**1. Краткое описание проекта**

Название мобильного приложения: Secure Vault

Описание: Персональное хранилище документов со сквозным шифрованием

Цель: Разработать мобильное приложение для хранения документов пользователя

Аналоги: Google Drive, OneDrive, Proton Drive

Дата сдачи — 08.12.2025

**2. Основные требования**

**2.1. Пользовательские роли**

* Читатель – может только просматривать документы
* Редактор – может добавлять и удалять документы
* Администратор – управляет правами доступа

**2.2. Базовый функционал (MVP)**

1. Регистрация и аутентификация пользователей
   * Пользователь создаёт аккаунт и получает уникальную пару ключей (public/private).
   * Приватный ключ хранится на сервере в зашифрованном виде (паролем пользователя).
2. Загрузка файлов в хранилище
   * Файл перед загрузкой шифруется на клиенте с помощью случайного симметричного ключа (AES).
   * Сервер получает только ciphertext и зашифрованный session\_key.
3. Хранение файлов
   * Файлы хранятся в объектном хранилище (например, AWS S3 или аналоги).
   * В базе данных хранятся только метаданные и зашифрованные ключи.
4. Организация файлов
   * Пользователь может создавать папки и перемещать файлы.
   * Названия файлов и папок тоже зашифрованы (недоступны серверу).
5. Загрузка и расшифровка файлов
   * Пользователь скачивает зашифрованный файл и session\_key\_encrypted.
   * На клиенте session\_key расшифровывается приватным ключом, после чего файл расшифровывается.
6. Удаление файлов
   * Пользователь может удалять файлы и их метаданные.
   * После удаления файл становится недоступным для всех участников.
7. Совместный доступ между пользователями системы
   * Владелец может расшифровать session\_key приватным ключом и зашифровать его публичным ключом другого пользователя.
   * В БД сохраняется session\_key\_encrypted для этого пользователя.
8. Поиск по имени и метаданным
   * Поиск по имени и метаданным возможен только после локальной расшифровки.
9. Проверка целостности файлов

* При загрузке файла вычисляется хэш (SHA-256) от оригинального содержимого.
* Хэш сохраняется в зашифрованном виде вместе с метаданными.
* При скачивании клиент сверяет хэш, чтобы убедиться, что файл не повреждён.

**2.3. Нефункциональные требования**

* Кроссплатформенность: iOS 13+/Android 8+.
* Время отклика UI: <200 ms на основные экраны (при наличии сети).
* Безопасность: HTTPS, шифрование данных, ограничение доступов.
* Конфиденциальность: соответствие общим требованиям (анонизированные логи), возможность удаления аккаунта.
* Локализация: RU

**3. Предложенный стек технологий**

* Frontend (мобильное): **React Native** — кроссплатформенно, быстрая разработка UI.
* Backend: **FastAPI** — позволяет гибко настраивать модель данных.
* База данных: **PostgreSQL**
* React Native позволяет быстро разрабатывать кроссплатформенный интерфейс. FastAPI дает возможность быстро поднять сервер и гибко настроить модель данных. PostgreSQL выдерживает большие нагрузки и обладает обширным функционалом.

**4. Инструменты разработки**

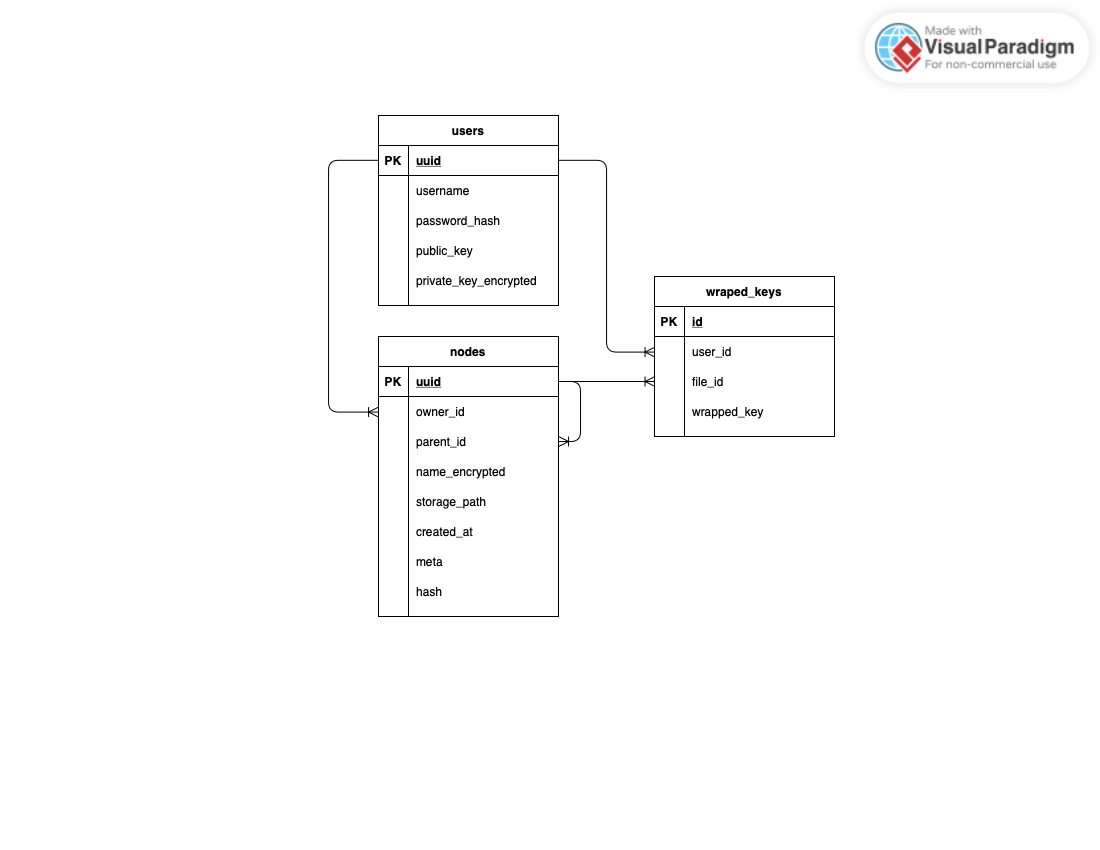
* IDE: VSCode.
* GitHub — исходники.
* Postman — отладка API.

