**CAPSTONE PROJECT REPORT**

(Project Term January-May 2023)

Title of the Project

**INVOICE GENERATOR**

Submitted By:

**Name:Aravind Kontham** **Registration Number: 12112090**

Under the Guidance of

**Waseem Ud Din Wani Sir**

**School of Computer Science and Engineering**



**DECLARATION**

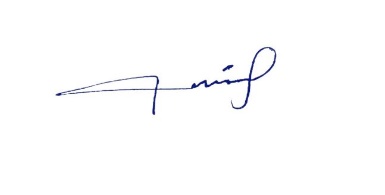
I hereby declare that the project work entitled (“**Invoice Generator**”) is an authentic record of my own

work carried out as requirements of Capstone Project for the award of B. Tech degree in \_**Computer**

**Science**\_\_ from **Lovely Professional University**, Phagwara, under the guidance of (**Waseem Ud Din Wani**

**sir**), during January to May 2023. All the information furnished in this capstone project report is based

onmy own intensive work and is genuine.

I have worked very well to understand the various concepts of python which includes python libraries such as Numpy, Pandas and Matplotlib.

**CERTIFICATE**

This is to certify that the declaration statement made by this this student is correct to the best of my knowledge and belief. He has completed this Capstone Project under my guidance and supervision. The present work is the result of his original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B. Tech degree in CSE from Lovely Professional University, Phagwara.

Signature and Name of the Mentor: Waseem Ud Din Wani Sir

Designation:Teaching Assistant

School of Computer Science and Engineering,

Lovely Professional University,

Phagwara, Punjab.

Date: 25/04/2023

**ACKNOWLEDGEMENT**

"I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful completion of this capstone project. First and foremost, I would like to thank “**Waseem Ud Din Wani Sir**”, whose guidance, encouragement, and feedback have been invaluable throughout this project. Their insights and suggestions have helped me to improve my work and bring it to completion.

I would also like to express my gratitude to the faculty members and staff of Lovely Professional University, who have provided me with the necessary resources and support to undertake this project. Their unwavering support and encouragement have been critical in making this project a success.

I am also grateful to my friends, who have been a constant source of support and motivation throughout my academic journey. Their love, encouragement, and belief in me have been instrumental in helping me achieve my goals.

Lastly, I would like to express my gratitude to all the participants and stakeholders who generously shared their time, knowledge, and expertise with me during the course of this project. Their insights and contributions have been crucial in shaping the direction of my research and have helped me to develop a deeper understanding of the subject matter.

Once again, I would like to thank everyone who has contributed to this project in any way, and I am truly grateful for the opportunity to undertake this work."

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Content** | **Page No** |
| 1 | Declaration | 2 |
| 2 | Certificate | 2 |
| 3 | Acknowledgement | 3 |
| 4 | Abstract | 5 |
| 5 | Introduction | 5 |
| 6 | Objective of the Project | 6 |
| 7 | Description of the Project | 6 |
| 8 | Source Code | 7-13 |
| 9 | Output | 14-19 |
| 10 | Scope of the Project | 20 |
| 11 | Development | 21 |
| 12 | Conclusion | 22 |

**Abstract**

The Invoice Generator is a Python-based software application developed to provide a simple and practical tool for managing invoice data.

The application allows users to input customer information, including name and phone number, as well as information about the items purchased, such as the item name, quantity, and price. Users can also edit or delete any invoice data as needed, and the changes are automatically reflected in the Word and Excel files.

The treeview widget was used to display the invoice data in a structured format, making it easy to view and manage. The application was developed using the Python programming language, with the GUI built using the Tkinter library and data management using the treeview widget and docx and xlsxwriter libraries. This report provides an overview of the project's design, implementation, features, and potential uses.

**INTRODUCTION**

The aim of this capstone project was to develop an invoice generator that allows users to create, edit, and delete invoices, store them in a user-friendly format, and generate Word and Excel files for easy access and management. The project was developed using the Python programming language, with the GUI built using the Tkinter library and data management using the treeview widget and docx and xlsxwriter libraries.

The invoice generator allows users to input customer information, including name and phone number, as well as information about the items purchased, such as the item name, quantity, and price. Users can also edit or delete any invoice data as needed, and the changes are automatically reflected in the Word and Excel files. The treeview widget was used to display the invoice data in a structured format, making it easy to view and manage.

