# ABSTRACT

This is a management system that is made in order to keep a track record of flights, staff, and updates on flight details. User can view all available flights and search based on source and destination airports. The design is so simple that the user won’t find any difficulties while working on it. Maintenance of all these information manually is a very complex task. Owing to the advancement of technology, organization of a Flight Management System is the need of the hour. The Flight Management has been designed to computerize and automate the operations performed over the information. This computerization of Flight management helps in many instances of its maintenances. It reduces the workload of the management as most of the manual work done is reduced and automated through the Digital system.

# **Contents**

TERM - 1

1. INTRODUCTION

1.1 PROJECT AIMS AND OBJECTIVES

1.2 BACKGROUND OF THE PROJECT

2. SYSTEM ANALYSIS

2.1 SOFTWARE REQUIREMENT SPECIFICATION

2.2 EXISTING Vs PROPOSED

2.3 HARDWARE & SOFTWARE SPECIFICATIONS

3. SYSTEM DESIGN

3.1 DATA FILE / TABLE DESIGN

3.2 MENU STRUCTURE

3.3 DATA FLOW DIAGRAMS

TERM - 2

4. SYSTEM IMPLEMENTATION

4.1 SOURCE CODE AND MODULE DESCRIPTION

4.2 SCREEN SHOTS

5. SYSTEM TESTING

5.1 UNIT TESTING

5.2 INTEGRATION TESTING

6. CONCLUSION

7. REFERENCES

# 1. INTRODUCTION

* 1. PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter.

The aims and objectives are as follows:

 User Management

 Flight management

 Option to view scheduled and cancelled flights for users.

 A Manager login page where manager can view and edit flights.

 A Manager login page where manager can manage all users except the admin.

 An Admin login page where admin can view and edit flights.

 An Admin login page where admin can manage all users.

 Option to search based on source and destination airports.

* 1. BACKGROUND OF THE PROJECT   
     The Flight Management System is a Python-based solution that allows users to quickly book flights and manage booking information, updates, and cancellations easily. It consolidates data from different airline carriers and thus provides all the necessary details and rates in real-time. In addition, administrators of flight data can also quickly view, create, and update any information about flights, bookings, routes, and schedules. A flight management system (FMS) is a fundamental component of a modern airliner's [avionics](https://en.wikipedia.org/wiki/Avionics). An FMS is a specialized computer system that automates a wide variety of in-flight tasks, reducing the workload on the flight crew to the point that modern civilian aircraft no longer carry [flight engineers](https://en.wikipedia.org/wiki/Flight_engineer) or [navigators](https://en.wikipedia.org/wiki/Navigator). A primary function is in-flight management of the flight plan. The modern FMS was introduced on the Boeing 767, though earlier methods did exist. Now, systems similar to FMS exist on aircraft as small as the Cessna 182. In its evolution an FMS has had many different sizes, capabilities and controls. However certain characteristics are common to all FMSs.

# 2. SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of flight Management System including software requirement specification (SRS) and comparison between existing and proposed system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs proposed provides a view of how the proposed system will be more efficient than the existing one.

2.1 SOFTWARE REQUIREMENT SPECIFICATION

Flight Management System is a computerized system which helps the user to manage the activities of the airport in an electronic format. It reduces the risk of paper work and other time-consuming activities in the airport. It can help user to manage the transaction of record more effectively and, in a time, saving manner.

PROBLEM STATEMENT:

The problem that occurred before having computerized system includes:

* Manpower
* High cost
* Complexity
* Entering record and Time consuming
* Searching the record  
   Due to absence of unique identification of a flight, the searching of record takes   
   much time and increases the time wastage.
* Maintenance
* Low Accuracy
* Storing

Maintaining and managing data is very costly and time consuming, because there are many documents that have to be maintained by each branch and copies have to be transferred to relative branches. Transfer of information within the branches is costly and time consuming.

2.2 EXISTING Vs PROPOSED

Existing System:

The existing system is a static system and has called for improvements in development of this Project. The passengers are unable to directly use the system and have to contact the officials. The system is very time consuming and lazy. This system is more prone to errors and sometimes the approach to various problems is unstructured. If any old data or information is to be fetched then it is a great problem for user to get the information in short span of time as to get information from files is not an easy task. As everything is done manually, so if any record is misplaced then agency has to take full responsibility.

Proposed System:

To solve the inconveniences as mentioned in the existing system, a new system is proposed. The proposed system contains the following features.

