

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018, Karnataka, INDIA



MINI PROJECT REPORT

On

“Courier Complaint System”

Submitted in partial fulfillment of the requirements for the VI Semester

Bachelor of Engineering
In
INFORMATION SCIENCE AND ENGINEERING

For the Academic year
2018-2019

BY

Manish Kumar
Karan Kumbhani

1PE16IS053
1PE16IS044



Department of Information Science and Engineering
PESIT Bangalore South Campus
Hosur Road, Bengaluru-560100

PESIT Bangalore South Campus
Hosur Road, Bengaluru-560100
Department of Information Science and Engineering



CERTIFICATE

*This is to certify that the project work entitled "**Courier Complaint System**" is a bonafide work carried out by **Manish Kumar, Karan Kumbhani** and bearing USNs 1PE16IS053, 1PE16IS044 respectively in partial fulfilment of DBMS mini project (15CSL58) in 5th semester of Degree of Bachelors (**Bachelors of Engineering**) in **Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the year 2018-2019.*

It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the Report. The project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for said

Signatures:

Project Guide
Asst./Associate Professor,
Dept. of ISE
PESIT-BSC, Bengaluru

Head Dept of ISE
Dr. Annapurna D
PESIT-BSC, Bengaluru

External Viva

Name of the Examiners

- 1.
- 2.

Signature with date

ACKNOWLEDGEMENT

We would like to thank our Lab Teachers, Prof. Kakoli, Prof. Reshma and Prof. Pragya for their aid and evaluation over the course of completion of this project. We would also like to thank our University, Visvesvaraya Technological University for the opportunity to make this project which has given us the chance to learn about these topics. Additionally, we would like to thank our college, PES Institute of Technology for allowing us to use the college resources. Finally, I would like to thank our friends and family for their ceaseless and tireless support throughout this project.

CONTENTS

ABSTRACT	III
Chapter 1: Introduction	1
1.1. Introduction	1
1.1.1. Purpose.....	1
1.1.2. Scope.....	1
1.2. Literature Survey.....	2
1.3. Problem Definition.....	2
1.4. FS Technique Used.....	2
Chapter 2: Software Requirement Specifications.....	4
2.1. Software Requirements Specifications.....	4
2.2. Operating Environment.....	4
2.2.1. Hardware Requirements.....	4
2.2.2. Software Requirements.....	4
2.3. Functional Requirements	5
2.4. Non-functional Requirements.....	5
Chapter 3: Design.....	6
3.1. High Level Diagram.....	6
3.2. Use Case Diagram.....	6
3.3. Data Flow Diagrams	7
3.4. Modules.....	7
Chapter 4: Implementation.....	9
4.1. Implementation	9
4.2. Programming Language Selection	9
4.3 Important Code Snippets	10
Chapter 5: Results Snaps	16
Chapter 6: Conclusion	21

6.1. Conclusion.....	21
6.2. Limitations of the Project.....	21
6.3. Future Enhancements	21
References.....	22

ABSTRACT

Courier Complaint System is a desktop application which is designed for the ultimate user satisfaction and comfort. It allows the employees of a courier company to lodge complaints against the delivery services through the click of a button. The user of this system has the option to lodge a complaint, modify the status of the complaint, search for the complaint using complaint ID or name of the complainer and cancel the complaint.

The application makes use of files to maintain records related to complaint data, status of the complaint and the timing of the complaint.

The file structures technique used is Indexing(Primary indexing and Secondary indexing).

Python is used to design the backend and the file structure system. Tkinter has been used to develop the GUI.

Chapter 1

Introduction

1.1. Introduction

Courier Complaint System is a file system project used to report complaints against the courier company. This project is mainly useful for different courier delivery service providers across the world. It will help to manage all the complaint related activities in customer relations department of the company using computers. Currently all the work are done manually, by computerizing all the activities inside a Courier Company it can be managed easily and efficiently.

There is no record of complaints that can be accessed by a Company officials, except the manual records relating to the customer relations department. The goals of the system are to facilitate collection, storage, retrieval, analysis, transfer and sharing of data and information at customer relations department of a local office and the centralized office of the company seamlessly.

1.1.1. Purpose

This project is aimed to reduce the manual work involved in data maintenance in the Complaint system and automates the whole system. This project is developed mainly to simplify the manual work and allows smooth administration of the operations of a Courier Company. The purpose of the project is to computerize the administrative operations of a Courier Company and to develop software which is user friendly, simple, fast, and cost – effective. It deals with the collection of Users, Employees, Delivery agents and Complaint and delivery information, etc. Traditionally, it was done manually. The main function of the system is to enter and lodge complaints and retrieve these details as and when required, and also to manipulate these details meaningfully.

1.1.2. Scope

The project provides a very simple application which simplifies the manual work done by the customer relations team of a Courier Company. This application saves the data of complaints in the files. Allows users to search for complaint, modify or delete the existing complaints.

Our project allows users to view the lodged complaint's data stored in the files and to see the statistics.

1.2. Literature Survey

The desktop application Courier Complaint System helps company admins easily lodge and modify complaints. An easy to use interface which uses indexing to search the respective complaints' details and display or modify it. This application helps the admins save a lot of time and make the process of lodging and managing complaints more user friendly. Search Engine for Local Host is a standalone application that helps the user to find all the files in the Home Directory that contains the input keyword through an intuitive, easy to use interface. This application helps the user saves a lot of time, which the user would spend in going through all the files to check if the keyword is present in the file or not.

1.3. Statement of the Problem

Courier Complaint System is a desktop application which is designed for the ultimate user satisfaction and comfort. It allows the employees of a courier company to lodge complaints against the delivery services through the click of a button. The user of this system has the option to lodge a complaint, modify the status of the complaint, search for the complaint using complaint ID or name of the complainer and cancel the complaint.

