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

Software Requirements Specification

Version 1.0

01.04.2025

Aranzhyi Ruslan

Backend Developer

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| **01.04.2025** | **Version 1** | **Aranzhyi Ruslan** | **First revision** |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[Introduction](#_Toc105146293) 5

[1.1 Purpose](#_Toc105146294) 5

[1.2 Scope](#_Toc105146300) 5

[1.3 Definitions, Acronyms, and Abbreviations](#_Toc105146301) 5

1.4 References 6

1.5 Overview 6

2. General Description 7

2.1 Product Perspective 7

2.2 Product Functions 7

2.3 User Characteristics 9

2.4 General Constraints 11

2.4.1 Технічні обмеження 11

2.4.2 Операційні обмеження 11

2.5 Assumptions and Dependencies 11

2.5.1 Припущення 11

2.5.2 Залежності 12

3. Specific Requirements 14

3.1 External Interface Requirements 14

3.1.1 User Interfaces 14

3.1.2 Hardware Interfaces 14

3.1.3 Software Interfaces 14

3.1.4 Communications Interfaces 14

3.2 Functional Requirements 15

3.2.1 Додавання інформації про тварину 15

3.2.2 Оновлення та редагування інформації про тварин 16

3.2.3 Пожертви та фінансова допомога 17

3.2.4 Авторизація та аутентифікація користувачів 17

3.2.5 Адміністрування користувачів та системи 18

3.3 Use Cases 19

3.3.1 Use Case #1: Авторизація користувача 19

3.3.2 Use Case #2: Додавання інформації про тварину 20

3.3.3 Use Case #3: Адміністрування користувачів 21

3.3.4 Use Case #4: Перегляд списку тварин 21

3.3.5 Use Case #5: Перегляд інформації про конкретну тварину 22

3.3.6 Use Case #6: Взаємодія з платіжними сервісами 22

3.4 Classes 23

3.5 Non-Functional Requirements 27

3.5.1 Продуктивність та Масштабованість 27

3.5.2 Безпека 27

3.5.3 Юзабіліті (Зручність Використання) 27

3.5.4 Надійність та Відновлення після Збоїв 27

3.5.5 Сумісність 28

3.6 Зворотні Вимоги (Inverse Requirements) 28

3.7 Обмеження Дизайну (Design Constraints) 28

3.8 Логічні Вимоги до Бази Даних (Logical Database Requirements) 29

3.9 Інші Вимоги (Other Requirements) 29

**1. Introduction**

**1.1 Purpose**

Даний застосунок призначений для оптимізації процесів порятунку, лікування, реабілітації та повернення у природне середовище диких тварин, які зазнали травм, були переміщені або опинилися в небезпеці. Він спрямований на підтримку роботи реабілітаційних центрів, ветеринарів, волонтерів та широкого загалу шляхом створення структурованої та доступної цифрової платформи для реєстрації, відстеження та управління порятунком тварин.

**1.2 Scope**

Основні завдання застосунку:

* Оптимізація процесу порятунку – надання можливості користувачам повідомляти про знайдених постраждалих тварин та оперативно зв’язуватися з відповідними спеціалістами.
* Відстеження реабілітації тварин – ведення обліку врятованих тварин, включаючи медичну історію, перебіг лікування та план реабілітації.
* Покращення координації – забезпечення ефективної комунікації між волонтерами, ветеринарами, фахівцями з дикої природи та реабілітаційними центрами.
* Зменшення часу реагування – забезпечення своєчасного втручання шляхом автоматизованих сповіщень та надання необхідної інформації найближчим рятувальним групам.
* Підвищення обізнаності населення – надання освітніх матеріалів щодо правильного поводження з постраждалими тваринами та алгоритмів дій у випадку їх виявлення.

Ця система є особливо актуальною для регіонів, які постраждали від екологічних катастроф, військових дій та інших антропогенних загроз для дикої природи. Завдяки застосунку організації зможуть ефективніше розподіляти ресурси, покращити прозорість даних та збільшити шанси на виживання тварин шляхом забезпечення своєчасної допомоги та належного догляду

**1.3 Definitions, Acronyms, and Abbreviations**

SRS – Software Requirement Specification

UI – User Interface

UX – User Experience

API - Application Programming Interface (Application Programming Interface)

**1.4 References**

(No references)

**1.5 Overview**

Цей документ містить специфікацію вимог до програмного застосунку для порятунку та реабілітації диких тварин. У ньому описані цілі, функціональні та нефункціональні вимоги, а також обмеження, які необхідно враховувати під час розробки.

