**Software Requirements**

**Specification**

**for**

HealthyHelper

Version 1.0 approved

Prepared by Tolstik Oleksii

NURE

15.05.2025

Історія версій

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Опис** | **Автор** | **Коментар** |
| 15.05.2025 | Версія 1.0 | Толстік Олексій | Створення документу |

**Зміст**

[1 Вступ 4](#_Toc200615982)

[1.1 Огляд продукту 4](#_Toc200615983)

[1.2 Мета 5](#_Toc200615984)

[1.3 Межі 6](#_Toc200615985)

[1.4 Посилання 6](#_Toc200615986)

[1.5 Означення та абревіатури 7](#_Toc200615987)

[2 Загальний опис 8](#_Toc200615988)

[2.1 Перспективи продукту 8](#_Toc200615989)

[2.2 Функції продукту 9](#_Toc200615990)

[2.3 Характеристики користувачів 11](#_Toc200615991)

[2.4 Загальні обмеження 12](#_Toc200615992)

[2.5 Припущення й залежності 13](#_Toc200615993)

[3 Конкретні вимоги 14](#_Toc200615994)

[3.1 Вимоги до зовнішніх інтерфейсів 14](#_Toc200615995)

[3.1.1 Інтерфейс користувача 14](#_Toc200615996)

[3.1.2 Апаратний інтерфейс 15](#_Toc200615997)

[3.1.3 Програмний інтерфейс 17](#_Toc200615998)

[3.1.4 Комунікаційний прототип 18](#_Toc200615999)

[3.1.5 Обмеження пам'яті 19](#_Toc200616000)

[3.1.6 Операції 20](#_Toc200616001)

[3.1.7 Функції продукту 21](#_Toc200616002)

[3.1.8 Припущення й залежності 23](#_Toc200616003)

[3.2. Властивості програмного продукту 24](#_Toc200616004)

[3.3. Атрибути програмного коду 25](#_Toc200616005)

[3.3.1 Надійність 25](#_Toc200616006)

[3.3.2 Доступність 26](#_Toc200616007)

[3.3.3 Безпека 27](#_Toc200616008)

[3.3.4 Супроводжуваність 28](#_Toc200616009)

[3.3.5 Переносимість 28](#_Toc200616010)

[3.3.6 Продуктивність 29](#_Toc200616011)

[3.4 Вимоги бази даних 30](#_Toc200616012)

[3.5 Інші вимоги 30](#_Toc200616013)

[4 Додаткові матеріали 32](#_Toc200616014)

# 1 Вступ

## 1.1 Огляд продукту

За останні роки спостерігається стрімке зростання кількості пацієнтів, що потребують медикаментозного лікування. Це зумовлено як поширенням хронічних захворювань, так і наслідками пандемії COVID-19, що спричинила довготривалі ускладнення у багатьох пацієнтів. Додатковим чинником стала війна, що призвело до фізичних і психологічних травм у значної кількості людей. Водночас зросло навантаження на лікарів і медсестер. У таких умовах автоматизація процесів догляду за пацієнтами, зокрема контролю за прийомом ліків, стає не лише актуальною, а й критично важливою для підтримки ефективності та безпеки медичного обслуговування.

Програмна система "HealthyHelper" призначена для автоматизації процесу призначення, видачі та моніторингу прийому лікарських засобів у медичних закладах. Вона охоплює повний цикл взаємодії між лікарем, пацієнтом та медичним персоналом, контролюючи процес прийому від створення призначення до фізичної видачі ліків пацієнту.

Програмна система буде реалізовувати функціональність реєстрації та подальшого управління пацієнтами, створення та зберігання призначень, управління контейнерами з ліками, сповіщень про час прийому, а також ведення статистики та генерації звітів. Система складатиметься з наступних основних компонентів:

* серверна частина, що керуватиме бізнес-логікою та взаємодією з базою даних;
* мобільний застосунок для пацієнтів та медсестер, який дозволятиме переглядати лікування, отримувати нагадування про прийом та управляти контейнерами (для медсестер);
* веб-інтерфейс для лікарів і адміністраторів, з можливістю створення та редагування призначень, перегляду статистики, а також отриманням звітностей;
* IoT-пристрій (у вигляді контейнеру для ліків), який автоматично відкриває відповідні відсіки з ліками.

Система орієнтована на поліпшення медичного обслуговування, зменшення кількості помилок у прийомі препаратів і оптимізацію роботи медичного персоналу.

## 1.2 Мета

Метою створення програмної системи є розробка комплексного інструменту для автоматизації процесів призначення та видачі медичних препаратів у закладах охорони здоров’я. Система має на меті усунути типові помилки ручного контролю прийому ліків, підвищити точність виконання призначень, забезпечити повну взаємодію між лікарями та пацієнтами, а також мінімізувати вплив людського фактору на ефективність лікувального процесу.

Система повинна забезпечити:

* зручне середовище для лікарів та медичного персоналу для формування призначень та керування графіками лікування;
* можливість пацієнтам своєчасно отримувати нагадування про прийом препаратів з контролем прийняття;
* автоматизовану видачу препаратів за допомогою IoT-пристрою;
* централізоване зберігання медичних даних з можливістю перегляду статистики та експорту звітностей;
* інтеграцію між мобільним додатком, веб-інтерфейсом і пристроєм видачі ліків для забезпечення безперервної взаємодії між усіма учасниками лікувального процесу.

Реалізація даної системи дозволить підвищити ефективність лікування, зменшити кількість пропущених доз препаратів і покращити загальну якість медичних послуг.

## 1.3 Межі

У цьому розділі визначено межі функціонування програмної системи, які окреслюють функції системи, а також її елементи, що не входять до сфери відповідальності. До меж системи входить така функціональність:

* автоматизована видача лікарських засобів за допомогою IoT-пристрою, що відкриває відсіки згідно з розкладом прийому;
* створення та редагування призначень і графіків лікування через веб-інтерфейс лікаря;
* перегляд призначень і отримання нагадувань пацієнтами та медсестрами через мобільний застосунок;
* синхронізація даних між мобільним застосунком, серверною частиною та IoT-пристроєм;
* отримання повної інформації про медичний заклад та можливість управління персоналом для адміністратора;
* надсилання push-сповіщень щодо часу прийому препаратів через Firebase Cloud Messaging;
* підтримка мобільних операційних систем (Android).

Поза межами реалізації системи залишаються:

* реалізація відеозв’язку та чату між пацієнтом та медичним персоналом;
* інтеграція з державними електронними медичними системами.

## 1.4 Посилання

У цій специфікації згадуються такі нормативні документи, стандарти та технічні ресурси:

* IEEE 830-1998 – IEEE Recommended Practice for Software Requirements Specifications;
* PostgreSQL Documentation – офіційна документація до системи керування базами даних PostgreSQL. Доступ: <https://www.postgresql.org/docs/>;
* Node.js Documentation – офіційна документація до середовища виконання JavaScript. Доступ: <https://nodejs.org/en/docs>;
* Android Developers Guide – документація для розробників Android.