Overall, the invoice generator capstone project was developed with the user in mind, providing a simple and practical tool for managing invoice data. This report will provide a detailed overview of the project's design, implementation, features, and potential uses.

**Objective**

The objective of this capstone project is to design and develop an Invoice Generator that provides a user-friendly tool for managing invoice data. The application will allow users to input customer information and information about the items purchased, edit or delete invoice data as needed, and store the data in an organized format.

The application will also provide the ability to generate Word and Excel files for easy access and management.

The project will be developed using the Python programming language, with the GUI built using the Tkinter library and data management using the treeview widget and docx and xlsxwriter libraries. The end product will be a practical and efficient solution for individuals or small businesses looking to manage their invoice data.

**Description**

The Invoice Generator is a Python-based software application designed to provide a simple and practical tool for managing invoice data. The application provides users with the ability to input customer information, including name and phone number, as well as information about the items purchased, such as the item name, quantity, and price.

The application's user interface was designed using the Tkinter library, with a treeview widget used to display the invoice data in a structured format. Users can also edit or delete any invoice data as needed, and the changes are automatically reflected in the Word and Excel files.

The Invoice Generator provides the ability to generate a Word file containing the invoice details, making it easy to view and manage. In addition, the application generates an Excel file containing the invoice data in a structured format, allowing users to create a dataframe for further analysis.

Overall, the Invoice Generator provides a practical and efficient solution for individuals or small businesses looking to manage their invoice data. The application was developed using Python programming language and libraries such as docx and xlsxwriter for data management and storage.

**Source Code**

import tkinter

import customtkinter as ctk

from tkinter import ttk

from docxtpl import DocxTemplate

import datetime

from tkinter import messagebox

import openpyxl

import pathlib

from tkinter import filedialog

import pandas as pd

import random

window = ctk.CTk()

file = pathlib.Path("invoice.xlsx")

invoice\_list = []

def add\_item():

if desc\_entry.get() == "":

messagebox.showerror("ERROR", "item not selected")

else:

qty = int(qnt\_spinbox.get())

desc = desc\_entry.get()

price = float(price\_spin.get())

line\_total = qty \* price

workbook = openpyxl.load\_workbook("invoice.xlsx")

sheet = workbook.active

sheet.cell(column=1, row=sheet.max\_row + 1, value=desc)

sheet.cell(column=2, row=sheet.max\_row, value=qty)

sheet.cell(column=3, row=sheet.max\_row, value=price)

sheet.cell(column=4, row=sheet.max\_row, value=line\_total)

workbook.save("invoice.xlsx")

invoice\_item = [desc, qty, price, line\_total]

tree.insert('', 0, values=invoice\_item)

clear\_item()

invoice\_list.append(invoice\_item)

def clear\_item():

qnt\_spinbox.delete(0, tkinter.END)

qnt\_spinbox.insert(0, "1")

desc\_entry.delete(0, tkinter.END)

price\_spin.delete(0, tkinter.END)

price\_spin.insert(0, "0.0")

def new\_invoice():

first\_name\_entry.delete(0, tkinter.END)

last\_name\_entry.delete(0, tkinter.END)

phone\_entry.delete(0, tkinter.END)

tree.delete(\*tree.get\_children())

invoice\_list.clear()

def generate\_invoice():

doc = DocxTemplate("C:\\Users\\ARWINDD\\ETP\\Project\\invoice\_template.docx")

name = first\_name\_entry.get() + " " + last\_name\_entry.get()

phone = phone\_entry.get()

subtotal = sum(item[3] for item in invoice\_list)

discount = 0.06

gst = 0.18

invoice\_number = random.randint(1000, 100000)

date = datetime.datetime.now().strftime("%d-%m-%Y")

total = subtotal \* (1 + gst) - subtotal \* discount

doc.render({"name": name,

"phone": phone,

"invoice": invoice\_number,

"invoice\_list": invoice\_list,

"date": date,

"subtotal": subtotal,

'discount': str(discount \* 100) + "%",

'GST': str(gst \* 100) + "%",

"total": round(total, 2)})

doc\_name = "new\_invoice" + name + ".docx"

doc.save(doc\_name)

print(invoice\_list)

new\_invoice()

root = ctk.CTk()

ctk.set\_appearance\_mode("system")

ctk.set\_default\_color\_theme("green")

def upload\_file():

file = filedialog.askopenfilename(filetypes=[("Excel file", '.xlsx'), ("Word file", '.docx')])

df = pd.read\_excel(file)

