Symbiosis Artificial Intelligence Institute (SAII)

Symbiosis International University

A Mini Project Report On

'BUILD A PYTHON APPLICATION: BILLIT'

Submitted By:

Vasva Kumar - 25030422129

Vinayak Goyal - 25030422133

Parth Arora - 25030422082

Submitted To:

Dr. Dawa Chyophel Lepcha

(Assistant professor)

Submitted on – 05/11/2025

CERTIFICATE

This is to certify that the that the project work titled ‘Build A Python Application: BILLIT’ was completed and carried out under the direct supervision and guidance of the department of SAII by Vinayak Goyal, Vasva Kumar and Parth Arora of BBA(AI) Section-B.

Signature of Faculty

DECLARATION

We hereby as a team [Vinayak Goyal, Vasva Kumar and Parth Arora] that the project entitled ‘Build A Python Application: BILLIT’ is a record of an original work done by us under the guidance and supervision of the department of SAII and this project is submitted in the partial fulfilment of the requirements of the project. We further declare that this work is authentic and has not been copied from any other source.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude and thanks to our teacher Mr. Dawa chyophel Lepcha and the department of SAII for their continuous support and guidance throughout the process of completing this Project

ABSTRACT

The project “BILLIT – A Simple Billing Software” is a desktop-based application developed using Python’s Tkinter library to simplify the process of billing and transaction management for small-scale businesses and retail shops. The primary objective of this project is to design a lightweight, user-friendly, and efficient system that helps calculate the total cost of products based on inputs such as product name, quantity, and price. The software provides an intuitive graphical user interface where users can easily enter details and instantly view the total bill amount. It incorporates basic validation techniques to ensure accurate data entry and minimize user errors. The project demonstrates the practical use of GUI programming, event-driven logic, and input handling in Python. By focusing on simplicity and reliability, BILLIT provides an accessible tool for small vendors who may not require advanced inventory or database management systems. It is also an excellent learning project for students beginning with GUI-based application development. Compared to existing billing software, BILLIT stands out for its minimal setup, low resource usage, and ease of customization. Overall, the project successfully bridges theoretical programming knowledge with practical application, resulting in a compact yet effective billing solution suited for real-world use.

TABLE OF CONTENTS

INTRODUCTION

The motivation behind developing **BILLIT** is to create a **simple, interactive, and efficient** billing system that automates the calculation of total cost based on user inputs.

Many small vendors, learners, or individuals performing repetitive billing tasks can benefit from a lightweight solution that doesn't require database setup or complex installation.

Additionally, for computer science students, this project serves as an **excellent learning exercise** in GUI programming, event-driven logic, and exception handling.

PROBLEM STATEMENT

In most small shops and local businesses, billing is still done manually using calculators or handwritten notes. This manual process often leads to mistakes in price calculation, missing records, and wasted time. Even though many billing software systems exist, most of them are complex, costly, and require technical setup, making them unsuitable for small-scale users or beginners learning programming.

There is therefore a clear need for a simple, user-friendly, and efficient billing application that can quickly compute the total cost of a product based on quantity and price without requiring database integration or advanced configurations. Such an application should be easy to use, accurate, and reliable.

The BILLIT project aims to solve this problem by developing a basic billing system using Python's Tkinter library. The software allows users to enter a product name, its quantity, and price, and automatically calculates the total cost. The result is displayed instantly using a message box, making billing faster and error-free.

In summary, the problem addressed by this project is the lack of a lightweight, easy-to-use billing solution for small-scale users. BILLIT provides a minimal yet effective approach to automate simple billing tasks, demonstrating how Python programming can be applied to real-world problems in a practical and efficient manner.

LITERATURE REVIEW / RELATED WORK

Billing systems have evolved significantly over the years, moving from manual calculations to automated digital solutions. Traditionally, small businesses and shops maintained handwritten records and used calculators to compute totals. This manual process was slow, error-prone, and lacked data storage or tracking capabilities. As businesses began adopting computers, spreadsheet-based billing (like Microsoft Excel) became common, but it still required manual data entry and formulas for each transaction.

METHODOLOGY

System Design

The system design for BILLIT follows a straightforward **input–process–output** model:

- **Input:** The user enters three pieces of information — Product Name, Quantity, and Price.
- **Process:** The program validates the inputs, converts the numeric values, and multiplies Quantity × Price to calculate the total.
- **Output:** The total amount is displayed in a pop-up message box using Tkinter's messagebox.showinfo() function.

This design ensures that every step is simple, linear, and easily understood by beginners.

Tools and Technologies Used

1. **Programming Language:** Python 3.x
2. **Library:** Tkinter (for GUI creation)
3. **IDE Used:** Visual Studio Code / IDLE / PyCharm
4. **Platform:** Windows OS
5. **Key Modules:**
    - tkinter: for creating GUI components like Labels, Entry boxes, and Buttons.
    - messagebox: for displaying results and error messages.

