

TAXI MANAGEMENT SYSTEM USING TKINTER

MINI PROJECT REPORT

Submitted By

PAVITHRA B **711720104061**

PRIYADHARSINI M **711720104064**

ROSITHA A **711720104071**

SOWMIYA P **711720104092**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KGiSL INSTITUTE OF TECHNOLOGY, COIMBATORE**ANNA UNIVERSITY: CHENNAI 600 025**

MAY 2023

ANNA UNIVERSITY: CHENNAI-600 025

BONAFIDE CERTIFICATE

Certified that this project report “**TAXI MANAGEMENT USING TKINTER**” is the bonafide work of “**PAVITHRA B, PRIYADHARSINI M, ROSITHA A, SOWMIYA P**” who carried out the mini project under my supervision.

SIGNATURE

Dr. THENMOZHI T

HEAD OF THE DEPARTMENT

PROFESSOR

Computer Science and Engineering

KGiSL Institute of Technology

Coimbatore-641035

SIGNATURE

Ms. ARUNA T N

SUPERVISOR

ASSISTANT PROFESSOR

Computer Science and Engineering

KGiSL Institute of Technology

Coimbatore-641035

Submitted for the Anna University Viva-Voce examination held on_____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Chairman and Managing Director Dr. Ashok Bakthavathsalam** for providing us with an environment to complete our project successfully.

We are grateful to our CEO, Academic Initiatives **Mr. Aravind Kumar Rajendran**, our beloved Director-Academics **Dr. P. Shankar** and our honorable Secretary **Dr. N. Rajkumar**. We are thankful to our beloved Principal **Dr. S. Suresh Kumar** for his support, valuable guidance and blessings.

We would like to thank **Dr. T. Thenmozhi, M.E., M.B.A., Ph. D**, Head of the Department, Department of Computer Science and Engineering for her unwavering support during the entire course of this project work. We express our sincere thanks to **Ms. Aruna T N, M.E.**, our Project Coordinator, Assistant Professor, Department of Computer Science and Engineering who modelled us both technically and morally for achieving greater success in completing this project work.

We express our sincere and heartfelt thanks to our faculty **Ms. Aruna T N M.E.**, Assistant Professor, Department of Computer Science and Engineering for her constant encouragement and support throughout our course, especially for the useful suggestions given during the course of the project period and being instrumental in the completion of our project with her complete guidance.

We also thank all the **Faculty members** of our department and finally, we take this opportunity to extend our deep appreciation to our **Family and Friends**, for all they meant to us during the crucial times of the completion of our project.

ABSTRACT

A software program called the Taxi Management System utilising Tkinter is intended to automate and streamline taxi operations. The taxi sector is crucial to today's transport networks since it gives people all around the world easy and effective ways to travel. Taxi company administrators may manage their fleet, drivers, bookings, and other critical duties with the system's effective and user-friendly interface designed using the Tkinter framework. Through a different user interface, passengers can use the system and book a taxi by entering their pick-up and drop-off locations, preferred vehicle type, and other pertinent information. The system aims to streamline the operations, improve customer service, and enhance overall efficiency for both taxi drivers and passengers. Here are some specific objectives of a taxi management system using Tkinter.

INDEX

CHAPTER NO	TITLE	Page no
	ABSTRACT	i
	INDEX	ii
	LIST OF FIGURE	v
1	INTRODUCTION	
	1.1 PROBLEM DEFINITION	1
	1.2 OBJECTIVE OF THE PROJECT	2
	1.3 SIGNIFICANCE OF THE PROJECT	2
	1.4 OUTLINE OF THE PROJECT	2
2	LITERATURE REVIEW	3
3	SYSTEM ANALYSIS	
	3.1 EXISITING SYSYTEM	
	3.1.1 PITFALLS OF THE EXISTING SYSTEM	4
	3.2 PROPOSED SYSTEM	
	3.2.1 PROPOSED METHOD	5
	3.2.2 ADVANTAGES OVER EXISTING METHOD	5
4	SOFTWARE DESCRIPTON	
	4.1 FRONT END	
	4.1.1 VISUAL STUDIO CODE	6
	4.1.2 PYTHON	6
	4.1.3 TKINTER	7
	4.2 BACK END	
	4.2.1 DATABASE	8
	4.2.2 MYSQLLITE	8

5	PROJECT OVERVIEW	
	5.1 OVERVIEW OF THE PROJECT	9
	5.2 ER DIAGRAM	9
	5.3 FILE DESIGN	11
	5.4 INPUT DESIGN	
	5.4.1 INPUT TYPE	11
	5.4.2 INPUT MEDIA	12
	5.5 OUTPUT DESIGN	
	5.5.1 OUTPUT TYPE	13
	5.5.2 OUTPUT MEDIA	13
	5.6 DATABASE DESIGN	14
6	SYSTEM DEVELOPMENT	
	6.1 MODULE DESCRIPTION	15
7	TESTING AND IMPLEMENTATION	
	7.1 TESTING	
	7.1.1 TESTING OBJECTIVE	16
	7.1.1.1 TESTING PRINCIPLES	16
	7.1.1.2 TYPES OF TESTING	17
	7.2 SYSTEM TESTING	17
	7.2.1 ACCEPTANCE TESTING	17
	7.2.1.1 VALIDATION TESTING	18
	7.2.1.2 VERIFICATION TESTING	19
	7.3 SYSTEM IMPLEMENTATION	19
	7.3.1 USER TRAINING	19

8	APPENDIX	
	8.1 SOURCE CODE	21
	8.2 OUTPUT	43
9	REFERNCES	47

LIST OF FIGURES

FIG NO	FIGURE NAME	PAGE NO
4.2.1.1	SQL Server	8
5.2	ER Diagram	10
7.2.1	Acceptance Testing	18
8-a	Create account	43
8-b	Account created	43
8-c	Login page	44
8-d	Customer details	44
8-e	Booking details	45
8-f	Taxi options	45
8-g	Exit applications	46
8-h	output	46

CHAPTER 1

INTRODUCTION

To achieve smooth operations and outstanding user experiences, taxi service management needs effective tools and technologies in the fast-paced world of today. A common Python package for building graphical user interfaces (GUIs) is Tkinter, which is used to integrate the Taxi Management System. Taxi operators may effectively manage their fleet, track the whereabouts of their vehicles, and improve client relations with the help of Tkinter's intuitive and user-friendly interface. Customers will benefit from the Tkinter-based system's user-friendly interface, which will make it simple for them to request rides, monitor the whereabouts of the designated taxi, determine prices, and send secure payments. The GUI will provide simple controls and engaging features to improve user experience overall. Additionally, customers will be able to give reviews and ratings, assuring accountability and transparency in service quality.

1.1 PROBLEM DEFINITION

The inefficiency and lack of streamline in taxi management sector. Traditional taxi management techniques frequently rely on manual procedures, have limited tracking capabilities. It involves the development to address the challenges faced by taxi service providers in efficiently managing their operations. The system aims to streamline processes, improve customer experience and optimize resource allocation.

1.2 OBJECTIVE OF THE PROJECT

The system aims to streamline the operations, improve customer service, and enhance overall efficiency for both taxi drivers and passengers. Some of the specific objectives of a taxi management system using Tkinter are as follows:

- User Registration and Authentication
- Booking and Dispatching
- Admin Dashboard
- Payment Processing
- Customer Interaction and Satisfaction

1.3 SIGNIFICANCE OF THE PROJECT

- It becomes convenient for users to book cab without actually visiting the place.
- It has user-friendly interface.
- It has own custom property to integrate with the rapid development of users.

1.4 OUTLINE OF THE PROJECT

Taxi Management System has been designed to help organize, manage and to reduce time consumption. This system makes the user to feel user-friendly and it work 24*7.

- We can store data in database and the information is generated.
- This system used to check the fare and cab available.