Доступ: <https://developer.android.com/guide>;

* React Documentation – офіційна документація до бібліотеки React.

Доступ: <https://reactjs.org/docs/getting-started.html>.

## 1.5 Означення та абревіатури

У цьому документі використовуються такі терміни та абревіатури:

* API (Application Programming Interface) – інтерфейс прикладного програмування, набір функцій для взаємодії між компонентами системи;
* IoT (Internet of Things) – концепція підключення фізичних пристроїв до мережі Інтернет для обміну даними;
* ESP32 – мікроконтролер з підтримкою Wi-Fi і Bluetooth, який використовується як основа для IoT-пристрою видачі ліків;
* JWT (JSON Web Token) – стандарт для безпечної передачі даних між клієнтом і сервером, що використовується для автентифікації користувачів;
* REST (Representational State Transfer) – стиль архітектури веб-сервісів, який використовується для реалізації API у системі;
* CRUD (Create, Read, Update, Delete) – базові операції над даними в базі даних;
* UI (User Interface) – інтерфейс користувача, засоби взаємодії користувача із системою;
* FCM (Firebase Cloud Messaging) – сервіс для надсилання push-сповіщень на мобільні пристрої.

# 2 Загальний опис

## 2.1 Перспективи продукту

Програмна система "HealthyHelper" є стратегічно важливою частиною сучасної цифровізації охорони здоров’я. Її основна мета – автоматизація та оптимізація процесів, пов’язаних із прийомом лікарських засобів у медичних закладах різного масштабу. Система орієнтована на створення інноваційної екосистеми, яка поєднує програмні рішення разом з апаратними компонентами (IoT). Вона не є розширенням або продовженням існуючих рішень, а є новим, самостійним програмно-апаратним продуктом.

Система складатиметься з кількох взаємопов’язаних компонентів:

1. веб-інтерфейс для лікарів та адміністраторів для реєстрації пацієнтів, створення призначень, перегляду статистики.
2. мобільний застосунок для пацієнтів та персоналу, що забезпечує взаємодію з розкладом лікування, керування пристроями;
3. IoT-контейнери, які автоматично видають медикаменти відповідно до призначеного графіку;
4. серверна частина, що обробляє дані, зберігає інформацію в базі даних, керує авторизацією, ролями та взаємодією з пристроями.

Система має модульну архітектуру, що дозволяє масштабувати її функціональність і адаптувати до потреб різних установ – від невеликих приватних клінік до державних багатопрофільних лікарень. Гнучкість архітектури також передбачає можливість додавання нових компонентів без кардинальних змін.

У подальшому розробка "HealthyHelper" передбачає значне розширення. Зокрема, планується підтримка IOS-платформи, що забезпечить охоплення ширшої аудиторії користувачів, особливо у випадку персональних мобільних пристроїв пацієнтів або медичного персоналу.

Ще одним перспективним напрямком є впровадження модуля аналітики та прогнозування, який дозволить автоматично виявляти ризики пропуску прийомів, знижувати навантаження на персонал за допомогою інтелектуальних сповіщень і навіть рекомендувати зміну часу прийому на основі поведінкових шаблонів. Це значно підвищить рівень персоналізації лікування.

Крім того, проєкт передбачає гнучкість оновлення – можливість внесення змін у відповідь на нові вимоги нормативно-правових актів, появу нових технологій або зміну структури медичних закладів. Завдяки цьому "HealthyHelper" може успішно функціонувати як сьогодні, так і в майбутньому, у динамічному та складному середовищі сучасної медицини.

## 2.2 Функції продукту

Нижче наведено перелік основних функціональних можливостей системи, згрупованих за компонентами програмної системи та ролями користувачів:

а) серверна частина:

1. REST API для мобільного застосунку, веб-клієнта та IoT-пристрою;
2. база даних PostgreSQL для зберігання інформації;
3. система авторизації та аутентифікації з 4-ма ролями: лікар, медсестра, адміністратор, пацієнт;
4. функціонал для адмін-панелі (веб-частина) для управління персоналом, пацієнтами, пристроями та резервним копіюванням;

б) мобільний застосунок:

1. для медсестри:

* отримання списку пацієнтів, які на даний момент перебувають на лікуванні з їх персональною інформацією;
* отримання списку контейнерів для ліків з функціоналом для їх підключення та заповнення;
* отримання сповіщень в разі пропущеної пацієнтом дози;
* отримання інформації про поточне призначення для певного пацієнта, можливість закріпити за ним контейнер для ліків;
* можливість перегляду статистики прийому (для урахування пропущених доз);

1. для пацієнта:

* отримання інформації про стан поточного лікування у вигляді плану прийому ліків на декілька днів;
* отримання історії своїх лікувань, з можливостю перегляду призначень з вказанням лікаря та дати призначення;
* отримання нотифікацій у разі пропуску прийому;
* можливість перегляду та редагування профілю;

в) веб-частина:

1. для лікарів:

* отримання списку пацієнтів з можливістю пошуку та сортування;
* перегляд інформації про пацієнтів, можливість їх видалення, редагування або додавання нових;
* призначення лікування для пацієнтів;
* отримання звіту про лікування певного пацієнта;

1. для адміністраторів:

* перегляд та керування медичним персоналом;
* перегляд інформації про всіх зареєстрованих пацієнтів;
* отримання статистики по IoT-пристроях;
* отримання загальної інформації про медзаклад;
* можливість резервного копіювання;

г) IoT-пристрої – це розумні контейнери для автоматизованої видачі лікарських засобів, які інтегруються з центральною системою через мережу Wi-Fi. Кожен контейнер виконуватиме функції зберігання, обліку та видачі медикаментів відповідно до розкладу, сформованого лікарем у веб-інтерфейсі.

Контейнер повинен мати 9 окремих відсіків, кожен з яких оснащено індивідуальним сервомотором, що забезпечує точну механічну видачу препарату в потрібний момент часу. Відсіки можуть бути попередньо заповнені відповідальним медичним персоналом згідно з графіком прийому.

