Програмна система для порятунку та реабілітації диких тварин з використанням геолокаційних сервісів та QR-кодів

Software Requirements Specification

Version 1.0

01.04.2025

Morhunov Ihor

Mobile Developer

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 11.04.2025 | Version 1.0 | Ihor Morhunov | Add first paragraph |
| 20.04.2025 | Version 1.0 | Ihor Morhunov | Add second paragraph and SRS-document |
| 03.05.2025 | Version 1.0 | Ihor Morhunov | Add third paragraph |
| 25.05.2025 | Version 1.0 | Ihor Morhunov | Add forth, fifth and sixth paragraphs |

# **Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Ihor Morhunov | Lead Software Eng. |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 69

1.1 Purpose 69

1.2 Scope 69

1.3 Definitions, Acronyms, and Abbreviations 70

1.4 References 71

1.5 Overview 72

2. General Description 72

2.1 Product Perspective 72

2.2 Product Functions 73

2.3 User Characteristics 75

2.4 General Constraints 77

2.5 Assumptions and Dependencies 78

3. Specific Requirements 78

3.1 External Interface Requirements 78

3.1.1 User Interfaces 78

3.1.2 Hardware Interfaces 79

3.1.3 Software Interfaces 79

3.1.4 Communications Interfaces 81

3.2 Functional Requirements 81

3.2.1 <Functional Requirement or Feature #1> 83

3.2.2 <Functional Requirement or Feature #2> 85

3.3 Use Cases 90

3.3.1 Use Case #1 90

3.3.2 Use Case #2 91

3.4 Classes / Objects 93

3.4.1 <Class / Object #1> 93

3.4.2 <Class / Object #2> 93

3.5 Non-Functional Requirements 95

3.5.1 Performance 95

3.5.2 Reliability 95

3.5.3 Availability 95

3.5.4 Security 95

3.5.5 Maintainability 95

3.5.6 Portability 95

3.6 Inverse Requirements 95

3.7 Design Constraints 97

3.8 Logical Database Requirements 98

3.9 Other Requirements 99

4. CHANGE MANAGEMENT PROCESS 100

# **1. Introduction**

## **1.1 Purpose**

## Цей документ є специфікацією вимог до програмного забезпечення (SRS) для веб-системи та мобільного застосунку з порятунку та реабілітації диких тварин з використанням геолокаційних сервісів та QR-кодів.

## Основна аудиторія цього документа:

## Розробники (backend, frontend, mobile) – для отримання чітких вимог до системи.

## Менеджери проєкту – для розуміння функціональності та координації розробки.

## Тестувальники – для створення тестових сценаріїв на основі функціональних вимог.

## Замовники та зацікавлені сторони – для отримання уявлення про можливості продукту та його відповідність вимогам.

## Документ описує функціональні та нефункціональні вимоги до системи, її основні функції та обмеження.

## **1.2 Scope**

## Програмний продукт включає веб-систему та мобільний застосунок, які працюватимуть у взаємозв'язку для організації процесу порятунку, лікування та реабілітації диких тварин.

## Продукт надає такі можливості:

## Реєстрація та авторизація користувачів через пошту та сторонні сервіси (Google).

## Створення повідомлень про тварин, яким потрібна допомога, з можливістю додавання фото, опису та геолокації.

## Карта з позначками місцезнаходження тварин, що потребують допомоги.

## QR-коди для тварин, що дозволяють отримати інформацію про їх стан, історію лікування.

## Адміністративна панель для управління користувачами, тваринами та системними параметрами.

## Доступ до історії лікування конкретної тварини для ветеринарів.

## Карта з ветеринарними клініками, які працюють у режимі реального часу.

## Автоматизована звітність про врятованих тварин та надану допомогу.

## Подальший розвиток системи може включати чат між користувачами, волонтерами та ветеринарами.

## Обмеження

## Система не виконує безпосереднє транспортування чи лікування тварин – вона лише координує процес між учасниками.

## Не передбачена можливість інтеграції з урядовими структурами на початкових етапах.

## Цілі та переваги

## Швидке реагування на випадки поранення чи хвороби тварин.

## Зручність для волонтерів та ветеринарів – єдина система для координації.

## Оптимізація порятунку через використання картографічних сервісів та QR-кодів.

## Прозорість – збереження історії лікування кожної тварини.

## **1.3 Definitions, Acronyms, and Abbreviations**

## У цьому розділі наведено визначення термінів, акронімів та абревіатур, які використовуються в цьому документі:

## SRS (Software Requirements Specification) – Специфікація вимог до програмного забезпечення.

## API (Application Programming Interface) – Програмний інтерфейс для взаємодії між компонентами системи.

## QR-код (Quick Response Code) – Двовимірний штрихкод, що містить інформацію, яка може бути швидко зчитана камерою смартфона.

## Геолокаційні сервіси – Сервіси, що визначають місцезнаходження користувача або об’єкта на карті.

## Волонтер – Особа, яка бере участь у рятуванні диких тварин, взаємодіє з платформою для отримання інформації та координації дій.

## Ветеринар – Фахівець, який надає медичну допомогу тваринам та використовує систему для перегляду історії лікування.

## Адміністратор – Користувач, який має доступ до керування даними в системі (користувачами, тваринами тощо).

## Мобільний застосунок – Програмне забезпечення, розроблене для смартфонів та планшетів, що дозволяє користувачам взаємодіяти із системою.

## Back-end – Серверна частина програмного забезпечення, яка обробляє запити та забезпечує взаємодію між компонентами системи.

## Front-end – Клієнтська частина програмного забезпечення, що відображає інтерфейс користувача.

## MariaDB – Система керування базами даних, яка використовується для зберігання інформації про тварин, користувачів та інші елементи системи.

## **1.4 References**

У цьому розділі наведено список документів та джерел, які використовувалися при підготовці специфікації або містять корисну інформацію для розробки продукту:

* IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications.
* REST API Documentation – Рекомендації щодо розробки REST API.
* MariaDB Documentation – Офіційна документація MariaDB (https://mariadb.com/kb/en/documentation/).
* Google Maps API Documentation – Документація API геолокаційних сервісів Google (https://developers.google.com/maps).
* OWASP Secure Coding Practices – Рекомендації щодо безпечної розробки (<https://owasp.org/>).