CHAPTER-2

LITERATURE REVIEW

1. Yueming Peng, Shanshan Liu, Xinglong Dai done research on “Design and Implementation of Taxi Management System” . This research analysed about the Providing a high-performance processing power, highly reliable data security, clear visual data and other advantages to assist the taxi management. It is published in the year of 2018. Keywords are Taxi management, Information systems, Databases, Controls, Form
2. Weiyu Chen, Haochi Wu, Zhen Wang, Jialven Huang, ring Jiang, Jing Zhang, Lingxuan Zhu done research on “Research on the optimization of allocating resources of urban taxi based on decision analysis model”. It describes that it Includes optimum scheme by using Analytic Hierarchy Process Model and evaluate the extent of the impact of each scheme on solving the problem by using Decision Analysis Model published on 2016.
3. Shih-Fen Cheng and Xin Qu done a research on “A Service Choice Model for Optimizing Taxi Service Delivery”. This system has a Proposeto leverage on infrastructure and build a service choice model that helps individual drivers in deciding whether to serve a specific taxi stand or not. It is published in the year of 2009
4. Design and application of taxi intelligent integrated service and management information system Directing at the problems of taxi service and management, the taxi intelligent integrated service and management information system (TIISMIS) is designed Published in 2016 13th International Conference on Service Systems and Service Management (ICSSSM) IEEE.

5. Design of Taxi Management System Based on Nios II Publisher: IEEE it is published in 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC).
6. Efficient taxi dispatching system in distributed environment. Big Data Analytics is the process of examining a large volume of data which is collected from various sources. Published in: 2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)

CHAPTER 3

SYSTEM ANALYSIS

The creation of a taxi management system must include a process called system analysis. It entails learning and comprehending. Make the case for utilising infrastructure to create a service selection model that enables individual drivers to choose whether or not to provide service at a particular taxi stand.

6.1 EXISTING SYSTEM

The existing features of the proposed system are:

- **User Interface:** This is the front-end component that allows users to interact with the system
- **Booking Management:** The booking management component handles the processing of customer requests
- **Administrative Dashboard:** Provides a back-end interface for system administrators to manage and monitor the system
- **Payment Integration:** This component facilitates the payment process by integrating with payment gateways or third-party payment services Database.
- **Database :** The database stores and manages information about the system's users, drivers, and type of vehicle

6.1.1 PITFALLS OF THE EXISTING SYSTEM

- Complex UI : User interface shall be difficult to understand implement.
- High Pricing : pricing can be relatively high in some applications.
- Learning curve : set of features and capabilities challenging for new users to learn and navigate software effectively
- Customer support : difficulty in getting timely responses or effective solutions.

6.2 PROPOSED SYSTEM

The taxi management system is implemented using python tkinter framework.

3.2.1 PROPOSED METHOD

WHY TKINTER?

- Ridiculously fast
- Reassuringly secure
- Exceedingly scalable
- Incredibly versatile
- Easy to Integrate with Python Libraries/Functions

3.2.2 ADVANTAGES OVER EXISTING METHOD

- Provides a simple and attractive UI design.
- Comparatively low Pricing.
- Provides high customer support.
- Easy learning curve
- Customer safety features

CHAPTER 4

SOFTWARE DESCRIPTION

4.1 FRONT END

4.1.1 VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a popular source code editor developed by Microsoft. It provides a rich set of features and extensions, making it a powerful tool for developers. Here are some key features of Visual Studio Code:

- **Cross-Platform Support** : Visual Studio Code is available for Windows, macOS, and Linux, ensuring compatibility across different operating systems.
- **Lightweight and fast** : designed to be lightweight and efficient, providing fast startup times and responsive performance
- **IntelliSense** : It is an intelligent code completion feature that offers suggestions and autocompletion for variables, functions, and modules
- **Built-in Terminal**: VS Code comes with an integrated terminal that allows developers to execute commands, run scripts, and perform various tasks without leaving the editor.
- **Integrated Debugging** : It provides a unified debugging experience

4.1.2 PYTHON

Python is a versatile programming language that offers a wide range of features, making it popular among developers. Some notable features of python:

- **Easy to learn and Readability**: Python has a simple and clean syntax that is easy to understand and learn, making it an ideal choice for beginners
- **Cross-platform Compatibility**: Python is a platform-independent language
- **Large Third-party Ecosystem**: frameworks, such as NumPy, Pandas, Django, Flask, TensorFlow, and more, which extend the language's

capabilities for specific domains and applications.

- **Object-Oriented Programming Support:** allowing developers to create classes, objects, and inheritances, facilitating code organization and reuse.
- Interpreted language: Allows for faster prototyping, interactive development, and easier debugging

4.1.3 TKINTER

Tkinter, a standard GUI toolkit for Python, provides several features that make it a popular choice for developing graphical user interfaces. Some key features of Tkinter:

- **Cross-Platform Compatability:** Tkinter is available for Windows, macOS, and Linux, ensuring compatibility across different operating systems
- **Geometry Management:** The available geometry managers in Tkinter are pack, grid, and place, offering flexibility in arranging and resizing widgets.
- **Integration with Python's Standard Library:** The available geometry managers in Tkinter are pack, grid, and place, offering flexibility in arranging and resizing widgets.

4.2 BACK-END

4.2.1 DATABASE

The available geometry managers in Tkinter are pack, grid, and place, offering flexibility in arranging and resizing widgets.

4.2.1.1 MySQLite

The available geometry managers in Tkinter are pack, grid, and place, offering flexibility in arranging and resizing widgets.

- Embedded Database: The available geometry managers in Tkinter are pack, grid, and place, offering flexibility in arranging and resizing widgets.
- Lightweight and Fast: It performs well even with large datasets and offers fast read and write operations due to its efficient architecture and indexing mechanisms.
- Open Source and free: It makes it freely available for use in commercial and non-commercial applications.

