Додаток В

Специфікація програмного забезпечення

**Software Requirements**

**Specification**

**for**

Software system for task management

Version 1.0 approved

Prepared by Velykotrav Violetta

NURE

20.05.2025

Історія версій

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Опис** | **Автор** | **Коментар** |
| 20.05.2025 | Версія 1.0 | Великотрав Віолетта | Створення документу та заповнення усіх позділів |

**ЗМІСТ**

[1 Вступ 4](#_Toc201790313)

[1.1 Огляд продукту 4](#_Toc201790314)

[1.2 Мета 4](#_Toc201790315)

[1.3 Межі 5](#_Toc201790316)

[1.4 Посилання 5](#_Toc201790317)

[1.5 Означення та абревіатури 6](#_Toc201790318)

[2 Загальний опис 7](#_Toc201790319)

[2.1 Перспективи продукту 7](#_Toc201790320)

[2.2 Функції продукту 7](#_Toc201790321)

[2.3 Характеристики користувачів 7](#_Toc201790322)

[2.4 Загальні обмеження 8](#_Toc201790323)

[2.5 Припущення й залежності 8](#_Toc201790324)

[3. Конкретні вимоги 9](#_Toc201790325)

[3.1 Інтерфейс користувача 9](#_Toc201790326)

[3.2 Апаратний інтерфейс 9](#_Toc201790327)

[3.3 Програмний інтерфейс 10](#_Toc201790328)

[3.5 Обмеження памʼяті 11](#_Toc201790329)

[3.6 Операції 11](#_Toc201790330)

[3.7 Властивості програмного продукту 11](#_Toc201790331)

[3.8 Вимоги до бази даних 12](#_Toc201790332)

[3.9 Інші вимоги 12](#_Toc201790333)

# 1 ВСТУП

## 1.1 Огляд продукту

Сучасні користувачі — як індивідуальні, так і командні — дедалі частіше шукають прості, адаптивні інструменти для керування особистими та робочими завданнями. Однак існуючі рішення або перевантажені інтерфейсом, або обмежені функціонально. Більшість з них вимагає постійного підключення до хмарних сервісів, що унеможливлює їх повноцінне використання в офлайн-умовах або без реєстрації.

Розробка програмного комплексу Self-Process Manager вирішує цю проблему шляхом створення системи, яка поєднує:

* автономність з можливістю локального зберігання даних;
* простий десктопний інтерфейс;
* кастомний синтаксис для швидкого введення задач;
* фізичні сповіщення через IoT-пристрій.

Основу рішення складає бекенд на FastAPI з локальною БД (SQLite), клієнтський застосунок на Flutter, та IoT-модуль на базі ESP32-C3 для виведення актуальних сповіщень.  
 Система підтримує REST API, автентифікацію через JWT, логування, фільтрацію задач, історію змін і функціонал групування.

## 1.2 Мета

Метою розробки є створення багатокомпонентного застосунку для управління завданнями, що дозволяє користувачам:

* зручно створювати, редагувати та переглядати задачі;
* групувати їх за категоріями, пріоритетами, термінами;
* отримувати фізичні сповіщення про термінові завдання через IoT-пристрій;
* працювати у гібридному або офлайн-режимі, із подальшою синхронізацією з сервером.

Основні цілі:

* реалізувати REST API для взаємодії з клієнтами;
* підтримати кастомний парсинг задач зі скороченого тексту;
* забезпечити простий десктопний інтерфейс (Flutter Desktop);
* реалізувати синхронізацію між локальними та серверними даними;
* створити IoT-модуль для фізичних сповіщень (OLED-дисплей, LED або звук);
* надати базову систему автентифікації користувачів;
* забезпечити мінімалістичний, інтуїтивний UI/UX.

## 1.3 Межі

**У межах реалізації:**

* обробка задач через REST API (створення, редагування, видалення, перегляд);
* підтримка текстового синтаксису для швидкого створення задач;
* локальне збереження даних у SQLite;
* синхронізація локальної БД з сервером;
* відправка актуальних задач на IoT-пристрій через запити до API;
* авторизація за допомогою JWT;
* використання Docker для розгортання бекенду;
* десктопна Flutter-програма для Windows, Linux, macOS.

**Поза межами реалізації:**

* мобільна версія клієнта (запланована у майбутньому);
* повноцінна система ролей або команд;
* інтеграція з календарями (Google Calendar, Outlook);
* голосовий ввід або AI-підказки;
* резервне копіювання та масштабування для великої кількості користувачів;
* багатомовність інтерфейсу;
* офіційна інтеграція з месенджерами (Telegram, Slack).