 Get flight between specific locations

 Check timing of flights

 Check cost of flights available

 Update flight details

2.3 SYSTEM SPECIFICATIONS

HARDWARE SPECIFICATIONS

The following is the hardware specification of the system on which the software has been developed: -

Operating System : Windows VISTA/ 7 /10 or UBUNTU

Windows 7 is used as the operating system as it is stable and supports more features and is more user friendly.

Machine : Pentium Dual Core Processor 2.6 GHz or above,

2 GB RAM or above,

500 GB Hard Disk or above

We used Intel core i5 2nd generation-based system, it is faster than other processors and provides reliable and stable performance and we can run our pc for longtime. By using this processor, we can keep on developing our project without any worries. 4GB RAM is used as it will provide fast reading and writing capabilities and will support in processing.

SOFTWARE SPECIFICATIONS

Front End Used : PYTHON 3.8.0 or above

Backend Used : Data Files

# 3. SYSTEM DESIGN

3.1 DATA FILE

Data Files Design

ADMIN.CSV

This file contains the details about the admin users.

USER\_ID

PASSWORD

MANAGER.CSV

This file contains the details about the manager users.

USER\_ID

PASSWORD

USER.CSV

This file contains the details about the standard users.

USER\_ID

PASSWORD

SCHEDULED.CSV

This file contains the details about the scheduled flights.

FLIGHT\_NO

ETA

DESTINATION

STATUS

SEATS AVAILABLE

COST

CANCELLED.CSV

This file contains the details about the cancelled flights.

FLIGHT\_NO

ETA

DESTINATION

STATUS

3.2 MENU STRUCTURE

ADMIN

VIEW FLIGHTS

ADD FLIGHTS

SEARCH FLIGHT

CANCEL FLIGHTS

USER MANAGEMENT

VIEW USERS

DELETE USERS

ADD USERS

EXIT

MANAGER

VIEW FLIGHTS

ADD FLIGHTS

CANCEL FLIGHTS

SEARCH FLIGHT

USER MANAGEMENT

VIEW USERS

DELETE USERS

ADD USERS

EXIT

USERS

VIEW FLIGHTS

SEARCH FLIGHT

EXIT

3.3 DATA FLOW DIAGRAMS

ADMIN MANAGEMENT

CANCELLED FLIGHT MANAGEMENT

MANAGER MANAGEMENT

SCHEDULED FLIGHT MANAGEMENT

USER MANAGEMENT

ADMIN MANAGEMENT

2

VIEW ADMINS

1

ADD ADMIN

3

DELETE ADMIN

ADMIN.CSV

MANAGER MANAGEMENT

2

VIEW MANAGER

1

ADD MANAGER

3

DELETE MANAGER

MANAGER.CSV

USER MANAGEMENT

2

VIEW USERS

1

ADD USER

3

DELETE USERS

USER.CSV

3

DELETE FLIGHT

SCHEDULED FLIGHT MANAGEMENT

2

MODIFY FLIGHT

1

ADD FLIGHT

4

LIST FLIGHT

SCHEDULED.CSV

5

SEARCH FLIGHT

2

ADD FLIGHT

CANCELLED FLIGHT MANAGEMENT

1

LIST FLIGHT

CANCELLED.CSV

# 4. SYSTEM IMPLEMENTATION

4.1 SOURCE CODE

import getpass

import warnings

import csv

import time

import pickle

import os

#Login

def login():

warnings.filterwarnings("ignore")

print()

print("\t\t\t\t\tWELCOME TO FLIGHT MANAGEMENT SYSTEM")

print()

time.sleep(1)

print("\t\t\t\t\tPLEASE ENTER YOUR LOGIN DETAILS")

time.sleep(1)

for i in range(0,3):

UID=input("\n Enter the User Name : ")

PWD=getpass.getpass(prompt="\n Enter the User Password : ")

global f

filein=open('ADMIN.csv','r')

reader=csv.reader(filein)

f=0

for row in reader:

if UID==row[0] and PWD==row[1]:

f=1

filein.close()

admins()

filein=open('MANAGER.csv','r')

reader=csv.reader(filein)

for row in reader:

if UID==row[0] and PWD==row[1]:

f=2

filein.close

managers()

filein=open('USER.csv','r')

reader=csv.reader(filein)

for row in reader:

if UID==row[0] and PWD==row[1]:

f=3

filein.close()

users()

if(f==0):

time.sleep(1)

print("\n... Entered User Name / Password is wrong ...")

time.sleep(1)

print(2-i,"attempts remaing")

if i==2:

time.sleep(1)

print("Sorry you have exhausted your attempts")

time.sleep(1)

print("Please try again later")

time.sleep(1)

print("... Closing Window ...")