## **1.5 Overview**

# Цей документ є специфікацією вимог до програмного забезпечення (SRS) для веб-системи та мобільного застосунку, які призначені для порятунку та реабілітації диких тварин. Його основна мета – надати детальний опис функціональності, вимог та обмежень, необхідних для розробки системи, а також забезпечити єдине бачення продукту для всіх учасників проєкту, включаючи розробників, тестувальників, менеджерів і зацікавлені сторони.

# Документ організований наступним чином: у Розділі 1: Вступ представлено загальну інформацію про проєкт, включаючи його призначення, область застосування, визначення термінів і скорочень, а також список використаних посилань. Він також містить цей огляд, який допомагає читачеві зрозуміти структуру SRS та подальший зміст документа. У Розділі 2: Загальний опис надано загальний огляд продукту, його основних можливостей, ключових характеристик користувачів, функціональних обмежень, залежностей та припущень, що впливають на розробку. Розділ 3: Специфічні вимоги є основним розділом документа, який містить детальний опис функціональних та нефункціональних вимог до системи. У ньому представлено конкретні сценарії використання, описи основних модулів, вимоги до інтерфейсу, безпеки, продуктивності та сумісності з іншими системами. Розділ 4: Додатки, якщо необхідно, міститиме додаткові матеріали, які не є частиною основної специфікації, але можуть бути корисними для розробників або інших учасників проєкту.

# SRS організований таким чином, щоб забезпечити логічний і послідовний опис всіх аспектів розроблюваної системи. Він починається з високорівневого опису цілей і можливостей продукту та поступово деталізує всі вимоги, необхідні для його створення. Така структура дозволяє легко знаходити потрібну інформацію та використовувати документ як довідник на всіх етапах розробки: від проєктування та програмування до тестування та впровадження.

# **2. General Description**

## **2.1 Product Perspective**

## Проєкт з порятунку та реабілітації диких тварин є частиною екосистеми цифрових рішень для охорони природи та волонтерської діяльності. Він надає платформу для онлайн-координації порятунку тварин, що потрапили в біду, забезпечуючи взаємодію між волонтерами, ветеринарами та реабілітаційними центрами.

## Система інтегрується з геолокаційними сервісами для точного визначення місця перебування тварин, ветеринарними базами даних для збереження історії лікування, а також із засобами комунікації, такими як push-сповіщення та внутрішній чат, для оперативного інформування учасників процесу порятунку.

## Існують аналогічні продукти, спрямовані на допомогу бездомним тваринам або пошук зниклих домашніх улюбленців, проте наша система орієнтована саме на рятування диких тварин. Вона відрізняється розширеним функціоналом, включаючи використання QR-кодів для швидкого доступу до інформації про тварину, інтерактивну карту поточних запитів про допомогу та можливість адміністрування процесу через спеціальну панель.

## Основна мета системи – підвищити ефективність порятунку та реабілітації тварин завдяки зручному інтерфейсу, автоматизації процесів та інтеграції з сучасними цифровими інструментами.

## **2.2 Product Functions**

## Система для порятунку та реабілітації диких тварин забезпечує комплексний набір функцій для всіх категорій користувачів, включаючи волонтерів, ветеринарів, реабілітаційні центри та адміністраторів. Вона дозволяє оперативно фіксувати випадки виявлення поранених або хворих тварин, координувати їхнє транспортування та лікування, а також вести облік усіх врятованих особин.

1. Управління користувачами:

* Реєстрація та авторизація через електронну пошту, Google-акаунт або інші сторонні сервіси.
* Відновлення пароля через email.
* Розподіл ролей користувачів (волонтер, ветеринар, адміністратор).
* Перегляд та редагування профілю користувача.

2. Повідомлення про тварин, що потребують допомоги:

* Додавання інформації про знайдену тварину: геолокація місця виявлення, фото тварини, попередній опис стану (поранення, хвороба, загальний вигляд).
* Надсилання повідомлення волонтерам і ветеринарам про новий випадок.
* Створення запиту на транспортування тварини в реабілітаційний центр або клініку.

3. Карта та геолокаційні сервіси

* Відображення знайдених тварин на інтерактивній карті.
* Побудова маршруту до тварини для волонтерів.
* Відображення ветеринарних клінік, які зараз працюють.
* Відстеження переміщення тварини (за необхідності).

4. Система QR-кодів

* Генерація унікального QR-коду для кожної врятованої тварини.
* Сканування QR-коду для перегляду історії лікування тварини.
* Використання QR-кодів у ветеринарних центрах для швидкого доступу до картки тварини.

5. Адміністративна панель

* Управління користувачами (додавання, блокування, зміна ролей).
* Моніторинг усіх запитів про допомогу тваринам.
* Керування базою даних про тварин.
* Перегляд статистики врятованих тварин.

6. Система сповіщень

* Push-сповіщення для волонтерів і ветеринарів про нові випадки допомоги.
* Повідомлення адміністраторам про критичні події (наприклад, масові звернення про допомогу).
* Сповіщення користувачів про статус обробки їхніх запитів.

7. Взаємодія між користувачами

* Вбудований чат між волонтерами, ветеринарами та адміністраторами (планується на наступних етапах розробки).
* Коментування та уточнення деталей у картках тварин.
* Система рейтингів для волонтерів та ветеринарів на основі їхньої активності.

8. Ведення історії лікування тварин

* Запис інформації про стан тварини після її порятунку.
* Додавання діагнозів, проведених маніпуляцій та рекомендованого лікування.
* Фіксація дати повернення тварини у природне середовище або передавання під опіку.

9. Генерація звітів та статистика

* Автоматичне створення звітів про нові врятовані тварини.
* Статистика за видами тварин, регіонами та рівнем небезпеки.
* Відстеження ефективності волонтерів та ветеринарних центрів.

## **2.3 User Characteristics**

## 1. Волонтери

## Опис: Волонтери – це люди, які допомагають у порятунку тварин, виявляють постраждалих особин, транспортують їх до ветеринарних клінік або реабілітаційних центрів.