## 2.3 Характеристики користувачів

Система розроблена з урахуванням потреб чотирьох основних категорій користувачів: адміністратора, лікаря, медсестри та пацієнта. Для кожної групи реалізовано відповідний інтерфейс, рівень доступу та набір функцій:

1) пацієнти є центральними користувачами системи з точки зору отримання медичних послуг. Вони взаємодіють із системою через мобільний застосунок (на базі Android), де мають змогу переглядати свої призначення, слідкувати за графіком прийому ліків, отримувати автоматичні нагадування та отримувати звітності по лікуваннях. Враховуючи, що більшість пацієнтів не мають спеціальної технічної підготовки, інтерфейс застосунку створено інтуїтивно зрозумілим та максимально простим для щоденного використання;

2) лікарі користуються веб-інтерфейсом для створення, редагування та перегляду медичних призначень. Вони формують план лікування, обирають препарати, задають графік прийому, а також можуть переглядати історію лікування пацієнтів. Їхня взаємодія з системою здійснюється через зручний веб-інтерфейс, адаптований під настільні комп’ютери або ноутбуки. Враховуючи специфіку роботи, лікарі мають мати базові навички роботи з веб-системами та формами, однак не потребують глибоких технічних знань;

3) медсестри відповідають за налаштування IoT-контейнерів, заповнення відсіків медикаментами згідно з призначеннями, а також за моніторинг стану пацієнтів. Вони працюють як через мобільний інтерфейс застосунку, так і безпосередньо з IoT-пристроєм;

4) адміністратори системи мають розширені права доступу і відповідають за підтримку загальної працездатності системи. Вони керують обліковими записами працівників, призначають ролі, відслідковують активність у системі, переглядають статистичні дані та забезпечують технічну підтримку у разі виникнення проблем.

## 2.4 Загальні обмеження

У процесі розробки враховувалися численні фактори, які визначають поточні межі функціональності системи. Нижче наведено ключові обмеження, які необхідно враховувати при використанні та супроводі продукту:

1. платформна залежність:
   1. мобільний застосунок доступний лише для пристроїв з операційною системою Android;
   2. підтримка IOS наразі не реалізована;
2. підключення до мережі:
   1. IoT-пристрій вимагає стабільного підключення до Wi-Fi для синхронізації з сервером та отримання актуальних даних;
   2. у разі втрати з’єднання деякі функції можуть бути тимчасово недоступні;
3. часова залежність:
   1. всі розрахунки часу (прийом, видача, сповіщення) виконуються з урахуванням локального часу України (UTC+3);
   2. некоректне налаштування дати й часу на пристрої може призвести до збоїв у графіку;
4. обмеження IoT-пристрою:
   1. один контейнер підтримує максимум 9 відсіків для одночасного зберігання різних препаратів;
   2. кожен відсік може містити лише один тип медикаменту на один час прийому;
5. безпека та автентифікація:
   1. усі користувачі повинні проходити автентифікацію перед доступом до системи;
   2. втрата доступу до облікового запису може призвести до необхідності ручного відновлення.

## 

## 2.5 Припущення й залежності

Припущення щодо повноцінної та правильної роботи системи:

* кожен пацієнт має особистий пристрій (смартфон) з доступом до Інтернету та встановленим мобільним додатком "HealthyHelper";
* медичні працівники мають доступ до веб-інтерфейсу через комп’ютери, підключені до внутрішньої мережі закладу;
* IoT-пристрій підключено до стабільної Wi-Fi мережі та має постійний доступ до серверної частини;
* дані про призначення, пацієнтів та медикаменти є достовірними та вводяться кваліфікованим медичним персоналом;
* графік прийому та час налаштовано згідно з локальним часом України, а сервер синхронізований з цим часом.

Разом з тим система має низку важливих зовнішніх і внутрішніх залежностей, які впливають на її стабільну та повноцінну роботу. Однією з ключових є залежність від зовнішніх сервісів, зокрема Firebase Cloud Messaging, який використовується для надсилання push-сповіщень на мобільні пристрої користувачів, а також Cloudinary, що забезпечує зберігання й обробку зображень.

Для надсилання електронних листів при реєстрації, відновленні пароля чи інших подіях система покладається на SMTP-служби.

Серверна частина реалізована за допомогою Node.js та Prisma ORM, що створює залежність від сумісності версій цих технологій і потребує регулярного оновлення та тестування для запобігання конфліктам. Вся інформація про користувачів, призначення, ліки та пристрої зберігається у базі даних PostgreSQL, тому робота системи напряму пов’язана з наявністю активного з’єднання з цією базою.

Функціонування IoT-пристроїв залежить від коректного налаштування їх медсестрою. Окрім цього, веб-інтерфейс системи вимагає сучасного браузера з підтримкою JavaScript, що є передумовою для коректного відображення й функціонування інтерфейсу на стороні користувача.

# 3 Конкретні вимоги

## 3.1 Вимоги до зовнішніх інтерфейсів

### 3.1.1 Інтерфейс користувача

Інтерфейс користувача програмної системи має бути реалізований у вигляді веб-додатку для адміністраторів і лікарів, а також мобільного застосунку для пацієнтів і медсестер. Усі інтерфейси повинні бути інтуїтивно зрозумілими, зручними для навігації, адаптивними до різних пристроїв і відповідати принципам доступності.

Вимоги до веб-інтерфейсу (для лікарів і адміністраторів):

* веб-інтерфейс повинен бути реалізований з використанням технології React.js;
* система має підтримувати інтернаціоналізацію та підтримку декількох мов інтерфейсу (українська та англійська);
* система повинна забезпечувати адаптивне відображення елементів інтерфейсу на різних розмірах екранів;
* система повинна підтримувати аутентифікацію, захист сесій, перевірку введених даних і спливаючі повідомлення;
* для зручності навігації повинно використовуватись бічне меню;
* інтерфейс має бути реалізований у кольорах, що відповідають медичній тематиці ( наприклад білий, сірий, синій, блакитний);
* інтерфейс веб-застосунку має бути адаптивним для правильного відображення на мобільних пристроях.

Вимоги до мобільного застосунку (для пацієнтів і медичних працівників):

* мобільний застосунок має бути розроблений для Android на мові програмування Kotlin;
* для пацієнтів головний екран повинен відображати список призначених ліків на поточний день, статус прийому, коло прогресу та календар;
* для медсестер головний екран повинен відображати список пацієнтів, що знаходяться на лікуванні.
* застосунок для пацієнтів має надавати доступ до історії лікування, діагнозів, сповіщень;
* для медсестер застосунок має надавати можливість керування контейнерами (закріплювати їх за пацієнтами, заправляти або очищати відсіки);
* застосунок для обох ролей повинен надавати доступ до керування профілем, зміни пароля або фото профілю.

Вимоги до інтерфейсу IoT-пристрою:

* пристрій має містити дисплей для відображення інформації про прийом (назва препарату, час прийому, відлік часу до прийому, номер відсіку);
* пристрій має бути обладнаний датчиком для фіксації факту підтвердження прийому;
* повинні бути реалізовані фізична кнопка включення та відключення живлення;
* пристрій повинен використовувати візуальні (світлодіодні) та звукові сигнали для інформування користувача про необхідність прийому препарату.

Загальні вимоги:

* інтерфейси мають бути уніфіковані за стилем (шрифти, кольори, розмітка, відступи);
* всі форми введення мають містити перевірку коректності даних;
* інтерфейси повинні відповідати принципам доступності та UX-дизайну.