Документ структурований наступним чином:

* Розділ 1 (Вступ) містить загальну інформацію про призначення застосунку, його основні функції та основні зацікавлені сторони.
* Розділ 2 (Загальний опис) надає огляд перспективи продукту, його ключових особливостей, категорій користувачів та вимог до операційного середовища.
* Розділ 3 (Системні вимоги) деталізує функціональні та нефункціональні вимоги до системи, включаючи конкретні сценарії використання (Use Cases).

Цей документ призначений для всіх членів команди розробки, аналітиків, тестувальників, а також зацікавлених сторін, які беруть участь у реалізації проєкту.

**2. General Description**

**2.1 Product Perspective**

Застосунок для порятунку та реабілітації диких тварин є автономною системою, яка розробляється як новий цифровий інструмент для волонтерських організацій, реабілітаційних центрів, ветеринарів та широкого загалу. Він не є частиною існуючих систем, але може інтегруватися з іншими сервісами, такими як картографічні платформи (Google Maps, OpenStreetMap), платіжні сервіси для благодійних внесків, а також соціальні мережі для поширення інформації про тварин, що потребують допомоги.

Застосунок працюватиме у вигляді мобільної та веб-платформи, що забезпечить доступ до основних функцій із будь-якого пристрою. Основна структура продукту передбачає:

* Базу даних тварин – збереження інформації про врятованих тварин, їхній стан та етапи реабілітації.
* Систему користувачів – можливість реєстрації для волонтерів, ветеринарів та звичайних користувачів з відповідними рівнями доступу.
* Комунікаційну платформу – обмін повідомленнями між волонтерами, реабілітаційними центрами та ветеринарами.
* Інтерактивну карту – для позначення місць, де були знайдені тварини, а також розташування найближчих ветеринарних клінік та реабілітаційних центрів.
* Систему сповіщень – автоматичні повідомлення про нові випадки тварин, що потребують допомоги, та оновлення щодо їхнього стану.

Даний застосунок відрізняється від існуючих рішень тим, що поєднує функції реєстрації, моніторингу, комунікації та фінансової підтримки, що дозволяє ефективніше організовувати роботу з порятунку тварин. Завдяки інтеграції з соціальними мережами та платіжними сервісами, він сприятиме підвищенню рівня залученості громадськості та збиранню коштів на потреби реабілітаційних центрів.

Таким чином, застосунок стане ключовим цифровим рішенням для екологічних та волонтерських організацій, підвищуючи ефективність порятунку та реабілітації диких тварин, особливо у кризових умовах, таких як екологічні катастрофи та зони бойових дій.

**2.2 Product Functions**

Застосунок для порятунку та реабілітації диких тварин надаватиме користувачам низку ключових функцій, спрямованих на оптимізацію процесів виявлення, порятунку, лікування та реабілітації постраждалих тварин. Основні функціональні можливості включають:

1. Реєстрація та управління профілями користувачів

Можливість створення облікового запису для різних категорій користувачів (волонтери, ветеринари, працівники реабілітаційних центрів, звичайні користувачі).

* Авторизація через електронну пошту, номер телефону або соціальні мережі.
* Налаштування особистих даних та рівнів доступу залежно від ролі користувача.

2. Додавання та управління інформацією про тварин

* Форма подачі заявки на порятунок тварини (опис, фото, місце виявлення, стан здоров'я, додаткові коментарі).
* Ведення бази даних врятованих тварин із можливістю оновлення статусу лікування та реабілітації.
* Додавання ветеринарних висновків та рекомендацій щодо догляду за твариною.

3. Взаємодія між користувачами

* Система обміну повідомленнями між волонтерами, ветеринарами та працівниками центрів реабілітації.
* Автоматичні сповіщення про нові заявки на порятунок та оновлення статусу тварин.
* Форум або групові чати для обговорення складних випадків та обміну досвідом.