## Характеристики:

## Може мати як низький, так і високий рівень цифрової грамотності.

## Найчастіше користується мобільною версією додатку.

## Потребує швидкого доступу до карти з позначенням тварин, що потребують допомоги.

## Очікує отримання push-сповіщень про нові випадки.

## Має обмежений час на заповнення детальної інформації про тварину, тому інтерфейс має бути максимально простим.

## Основні вимоги:

## Легке створення повідомлень про знайдених тварин.

## Інтерактивна карта з можливістю перегляду найближчих запитів.

## Інтуїтивно зрозумілий інтерфейс для швидкої взаємодії з системою.

## Можливість швидкого сканування QR-коду для отримання даних про тварину.

## 2. Ветеринари

## Опис: Це фахівці, які проводять медичний огляд, лікують і надають допомогу знайденим тваринам.

## Характеристики:

## Середній або високий рівень цифрової грамотності.

## Часто працює зі стаціонарним комп’ютером або планшетом.

## Очікує детальну інформацію про тварину до її прибуття у клініку.

## Має потребу в зручному веденні історії лікування.

## Основні вимоги:

## Перегляд історії лікування кожної тварини.

## Доступ до QR-кодів для швидкого отримання інформації.

## Інтерактивний список тварин, які потребують допомоги, з можливістю фільтрації за пріоритетом.

## Можливість додавання діагнозів, схем лікування та результатів аналізів.

## 3. Працівники реабілітаційних центрів

## Опис: Це організації та спеціалісти, які забезпечують відновлення та адаптацію тварин після лікування.

## Характеристики:

## Мають середній або високий рівень цифрової грамотності.

## Потребують доступу до повної інформації про тварину, її стан і перебіг лікування.

## Мають потребу у веденні документації щодо перебування тварини у центрі.

## Основні вимоги:

## Перегляд історії тварини з можливістю оновлення даних.

## Фіксація змін у стані здоров’я та процесі адаптації.

## Доступ до QR-кодів для швидкої ідентифікації.

## Генерація звітів про тварин, що перебувають у центрі.

## 4. Адміністратори системи

## Опис: Це спеціальні користувачі, які мають доступ до управління всією системою, користувачами та базами даних.

## Характеристики:

## Високий рівень цифрової грамотності.

## Переважно працює зі стаціонарним комп’ютером.

## Має потребу у детальному моніторингу всіх процесів у системі.

## Основні вимоги:

## Управління користувачами (блокування, зміна ролей, контроль активності).

## Контроль за базою тварин та історіями лікування.

## Генерація аналітичних звітів.

## Можливість реагування на збої в роботі системи та виправлення даних.

## **2.4 General Constraints**

Загальні обмеження на розробку системи порятунку та реабілітації диких тварин охоплюють низку факторів, що можуть вплинути на вибір технологій, архітектуру та проектування всієї системи. Система повинна бути сумісною з різними платформами та пристроями, зокрема веб-браузерами (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) та мобільними операційними системами (Android 8.0 і вище, iOS 14.0 і вище), а також коректно відображатися на мобільних пристроях, планшетах та десктопах. Це накладає вимоги до адаптивності інтерфейсу, що буде враховуватися під час розробки як фронтенду, так і мобільного застосунку, який повинен працювати на базі Flutter для підтримки обох платформ — Android та iOS.

Щодо технічних характеристик, система повинна мати можливість обробляти велику кількість одночасних запитів — орієнтовно до 1000 користувачів одночасно. Це вимагає особливої уваги до продуктивності, зокрема для забезпечення швидкої роботи карти з геолокацією тварин та обробки бази даних, де кількість записів може досягати десятків тисяч. Вибір технологій для серверної частини обмежений використанням Spring Boot для розробки backend, MariaDB для бази даних, а також React.js для фронтенду. Мобільний застосунок буде створений за допомогою Flutter, що дозволить підтримувати кросплатформеність.

Інтеграція з іншими системами, такими як платіжні сервіси (наприклад, Stripe або PayPal) для збору коштів на допомогу тваринам, а також з геолокаційними сервісами на базі Google Maps API, вимагає врахування певних обмежень на частоту запитів та використання їхніх безкоштовних квот. Також важливою є інтеграція з системами оповіщення, такими як Firebase Cloud Messaging, для сповіщень користувачів про нові тварини, що потребують допомоги.

З точки зору організаційних обмежень, розробка основної версії продукту має бути завершена в межах 6–9 місяців, а після цього має бути проведено тестування та виправлення помилок протягом 2–3 місяців. Враховуючи бюджетні обмеження, перевага віддається використанню відкритих технологій та хмарних рішень з гнучкою тарифікацією, щоб мінімізувати витрати на інфраструктуру та ліцензії.

Крім того, необхідно забезпечити відповідність вимогам щодо захисту персональних даних користувачів, зокрема відповідність стандартам GDPR, якщо в системі будуть оброблятися дані користувачів з Європейського Союзу. Особлива увага приділяється безпеці — система повинна бути захищена від SQL-ін’єкцій, атак XSS та CSRF, а також бути стійкою до DDoS-атак. Для цього передбачено використання токенів для автентифікації API та двофакторної автентифікації для адміністраторів.

Інтерфейс системи має бути простим і зручним, особливо для волонтерів, які можуть працювати в стресових ситуаціях. Інтерфейс повинен бути адаптованим до різних мов, зокрема української та англійської, для забезпечення доступності для більш широкої аудиторії. Також важливо забезпечити захист геолокаційних даних, щоб координати тварин були доступні тільки авторизованим користувачам і не розголошувалися без потреби. Дані повинні бути розмитими для запобігання зловживанням.

## **2.5 Assumptions and Dependencies**

# Фактори, які можуть вплинути на вимоги до програмного забезпечення, включають кілька ключових аспектів. По-перше, технічні вимоги до апаратного забезпечення можуть мати вплив на вимоги до програмного забезпечення. Якщо програмне забезпечення передбачає використання специфічних серверів або процесорів, зміна апаратної платформи може призвести до необхідності змінити вимоги щодо продуктивності або сумісності з операційними системами. Наприклад, якщо апаратне забезпечення не підтримує певний процесор чи операційну систему, може знадобитися розглянути альтернативні варіанти для забезпечення необхідної функціональності.

# По-друге, важливими є операційні системи та версії, на яких буде працювати система. Якщо на етапі розробки чи впровадження з’ясується, що ці операційні системи не будуть доступні або не підтримують потрібну функціональність, вимоги до програмного забезпечення можуть змінитися. Це може стосуватися таких систем як Android, iOS, Windows Server чи Linux, і в разі необхідності, може виникнути потреба в підтримці інших операційних систем або зміни технологій, що підтримуються.