Усі інтерфейси повинні бути чітко структурованими, функціонально повними та зручними для користувачів з різним рівнем технічної підготовки.

### 3.1.2 Апаратний інтерфейс

Система повинна взаємодіяти з апаратною частиною – розумним пристроєм (IoT-контейнером для ліків). Цей пристрій має забезпечувати фізичну реалізацію розкладу прийому лікарських засобів, визначеного у серверній частині системи.

Вимоги до пристрою:

* IoT-пристрій має бути обладнаний OLED-дисплеєм для виведення повідомлень про призначення: назва препарату, час прийому, відлік часу, номер відсіку;
* повинні бути реалізовані щонайменше 9 відсіків, кожен з яких має відкриватися автоматично за допомогою окремого серводвигуна;
* кожен відсік повинен відкриватися автоматично згідно з графіком прийому, наданим сервером;
* має бути реалізовано світлодіодну індикацію стану;
* пристрій повинен мати звуковий сигналізатор для нагадування про час прийому ліків;
* мікроконтролер ESP32 повинен забезпечувати підключення до Wi-Fi і здійснювати регулярні запити до серверної частини для отримання графіка та надсилання інформації про виконання дій.

Комунікаційні вимоги:

* з'єднання між контейнером та сервером має здійснюватися через протокол HTTPS, з використанням REST API;
* обмін даними повинен відбуватись у форматі JSON;
* пристрій має підтримувати регулярну синхронізацію з сервером та оновлення (при прийомі препарату).

Вимоги до живлення:

* пристрій має бути здатним працювати від зовнішнього джерела живлення (адаптер 5V) або акумулятора;
* бажано забезпечити підтримку резервного живлення для гарантії безперервної роботи у разі збоїв електромережі.

Усі компоненти пристрою повинні надійно працювати разом із серверною частиною через стандартизовані протоколи, забезпечуючи своєчасну видачу ліків і зворотний зв’язок про стан виконання.

### 3.1.3 Програмний інтерфейс

Програмна система повинна надавати зовнішнім клієнтам доступ до функціональності через REST API, що реалізований на серверній частині системи. Усі запити й відповіді повинні здійснюватися у форматі JSON із використанням протоколу HTTPS.

Інтерфейс повинен забезпечувати аутентифікацію та авторизацію користувачів за допомогою JWT. Для IoT-пристроїв має використовуватись унікальний токен доступу. Сервер повинен підтримувати контроль доступу до маршрутів залежно від ролі користувача (пацієнт, медсестра, лікар, адміністратор).

Мобільний застосунок, який використовують пацієнти та медсестри, повинен мати можливість надсилати запити до таких маршрутів, як:

* отримання пацієнтів, що знаходяться на лікуванні;
* отримання інформації про призначення пацієнта;
* закріплення пацієнта за контейнером;
* оновлення статусу контейнера;
* заповнення/очищення відсіків контейнера;
* отримання сповіщень, отримання та редагування профіля;
* отримання звітності з лікування.

Відповіді сервера повинні повертатися у форматі JSON і містити або дані, або повідомлення про помилки у стандартизованій формі.

Веб-застосунок для лікарів і адміністраторів також використовує REST API для взаємодії з сервером. Він повинен мати доступ до функцій:

* перегляду, створення, редагування та видалення призначень;
* управління пацієнтами;
* отримання інформації профіля та її редагування;
* перегляду контейнерів;
* перегляду працівників, з можливістю їх додавання редагування та видалення;
* завантаження звітів.

Кожен маршрут повинен передбачати перевірку прав доступу, щоб запобігти несанкціонованим діям.

Окрім цього, програмна система повинна взаємодіяти з зовнішніми сервісами. Зокрема, для надсилання push-сповіщень передбачено інтеграцію з Firebase Cloud Messaging (FCM), для зберігання зображень профілю – із сервісом Cloudinary, а для генерації звітів у форматах PDF і DOCX – з використанням відповідних бібліотек (docx, exceljs).

### 3.1.4 Комунікаційний прототип

Комунікаційна архітектура системи має забезпечити стабільний та захищений обмін інформацією між усіма компонентами. Вимоги до організації взаємодії:

* система повинна використовувати архітектурну модель "клієнт-сервер", де сервер є центральним вузлом обробки запитів, а веб-інтерфейс, мобільний застосунок і IoT-пристрій – клієнтами;
* передача даних між клієнтами та сервером має здійснюватися за протоколом HTTPS із використанням формату JSON;
* мобільний застосунок повинен надсилати запити (наприклад, при підтвердженні прийому ліків або відкритті екрана) та отримувати відповіді асинхронно; у разі відсутності мережевого з’єднання застосунок має зберігати дані локально з подальшою синхронізацією після відновлення зв’язку;
* веб-додаток, який працює в браузері, повинен встановлювати захищене з’єднання з сервером, використовувати JWT-токени для авторизації та надсилати запити до API у режимі реального часу;
* IoT-пристрій повинен регулярно звертатися до сервера через Wi-Fi для перевірки графіка прийому ліків. У випадку виявлення події (наприклад, пропущеного прийому) пристрій має ініціювати відповідний POST-запит;
* система має обробляти помилки зв’язку та повертати стандартизовані повідомлення про збої. У разі недоступності сервера має бути реалізовано повторні спроби підключення або локальна обробка;
* комунікаційна модель повинна бути масштабованою, здатною обслуговувати одночасні підключення великої кількості користувачів без втрати швидкодії.

### 3.1.5 Обмеження пам'яті

У процесі функціонування система повинна враховувати обмеження обсягів оперативної та постійної пам’яті, характерні для різних компонентів.

Передбачаються такі вимоги:

* мобільний застосунок має бути оптимізований для пристроїв з обсягом оперативної пам’яті від 2 ГБ; розмір встановленого пакета APK не повинен перевищувати 100 МБ;
* веб-інтерфейс повинен працювати у браузерах без значного навантаження на оперативну пам’ять; рекомендується не перевищувати 500 МБ використання ОЗП при активному користуванні;
* IoT-пристрій повинен працювати в межах апаратних обмежень: обсяг оперативної пам’яті не повинен перевищувати 512 КБ, а пам’яті програмного коду – 4 МБ;
* при генерації звітів на сервері має бути передбачене обмеження обсягу даних, що обробляються одночасно, з метою уникнення перевантаження пам’яті;
* усі API-запити повинні мати обмеження за обсягом переданих даних, не більше 1 МБ на один запит (особливо для IoT-пристроїв);
* зберігання зображень профілю має здійснюватися у хмарному сховищі, а не в базі даних, щоб зменшити навантаження на серверну пам’ять.