time.sleep(3)

exit()

else:

print("\n System Starting.... \a\a\a\a\a\a")

#USER MANAGEMENT

#For Viewing all users

def view\_all():

filein=open('ADMIN.csv','r')

reader=csv.reader(filein)

time.sleep(1)

print("Admins are :")

for row in reader:

print(row[0],"\t\t",end='')

print('\n')

time.sleep(1)

filein.close()

filein=open('MANAGER.csv','r')

reader=csv.reader(filein)

time.sleep(1)

print("Managers are :")

for row in reader:

print(row[0],"\t\t",end='')

print('\n')

time.sleep(1)

filein.close()

filein=open('USER.csv','r')

reader=csv.reader(filein)

time.sleep(1)

print("Users are :")

for row in reader:

print(row[0],"\t\t",end='')

print('\n')

time.sleep(1)

filein.close()

#For adding admin

def add\_admin():

try:

fileout = open("ADMIN.csv", "r")

fileout.close()

except IOError:

time.sleep(1)

print ("\n File doesn't exist")

fileout = open("ADMIN.csv", "w",newline='')

fields=["\_\_ADMINS\_\_","PWD"]

writer = csv.writer(fileout,delimiter=',')

writer.writerow(fields)

fileout.close()

time.sleep(1)

uid=input("UID : ")

time.sleep(1)

pwd=input("PWD : ")

fileout = open("ADMIN.csv", "a+",newline='')

writer = csv.writer(fileout,delimiter=',')

writer.writerow([uid,pwd])

fileout.close()

#For deleting admin

def del\_admin():

found=0

time.sleep(1)

pname=input("\n Enter the user to be removed: ")

while True:

if pname=="admin":

time.sleep(1)

print("Cannot delete Main Admin")

time.sleep(1)

del\_admin()

break

else:

filein=open('ADMIN.csv','r+')

reader=csv.reader(filein)

time.sleep(1)

print("User ID\t\tPassword")

time.sleep(1)

for row in reader:

if pname in row:

for col in row:

print(col,"\t\t\t",end='')

found=1

print('\n')

time.sleep(1)

break

filein.close()

if found==1:

filein=open('ADMIN.csv','r')

fileout=open('temp\_ADMIN.csv','w',newline='')

writer = csv.writer(fileout,delimiter=',')

reader=csv.reader(filein)

for row in reader:

if row[0]==pname:

pass

else:

writer.writerow(row)

filein.close()

fileout.close()

os.remove("ADMIN.csv")

os.rename("temp\_ADMIN.csv","ADMIN.csv")

time.sleep(1)

print("\n Deletion completed........")

time.sleep(1)

else:

time.sleep(1)

print("\n Entered Name not found in search : ")

time.sleep(1)

del\_admin()

# For viewing admin

def view\_admin():

filein=open('ADMIN.csv','r')

reader=csv.reader(filein)

time.sleep(1)

for row in reader:

print(row[0],"\t\t",end='')

print('\n')

time.sleep(1)

filein.close()

#For adding manager

def add\_manager():

try:

fileout = open("MANAGER.csv", "r")

fileout.close()

except IOError:

time.sleep(1)

print ("\n File doesn't exist")

fileout = open("MANAGER.csv", "w",newline='')

fields=["\_\_\_","PWD"]

writer = csv.writer(fileout,delimiter=',')

writer.writerow(fields)

fileout.close()

time.sleep(1)

uid=input("UID : ")

time.sleep(1)

pwd=input("PWD : ")

fileout = open("MANAGER.csv", "a",newline='')

writer = csv.writer(fileout,delimiter=',')

writer.writerow([uid,pwd])

fileout.close()

#For deleting manager

def del\_manager():

found=0

pname=input("\n Enter the user to be removed : ")

filein=open('MANAGER.csv','r+')

reader=csv.reader(filein)

time.sleep(1)

print("User ID\t\tPassword")

for row in reader:

if pname in row:

for col in row:

print(col,"\t\t\t",end='')

found=1

print('\n')

time.sleep(1)

filein.close()

if found==1:

filein=open('MANAGER.csv','r')

fileout=open('temp\_MANAGER.csv','w',newline='')

writer = csv.writer(fileout,delimiter=',')

reader=csv.reader(filein)

for row in reader:

if row[0]==pname:

pass

else:

writer.writerow(row)

filein.close()

fileout.close()

os.remove("MANAGER.csv")

os.rename("temp\_MANAGER.csv","MANAGER.csv")

time.sleep(1)

print("\n Deletion completed........")

