ДОДАТОК А

Код моделі товарів

Лістинг Д.1 – Модель товару (Cards) для MongoDB з використанням Mongoose

```
// Імпорт необхідних об'єктів з бібліотеки mongoose для створення схеми та моделі
const { Schema, model } = require("mongoose");

// Опис схеми для колекції товарів (Cards)
const schema = new Schema(
  {
    // Масив URL-адрес зображень товару, обов'язкове поле
    imageUrl: {
      type: Array,
      required: true,
    },
    // Назва товару, обов'язкове поле
    title: {
      type: String,
      required: true,
    },
    // Торгова марка (бренд) товару, обов'язкове поле
    brand: {
      type: String,
      required: true,
    },
    // Доступні розміри товару (масив), обов'язкове поле
    sizes: {
      type: Array,
      required: true,
    },
    // Матеріал (тканина), з якого виготовлено товар, обов'язкове поле
    fabric: {
```

```
    type: String,
    required: true,
  },
  // Ціна товару у числовому форматі, обов'язкове поле
  price: {
    type: Number,
    required: true,
  },
  // Категорія товару у вигляді числового ідентифікатора, обов'язкове поле
  category: {
    type: Number,
    required: true,
  },
  // Рейтинг товару, числове значення, за замовчуванням 0
  rating: {
    type: Number,
    default: 0,
  },
  // Номер замовлення або кількість замовлень товару, за замовчуванням 0
  order_number: {
    type: Number,
    default: 0,
  },
  // Опис товару, обов'язкове поле
  description: {
    type: String,
    required: true,
  },
  // Розміри або виміри товару, масив, за замовчуванням порожній
  measures: {
    type: Array,
    default: [],
```

```
        },
        // Відсоток знижки або інше числове значення, за замовчуванням 0
        procent: {
          type: Number,
          default: 0,
        },
        // Поле для зберігання векторного представлення зображення товару
        imageVector: {
          type: [Number],  // Масив чисел, що описують вектор зображення
          default: [],     // За замовчуванням порожній масив
        },
        // Поле для зберігання векторного представлення текстового опису товару
        textVector: {
          type: [Number],  // Масив чисел, що описують вектор тексту
          default: [],     // За замовчуванням порожній масив
        },
      },
      {
        // Автоматичне додавання полів createdAt та updatedAt із часовими мітками
        timestamps: true,
      }
    );

    // Експорт моделі Cards на основі описаної схеми для подальшого використання
    module.exports = model('Cards', schema);
```

ДОДАТОК Б

Макет головної сторінки інтернет-магазину (landing page)