1.4. FS Techniques Used

The concept of File Structure used in Courier Complaint System is Indexing(Primary and Secondary Indexing), and the data structure used is list(python) and dictionary(python). We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely.

Primary Indexing is a File structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing is defined based on its indexing attributes. Secondary Indexing can be built on any field of the data file,

or on combinations of fields. Secondary indexes will typically have multiple locations for a single key.

The reference field of a secondary index can also be an indirect reference to the location of the entry in the data file, through the primary key.

Chapter 2

Software Requirements Specification

2.1. Software Requirements Specification

The Software Requirements Specification (SRS) document will provide a detailed description of the steps, phases and design necessary for appointment management software. A clear understanding of the files and its' functionality will allow for the correct software system to be developed for the users of the software and will be used for the development of the future stages of the project. This SRS document will provide the foundation for the design, construction, implantation and testing of the code.

2.2: Operating Environment

The production ready software is meant to run on a variety of verified hardware and software. As such, many of the required dependencies are available cross platform, both for the front end as well as the backend.

Some of the verified software and hardware are specified below, along with software and hardware that are supposed to be compatible.

2.2.1: Hardware Requirements

The project has the following Hardware requirements:

1. A computer system with a suitable operating system.
2. Sufficient RAM.
3. A standard PS/2 keyboard
4. A generic PNP monitor

2.2.2: Software Requirements

1. The project has the following Software requirements:
2. A suitable operating system
3. A suitable web browser

4. Python 3.6 along with the following dependencies:
 - a. Pip : A package management system used to install used to install and manage software packages written in Python.
 - i. pip install tkinter
 - b. Tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications.

2.3 Functional Requirements

1. The admin can create a complaint and fill out all the details of the complaint.
2. The admin can search the complaint by typing the complaint ID.
3. The admin can also search the complaint by typing the complainer's name.
4. The admin can modify the status of an ongoing complaint.
5. The admin can delete a pre-existing complaint.

2.4. Non-Functional Requirements

1. Reliability: This application is a web-based application that runs on any device that has a browser.
2. Security: The system will be password-protected.
3. Portability: The system will be a screen-based application that can run in various environments.
4. Availability: The system shall be available during normal operating hours.
5. Performance: Search operations shall return results within five seconds.

Chapter 3

Design

3.1. High Level Design

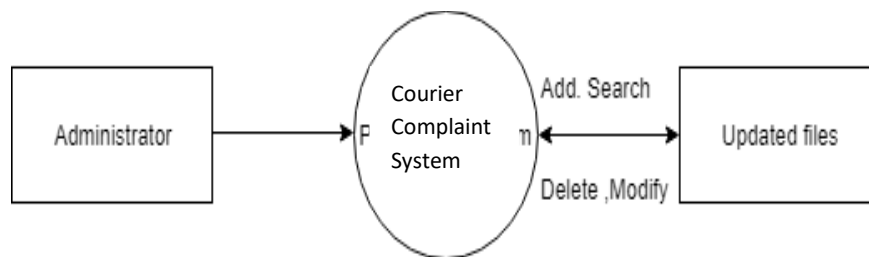


Fig. 3.1: High Level Design

3.2. Use Case Diagram

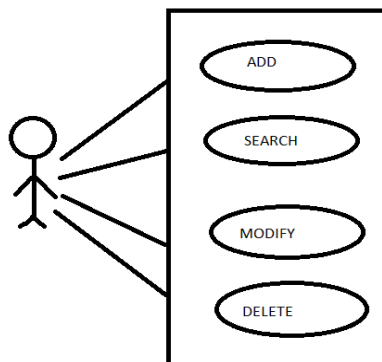


Fig. 3.2: Use Case Diagram

3.3. Data Flow Diagrams

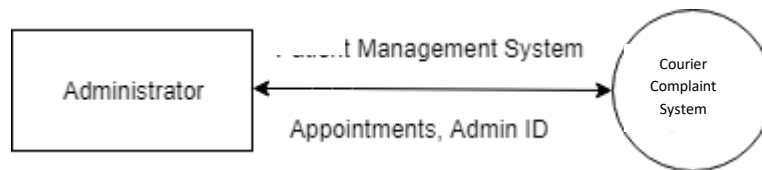


Fig. 3.3.1: Data flow Diagram (Level 0) for the application.

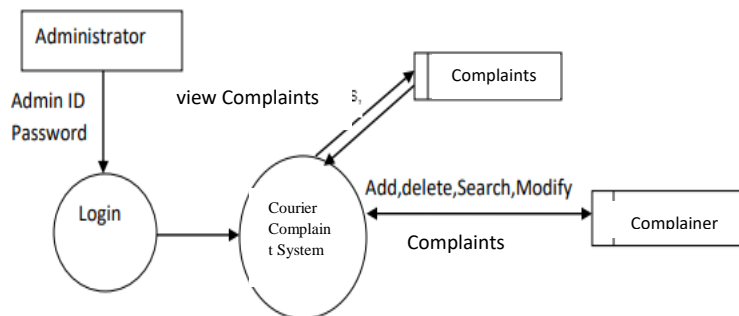


Fig. 3.3.2: Data flow Diagram (Level 1) for the application.

3.4. Modules

- **Simple Indexing** -Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done.
- **Secondary Indexing**- can be built on any field of the data file, or on combinations of fields. Secondary indexes will typically have multiple locations for a single key.
- **Search-** Search the index file for the specified Customer ID using Indexing and display the details or specified name using secondary indexing.
- **Insert-** Insert all the patient details into the patient record and create an entry for that patient id in the index file.
- **Modify-** Fetch the patient record of the respective Patient ID and update it with the new modified details.