# Інтеграція з зовнішніми сервісами також є важливим фактором. Програмне забезпечення може інтегруватися з такими сервісами, як Google Maps API для геолокації або платіжні системи Stripe та PayPal. Зміни в політиках цих сервісів, API або умовах використання можуть вимагати оновлення вимог або навіть зміну архітектури системи. Наприклад, зміни в тарифікації API або обмеження за запитами можуть потребувати коригування стратегії інтеграції.

# Ще одним важливим фактором є кількість користувачів та навантаження на систему. Якщо виявиться, що реальна кількість користувачів значно більша за прогнозовану, система може потребувати масштабування або змін у вимогах щодо обробки великої кількості одночасних запитів. Це може призвести до змін у підходах до забезпечення продуктивності або оптимізації архітектури системи.

# Зміни в законодавстві або нормативних вимогах також можуть вплинути на вимоги. Наприклад, якщо змінюються закони про захист персональних даних (наприклад, GDPR у ЄС), це може потребувати змін у вимогах безпеки, зберігання та обробки даних користувачів. Нові вимоги до зберігання чи обробки особистої інформації можуть призвести до необхідності коригування вимог до криптографічного захисту даних або правил доступу до персональних даних.

# Зміни в технічних стандартах або індустріальних практиках можуть також впливати на вимоги. Наприклад, нові стандарти для мобільних додатків або веб-систем можуть вимагати внесення змін у функціональність продукту для відповідності цим стандартам. Зміни в цих стандартах можуть потребувати додаткових функцій, таких як покращення безпеки чи сумісності з іншими системами.

# Бюджетні обмеження та зміни в ресурсах можуть значно вплинути на обсяг реалізованих функцій. Якщо бюджет буде обмежений, це може призвести до скорочення або відкладення певних можливостей продукту на пізніші етапи. Зміни в наявних ресурсах, таких як фінансові або людські, також можуть вплинути на обсяг роботи, яку можна виконати в межах проєкту.

# Вимоги до користувацького досвіду (UX) можуть змінюватися в процесі тестування та зворотного зв’язку з реальними користувачами. Якщо виявиться, що певні елементи інтерфейсу є занадто складними або неінтуїтивно зрозумілими, вимоги до інтерфейсу можуть бути змінені для покращення досвіду користувачів та зручності використання.

# З появою нових технологій та інструментів для розробки програмного забезпечення можуть з’явитися нові можливості для покращення функціональності, продуктивності або безпеки системи. Це може призвести до змін у вимогах для використання нових технологій або забезпечення сумісності з ними, що дозволить підвищити ефективність та надійність системи.

# **3. Specific Requirements**

## **3.1 External Interface Requirements**

## 3.1.1 Інтерфейси користувача Інтерфейси користувача описують, як користувачі будуть взаємодіяти з системою, включаючи макет, дизайн та функціональність графічного інтерфейсу користувача (GUI) як для веб-платформи, так і для мобільного застосунку. Інтерфейси повинні бути інтуїтивно зрозумілими, зручними для користувачів і підтримувати різні пристрої, такі як смартфони, планшети та настільні комп'ютери. Наприклад, інтерфейс системи реєстрації тварин повинен дозволяти користувачам швидко додавати нових тварин до бази, переглядати деталі поранених або хворих тварин, а також отримувати актуальну інформацію про їх стан. Крім того, адміністратори повинні мати можливість управляти даними користувачів, тварин та переглядати звітність через окрему адміністративну панель.

## 3.1.2 Інтерфейси апаратного забезпечення Інтерфейси апаратного забезпечення визначають вимоги до фізичних пристроїв, з якими система повинна взаємодіяти. Це включає серверну інфраструктуру, яка буде хостити веб-платформу та базу даних, а також зовнішні пристрої, такі як термінали для прийому платежів або IoT-пристрої, які можуть бути інтегровані. Система повинна працювати з зазначеними апаратними платформами (наприклад, сервери на базі x86, мобільні пристрої з певними характеристиками) та забезпечити сумісність з периферійними пристроями, такими як сканери штрих-кодів або принтери в місцях реєстрації тварин у реабілітаційних центрах.

## 3.1.3 Програмні інтерфейси Програмні інтерфейси описують, як система буде взаємодіяти з іншими програмними застосунками або платформами. Це може включати інтеграцію з API сторонніх сервісів для обробки платежів (наприклад, Stripe, PayPal), зовнішні сервіси для геолокації (наприклад, Google Maps) або вже існуючі системи управління реабілітацією диких тварин. Крім того, система повинна надавати API для доступу до даних про користувачів, інформації про тварин та звітів, забезпечуючи сумісність і плавний обмін даними з іншими програмними компонентами або внутрішніми системами в межах організації, що займається порятунком та реабілітацією тварин.

## 3.1.4 Комунікаційні інтерфейси Комунікаційні інтерфейси визначають вимоги до мережевого зв'язку та передачі даних між компонентами системи. Це включає протоколи, мережеві стандарти (наприклад, HTTP/HTTPS, WebSockets) і методи комунікації для внутрішньої та зовнішньої взаємодії. Наприклад, система повинна підтримувати захищені канали зв'язку для аутентифікації користувачів, обробки платежів та оновлення інформації про доступність тварин в реальному часі. Також може бути необхідною інтеграція з зовнішніми сервісами для надсилання повідомлень або електронних листів з підтвердженнями реєстрації тварин або сповіщеннями для волонтерів і ветеринарів.

## **3.2 Functional Requirements.**

## 3.2.1 Реєстрація та автентифікація користувачів

## 3.2.1.1 Опис

## Система повинна дозволяти користувачам (волонтерам, ветеринарам та адміністраторам) реєструвати акаунт та автентифікуватися за допомогою електронної пошти або сторонніх сервісів, таких як Google. Також має бути підтримка функції відновлення пароля.