Program Flow

The project follows an **event-driven flow**, meaning the application waits for user actions and responds accordingly. The flow is as follows:

1. **Start the Program** → Tkinter window opens.
2. **User Input** → Product name, quantity, and price are entered.
3. **Button Click** → The “Add Item to Bill” button triggers the function billing_soft().
4. **Validation** → Inputs are checked to ensure Quantity and Price are numeric values.
5. **Computation** → The total amount is calculated as quantity × price.

6. **Display Output** → The total cost is shown in a message box.
7. **Error Handling** → If inputs are invalid, an error message appears.

Algorithm / Pseudocode

```
Step 1: Start
Step 2: Create GUI window
Step 3: Take user input for product name, quantity, and price
Step 4: When button is clicked:
          → Validate that quantity and price are numbers
          → Compute total = quantity * price
          → Display total in message box
          → If invalid input, show error message
Step 5: End
```

System Interface

The interface is designed for ease of use with minimal clutter. Each element is arranged vertically:

- Labels guide the user for each field.
- Entry boxes allow typing of data.
- A single button triggers the billing process.
  The background colour and fonts were chosen for clarity and readability.

Error Handling and Validation

The project uses Python's **try–except** structure to handle errors gracefully. If the user enters non-numeric values for Quantity or Price, the system displays an error message instead of crashing. This ensures reliability and user confidence while using the software.

Summary

The chosen methodology focuses on **simplicity, clarity, and robustness**. By using Tkinter, the project demonstrates how even basic Python programming concepts can produce an interactive and functional desktop application. The structured design approach ensures that the system is easy to understand, modify, and extend for future improvements such as multi-item billing, database storage, or invoice generation.

IMPLEMENTATION

The project was implemented in Visual Studio Code, using Tkinter for graphical user interface development. The entire code is contained within a single Python file, making it lightweight and easy to execute. The GUI components — such as labels, entry boxes, and buttons — are used to interact with the user, while the backend logic handles data validation and total cost calculation.

The application window (Tk() object) is created with a defined size and background color. It contains three input fields:

1. **Product Name** – Text input for entering the product name.
2. **Quantity** – Numeric input for the number of units.
3. **Price** – Numeric input for the price per unit.

