	ЗВО: Національний університет «Львівська політехніка» **Навчальний рік:** 2024/2025 **Семестр:** весняний	**Навчальна дисципліна:** Розробка кросплатформених додатків **Лабораторна робота № 3:** Розроблення додатку для управління каталогом товарів
	Кафедра систем автоматизованого проектування **Викладач:** Юрій ПАТЕРЕГА	**Група:** ПП-26 **Студент:** Ігор ЯКІБ'ЮК

Мета роботи

Розробити додаток на основі бібліотеки tkinter для управління каталогом товарів.
Додаток повинен містити:

- групу віджетів для вибору та перегляду даних, зокрема, список/дерево та смугу прокрутку для переглядання каталогу товарів, який містить назву товару, його ціну та доступність на складі;
- групу віджетів для введення та редагування даних, зокрема, текстове поле для введення назви товару, повзунок для встановлення ціни та випадаючий список для вибору доступності.
- групу кнопок для управління даними у списку/дереві, зокрема, додавання товару до списку, видалення вибраного товару зі списку, завантаження даних вибраного товару у віджети введення та оновлення у списку/дереві інформації про вибраний товар.

Інструкція до виконання роботи

1. Вибрати категорію товарів згідно із вашим номером варіанту:

1	2	3	4	5	6	7	8
Ноутбуки	Монітори	Материнські плати	Процесори	Оперативна пам'ять	Жорсткі диски (HDD)	Твердотільні накопичувачі (SSD)	Відеокарти
9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Блоки живлення	Корпуси для ПК	Системи охолодження	Мережеве обладнання	Клавіатури	Миші	Вебкамери	Принтери
17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Сканери	Графічні планшети	Акустичні системи	Навушники	Мікрофони	Ігрові контролери	Флеш-накопичувачі	Кабелі та перехідники
25	**26**	**27**	**28**	**29**	**30**	**31**	**32**
Сервери та серверні комплектуючі	Робочі станції	NAS-сховища та хмарні сервери	Оптоволоконне обладнання (SFP-модулі, конвертери)	VPN-роутери та захисні шлюзи	Операційні системи та офісні пакети	IP-камери та системи відео-спостереження	Криптовалютні майнінг-ферми та ASIC-майнери

2. Створити головне вікно та налаштувати його параметри.
3. Додати віджети для вибору та переглядання даних.
3.1. Додати фрейм (Frame) для об'єднання віджетів вибору та переглядання даних.
3.2. Додати дерево (ttk.Treeview) з трьома колонками (назва товару, ціна, статус).
3.3. Додати смугу прокрутки для переглядання великих каталогів.
4. Додати віджети для введення та редагування даних.
4.1. Додати фрейм (LabelFrame) для об'єднання віджетів керування.
4.2. Додати текстове поле (Entry) для введення назви товару.
4.3. Додати повзунок (Scale) для встановлення ціни товару.
4.4. Додати випадаючий список (Combobox) для вибору статусу товару.
5. Додати кнопки для управління даними.
5.1. Додати фрейм (LabelFrame) для об'єднання кнопок.
5.2. Додати кнопку «Insert» та обробник для додавання нового товару.
5.3. Додати кнопку «Delete» та обробник для видалення вибраного товару.
5.4. Додати кнопку «Get» та обробник для завантаження вибраного товару в поля введення.
5.5. Додати кнопку «Set» та обробник для оновлення даних щодо вибраного товару.

6. Запустити головний цикл обробки подій (mainloop()).