- **Delete**-Search the index file for the Patient ID and fetch the respective patient record and delete it.

Chapter 4

Implementation

4.1. Implementation

The project is implemented in Python using primary indexing. It has a three-tier architecture with Front-End forming Presentation Tier, Python for the Logic Tier and file structures as the Database.

4.2. Programming Language Selection

Python 3.6 was used in the backend. Python is aspect-oriented (separates the program functionality and splits the program into separate modules). It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Libraries used:

re:

A *regular expression* is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are widely used in UNIX world.

os:

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

Tkinter:

Tk/Tcl provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter package, and its extension, the tkinter.tix and the tkinter.ttk modules.

4.7. Important Codes Snippets

4.7.1. Insert Record

```

|case_desc1=""
txt_filename="complaint.txt"
txt_fil = open(txt_filename, "a")
txt1 = open("complaint.txt", "r")
txt_indexname="index_file.idx"
n=len(sys.argv)
C_ID=sys.argv[1]
C_name=sys.argv[2]
C_against=sys.argv[3]
C_date=sys.argv[4]
C_time=sys.argv[5]

C_status=sys.argv[6]
flag=0
for x in txt1:
    if(C_ID in x):
        flag=1
if flag==0:
    for i in range(7,n):
        C_desc=case_desc1+sys.argv[i]+" "
        entry=C_ID+' '+'|'+C_name+'|'+C_against+'|'+C_date+'|'+C_time+'|'+C_desc+'|'
        txt_fil.write(entry)
    else:
        print("Already exists")
txt_fil.close()

def index_text_file():

    d={}
    dl={}
    comp_file = open("complaint.txt", "r")
    idx_file_r = open("index_file.idx", "r")
    l=0
    for line in comp_file:
        l=l+1
    for x in idx_file_r:
        list = x.split()
        dl = {list[0]:list[1]}
        d.update(dl)
    dl={C_ID:l}
    dl={C_ID:l}
    d.update(dl)
    idx_file_w = open("index_file.idx", "w")
    for i in sorted(d):
        #data =
        idx_file_w.write(i+" "+str(d[i])+"\n")
    print(d)
    comp_file.close()
    comp_file = open("complaint.txt", "r")
    sec_file=open("sec_index.txt", "w")
    secd={}
    secd2={}
    for line in comp_file:
        list=line.split("|")
        secd2={list[1]:list[0]}
        secd.update(secd2)
    for i in sorted(secd):
        sec_file.write(i+" "+str(secd[i])+"\n")
    print(secd)

index_text_file()

```


4.7.2. Search Record(Primary Indexing)

```
txt_file="complaint.txt"
idx_file="index_file.idx"
key=sys.argv[1]

record=[]
record=open(txt_file).readlines()

def search(txt_file,idx_file,key):
    flag=0
    idx_f=open(idx_file,"r")
    for line in idx_f:
        if re.match(key,line):
            flag=1
            l=line.split()
            n=len(l)
            txt_f=open(txt_file,"r")
            for i in range (1,n):
                c=int(l[i])
                #print(record[c-1])
            #return 1
            l2=record[c-1].split('|')
            print("\nComplaint ID:"+l2[0])
            print("Complainee's name:"+l2[1])
            print("Complaint against:"+l2[2])
            print("Complaint date:"+l2[3])
            print("Complaint timme:"+l2[4])
            print("complaint description:"+l2[5])
            print("Complaint status:"+l2[6]+"\\n")

            txt_f.close()

    if(flag==0):
        print("No such record exists")
    idx_f.close()

search(txt_file,idx_file,key)
```

4.7.3.Search Record(Secondary Indexing)

```
def sec_search(sec_file, key):
    flag=0
    sec_f=open(sec_file, "r")
    for line in sec_f:
        list=line.split()
        if re.match(key, list[0]):
            flag=1
            val=list[1]
            search(txt_file, idx_file, val)
            break
    if (flag==0):
        print("No such record exists")
    sec_f.close()

def search(txt_file, idx_file, key):
    flag=0
    idx_f=open(idx_file, "r")
    for line in idx_f:
        if re.match(key, line):
            flag=1
            l=line.split()
            n=len(l)
            txt_f=open(txt_file, "r")
            for i in range (1, n):
                c=int(l[i])
                l2=record[c-1].split('|')
                print("\nComplaint ID:"+l2[0])
                print("Complainee's name:"+l2[1])
                print("Complaint against:"+l2[2])
                print("Complaint date:"+l2[3])
                print("Complaint timme:"+l2[4])
                print("complaint description:"+l2[5])
                print("Complaint status:"+l2[6]+"\n")
            txt_f.close()
    if (flag==0):
        print("No such record exists")
    idx_f.close()

sec_search(sec_file, key)
```

4.7.4. Delete Record

```
def index_text_file(txt_filename, idx_filename,
    delimiter_chars=".,:;!?" ):
    txt_fil = open(txt_filename, "r")
    """
    Dictionary to hold words and the line numbers on which
    they occur. Each key in the dictionary is a word and the
    value corresponding to that key is a list of line numbers
    on which that word occurs in txt_filename.
    """
    word_occurrences = {}
    line_num = 0
    for lin in txt_fil:
        line_num += 1
        words=re.findall('C...',lin)
        words2 = [ word.strip(delimiter_chars) for word in words ]
        # Find and save the occurrences of each word in the line.
        for word in words2:
            if word in word_occurrences:
                word_occurrences[word].append(line_num)
            else:
                word_occurrences[word] = [ line_num ]
    if line_num < 1:
        print("No lines found in text file, no index file created.")
        txt_fil.close()
        sys.exit(0)
    # Display results.
    word_keys = word_occurrences.keys()
    #print "{} unique words found.".format(len(word_keys))
    word_keys = word_occurrences.keys()
    # Sort the words in the word_keys list.
    sorted(word_keys)
    # Create the index file.
    idx_fil = open(idx_filename, "w")
    for word in word_keys:
        line_nums = word_occurrences[word]
        idx_fil.write(word + " ")
        for line_num in line_nums:
            idx_fil.write(str(line_num) + " ")
        idx_fil.write("\n")
    txt_fil.close()
    idx_fil.close()
```

