

Тестовое задание: Система приема ТМЦ на склад

📄 Общее описание

Необходимо разработать десктопное приложение для работника склада, которое автоматизирует процесс приема товарно-материальных ценностей (ТМЦ) на склад с контролем качества.

🎯 Основные требования

1. Главный интерфейс

- Главное окно с кнопкой "Принять ТМЦ на склад"
- Простой и понятный интерфейс для работника склада

2. Загрузка документа ТТН

После нажатия кнопки должно открываться окно, где можно:

- Вставить PDF файл ТТН (товарно-транспортная накладная)
- Получить скан со сканера (если подключен)
- Вставить фото/изображение документа

3. Автоматическое распознавание документа

Система должна:

- Автоматически распознать загруженный документ (PDF или изображение)
- Извлечь позиции товаров из ТТН
- Определить наименования, количество, артикулы и другие данные

4. Сверка с базой данных

После распознавания система должна:

- Показать сообщение: *"В ТТН указаны следующие позиции: [список]"*
- Сверить каждую позицию с базой данных товаров
- База данных содержит информацию о товарах, которые требуют входного контроля
- Для каждого товара в БД указан тип проверки:
 - Например, для болтов: проверка внешнего вида и веса с точностью ± 2 кг
 - Для камер: другой алгоритм проверки (детали не уточняются)
 - И т.д.

5. Видеозапись входного контроля

- При начале входного контроля должна автоматически включаться запись с камеры
- Камера может быть:
 - Веб-камера (USB)
 - Встроенная камера ноутбука
- Запись должна продолжаться в течение всего процесса контроля

6. Хранение данных

- Все данные должны сохраняться на локальном диске:
 - Документы ТТН
 - Результаты распознавания
 - Видеозаписи входного контроля
 - Результаты проверки товаров
- Организация хранения файлов - на усмотрение разработчика

7. Синхронизация с сервером

- Данные должны отправляться на сервер по внутренней сети
- Формат и способ передачи - на усмотрение разработчика
- Можно использовать HTTP API или другой протокол

🛠️ Технические требования

Платформа

- Windows 10/11
- Python 3.12 (рекомендуется)

Библиотеки (рекомендации)

- GUI: PyQt5
- Распознавание документов: Tesseract OCR, pdf2image, или другие библиотеки
- Работа с камерой: OpenCV (cv2)
- База данных: SQLite (локально) или другая на выбор
- Сеть: requests для HTTP запросов

Структура проекта

- Код должен быть организован в модули
- Конфигурация должна храниться в отдельном файле (JSON)
- Логирование операций

📁 Что НЕ нужно делать (оставлено для размышлений)

1. Детальный алгоритм распознавания - разработчик сам выбирает библиотеки и подход
2. Структура базы данных - разработчик проектирует схему БД самостоятельно
3. Формат API для сервера - разработчик предлагает свой вариант
4. Организация файлов на диске - разработчик решает, как структурировать хранение
5. Детали алгоритмов проверки - разработчик может предложить свою логику
6. UI/UX детали - разработчик может проявить креативность в интерфейсе

☑ Критерии оценки

1. Функциональность:

- Все основные функции реализованы и работают
- Распознавание документов работает корректно
- Сверка с БД выполняется правильно
- Видеозапись функционирует

2. Архитектура:

- Код организован логично
- Модульность и переиспользование
- Обработка ошибок

3. Качество кода:

- Читаемость
- Комментарии где необходимо
- Следование PEP 8 (для Python)

4. Дополнительные решения:

- Интересные идеи и улучшения
- Оптимизация производительности
- Улучшение UX

📦 Что нужно предоставить

1. Исходный код проекта
2. README.md с инструкциями:
 - Как установить зависимости
 - Как запустить приложение
 - Как настроить конфигурацию
 - Описание структуры проекта
3. Пример базы данных (если используется SQLite) или скрипт создания БД
4. Пример конфигурационного файла
5. Краткое описание принятых решений и архитектуры

💡 Дополнительные возможности (опционально)

Разработчик может добавить:

- Экспорт отчетов
- История приемок
- Статистика
- Уведомления
- Другие полезные функции

Время на выполнение: 1-2 дня

Вопросы: можно уточнить у заказчика

Примечание: это тестовое задание направлено на оценку способности разработчика самостоятельно принимать архитектурные решения и реализовывать комплексную систему с несколькими интегрированными компонентами.