4. Інтерактивна карта

* Відображення місць, де було знайдено тварин, та позначення найближчих реабілітаційних центрів, ветеринарних клінік та волонтерських груп.
* Фільтрація об’єктів на карті (наприклад, лише ветеринарні клініки або тільки тварини, які потребують допомоги).
* Побудова маршрутів до місць, де перебувають постраждалі тварини.

5. Моніторинг реабілітації та історія тварини

* Відстеження стану тварини з моменту її реєстрації до повернення у природне середовище.
* Доступ до історії лікування, фотографій та звітів про стан тварини.
* Ведення реабілітаційного щоденника ветеринарами та волонтерами.

6. Фінансова підтримка та благодійні внески

* Інтеграція з платіжними сервісами для збору коштів на лікування та реабілітацію тварин.
* Прозорість фінансової допомоги: користувачі можуть переглядати звіти про витрати.
* Сповіщення про активні збори коштів та можливість регулярних донатів.

7. Інформаційний та освітній розділ

* Інструкції для користувачів щодо правильного поводження з постраждалими тваринами.
* Контактна інформація реабілітаційних центрів та спеціалістів.
* Новини про збереження дикої природи, екологічні ініціативи та історії успішної реабілітації тварин.

Завдяки реалізації цих функцій застосунок дозволить забезпечити ефективний механізм порятунку тварин, покращити координацію між фахівцями, підвищити залученість суспільства до проблем дикої природи та сприяти швидкому реагуванню на випадки загрози для тварин.

**2.3 User Characteristics**

Застосунок для порятунку та реабілітації диких тварин передбачає різні рівні доступу для користувачів залежно від їхньої ролі у процесі порятунку та догляду за тваринами. Кожен клас користувачів має певні дозволи та обмеження, що забезпечують ефективне функціонування системи та безпеку даних.

1. Незареєстровані користувачі

Це будь-які відвідувачі застосунку, які не мають облікового запису. Їх можливості обмежені базовими функціями:

* Перегляд публічної інформації (новини, освітні матеріали, контакти реабілітаційних центрів).
* Додавання інформації про знайдену тварину (опис, фото, місце виявлення).
* Отримання загальних рекомендацій щодо поводження із дикими тваринами.

2. Звичайні користувачі

Зареєстровані користувачі, які можуть взаємодіяти з системою, але мають обмежений доступ до редагування даних. Їхні можливості включають:

* Додавання інформації про знайдених тварин.
* Перегляд інформації про тварин та процес їхньої реабілітації.
* Отримання сповіщень про оновлення щодо тварин, якими вони цікавляться.
* Пожертвування коштів на підтримку реабілітації тварин.

3. Волонтери та ветеринари

Користувачі, які беруть активну участь у процесі догляду за тваринами та їхньому лікуванні. Вони мають розширені права у застосунку:

* Редагування даних про тварин (оновлення стану, додавання коментарів та історії лікування).
* Завантаження медичних звітів та рекомендацій щодо догляду.
* Ведення реабілітаційного щоденника тварини.
* Координація з іншими волонтерами та ветеринарами через систему повідомлень.

4. “Локальні” адміністратори

Це працівники ветеринарних клінік та реабілітаційних центрів, які здійснюють управління ветеринарним персоналом у своїх організаціях. Їхні обов’язки включають:

* Додавання, редагування та видалення ветеринарів у межах своєї клініки або реабілітаційного центру.
* Управління медичною документацією в межах установи.
* Моніторинг активності ветеринарів та волонтерів, закріплених за клінікою.

5. “Глобальний” адміністратор

Це користувач із повним контролем над системою, який керує всіма адміністративними процесами та забезпечує стабільну роботу застосунку. Його функції включають:

* Додавання, редагування та видалення локальних адміністраторів.
* Управління загальними налаштуваннями системи та правами доступу.
* Контроль коректності збережених даних та вирішення конфліктних ситуацій.
* Забезпечення безпеки та підтримки роботи сервісу.

