# Техническое задание (ТЗ) Система тестирования командных навыков (на Python)

Версия: 1.0  
Дата: 30.09.2025  
Заказчик: Ким С.В.  
Исполнитель: \_\_\_\_\_\_\_\_

Имя бота: psychtestteambot

## 1. Цели и задачи

Создать кроссплатформенную систему тестирования на командные навыки (Python), позволяющую проходить несколько опросников, автоматически рассчитывать шкалы, формировать индивидуальный психологический портрет и экспортировать результаты.

Ключевые задачи:

• Реализовать опросники: Адизес (PAEI), HEXACO, DISC (минимум).

• Обеспечить ввод ответов через CLI и/или веб-интерфейс (Streamlit).

• Рассчитывать шкальные баллы с учётом обратного кодирования и нормирования.

• Формировать отчёт с интерпретациями по образцу ранее полученного портрета.

• Экспортировать результаты: PDF/DOCX, CSV/Excel, JSON.

• Хранить анонимизированные данные локально (SQLite/CSV).

## 2. Область применения и ограничения

Инструмент предназначен для обучения и самооценки. Не заменяет клиническую диагностику.

## 3. Состав тестов и методики

3.1. Адизес (PAEI): выбор из 4 утверждений; суммирование баллов по ролям P/A/E/I.

3.2. HEXACO (краткая версия): шкалы H, E, X, A, C, O (1–5), обратное кодирование.

3.3. DISC (краткая версия): суммы по D/I/S/C, построение профиля.

## 4. Требования к данным

Источник: встроенные CSV с банком вопросов и ключами. Форматы: SQLite/CSV.

|  |  |  |
| --- | --- | --- |
| Сущность | Поля | Комментарий |
| tests | test\_id, name, version, lang | Регистрация тестов |
| items | item\_id, test\_id, text, key, reverse (0/1), lang | Вопросы и ключи |
| responses | session\_id, user\_id?, test\_id, item\_id, answer, ts | Ответы |
| scores | session\_id, test\_id, scale, raw, norm, ts | Итоговые баллы |

## 5. Пользовательские сценарии и интерфейс

CLI: пошаговый опрос с валидацией ввода и сохранением прогресса.

Веб (Streamlit): выбор тестов, прогресс-бар, кнопка экспорта.

## 6. Алгоритмы расчёта и интерпретации

Адизес: сумма пунктов по P/A/E/I; текст по диапазонам.

HEXACO: суммы по шкалам; реверс отрицательных пунктов; процентили при наличии норм.

DISC: суммы по D/I/S/C; описание профиля.

Пропуски: предупреждение и стратегия pairwise deletion.

## 7. Архитектура и API

Стек: Python 3.10+, pandas, numpy, pydantic, sqlite3, python-docx, reportlab (опц.), streamlit.

• data: загрузка/валидация банков вопросов; репозитории SQLite/CSV

• domain: модели Item, Test, Response, Score; валидаторы

• services: прохождение тестов, скоринг, интерпретации, экспорт

• ui: CLI/Streamlit

— load\_bank(path: str) -> TestBank

— run\_session(tests: list[str], ui: Literal["cli","web"]) -> SessionResult

— score(session) -> list[Score]

— render\_report(scores, template="docx") -> Path

— export\_data(session, scores, fmt="csv|json|xlsx") -> Path

## 8. Локализация

RU (по умолчанию), KZ, EN через поля lang и шаблоны отчёта.

## 9. Безопасность и приватность

Анонимный режим по умолчанию; опциональный user\_id (хеш). Локальное хранение.

## 10. Нефункциональные требования

Производительность: <1 с на расчёт 3 опросников. Покрытие тестами ключевой логики ≥80%.

## 11. Тест-кейсы (выдержка)

• PAEI → корректный профиль и текст отчёта.

• HEXACO с обратными пунктами → верное реверсирование.

• DISC с пропусками → предупреждение и дозаполнение.

• Экспорт DOCX/PDF/CSV/JSON → файлы создаются и открываются.

## 12. Критерии приёмки

• Совпадение расчётов с контрольными примерами на ≥95%.

• Отчёт по структуре соответствует образцу портрета.

• Доступны CLI и/или Streamlit интерфейсы.

## 13. План работ (MVP → v1.0)

Спринт 1: каркас проекта, PAEI, CLI.

