may-2022", bg="#F6E7D8",

fg="black", font=("Consolas", 12), width=12).place(x=825, y=180)

flightno4 = Label(frame\_search, text="604", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=30, y=220)

flightname4 = Label(frame\_search, text="SpiceJet", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=160, y=220)

flightfrom4 = Label(frame\_search, text="Amritsar", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=290, y=220)

flighto4 = Label(frame\_search, text="Patna", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=420, y=220)

flightdepttime4 = Label(frame\_search, text="14:00", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=550, y=220)

flightarrtime4 = Label(frame\_search, text="16:20", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=13).place(x=680, y=220)

flight4date = Label(frame\_search, text="25-may-2022", bg="#F6E7D8",

fg="black", font=("Consolas", 12), width=12).place(x=825, y=220)

flightno5 = Label(frame\_search, text="605", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=30, y=260)

flightname5 = Label(frame\_search, text="Vistara", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=160, y=260)

flightfrom5 = Label(frame\_search, text="Amritsar", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=290, y=260)

flighto5 = Label(frame\_search, text="Hyderabad", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=420, y=260)

flightdepttime5 = Label(frame\_search, text="11:30", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=550, y=260)

flightarrtime5 = Label(frame\_search, text="15:10", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=13).place(x=680, y=260)

flight6date = Label(frame\_search, text="23-may-2022", bg="#F6E7D8",

fg="black", font=("Consolas", 12), width=12).place(x=825, y=260)

flightno6 = Label(frame\_search, text="606", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=30, y=300)

flightname6 = Label(frame\_search, text="Jet Airways", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=160, y=300)

flightfrom6 = Label(frame\_search, text="New Delhi", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=290, y=300)

flighto6 = Label(frame\_search, text="Kolkata", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=420, y=300)

flightdepttime6 = Label(frame\_search, text="17:45", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=11).place(x=550, y=300)

flightarrtime6 = Label(frame\_search, text="19:50", bg="#F6E7D8", fg="black", font=(

"Consolas", 12), width=13).place(x=680, y=300)

flight6date = Label(frame\_search, text="26-may-2022", bg="#F6E7D8",

fg="black", font=("Consolas", 12), width=12).place(x=825, y=300)

selectflightlabel = Label(frame\_search, text="Select flight no.", font=(

"Sitka Small Semibold", 12), width=13, bg="#120c6e", fg="white").place(x=150, y=380)

flightno\_combobox = Combobox(frame\_search, values=[601, 602, 603, 604, 605, 606], font=(

"Segoe UI Semibold", 10), textvariable=f).place(x=150, y=420)

selectdatelabel = Label(frame\_search, text="Select class", font=(

"Sitka Small Semibold", 12), width=12, bg="#120c6e", fg="white").place(x=350, y=380)

date\_combobox = Combobox(frame\_search, values=['Business', 'Economy'], font=(

"Segoe UI Semibold", 10), textvariable=c).place(x=350, y=420)

def back():

searchscreen.destroy()

frontscreen()

flightbook\_button = Button(frame\_search, text="Proceed", width=9, font=("Sitka Small Semibold", 10), bd=3, bg="#035397",

fg="white", activebackground="#B7CADB", command=lambda: paymentdetails(searchscreen)).place(x=850, y=550)

backhome\_button = Button(frame\_search, text="Back", width=9, font=("Sitka Small Semibold", 10),

bd=3, bg="#035397", fg="white", activebackground="#B7CADB", command=back).place(x=746, y=550)

searchscreen.mainloop()

def frontscreen():

root = Tk()

root.title("Flight reservation system")

root.geometry("800x533+200+100")

root.resizable(0, 0)

frame1 = Frame(root, bg="#E9D5DA").place(width=400, height=800, x=0, y=0)

frame2 = Frame(root, bg="#827397").place(width=400, height=533, x=400, y=0)

x = ImageTk.PhotoImage(Image.open("planewall.jpg"))

y = Label(root, image=x).place(width=800, height=533)

heading = Label(frame1, text="FLIGHT\t RESERVATION\tSYSTEM", width=60, fg="white",

bg='#120c6e', relief="sunken", font=("Source Serif Pro Semibold", 25)).pack()

search\_button = Button(frame2, text="BOOK FLIGHT", font=("Source Sans Pro Semibold", 13), relief="raised", bg="#035397",