Такий розподіл ролей забезпечує структуровану та безпечну систему управління даними про тварин та їхній реабілітаційний процес. Кожен клас користувачів має доступ лише до тих функцій, які необхідні для виконання їхніх завдань, що допомагає уникнути хаосу та неправомірних змін у базі даних.

**2.4 General Constraints**

Розробка та використання застосунку підпорядковується наступним обмеженням:

**2.4.1 Технічні обмеження**

Підтримка платформ: Веб-версія застосунку має коректно працювати у браузерах Google Chrome, Mozilla Firefox, Microsoft Edge (останні стабільні версії). Мобільний застосунок підтримує операційні системи Android 8.0+.

Сумісність із серверним середовищем: Бекенд використовує Python 3.11+, Django/Flask, MariaDB, Redis та працює в середовищі Docker. Веб-сервер Nginx використовується для розподілу навантаження та обслуговування запитів.

Обмеження на розмір даних: Завантажені зображення тварин обмежені 5 МБ на файл. Максимальний обсяг текстових полів (опис тварини, історія хвороби тощо) – 10000 символів.

Продуктивність: Час відповіді сервера на стандартні запити не повинен перевищувати 2-3 секунди при середньому навантаженні.

**2.4.2 Операційні обмеження**

Доступ до Інтернету: Для коректної роботи застосунку необхідне постійне підключення до мережі. Відсутність підключення обмежує функціонал, дозволяючи переглядати лише кешовані дані без можливості оновлення.

Безпека та конфіденційність: Всі дані користувачів та тварин повинні відповідати стандартам захисту конфіденційності (наприклад, GDPR). Застосунок використовує SSL/TLS для безпечного передавання даних.

Правові обмеження: Усі зібрані дані можуть використовуватися лише в межах діяльності волонтерських організацій і не можуть передаватися третім особам без згоди користувачів. Реєстрація ветеринарів та волонтерів передбачає перевірку документів для підтвердження їхнього статусу.

Обмеження на інтеграцію з зовнішніми сервісами: Використання API (Google Maps, платіжні сервіси, соціальні мережі) обмежується їхніми політиками та лімітами запитів.

**2.5 Assumptions and Dependencies**

**2.5.1 Припущення**

Для успішної роботи застосунку передбачається виконання наступних умов:

* Доступ до мережі Інтернет
* Застосунок передбачає постійне підключення до Інтернету для взаємодії між користувачами, сервером та базою даних.
* Сумісність з сучасними веб-браузерами та мобільними пристроями
* Веб-версія застосунку повинна коректно працювати в останніх версіях браузерів Google Chrome, Mozilla Firefox, Microsoft Edge.
* Мобільний застосунок підтримуватиме Android 8.0+.
* Коректна робота серверної інфраструктури
* Сервери, на яких розгорнуто застосунок, повинні мати необхідні ресурси для підтримки навантаження користувачів.
* Регулярне оновлення серверного ПЗ та бази даних є необхідним для забезпечення стабільності та безпеки системи.
* Актуальність даних у системі: Волонтери та ветеринари регулярно оновлюватимуть дані про стан тварин, щоб користувачі отримували лише актуальну інформацію.
* Безпека та захист персональних даних: Вся інформація про користувачів та тварин повинна оброблятися згідно з політикою конфіденційності та вимогами GDPR. Реалізована система автентифікації та авторизації гарантуватиме захист персональних даних.

**2.5.2 Залежності**

Робота застосунку залежить від наступних компонентів і сервісів:

Програмне середовище

* Мова програмування: Python 3.11+
* Фреймворк для бекенду: FastAPI
* База даних: MariaDB (збереження інформації про тварин, користувачів, історію лікувань)
* Кешування та обробка черг: Redis
* Веб-сервер: Nginx
* Контейнеризація: Docker

Зовнішні сервіси та API

* Карти та геолокація: Google Maps API або OpenStreetMap для визначення місцезнаходження тварин і побудови маршрутів до найближчих центрів реабілітації.
* Платіжні системи: Інтеграція з PayPal, Stripe або іншими сервісами для збору пожертвувань.
* Системи автентифікації: OAuth 2.0 (Google, Facebook) для швидкої реєстрації та входу.
* Сервіси повідомлень: Firebase Cloud Messaging (для push-сповіщень), SMTP (для електронної пошти).