## 3.2.1.2 Вхідні дані

## Адреса електронної пошти або дані стороннього акаунта (Google).

## Пароль (для реєстрації через електронну пошту).

## CAPTCHA або інші методи перевірки для уникнення автоматичних реєстрацій.

## 3.2.1.3 Обробка

## Система перевіряє облікові дані користувача під час входу.

## При використанні електронної пошти, пароль хешується та порівнюється з даними в базі.

## Для сторонньої автентифікації здійснюється виклик API сервісу (Google).

## Система перевіряє наявність електронної пошти або акаунта стороннього сервісу та надсилає лист для підтвердження.

## 3.2.1.4 Вихідні дані

## Після успішного входу користувач перенаправляється на свою панель управління.

## У разі неуспіху відображається повідомлення про помилку з підказкою (неправильний пароль або непідтверджена електронна пошта).

## Лист для підтвердження електронної пошти надсилається на вказану адресу.

## 3.2.1.5 Обробка помилок

## Невірні дані для входу призводять до загального повідомлення про помилку з підказкою спробувати знову.

## Якщо електронна пошта вже зареєстрована, відображається відповідне повідомлення.

## При забутому паролі надсилається електронний лист з інструкціями для відновлення пароля.

## 3.2.2 Звітність про допомогу тваринам

## 3.2.2.1 Опис

## Система повинна дозволяти користувачам повідомляти про поранених або хворих тварин, надаючи детальну інформацію, включаючи місце розташування, опис тварини та рівень терміновості.

## 3.2.2.2 Вхідні дані

## Опис тварини (вид, стан, інші важливі деталі).

## Геолокація (автоматичне визначення або ручне введення).

## Рівень терміновості (високий, середній, низький).

## 3.2.2.3 Обробка

## Система перевіряє подану інформацію та позначає місце розташування на карті.

## Повідомлення відправляється волонтерам або ветеринарам відповідно до рівня терміновості.

## Запис про звернення зберігається для подальшого відстеження.

## 3.2.2.4 Вихідні дані

## Після подачі форми відображається повідомлення про підтвердження.

## Оновлення карти в реальному часі з позначенням місця тварини та її статусу.

## Сповіщення надсилаються відповідним фахівцям (волонтерам, ветеринарам).

## 3.2.2.5 Обробка помилок

## Наявність відсутньої інформації призводить до запиту на додавання необхідних деталей.

## Якщо геолокація недоступна, користувач може ввести місце вручну.

## У разі помилки у системі або проблем із мережею, користувачу надсилається сповіщення про необхідність спробувати пізніше.

## 3.2.3 Історія лікування тварин

## 3.2.3.1 Опис

## Система повинна дозволяти ветеринарам та адміністраторам додавати та переглядати історію лікування тварин, що лікуються або реабілітуються.

## 3.2.3.2 Вхідні дані

## Ідентифікатор тварини або QR-код.

## Деталі лікування (діагноз, медикаменти, процедури).

## Дата та час лікування.

## Примітки ветеринара.

## 3.2.3.3 Обробка

## Система зв’язує записи про лікування з конкретним профілем тварини.

## Дані зберігаються в безпечному режимі та доступні відповідно до ролі користувача (ветеринар, адміністратор).

## Історія лікування відображається у вигляді хронологічної лінії.

## 3.2.3.4 Вихідні дані

## Історія лікування тварини відображається в легкому для читання форматі.

## Ветеринари можуть оновлювати історію або додавати нові записи про лікування.

## 3.2.3.5 Обробка помилок

## Невірний ідентифікатор тварини або QR-код призводить до повідомлення про помилку з підказкою спробувати знову.

## Неповні записи лікування відзначаються для перевірки.

## 3.2.4 Геолокація місць знаходження тварин

## 3.2.4.1 Опис

## Система повинна дозволяти користувачам (волонтерам, ветеринарам) переглядати поточне місцезнаходження тварин на карті, що спрощує координацію рятувальних операцій.

## 3.2.4.2 Вхідні дані

## Дані GPS з пристрою користувача або введення вручну.

## Дані про тварину.

## 3.2.4.3 Обробка

## Система використовує GPS-дані для відображення місця знаходження тварини на карті.

## Волонтери та ветеринари можуть фільтрувати тварин за близькістю або рівнем терміновості.

## 3.2.4.4 Вихідні дані

## Інтерфейс карти, що показує місцезнаходження тварин з маркерами для різних рівнів терміновості.

## Оновлення місцезнаходжень тварин у реальному часі при подачі нових даних.

## 3.2.4.5 Обробка помилок

## Відсутність GPS-сигналу викликає запит на ручне введення місцезнаходження.

## При некоректно введених даних система запитує повторне введення.

## 3.2.5 Інтеграція платіжної системи

## 3.2.5.1 Опис

## Система повинна підтримувати інтеграцію з платіжними системами (наприклад, Stripe, PayPal) для здійснення донатів або оплати за лікування тварин та послуги.

## 3.2.5.2 Вхідні дані

## Сума платежу.

## Дані платіжного засобу (кредитна картка, аккаунт PayPal).

## 3.2.5.3 Обробка

## Дані платіжного засобу безпечно передаються в сторонню платіжну систему.

## Система підтверджує статус платежу (успіх або помилка).

## Після успішного платежу користувач отримує квитанцію та підтвердження.

## 3.2.5.4 Вихідні дані

## Сторінка підтвердження платежу з деталями транзакції.

## Квитанція відправляється користувачу на електронну пошту.

## 3.2.5.5 Обробка помилок

## Неуспішні платежі призводять до відображення описової помилки (наприклад, відмова картки, недостатньо коштів).

## У разі проблем з платіжним шлюзом користувачу надається повідомлення про необхідність спробувати пізніше.

## 3.2.6 Панель управління адміністратора

## 3.2.6.1 Опис

## Адміністратори повинні мати можливість керувати користувачами, моніторити звернення про допомогу тваринам та генерувати звіти за допомогою панелі адміністратора.

## 3.2.6.2 Вхідні дані

## Дані користувачів та тварин.

## Параметри пошуку та фільтрації для управління записами.

## 3.2.6.3 Обробка

## Система дозволяє адміністраторам створювати, оновлювати та видаляти акаунти користувачів.

## Адміністратори можуть переглядати та управляти зверненнями про допомогу, історіями лікування та статусами тварин.

## Генерація звітів за певними параметрами (наприклад, кількість врятованих тварин, надані лікування).

## 3.2.6.4 Вихідні дані

## Інтерфейс управління користувачами для додавання або видалення акаунтів.

## Генератор звітів, який виводить PDF або CSV файли для завантаження.

## 3.2.6.5 Обробка помилок

## Несанкціоновані спроби доступу реєструються та блокуються.

## Наявність неповних даних викликає повідомлення для адміністратора.

## 3.2.7 Сканування QR-кодів тварин

## 3.2.7.1 Опис

## Система повинна дозволяти користувачам сканувати QR-коди, прикріплені до тварин, для швидкого доступу до їхніх профілів та історії лікування.