## 1.4 Посилання

У специфікації посилаємося на такі документи та ресурси:

* **FastAPI Documentation** – https://fastapi.tiangolo.com/
* **Flutter Documentation** – https://docs.flutter.dev/
* **ESP32-C3 Technical Reference Manual** – https://www.espressif.com/en/support/documents
* **Docker Docs** – https://docs.docker.com/
* **SQLite Documentation** – https://www.sqlite.org/docs.html

## 1.5 Означення та абревіатури

* **API** — Application Programming Interface
* **CRUD** — Create, Read, Update, Delete
* **ESP32** — Embedded Serial Processor (32-bit MCU від Espressif)
* **IoT** — Internet of Things
* **JWT** — JSON Web Token
* **REST** — Representational State Transfer
* **SQL** — Structured Query Language
* **UI/UX** — User Interface / User Experience
* **JSON** — JavaScript Object Notation

# 2 ЗАГАЛЬНИЙ ОПИС

## 2.1 Перспективи продукту

Self-Process Manager — це відповідь на сучасний запит користувачів на прості, автономні та гнучкі системи керування завданнями. Більшість існуючих таск-менеджерів або перевантажені функціональністю, або залежать від хмарної інфраструктури, що обмежує їх застосування в офлайн-режимі.

Система орієнтована на підтримку особистої продуктивності, командної взаємодії та фізичних сповіщень. Вона поєднує в собі:

* Flutter-додаток для десктопу з інтерфейсом Inbox + Kanban;
* RESTful API на базі FastAPI;
* IoT-модуль на ESP32 для виводу сповіщень через дисплей або світлодіод.

Архітектура системи дозволяє легко масштабуватись, додавати нові канали сповіщень або інтегрувати з іншими платформами. У майбутньому можливе створення мобільного застосунку, багатокористувацької взаємодії, розширеної статистики, аналітики задач і повноцінної синхронізації з хмарними сервісами.

## 2.2 Функції продукту

Система реалізує такі основні функції:

* **Управління задачами, с**творення задач через UI або текстовий синтаксис (@, #, ++, ! тощо), групування задач за типами, важливістю, дедлайнами, відстеження виконання, редагування, видалення;
* **Синхронізація, з**берігання задач локально (SQLite), виявлення конфліктів при синхронізації з сервером, пріоритет останніх змін при об’єднанні;
* **IoT-функціонал, а саме н**адсилання актуальних задач на пристрій ESP32-C3, відображення інформації на OLED-дисплеї, фізичні сповіщення (світло, звук);
* **API-функції а саме R**EST API для доступу до всіх функцій системи, авторизація через JWT, вебхуки або планування оновлень даних.

## 2.3 Характеристики користувачів

**Основний користувач — індивідуальний користувач (self-manager):**

* Створює й виконує задачі;
* Отримує сповіщення через IoT;
* Може працювати повністю офлайн з подальшою синхронізацією.
* **Адміністратор (опціонально):**
* Керує IoT-пристроями (призначення, діагностика);

У перспективі можуть бути додані інші ролі (аналітик, керівник команди).

## 2.4 Загальні обмеження

* **Фокус на MVP:** реалізовано тільки ключові функції: задачі, IoT, синхронізація, без ролей.
* **Обмежена кількість користувачів:** система розрахована на одного користувача на пристрій.
* **Обмеження IoT:** ESP32 працює з GET-запитами; зворотній зв’язок поки не реалізований.
* **Без мобільного клієнта:** лише десктопний клієнт (Windows, Linux, macOS).
* **Немає повноцінного GUI для адміністратора** — лише CLI/API.

## 2.5 Припущення й залежності

**Інфраструктурні припущення:**

* Наявність стабільного Wi-Fi з доступом до API.
* Сервер запускається в Docker-контейнері.

**Користувацькі припущення:**

* Користувач має базові технічні навички для запуску застосунку.
* IoT-пристрій попередньо налаштований та під’єднаний.

**Технічні залежності:**

* FastAPI + Uvicorn + SQLite + SQLModel.
* Flutter (з hooks\_riverpod, dio).
* ESP32 з MicroPython або Arduino C++.
* JSON як стандарт обміну даними.