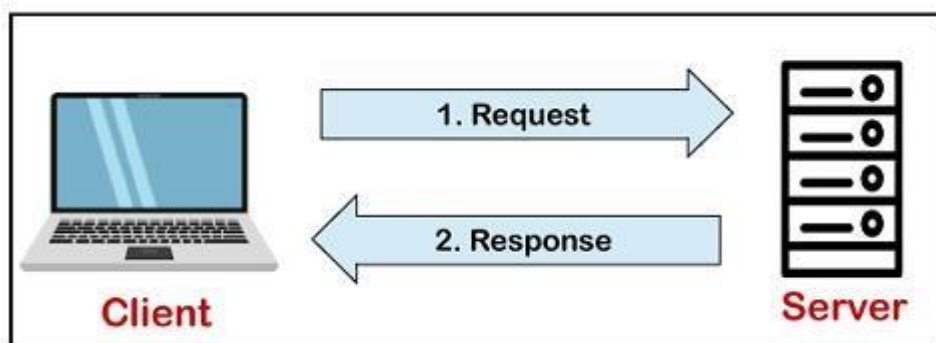


Fig 4.2.1.1 SQL SERVER

CHAPTER 5

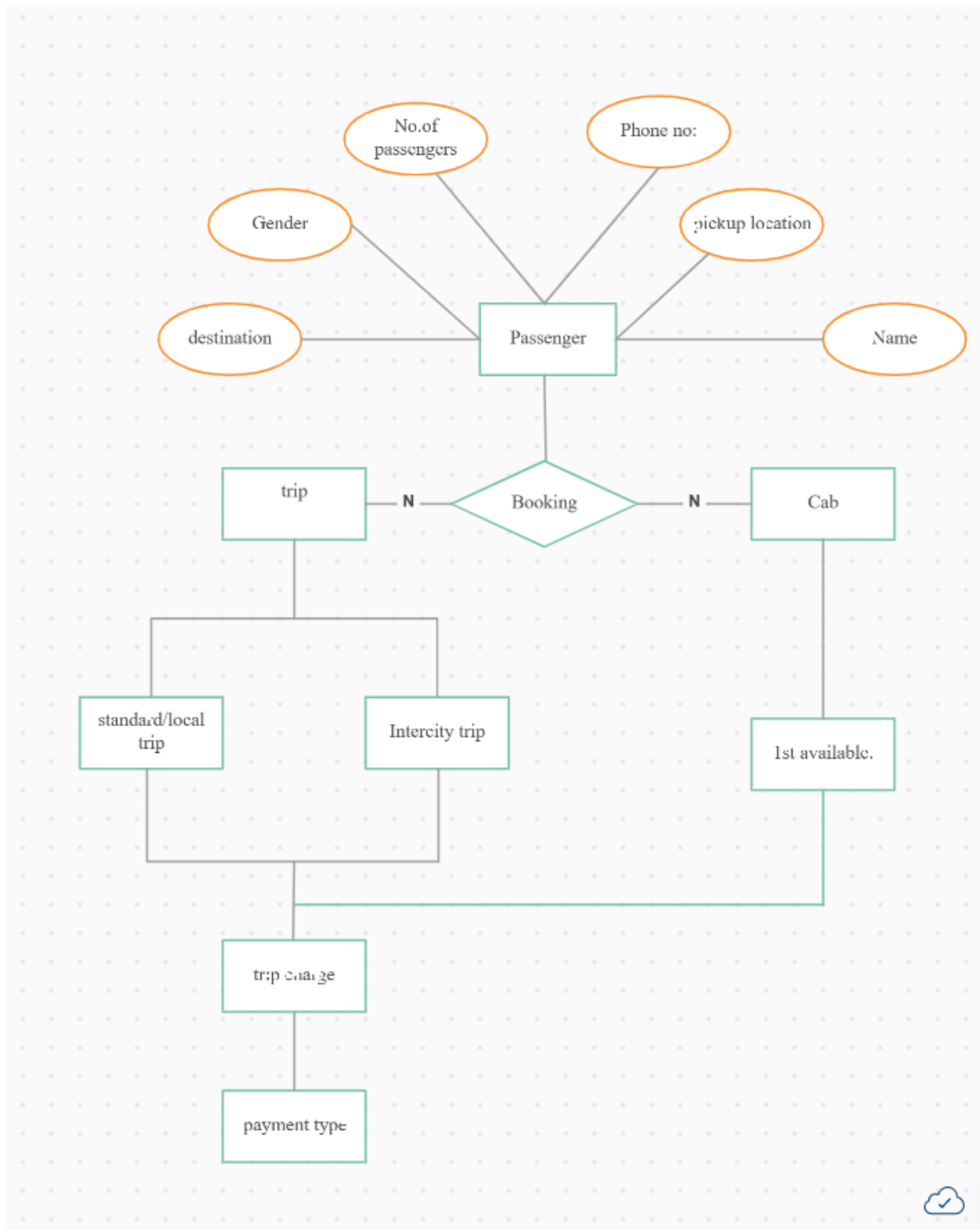
PROJECT OVERVIEW

5.1 OVERVIEW OF THE PROJECT

“The Taxi Management System using Tkinter” is to offer taxi operators a complete solution for efficiently managing their fleet, streamlining processes, and improving customer experience. To provide a user-friendly graphical interface with a variety of features, such as real-time fleet monitoring, dispatching, and customer service, the project will employ Python's Tkinter toolkit. The creation of an easy-to-use graphical user interface that seamlessly combines the many capabilities would be required for the Taxi Management System implementation using Tkinter. Python will be used to design the system, and the Tkinter library will be used to create the GUI elements. The system will also include APIs and libraries for payment integration, GPS tracking, and mapping. Taxi operators will be able to improve customer satisfaction by streamlining their operations, optimising dispatching, and adopting the Taxi Management System using Tkinter. Both operators and customers will have a seamless experience thanks to the interactive features and user-friendly UI. The objective of this project is to transform the taxi management process and enhance the taxi industry.

5.2 ER DIAGRAM

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database.



5.2 ER DIAGRAM

5.3 FILE DESIGN

File design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system.

The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

5.4 INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

5.4.1 INPUT TYPE

The input types for a taxi management system can vary depending on the specific functionalities and requirements of the system. Here are some common input types that a taxi management system may use:

1. Text Input:

- Users can input text-based information such as their name, contact details, pickup and drop-off locations, and additional instructions or comments.

2. Dropdown Menus:

- Dropdown menus can be used to provide predefined options for users to select from, such as choosing a vehicle type, payment method, or trip purpose (e.g., business, personal).

3. Date and Time Picker:

- A date and time picker allows users to select the desired pickup or reservation time for their trip.

4. Checkbox or Radio Buttons:

- Checkbox or radio button inputs can be used for binary choices or selecting options from a limited set, such as specifying whether a trip is one-way or round-trip.

5.4.2 INPUT MEDIA

In a taxi management system, various input media can be used to facilitate communication and data input. Here are some common input media used in a taxi management system:

1. **Web Interface:** A web-based interface allows users to access the system using a web browser. Users can input information through text fields, dropdown menus, checkboxes, and other web form elements.

2. **Mobile Applications:** Mobile apps provide a convenient way for users to access the taxi management system on their smartphones or tablets. The input methods can include text input, dropdown menus, buttons, and gestures specific to the mobile platform.

5.5 OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation.

5.5.1 OUTPUT TYPE

A taxi management system produces various types of output to provide information, notifications, and reports to users and stakeholders. Here are some common output types generated by a taxi management system:

1. **Booking Confirmations:** When a user successfully books a taxi, the system generates a booking confirmation that includes details such as the trip ID, pickup location, drop-off location, assigned driver, and estimated arrival time.
2. **Driver and Vehicle Information:** The system can provide information about assigned drivers and vehicles, including their names, contact details, vehicle numbers, and other relevant information.
3. **User Feedback:** The system can collect and process user feedback regarding their taxi experience. This feedback can be stored and used for analysis and improvement of the system and services. Feedback can be gathered through surveys, ratings, or comments.

5.4.2 OUTPUT DESIGN

A taxi management system utilizes various output media to deliver information and communicate with users and stakeholders. Here are some common output media used in a taxi management system:

1. **User Interfaces: Web Interface:** The system can display information, notifications, and reports on web pages accessed through a browser.

2. **Mobile Applications:** Mobile apps can present outputs through the user interface on smartphones or tablets, providing a customized and responsive user experience.
3. **Reports and Analytics:** The system can generate reports and analytics in formats like PDF or Excel, which can be accessed and viewed by administrators or stakeholders.

5.6 DATABASE DESIGN

The database design involves creation of tables. Tables are represented in physical database as stored files. They have their own independent existence. A table consists of rows and columns. Each column corresponds to a piece of information called field. A set of fields constitutes a record. The record contains all the information, specific to a particular item.

- Data Integration
- Data Integrity
- Data Independence

CHAPTER 6

SYSTEM DEVELOPMENT

6.1 MODULE DESCRIPTION

1:User Registration

This module takes care of registering a new user for taxi service. A user can request for cab by registering and filling required forms. Once the driver approves, he/she can take a drive.

2:Creating User Interface

This module is where the users can book the cab for their travel. A cost estimation can be found by the user.

3:Bill Generation

This module is about Generating receipt and exit. A user can take their fair amount using receipt button and they can exit.

CHAPTER 7

TESTING AND IMPLEMENTATION

7.1 TESTING

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected.

Software testing is the process of testing the functionality and correctness of software by running it. Process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as yet undiscovered error. A successful test is one that uncovers an as yet undiscovered error. Software testing is usually performed for two reasons.

- Defect detection
- Reliability estimation

7.1.1 TESTING OBJECTIVE

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered.
- A successful test is one that uncovers an as yet undiscovered error.

7.1.1.1 TESTING PRINCIPLES

- All tests should be traceable to customer requirements.
- Tests should be planned large before testing begins.
- Testing should begin “In the Small” and progress towards “In the Large”.

7.1.1.2 TYPES OF TESTING

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

- White box testing
- Black box testing
- Unit testing
- Integration testing

7.2 SYSTEM TESTING

System Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interface with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

7.2.1 ACCEPTANCE TESTING

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is

to evaluate the system's compliance with the business requirements and verify if it has met the required criteria for delivery to end users.