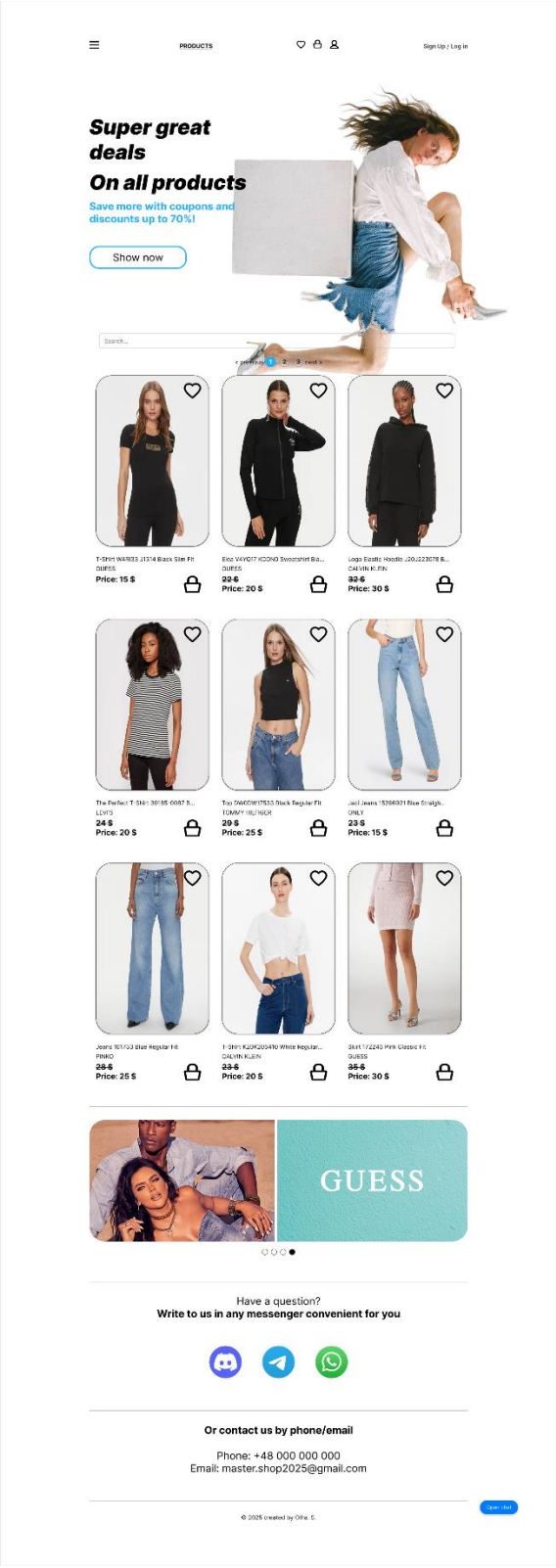

Рисунок Б.1 – Макет головної сторінки сайту

ДОДАТОК В

Код NLP моделі

Лістинг В.1 – Код нейронної мережі на PyTorch

```
import torch.nn as nn   # Імпорт модуля для побудови нейронних мереж з PyTorch

class NeuralNet(nn.Module):  # Визначення класу нейронної мережі, що наслідує nn.Module
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()  # Ініціалізація базового класу

        # Перший лінійний шар, який перетворює вхідний вектор розміром input_size
        # у вектор розміром hidden_size
        self.l1 = nn.Linear(input_size, hidden_size)

        # Другий лінійний шар, який приймає вектор розміром hidden_size і повертає вектор такого ж розміру
        self.l2 = nn.Linear(hidden_size, hidden_size)

        # Третій (вихідний) лінійний шар, що перетворює вектор розміром hidden_size
        # у вектор розміром num_classes – кількість класів для класифікації
        self.l3 = nn.Linear(hidden_size, num_classes)

        # Функція активації ReLU (Rectified Linear Unit) для нелінійності
        self.relu = nn.ReLU()

    def forward(self, x):  # Метод прямого проходу (forward propagation)
```

```
        out = self.l1(x)  # Передача вхідних даних через перший лінійний шар
        out = self.relu(out)    # Застосування функції активації ReLU

        out = self.l2(out)  # Передача результату через другий лінійний шар
        out = self.relu(out)  # Знову застосування ReLU для нелінійності

        out = self.l3(out)    # Передача через третій (вихідний) лінійний шар

        # На виході немає функції активації (наприклад, softmax),
        # оскільки її зазвичай застосовують окремо під час обчислення функції втрат
        return out
```