# 3. КОНКРЕТНІ ВИМОГИ

## 3.1 Інтерфейс користувача

Кінцевий інтерфейс користувача реалізовано у вигляді десктопного застосунку на Flutter, що взаємодіє з REST API. Додаток має простий, мінімалістичний дизайн і підтримує:

* **Inbox-подібний вигляд** (всі задачі в одному вхідному потоці),
* **Kanban-представлення** для перегляду задач за статусами,
* **Парсинг задач** із короткого текстового вводу (@, #, ++, !, //),
* **Фільтрацію задач** за групами, важливістю, дедлайнами,
* **Огляд історії змін**.

Інтерфейс очікує від сервера структуровані відповіді у форматі JSON та використовує JWT для авторизації користувачів.

## 3.2 Апаратний інтерфейс

IoT-модуль (ESP32-C3) взаємодіє із сервером через HTTP-запити. Пристрій отримує список найбільш актуальних задач користувача у форматі JSON для виведення їх на OLED-дисплей або для сповіщення світлодіодом/звуком. Приклад формату відповіді:

**{**

**"tasks": [**

**{**

**"id": "1",**

**"title": "⚠️ До 18:00 здати звіт",**

**"priority": "high"**

**}**

**]**

**}**

**ESP32 надсилає або отримує дані за попередньо узгодженими маршрутами** /iot/notify**, аутентифікуючись за device token або через авторизованого користувача.**

## **3.3 Програмний інтерфейс**

**Система реалізує REST API на FastAPI. Основні характеристики:**

* Формат обміну: **JSON**
* Методика: **GET**, **POST**, **PUT**, **DELETE**
* Авторизація: **JWT**
* Маршрути: згруповані за ресурсами (/tasks, /users, /sync, /iot)
* Валідація: через Pydantic
* Документація: автоматично генерується за OpenAPI на /docs

**3.4 Комунікаційна модель**

**Модель побудована за архітектурою "клієнт — сервер", де Flutter-додаток та IoT-пристрій є клієнтами. Уся взаємодія відбувається через HTTP(S) запити з авторизацією через JWT.**

Після логіну користувач отримує токен і використовує його в усіх запитах. IoT-пристрій періодично запитує найважливіші задачі та показує їх фізично.

## **3.5 Обмеження памʼяті**

* Flutter-додаток зберігає задачі локально в SQLite (ORM drift/hive);
* Максимальна довжина тексту задачі — 512 символів;
* Обмежено до 100 задач на одного користувача в MVP;
* IoT-пристрій обмежений у відображенні — максимум 3 задачі виводяться одночасно.

## 3.6 Операції

Основні REST API операції:

| Операція | Метод | URL | Доступно |
| --- | --- | --- | --- |
| Авторизація | POST | /auth/login | всім |
| Отримати задачі | GET | /tasks/ | користувач |
| Створити задачу | POST | /tasks/ | користувач |
| Оновити задачу | PUT | /tasks/{id} | користувач |
| Видалити задачу | DELETE | /tasks/{id} | користувач |
| Синхронізація з IoT | GET | /iot/notify | пристрій |
| Отримати історію змін | GET | /tasks/history | користувач |

## 3.7 Властивості програмного продукту

**Функціональність**:

* Створення та парсинг задач;
* Синхронізація локальних і серверних задач;
* IoT-сповіщення;
* Авторизація та захист даних.

**Надійність**:

* Перевірка коректності даних;
* Обробка винятків (наприклад, дублікати, порожні задачі);
* Автоматичне логування.

**Безпека**:

* Хешування паролів (bcrypt);
* JWT з терміном дії;
* Middleware для перевірки доступу.

**Супроводжуваність**:

* Розділення на модулі: моделі, сервіси, роутери;
* Документація OpenAPI;
* Покриття тестами (pytest).

**Масштабованість**:

* Можливість переходу з SQLite на PostgreSQL;
* Підтримка асинхронних запитів через FastAPI;
* IoT-модуль з відкритим API.

## 3.8 Вимоги до бази даних

Система використовує SQLite, з можливістю переходу на PostgreSQL. Основні таблиці:

* users — користувачі;
* tasks — задачі;
* iot\_devices — пристрої (опціонально).

БД забезпечує:

* Цілісність даних (зовнішні ключі),
* Унікальність задач користувача,
* Швидкий пошук по тегах/групах.

## 3.9 Інші вимоги

* Вся документація API доступна за /docs;
* Система повинна запускатися у Docker;
* Flutter-клієнт має працювати в офлайн-режимі з подальшою синхронізацією;
* Код повинен бути покритий unit-тестами мінімум на 70%.