fg='white', bd=5, activebackground="#B7CADB", width=14, command=lambda: bookticket(root)).place(x=50, y=440)

ticketdetails\_button = Button(frame2, text="TICKET DETAILS", font=("Source Sans Pro Semibold", 13), bd=5, bg="#035397",

fg='white', activebackground="#B7CADB", relief="raised", width=14, command=lambda: ticketdetails(root)).place(x=320, y=440)

admin\_button = Button(frame2, text="ADMIN LOGIN", font=("Source Sans Pro Semibold", 13), relief="raised", bd=5,

bg="#035397", fg='white', activebackground="#B7CADB", width=14, command=lambda: adminlogin(root)).place(x=600, y=440)

root.mainloop()

def ticketdetails(root):

root.destroy()

ticketroot = Tk()

ticketroot.geometry("700x560+200+100")

ticketroot.resizable(0, 0)

ticketroot.title("TICKET DETAILS")

ticketroot.config(bg="lavender")

passid = StringVar()

con = cx\_Oracle.connect("C##Anjar/123456")

cur = con.cursor()

ticketframe = Frame(ticketroot, bg="white").place(

width=640, height=340, x=30, y=170)

ticketpic = ImageTk.PhotoImage(Image.open("2ticketpic.png"))

ticketpiclabel = Label(ticketroot, image=ticketpic, bg="lavender").place(

height=128, width=200, x=450, y=40)

searchpic = ImageTk.PhotoImage(Image.open("2search.png"))

searchpiclabel = Label(ticketroot, image=searchpic).place(

height=15, width=15, x=160, y=130)

profilepic = ImageTk.PhotoImage(Image.open("2profilepic.png"))

profilepiclabel = Label(ticketframe, image=profilepic).place(

height=130, width=130, x=40, y=180)

ticketdetailsheading\_label = Label(ticketroot, text="TICKET DETAILS", width=60,

bg="#120c6e", fg="white", font=("Sitka Small Semibold", 15)).pack()

enterpassangeid\_label = Label(ticketroot, text="ENTER PASSANGER ID :", font=(

"Source Serif Pro", 13), bg="lavender", fg="black").place(x=30, y=40)

enterpassangeid\_entry = Entry(ticketroot, font=(

"Source Serif Pro", 14), bg="#EEEEEE", width=20, textvariable=passid).place(x=30, y=75)

def filldetails():

psid = passid.get()

stat1 = '''select \* from passanger\_details where passanger\_id = :a'''

stat2 = '''select \* from ticket\_details where passanger\_id = :b'''

stat3 = '''select \* from flight where flight\_no = :c'''

stat4 = ''' select \* from journey\_details where flight\_no = :d'''

cur.execute(stat1, a=psid)

var1 = cur.fetchone()

cur.execute(stat2, b=psid)

var2 = cur.fetchone()

if var1 == None and var2 == None:

messagebox.showerror("error", "PASSANGER ID DOES NOT EXIST")

else:

passangeridentry\_label = Label(ticketframe, text=passid.get(), bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=10).place(x=219, y=205)

ticketnoentry\_label = Label(ticketframe, text=var2[0], bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=8).place(x=540, y=205)

Flightnoentry\_label = Label(ticketframe, text=var1[6], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=5).place(x=219, y=290)

flno = var1[6]

cur.execute(stat3, c=flno)

var3 = cur.fetchone()

Flightnameentry\_label = Label(ticketframe, text=var3[1], bg="#F7E9D7", fg="black", font=(

"Source Serif Pro", 13), width=9).place(x=540, y=290)

Passangernameentry\_label = Label(ticketframe, text=var1[1], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=13).place(x=215, y=380)

classentry\_label = Label(ticketframe, text=var2[2], bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=10).place(x=380, y=290)

Passangerageentry\_label = Label(ticketframe, text=var1[2], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=4).place(x=430, y=380)

