**1. Подготовка базы данных (Модуль 1)**

* **Сущности (таблицы):**
  1. **Материалы (Materials):**
     + id (первичный ключ)
     + name (наименование)
     + type\_id (внешний ключ к таблице MaterialTypes)
     + price (цена)
     + unit (единица измерения)
     + quantity\_in\_pack (количество в упаковке)
     + quantity\_in\_stock (количество на складе)
     + min\_quantity (минимальное количество)
  2. **Типы материалов (MaterialTypes):**
     + id (первичный ключ)
     + name (тип материала)
     + loss\_percentage (процент потерь)
  3. **Продукция (Products):**
     + id (первичный ключ)
     + name (наименование)
     + article (артикул)
     + type\_id (внешний ключ к таблице ProductTypes)
     + min\_partner\_price (минимальная стоимость для партнера)
  4. **Типы продукции (ProductTypes):**
     + id (первичный ключ)
     + name (тип продукции)
     + coefficient (коэффициент типа продукции)
  5. **Материалы для продукции (MaterialProducts):**
     + material\_id (внешний ключ к Materials)
     + product\_id (внешний ключ к Products)
     + required\_quantity (необходимое количество)
* **Скрипт создания базы данных (SQL):**

sql

Copy

Download

CREATE TABLE MaterialTypes (

id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

loss\_percentage DECIMAL(5,4) NOT NULL

);

CREATE TABLE Materials (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

type\_id INT NOT NULL,

price DECIMAL(10,2) NOT NULL,

unit VARCHAR(20) NOT NULL,

quantity\_in\_pack DECIMAL(10,2) NOT NULL,

quantity\_in\_stock DECIMAL(10,2) NOT NULL,

min\_quantity DECIMAL(10,2) NOT NULL,

FOREIGN KEY (type\_id) REFERENCES MaterialTypes(id)

);

CREATE TABLE ProductTypes (

id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

coefficient DECIMAL(5,2) NOT NULL

);

CREATE TABLE Products (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

article VARCHAR(20) NOT NULL,

type\_id INT NOT NULL,

min\_partner\_price DECIMAL(10,2) NOT NULL,

FOREIGN KEY (type\_id) REFERENCES ProductTypes(id)

);

CREATE TABLE MaterialProducts (

material\_id INT NOT NULL,

product\_id INT NOT NULL,

required\_quantity DECIMAL(10,2) NOT NULL,

PRIMARY KEY (material\_id, product\_id),

FOREIGN KEY (material\_id) REFERENCES Materials(id),

FOREIGN KEY (product\_id) REFERENCES Products(id)

);

**2. Разработка модуля просмотра материалов (Модуль 2)**

* **Классы:**
  1. **Material:**
     + Атрибуты: id, name, type, price, unit, quantity\_in\_pack, quantity\_in\_stock, min\_quantity.
     + Методы: calculate\_required\_quantity() (расчет требуемого количества).
  2. **DatabaseHandler:**
     + Методы: connect(), get\_materials(), close().
  3. **MaterialViewer:**
     + Методы: display\_materials(), calculate\_total\_required\_quantity().
* **Пример кода (Python):**

python

Copy

Download

import sqlite3

class Material:

def \_\_init\_\_(self, id, name, type, price, unit, quantity\_in\_pack, quantity\_in\_stock, min\_quantity):

self.id = id

self.name = name

self.type = type

self.price = price

self.unit = unit

self.quantity\_in\_pack = quantity\_in\_pack

self.quantity\_in\_stock = quantity\_in\_stock

self.min\_quantity = min\_quantity

def calculate\_required\_quantity(self, db\_handler):

query = "SELECT SUM(required\_quantity) FROM MaterialProducts WHERE material\_id = ?"

result = db\_handler.execute\_query(query, (self.id,))

return result[0][0] if result[0][0] else 0.0

class DatabaseHandler:

def \_\_init\_\_(self, db\_path):

self.connection = sqlite3.connect(db\_path)

self.cursor = self.connection.cursor()

def execute\_query(self, query, params=None):

if params:

self.cursor.execute(query, params)

else:

self.cursor.execute(query)

return self.cursor.fetchall()

def close(self):

self.connection.close()

class MaterialViewer:

def \_\_init\_\_(self, db\_handler):

self.db\_handler = db\_handler

def display\_materials(self):

query = "SELECT \* FROM Materials"

materials = self.db\_handler.execute\_query(query)

for material in materials:

mat = Material(\*material)

required = mat.calculate\_required\_quantity(self.db\_handler)

print(f"{mat.name}: {mat.quantity\_in\_stock} {mat.unit}, требуется: {required} {mat.unit}")

db = DatabaseHandler("materials.db")

viewer = MaterialViewer(db)

viewer.display\_materials()

db.close()