4.7.5. Modify Record

```
def search(txt_file,idx_file,key):
    flag=0
    idx_f=open(idx_file,"r")
    for line in idx_f:
        if re.match(key,line):
            flag=1
            l=line.split()
            n=len(l)
            txt_f=open(txt_file,"r")
            for i in range(1,n):
                c=int(l[i])
                l2=record[c-1].split('|')
            txt_f.close()
            n=int(l[1])
            l3=record[n-1].split('|')
            l3[6]=sys.argv[2]
            file1=open("complaint.txt","w")
            file_size=len(record)
            record2=[]
            for i in range(1,file_size+1):
                if(i!=n):
                    record2.append(record[i-1])
            file1.writelines(record2)
            file1.close()
            txt_fil=open("complaint.txt","a")
            C_ID=l3[0]
            C_name=l3[1]
            C_against=l3[2]
            C_date=l3[3]
            C_time=l3[4]
            C_desc=l3[5]
            C_stat=l3[6]
            entry=C_ID+'|'+C_name+'|'+C_against+'|'+C_date+'|'+C_time+'|'+C_desc+'|'+C_stat+'|'+'\n'
            txt_fil.write(entry)
            print("\nComplaint ID:"+l3[0])
            print("Complainee's name:"+l3[1])
            print("Complaint against:"+l3[2])
            print("Complaint date:"+l3[3])
            print("Complaint time:"+l3[4])
            print("Complaint description:"+l3[5])
            print("Complaint status:"+l3[6]+"\n")
            print("Complaint description:"+l3[5])
            print("Complaint status:"+l3[6]+"\n")
            print("\nRecord modified.\n")
            txt_fil.close()
            index_text_file("complaint.txt","index_file.idx")

    if(flag==0):
        print("No such record exists")
        return -1
    #return 1
    idx_f.close()

search("complaint.txt","index_file.idx",sys.argv[1])
```

4.7.6. GUI

```
#Main Program
root = Tk(className = "GUI")
frame=add_frame=search_frame=delete_frame=modify_frame = Frame(root, bg='blue')
root.minsize(200,200)
root.geometry("600x800")
root.title("Courier Complaint System")

add_patient_frame = add_case_frame = Frame(root)
search_patient_frame = Frame(root)
delete_patient_frame = Frame(root)
modify_patient_frame = Frame(root)

#Creating Main Menu Frame
button = ['Add', 'Search', 'Delete', 'Modify']
function = [add, search, delete, modify]
for i in range(len(button)) :
    Button(frame, text=button[i], command=function[i], height=1, width=20).pack()
#Display
display_frame(frame)
root.mainloop()
```