Passangersex\_label = Label(ticketframe, text=var1[3], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=7).place(x=570, y=380)

fromentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=10).place(x=120, y=450)

destinationentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=10).place(x=360, y=450)

seatnoentry\_label = Label(ticketframe, text=var2[1], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=5).place(x=570, y=450)

cur.execute(stat4, d=flno)

var4 = cur.fetchone()

fromentry\_label = Label(ticketframe, text=var4[0], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=10).place(x=120, y=450)

destinationentry\_label = Label(ticketframe, text=var4[1], bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=10).place(x=360, y=450)

date = str(var4[2])

x = date.split()

dateentry\_label = Label(ticketframe, text=x[0], bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=10).place(x=380, y=205)

passid.set('')

search\_button = Button(ticketroot, text="SEARCH", font=("Microsoft Sans Serif", 11), activebackground="#B7CADB",

width=12, bg="#035397", fg="white", bd=4, relief="raised", command=filldetails).place(x=30, y=118)

passangeridheading\_label = Label(ticketframe, text="Passanger Id:", bg="white", font=(

"Source Serif Pro", 13)).place(x=220, y=175)

passangeridentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=10).place(x=219, y=205)

dateheading\_label = Label(ticketframe, text="Date:", bg="white", font=(

"Source Serif Pro", 13)).place(x=400, y=175)

dateentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=10).place(x=380, y=205)

ticketnoheading\_label = Label(ticketframe, text="Ticket No:", bg="white", font=(

"Source Serif Pro", 13)).place(x=540, y=175)

ticketnoentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=8).place(x=540, y=205)

Flightnotitle\_label = Label(ticketframe, text="Flight No:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=219, y=255)

Flightnametitle\_label = Label(ticketframe, text="Flight Name:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=540, y=255)

classheading\_label = Label(ticketframe, text="Class:", bg="white", font=(

"Source Serif Pro", 13)).place(x=400, y=255)

classentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), fg="black", width=10).place(x=380, y=290)

Flightnoentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=5).place(x=219, y=290)

Flightnameentry\_label = Label(ticketframe, text="", bg="#F7E9D7", fg="black", font=(

"Source Serif Pro", 13), width=9).place(x=540, y=290)

separation\_label = Label(

ticketframe, text="--------------------------------------------------------------------------------------------------------------------------", bg="white").place(x=40, y=340)

PassNametitle\_label = Label(ticketframe, text="Passanger name:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=60, y=380)

Passangernameentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=13).place(x=215, y=380)

PassAgetitle\_label = Label(ticketframe, text="Age:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=380, y=380)

Passangerageentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=4).place(x=430, y=380)

Passsextitle\_label = Label(ticketframe, text="Sex:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=520, y=380)

Passangersex\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=7).place(x=570, y=380)

fromorigin\_label = Label(ticketframe, text="From:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=60, y=450)

fromentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=10).place(x=120, y=450)

destinationtitle\_label = Label(ticketframe, text="Destination:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=250, y=450)

destinationentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=10).place(x=360, y=450)

seatnotitle\_label = Label(ticketframe, text="Seat No:", font=(

"Source Serif Pro", 13), bg="white", fg="black").place(x=490, y=450)

seatnoentry\_label = Label(ticketframe, text="", bg="#F7E9D7", font=(

"Source Serif Pro", 13), width=5).place(x=570, y=450)

def back():

ticketroot.destroy()

frontscreen()

backbuttn = Button(ticketroot, text="Back", font=("Microsoft Sans Serif", 10), activebackground="#B7CADB",

width=8, bg="#035397", fg="white", bd=4, relief="raised", command=back).place(x=600, y=519)

ticketroot.mainloop()

def adminlogin(root):

root.destroy()

adminroot = Tk()

adminroot.geometry("500x550+400+100")

adminroot.resizable(0, 0)

adminroot.title("ADMIN LOGIN")

adminroot.config(bg="lavender")

global usrnme

global pswd

usrnme = StringVar()

pswd = StringVar()

adminframe = Frame(adminroot, bg="white").place(

width=300, height=420, x=100, y=90)

adminloginheading\_label = Label(adminroot, text="Admin Login", font=(

"Sitka Small Semibold", 14), bg="#120c6e", fg="white", width=60).pack()

admindp = ImageTk.PhotoImage(Image.open("2admindpnew.png"))

admindp\_label = Label(adminframe, image=admindp).place(x=200, y=100)