1. Теоретичні відомості

2. Лістинг коду програми

```
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import messagebox
import google.generativeai as genai
import threading
import re

root = tk.Tk()
root.title("Розумний список")
root.geometry("670x550+350+0")

# important DATA
columns = ("Назва", "Ціна", "Статус")
statusData = ["Критично необхідно", "Варто придбати", "За бажанням/Необов'язково"]
API_KEY = 'AIzaSyCBWehUMh-LGBzA2ddnmcXyObMrZ2CwSt0'
genai.configure(api_key=API_KEY)

# important functions

def ask_gemini(prompt):
    try:
        model = genai.GenerativeModel("gemini-2.0-flash")
        response = model.generate_content(prompt)
        return response.text
    except Exception as e:
        return f"Помилка AI: {str(e)}"

def process_analysis():
    try:
            items = treeView.get_children()
            catalog_text = "Каталог товарів:\n"
            for item in items:
                values = treeView.item(item, "values")
                catalog_text += f"{values[0]} - {values[1]}, {values[2]}\n"

            prompt_analysis = f"Проаналізуй цей список Оптоволоконне обладнання (SFP-
модулі, конвертери). Визнач, що необхідно додати зі Статусом. Можливі статуси: 'Критично
необхідно', 'Варто придбати', 'За бажанням/Необов'язково'. А що можна забрати
\n{catalog_text}"
            analysis_response = ask_gemini(prompt_analysis)

            messagebox.showinfo("Рекомендації AI", analysis_response)
    except Exception as e:
            error_message = f"Помилка під час аналізу: {str(e)}"
            root.after(0, messagebox.showerror, "Помилка", error_message)

def add_products_AI():
    try:
            items = treeView.get_children()
            catalog_text = "Каталог товарів:\n"
            for item in items:
                values = treeView.item(item, "values")
                catalog_text += f"{values[0]} - {values[1]}, {values[2]}\n"
            new_list = f"На основі цього списку: \n{catalog_text}. Добав те, чого не
вистачає або повністю зміни список для найкращого результату. Поверни його у такому
форматі, у якому отримав, і лише список. Ціну вказуй лише цифри. Можливі статуси:
'Критично необхідно', 'Варто придбати', 'За бажанням/Необов'язково'. І поверни цей
```

```
список"
            new_products_response = ask_gemini(new_list)


            new_list = []
            for line in new_products_response.split("\n"):
                    parts = line.split(" - ")
                    if len(parts) == 2:
                        price_and_status = parts[1].split(',')
                        newPart = [parts[0], float(price_and_status[0]),
price_and_status[1]]
                        if len(newPart) == 3:
                            name = newPart[0].strip()
                            price = newPart[1]
                            availability = newPart[2].strip()
                        name = re.sub(r'^\d+\.\s*', '', name)
                        name = re.sub(r'^[\*\s]+', '', name)

                        if  "Критично необхідно" in availability:
                            availability = "Критично необхідно"
                        elif "Варто придбати" in availability:
                            availability = "Варто придбати"
                        else:
                            availability = "За бажанням/Необов'язково"
                        try:
                            price = float(price)
                            new_list.append((name, f"{price:.2f}", availability))
                        except ValueError:
                            continue
            # print(new_products_response)
            for product in new_list:
                    name, price, availability = product
                    treeView.insert("", "end", values=product)
    except Exception as e:
            error_message = f"Помилка під час аналізу: {str(e)}"
            root.after(0, messagebox.showerror, "Помилка", error_message)

def clearInputs():
        entry_var.set("")
        price_scale.set(0)
        status_var.set(value="Варто придбати")

def insertGood():
    godName = entry_var.get()
    if godName:
        treeView.insert("", "end", values=(godName, round(price_scale.get(), 2),
status_var.get()))
        clearInputs()
    else:
        return messagebox.showwarning("Помилка створення", "виберіть назву для товару")

def update_price_label(value):
    price_label.config(text=f"{float(value):.2f} ₴")

def decrement():
    price_scale.set(price_scale.get() - 1)
    update_price_label(price_scale.get())

def increment():
    price_scale.set(price_scale.get() + 1)
    update_price_label(price_scale.get())

def deleteGood():
    id = treeView.selection()
```

```
    if not id:
        return messagebox.showwarning("Помилка видалення", "виберіть товар")
    treeView.delete(id[0])

def getGood():
    id = treeView.selection()
    if not id:
        return messagebox.showwarning("Помилка отримання", "виберіть товар")
    value = treeView.item(id[0], "values")
    entry_var.set(value=value[0])
    price_scale.set(float(value[1]))
    update_price_label(price_scale.get())
    status_var.set(value=value[2])

def changeGood():
    selected_item = treeView.selection()
    if not selected_item:
       return messagebox.showwarning("Помилка редагування", "виберіть товар")
    treeView.item(selected_item, values=(entry_var.get(), f"{price_scale.get():.2f}",
status_var.get()))
    clearInputs()



# main frame
mainFrame = tk.Frame(root)
mainFrame.grid(row=0, sticky="nsew")


mainFrame.grid_rowconfigure(0, weight=1)
mainFrame.grid_rowconfigure(1, weight=0)
mainFrame.grid_columnconfigure(0, weight=1)


# Scrollbar
scrollbar = tk.Scrollbar(mainFrame)
scrollbar.grid(row=1, column=1, sticky="ns")

# TreeView

treeView = ttk.Treeview(
    mainFrame,
    columns=columns,
    height=10,
    show="headings",
    yscrollcommand=scrollbar.set,
)

style = ttk.Style()
style.configure("Treeview", font=("Arial", 12))
style.configure("Treeview.Heading", font=("Arial", 14, "bold"))

for col in columns:
    treeView.heading(col, text=col)
    treeView.column(col, anchor="center")

treeView.grid(row=1, column=0, padx=10, pady=10, sticky="ew")
scrollbar.config(command=treeView.yview)


fiber_optic_items = [
    ("SFP Модуль Cisco", 1500, "Критично необхідно"),
    ("SFP+ Модуль 10G", 2800, "Критично необхідно"),
    ("Медіаконвертер TP-Link", 1200, "Варто придбати"),
```