Спринт 2: HEXACO, отчёт DOCX.

Спринт 3: DISC, экспорт данных, SQLite.

Спринт 4: Streamlit, локализация, тестирование.

## 14. Материалы и образцы

Образец интерпретации: "Психологический портрет КимСВ.docx".

## 15. Шаблон отчёта (DOCX): структура

Разделы отчёта:

• Титул: ФИО/код участника, дата, список пройденных методик.

• Сводная таблица профиля: шкалы, сырые и нормированные баллы, процентиль.

• Графики (по желанию): радары/столбцы для ключевых шкал.

• Интерпретации: по каждой шкале (низкий/средний/высокий) + рекомендации.

• Итоговый профиль и краткие выводы.

## 16. Банки вопросов: форматы CSV

Общие поля: item\_id, test\_id, text, scale, key, reverse, lang

Пример строк для PAEI:

item\_id,test\_id,text,scale,key,reverse,lang  
1,PAEI,"Мне проще доводить дела до результата",P,1,0,ru  
2,PAEI,"Я люблю порядок и регламенты",A,1,0,ru  
3,PAEI,"Мне нравится запускать новое",E,1,0,ru  
4,PAEI,"Я поддерживаю гармонию в команде",I,1,0,ru

Пример строк для HEXACO (обратные пункты reverse=1):

item\_id,test\_id,text,scale,key,reverse,lang  
101,HEXACO,"Честно признаю ошибки",H,1,0,ru  
102,HEXACO,"Иногда преувеличиваю свои заслуги",H,1,1,ru

## 17. Нормы и процентили: формат CSV

norms.csv: test\_id,scale,sex,age\_min,age\_max,mean,sd,p10,p25,p50,p75,p90

Использование: norm = (raw-mean)/sd; процентиль — по таблице pXX.

## 18. Каркас проекта: структура

psytest/  
├─ pyproject.toml  
├─ README.md  
├─ data/  
│ ├─ bank/  
│ │ ├─ paei\_items.csv  
│ │ ├─ disc\_items.csv  
│ │ ├─ hexaco\_items.csv  
│ │ └─ norms\_hexaco.csv  
├─ templates/  
│ └─ report\_template.docx  
├─ src/psytest/  
│ ├─ \_\_init\_\_.py  
│ ├─ bank.py  
│ ├─ scoring.py  
│ ├─ report.py  
│ ├─ storage.py  
│ ├─ cli\_main.py  
│ └─ web\_app.py (Streamlit)

## 19. Доп. требования к приватности

Согласие: явное подтверждение перед началом; анонимизация по hash(user\_id+salt).

## 20. Потоки в Streamlit

Экран 1: выбор тестов → Экран 2: ответы с прогрессом → Экран 3: итоги, экспорт.

02.10.2025:

Сценарий использования. Корпоративные клиенты платят за фиксированное количество тестирований и дают адрес бота и пароль своим сотрудникам, которых нужно оттестировать. Сотрудники получают ссылку на телефон, проходят тестирование на телефоне и получают отчет на свой телефон в pdf формате. Корпоративный клиент также получает эти отчеты.   
Вопрос: как организовать временное хранилище pdf документов, формируемых ботом? Куда корпоративный клиент или его представитель может получать отчеты? в гугл аккаунт?

Корпоративный клиент или представитель корпоративного клиента регистрируется и оплачивает энное количество тестирований. Придумывает пароль для доступа к тестированию для сотрудников. Пароль служит пропуском и одновременно идентификатором, какому к корпоративному клиенту относится тестируемый сотрудник.

Необходимо предусмотреть заглушку, вместо которой потом будет модуль интернет-оплаты (веб интерфейс).

Нужно предусмотреть отдельный пароль для суперпользователя (для меня), чтобы раздавать нужным людям для тестирования. По всей видимости, необходимо каждый раз менять пароль. Видимо нужен какой-то пункт управления аккаунтами.

Представим ситуацию, когда одновременно несколько человек 3 - 5 начинают проходить тестирование, тогда, видимо, потребуется асинхронной ввод данных с отслеживанием поступающих ответов по идентификатору в тестируемого?

Ты можешь приблизительно подсчитать, какую нагрузку на оперативную память может создать такое режим, когда 3-5 человек одновременно проходят тестирование?