**5. Архитектура и данные**

**5.1. Архитектурная схема**

Мобильное приложение (React Native) ↔ FastAPI backend и PostgreSQL; аутентификация через JWT токены.

**5.2. Структура БД**



**6. API / Endpoints (пример для собственного backend)**

* POST /api/auth/register
* POST /api/auth/login
* POST /api/files
* GET /api/files
* GET /api/files/{file\_id}
* DELETE /api/files/{file\_id}
* POST /api/files/{file\_id}/share
* GET /api/search

**7. UI/UX: основные экраны + требования**

1. Экран входа / регистрации — логин и пароль.
2. Мое хранилище — файлы и папки, меню действий: поделиться, удалить.
3. Общий доступ — просмотр выданных доступов, отзыв доступов.
4. Профиль — Имя пользователя и логин. Переход к настройкам.
5. Аккаунт и пароль — смена имени пользователя и пароля.

**8. Тестирование**

* Unit tests.
* Integration: минимальный сценарий «войти → загрузить файл → поделиться доступом».
* Регрессионное тестирование перед сдачей.
* Тестирование на реальных устройствах (Android & iOS).

**9. Релиз**

* Репозиторий: Git (feature-branch workflow).
* GitHub Actions: запускает lint, unit tests.
* Release: Google Play / RuStore / TestFlight.

**10. Безопасность и приватность**

* Все запросы по HTTPS.
* Шифрование файлов.
* Безопасное хранилище токенов.
* Возможность удаления аккаунта / данных.

**11. Роадмап / план по дням с конкретными датами**

**Sprint 0 — Подготовка / Discovery (16 — 21 сентября 2025) — 6 дней**

**Цели:** финальные требования, дизайн архитектуры, инфраструктура.

* 16–17.09: Сбор и уточнение требований, приоритизация фич (MVP).
* 18.09: Составление ТЗ, проектирование архитектуры БД и API.
* 19–20.09: Вайрфреймы основных экранов (загрузка/просмотр/поиск файлов).
* 21.09: Настройка репозитория.

**Доставка:** финализированные требования, архитектура БД/API, базовый репозиторий.

**Sprint 1 — Core: пользователи и файлы (22 сентября — 5 октября 2025) — 2 недели**

**Задачи:**

* Реализовать регистрацию/логин (email + пароль, хранение public/private ключей).
* CRUD для файлов: загрузка (с шифрованием на клиенте), просмотр списка, скачивание, удаление.
* Интеграция AWS S3 (или MinIO локально).
* Метаданные: название (зашифровано), размер, дата.
* Unit-тесты для Auth и файлового модуля.

**Доставка:** рабочая версия: можно загрузить, хранить и скачивать файлы с E2EE.

**Sprint 2 — Шаринг и структура (6 — 19 октября 2025) — 2 недели**

**Задачи:**

* Шаринг файлов между пользователями (перешифровка session\_key под public\_key получателя).
* Просмотр, кто имеет доступ к файлу.
* Создание иерархии папок (parent\_id).
* API для управления доступом.
* Интеграционные тесты «владелец → расшарил → получатель скачал».

**Доставка:** поддержка шаринга, базовые папки.

**Sprint 3 — Поиск, теги, UX (20 октября — 2 ноября 2025) — 2 недели**

**Задачи:**

* UI/UX улучшения: превью файлов (иконки/метаданные).
* Кеширование списка файлов для оффлайн-доступа.
* Начало документации (README, архитектура, как поднять локально).

**Доставка:** улучшенный UI, оффлайн-режим (частично).

**Sprint 4 —** Интеграция, безопасность, админка (3 — 16 ноября 2025) — 2 недели

Задачи:

* Полная интеграция модулей: Auth, Files, Sharing, Search.
* Безопасность: валидация токенов, шифрование приватных ключей.
* Простая админ-панель (веб): список пользователей, файлов, статистика.
* Настройка CI/CD для staging.
* Внутреннее UAT (тестирование с участниками команды).

**Доставка:** release-кандидат с рабочей админкой и защищённым API.

**Stabilization & UAT fixes (17 — 25 ноября 2025) — 9 дней**

**Задачи:**

* Исправление багов после тестирования.
* Оптимизация скорости загрузки/скачивания файлов.
* Полный регрессионный тест.
* Подготовка презентации (скриншоты, demo-видео, инструкции).
* Финализация документации (установка, API спецификация).

**Доставка:** финальные сборки (frontend + backend), документация, демо.

**Final polishing & submission (26 ноября — 8 декабря 2025) — 13 дней**

**Задачи:**

* Финальные баг-фиксы.
* Релизная версия фронтенда и бэкенда.
* Презентация и демонстрация проекта (3–5 минут).
* Передача проекта преподавателю: код, инструкции, учётные записи.

**Финиш:**сдача до 08.12.2025 включительно.

**12. Уникальные функции проекта**

* Возможность делиться целой папкой, а не отдельными файлами
* Двухфакторная аутентификация
* Автоматическая синхронизация с локальной папкой