Серверна інфраструктура

* Хмарні сервіси: AWS, Google Cloud, Azure або власні сервери з підтримкою Docker.

Мобільні платформи

* Розробка для Android: Kotlin або Java

Кросплатформені технології: React Native або Flutter (за потреби)

Безпека та шифрування

* Використання SSL/TLS для шифрування переданих даних між клієнтами та сервером.
* Зберігання паролів у базі даних із використанням алгоритмів хешування (наприклад, bcrypt).

**3. Specific Requirements**

**3.1 External Interface Requirements**

**3.1.1 User Interfaces**

**UI-1**: Користувацький інтерфейс буде доступний через настільні та мобільні браузери, забезпечуючи адаптивний дизайн для різних розмірів екранів.

**UI-2**: Інтерфейс взаємодії з тваринами дозволятиме користувачам:

* Шукати інформацію про врятованих тварин.
* Додавати дані про знайдених або травмованих тварин (для зареєстрованих користувачів).
* Переглядати історію лікування та статус реабілітації (для ветеринарів та волонтерів).
* Робити фінансові пожертви на підтримку тварин.

**UI-3**: Панель адміністратора для волонтерів і ветеринарів дозволятиме переглядати, змінювати й оновлювати інформацію про тварин, історію їхнього лікування та реабілітації.

**UI-4**: Підтримка багатомовного інтерфейсу для міжнародних користувачів.

**3.1.2 Hardware Interfaces**

**HW-1**: Система буде сумісною зі стандартними настільними та мобільними пристроями (наприклад, ПК, смартфонами, планшетами), що працюють під управлінням Windows, macOS, Linux, Android та iOS.

**HW-2**: Для доступу до системи не потрібно спеціалізованого апаратного забезпечення.

**3.1.3 Software Interfaces**

**SW-1**: Система буде інтегруватися з платіжними шлюзами (наприклад, PayPal, Stripe) для обробки фінансових пожертв.

**SW-2**: Інтеграція з картографічними сервісами (Google Maps, OpenStreetMap) для відображення місць розташування знайдених тварин та реабілітаційних центрів.

**SW-3**: Система буде використовувати API для отримання оновлень про стан тварин, що перебувають у реабілітаційних центрах.

**SW-4**: Підтримка взаємодії з сервісами автентифікації (OAuth 2.0, Google/Facebook API) для швидкого входу користувачів.

**3.1.4 Communications Interfaces**

**COM-1**: Обмін даними з зовнішніми сервісами здійснюватиметься через RESTful API із використанням HTTPS для безпечної передачі даних.

**COM-2**: Надсилання електронних сповіщень про оновлення статусу тварин, підтвердження реєстрації та інформаційні повідомлення через SMTP-сервери.

**COM-3**: Впровадження push-сповіщень через Firebase Cloud Messaging для швидкого інформування волонтерів і ветеринарів про нові випадки порятунку тварин.

**3.2 Functional Requirements**

**3.2.1 Додавання інформації про тварину**

**3.2.1.1 Introduction**

Функція додавання інформації про тварину дозволить користувачам (зареєстрованим і незареєстрованим) вводити дані про знайдених або врятованих тварин, щоб волонтери та ветеринари могли швидко реагувати на випадки порятунку.

**3.2.1.2 Inputs**

**IN-1**: Вид тварини (собака, кіт, дика тварина тощо).

**IN-2**: Локація, де було знайдено тварину (GPS-координати або введена вручну адреса).

**IN-3**: Стан тварини (здоровий, травмований, потребує допомоги тощо).

**IN-4**: Контактні дані особи, яка повідомляє про тварину.

**IN-5**: Фотографії або відео знайденої тварини (за можливості).

**3.2.1.3 Processing**

**PR-1**: Система перевірить правильність введених даних (обов’язкові поля).

**PR-2**: Дані будуть збережені у базі даних і автоматично надіслані до відповідного волонтерського або ветеринарного центру, якщо такий визначено за локацією.