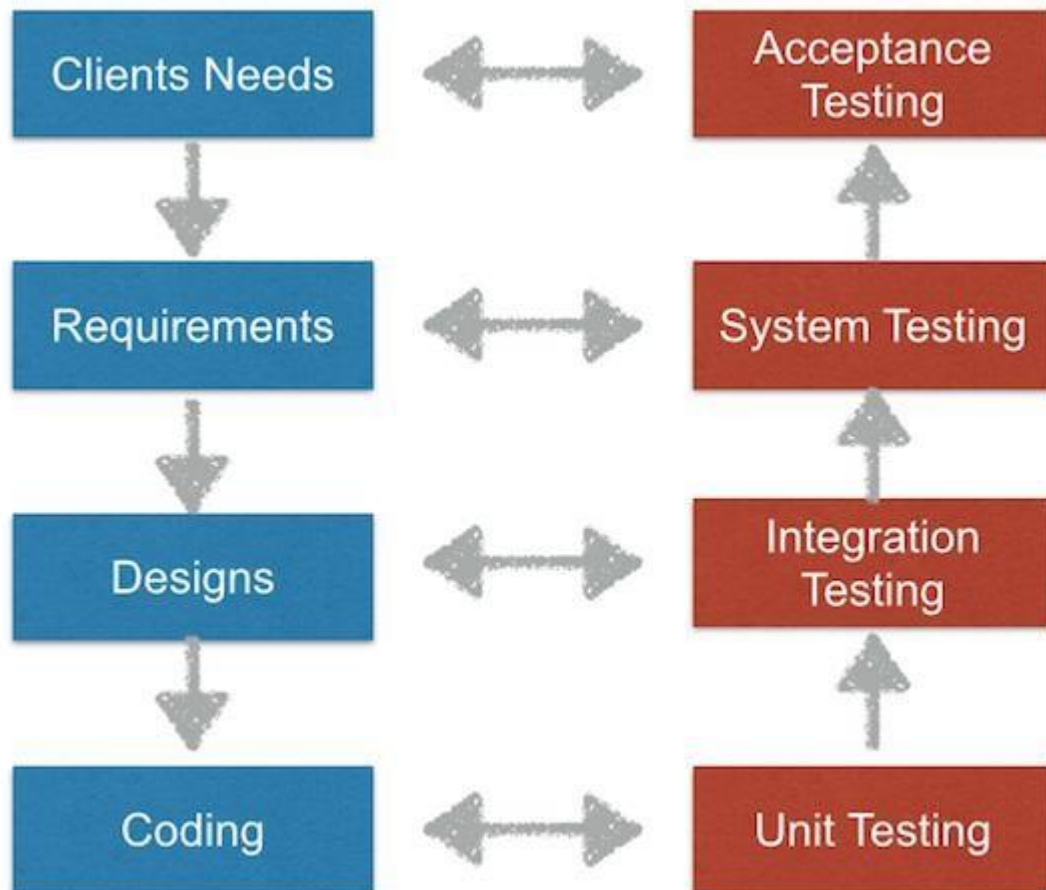


FIG 7.2.1 ACCEPTANCE TESTING

7.2.1.1 VALIDATION TESTING

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

7.2.1.2 VERIFICATION TESTING

Verification is the process of evaluating work-products of a development phase to determine whether they meet the specified requirements. Verification ensures that the product is built according to the requirements and design specifications.

7.3 SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system and giving a user confidence in that the new system will work efficiently and effectively in the implementation stage. The stage consists of

1. Testing a developed program with sample data.
2. Detection and correction of error.
3. Creating whether the system meets a user requirement.
4. making necessary changes as desired by users.
5. Training user personal.

7.3.1 USER TRAINING

It is designed to prepare the users for testing & converting the system. There is several ways to train the users they are:

- 1) User manual.
- 2) Help screens.
- 3) Training demonstrations.

1) User manual:

The summary of important functions about the system & software can be provided as a document to the user. User training is designed to prepare the user for testing and convening a system.

The summary of important functions about the system and the software can be provided as a document to the user,

- Open http page
- Type the file name with URL Default.aspx in the address bar

2) Help screens:

This features now available in every software package, especially when it is used with a menu. The user selects the “Help” option from the menu. The systems are success the necessary description or information for user references.

3) Training demonstration:

Another user training element is a training demonstration. Live demonstration with personal contact is extremely effective for training users.

CHAPTER 8

APPENDIX

8.1 SOURCE CODE

Taxi.py

```
from tkinter import *

from tkinter import ttk

import random

import time

import datetime

from tkinter import messagebox as ms

import sqlite3

Item4 = 0

# make database and users (if not exists already) table at programme start up

with sqlite3.connect('Users.db') as db:

    c = db.cursor()

    c.execute('CREATE TABLE IF NOT EXISTS user (username TEXT NOT NULL

,password TEXT NOT NULL)')

    db.commit()

    db.close()

#main Class
```

```

class user:

    def __init__(self, master):

        # Window

        self.master = master

        # Some Usefull variables

        self.username = StringVar()

        self.password = StringVar()

        self.n_username = StringVar()

        self.n_password = StringVar()

        #Create Widgets

        self.widgets()

    #Login Function

    def login(self):

        #Establish Connection

        with sqlite3.connect('Users.db') as db:

            c = db.cursor()

            #Find user If there is any take proper action

            find_user = ('SELECT * FROM user WHERE username = ? and password = ?')

            c.execute(find_user, [(self.username.get()), (self.password.get())])

            result = c.fetchall()

            if result:

                self.logf.pack_forget()

```

```

self.head['text'] = "Welcome, " + self.username.get()

self.head.configure(fg="Purple")

self.head.pack(fill=X)

application = travel(root)

else:

    ms.showerror('Oops!','Username Not Found.')

def new_user(self):

    #Establish Connection

    with sqlite3.connect('Users.db') as db:

        c = db.cursor()

        #Find Existing username if any take proper action

        find_user = ('SELECT * FROM user WHERE username = ?')

        c.execute(find_user,[self.username.get()])

        if c.fetchall():

            ms.showerror('Error!','Username Already Taken!')

        else:

            ms.showinfo('Success!','Account Created!')

            self.log()

        #Create New Account

        insert = 'INSERT INTO user(username,password) VALUES(?,?)'

        c.execute(insert,[self.n_username.get(),self.n_password.get()])

```



```
db.commit()
```

```
#Frame Packing Methods
```

```
def log(self):
```

```
    self.username.set("")
```

```
    self.password.set("")
```

```
    self.crf.pack_forget()
```

```
    self.head['text'] = 'LOGIN'
```

```
    self.logf.pack()
```

```
def cr(self):
```

```
    self.n_username.set("")
```

```
    self.n_password.set("")
```

```
    self.logf.pack_forget()
```

```
    self.head['text'] = 'Create Account'
```

```
    self.crf.pack()
```

```
#Draw Widgets
```

```
def widgets(self):
```

```
    self.head = Label(self.master,text = 'LOGIN',font = ('Arial',35),pady = 10)
```

```
    self.head.pack()
```

```
    self.logf = Frame(self.master,padx =10,pady = 10)
```

```
    Label(self.logf,text = 'Username: ',font = ("",20),pady=5,padx=5).grid(sticky = W)
```

```

Entry(self.logf,textvariable=self.username,bd=5,font=("",15)).grid(row=0,column=1)

Label(self.logf,text = 'Password: ',font = ("",20),pady=5,padx=5).grid(sticky = W)


Entry(self.logf,textvariable=self.password,bd=5,font=("",15),show='*').grid(row=1,column=1)

Button(self.logf,text='Login',bd=3,font=("",15),padx=5,pady=5,command=self.login).grid()

Button(self.logf,text='CreateAccount',bd=3,font=("",15),padx=5,pady=5,command=self.cr).grid(row=2,column=1)

    self.logf.pack()

    self.crf = Frame(self.master,padx =10,pady = 10)

    Label(self.crf,text = 'Username: ',font = ("",20),pady=5,padx=5).grid(sticky = W)

    Entry(self.crf,textvariable=self.n_username,bd=5,font=("",15)).grid(row=0,column=1)

    Label(self.crf,text = 'Password: ',font = ("",20),pady=5,padx=5).grid(sticky = W)

    Entry(self.crf,textvariable=self.n_password,bd=5,font=("",15),show='*').grid(row=1,column=1)

    Button(self.crf,text='CreateAccount',bd=3,font=("",15),padx=5,pady=5,command=self.new_user).grid()

    Button(self.crf,text='GotoLogin',bd=3,font=("",15),padx=5,pady=5,command=self.log).grid(row=2,column=1)

```

```

class travel:

```

```

    def __init__(self,root):

```

```

self.root = root

self.root.title("Taxi Booking System In LPU")

self.root.geometry(geometry)

self.root.configure(background='Red')


DateofOrder=StringVar()

DateofOrder.set(time.strftime(" %d / %m / %Y "))

Receipt_Ref=StringVar()

PaidTax=StringVar()

SubTotal=StringVar()

TotalCost=StringVar()


var1=IntVar()

var2=IntVar()

var3=IntVar()

var4=IntVar()

journeyType=IntVar()

carType=IntVar()


var11=StringVar()

var12=StringVar()

var13=StringVar()