# df=pd.read\_

t1.insert(tkinter.END, df.head())

l1 = ctk.CTkLabel(root, text="Invoice Generated", font=('Century Gothic', 60))

l1.place(x=160, y=40)

l3 = ctk.CTkLabel(root, text="Please collect from the system", font=('Century Gothic', 20, "italic"))

l3.place(x=220, y=130)

l2 = ctk.CTkLabel(root, text="Create DataFrame", font=('Times New Roman', 20))

l2.place(x=300, y=180)

b1 = ctk.CTkButton(root, text="Read Excel File", width=20, command=lambda: upload\_file(), hover\_color="blue")

b1.place(x=320, y=210)

t1 = ctk.CTkTextbox(root, height=300, width=300, border\_color='white', fg\_color='white', text\_color='black')

t1.place(x=220, y=250)

root.geometry('800x600')

root.title("Invoice Generator")

root.mainloop()

window.title("Invoice\_Generator\_Form")

frame1 = tkinter.Frame(window, bg="#4a7a8c")

frame1.pack(padx=10, pady=40)

lbl = ctk.CTkLabel(frame1, text="Get Your Invoice", font=('Bell MT', 40, 'bold'), text\_color='white')

lbl.pack()

frame = tkinter.Frame(frame1)

frame.pack(padx=12, pady=20)

frame.config(bg="#116562")

first\_name\_lbl = tkinter.Label(frame, text="First Name", fg='white', bg="#116562", font=('Bookman Old Style', 14))

first\_name\_lbl.grid(row=0, column=0)

first\_name\_entry = ctk.CTkEntry(frame, font=("tahoma", 16), corner\_radius=6, border\_color='black', fg\_color="white", text\_color="black")

first\_name\_entry.grid(row=1, column=0)

last\_name\_lbl = tkinter.Label(frame, text="Last Name", fg='white', bg="#116562", font=('Bookman Old Style', 14))

last\_name\_lbl.grid(row=0, column=1)

last\_name\_entry = ctk.CTkEntry(frame, font=("tahoma", 16), corner\_radius=6, border\_color='black', fg\_color="white" , text\_color="black")

last\_name\_entry.grid(row=1, column=1)

phone\_lbl = tkinter.Label(frame, text="Phone", fg='white', bg="#116562", font=('Bookman Old Style', 14))

phone\_lbl.grid(row=0, column=2)

phone\_entry = ctk.CTkEntry(frame, font=("tahoma", 16), corner\_radius=6, border\_color='black', fg\_color="white",

text\_color="black")

phone\_entry.grid(row=1, column=2)

qnt\_label = tkinter.Label(frame, text="Quantity", fg='white', bg="#116562", font=('Bookman Old Style', 14))

qnt\_label.grid(row=3, column=1)

qnt\_spinbox = tkinter.Spinbox(frame, from\_=1, to=100, width=20, borderwidth=4)

qnt\_spinbox.grid(row=4, column=1)

Description\_label = tkinter.Label(frame, text="Product", fg='white', bg="#116562", font=('Bookman Old Style', 14))

Description\_label.grid(row=3, column=0)

desc\_entry = ctk.CTkEntry(frame, font=("tahoma", 16), corner\_radius=6, border\_color='black', fg\_color="white",

text\_color="black")

desc\_entry.grid(row=4, column=0)

Unit\_price\_label = tkinter.Label(frame, text="Unit Price", fg='white', bg="#116562", font=('Bookman Old Style', 14))