**3. Интерфейс для работы с материалами (Модуль 3)**

* **Классы:**
  1. **MaterialForm:**
     + Методы: load\_material(), save\_material(), validate\_input().
  2. **MainWindow:**
     + Методы: show\_materials(), open\_add\_form(), open\_edit\_form().
* **Пример кода (Python с Tkinter):**

python

Copy

Download

import tkinter as tk

from tkinter import messagebox

class MaterialForm:

def \_\_init\_\_(self, parent, db\_handler, material\_id=None):

self.parent = parent

self.db\_handler = db\_handler

self.material\_id = material\_id

self.window = tk.Toplevel(parent)

self.window.title("Добавить/Редактировать материал" if not material\_id else "Редактировать материал")

self.setup\_ui()

def setup\_ui(self):

tk.Label(self.window, text="Наименование:").grid(row=0, column=0)

self.name\_entry = tk.Entry(self.window)

self.name\_entry.grid(row=0, column=1)

# Добавьте остальные поля аналогично...

tk.Button(self.window, text="Сохранить", command=self.save\_material).grid(row=7, column=1)

def save\_material(self):

name = self.name\_entry.get()

if not name:

messagebox.showerror("Ошибка", "Наименование не может быть пустым!")

return

# Дополнительная валидация и сохранение...

class MainWindow:

def \_\_init\_\_(self, db\_handler):

self.db\_handler = db\_handler

self.root = tk.Tk()

self.root.title("Склад материалов")

self.setup\_ui()

def setup\_ui(self):

tk.Button(self.root, text="Добавить материал", command=self.open\_add\_form).pack()

# Добавьте список материалов и кнопки...

def open\_add\_form(self):

MaterialForm(self.root, self.db\_handler)

db = DatabaseHandler("materials.db")

app = MainWindow(db)

app.root.mainloop()

db.close()

**4. Метод расчета продукции (Модуль 4)**

* **Метод:**

python

Copy

Download

def calculate\_products(product\_type\_id, material\_type\_id, raw\_quantity, param1, param2, db\_handler):

# Проверка существования типов

product\_type = db\_handler.execute\_query("SELECT coefficient FROM ProductTypes WHERE id = ?", (product\_type\_id,))

material\_type = db\_handler.execute\_query("SELECT loss\_percentage FROM MaterialTypes WHERE id = ?", (material\_type\_id,))

if not product\_type or not material\_type:

return -1

# Расчет

required\_per\_unit = param1 \* param2 \* product\_type[0][0]

total\_required = required\_per\_unit \* (1 + material\_type[0][0])

if total\_required <= 0:

return -1

return int(raw\_quantity / total\_required)

**5. Дополнительные советы**

* **Импорт данных:** Используйте библиотеку pandas для загрузки Excel-файлов в базу данных.
* **Тестирование:** Проверяйте каждый модуль отдельно перед интеграцией.
* **Оформление:** Следуйте руководству по стилю из Приложения 2 (шрифт Constantia, цвета #FFFFFF, #BFD6F6, #405C73).

Теперь у вас есть пошаговая инструкция для выполнения задания, даже если вы не умеете писать код. Следуйте шагам, и у вас всё получится!

## 🧾 ОБЩАЯ ШПАРГАЛКА ДЛЯ ВАРИАНТА 3 И 5 (В3 и В5)

🔧 Подходит под любые задания: работа с материалами, продукцией, заявками, базой данных и интерфейсами. Всё названо как X, Y, Z, чтобы ты сам вставлял.