### 3.1.6 Операції

Система "HealthyHelper" повинна забезпечувати виконання базових операцій, необхідних для підтримки циклу лікування. До таких операцій належать авторизація користувачів, створення та редагування призначень, розподіл медикаментів по відсіках, генерація звітів, надсилання сповіщень, підтвердження прийому та управління пристроями.

Вимоги до реалізації операцій:

1. чіткі точки входу: кожна операція повинна мати однозначно визначений маршрут або UI-компонент;
2. контроль доступу: усі запити перевіряються відповідно до ролей користувачів (пацієнт, лікар, медсестра, адміністратор);
3. валідація даних: усі вхідні дані мають перевірятися на рівні клієнта та сервера згідно з бізнес-логікою;
4. реакція в реальному часі: зміни, що стосуються лікування або пристрою, повинні оновлюватись на інтерфейсі користувача без затримок або при наступному запиті;
5. надійність і відновлення: усі критичні операції (створення призначень, фіксація прийому) повинні дублюватися в логах та оброблятися з урахуванням можливих збоїв у мережі або на пристрої.

Кожна операція повинна мати чітко визначену точку входу (через інтерфейс користувача або API) і супроводжуватися перевіркою прав доступу. Дані, які надсилаються або зберігаються в результаті виконання операцій, мають бути валідовані відповідно до логіки системи. Результати операцій повинні відображатися в інтерфейсі в реальному часі або найближчим можливим запитом.

Операційна логіка системи має бути реалізована з урахуванням надійності, цілісності даних та швидкого зворотного зв’язку.

### 3.1.7 Функції продукту

Функціональні можливості програмної системи напряму залежать від ролі користувача, оскільки кожна роль має доступ лише до частини призначеного для неї функціоналу.

1) Функції лікаря:

* MF 1.1 Система повинна надавати лікарю можливість створювати нове призначення з вказанням діагнозу, медикаментів, дозування, тривалості та часу прийому;
* MF 1.2 Лікар повинен мати можливість реєструвати, редагувати та видаляти пацієнтів;
* MF 1.3 Лікар повинен мати можливість переглядати, редагувати та видаляти призначення;
* MF 1.4 Лікар повинен мати доступ до історії лікування кожного пацієнта;
* MF 1.5 Система повинна дозволяти лікарю формувати звіти про лікування у форматі Excel;
* MF 1.6 Лікар повинен мати змогу переглядати статус прийому препаратів пацієнтами (прийнято/пропущено).

2) Функції медсестри:

* MF 2.1 Медсестра повинна мати доступ до списку пацієнтів, що перебувають на лікуванні;
* MF 2.2 Медсестра повинна мати можливість переглянути призначення, призначене лікарем;
* MF 2.3 Медсестра повинна мати можливість прив’язати IoT-контейнер до пацієнта;
* MF 2.4 Система повинна відображати список призначених препаратів для наповнення відповідних відсіків;
* MF 2.5 Медсестра повинна мати змогу вручну відкривати, очищати та перевіряти статус кожного відсіку;
* MF 2.6 Система повинна дозволяти оновлювати інформацію про стан контейнера та синхронізувати її з сервером;
* MF 2.7 Медсестра повинна мати доступ до журналу прийомів пацієнта;
* MF 2.8 Медсестра повинна отримувати сповіщення, у разі пропуску прийому пацієнтам;
* MF 2.9 Медсестра повинна мати доступ до власного профіля і мати можливість змінити пароль або фото профілю.

3) Функції пацієнта:

* MF 3.1 Пацієнт повинен мати змогу переглядати свій графік прийому препаратів у мобільному застосунку;
* MF 3.2 Пацієнт повинен мати змогу переглядати історію лікувань з можливістю завантаження звітності у форматі PDF;
* MF 3.3 Пацієнт повинен отримувати push-сповіщення про час прийому ліків;
* MF 3.4 Пацієнт повинен мати доступ до власного профіля і мати можливість змінити пароль або фото профілю;
* MF 3.5 Інтерфейс мобільного застосунку повинен бути адаптований до користувачів похилого віку.

4) Функції адміністратора:

* MF 4.1 Адміністратор повинен мати змогу керувати обліковими записами користувачів (створення, редагування, видалення);
* MF 4.2 Система повинна надавати адміністратору можливість редагувати ролі користувачів;
* MF 4.3 Адміністратор повинен мати доступ до глобальної статистики медичного закладу;
* MF 4.4 Адміністратор повинен мати змогу переглядати стан усіх пристроїв та журнал подій.

5) Функції IoT-пристрою:

* MF 5.1 IoT-пристрій повинен отримувати з сервера інформацію про час і зміст призначення;
* MF 5.2 При досягненні часу призначення пристрій повинен автоматично відкривати відповідний відсік;
* MF 5.3 Пристрій має відображати на дисплеї назву препарату, номер відсіку, час прийому та відлік часу;
* MF 5.4 Пристрій повинен активувати світлові та звукові сигнали для нагадування;
* MF 5.5 Після спрацювання датчика руху, пристрій повинен передавати цю інформацію на сервер;
* MF 5.6 У разі пропущеного прийому пристрій повинен автоматично ініціювати надсилання повідомлення про це на сервер та клієнтські пристрої.

### 3.1.8 Припущення й залежності

Для коректної роботи системи передбачається виконання низки початкових умов і зовнішніх технічних залежностей, без яких реалізація повного функціоналу є неможливою. Передбачається, що:

* усі користувачі (пацієнти, лікарі, медичний персонал, адміністратори) мають відповідні пристрої з доступом до Інтернету;
* мобільний застосунок використовується на смартфонах з Android 8.0 або новішої версії, з обсягом оперативної пам’яті не менше 2 ГБ;
* веб-інтерфейс використовується через сучасні браузери (Google Chrome, Mozilla Firefox, Microsoft Edge) з підтримкою JavaScript;
* IoT-пристрій (ESP32) підключено до стабільної Wi-Fi мережі, має доступ до Інтернету та здатен надсилати HTTP-запити;
* серверна частина функціонує на хмарному або локальному сервері з доступом до PostgreSQL бази даних;
* усі API-запити проходять через захищене з’єднання (HTTPS) з використанням токенів доступу (JWT або токен пристрою);
* зображення профілів зберігаються у зовнішньому хмарному сховищі (Cloudinary);
* push-сповіщення надсилаються через зовнішній сервіс Firebase Cloud Messaging, і доступ до нього стабільний;
* призначення, які вводяться лікарем, є коректними, повними та не потребують додаткової верифікації в системі;
* пацієнт або медичний працівник підтверджує прийом ліків вручну, система не здійснює фізичну перевірку факту прийому.