Unit\_price\_label.grid(row=3, column=2)

price\_spin = tkinter.Spinbox(frame, from\_=0.0, to=500, increment=0.5, width=20, borderwidth=4)

price\_spin.grid(row=4, column=2)

add\_item\_btn = ctk.CTkButton(frame,

text="Add Item",

text\_color="black",

command=add\_item,

font=('Goudy Old Style', 16, "bold"), corner\_radius=6, border\_color='white',

hover\_color="green")

add\_item\_btn.grid(row=5, column=2, pady=5)

columns = ('desc', 'qty', 'price', 'total')

tree = ttk.Treeview(frame, columns=columns, show="headings")

tree.heading('qty', text="Qty")

tree.heading('desc', text="Description")

tree.heading('price', text="Unit Price")

tree.heading('total', text="Total")

tree.grid(row=6, column=0, columnspan=3, padx=20, pady=10)

style = ttk.Style(frame)

style.theme\_use('clam')

style.configure("Treeview", background="#3eb489", fieldbackground="white", foreground="white",

font=("Times New Roman", 14, 'italic'))

style.configure("Treeview.Heading", background="#00a8e1", font=("Times New Roman", 15, 'bold'))

# scrollbar

vs = ttk.Scrollbar(frame, orient='vertical', command=tree.yview)

tree.configure(yscrollcommand=vs.set)

vs.grid(row=6, column=4, sticky="ns")

def delete\_fun(\_):

for i in tree.selection():

dlt = []

dlt.append(tree.item(i)['values'])

tree.delete(i)

s\_d = sum(dlt, [])

s\_d[2] = float(s\_d[2])

s\_d[3] = float(s\_d[3])

ind = invoice\_list.index(s\_d)

invoice\_list.pop(ind)

# delete event

tree.bind('<Delete>', delete\_fun)

save\_invoice\_btn = tkinter.Button(frame,

text="Generate Invoice",

command=generate\_invoice,

bg='black', fg='white',

font=('Bell MT', 14))

save\_invoice\_btn.grid(row=7, column=0, columnspan=3, sticky="news", padx=20, pady=5)

new\_invoice\_btn = tkinter.Button(frame, text="New Invoice",

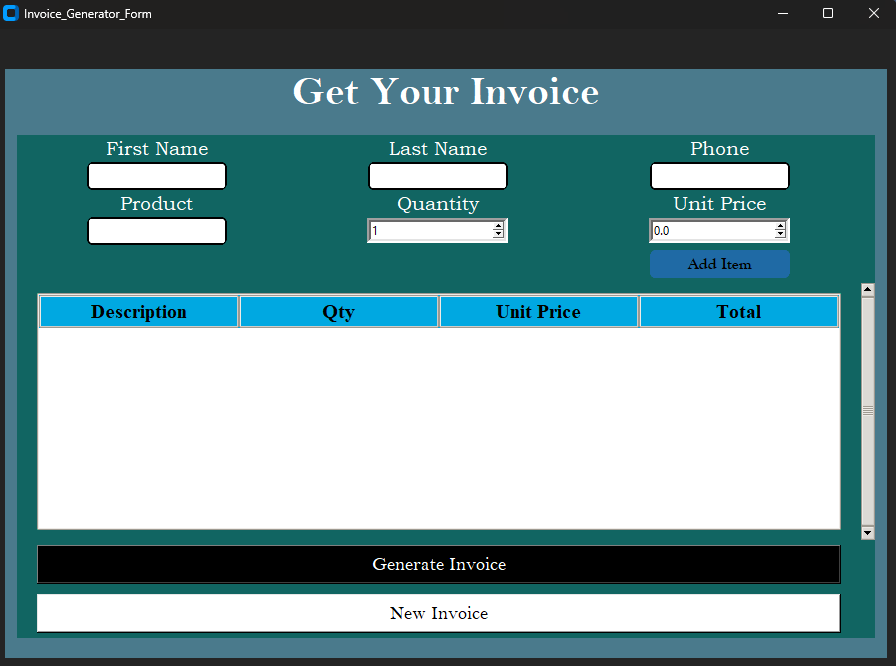
command=new\_invoice, bg='white', fg='black',

font=('Bell MT', 14))

new\_invoice\_btn.grid(row=8, column=0, columnspan=3, sticky="news", padx=20, pady=5)