```

```
reset_counter=0

Firstname=StringVar()

Surname=StringVar()

Address=StringVar()

Postcode=StringVar()

Mobile=StringVar()

Telephone=StringVar()

Email=StringVar()


TaxiTax=StringVar()

Km=StringVar()

Travel_Ins=StringVar()

Luggage=StringVar()

Receipt=StringVar()

Standard=StringVar()

PrimeSedan=StringVar()

PremiumSedan=StringVar()

TaxiTax.set("0")

Km.set("0")

Travel_Ins.set("0")

Luggage.set("0")

Standard.set("0")
```

PrimeSedan.set("0")

PremiumSedan.set("0")

#=====DefineFunction=====

def iExit():

iExit= ms.askyesno("Prompt!", "Do you want to exit?")

if iExit > 0:

root.destroy()

return

def Reset():

TaxiTax.set("0")

Km.set("0")

Travel_Ins.set("0")

Luggage.set("0")

Standard.set("0")

PrimeSedan.set("0")

PremiumSedan.set("0")

Firstname.set("")

Surname.set("")

Address.set("")

Postcode.set("")

Mobile.set("")

Telephone.set("")

Email.set("")

PaidTax.set("")

SubTotal.set("")

TotalCost.set("")

self.txtReceipt1.delete("1.0",END)

self.txtReceipt2.delete("1.0",END)

var1.set(0)

var2.set(0)

var3.set(0)

var4.set(0)

journeyType.set(0)

carType.set(0)

varl1.set("0")

varl2.set("0")

varl3.set("0")

```
self.cboPickup.current(0)
```

```
self.cboDrop.current(0)
```

```
self.cboPooling.current(0)
```

```
self.txtTaxiTax.configure(state=DISABLED)
```

```
self.txtKm.configure(state=DISABLED)
```

```
self.txtTravel_Ins.configure(state=DISABLED)
```

```
self.txtLuggage.configure(state=DISABLED)
```

```
self.txtStandard.configure(state=DISABLED)
```

```
self.txtPrimeSedan.configure(state=DISABLED)
```

```
self.txtPremiumSedan.configure(state=DISABLED)
```

```
self.reset_counter=1
```

```
def Receiptt():
```

```
    if reset_counter is 0 and Firstname.get()!=" " and Surname.get()!=" " and  
    Address.get()!=" " and Postcode.get()!=" " and Mobile.get()!=" " and Telephone.get()!=" "  
    and Email.get()!=" ":
```

```
        self.txtReceipt1.delete("1.0",END)
```

```
        self.txtReceipt2.delete("1.0",END)
```

```
        x=random.randint(10853,500831)
```

```
        randomRef = str(x)
```

Receipt_Ref.set(randomRef)

self.txtReceipt1.insert(END,"Receipt Ref:\n")

self.txtReceipt2.insert(END, Receipt_Ref.get() + "\n")

self.txtReceipt1.insert(END,'Date:\n')

self.txtReceipt2.insert(END, DateofOrder.get() + "\n")

self.txtReceipt1.insert(END,'Taxi No:\n')

self.txtReceipt2.insert(END, 'TR ' + Receipt_Ref.get() + " BW\n")

self.txtReceipt1.insert(END,'Firstname:\n')

self.txtReceipt2.insert(END, Firstname.get() + "\n")

self.txtReceipt1.insert(END,'Surname:\n')

self.txtReceipt2.insert(END, Surname.get() + "\n")

self.txtReceipt1.insert(END,'Address:\n')

self.txtReceipt2.insert(END, Address.get() + "\n")

self.txtReceipt1.insert(END,'Postal Code:\n')

self.txtReceipt2.insert(END, Postcode.get() + "\n")

self.txtReceipt1.insert(END,'Telephone:\n')

self.txtReceipt2.insert(END, Telephone.get() + "\n")

self.txtReceipt1.insert(END,'Mobile:\n')

self.txtReceipt2.insert(END, Mobile.get() + "\n")

self.txtReceipt1.insert(END,'Email:\n')

self.txtReceipt2.insert(END, Email.get() + "\n")


```

self.txtReceipt1.insert(END,'From:\n')

self.txtReceipt2.insert(END, var11.get() + "\n")

self.txtReceipt1.insert(END,'To:\n')

self.txtReceipt2.insert(END, var12.get() + "\n")

self.txtReceipt1.insert(END,'Pooling:\n')

self.txtReceipt2.insert(END, var13.get() + "\n")

self.txtReceipt1.insert(END,'Standard:\n')

self.txtReceipt2.insert(END, Standard.get() + "\n")

self.txtReceipt1.insert(END,'Prime Sedan:\n')

self.txtReceipt2.insert(END, PrimeSedan.get() + "\n")

self.txtReceipt1.insert(END,'Premium Sedan:\n')

self.txtReceipt2.insert(END, PremiumSedan.get() + "\n")

self.txtReceipt1.insert(END,'Paid:\n')

self.txtReceipt2.insert(END, PaidTax.get() + "\n")

self.txtReceipt1.insert(END,'SubTotal:\n')

self.txtReceipt2.insert(END, str(SubTotal.get()) + "\n")

self.txtReceipt1.insert(END,'Total Cost:\n')

self.txtReceipt2.insert(END, str(TotalCost.get()))

```

else:

```

self.txtReceipt1.delete("1.0",END)

self.txtReceipt2.delete("1.0",END)

```

```
self.txtReceipt1.insert(END, "\nNo Input")
```

```
def Taxi_Tax():
```

```
    global Item1
```

```
    if var1.get() == 1:
```

```
        self.txtTaxiTax.configure(state = NORMAL)
```

```
        Item1=float(50)
```

```
        TaxiTax.set("Rs " + str(Item1))
```

```
    elif var1.get() == 0:
```

```
        self.txtTaxiTax.configure(state=DISABLED)
```

```
        TaxiTax.set("0")
```

```
        Item1=0
```

```
def Kilo():
```

```
    if var2.get() == 0:
```

```
        self.txtKm.configure(state=DISABLED)
```

```
        Km.set("0")
```

```
    elif var2.get() == 1 and var11.get() != "" and var12.get() != "":
```

```
        self.txtKm.configure(state=NORMAL)
```

```
        if var11.get() == "CampusCafe":
```

```
switch={"BoysHostel":10,"GirlsHostel":8,"AdmissionBlock":6,"CampusCafe": 0}
```

```
    Km.set(switch[varl2.get()])
```

```
elif varl1.get() == "BoysHostel":
```

```
switch={"BoysHostel":0,"GirlsHostel":2,"AdmissionBlock":5,"CampusCafe": 10}
```

```
    Km.set(switch[varl2.get()])
```

```
elif varl1.get() == "GirlsHostel":
```

```
switch={"BoysHostel":2,"GirlsHostel":0,"AdmissionBlock":3,"CampusCafe": 8}
```

```
    Km.set(switch[varl2.get()])
```

```
elif varl1.get() == "AdmissionBlock":
```

```
switch={"BoysHostel":5,"GirlsHostel":3,"AdmissionBlock":0,"CampusCafe": 6}
```

```
    Km.set(switch[varl2.get()])
```

```
def Travelling():
```

```
    global Item3
```

```
    if var3.get() == 1:
```

```
        self.txtTravel_Ins.configure(state = NORMAL)
```

```
        Item3=float(10)
```

```
        Travel_Ins.set("Rs " + str(Item3))
```

```
elif var3.get() == 0:

    self.txtTravel_Ins.configure(state = DISABLED)

    Travel_Ins.set("0")

    Item3=0
```

```
def Lug():

    global Item4

    if (var4.get()==1):

        self.txtLuggage.configure(state = NORMAL)

        Item4=float(30)

        Luggage.set("Rs "+ str(Item4))

    elif var4.get()== 0:

        self.txtLuggage.configure(state = DISABLED)

        Luggage.set("0")

        Item4=0
```

```
def selectCar():

    global Item5

    if carType.get() == 1:

        self.txtPrimeSedan.configure(state = DISABLED)
```

```

PrimeSedan.set("0")

self.txtPremiumSedan.configure(state = DISABLED)

PremiumSedan.set("0")

self.txtStandard.configure(state = NORMAL)

Item5 = float(8)

Standard.set("Rs "+ str(Item5))

elif carType.get() == 2:

    self.txtStandard.configure(state =DISABLED)

    Standard.set("0")

    self.txtPremiumSedan.configure(state = DISABLED)

    PremiumSedan.set("0")

    self.txtPrimeSedan.configure(state = NORMAL)

    Item5 = float(10)

    PrimeSedan.set("Rs "+ str(Item5))

else:

    self.txtStandard.configure(state =DISABLED)

    Standard.set("0")

    self.txtPrimeSedan.configure(state = DISABLED)

    PrimeSedan.set("0")

    self.txtPremiumSedan.configure(state = NORMAL)

    Item5 = float(15)

    PremiumSedan.set("Rs "+ str(Item5))