time.sleep(1)

else:

time.sleep(1)

print("\n Entered Name not found in search : ")

time.sleep(1)

del\_manager()

#For viewing manager

def view\_manager():

filein=open('MANAGER.csv','r')

reader=csv.reader(filein)

time.sleep(1)

for row in reader:

print(row[0],"\t\t",end='')

print('\n')

time.sleep(1)

filein.close()

#For adding user

def add\_user():

try:

fileout = open("USER.csv", "r")

fileout.close()

except IOError:

time.sleep(1)

print ("\n File doesn't exist")

fileout = open("USER.csv", "w",newline='')

fields=["\_\_USERS\_\_","PWD"]

writer = csv.writer(fileout,delimiter=',')

writer.writerow(fields)

fileout.close()

time.sleep(1)

uid=input("UID : ")

time.sleep(1)

pwd=input("PWD : ")

fileout = open("USER.csv", "a",newline='')

writer = csv.writer(fileout,delimiter=',')

writer.writerow([uid,pwd])

fileout.close()

#For deleting user

def del\_user():

found=0

pname=input("\n Enter the user to be removed : ")

filein=open('USER.csv','r+')

reader=csv.reader(filein)

time.sleep(1)

print("User ID\t\tPassword")

for row in reader:

if pname in row:

for col in row:

print(col,"\t\t\t",end='')

found=1

print('\n')

time.sleep(1)

filein.close()

if found==1:

filein=open('USER.csv','r')

fileout=open('temp\_USER.csv','w',newline='')

writer = csv.writer(fileout,delimiter=',')

reader=csv.reader(filein)

for row in reader:

if row[0]==pname:

pass

else:

writer.writerow(row)

filein.close()

fileout.close()

os.remove("USER.csv")

os.rename("temp\_USER.csv","USER.csv")

time.sleep(1)

print("\n Deletion completed........")

time.sleep(1)

else:

time.sleep(1)

print("\n Entered Name not found in search : ")

time.sleep(1)

del\_user()

#For viewing user

def view\_user():

filein=open('USER.csv','r')

reader=csv.reader(filein)

time.sleep(1)

for row in reader:

print(row[0],"\t\t",end='')

print('\n')

time.sleep(1)

filein.close()

#For viewing flights

def view\_flights():

time.sleep(1)

print("SCHEDULED FLIGHTS", end='\n\n')

time.sleep(1)

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

for row in reader:

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

print('\n')

time.sleep(1)

filein.close()

time.sleep(1)

print("CANCELLED FLIGHTS", end='\n\n')

time.sleep(1)

filein=open('cancelled.csv','r')

reader=csv.reader(filein)

for row in reader:

print("%20s %20s %20s %20s" %(row[0],row[1],row[2],row[3]))

print('\n')

time.sleep(1)

filein.close()

#For adding flights

def add\_flights():

fileout=open('scheduled.csv','a',newline='')

writer = csv.writer(fileout,delimiter=',')

time.sleep(1)

fno=input("Enter flight no:- ")

time.sleep(1)

eta=input("Enter time of arrival:- ")

time.sleep(1)

dest=input("Enter destination:- ")

time.sleep(1)

stat=input("Enter status:- ")

time.sleep(1)

seats=input("Enter the number of seats:- ")

time.sleep(1)

cst=input("Enter the cost:- ")

time.sleep(1)

writer.writerow([fno,eta,dest,stat,seats,cst])

fileout.close()

time.sleep(1)

#For cancelling flights

def cancel\_flights():

found=0

time.sleep(1)

fno=input("Enter the number of the flight to be cancelled:- ")

filein=open('scheduled.csv','r+')

reader=csv.reader(filein)

for row in reader:

if fno in row:

for col in row:

found=1

filein.close()

if(found==0):

time.sleep(1)

print('flight not found')

cancel\_flights()

time.sleep(1)

else:

fileout=open('cancelled.csv','a')

filein=open('scheduled.csv','r')

writer=csv.writer(fileout,delimiter=',')

reader=csv.reader(filein)

for row in reader:

if row[0]==fno:

l=[row[0],row[1],row[2],"cancelled"]

writer.writerow(l)

fileout.close()

filein.close()

filein=open('scheduled.csv','r')

fileout=open('temp\_scheduled.csv','w',newline='')

writer = csv.writer(fileout,delimiter=',')

reader=csv.reader(filein)

for row in reader:

if row==[]:

pass

elif row[0]==fno:

pass

else:

writer.writerow(row)

filein.close()

fileout.close()

os.remove("scheduled.csv")

os.rename("temp\_scheduled.csv","scheduled.csv")

time.sleep(1)

print("Flight cancelled successfully")

time.sleep(1)

def sort\_helper\_by\_cost(x):

return int(x[-1])

def sort\_helper\_by\_arrival(x):

if x[3] == 'Scheduled':

return 2

return 1

#For sorting flights by destination

def sort\_dest\_admins():

found=0

time.sleep(1)

dest=input('Enter the destination: ')

while True:

time.sleep(1)

type = input("""