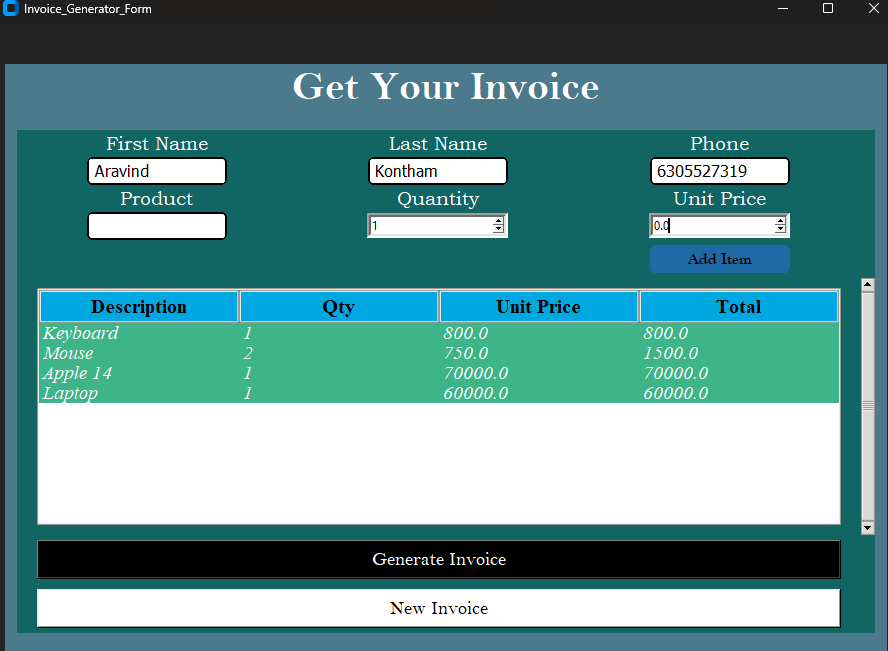
window.mainloop()

**Input/Output Screenshots**



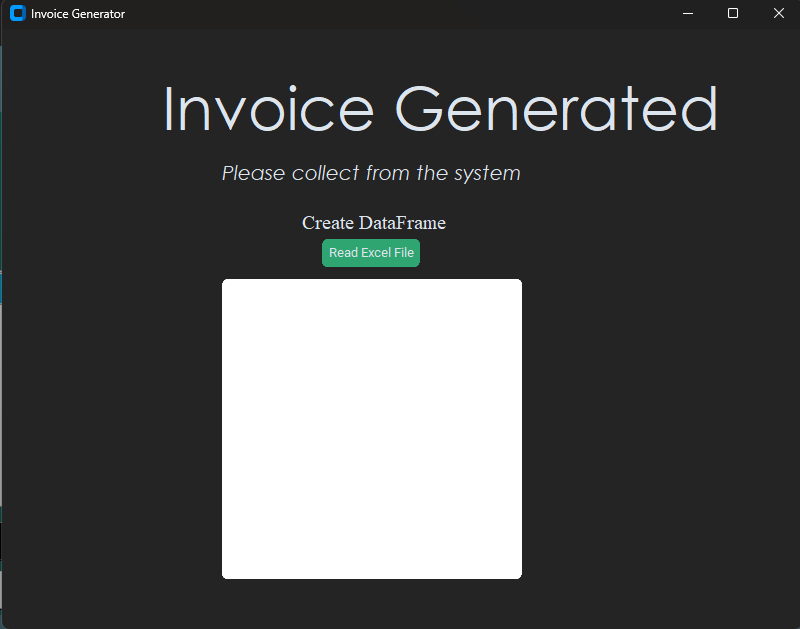
**Figure 1 Initial interface**

* This is the starting interface when the code is run in Pycharm.
* The above tkinter interface provides a brief information what a user can give the data as a input like name, phone.
* It takes the data given by the user in treeview widgets and stores in the word file and excel file at the same time when the add item button I clicked.