## 3.2. Властивості програмного продукту

Програмний продукт "HealthyHelper" має задовольняти низку ключових властивостей, що забезпечують його стабільне та ефективне функціонування у середовищі медичного закладу. До таких властивостей належать:

* функціональність. Програма повинна точно виконувати всі визначені функції згідно з призначенням: створення та збереження призначень, взаємодія з IoT-пристроєм, надсилання сповіщень, підтвердження прийому ліків, генерація звітів та підтримка різних ролей користувачів. Усі дії мають супроводжуватися перевіркою прав доступу та відповідати очікуваній поведінці;
* надійність. Програмний продукт має бути здатним стабільно працювати протягом тривалого часу без збоїв або втрати даних. У разі збоїв зв’язку із сервером або пристроєм повинні бути реалізовані механізми повторного з’єднання або локального кешування даних до відновлення доступу;
* зручність використання. Інтерфейси користувача повинні бути інтуїтивно зрозумілими, локалізованими (підтримка української та англійської мов), доступними на мобільних пристроях і в сучасних браузерах. Особливу увагу слід приділити користувачам похилого віку: збільшені елементи, мінімальна кількість кроків, підказки;
* ефективність. Продукт повинен використовувати системні ресурси (оперативну пам’ять, обчислювальну потужність, мережеву пропускну здатність) оптимально, без перевантаження слабких пристроїв, таких як IoT-модулі або бюджетні смартфони. Обробка запитів, збереження даних і відповіді інтерфейсу мають відбуватися без помітних затримок;
* підтримуваність. Система повинна бути структурована таким чином, щоб її можна було легко розширювати або змінювати. Код серверної частини, мобільного застосунку та інтерфейсів має бути документованим і модульним, що дозволяє вносити оновлення без необхідності повної переробки;
* портативність. Мобільний застосунок має бути сумісним з більшістю сучасних Android-пристроїв. Серверна частина має бути розгорнута на будь-якій платформі, що підтримує Node.js та PostgreSQL. Інтерфейс повинен коректно працювати в усіх популярних браузерах без встановлення додаткових компонентів.

"HealthyHelper" як програмний продукт повинен відповідати високим вимогам до функціональності, стабільності та зручності, необхідним для використання в умовах медичних закладів. Властивості програмного забезпечення мають гарантувати безпеку, надійність і масштабованість системи як у локальному середовищі, так і при віддаленому доступі.

## 3.3. Атрибути програмного коду

### 3.3.1 Надійність

Програмний код системи "HealthyHelper" повинен бути написаний з урахуванням вимог до надійності, що передбачає здатність системи функціонувати без збоїв протягом заданого часу в умовах реальної експлуатації. Надійність має досягатися за рахунок структурованості програмного коду, захисту від типових помилок (null, divide-by-zero, out-of-bounds), правильної обробки виключень, а також чіткого контролю станів компонентів. Усі критичні дії (наприклад, запис у базу даних, відкриття відсіків пристрою, підтвердження прийому препарату) повинні бути атомарними, перевіреними на помилки й супроводжуватись логуванням.

Система повинна автоматично виявляти відмови зовнішніх компонентів (недоступність API, втрата зв’язку з пристроєм або базою даних) і реагувати на них шляхом повторних спроб, кешування або повідомлення користувача без аварійного завершення програми. Дані не повинні втрачатися навіть при перериванні з’єднання – інформація тимчасово зберігається локально або в черзі на відправку.

### 3.3.2 Доступність

Програмний код системи повинен бути реалізований таким чином, щоб забезпечити постійну доступність функціональності для користувачів у межах визначених ролей та прав доступу. Система має працювати 24/7, допускаючи лише короткі вікна обслуговування або оновлення, заздалегідь повідомлені адміністраторам.

Кожен компонент системи: сервер, веб-інтерфейс, мобільний додаток та IoT-пристрій – повинен бути спроєктований з урахуванням мінімізації "вузьких місць", які можуть стати причиною недоступності. Взаємодія між модулями має бути розділена, щоб помилка одного з компонентів (наприклад, тимчасова недоступність IoT-пристрою) не блокувала роботу інших.

Серверна частина має підтримувати горизонтальне масштабування та обробку численних одночасних запитів. Мобільний застосунок повинен функціонувати навіть у разі тимчасової втрати мережі, з подальшою синхронізацією при відновленні з’єднання. Для IoT-пристрою важливо забезпечити буферизацію даних у разі втрати зв’язку з сервером.

Доступ до функцій системи має регулюватися механізмами аутентифікації та авторизації. Кожен користувач отримує лише ті функції, які передбачені його роллю. Усі спроби несанкціонованого доступу повинні фіксуватись у журналі подій.

### 3.3.3 Безпека

Програмний код повинен забезпечувати захист даних користувачів і гарантувати недопущення несанкціонованого доступу до функцій, персональної та медичної інформації. Безпека є критично важливою, оскільки система працює з чутливими персональними та медичними даними пацієнтів.

Захист доступу:

* усі запити до серверної частини мають здійснюватися виключно через захищений протокол HTTPS;
* аутентифікація користувачів реалізується через JWT-токени, які передаються в заголовках запитів і перевіряються на валідність та відповідність правам доступу;
* IoT-пристрої ідентифікуються за допомогою унікальних токенів, які перевіряються під час запиту.

Розмежування доступу:

* ролі користувачів (пацієнт, медичний працівник, лікар, адміністратор) мають чітко визначені права доступу до даних і функцій.

Захист даних:

* паролі користувачів зберігаються у зашифрованому вигляді з використанням сучасних хеш-фукцій (bcrypt);
* валідація введених даних виконується як на клієнтській, так і на серверній стороні з метою запобігання SQL-ін’єкціям, захисту від XSS-атак, фільтрації некоректного або шкідливого контенту.

Захист від втрати даних:

* термін дії токенів повинен бути обмеженим у часі з можливістю оновлення по сесії;
* система має підтримувати механізм відновлення доступу через email (посилання на зміну пароля з унікальним тимчасовим токеном).

### 3.3.4 Супроводжуваність

Програмний код системи "HealthyHelper" має бути організований і структурований так, щоб забезпечити легкість супроводу, оновлення та вдосконалення протягом усього життєвого циклу системи.

Усі компоненти системи (бекенд, фронтенд, мобільний застосунок, логіка пристрою) повинні бути реалізовані у модульному стилі, з чітким розмежуванням відповідальностей між модулями. Код має бути зрозумілим, стандартизованим, з єдиними підходами до іменування, форматування та обробки даних. Вся бізнес-логіка повинна бути винесена в окремі сервіси та контролери для полегшення подальшого розширення.

Програмний код повинен супроводжуватись вбудованими коментарями, а загальна документація має включати опис архітектури, структури бази даних, API-маршрутів. Усі зовнішні залежності (бібліотеки, фреймворки) повинні бути чітко задекларовані у відповідних файлах (package.json, build.gradle), з фіксованими версіями для уникнення конфліктів при оновленні.

Система повинна бути побудована з урахуванням зворотної сумісності, щоб у разі змін одна частина коду не порушувала роботу всієї системи.