To sort flights by cost press 1

To sort flights by status of arrival press 2

To return to control panel press 0

""")

if type.isnumeric():

type = int(type)

if type==1 or type==2 or type==0:

break

else:

time.sleep(1)

print("Invalid input. Please Try Again")

else:

time.sleep(1)

print("Invalid input. Please try again")

if type == 1:

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

temp\_list = []

for row in reader:

if row[2]==dest:

temp\_list.append(list(row))

found += 1

if found==0:

time.sleep(1)

print("Currently there are No Flights for",dest)

else:

temp\_list.sort(key=sort\_helper\_by\_cost)

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %("Flight No","ETA","Destination","Status","Seats Available","Cost"))

for row in temp\_list:

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

filein.close()

elif type == 2:

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

temp\_list = []

for row in reader:

if row[2]==dest:

temp\_list.append(list(row))

found += 1

if found==0:

time.sleep(1)

print("Currently there are no flights for",dest)

else:

temp\_list.sort(key=sort\_helper\_by\_arrival)

temp\_list = temp\_list[::-1]

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %("Flight No","ETA","Destination","Status","Seats Available","Cost"))

for row in temp\_list:

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

filein.close()

elif type==0:

admins()

def admins():

global time

q=1

while q==1:

try:

time.sleep(1)

time.sleep(1)

option=int(input('''

To view Flight details press 1

To add a new flight press 2

To cancel a flight press 3

To manage users press 4

To search a Flight press 5

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if option==1:

try:

view\_flights()

except:

admins()

elif option==2:

add\_flights()

admins()

elif option==3:

cancel\_flights()

admins()

elif option==4:

try:

time.sleep(1)

user\_option=int(input('''

To view users press 1

To add new user press 2

To delete existing user press 3

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if user\_option==1:

try:

time.sleep(1)

view\_option=int(input('''

To view admins press 1

To view managers press 2

To view users press 3

To view all users press 4

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if view\_option==1:

view\_admin()

elif view\_option==2:

view\_manager()

elif view\_option==3:

view\_user()

elif view\_option==4:

view\_all()

elif view\_option==0:

admins()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif user\_option==2:

try:

time.sleep(1)

add\_user\_option=int(input('''

To add admin press 1

To add manager press 2

To add user press 3

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

time.sleep(3)

exit()

if add\_user\_option==1:

add\_admin()

admins()

elif add\_user\_option==2:

add\_manager()

admins()

elif add\_user\_option==3:

add\_user()

admins()

elif add\_user\_option==0:

admins()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif user\_option==3:

try:

time.sleep(1)

add\_user\_option=int(input('''

To remove admin press 1

To remove manager press 2

To remove user press 3

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if add\_user\_option==1:

del\_admin()

admins()

elif add\_user\_option==2:

del\_manager()

admins()

elif add\_user\_option==3:

del\_user()

admins()

elif add\_user\_option==0:

admins()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif user\_option==0:

admins()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif option==5:

sort\_dest\_admins()

admins()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

def sort\_dest\_managers():

found=0

time.sleep(1)

dest=input('Enter the destination: ')

while True:

time.sleep()

type = input("""

To sort flights by cost press 1

To sort flights by status of arrival press 2

To return to control panel press 0

Press any other key to exit

""")

if type.isnumeric():

type = int(type)

if type in (0,2):

break

else:

time.sleep(1)

print("Invalid input. Please Try Again")

else:

time.sleep(1)

print("Invalid input. Please try again")

if type == 1:

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

temp\_list = []

for row in reader:

if row[2]==dest:

temp\_list.append(list(row))

found += 1

if found==0:

time.sleep(1)

print("Currently there are No Flights for",dest)

else:

temp\_list.sort(key=sort\_helper\_by\_cost)

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %("Flight No","Time","Destination","Status","Seats Available","Cost"))

for row in temp\_list:

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

filein.close()

elif type == 2:

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

temp\_list = []

for row in reader:

if row[2]==dest:

temp\_list.append(list(row))

found += 1

if found==0:

time.sleep(1)

print("currently there are no flights for",dest)

else:

temp\_list.sort(key=sort\_helper\_by\_arrival)

temp\_list = temp\_list[::-1]