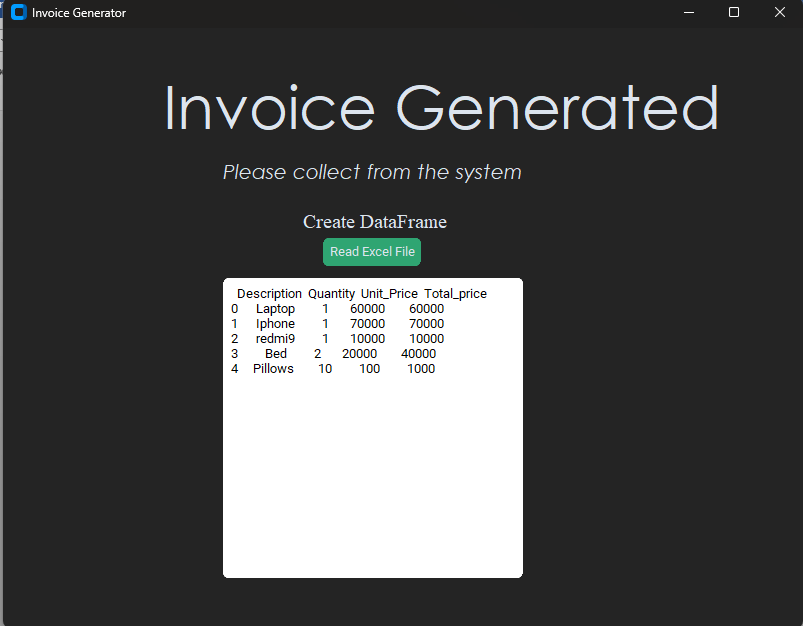
**Figure 2 After entering user data**

* When the user gives the data into the respective fields to the interface, the data is stored in the treeview widget.
* In this case the user enters Laptop, Apple14, Mouse and Keyboard as the product.



**Figure 3 After clicking Generate Invoice**

* After giving the details of the product, when the user click the generate invoice button, this interface open in which there will be a message showing “Invoice Generated” “Please collect from the system”
* This tkinter interface helps the user to view the entered data in the form of dataframe, where the data is read from the excel file and “PANDAS” library converts the data into a dataframe.

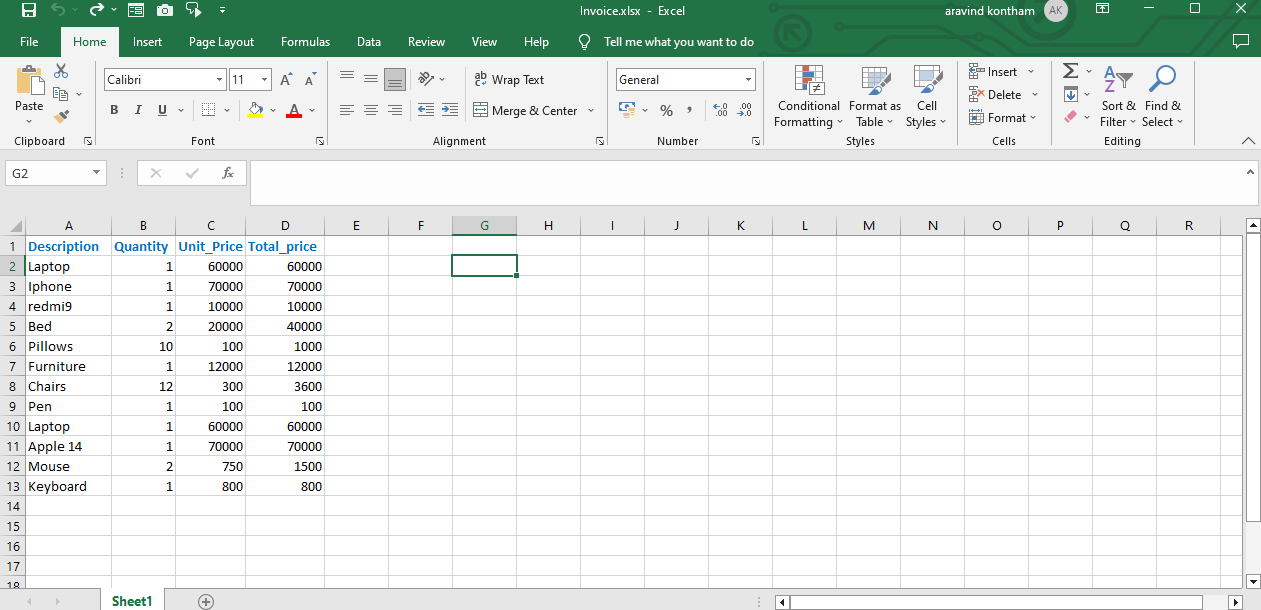


**Figure 4 Showing a dataframe of the data**

* Finally a dataframe is shown in the text box of the interface.
* In this way any user can view and analyze the data that is provided initially to the interface.



