Micro Project on Invoice Generator

BY

Dhanya Patel
226040307115

Jay Kansara

226040307050

Vraj Rajan

226040307165

ON

Invoice Generator Using Python

A Micro Project in Python (4330701) Submitted to

Computer (GIA) Department

B & B Institute of Technology, Vallabh Vidyanagar

Table of Content

1. Introduction

1.1 About of Invoice Generator

In the dynamic landscape of modern business operations, the seamless and accurate generation of invoices stands as a crucial element for maintaining financial transparency and facilitating efficient transactions.

The Invoice Generator project represents a comprehensive solution developed in Python to address the burgeoning need for a user-friendly, customizable, and automated tool for creating invoices.

1.2 Use of Invoice Generator

The Invoice Generator, with its user-friendly design and versatile features, serves as a valuable tool for businesses across various sectors. Here are some key use cases and applications for the Invoice Generator:

1. Small and Medium-Sized Businesses (SMEs):
2. Freelancers and Independent Contractors:
3. Entrepreneurs and Startups:
4. Service-Based Industries:
5. Retail and E-Commerce:
6. Customization for Branding:
7. Time and Resource Efficiency:

1.3 Scope of Project

The scope of the Invoice Generator project encompasses various dimensions, addressing the needs of businesses seeking an efficient, customizable, and automated solution for invoice generation. Here's an elaboration on the scope of the project:

1. User-Friendly Design:
2. Customization Options:
3. Automation of Invoicing Process:
4. Adaptability to Business Needs:
5. Accuracy in Financial Calculations:
6. Integration with Business Workflows:
7. Scalability:
8. Documentation and Reporting:

1.4 History of Invoice Generator

The scope of the project extends to future enhancements and updates. It should be flexible enough to incorporate additional features and improvements based on user feedback, technological advancements, and evolving business requirements.

Early Days (1990s):

- The concept of software automating invoices emerged in the early 1990s, alongside the rise of personal computers and business software.
- Initial iterations were simple desktop applications with basic data entry and calculation capabilities.
- Adoption was limited, mostly used by early tech-savvy businesses and freelancers.

Growth and Expansion (2000s):

- The internet boom in the 2000s fueled the rise of web-based Invoice Generators.
- Features expanded to include templates, customization options, and email integration.
- SaaS (Software as a Service) models gained popularity, offering affordable access for even small businesses.

Mobile Revolution (2010s):

- Smartphones and tablets ushered in a new era of mobility for invoice creation.
- Mobile apps emerged, allowing entrepreneurs to send invoices on the go.
- Focus on user-friendliness and intuitive interfaces for easy on-the-spot invoicing.

Modern Landscape (2020s):

Today, Invoice Generators are sophisticated tools packed with advanced features.

Integration with accounting software, payment gateways, and CRMs creates seamless workflows.

1.5 Importance of Invoice Generator

The Invoice Generator holds significant importance for businesses across various industries due to its numerous benefits. Here are some key aspects that highlight the importance of using an Invoice Generator:

- Efficiency and Time Savings:
- Accuracy in Financial Transactions:
- Professionalism and Brand Image:
- Customization for Business Needs:
- Financial Record Keeping:
- Streamlined Billing Processes:
- Accessibility and Collaboration:
- Real-Time Financial Insights:
- Integration with Other Business Tools:

2. Software for Invoice Generator

2.1 Python

Python is a general-purpose high level programming language that's widely utilized in data science and for producing deep learning algorithms.

Python is a simple to find out, powerful programming language. it's efficient high-level data structures and an easy but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, alongside its interpreted nature, make it a perfect language for scripting and rapid application development in many areas on most platforms.

The Python interpreter is definitely extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is additionally suitable as an extension language for customizable applications

2.2 Python libraries

Python libraries are reusable collections of code that provide pre-built functionalities to perform specific tasks. These libraries save developers time and effort by offering pre-implemented solutions for common

programming challenges. In the context of building an invoice generator in Python, you might use libraries for tasks such as PDF generation, templating, and data manipulation.

2.2.1 Dox Generation Libraries:

Libraries that assist in creating word documents programmatically. This is crucial for generating invoices that can be easily shared and printed.

2.2.2 Templating Libraries:

Templating engines allow you to create dynamic templates for documents like invoices. They help separate the structure of the document from the actual data, making it easier to generate consistent and customized invoices.

2.2.3 Data Manipulation Libraries:

Libraries that facilitate the manipulation and formatting of data. This is important for organizing and presenting the information that goes into the invoice.

3. Pseudo Code & Algorithm

3.1 Pseudo Code

```
# Import necessary libraries
import tkinter
from tkinter import ttk
from docxtpl import DocxTemplate
import datetime
from tkinter import messagebox

# Function to clear item inputs
function clear_item():
    qty_spinbox.delete(0, tkinter.END)
    qty_spinbox.insert(0, "1")
    desc_entry.delete(0, tkinter.END)
    price_spinbox.delete(0, tkinter.END)
    price_spinbox.insert(0, "0.0")

# Initialize global variable for invoice items
invoice_list = []

# Function to add item to the invoice
function add_item():
    qty = int(qty_spinbox.get())
    desc = desc_entry.get()
    price = float(price_spinbox.get())
    line_total = qty * price
    invoice_item = [qty, desc, price, line_total]
    tree.insert('', 0, values=invoice_item)
    clear_item()
    invoice_list.append(invoice_item)

# Function to create a new invoice
function new_invoice():
    first_name_entry.delete(0, tkinter.END)
    last_name_entry.delete(0, tkinter.END)
    phone_entry.delete(0, tkinter.END)
    clear_item()
    tree.delete(*tree.get_children())
    invoice_list.clear()

# Function to generate and save the invoice
function generate_invoice():
    doc = DocxTemplate("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)

    salestax = 0.1
    total = subtotal * (1 - salestax)

    # Render and save the invoice
    doc.render({"name": name,
                "phone": phone,
                "invoice_list": invoice_list,
                "subtotal": subtotal,
                "salestax": str(salestax * 100) + "%",
                "total": total})

    doc_name = "new_invoice" + name + datetime.datetime.now().strftime("%Y-%m-
%d-%H%M%S") + ".docx"
    doc.save(doc_name)

    # Display completion message
    messagebox.showinfo("Invoice Complete", "Invoice Complete")

    # Create a new invoice
    new_invoice()

# Create the main window
window = tkinter.Tk()
window.title("Invoice Generator Form")

# ... (UI setup code)

# Run the Tkinter main loop
window.mainloop()
```