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %("Flight No","Time","Destination","Status","Seats Available","Cost"))

for row in temp\_list:

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

filein.close()

elif type==0:

managers()

def managers():

global time

q=1

while q==1:

try:

time.sleep(1)

time.sleep(1)

option=int(input('''

To view Flight details press 1

To add a new flight press 2

To cancel a flight press 3

To manage users press 4

To search a Flight press 5

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if option==1:

try:

view\_flights()

except:

managers()

elif option==2:

add\_flights()

managers()

elif option==3:

cancel\_flights()

managers()

elif option==4:

try:

time.sleep(1)

user\_option=int(input('''

To view users press 1

To add new user press 2

To delete existing user press 3

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if user\_option==1:

try:

time.sleep(1)

view\_option=int(input('''

To view admins press 1

To view managers press 2

To view users press 3

To view all users press 4

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if view\_option==1:

view\_admin()

elif view\_option==2:

view\_manager()

elif view\_option==3:

view\_user()

elif view\_option==4:

view\_all()

elif view\_option==0:

managers()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif user\_option==2:

try:

time.sleep(1)

add\_user\_option=int(input('''

To add manager press 1

To add user press 2

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if add\_user\_option==1:

add\_manager()

managers()

elif add\_user\_option==2:

add\_user()

managers()

elif add\_user\_option==0:

managers()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif user\_option==3:

try:

time.sleep(1)

add\_user\_option=int(input('''

To remove manager press 1

To remove user press 2

To return to control panel press 0

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if add\_user\_option==1:

del\_manager()

managers()

elif add\_user\_option==2:

del\_user()

managers()

elif add\_user\_option==0:

managers()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif user\_option==0:

managers()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

elif option==5:

sort\_dest\_managers()

managers()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

def sort\_dest\_users():

found=0

time.sleep(1)

dest=input('Enter the destination: ')

while True:

time.sleep()

type = input("""

To sort flights by cost press 1

To sort flights by status of arrival press 2

To return to control panel press 0

Press any other key to exit

""")

if type.isnumeric():

type = int(type)

if type in (0,2):

break

else:

time.sleep(1)

print("Invalid input. Please Try Again")

else:

time.sleep(1)

print("Invalid input. Please try again")

if type == 1:

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

temp\_list = []

for row in reader:

if row[2]==dest:

temp\_list.append(list(row))

found += 1

if found==0:

time.sleep(1)

print("Currently there are No Flights for",dest)

else:

temp\_list.sort(key=sort\_helper\_by\_cost)

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %("Flight No","Time","Destination","Status","Seats Available","Cost"))

for row in temp\_list:

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

filein.close()

elif type == 2:

filein=open('scheduled.csv','r')

reader=csv.reader(filein)

temp\_list = []

for row in reader:

if row[2]==dest:

temp\_list.append(list(row))

found += 1

if found==0:

time.sleep(1)

print("currently there are no flights for",dest)

else:

temp\_list.sort(key=sort\_helper\_by\_arrival)

temp\_list = temp\_list[::-1]

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %("Flight No","Time","Destination","Status","Seats Available","Cost"))

for row in temp\_list:

time.sleep(1)

print("%20s %20s %20s %20s %20s %20s" %(row[0],row[1],row[2],row[3],row[4],row[5]))

filein.close()

elif type==0:

users()

def users():

global time

q=1

while q==1:

try:

time.sleep(1)

time.sleep(1)

option=int(input('''

To view Flight details press 1

To search a Flight press 2

Press any other key to exit

'''))

except:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

if option==1:

try:

view\_flights()

except:

users()

elif option==2:

sort\_dest\_users()

users()

else:

time.sleep(1)

print('... Thank you...')

time.sleep(1)

print('...Have a nice day...')

time.sleep(1)

print('... Closing Window ...')

time.sleep(2)

exit()

login()

4.2 MODULE DESCRIPTION

1. VIEW FLIGHTS

It is used to view all the flights including scheduled and cancelled. It is available for all

the users using the program.

1. ADD FLIGHTS  
     
   It is used to add flights to the scheduled list. It can be used by manager and admin users.
2. CANCEL FLIGHTS  
     
   It is used to cancel flights. It removes the flight from the scheduled list and adds it to cancelled list. It can be used by manager and admin users only.
3. SEARCH FLIGHTS

It is used to sort flights according to the user’s preference. We can sort it by cost and destination or status and destination. It can be used by all the users.