Username\_label = Label(adminframe, text="User name", font=(

"Microsoft YaHei UI", 12), fg="black", bg="white").place(x=115, y=210)

username\_entry = Entry(adminframe, width=25, bg="#EEEEEE", font=(

"Microsoft PhagsPa", 14), textvariable=usrnme).place(x=120, y=240)

password\_label = Label(adminframe, text="Password", font=(

"Microsoft YaHei UI", 12), fg="black", bg="white").place(x=115, y=300)

password\_entry = Entry(adminframe, width=25, show="\*", bg="#EEEEEE",

font=("Microsoft PhagsPa", 14), textvariable=pswd).place(x=115, y=330)

login\_button = Button(adminframe, text="Login", font=("Sitka Small Semibold", 11), activebackground="#B7CADB", width=11,

bg="#035397", fg="white", bd=4, relief="raised", command=lambda: bookingdetails(adminroot)).place(x=185, y=400)

def back():

adminroot.destroy()

frontscreen()

backbutton = Button(adminroot, text="Back", font=("Microsoft Sans Serif", 9), activebackground="#B7CADB",

width=8, bg="#035397", fg="white", bd=4, command=back).place(x=5, y=40)

adminroot.mainloop()

def bookingdetails(adminroot):

if usrnme.get() == "admin" and pswd.get() == "admin":

adminroot.destroy()

passroot = Tk()

passroot.title("All Bookings")

passroot.geometry("900x600+200+100")

passroot.configure(bg="lavender")

passroot.resizable(0, 0)

bookingdetails\_label = Label(passroot, text="All Bookings", font=(

"Sitka Small Semibold", 14), bg="#120c6e", fg="white", width=90).pack()

con = cx\_Oracle.connect("C##Anjar/123456")

cur = con.cursor()

Quote = pd.read\_sql\_query("select \* from passanger\_details", con)

Quote = Quote.set\_index('PASSANGER\_ID')

T = Text(passroot, height=60, width=200)

T.pack()

T.insert(tk.END, Quote)

T.config(state='disabled')

def back():

passroot.destroy()

frontscreen()

backbutton = Button(passroot, text="Home", font=("Microsoft Sans Serif", 11), activebackground="#B7CADB",

width=8, bg="#035397", fg="white", bd=4, command=back).place(x=800, y=550)

passroot.mainloop()

elif usrnme.get() == "" or pswd.get() == "":

messagebox.showerror("Error!", "missing credentials !")

else:

messagebox.showerror("Error!", "Bad credentials !")

frontscreen()