3.2 Algorithm

Step 1: Initialize:
Import necessary libraries (tkinter, ttk, DocxTemplate, datetime).
Define global variables.

Step 2: Define Functions:
clear_item: Clears input fields for item details.
add_item: Adds item details to the invoice list and updates the TreeView.
new_invoice: Clears all input fields and the TreeView to start a new invoice.
generate_invoice: Uses the entered details to generate, render, and save a new invoice.

Step 3: Create UI Elements:
Labels, Entry widgets, Spinbox widgets, Buttons, TreeView, etc.

Step 4:
Event Handling:
Attach event handlers to buttons for adding items, generating invoices, and creating new invoices.

Step 5:
Main Program Loop:
Start the Tkinter main loop for the GUI to interact with the user.

4. Source Code

Main.py

```
import tkinter
from tkinter import ttk
from docxtpl import DocxTemplate
import datetime
from tkinter import messagebox

def clear_item():
    qty_spinbox.delete(0, tkinter.END)
    qty_spinbox.insert(0, "1")
    desc_entry.delete(0, tkinter.END)
    price_spinbox.delete(0, tkinter.END)
    price_spinbox.insert(0, "0.0")

invoice_list = []
def add_item():
    qty = int(qty_spinbox.get())
    desc = desc_entry.get()
    price = float(price_spinbox.get())
    line_total = qty*price
    invoice_item = [qty, desc, price, line_total]
    tree.insert('',0, values=invoice_item)
    clear_item()

    invoice_list.append(invoice_item)


def new_invoice():
    first_name_entry.delete(0, tkinter.END)
    last_name_entry.delete(0, tkinter.END)
    phone_entry.delete(0, tkinter.END)
    clear_item()
    tree.delete(*tree.get_children())

    invoice_list.clear()

def generate_invoice():
    doc = DocxTemplate("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)
    salestax = 0.1
    total = subtotal*(1-salestax)

    doc.render({"name":name,
            "phone":phone,
            "invoice_list": invoice_list,
            "subtotal":subtotal,
```

```
            "salestax":str(salestax*100)+"%",
            "total":total})

    doc_name = "new_invoice" + name + datetime.datetime.now().strftime("%Y-%m-
%d-%H%M%S") + ".docx"
    doc.save(doc_name)

    messagebox.showinfo("Invoice Complete", "Invoice Complete")

    new_invoice()

window = tkinter.Tk()
window.title("Invoice Generator Form")

frame = tkinter.Frame(window)
frame.pack(padx=20, pady=10)

first_name_label = tkinter.Label(frame, text="First Name")
first_name_label.grid(row=0, column=0)
last_name_label = tkinter.Label(frame, text="Last Name")
last_name_label.grid(row=0, column=1)

first_name_entry = tkinter.Entry(frame)
last_name_entry = tkinter.Entry(frame)
first_name_entry.grid(row=1, column=0)
last_name_entry.grid(row=1, column=1)

phone_label = tkinter.Label(frame, text="Phone")
phone_label.grid(row=0, column=2)
phone_entry = tkinter.Entry(frame)
phone_entry.grid(row=1, column=2)

qty_label = tkinter.Label(frame, text="Qty")
qty_label.grid(row=2, column=0)
qty_spinbox = tkinter.Spinbox(frame, from_=1, to=100)
qty_spinbox.grid(row=3, column=0)

desc_label = tkinter.Label(frame, text="Description")
desc_label.grid(row=2, column=1)
desc_entry = tkinter.Entry(frame)
desc_entry.grid(row=3, column=1)

price_label = tkinter.Label(frame, text="Unit Price")
price_label.grid(row=2, column=2)
price_spinbox = tkinter.Spinbox(frame, from_=0.0, to=500, increment=0.5)
price_spinbox.grid(row=3, column=2)

add_item_button = tkinter.Button(frame, text = "Add item", command = add_item)
add_item_button.grid(row=4, column=2, pady=5)

columns = ('qty', 'desc', '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=5, column=0, columnspan=3, padx=20, pady=10)


save_invoice_button = tkinter.Button(frame, text="Generate Invoice",
command=generate_invoice)
save_invoice_button.grid(row=6, column=0, columnspan=3, sticky="news", padx=20,
pady=5)

new_invoice_button = tkinter.Button(frame, text="New Invoice",
command=new_invoice)
new_invoice_button.grid(row=7, column=0, columnspan=3, sticky="news", padx=20,
pady=5)


window.mainloop()
```

Doc_gen.py

```
from docxtpl import DocxTemplate

doc = DocxTemplate("invoice_template.docx")

invoice_list = [[2, "pen", 0.5, 1],
                [1, "paper pack", 5, 5],
                [2, "notebook", 2, 4]]


doc.render({"name":"john",
            "phone":"555-55555",
            "invoice_list": invoice_list,
            "subtotal":10,
            "salestax":"10%",
            "total":9})
doc.save("new_invoice.docx")
```

5.References

https://www.youtube.com/watch?v=mJc5gAnnlVQ&t=4s