## 3.2.7.2 Вхідні дані

## Дані QR-коду, скановані за допомогою мобільного пристрою або планшета.

## 3.2.7.3 Обробка

## Система декодує QR-код і отримує відповідні дані про тварину з бази даних.

## 3.2.7.4 Вихідні дані

## Профіль тварини та історія лікування відображаються на пристрої користувача.

## Можливість для ветеринарів або волонтерів оновлювати дані про лікування.

## 3.2.7.5 Обробка помилок

## Якщо QR-код недійсний або не знайдений, система запитує повторне сканування або введення вручну.

## **3.3 Use Cases**

## 3.3.1 Use Case #1: Реєстрація користувача

## Актори: Користувач, система

## Опис: Користувач створює обліковий запис у системі за допомогою адреси електронної пошти або сторонніх сервісів (наприклад, Google).

## Вхід: Адреса електронної пошти, пароль (або авторизація через сторонній сервіс).

## Процес: Користувач заповнює форму для реєстрації, після чого отримує підтвердження реєстрації.

## Вихід: Підтвердження реєстрації, створення облікового запису.

## Обробка помилок: Якщо введено некоректну електронну пошту або пароль, система видає повідомлення про помилку.

## 3.3.2 Use Case #2: Авторизація користувача

## Актори: Користувач, система

## Опис: Користувач входить у систему, використовуючи свою електронну пошту і пароль або сторонні сервіси (Google).

## Вхід: Електронна пошта, пароль або авторизація через Google.

## Процес: Користувач вводить свої дані для входу, система перевіряє їх і надає доступ.

## Вихід: Доступ до системи.

## Обробка помилок: У разі неправильних даних система відображає повідомлення про помилку.

## 3.3.3 Use Case #3: Створення повідомлення про допомогу тварині

## Актори: Користувач, система

## Опис: Користувач створює повідомлення для волонтерів або ветеринарів про тварину, яка потребує допомоги.

## Вхід: Дані про тварину (тип, місцезнаходження, стан).

## Процес: Користувач заповнює форму, що містить деталі про тварину, та надсилає повідомлення.

## Вихід: Підтвердження створення повідомлення, інформація про тварину відправляється ветеринарам та волонтерам.

## Обробка помилок: У разі відсутності обов’язкових даних система повідомляє про помилку.

## 3.3.4 Use Case #4: Перегляд карти з тваринами, що потребують допомоги

## Актори: Волонтери, ветеринари

## Опис: Ветеринари та волонтери переглядають карту, на якій відображаються місця, де тварини потребують допомоги.

## Вхід: Запит на карту з тваринами в радіусі.

## Процес: Система надає актуальну інформацію на карті про тварин, що потребують допомоги, з можливістю фільтрації за категоріями.

## Вихід: Відображення карти з тваринами та їхнім станом.

## Обробка помилок: У разі відсутності даних система повідомляє про це.

## 3.3.5 Use Case #5: Перегляд історії лікування тварини

## Актори: Ветеринари, користувачі

## Опис: Ветеринари переглядають історію лікування конкретної тварини.

## Вхід: Ідентифікація тварини (QR-код або унікальний ідентифікатор).

## Процес: Ветеринар вводить або сканує QR-код тварини і отримує доступ до історії її лікування.

## Вихід: Детальна історія лікування.

## Обробка помилок: Якщо історія не знайдена, система видає повідомлення про відсутність даних.

## 3.3.6 Use Case #6: Сканування QR-коду тварини

## Актори: Ветеринар, волонтер

## Опис: Сканування QR-коду тварини для отримання інформації про неї, такої як медична історія та стан.

## Вхід: QR-код тварини.

## Процес: Ветеринар або волонтер сканує QR-код, система надає інформацію про тварину.

## Вихід: Дані про тварину (медичні записи, статус).

## Обробка помилок: Якщо код не може бути зчитаний, система відображає повідомлення про помилку.

## 3.3.7 Use Case #7: Додавання нових тварин до системи

## Актори: Адміністратор, користувач

## Опис: Адміністратор або волонтер додає нову тварину до системи для подальшого відстеження її лікування.

## Вхід: Дані про тварину (ім’я, вид, місцезнаходження, опис проблеми).

## Процес: Користувач заповнює форму для додавання нової тварини в систему.

## Вихід: Підтвердження додавання тварини в систему.

## Обробка помилок: Якщо дані неповні або некоректні, система повідомляє про помилку.

## **3.4 Classes / Objects**

## 3.4.2 Клас "Тварина" (Animal) Цей клас використовується для зберігання інформації про тварин, яким потрібна допомога.

## 3.4.2.1 Атрибути

## animal\_id (String): Унікальний ідентифікатор тварини.

## name (String): Ім'я тварини.

## species (String): Вид тварини (наприклад, лис, орел).

## age (Integer): Вік тварини.

## health\_status (String): Стан здоров'я тварини (наприклад, поранена, хвора, здорова).

## location (String): Геолокація тварини на карті (використовується для визначення місця, де вона потребує допомоги).

## qr\_code (String): Код для сканування, який веде до детальної інформації про тварину.

## date\_found (Date): Дата, коли тварина була знайдена або зареєстрована.

## 3.4.2.2 Функції

## registerAnimal(): Реєстрація нової тварини в системі.

## updateHealthStatus(): Оновлення статусу здоров'я тварини (наприклад, після лікування).

## assignRescuer(): Призначення волонтера або ветеринара для порятунку або лікування тварини.

## getAnimalDetails(): Отримання детальної інформації про тварину через QR-код або інтерфейс користувача.

## reportAnimalStatus(): Повідомлення про зміни в статусі тварини (наприклад, вона одужала або померла).

## 3.4.2.3 Посилання на функціональні вимоги / використання випадків

## Створення та перегляд профілю тварини.

## Оновлення стану здоров'я тварини.

## 3.4.3 Клас "Запит на допомогу" (HelpRequest) Цей клас описує запити, які створюються користувачами для інформування про тварин, яким потрібна допомога.