ДОДАТОК Г

Модель пошуку схожих товарів за вектором зображення на базі CLIP

Лістинг Г.1 – Програмний код функції пошуку схожих товарів за вектором зображення із використанням моделі CLIP

```
import torch
from transformers import CLIPProcessor, CLIPModel
from PIL import Image
import io
import numpy as np

# Визначення пристрою для обчислень: GPU (cuda) якщо доступний, інакше CPU
device = "cuda" if torch.cuda.is_available() else "cpu"

# Завантаження попередньо навчених моделей CLIP та препроцесора
model = CLIPModel.from_pretrained("patrickjohncyh/fashion-clip").to(device)
processor = CLIPProcessor.from_pretrained("patrickjohncyh/fashion-clip")

def cosine_similarity(a, b):
    """
    Обчислення косинусної схожості між двома векторами a і b.
    Параметри:
    - a, b: вектори у вигляді списку або numpy-масиву.
    Повертає: скалярне значення схожості (float).
    """
    a_tensor = torch.tensor(a, dtype=torch.float32).to(device)  # Конвертація в тензор і перенесення на device
```

```
    b_tensor = torch.tensor(b, dtype=torch.float32).to(device)
    similarity = torch.nn.functional.cosine_similarity(a_tensor, b_tensor, dim=-1)  # Обчислення косинусної схожості
    return similarity.item()  # Повернення значення як число

def find_similar_by_image_vector(image_bytes, products, top_k=5, image_weight=0.7, text_weight=0.3):
    """
    Пошук найбільш схожих товарів за вектором зображення.
    Параметри:
    - image_bytes: байти зображення користувача.
    - products: список товарів, кожен з полями imageVector і textVector.
    - top_k: кількість найкращих результатів.
    - image_weight, text_weight: ваги для комбінування схожості за зображенням і текстом.
    Повертає: список топ-товарів із відсотком схожості.
    """
    print("1. Отримання вектору зображення клієнта...")
    # Завантаження зображення з байтів і конвертація у RGB формат
    image = Image.open(io.BytesIO(image_bytes)).convert("RGB")

    # Підготовка зображення для моделі CLIP
    inputs_image = processor(images=image, return_tensors="pt").to(device)

    # Отримання вектору ознак зображення через модель CLIP
    query_vector = model.get_image_features(**inputs_image)
```

```
    # Нормалізація вектора (піднесення до одиничної довжини)
    query_vector = query_vector / query_vector.norm(p=2, dim=-1, keepdim=True)
    print("   Вектор зображення клієнта отримано!")

    image_vectors = []
    text_vectors = []
    valid_indices = []

    print("2. Підготовка векторів товарів...")
    # Обробка кожного товару у списку
    for i, product in enumerate(products):
        img_vec = product.get("imageVector")    # Вектор зображення товару
        txt_vec = product.get("textVector")   # Текстовий вектор товару

        # Пропуск товарів без векторів
        if img_vec is None and txt_vec is None:
            continue

        print(f"   Обробляємо товар {product.get('title', 'Без назви')}...")

        # Нормалізація векторів, якщо вони є, інакше заміна на вектори нулів
        if img_vec:
            img_tensor = torch.tensor(img_vec, dtype=torch.float32).to(device)
            img_tensor = img_tensor / img_tensor.norm(p=2)
        else:
            img_tensor = torch.zeros(query_vector.shape[-1]).to(device)

        if txt_vec:
```

```
            txt_tensor = torch.tensor(txt_vec,
dtype=torch.float32).to(device)
            txt_tensor = txt_tensor / txt_tensor.norm(p=2)
        else:
            txt_tensor = torch.zeros(query_vector.shape[-
1]).to(device)

        image_vectors.append(img_tensor)
        text_vectors.append(txt_tensor)
        valid_indices.append(i)

    # Формування батчу векторів у вигляді матриць
    image_tensor = torch.stack(image_vectors)
    text_tensor = torch.stack(text_vectors)

    print("3. Обчислення косинусного схожості...")
    # Обчислення косинусної схожості між вектором запиту і
векторами товарів
    sim_image = torch.matmul(query_vector,
image_tensor.T).squeeze()
    sim_text = torch.matmul(query_vector,
text_tensor.T).squeeze()

    # Комбінування схожості з урахуванням ваг
    combined_similarity = image_weight * sim_image +
text_weight * sim_text
    print("   Обчислення схожості завершено!")

    # Визначення індексів top_k найкращих за схожістю
товарів
    top_indices =
combined_similarity.topk(top_k).indices.cpu().tolist()

    top_products = []
    print("4. Виведення результатів...")
    for i in top_indices:
```

```
        product = products[valid_indices[i]]

        # Обчислення відсотка схожості
        similarity_percentage =
combined_similarity[i].item() * 100
        product["_id"] = str(product["_id"])  # Конвертація
ідентифікатора в рядок
        product["similarity_percentage"] =
round(similarity_percentage, 2)    # Додавання поля з
відсотком схожості

        print(f"      Товар: {product.get('title', 'Без
назви')},      ID:      {product['_id']},      Схожість:
{similarity_percentage:.2f}%")
        top_products.append(product)

    print("5. Пошук завершено!")
    return top_products
```

ДОДАТОК Д

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	93 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробила		Шумілова О.С.			Інтернет-магазин із чат-ботом на основі машинного навчання	Шифр групи	Код напр./спец.
Перевірив		Погурська М.М.				ІТШІз-21-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					