**PR-3**: У разі недостатньої інформації система попросить користувача додати більше деталей.

**3.2.1.4 Outputs**

**OUT-1**: Сторінка підтвердження успішного додавання інформації про тварину.

**OUT-2**: Повідомлення електронною поштою або push-сповіщення волонтерам і ветеринарам у відповідному регіоні.

**3.2.1.5 Error handling**

**ER-1**: Якщо користувач не заповнив обов’язкові поля, система виведе повідомлення про необхідність введення всіх даних.

**ER-2**: Якщо немає підключення до мережі, система збереже дані локально та дозволить надіслати їх пізніше.

**ER-3**: Якщо сервер недоступний, система відобразить повідомлення про тимчасову недоступність сервісу.

**3.2.2 Оновлення та редагування інформації про тварин**

**3.2.2.1 Introduction**

Функція оновлення дозволяє ветеринарам і волонтерам змінювати інформацію про тварин, додавати історію лікування, змінювати статус та оновлювати фото/відео.

**3.2.2.2 Inputs**

**IN-1**: Унікальний ідентифікатор тварини в системі.

**IN-2**: Новий статус тварини (лікується, готовий до випуску, переданий в притулок тощо).

**IN-3**: Додаткові медичні записи (діагнози, результати аналізів, лікування).

**IN-4**: Нові фотографії або відео.

**3.2.2.3 Processing**

**PR-1**: Система перевірить права доступу користувача (оновлювати можуть тільки ветеринари та волонтери).

**PR-2**: Дані будуть оновлені у базі та синхронізовані між усіма пристроями.

**PR-3**: У разі змін статусу тварини система сповістить відповідальних осіб.

**3.2.2.4 Outputs**

**OUT-1**: Оновлена сторінка тварини з відображенням нових даних.

**OUT-2**: Push-сповіщення для волонтерів та ветеринарів, яких стосується оновлення.

**3.2.2.5 Error handling**

**ER-1**: Якщо користувач не має прав на редагування, система відобразить відповідне повідомлення.

**ER-2**: Якщо сервер недоступний, система запропонує повторити оновлення пізніше.

**3.2.3 Пожертви та фінансова допомога**

**3.2.3.1 Introduction**

Функція надання фінансової допомоги дозволяє користувачам здійснювати благодійні внески на підтримку тварин і реабілітаційних центрів.

**3.2.3.2 Inputs**

**IN-1**: Сума пожертви.

**IN-2**: Вибір реабілітаційного центру (опціонально).

**IN-3**: Дані платіжної картки або електронного гаманця.

**3.2.3.3 Processing**

**PR-1**: Підключення до платіжного шлюзу (Stripe, PayPal тощо).

**PR-2**: Перевірка валідності платіжних даних.

**PR-3**: Формування фінансового звіту про пожертву.

**3.2.3.4 Outputs**

**OUT-1**: Підтвердження успішного платежу.

**OUT-2**: Електронний лист із квитанцією про пожертву.

**OUT-3**: Оновлення загального списку зборів на сайті.

**3.2.3.5 Error handling**

**ER-1**: Якщо платіж не пройшов, система виведе відповідне повідомлення.

**ER-2**: Якщо сервер платіжного шлюзу недоступний, система запропонує повторити операцію пізніше.

**3.2.4 Авторизація та аутентифікація користувачів**

**3.2.4.1 Introduction**

Функція авторизації та аутентифікації дозволяє користувачам безпечно входити в систему, керувати своїми даними та виконувати дії відповідно до своїх ролей.

**3.2.4.2 Inputs**

**IN-1**: Логін (електронна пошта або ім'я користувача).

**IN-2**: Пароль.

**IN-3**: Двофакторний код (якщо увімкнено 2FA).

**IN-4**: Токен авторизації (для повторних входів).

**3.2.4.3 Processing**

**PR-1**: Система перевіряє відповідність введеного логіну та пароля збереженим даним.

**PR-2**: Якщо активний 2FA, користувач має ввести код підтвердження з SMS або додатку аутентифікації.

**PR-3**: Успішний вхід створює сеансовий токен, що дозволяє користувачеві взаємодіяти із системою без повторного введення логіну.