Chapter 5

Result Snaps

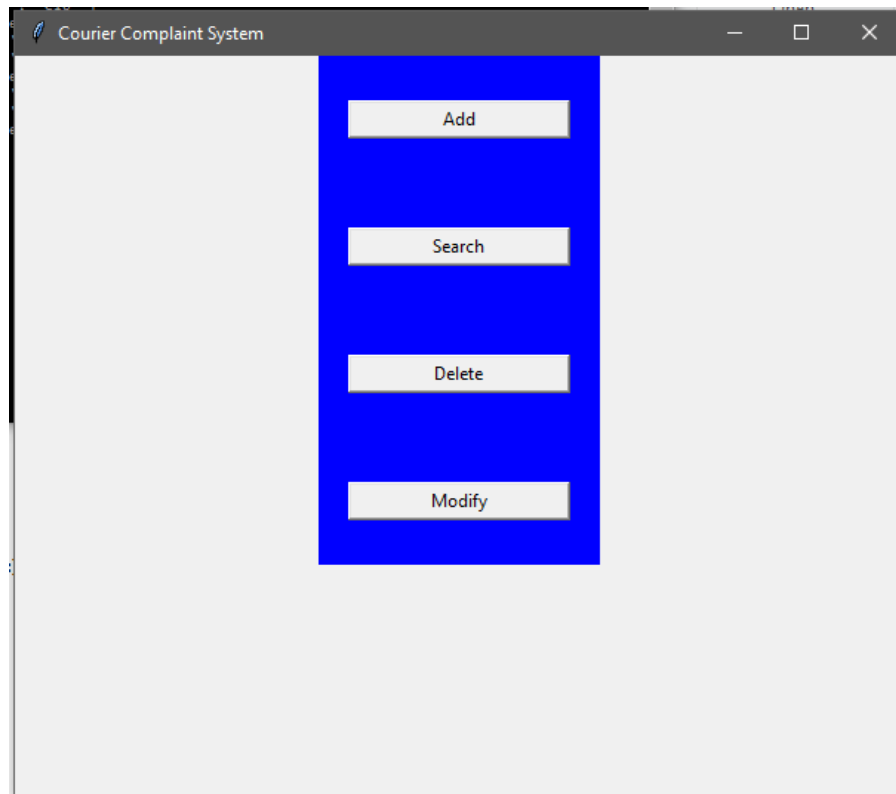


Figure 1 Main Page

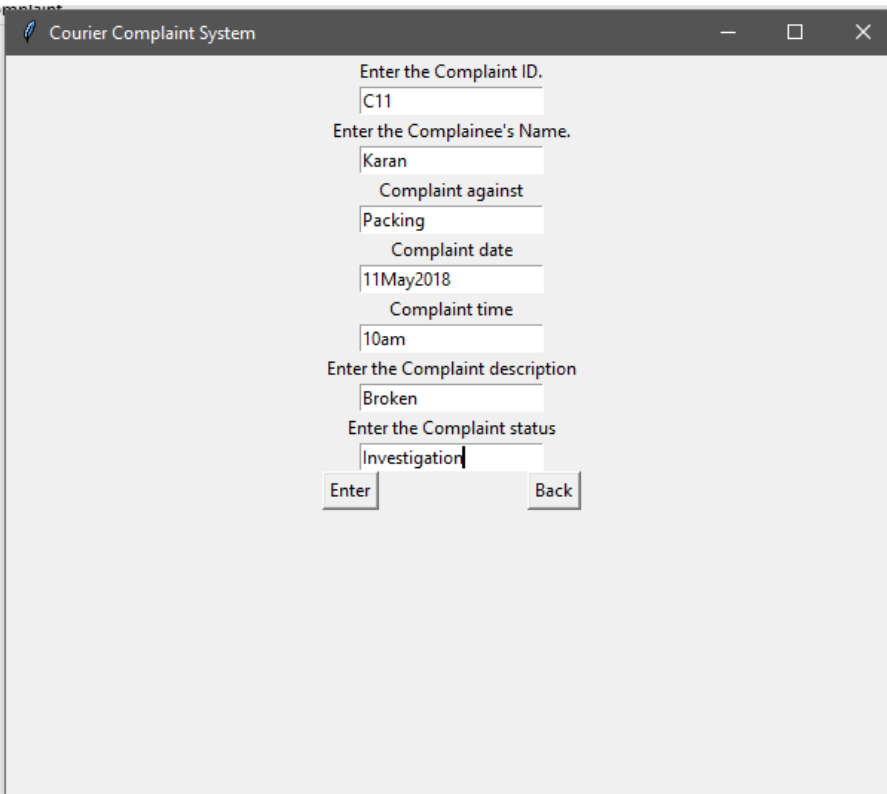
A screenshot of a web application window titled "Courier Complaint System". The window displays a form for adding a complaint. The form fields are as follows:
- "Enter the Complaint ID." with a text input containing "C11".
- "Enter the Complainee's Name." with a text input containing "Karan".
- "Complaint against" with a text input containing "Packing".
- "Complaint date" with a text input containing "11May2018".
- "Complaint time" with a text input containing "10am".
- "Enter the Complaint description" with a text input containing "Broken".
- "Enter the Complaint status" with a text input containing "Investigation".
At the bottom of the form, there are two buttons: "Enter" and "Back".