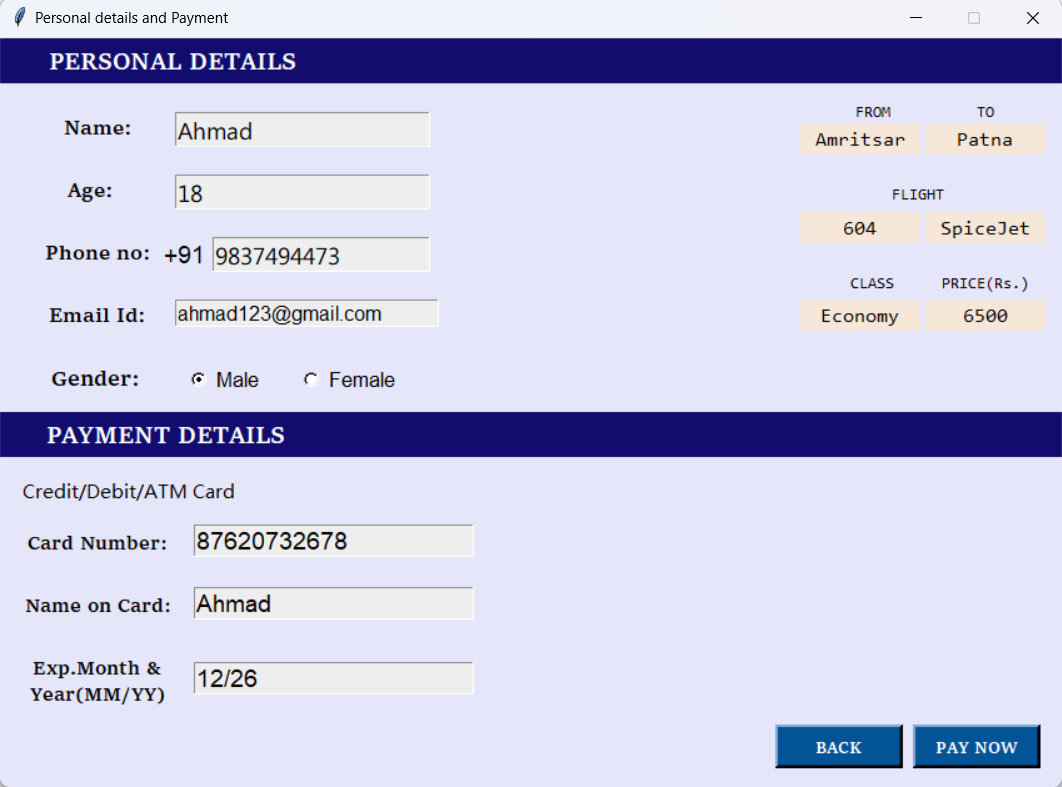
# SNAPSHOTS

# 

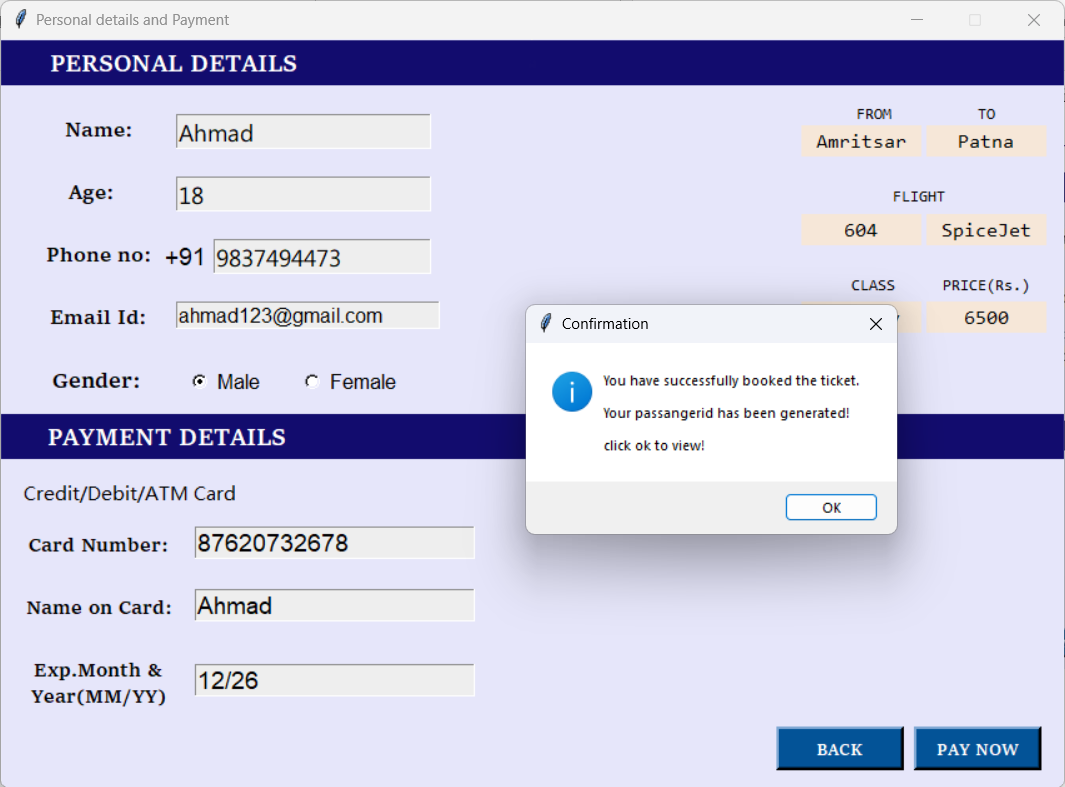
* This is the Main screen from where you can book the ticket, See the booked ticket details and admin login.



* This page opens when we click on „BOOK FLIGHT‟. From here, we can see all the available flghts , then select flight number and class according to our preference.