1. MANAGE USERS

VIEW USERS

VIEW ADMIN : This is used to view the admin users .   
VIEW MANAGER : This is used to view the manager users .  
VIEW USER : This is used to view the normal users  
VIEW ALL USERS : This is used to view all users

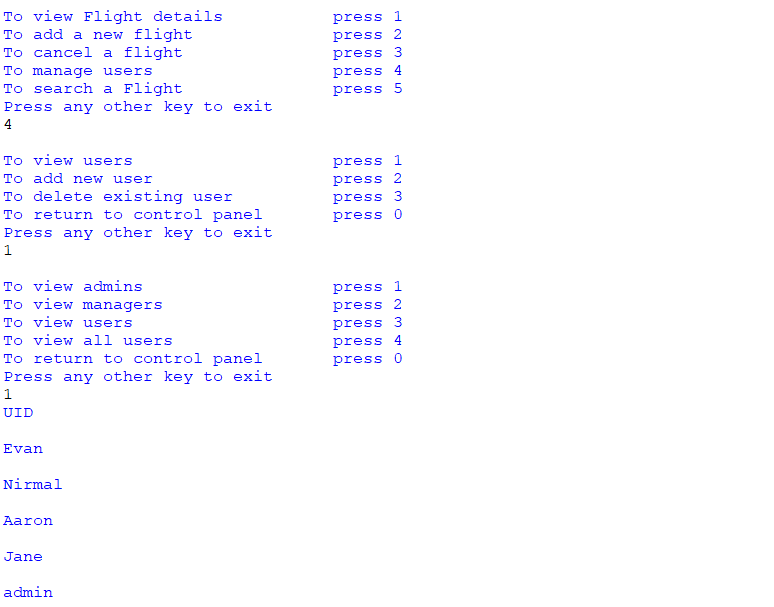
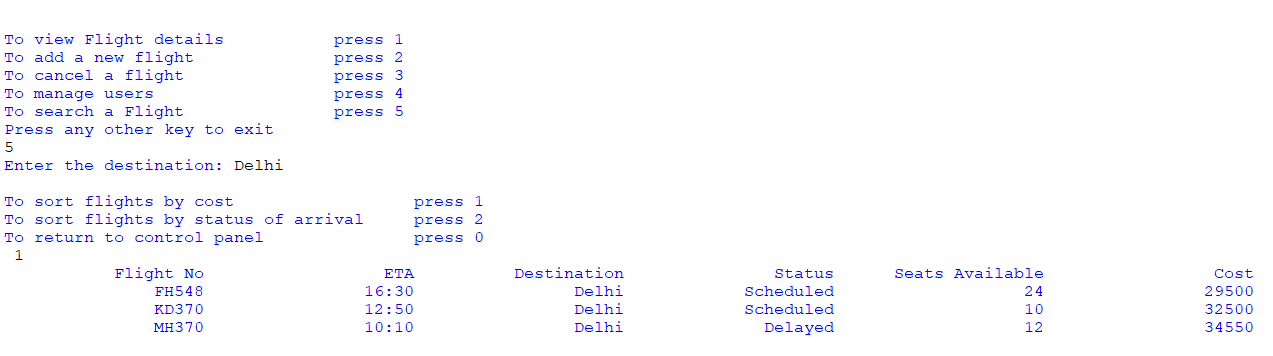
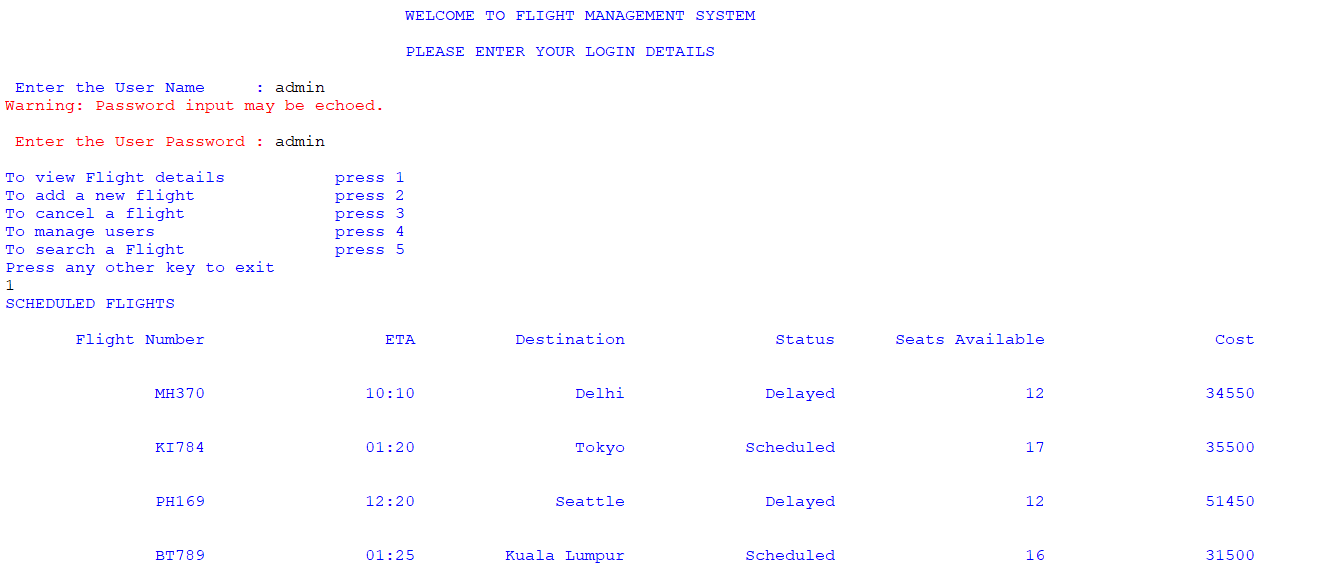
ADD USERS