Figure 2 Add Complaint

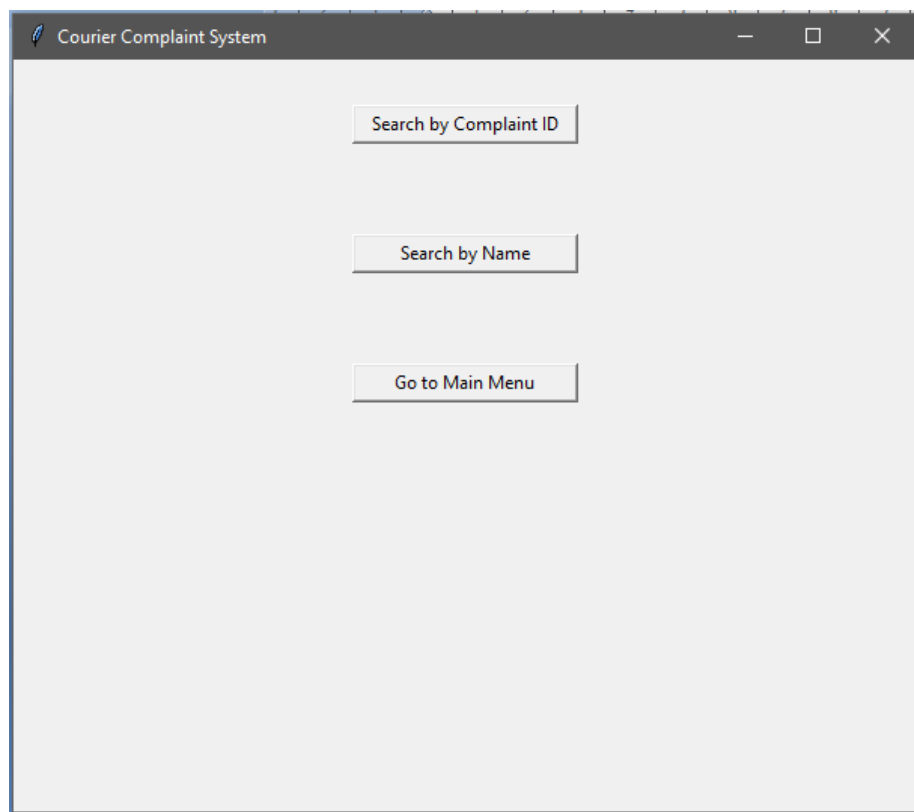


Figure 3 Search Window

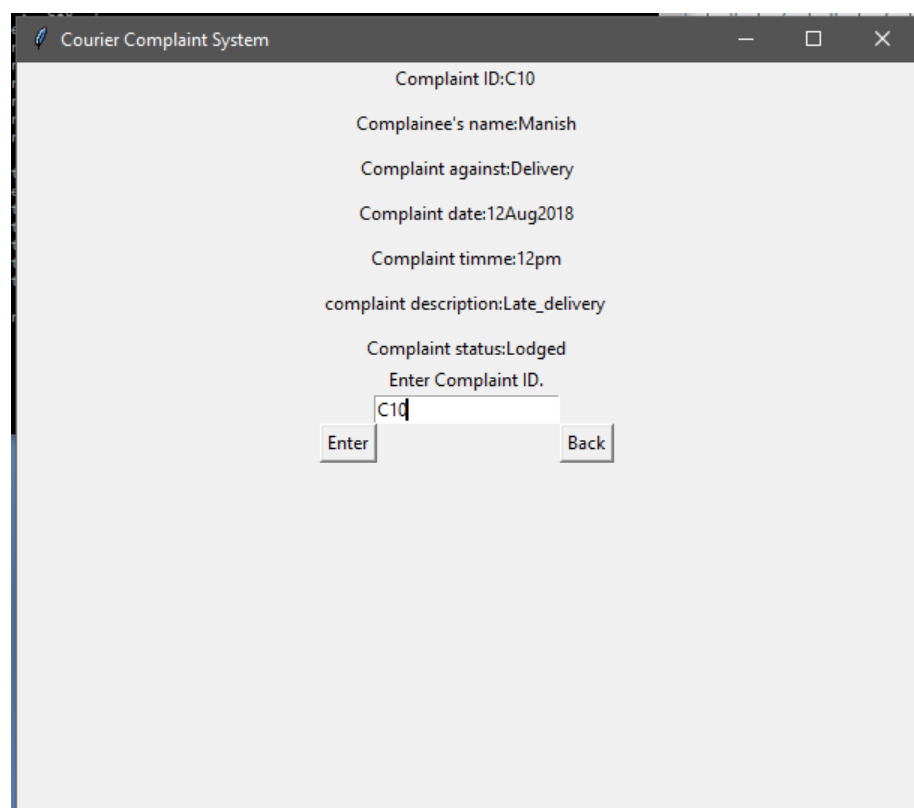
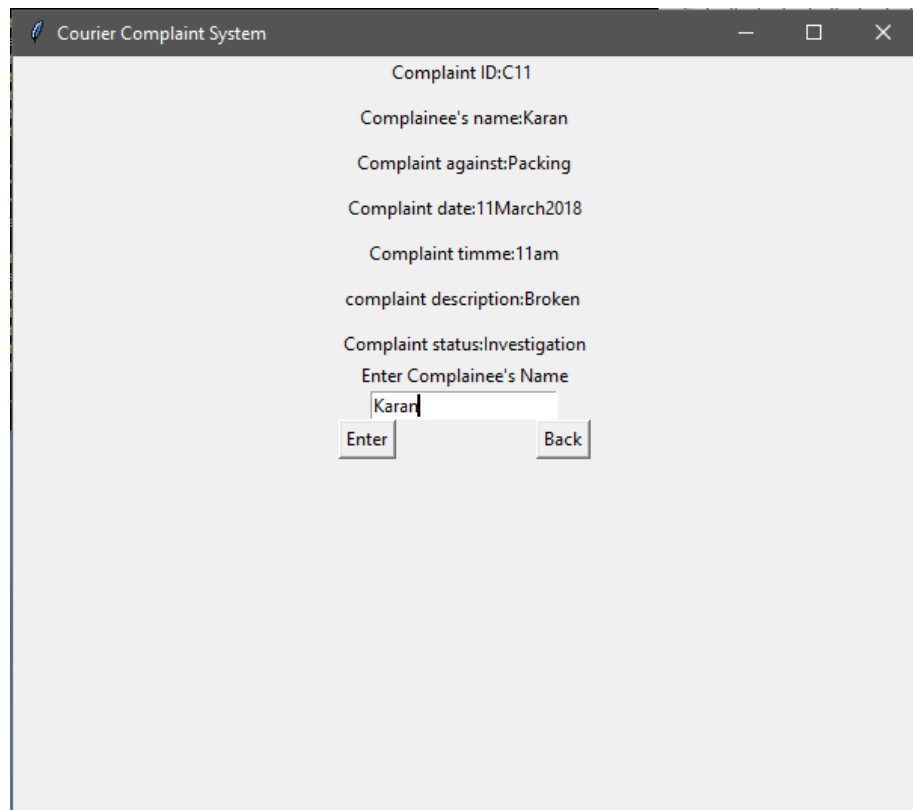


Figure 4 Search by Customer ID



The screenshot shows a web application window titled "Courier Complaint System". It displays the details of a complaint with ID "C11". The details listed are: Complainee's name: Karan, Complaint against: Packing, Complaint date: 11March2018, Complaint timme: 11am, complaint description: Broken, and Complaint status: Investigation. Below these details, there is a section titled "Enter Complainee's Name" with a text input field containing "Karan". At the bottom of this section are two buttons: "Enter" and "Back".