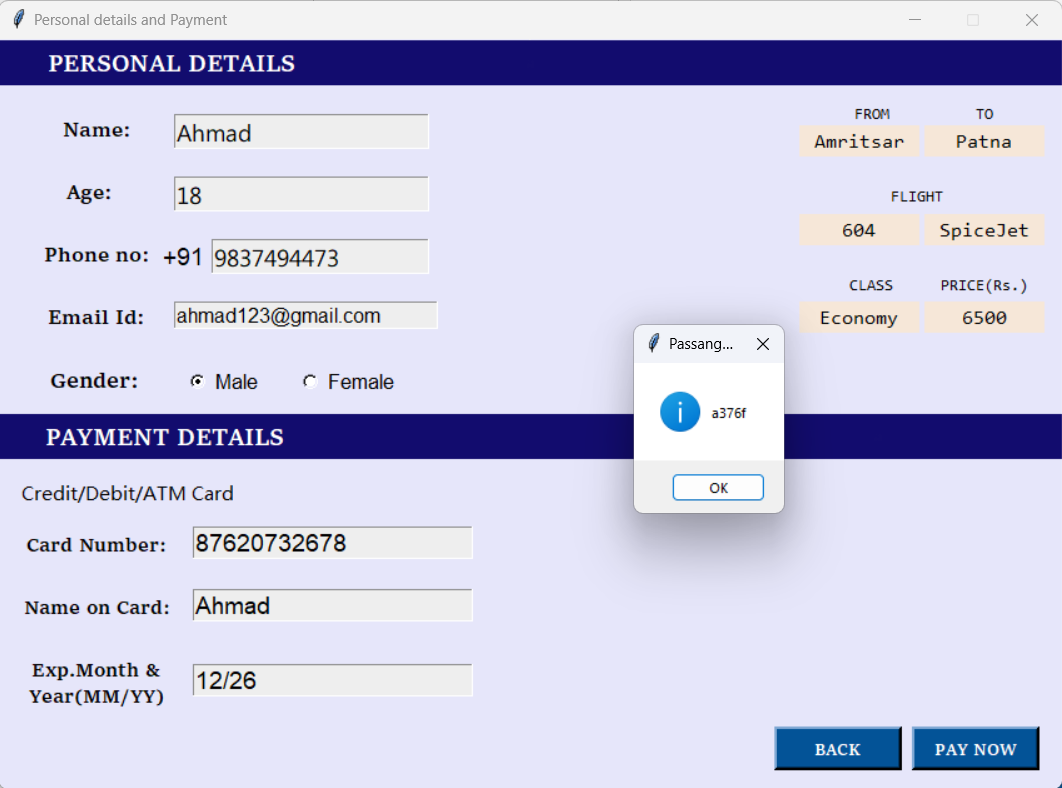


* This page appears after clicking on „PROCEED‟ in „BOOK FLIGHT‟ Window. Here, we will enter all the required details.

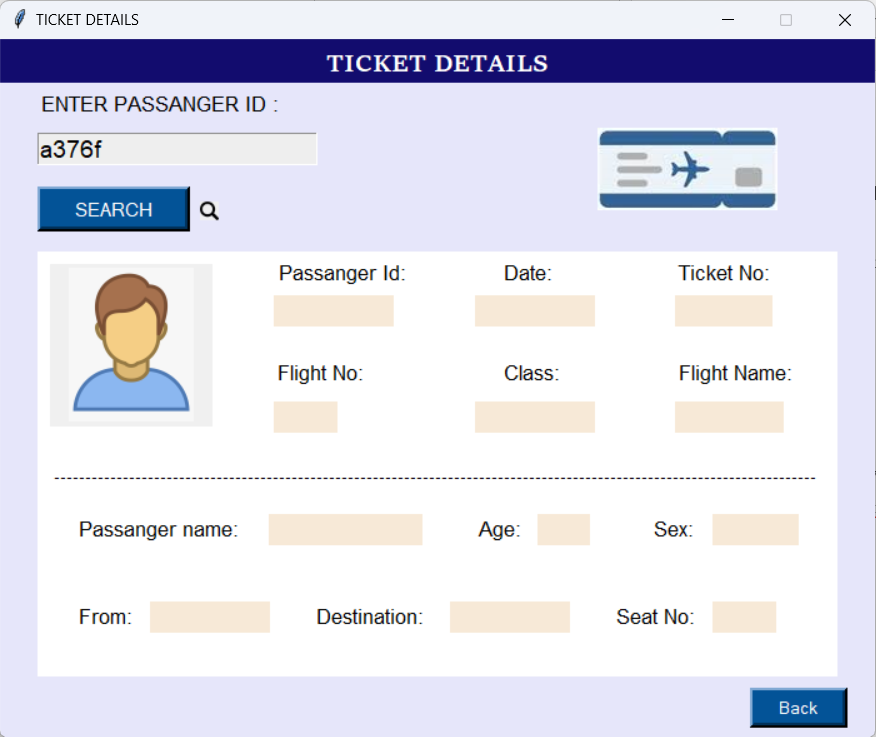


* After clicking on „PAY NOW‟, a confirmation messagebox will appear. .