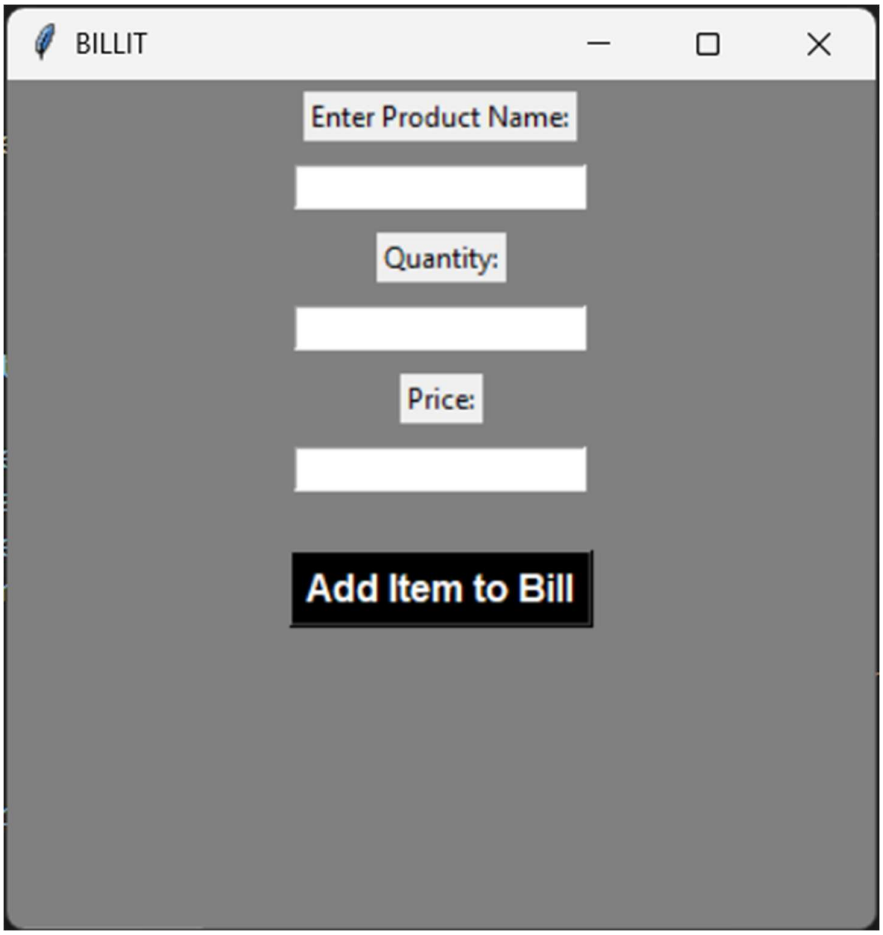
RESULTS AND DISCUSSION

Screenshot of the code

Billlt.py

```python
import tkinter as tk
from tkinter import *
from tkinter import messagebox

project = tk.Tk()
project.geometry("800x500")
project.title("BILLIT")
project.config(background="grey")

def billing_soft():
    try:
        product_name = product_name_entry.get()
        quantity = quantity_entry.get()
        price = price_entry.get()
        quantity_value = int(quantity)
        product_price = int(price)
        total_price = quantity_value*product_price
        messagebox.showinfo("Item Added",
                            f"Product: {product_name}\n"
                            f"Quantity: {quantity_value}\n"
                            f"Price: ₹{product_price:.2f}\n"
                            f"Item Total: ₹{total_price:.2f}")
    except ValueError:
        messagebox.showerror("Input Error", "Please ensure Quantity is a whole number and Price is a valid decimal number.")

#label for product_name
Label(project ,text="Enter Product Name:").pack(pady=5)
product_name_entry = Entry(project)
product_name_entry.pack(pady=5)

#label for quantity
Label(project, text="Quantity:").pack(pady=5)
quantity_entry = Entry(project)
quantity_entry.pack(pady=5)

# label for Price
Label(project, text="Price:").pack(pady=5)
price_entry = Entry(project)
price_entry.pack(pady=5)

Button(project, text="Add Item to Bill", command = billing_soft ,
       font=('Arial', 12, 'bold'), bg='black', fg='white').pack(pady=20)

project.mainloop()
```

Screenshot of the Output



APPLICATIONS AND LIMITATIONS

The BILLIT project has been designed as a simple and practical tool that demonstrates how programming can be applied to solve real-world problems. Despite being a small-scale application, it has several useful applications in both educational and commercial contexts.

Although the project achieves its goal of simplifying billing through automation, it has certain limitations due to its lightweight and prototype nature. Identifying these limitations is important for understanding the system's current boundaries and opportunities for future improvement.

- Single Item Limitation
- No Data Storage
- No Print or Save Option
- Lack of Tax or Discount Calculations
- Limited Interface Design
- No Database Integration
- Platform Limitation

CONCLUSION AND FUTURE WORK

Conclusion

The **BILLIT** project successfully demonstrates how Python's **Tkinter library** can be used to create a simple yet effective graphical user interface application for billing purposes. The main goal of the project — to design a lightweight, easy-to-use tool that calculates the total cost of a product based on user input — has been achieved effectively.

Throughout the development process, emphasis was placed on **simplicity, usability, and reliability**. The software allows users to input basic product details such as name, quantity, and price, and instantly computes the total amount. Its error-handling mechanism ensures that the application remains stable even when invalid inputs are entered, making it robust and user-friendly.

Future Work

In the future, the **BILLIT** project can be enhanced in several ways to make it more functional and practical for real-world use. The current version handles only one product at a time, so an important improvement would be to enable **multi-item billing**, allowing users to generate complete bills with multiple entries. Integration with a **database system** such as MySQL or SQLite would help store and retrieve past billing records, providing better data management. The system can also be upgraded to **generate invoices automatically** in formats like PDF or text files for printing and sharing. Additionally, incorporating features such as **tax and discount calculations** would make the software more suitable for commercial use. The **user interface** can be enhanced with modern design elements like tables, themes, and menus to make it visually appealing and easy to navigate. Introducing a **login system** would further improve security by maintaining user-specific billing records. These improvements would transform BILLIT from a simple billing calculator into a complete, professional billing and invoicing solution.

REFERENCES

1. Python Software Foundation. (2024). *Tkinter — Python Interface to Tcl/Tk*. Retrieved from https://docs.python.org/3/library/tkinter.html
2. TutorialsPoint. (2023). *Tkinter Programming – Python GUI*. Retrieved from https://www.tutorialspoint.com/python/python_gui_programming.htm
3. Real Python. (2024). *Python GUI Programming With Tkinter*. Retrieved from https://realpython.com/python-gui-tkinter/
4. Stack Overflow. (2024). *Discussions and Solutions for Python Tkinter Projects.* Retrieved from https://stackoverflow.com
5. Reitz, K., & Schlusser, T. (2020). *The Hitchhiker's Guide to Python: Best Practices for Development.* O'Reilly Media.
6. Lutz, M. (2021). *Programming Python (5th Edition).* O'Reilly Media.
7. YouTube Tutorials. (2024). *Creating Simple Billing System in Python Using Tkinter.*

Thank You!