Courier Complaint System

Complaint ID:C11

Complainee's name:Karan

Complaint against:Packing

Complaint date:11March2018

Complaint timme:11am

complaint description:Broken

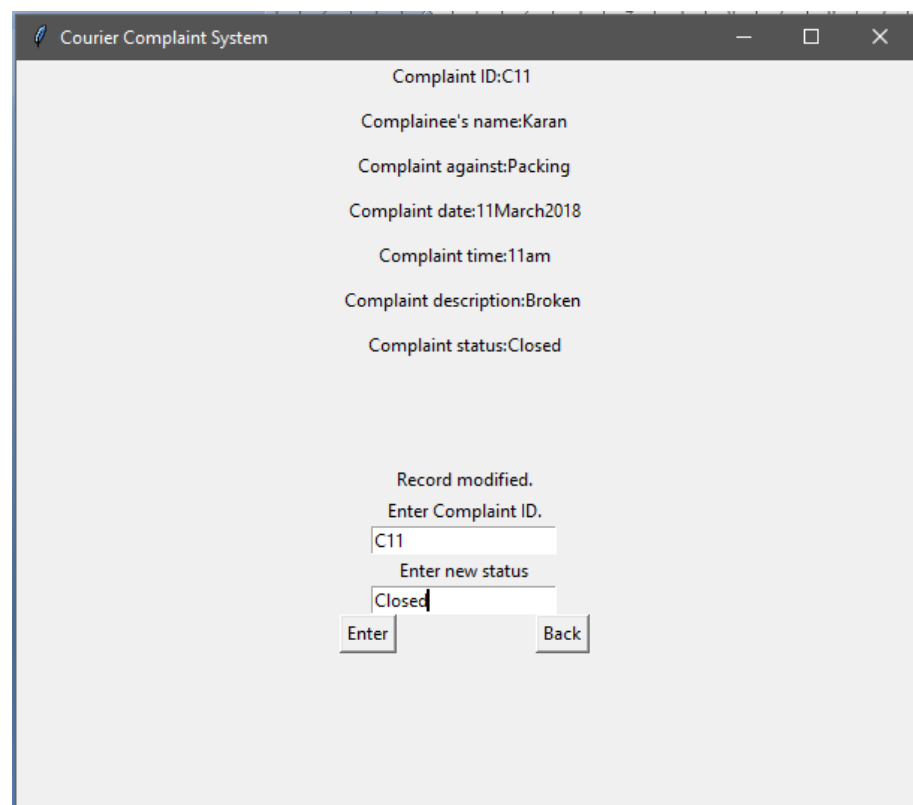
Complaint status:Investigation

Enter Complainee's Name

Karan

Enter Back

Figure 5 Search by name



The screenshot shows the same web application window, but now it is in the "Modify Complaint" mode. The details for complaint ID "C11" are shown, but the status is now "Closed". Below the details, there is a message "Record modified." followed by a section titled "Enter Complaint ID." with a text input field containing "C11". Below that is a section titled "Enter new status" with a text input field containing "Closed". At the bottom of this section are two buttons: "Enter" and "Back".

Courier Complaint System

Complaint ID:C11

Complainee's name:Karan

Complaint against:Packing

Complaint date:11March2018

Complaint time:11am

Complaint description:Broken

Complaint status:Closed

Record modified.

Enter Complaint ID.

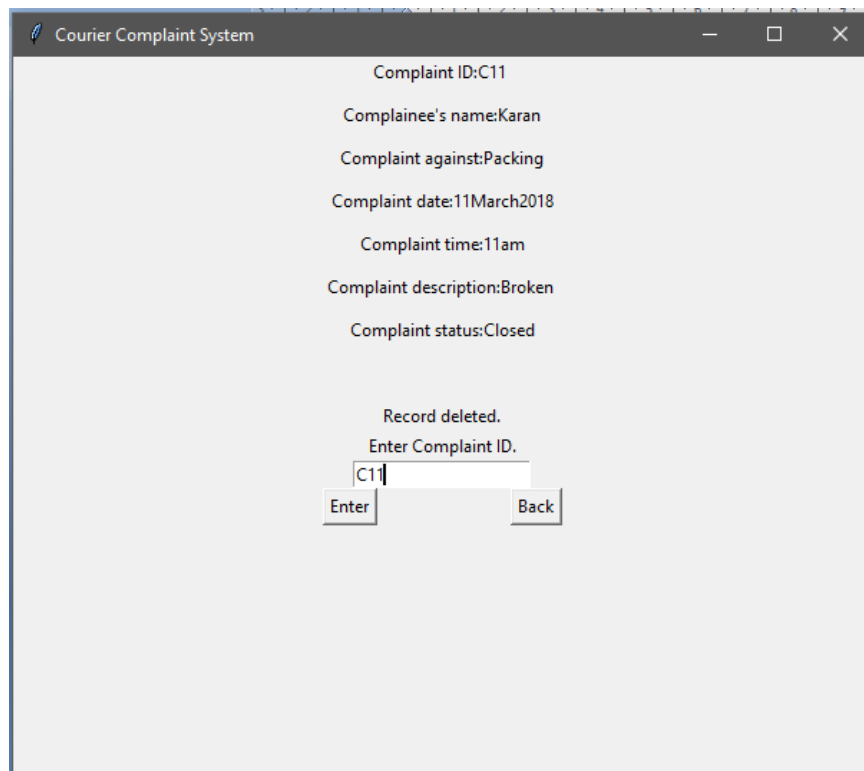
C11

Enter new status

Closed

Enter Back

Figure 6 Modify Complaint



Courier Complaint System

Complaint ID:C11

Complainee's name:Karan

Complaint against:Packing

Complaint date:11March2018

Complaint time:11am

Complaint description:Broken

Complaint status:Closed

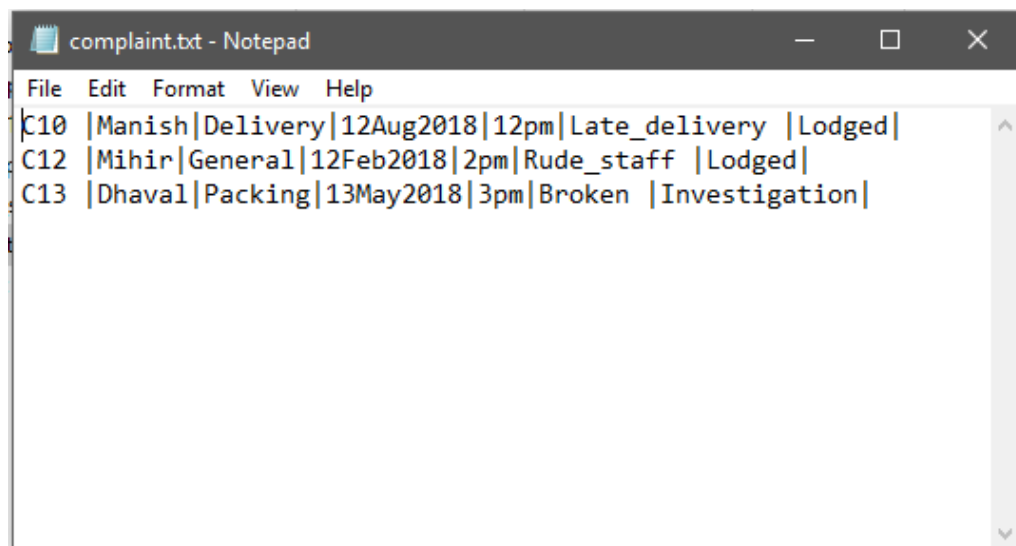
Record deleted.