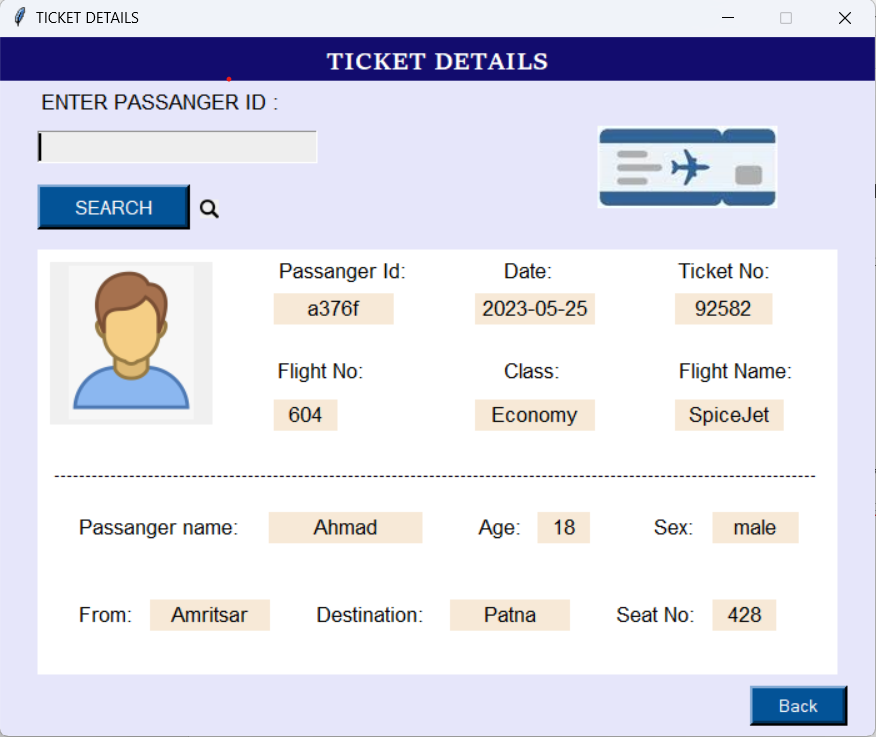
FIG. 4



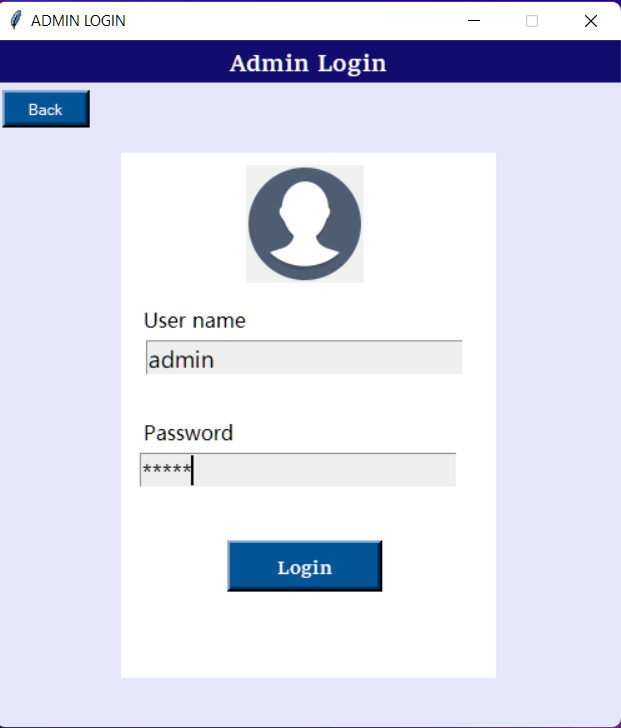
* After clicking on „ok‟, a unique passanger id will be generated .