## 3.4.3.1 Атрибути

## request\_id (String): Унікальний ідентифікатор запиту.

## user\_id (String): Ідентифікатор користувача, який створив запит.

## animal\_id (String): Ідентифікатор тварини, для якої створено запит на допомогу.

## message (String): Опис проблеми або потреби тварини (наприклад, поранена лапа, отруєння).

## priority (String): Пріоритетність запиту (високий, середній, низький).

## status (String): Статус запиту (обробляється, виконано, скасовано).

## 3.4.3.2 Функції

## createRequest(): Створення нового запиту на допомогу для конкретної тварини.

## updateRequestStatus(): Оновлення статусу запиту (наприклад, коли допомогу вже надано).

## viewRequestDetails(): Перегляд деталей запиту (тільки для користувачів з відповідними правами).

## assignRescuerToRequest(): Призначення волонтера або ветеринара для виконання запиту.

## 3.4.3.3 Посилання на функціональні вимоги / використання випадків

## Створення та управління запитами на допомогу.

## 3.4.4 Клас "Ветеринар" (Veterinarian) Цей клас описує ветеринарів, які беруть участь у лікуванні тварин.

## 3.4.4.1 Атрибути

## vet\_id (String): Унікальний ідентифікатор ветеринара.

## name (String): Ім'я ветеринара.

## specialization (String): Спеціалізація ветеринара (наприклад, хірург, терапевт).

## location (String): Локація ветеринара для визначення його доступності.

## availability\_status (String): Статус доступності ветеринара (доступний, зайнятий).

## 3.4.4.2 Функції

## viewAssignedRequests(): Перегляд запитів, призначених ветеринару.

## updateAnimalHealthStatus(): Оновлення стану здоров'я тварини після лікування.

## assignToRequest(): Призначення ветеринара для виконання запиту на допомогу.

## viewAnimalHistory(): Перегляд історії лікування конкретної тварини.

## 3.4.4.3 Посилання на функціональні вимоги / використання випадків

## Перегляд запитів на допомогу.

## Оновлення стану здоров'я тварин.

## 3.4.5 Клас "Волонтер" (Volunteer) Цей клас описує волонтерів, які допомагають у порятунку диких тварин.

## 3.4.5.1 Атрибути

## volunteer\_id (String): Унікальний ідентифікатор волонтера.

## name (String): Ім'я волонтера.

## location (String): Локація волонтера для визначення доступності.

## availability\_status (String): Статус доступності волонтера (доступний, зайнятий).

## 3.4.5.2 Функції

## viewAssignedRequests(): Перегляд призначених запитів для волонтера.

## assignToRequest(): Призначення волонтера для виконання запиту на допомогу.

## updateRescueStatus(): Оновлення статусу рятувальної операції (наприклад, допомога надана, рятування завершено).

## 3.4.5.3 Посилання на функціональні вимоги / використання випадків

## Перегляд запитів на допомогу.

## Оновлення статусу порятунку тварин.

## **3.5 Non-Functional Requirements**

## 3.5.1 Продуктивність Продуктивність системи описує, як швидко і ефективно система повинна виконувати свої завдання при збереженні високої якості обслуговування.

## Система повинна обробляти 95% запитів користувачів протягом 2 секунд.

## Мобільний додаток та веб-платформа повинні мати час відгуку не більше 1 секунди для основних операцій (реєстрація, авторизація, перегляд тварини).

## Система повинна забезпечувати обробку не менше 500 запитів на секунду під час пікових навантажень без значного зниження продуктивності.

## 3.5.2 Надійність Надійність системи вказує на її здатність виконувати функції без помилок і збоїв протягом тривалого часу.

## Всі основні функції (реєстрація користувачів, створення запитів на допомогу, перегляд історії тварин) повинні бути доступні 99,9% часу.

## Частота помилок не повинна перевищувати 0,1% для критичних операцій (наприклад, реєстрація тварин або запитів на допомогу).

## Система повинна витримувати стрес-тести з максимальним навантаженням, що дорівнює 1000 одночасним користувачам без збою.

## 3.5.3 Доступність Доступність системи описує її здатність бути доступною для користувачів у будь-який час, з мінімальними перервами на обслуговування.

## Система повинна бути доступна для користувачів 99,9% часу протягом року.

## Плани на обслуговування повинні бути чітко визначені, і система повинна мати не більше 1 години планового часу простою на місяць.

## Час відновлення після аварії (RTO) не повинен перевищувати 1 години, а втрати даних не повинні перевищувати 5 хвилин.

## 3.5.4 Безпека Безпека системи включає в себе заходи, які захищають дані та ресурси від несанкціонованого доступу та порушень.

## Система повинна використовувати HTTPS для всіх з'єднань, що забезпечує шифрування даних під час передачі.

## Усі паролі повинні зберігатися в зашифрованому вигляді за допомогою сучасних алгоритмів хешування (наприклад, bcrypt).

## Всі користувачі повинні проходити двоетапну аутентифікацію для доступу до адміністративної панелі.

## Усі дані, що стосуються тварин, повинні бути захищені відповідно до законодавства про захист персональних даних.

## Система повинна забезпечувати моніторинг і журналювання всіх підозрілих дій.

## 3.5.5 Легкість обслуговування Легкість обслуговування описує здатність системи бути простим у підтримці та оновленні, мінімізуючи витрати на управління та оновлення.

## Час на виявлення та виправлення помилок не повинен перевищувати 4 годин.

## Система повинна підтримувати автоматичне оновлення без перерви в роботі для критичних оновлень без втрати даних.

## Документація повинна бути доступною для розробників і містити чіткі інструкції щодо розгортання, оновлення та обслуговування системи.

## Кожен модуль системи повинен мати детальну документацію для підтримки швидкого тестування та діагностики.

## 3.5.6 Переносимість Переносимість описує здатність системи працювати на різних платформах і в різних середовищах без потреби в значних модифікаціях.

## Система повинна бути сумісною з основними браузерами (Chrome, Firefox, Safari, Edge) для веб-платформи і з Android та iOS для мобільного додатку.