### 3.3.5 Переносимість

Програмна система повинна бути розроблена з урахуванням вимог до переносимості, тобто здатності її компонентів функціонувати в різних програмно-апаратних середовищах з мінімальними змінами або без них.

Серверна частина системи повинна бути реалізована у кросплатформеному середовищі Node.js, що дозволяє розгортати її як на Linux-серверах (наприклад, Ubuntu), так і на Windows або хмарних платформах (Heroku, Render, AWS, Azure). База даних PostgreSQL також є переносимою і підтримується більшістю серверних платформ.

Веб-інтерфейс, розроблений із використанням React, повинен коректно відображатися у всіх сучасних браузерах без необхідності встановлення додаткових плагінів. Сумісність повинна бути забезпечена щонайменше з останніми стабільними версіями: Google Chrome, Mozilla Firefox, Microsoft Edge і Safari.

Мобільний застосунок має працювати на більшості пристроїв із операційною системою Android версії 8.0 (Oreo) або новішої. При цьому всі зовнішні API та бібліотеки, що використовуються в застосунку, повинні бути сумісними з широким діапазоном пристроїв за розміром екрана, типом архітектури процесора та обсягом пам’яті.

Для IoT-пристрою використовується мікроконтролер ESP32, який підтримує прошивку MicroPython. Усі запити мають бути реалізовані через стандартні HTTP-з'єднання, що дозволяє використовувати пристрій у будь-якому середовищі, де є стабільний Wi-Fi-зв’язок і доступ до сервера.

### 3.3.6 Продуктивність

Програмна система має забезпечувати стабільну продуктивність під час роботи з типовими обсягами навантаження в умовах реального використання в медичних установах. Основні вимоги до продуктивності стосуються швидкодії інтерфейсів, часу відповіді на запити, обробки даних та взаємодії з IoT-пристроями.

Серверна частина системи має опрацьовувати запити REST API з максимальною затримкою не більше 500 мс при стандартному навантаженні (до 50 одночасних користувачів). Усі запити до бази даних PostgreSQL мають бути оптимізовані з ефективних операторів, щоб зменшити час відповіді навіть при великій кількості записів у таблицях.

Веб-інтерфейс для лікарів та адміністраторів повинен завантажуватись не довше ніж за 2 секунди при нормальному з’єднанні. Перехід між сторінками, фільтрація таблиць, перегляд пацієнтів чи створення призначень мають бути реалізовані без помітних затримок.

Мобільний застосунок має працювати плавно на пристроях із середнім рівнем потужності, забезпечуючи відкриття основних екранів не довше ніж за 1 секунду. Головний екран з колом прогресу і календарем повинен оновлюватися миттєво, без блокувань інтерфейсу.

IoT-пристрій повинен мати можливість своєчасно (в межах 1 с) реагувати на команди відкриття відсіку, надсилати звіт про виконану дію й обробляти розклад прийому без затримок, щоб уникнути збоїв у циклі видачі медикаментів.

## 3.4 Вимоги бази даних

Система "HealthyHelper" повинна використовувати реляційну базу даних PostgreSQL як основне сховище для всієї інформації, пов’язаної з користувачами, призначеннями, контейнерами, відсіками, сповіщеннями та історією лікування. База даних має забезпечувати надійне зберігання, цілісність, безпечний доступ та ефективний пошук даних.

Усі таблиці повинні мати первинні ключі, логічні зв’язки через зовнішні ключі.

Модель бази даних має передбачати:

* нормалізацію даних не нижче 3-ї нормальної форми;
* унікальність ключових полів (логіни, id контейнерів тощо);
* зберігання паролів у зашифрованому вигляді;
* логічне розділення між пацієнтами, лікарями, медперсоналом і адміністраторами.

## 3.5 Інші вимоги

Програмна система "HealthyHelper" повинна відповідати низці додаткових вимог, які не були охоплені в попередніх підрозділах, але є важливими для її стабільного функціонування в медичному середовищі.

Система має підтримувати мультимовність: усі інтерфейси повинні бути доступні щонайменше українською та англійською мовами. Реалізація багатомовності повинна бути розширюваною – з можливістю додавання нових мов у майбутньому без зміни основного коду застосунку.

Інтерфейси (веб, мобільний, IoT) повинні відповідати принципам доступності (accessibility), зокрема для людей похилого віку та осіб з вадами зору. Застосунок має використовувати чіткі контрасти кольорів, великий текст, мінімальну кількість одночасно видимих елементів та прості навігаційні дії.

Також система повинна бути сумісною з політиками безперервної інтеграції (CI) та безперервного розгортання (CD), що дозволить ефективно тестувати, оновлювати й масштабувати окремі модулі без зупинки всієї системи.

# 4 Додаткові матеріали

До цієї специфікації додаються допоміжні матеріали (див. рис. 4.1 - 4.9), які забезпечують повне розуміння вимог до програмної системи "HealthyHelper", її структури, функціонування та особливостей впровадження.

Зображення, що містить текст, знімок екрана, Шрифт, Електрик синій

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.1 – UML-діаграма прецедентів для Web-інтерфейсу

Зображення, що містить текст, знімок екрана, схема, Шрифт

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.2 – UML-діаграма прецедентів для Mobile-застосунку

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.3 – UML-діаграма прецедентів для IoT-пристрою

Зображення, що містить текст, знімок екрана, Шрифт, схема

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.4 – UML-діаграма компонентів системи

Зображення, що містить текст, знімок екрана, Шрифт, схема

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.5 – UML-діаграма розгортування системи

Изображение выглядит как текст, диаграмма, Параллельный, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.6 – Схема бази даних