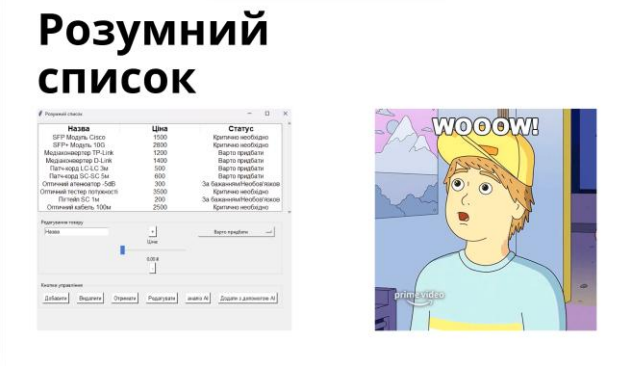
```
    ("Медіаконвертер D-Link", 1400, "Варто придбати"),
    ("Патч-корд LC-LC 3м", 500, "Варто придбати"),
    ("Патч-корд SC-SC 5м", 600, "Варто придбати"),
    ("Оптичний атенюатор -5dB", 300, "За бажанням/Необов'язково"),
    ("Оптичний тестер потужності", 3500, "Критично необхідно"),
    ("Пігтейл SC 1м", 200, "За бажанням/Необов'язково"),
    ("Оптичний кабель 100м", 2500, "Критично необхідно")
]

for item in fiber_optic_items:
    treeView.insert("", "end", values=item)

# data controls & inputs
controlsFrame = ttk.LabelFrame(mainFrame, text="Редагування товару")
controlsFrame.grid(row=2, column=0, padx=10, pady=10, sticky="ew")
controlsFrame.grid_columnconfigure(0, weight=1)
controlsFrame.grid_columnconfigure(1, weight=1)
controlsFrame.grid_columnconfigure(2, weight=1)

#entry
entry_var = tk.StringVar(value="Назва")
entry = tk.Entry(
    controlsFrame,
    font=("Arial", 10, "normal"),
    textvariable=entry_var,
    width=13
)
entry.grid(row=0, column=0, padx=15, sticky="ew")

# scale
ttk.Label(controlsFrame, text="Ціна:").grid(row=1, column=1)
price_scale = ttk.Scale(controlsFrame, from_=0, to=10000, orient="horizontal",
command=update_price_label)
price_scale.grid(padx=15, row=2, column=1, sticky="ew")
price_label = ttk.Label(controlsFrame, text="0.00 ₴")
price_label.grid(row=3, column=1)

btnDec = tk.Button(
    controlsFrame,
    text="-",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=decrement
)
btnDec.grid(row=4, column=1, padx=10)

btnInc = tk.Button(
    controlsFrame,
    text="+",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=increment
)
btnInc.grid(row=0, column=1)


#Option
status_var = tk.StringVar(value="Варто придбати")
statusMenu = tk.OptionMenu(
    controlsFrame, status_var, *statusData)
statusMenu.grid(row=0, column=2, padx=15, sticky="ew")
```

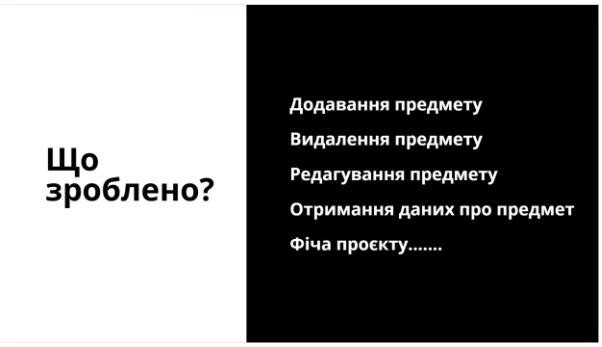
```
# buttons controls
btnFrame = ttk.LabelFrame(mainFrame, text="Кнопки управління")
btnFrame.grid(row=3, column=0, padx=10, pady=10)

btnAdd = tk.Button(
    btnFrame,
    text="Добавити",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=insertGood
)
btnAdd.grid(row=3, column=0, padx=10, pady=10)

btnDelete = tk.Button(
    btnFrame,
    text="Видалити",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=deleteGood
)
btnDelete.grid(row=3, column=1, padx=10, pady=10)

btnGet = tk.Button(
    btnFrame,
    text="Отримати",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=getGood
)
btnGet.grid(row=3, column=2, padx=10, pady=10)

btnSet = tk.Button(
    btnFrame,
    text="Редагувати",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=changeGood
)
btnSet.grid(row=3, column=3, padx=10, pady=10)

btnAI = tk.Button(
    btnFrame,
    text="аналіз AI",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=process_analysis
)
btnAI.grid(row=3, column=4, padx=10, pady=10)

btnAIAdd = tk.Button(
    btnFrame,
    text="Додати з допомогою AI",
    font=("Arial", 10),
    bg="white",
    fg="black",
    command=add_products_AI
)
btnAIAdd.grid(row=3, column=5, padx=10, pady=10)
```