## Для розгортання системи на сервері повинна бути підтримка популярних операційних систем, таких як Windows Server та Linux (Ubuntu).

## Мобільний додаток має бути здатний працювати на всіх сучасних моделях смартфонів і планшетів з Android 10.0+ та iOS 12.0+.

## Для зміни API або інтеграцій з іншими системами не повинно бути необхідності в масштабних переписах коду.

## **3.6 Inverse Requirements**

Система повинна відповідати наступним вимогам:

## Система не повинна включати використання застарілих технологій або платформ, які можуть викликати проблеми з безпекою або сумісністю.

## Не допускається підтримка операційних систем, які більше не підтримуються виробниками або не мають регулярних оновлень безпеки.

## Система не повинна вимагати обов'язкового підключення до Інтернету для виконання базових функцій (наприклад, відправлення запитів на допомогу), щоб забезпечити доступність у умовах обмеженого або відсутнього Інтернету.

## Система не повинна бути надмірно складною для використання кінцевими користувачами, щоб уникнути незадоволення і відмови від використання.

## **3.7 Design Constraints**

Система повинна мати наступні обмеження:

## Обмеження за апаратним забезпеченням: Система повинна бути сумісна з апаратними платформами, які підтримуються на пристроях для користувачів (смартфони, планшети, ПК). Для серверної частини потрібно використовувати платформи, які підтримують Windows Server і Linux (Ubuntu).

## Стандарти безпеки: Програмне забезпечення повинно відповідати вимогам безпеки, таким як GDPR для захисту персональних даних користувачів в ЄС.

## Політики компанії: Розробка повинна враховувати внутрішні політики компанії, пов'язані з зберіганням і обробкою даних, а також вимоги до програмного забезпечення, що використовує сторонні API (наприклад, зовнішні платіжні системи та геолокаційні сервіси).

## Обмеження за часом відгуку: Для веб- і мобільних додатків потрібно забезпечити час відгуку не більше 2 секунд для критичних операцій, таких як створення запису про врятовану тварину або здійснення транзакції.

## Технологічні обмеження: Програмне забезпечення повинно бути розроблене з використанням сучасних технологій і фреймворків, таких як Python, Django для серверної частини, а також React Native для мобільних додатків.

## **3.8 Logical Database Requirements**

## Для системи буде використовуватися база даних, яка повинна відповідати наступним логічним вимогам:

## Формати даних: База даних повинна підтримувати зберігання структурованих даних, таких як інформація про тварин (ім'я, вид, вік, історія захворювань), дані про користувачів (волонтери, ветеринари), а також записи про врятованих тварин, події та транзакції.

## Зберігання даних: Усі дані, що стосуються врятованих тварин та їх лікування, повинні зберігатися в базі даних з урахуванням вимог до термінів зберігання та доступності. Для цього буде потрібно використовувати системи управління базами даних, такі як MariaDB або PostgreSQL.

## Цілісність даних: База даних повинна забезпечувати цілісність даних через використання обмежень і індексів (наприклад, унікальні ідентифікатори для кожної тварини, а також перевірка коректності введених даних). Необхідно передбачити механізми для захисту від втрати даних, такі як регулярні резервні копії та механізми відновлення.

## Послідовність даних: Усі транзакції повинні бути атомарними, консистентними і виконуваними з використанням механізмів транзакційного управління, щоб уникнути помилок у процесі запису даних.

## Підтримка запитів: База даних повинна підтримувати складні запити, такі як пошук тварин за їх місцезнаходженням, датою врятування, станом здоров'я, а також звіти для адміністраторів про статистику порятунку та лікування тварин.

## **3.9 Other Requirements**

Система також повинна підтримувати наступні вимоги:

# Інтерфейс API: Необхідно розробити і надати API для взаємодії з зовнішніми системами, такими як сторонні платіжні сервіси, геолокаційні системи і сервіси для відправки повідомлень користувачам. API повинно бути документоване і забезпечувати безпеку через ключі API і автентифікацію.

# Інтернаціоналізація: Система повинна підтримувати кілька мов, включаючи українську, англійську та, можливо, інші мови для міжнародних користувачів. Це стосується як веб-інтерфейсу, так і мобільних додатків.

# Система повідомлень: Для користувачів (волонтерів, ветеринарів) необхідно розробити систему сповіщень через push-повідомлення або email-повідомлення про нових врятованих тварин, зміни їхнього статусу або надходження нової інформації.

# Мобільні пристрої: Програмне забезпечення повинно бути оптимізовано для роботи на мобільних пристроях з урахуванням обмежень екрану і особливостей інтерфейсу.

# Резервне копіювання: Система повинна підтримувати механізми для регулярного резервного копіювання даних, щоб мінімізувати втрату даних у разі збоїв.

# Тестування і відлагодження: Процес розробки повинен включати регулярне тестування системи для виявлення і усунення помилок, а також оптимізацію продуктивності*.*

# **4. Change Management Process**

## Процес управління змінами в SRS-документі полягає в тому, щоб вчасно виявляти, оцінювати і затверджувати будь-які зміни, що можуть вплинути на функціональність, дизайн або архітектуру системи:

## Ініціація змін: Зміни можуть бути запропоновані будь-яким учасником проекту. Ініціатор подає детальний опис змін, пояснює причини та оцінює потенційний вплив на систему, щоб визначити їх доцільність.

## Оцінка впливу: Команда розробки проводить аналіз пропонованої зміни: як це вплине на існуючі функціональні вимоги, строки виконання та загальну реалізацію проекту. Якщо зміна торкається інших частин системи (наприклад, змін в API чи на серверній частині), проводиться координація з відповідними підрозділами.

## Прийняття рішення: Після оцінки, команда приймає рішення щодо реалізації зміни. Якщо зміни є значними або потребують додаткових ресурсів, питання може бути узгоджене з викладачем або керівником проекту.

## Оновлення документації: У разі затвердження зміни, відповідні розділи SRS оновлюються. Для кожної зміни ведеться обов'язкове маркування (наприклад, через систему контролю версій або ведення історії редагувань), щоб можна було відстежувати зміни в документації.

## Сповіщення про оновлення: Після внесення змін усі учасники проекту отримують повідомлення про оновлені вимоги, щоб бути в курсі останніх змін і впровадити їх у процес розробки.