**Figure 5Generated Invoice in Word file**



**Figure 6 Data in Excel sheet**

* This is the excel file where the user data is stored for further development of the dataframes.
* And the above page contains the invoice generated in the word file which contains the product details along with the price .
* The final price contains the discounted price and the added GST price.
* In this way this project is very helpful in viewing the invoice of the products based on the given data.
* This project has a very good future scope.

**Scope Of the Project**

1. Integration with Accounting Software: The Invoice Generator could be integrated with popular accounting software like QuickBooks or Xero, allowing for a seamless transfer of invoice data and further streamlining the invoicing process.

2. Multi-Language Support: The application could be developed to support multiple languages, making it accessible to users around the world who may not be comfortable with English.

3. Online Payment Integration: The Invoice Generator could be integrated with online payment systems such as PayPal or Stripe, allowing for easy and secure payment processing.

4. Advanced Reporting: The application could be enhanced to provide advanced reporting features, such as graphical representations of invoice data, trends, and patterns.

5. Mobile Application: The Invoice Generator could be developed as a mobile application, providing users with the ability to manage their invoicing data on-the-go.

6. Cloud Storage: The application could be integrated with cloud storage services like Google Drive or Dropbox, allowing for easy and secure storage of invoice data and making it accessible from anywhere.

Overall, there are many potential future scopes for the Invoice Generator, and these are just a few examples of how the application could be further developed and enhanced to better serve its users.

**DEVELOPMENT**

1. **Integration with Accounting Software**: To integrate the Invoice Generator with accounting software, you would need to research the APIs of the software you want to integrate with and develop code to make the connection. You would also need to develop a user interface for entering the accounting software credentials and configuring the integration. Testing and debugging would be crucial to ensure that the integration works seamlessly.

2. **Multi-Language Support**: To add multi-language support, you would need to research and integrate Python libraries that support translation, such as the gettext library. You would also need to design and develop a user interface for language selection, and then work on translating all text within the application. Testing and debugging would be critical to ensure that the translations are accurate and the application works as intended in all languages.

3. **Online Payment Integration**: To integrate online payment processing, you would need to research and integrate payment processing APIs, such as PayPal or Stripe. You would also need to design and develop a user interface for entering payment information and processing payments. Testing and security would be critical to ensure that payment processing is secure and reliable.

4. **Advanced Reporting**: To add advanced reporting features, you would need to research and integrate Python libraries that support data visualization, such as matplotlib or seaborn. You would then need to design and develop a user interface for accessing the reports and customizing them based on user preferences. Testing and debugging would be crucial to ensure that the reports are accurate and informative.

5. **Mobile Application**: To develop a mobile application, you would need to research and select a mobile development framework, such as React Native or Flutter. You would then need to design and develop a user interface that is optimized for mobile devices, and integrate the existing Python code with the mobile application. Testing and debugging would be critical to ensure that the mobile application works seamlessly and provides a good user experience.

6. **Cloud Storage**: To integrate cloud storage, you would need to research and select a cloud storage provider, such as Google Drive or Dropbox. You would then need to develop code to integrate the application with the cloud storage provider's APIs, and design and develop a user interface for configuring and accessing the cloud storage. Testing and debugging would be important to ensure that the cloud storage integration works seamlessly and that the application data is secure.

**CONCLUSION**

In conclusion, the invoice generator capstone project is a powerful tool for small business owners and entrepreneurs to easily create professional-looking invoices. By using the Python programming language, the project showcases the potential of programming in automating mundane and repetitive tasks.

The project uses the tkinter and docx libraries for the user interface and generating Word documents, respectively. The data is stored in a treeview widget that can be easily exported to an Excel file.

Through the development of this project, several objectives were achieved such as simplifying the invoicing process, reducing manual errors, and providing a user-friendly interface. Additionally, the future scope of the project includes integration with accounting software, multi-language support, online payment integration, advanced reporting, mobile application development, and cloud storage integration.

Overall, this project highlights the potential of programming in streamlining business processes and the importance of continuous development to meet evolving business needs. The project can serve as a starting point for further development and customization to meet specific business requirements, ultimately improving efficiency and productivity.

THANK YOU