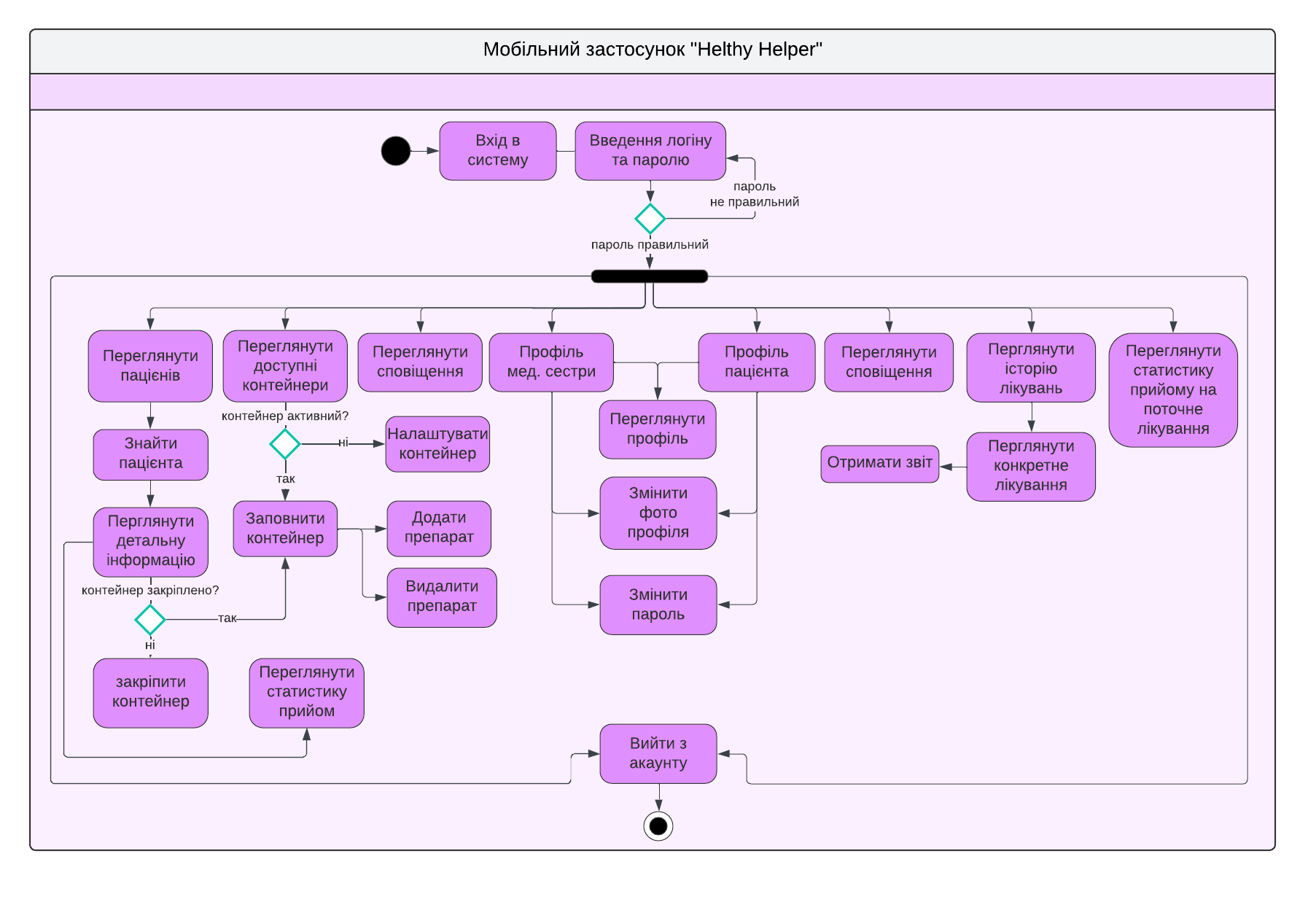


Рисунок 4.7 – Діаграма активності роботи мобільного застосунку

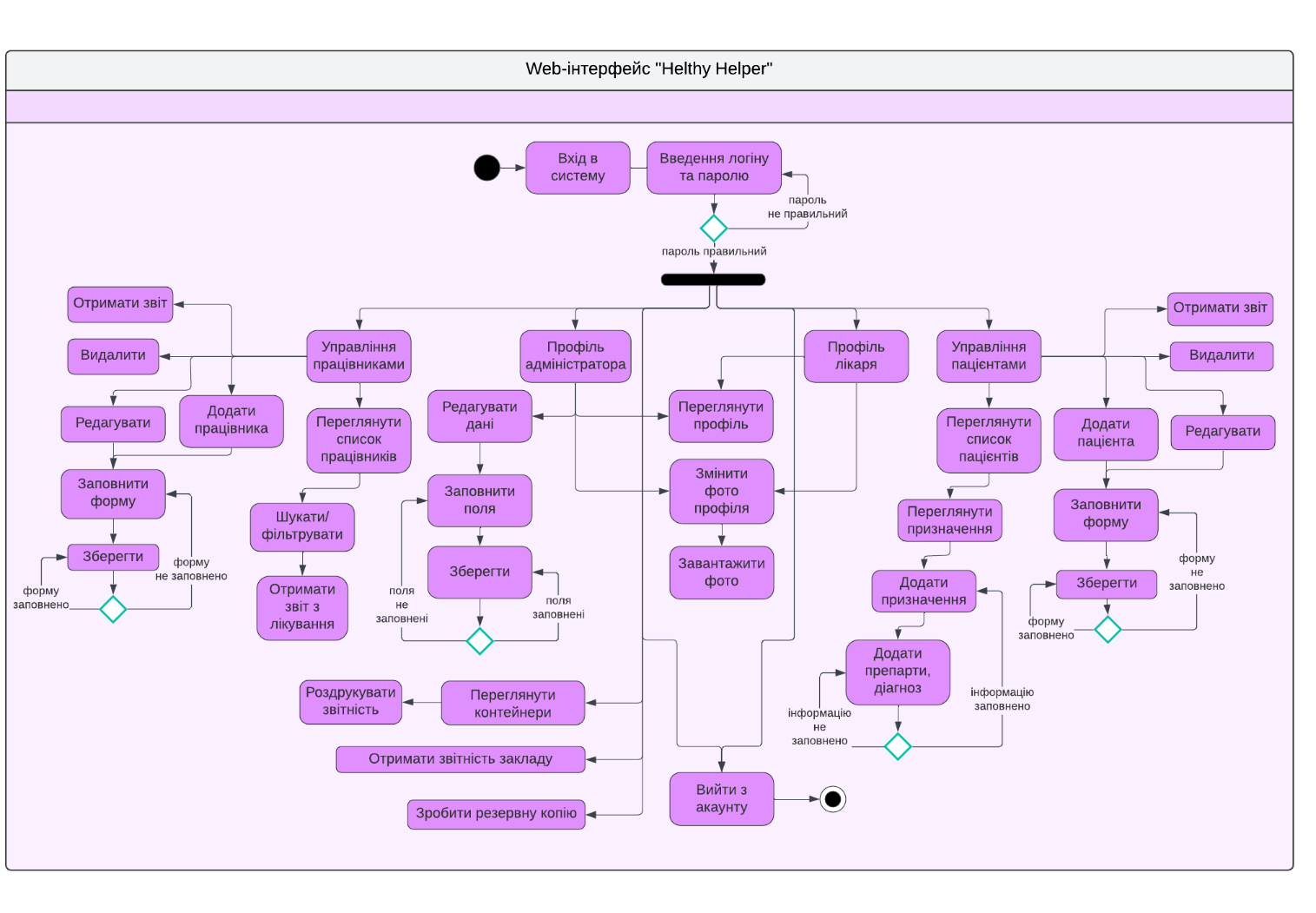


Рисунок 4.8 – Діаграма активності роботи веб-застосунку

Зображення, що містить текст, знімок екрана, схема, Шрифт

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.9 – Діаграма активності роботи IoT-контейнера