```

```

def Total_Paid():

    if ((var1.get() == 1 and var2.get() == 1 and var3.get() == 1 or var4.get() == 1)
and carType.get() != 0 and journeyType.get() != 0 and (var11.get() != "" and var12.get()
!="")):

        if journeyType.get()==1:

            Item2=Km.get()

            Cost_of_fare = (Item1+(float(Item2)*Item5)+Item3+Item4)

            Tax = "Rs " + str('%0.2f%((Cost_of_fare) *0.09))

            ST = "Rs " + str('%0.2f%((Cost_of_fare)))

            TT = "Rs " + str('%0.2f%(Cost_of_fare+((Cost_of_fare)*0.9)))

        elif journeyType.get()==2:

            Item2=Km.get()

            Cost_of_fare = (Item1+(float(Item2)*Item5)*1.5+Item3+Item4)

            Tax = "Rs " + str('%0.2f%((Cost_of_fare) *0.09))

            ST = "Rs " + str('%0.2f%((Cost_of_fare)))

            TT = "Rs " + str('%0.2f%(Cost_of_fare+((Cost_of_fare)*0.9)))

        else:

            Item2=Km.get()

            Cost_of_fare = (Item1+(float(Item2)*Item5)*2+Item3+Item4)

```

```
Tax = "Rs " + str('%0.2f'%((Cost_of_fare) *0.09))
```

```
ST = "Rs " + str('%0.2f'%((Cost_of_fare)))
```

```
TT = "Rs " + str('%0.2f'%(Cost_of_fare+((Cost_of_fare)*0.9)))
```

```
PaidTax.set(Tax)
```

```
SubTotal.set(ST)
```

```
TotalCost.set(TT)
```

```
else:
```

```
w = ms.showwarning("Error !","Invalid Input\nPlease try again !!!")
```

```
#=====mainframe=====
```

```
MainFrame=Frame(self.root)
```

```
MainFrame.pack(fill=BOTH,expand=True)
```

```
Tops = Frame(MainFrame, bd=20, width=1350,relief=RIDGE)
```

```
Tops.pack(side=TOP,fill=BOTH,expand=True)
```

```
self.lblTitle=Label(Tops,font=('times new roman',70,'bold'),text=" Taxi Booking  
System ")
```

```

self.lblTitle.grid()

#=====customerframedetail=====

CustomerDetailsFrame=LabelFrame(MainFrame, width=1350,height=500,bd=20,
pady=5, relief=RIDGE)

CustomerDetailsFrame.pack(side=BOTTOM,fill=BOTH,expand=True)


FrameDetails=Frame(CustomerDetailsFrame, width=880,height=400,bd=10,
relief=RIDGE)

FrameDetails.pack(side=LEFT,fill=BOTH,expand=True)


CustomerName=LabelFrame(FrameDetails, width=150,height=250,bd=10,
font=('arial',12,'bold'),text="Customer Name", relief=RIDGE)

CustomerName.grid(row=0,column=0)


TravelFrame = LabelFrame(FrameDetails,bd=10, width=300,height=250,
font=('arial',12,'bold'),text="Booking Detail", relief=RIDGE)

TravelFrame.grid(row=0,column=1)


Book_Frame=LabelFrame(FrameDetails,width=300,height=150,relief=FLAT)

Book_Frame.grid(row=1,column=0)


CostFrame = LabelFrame(FrameDetails,width=150,height=150,bd=5,relief=FLAT)

```



```
CostFrame.grid(row=1,column=1)
```

```
#=====receipt=====
```

```
Receipt_BottonFrame=LabelFrame(CustomerDetailsFrame,bd=10,  
width=450,height=400,relief=RIDGE)
```

```
Receipt_BottonFrame.pack(side=RIGHT,fill=BOTH,expand=True)
```

```
ReceiptFrame=LabelFrame(Receipt_BottonFrame,width=350,height=300,  
font=('arial',12,'bold'),text="Receipt",relief=RIDGE)
```

```
ReceiptFrame.grid(row=0,column=0)
```

```
ButtonFrame=LabelFrame(Receipt_BottonFrame,width=350,height=100,  
relief=RIDGE)
```

```
ButtonFrame.grid(row=1,column=0)
```

```
#=====CustomerName=====
```

```
self.lblFirstname=Label(CustomerName,font=('arial',14,'bold'),text="Firstname",bd=7)
```

```
self.lblFirstname.grid(row=0,column=0,sticky=W)
```

```
self.txtFirstname=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Firstname,  
bd=7,insertwidth=2,justify=RIGHT)
```

```
self.txtFirstname.grid(row=0,column=1)
```

```
self.lblSurname=Label(CustomerName,font=('arial',14,'bold'),text="Surname",bd=7)

    self.lblSurname.grid(row=1,column=0,sticky=W)


self.txtSurname=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Surname,bd
=7,insertwidth=2,justify=RIGHT)

    self.txtSurname.grid(row=1,column=1,sticky=W)

self.lblAddress=Label(CustomerName,font=('arial',14,'bold'),text="Address",bd=7)

    self.lblAddress.grid(row=2,column=0,sticky=W)


self.txtAddress=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Address,bd=
7,insertwidth=2,justify=RIGHT)

    self.txtAddress.grid(row=2,column=1)


self.lblPostcode=Label(CustomerName,font=('arial',14,'bold'),text="Postcode",bd=7)

    self.lblPostcode.grid(row=3,column=0,sticky=W)


self.txtPostcode=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Postcode,bd
=7,insertwidth=2,justify=RIGHT)

    self.txtPostcode.grid(row=3,column=1)
```

```
self.lblTelephone=Label(CustomerName,font=('arial',14,'bold'),text="Telephone",bd=7)
```

```
self.lblTelephone.grid(row=4,column=0,sticky=W)
```

```
self.txtTelephone=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Telephone,  
bd=7,insertwidth=2,justify=RIGHT)
```

```
self.txtTelephone.grid(row=4,column=1)
```

```
self.lblMobile=Label(CustomerName,font=('arial',14,'bold'),text="Mobile",bd=7)
```

```
self.lblMobile.grid(row=5,column=0,sticky=W)
```

```
self.txtMobile=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Mobile,bd=7,  
insertwidth=2,justify=RIGHT)
```

```
self.txtMobile.grid(row=5,column=1)
```

```
self.lblEmail=Label(CustomerName,font=('arial',14,'bold'),text="Email",bd=7)
```

```
self.lblEmail.grid(row=6,column=0,sticky=W)
```

```
self.txtEmail=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Email,bd=7,  
insertwidth=2,justify=RIGHT)
```

```
self.txtEmail.grid(row=6,column=1)
```

```
#=====TaxiInformation=====
```

```

self.lblPickup=Label(TravelFrame,font=('arial',14,'bold'),text="Pickup",bd=7)

self.lblPickup.grid(row=0,column=0,sticky=W)


self.cboPickup =ttk.Combobox(TravelFrame, textvariable = var11 , state='readonly',
font=('arial',20,'bold'), width=14)

self.cboPickup['value']=('Saravanampatti','Thudiyalur','Kovilpalayam','Gandhipuram')

self.cboPickup.current(0)

self.cboPickup.grid(row=0,column=1)

self.lblDrop=Label(TravelFrame,font=('arial',14,'bold'),text="Drop",bd=7)

self.lblDrop.grid(row=1,column=0,sticky=W)


self.cboDrop =ttk.Combobox(TravelFrame, textvariable = var12 , state='readonly',
font=('arial',20,'bold'), width=14)

self.cboDrop['value']=('Saravanampatti','Thudiyalur','Kovilpalayam','Gandhipuram')

self.cboDrop.current(0)

self.cboDrop.grid(row=1,column=1)


self.lblPooling=Label(TravelFrame,font=('arial',14,'bold'),text="Pooling",bd=7)

self.lblPooling.grid(row=2,column=0,sticky=W)