**PR-4**: Якщо спроби входу неуспішні більше ніж 5 разів, система тимчасово блокує обліковий запис.

**PR-5**: Використовується захищене з'єднання (HTTPS) та шифрування паролів.

**3.2.4.4 Outputs**

**OUT-1**: Успішний вхід перенаправляє користувача на головну сторінку або особистий кабінет.

**OUT-2**: У разі невдалого входу користувач отримує повідомлення про неправильний логін або пароль.

**OUT-3**: Якщо увімкнено 2FA, система запитує код підтвердження.

**OUT-4**: Якщо акаунт заблоковано, виводиться відповідне попередження з інструкціями щодо розблокування.

**3.2.4.5 Error handling**

**ER-1**: Якщо сервер авторизації недоступний, система виведе повідомлення про тимчасову проблему.

**ER-2**: Якщо введено неправильний пароль більше 5 разів, система тимчасово заблокує вхід.

**ER-3**: Якщо користувач не підтвердив електронну пошту, система запропонує повторно відправити код підтвердження.

**3.2.5 Адміністрування користувачів та системи**

**3.2.5.1 Introduction**

Функція адміністрування дозволяє глобальному адміністратору керувати користувачами, призначати ролі, редагувати налаштування безпеки та перевіряти роботу системи.

**3.2.5.2 Inputs**

**IN-1**: Запит на зміну ролі користувача (локальний адміністратор, волонтер, ветеринар).

**IN-2**: Запит на блокування/розблокування облікового запису.

**IN-3**: Запит на перегляд журналу подій.

**IN-4**: Запит на оновлення параметрів безпеки (пароль, 2FA, права доступу).

**3.2.5.3 Processing**

**PR-1**: Глобальний адміністратор може створювати, редагувати та видаляти локальних адміністраторів.

**PR-2**: Локальні адміністратори можуть додавати ветеринарів і волонтерів для конкретних центрів.

**PR-3**: Система веде журнал подій, де записуються всі дії адміністраторів і користувачів.

**PR-4**: Якщо змінюються критичні налаштування безпеки (наприклад, пароль адміністратора), система запитує повторне підтвердження особи.

**3.2.5.4 Outputs**

**OUT-1**: Список активних користувачів та їхні ролі.

**OUT-2**: Журнал подій із фільтрацією за діями (авторизація, зміна ролей, видалення записів).

**OUT-3**: Повідомлення про успішне оновлення налаштувань безпеки.

**OUT-4**: Попередження у разі спроби несанкціонованого доступу.

**3.2.5.5 Error handling**

**ER-1**: Якщо адміністратор не має прав для виконання певної дії, система видасть відповідне повідомлення.

**ER-2**: Якщо сервер недоступний, зміни не будуть застосовані, і система запропонує повторити пізніше.

**ER-3**: Якщо користувач намагається видалити глобального адміністратора, система заблокує таку дію.

**3.3 Use Cases**

**3.3.1 Use Case #1: Авторизація користувача**

**Actors**: Незареєстрований користувач, Система аутентифікації

**Preconditions**: Користувач має доступ до інтернету та відкрив сторінку входу в систему.

**Basic Flow:**

* Користувач відкриває сторінку входу.
* Користувач вводить логін (електронну пошту або ім'я користувача) та пароль.
* Система перевіряє правильність введених даних (згідно з 3.2.4.3).
* Якщо активовано двофакторну аутентифікацію, система надсилає код підтвердження.
* Користувач вводить код підтвердження.
* Система створює сесію та дозволяє користувачу увійти в систему

**Alternative Flow (Помилка входу):**

* Якщо користувач ввів неправильний пароль, система видає повідомлення про помилку.
* Якщо кількість невдалих спроб перевищує 5, система тимчасово блокує обліковий запис.

**Postconditions**: Користувач успішно входить у систему та може використовувати функціонал відповідно до своєї ролі. У разі невдачі користувач отримує повідомлення про помилку або необхідність розблокування облікового запису.

**3.3.2 Use Case #2: Додавання інформації про тварину**

**Actors**: Волонтер, Ветеринар, Незареєстрований користувач

**Preconditions**: Користувач має доступ до інтернету та відкрив форму додавання тварини.

**Basic Flow:**

* Користувач відкриває сторінку додавання тварини.
* Вводить інформацію (ім'я, вид, стан здоров'я, місцезнаходження тощо).
* Додає фото або відео (за бажанням).
* Система перевіряє правильність введених даних (наприклад, відсутність порожніх обов’язкових полів).
* Система зберігає інформацію та відображає її у списку знайдених тварин.