Enter Complaint ID.

C11

Enter Back

Figure 7 Delete Complaint



complaint.txt - Notepad

File Edit Format View Help

C10	Manish	Delivery	12Aug2018	12pm	Late_delivery	Lodged
C12	Mihir	General	12Feb2018	2pm	Rude_staff	Lodged
C13	Dhaval	Packing	13May2018	3pm	Broken	Investigation

Figure 8 Record File

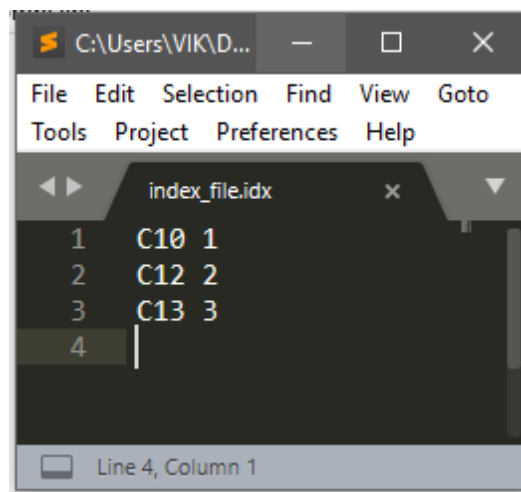


Figure 9 Index file

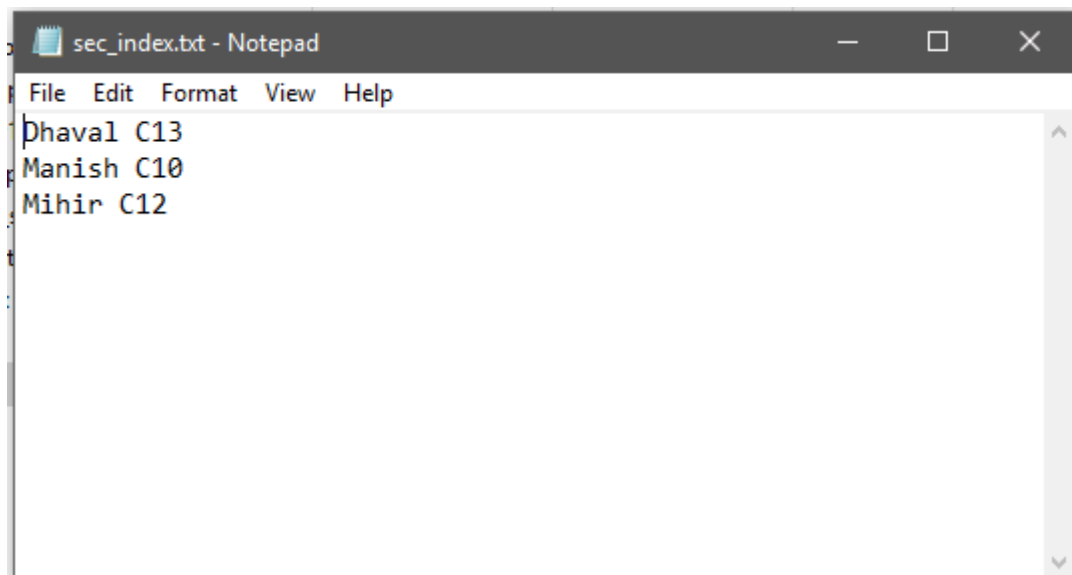


Figure 10 Secondary Index File

Chapter 6

Conclusion

6.1. Conclusion

The project implements an online complaint booking system that creates a complaint record with the Complaint ID as the primary index. This desktop application makes the tedious task of Lodging Complaints and managing them simple and convenient. Earlier people use to go to the offline office to lodge the complaints and also the admins had a lot of difficulty in keeping checks across all the complaints. This system hence enables them to remove the age old register system and use of file system.

6.2. Limitations of the Project

1. The application does not offer multiple views.
2. Since the project is hosted on the desktop, there is no provision for a multiple user system.

6.3. Future Enhancements

1. Enable a multiuser system.
2. Improve on the User Interface and User Experience to make sure it can effectively be used in real-time editors for all applications.
3. Reduction of search time by using more efficient techniques of search than simple primary indexing and secondary indexing for large files.

References

Book references

1. Think Python: How to Think Like a Computer Scientist by [Allen B. Downey](#) (Author)
2. Python for Everybody: Exploring Data in Python 3 by [Charles R. Severance](#) (Author)
3. File Structures: An Object Oriented Approach with C++ by [Michael J. Folk](#) (Author)

Website references

1. <https://www.codecademy.com/>
2. <https://dash.generalassemb.ly/>
3. <https://www.w3schools.com/>
4. <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html>
5. <https://www.w3schools.com/python/default.asp>
6. https://www.tutorialspoint.com/python/python_lists.html