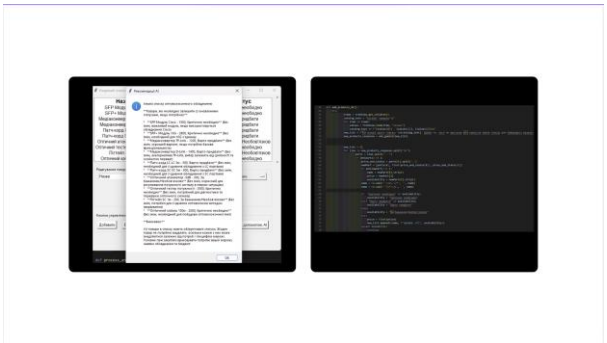
```
root.mainloop()
```

3. Перевірка та тестування програми

3.1. Перевірка додавання нового товару

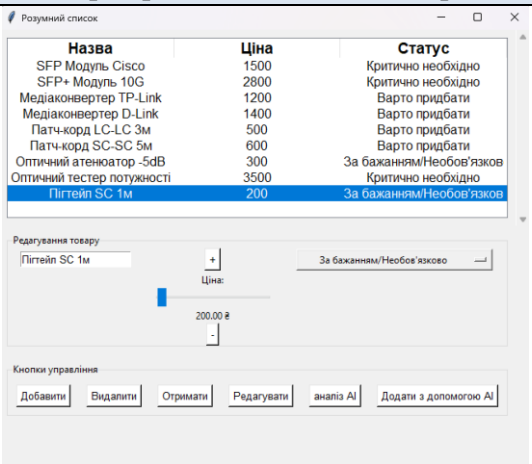




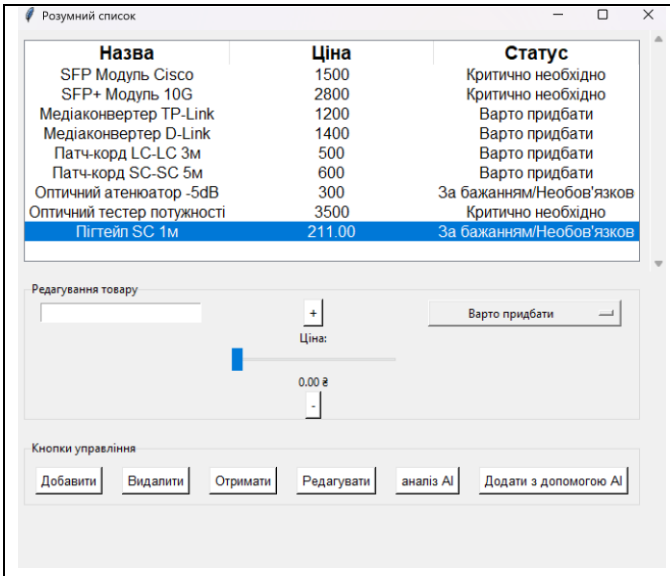


3.2. Перевірка видалення вказаного товару

3.3. Перевірка завантаження вибраного товару в поля введення



3.4. Перевірка оновлення даних щодо вибраного товару



Висновок

Висновок має відповісти на запитання «Що зроблено?», «Як зроблено?», «Що це дало?».

На даній лабораторній роботі, я навчився працювати з бібліотекою ttk. Ознайомився з віджетами: listBox, treeview. Закріпив теоретичний матеріал, виконавши лабораторну роботу згідно з своїм варіантом. Добавив список, якому доступні всі функції редагування: створення, видалення, редагування, отримання – товару. Особливістю цього проєкту – інтеграція AI, яке проводить аналіз нашого списку та дає корисний фідбек, а також має змогу на власний розсуд добавляти необхідні предмети до списку. Виконана робота продемонстрована вище.