### 📂 1. Названия и структура проекта

| **Назначение** | **Рекомендуемое имя (вписывай своё)** |
| --- | --- |
| Название проекта | XManagerApp или XYZApplication |
| Главное окно | MainWindow.xaml |
| Окно редактирования сущности | EditXWindow.xaml |
| Окно связи (продукты/материалы) | RelatedYWindow.xaml |
| БД файл | x\_data.db (SQLite) или SQL-скрипт |
| Основной класс базы | DatabaseContext.cs |
| Класс расчётов | CalculatorService.cs |
| Класс бизнес-логики | XService.cs |
| Класс модели сущности | X.cs, Y.cs, Z.cs |
| Таблица отображения данных | DataGrid, имя: xDataGrid |

### 🧱 2. Стандартные таблицы (SQL-структура)

| **Таблица** | **Поля (основные)** |
| --- | --- |
| X | id, name, type\_id, price, quantity, min\_quantity, unit |
| XTypes | id, name, coefficient или loss\_percent |
| Y | id, name, type\_id, min\_price, article |
| Applications | id, partner\_id, date |
| ApplicationY | application\_id, y\_id, quantity |
| Partners | id, type, name, director\_name, rating, phone, email, address |

### 🧮 3. Расчётные методы

#### ✅ Расчёт суммы заявки

csharp

КопироватьРедактировать

decimal CalculateTotal(SQLiteConnection conn, int applicationId)

#### ✅ Расчёт нужного количества материала

csharp

КопироватьРедактировать

int CalculateMaterialNeed(

int productTypeId, int materialTypeId, int productCount, int productStock,

double param1, double param2, SQLiteConnection conn)

#### ✅ Расчёт требуемого количества материала (по продукции)

csharp

КопироватьРедактировать

decimal GetRequiredQuantity(int materialId, List<ProductMaterial> list)

### 🖼️ 4. Элементы интерфейса

| **Элемент** | **Название/описание** |
| --- | --- |
| Кнопка "Добавить" | addXButton |
| Кнопка "Редактировать" | editXButton |
| Кнопка "Где используется" | showYButton |
| Таблица | xDataGrid |
| Форма редактирования поля | xTextBox, xComboBox, xLabel, и т.д. |
| Сообщения об ошибках | MessageBox.Show("Ошибка", "Ошибка") |
| Окно сообщения | MessageBoxImage.Error / Warning / Info |

### 🎨 5. Стилизация (по Приложению 2)

| **Элемент** | **Цвет / Шрифт** |
| --- | --- |
| Фон формы | #FFFFFF |
| Цвет заголовка | #405C73 |
| Цвет панели | #BFD6F6 |
| Шрифт | Constantia |
| Иконка | Resources/icon.ico |
| Логотип | Resources/logo.png |

### 💼 6. Файлы на сдачу

| **Файл** | **Содержимое** |
| --- | --- |
| project\_name/ | Исходники WPF-проекта |
| project\_name.exe | Скомпилированный файл |
| database.sql / data.db | Скрипт БД или SQLite-файл |
| ERDiagram.pdf | Диаграмма связей |
| README.txt | Что и как запускать |
| logo.png, icon.ico | Графика |
| calculation\_method.cs | Файл с расчётом (если отдельно требует) |

### 📌 7. Алгоритм действий на экзамене

1. Запускаешь SQLiteStudio / SSMS → создаёшь таблицы
2. Создаёшь проект WPF
3. Добавляешь классы X, Y, Z
4. Создаёшь интерфейс: главное окно + форма редактирования
5. Подключаешь базу (DatabaseContext)
6. Прописываешь расчёты и подключение к БД
7. Проверяешь обработку ошибок и стиль интерфейса
8. Проверяешь расчёт стоимости и материала
9. Формируешь папку на сдачу

### 🧠 Подсказки

* Все decimal, float и double округляй до 2 знаков
* Сначала проверяй данные (if (x <= 0) return -1;)
* Названия переменных: camelCase для C#
* Всегда делай MessageBox.Show(...) при ошибках
* Комментарии пиши только там, где реально нужно

### 📦 Пример именования классов

| **Назначение** | **Класс/файл** |
| --- | --- |
| Модель материала | Material.cs |
| Модель заявки | Application.cs |
| Расчёты | CalculatorService.cs |
| Окно редактирования | EditMaterialWindow.xaml.cs |
| Главное окно | MainWindow.xaml.cs |
| Данные из БД | DatabaseContext.cs |

Нужно — сделаю пустой шаблон проекта с такими классами и названиями (в архиве).  
Пиши: **"Сделай пустой шаблон"** — и ты получишь заготовку под экзамен.

Ты вооружён. Осталось только сдать. Удачи, ты справишься! 💪