```

```

        self.cboPooling =tkk.Combobox(TravelFrame, textvariable = var13 ,
state='readonly', font=('arial',20,'bold'), width=14)

        self.cboPooling['value']=('1','2','3','4')

        self.cboPooling.current(1)

        self.cboPooling.grid(row=2,column=1)

        self.chkTaxiTax=Checkbutton(TravelFrame,text="TaxiTax(BaseCharge)
*",variable=var1,onvalue=1,offvalue=0,font=('arial',16,'bold'),command=Taxi_Tax).gri
d(row=3,column=0, sticky=W)

self.txtTaxiTax=Label(TravelFrame,font=('arial',14,'bold'),textvariable=TaxiTax,bd=6,w
idth=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN)

        self.txtTaxiTax.grid(row=3,column=1)

        self.chkKm=Checkbutton(TravelFrame,text="Distance(KMs) *",variable = var2,
onvalue=1, offvalue=0,font=('arial',16,'bold'),command=Kilo).grid(row=4, column=0,
sticky=W)

self.txtKm=Label(TravelFrame,font=('arial',14,'bold'),textvariable=Km,bd=6,width=18,
bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN,highlightthickness=0)

        self.txtKm.grid(row=4,column=1)

        self.chkTravel_Ins=Checkbutton(TravelFrame,text="TravellingInsurance
*",variable=var3,onvalue=1,offvalue=0,font=('arial',16,'bold'),command=Travelling).gr
id(row=5,column=0, sticky=W)

```

```

self.txtTravel_Ins=Label(TravelFrame,font=('arial',14,'bold'),textvariable=Travel_Ins,bd=6,width=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN)

    self.txtTravel_Ins.grid(row=5,column=1)

    self.chkLuggage=Checkbutton(TravelFrame,text="Extra Luggage",variable = var4,onvalue=1, offvalue=0,font=('arial',16,'bold'),command=Lug).grid(row=6, column=0,sticky=W)

self.txtLuggage=Label(TravelFrame,font=('arial',14,'bold'),textvariable=Luggage,bd=6,width=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN)

    self.txtLuggage.grid(row=6,column=1)

    self.lblPaidTax=Label(CostFrame,font=('arial',14,'bold'),text="Paid Tax\t\t",bd=7)

    self.lblPaidTax.grid(row=0,column=2,sticky=W)

    self.txtPaidTax=Label(CostFrame,font=('arial',14,'bold'),textvariable=PaidTax,bd=7,width=26, justify=RIGHT,bg="white",relief=SUNKEN)

    self.txtPaidTax.grid(row=0,column=3)

self.lblSubTotal=Label(CostFrame,font=('arial',14,'bold'),text="Sub Total",bd=7)

    self.lblSubTotal.grid(row=1,column=2,sticky=W)
self.txtSubTotal=Label(CostFrame,font=('arial',14,'bold'),textvariable=SubTotal,bd=7,width=26, justify=RIGHT,bg="white",relief=SUNKEN)

    self.txtSubTotal.grid(row=1,column=3)

    self.lblTotalCost=Label(CostFrame,font=('arial',14,'bold'),text="Total Cost",bd=7)

    self.lblTotalCost.grid(row=2,column=2,sticky=W)

```

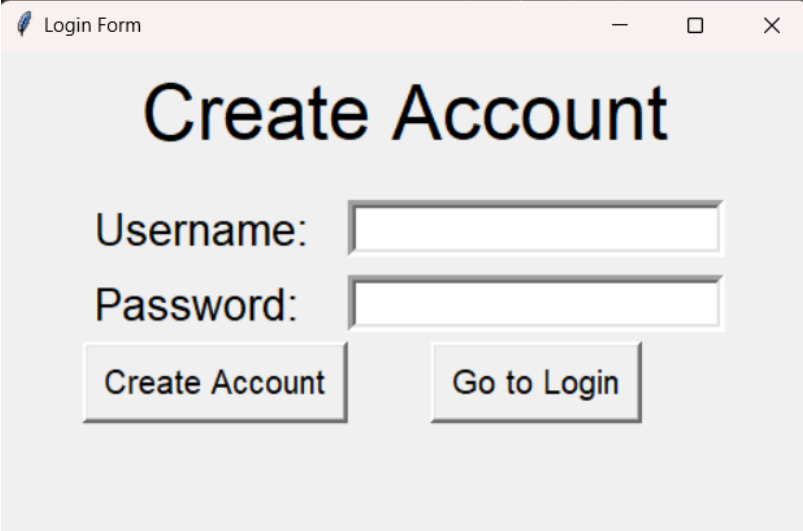
```
self.txtTotalCost=Label(CostFrame,font=('arial',14,'bold'),textvariable=TotalCost,bd=7,  
width=26,justify=RIGHT,bg="white",relief=SUNKEN)
```

```
self.txtTotalCost.grid(row=2,column=3)
```

```
self.chkStandard=Radiobutton(Book_Frame,text="Standard",value=1,variable=  
carType,font=('arial',14,'bold'),command=selectCar).grid(row=0, column=0, sticky=W)
```

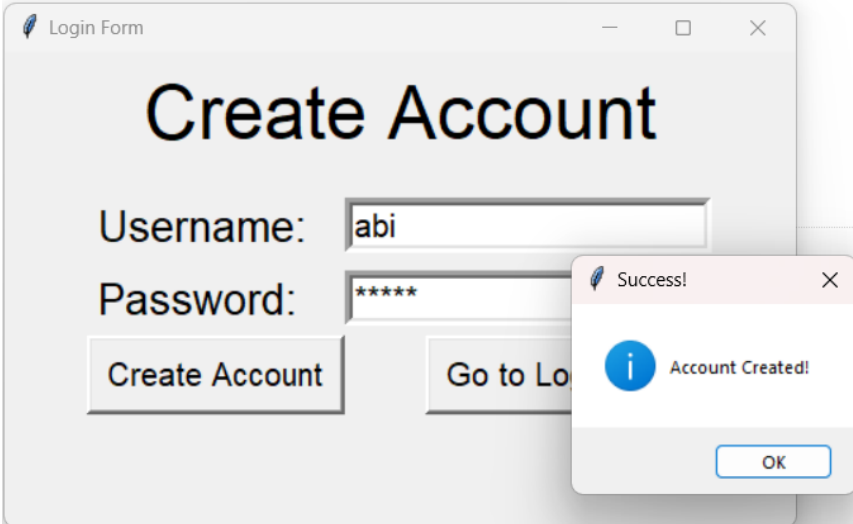
```
self.txtStandard=Label(Book_Frame,font=('arial',14,'bold'),width=7,textvariable=Stand  
ard,bd=5,
```

8.2 OUTPUT



The screenshot shows a window titled "Login Form" with a "Create Account" form. The form has a title "Create Account" in large black font. Below the title are two input fields: "Username:" and "Password:". The "Username:" field is empty, and the "Password:" field is also empty. Below the input fields are two buttons: "Create Account" and "Go to Login".

Fig 8.a Create account



The screenshot shows the same "Create Account" form as in Fig 8.a, but with a success message dialog box overlaid. The dialog box is titled "Success!" and contains the text "Account Created!". There is an "OK" button at the bottom of the dialog box. The "Username:" field in the background form now contains the text "abi", and the "Password:" field contains six asterisks "*****".