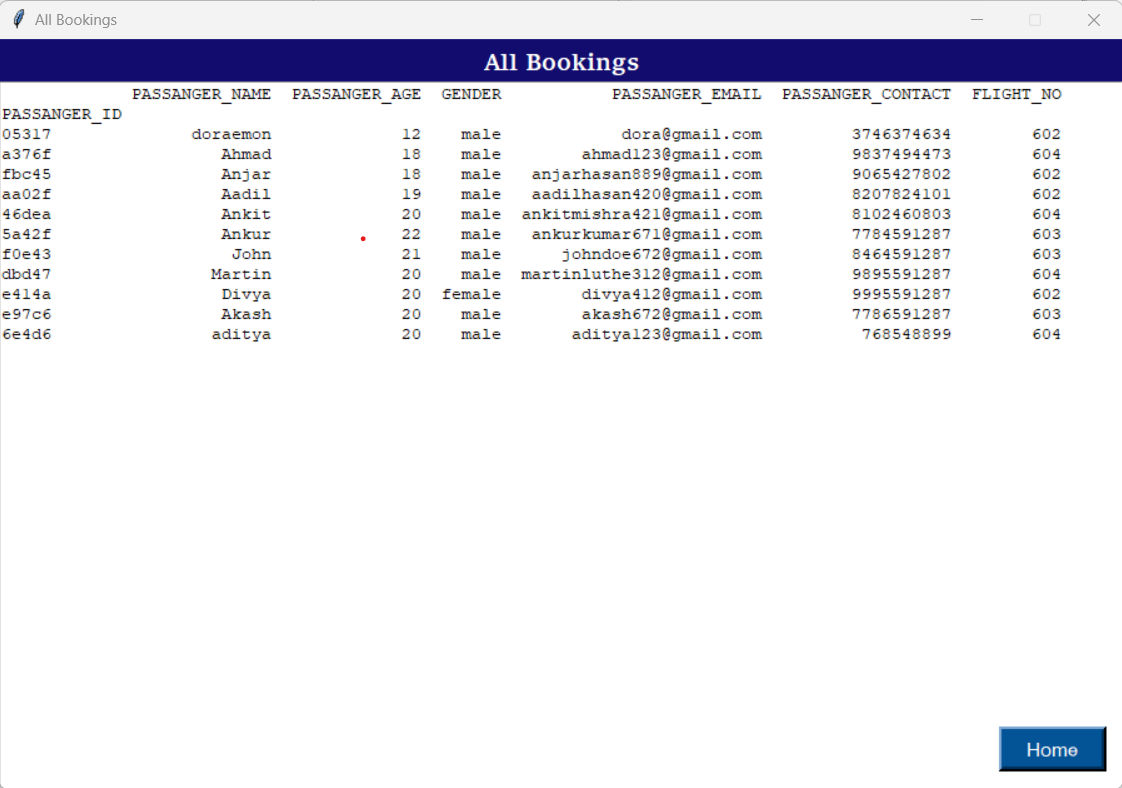
* This is Ticket details window, here, we will enter the unique passanger id to see all the ticket details.



* After entering the passanger id and clicking on search, all the ticket details will appear on the empty labels.



* This is the login page, here, admin will enter his username and password to see all the bookings.



* After clicking on Login in admin login window, all the booking details window will appear. Here. Admin can see every booking and their passanger details.

**REFERENCES**

* [https://lnu.diva-portal.org](https://lnu.diva-portal.org/)
* [https://www.geeksforgeeks.org](https://www.geeksforgeeks.org/)
* https://[www.javatpoint.com](http://www.javatpoint.com/)
* <https://www.freeprojectz.com/premium>[-](https://www.freeprojectz.com/premium-synopsis/synopsis-student-management-system#%3A~%3Atext%3DThe%20main%20objective%20of%20the%2Cadministrator%20is%20guaranteed%20the%20access) [synopsis/synopsisstudentmanagementsystem#:~:text=The%20main%20ob](https://www.freeprojectz.com/premium-synopsis/synopsis-student-management-system#%3A~%3Atext%3DThe%20main%20objective%20of%20the%2Cadministrator%20is%20guaranteed%20the%20access) [jective%20of%20t he,administrator%](https://www.freeprojectz.com/premium-synopsis/synopsis-student-management-system#%3A~%3Atext%3DThe%20main%20objective%20of%20the%2Cadministrator%20is%20guaranteed%20the%20access) [20is%20guaranteed%20the%20access.](https://www.freeprojectz.com/premium-synopsis/synopsis-student-management-system#%3A~%3Atext%3DThe%20main%20objective%20of%20the%2Cadministrator%20is%20guaranteed%20the%20access)

# Conclusion

Flight management is all about people – but that implies managing the data about the Flight and passengers. New technologies emerge on a regular basis and this leads to Flight management systems becoming more advanced. Thus, there is a big chance for modern development companies to come up with an innovative approach and offer an efficient and scalable solution.