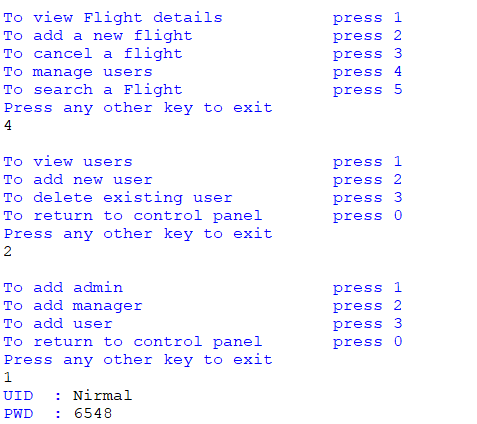
ADD ADMIN : This is used to add admin users  
ADD MANAGER : This is used to add manager users  
ADD USER : This is used to add normal users

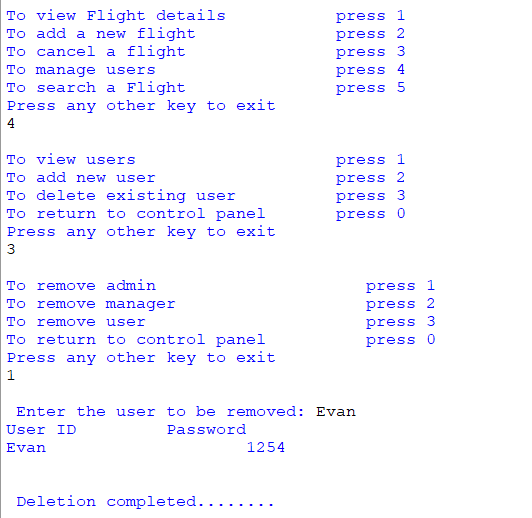
DELETE USERS

DELETE ADMIN : This is used to delete admin users  
DELET MANAGER : This is used to delete manager users  
DELETE USER : This is used to delete normal users

4.3 SCREEN SHOTS







# 5. SYSTEM TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product, with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the management to appreciate and understand the risks during the implementation of the software.

The aim of the system testing process was to determine all defects in our project. The program was subjected to a series of trial operations with test inputs and various observations were made and based on these observations, changes were made and again tested for better results. Our Project went through two levels of testing

1.Unit testing   
2. Integration testing

5.1 UNIT TESTING

Unit testing was undertaken when a module has been created and successfully reviewed. In order to test a single module, we need to provide a complete working environment.

5.2 INTEGRATION TESTING

After integrating the entire modules developed, we performed various checks by providing different set of test input. The primary objective is to test all the modules in order to ensure that no errors are occurring when one module invokes the other module.

# 6. CONCLUSION

The software for flight management is found to be working efficiently.   
The software appears very flexible since it is menu driven with user-friendly screens.  
No Formal programming knowledge is required for the user.   
Also, the user is not burdened with data storing and data retrieval procedures as both are done internally.

REFERENCES

1. Computer Science with Python by Sumita Arora
2. Python The Complete Reference by Martin C Brown
3. Programming & Problem Solving Through Python by Sathish Jain & Shashi Singh
4. Python for Beginners by Prof. Rahul E. Borate
5. Computer Science with Python Language Made Simple by Sathish Jain & Shashi Singh
6. Wikipedia, Flight Management