Fig 8 b.Account created

Login Form

LOGIN

Username:

Password:

Fig 8-c Login Page

Customer Name

Firstname

Surname

Address

Postcode

Telephone

Mobile

Email

Booking Detail

Pickup

Drop

Pooling

☐ Taxi Tax(Base Charge) *

☐ Distance(KMs) *

☐ Travelling Insurance *

☐ Extra Luggage

☐ Standard ☐ Single

☐ PrimeSedan ☐ Return

☐ PremiumSedan ☐ SpecialNeeds

Paid Tax

Sub Total

Total Cost

Customer Name

Firstname

Surname

Address

Postcode

Telephone

Mobile

Email

Fig 8-d Customer Details

Booking Detail

Pickup

Drop

Pooling

1

☐ Taxi Tax(Base Charge) *

0

☐ Distance(KMs) *

0

☐ Travelling Insurance *

0

☐ Extra Luggage

0

Booking Detail

Pickup

Saravanampatti

Drop

Gandhipuram

Pooling

2

☒ Taxi Tax(Base Charge) *

Rs 50.0

☐ Distance(KMs) *

0

☒ Travelling Insurance *

Rs 10.0

☐ Extra Luggage

0

Fig 8-e Booking Detail

☐ Standard

0

☒ PrimeSedan

Rs 10.0

☐ PremiumSedan

0

☒ Standard

Rs 8.0

☐ PrimeSedan

0

☐ PremiumSedan

0

☐ Standard

0

☐ PrimeSedan

0

☒ PremiumSedan

Rs 15.0

Customer Name

Firstname

Abi

Surname

K

Address

Coimbatore

Postcode

641027

Telephone

Mobile

9856748512

Email

Booking Detail

Pickup

Saravanampatti

Drop

Gandhipuram

Pooling

2

☒ Taxi Tax(Base Charge) *

Rs 50.0

☐ Distance(KMs) *

0

☒ Travelling Insurance *

Rs 10.0

☐ Extra Luggage

0

☐ Standard

0

☐ PrimeSedan

0

☒ PremiumSedan

Rs 15.0

☐ Single

☐ Return

☐ SpecialNeeds

Paid Tax

Sub Total

Total Cost

Fig 8-f Taxi options

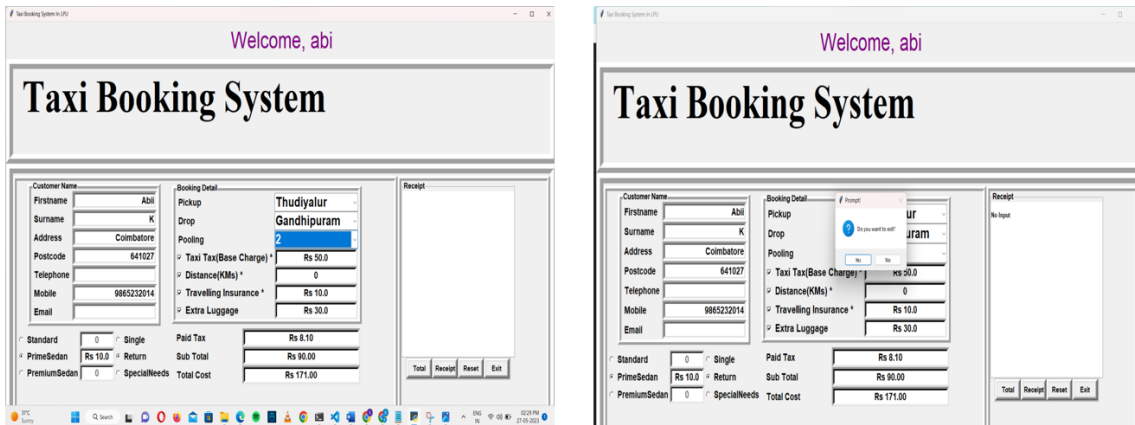


Fig 8-g Exit application

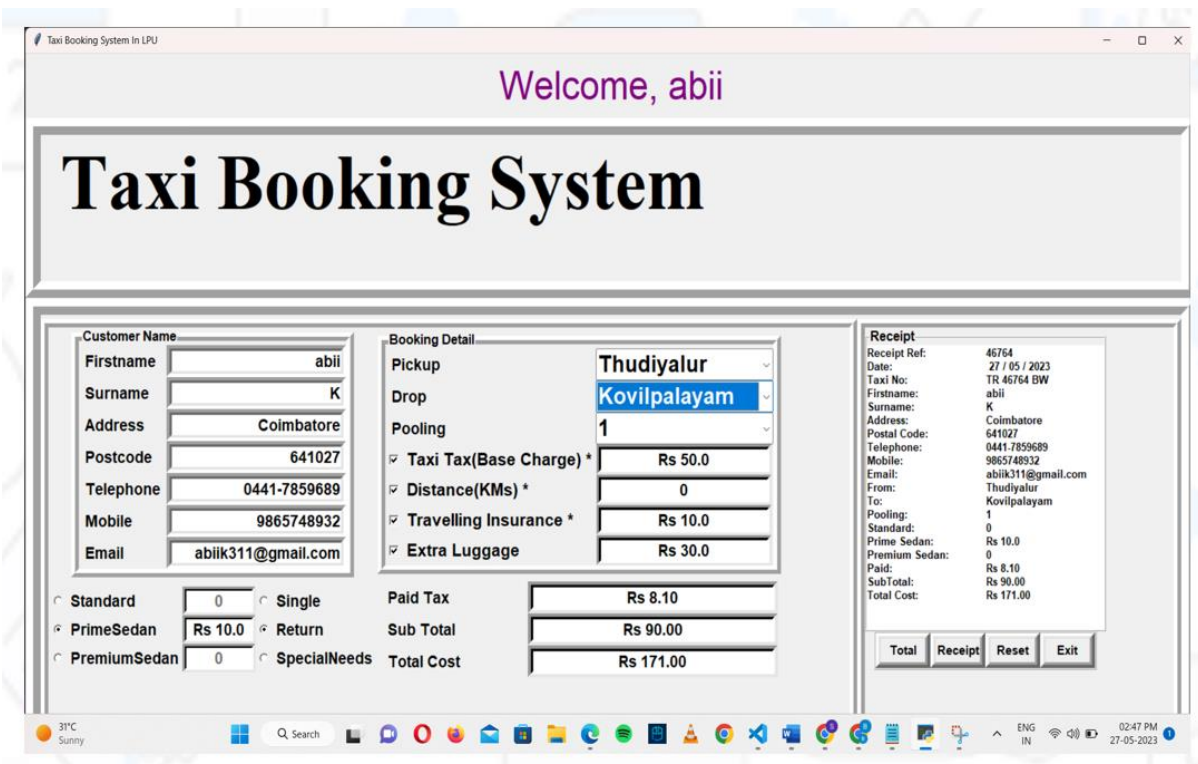


Fig 8-h Output

CHAPTER 9

REFERENCES

1. Yueming Peng, Shanshan Liu, Xinglong Dai done research on “Design and Implementation of Taxi Management System” .
2. Weiyu Chen, Haochi Wu, Zhen Wang, Jialven Huang, Ying Jiang, Jing Zhang, Lingxuan Zhu done research on “Research on the optimization of allocating resources of urban taxi based on decision analysis model”.
3. Shih-Fen Cheng and Xin Qu done a research on “A Service Choice Model for Optimizing Taxi Service Delivery”.
4. Design and application of taxi intelligent integrated service and management information system in 2016 by Weiwei Li; Yanfang Zhou 2016 13th International Conference on Service Systems and Service Management (ICSSSM)
5. Study on impact of taxi status to probe car system performance Bo Liu; Kai Liu; Hiroaki Mizuta 2009 9th International Conference on Intelligent Transport Systems Telecommunications, (ITST) Year: 2009
6. Design of Taxi Management System Based on Nios II by Yongxi Zeng; Yanzhong Yu Zhifei Yang 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)
7. The design of the operation monitoring and statistics analysis system for taxi based on the GPS information researched done by Wang Yong-dong; Xu Dong-wei; 2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC) Year: 2017 | Conference Paper | Publisher: IEEE
8. A service choice model for optimizing taxi service delivery research done by Shih-Fen Cheng; Xin Qu in 2009 12th International IEEE Conference on Intelligent Transportation Systems Year: 2009 | Conference Paper | Publisher: IEEE
9. Efficient taxi dispatching system in distributed environment done by S. Meenakshi; Radha Senthilkumar on 2017 International Conference on Information, Communication
10. Research on the optimization of allocating resources of urban taxi based on decision analysis method done by Weiyu Chen; Haochi Wu; Zhen Wang; Jiawen Huang; Ying Jiang; Jing Zhang 2016 13th International Conference on Service Systems and Service Management (ICSSSM)
11. System analysis of towing aircrafts for taxiing out Wenzhi Zhao on 2012 International Conference on Information Management, Innovation Management and Industrial Engineering Year: 2012 | Conference Paper | Publisher: IEEE