**Alternative Flow (Помилка введення даних):**

* Якщо користувач не заповнив обов’язкові поля, система повідомляє про необхідність їх заповнення.
* Якщо фото або відео завантажене у непідтримуваному форматі, система повідомляє про це.

**Postconditions**:

* Інформація про тварину успішно збережена в системі та доступна для перегляду.
* У разі помилки система видає відповідне повідомлення.

**3.3.3 Use Case #3: Адміністрування користувачів**

**Actors**: Глобальний адміністратор, Локальний адміністратор

**Preconditions**: Адміністратор увійшов у систему з відповідними правами доступу.

**Basic Flow:**

* Адміністратор відкриває панель керування користувачами.
* Обирає користувача зі списку.
* Обирає необхідну дію (зміна ролі, блокування, видалення).
* Система запитує підтвердження дії.
* Після підтвердження система оновлює дані та зберігає зміни.

**Alternative Flow (Помилка виконання операції):**

* Якщо адміністратор не має прав для виконання дії, система видає повідомлення про відмову.
* Якщо сервер адміністрації тимчасово недоступний, система повідомляє про неможливість виконання операції.

**Postconditions**:

* Дані користувача успішно оновлені.
* У разі помилки система видає відповідне повідомлення.

**3.3.4 Use Case #4: Перегляд списку тварин**

**Actors**: Незареєстрований користувач, Зареєстрований користувач, Волонтер, Ветеринар

**Preconditions**:

* Користувач має доступ до інтернету.
* У базі даних є зареєстровані тварини.

**Basic Flow:**

* Користувач відкриває сторінку зі списком тварин.
* Система відображає список доступних тварин з фото, ім’ям, статусом та місцем перебування.
* Користувач може скористатися фільтрами для пошуку конкретної тварини (вид, статус, місцезнаходження тощо).

**Alternative Flow (Помилка завантаження даних):**

* Якщо база даних тимчасово недоступна, система повідомляє про помилку.
* Якщо список тварин порожній, система повідомляє, що наразі немає доступних даних.

**Postconditions**:

* Користувач може переглянути список тварин та обрати конкретну тварину для детальнішого перегляду.

**3.3.5 Use Case #5: Перегляд інформації про конкретну тварину**

**Actors**: Незареєстрований користувач, Зареєстрований користувач, Волонтер, Ветеринар

**Preconditions**:

* Користувач має доступ до інтернету.
* У базі даних є інформація про тварину.

**Basic Flow:**

* Користувач натискає на картку тварини у списку.
* Система відкриває сторінку з детальною інформацією про тварину, включаючи стан здоров’я, історію лікування (для ветеринарів), дату та місце знаходження.
* Користувач може побачити контактні дані відповідального волонтера або ветеринара (якщо дозволено).

**Alternative Flow (Помилка завантаження даних):**

* Якщо інформація про тварину тимчасово недоступна, система повідомляє про помилку.

**Postconditions**:

* Користувач отримує доступ до детальної інформації про тварину.

**3.3.6 Use Case #6: Взаємодія з платіжними сервісами**

**Actors**: Зареєстрований користувач, Волонтер, Платіжна система

**Preconditions**:

* Користувач має доступ до інтернету.
* Користувач має бажання зробити фінансову допомогу на користь реабілітаційного центру або конкретної тварини.

**Basic Flow:**

* Користувач відкриває розділ «Допомогти» або натискає кнопку «Зробити внесок» на сторінці тварини.
* Вибирає суму пожертви та платіжний метод.
* Система перенаправляє користувача до платіжного сервісу (PayPal, Stripe або інший).
* Користувач вводить платіжні дані та підтверджує оплату.
* Система отримує підтвердження успішного платежу.
* Користувач отримує квитанцію на електронну пошту.

**Alternative Flow (Помилка платежу):**

* Якщо платіж не пройшов, система повідомляє користувача та пропонує повторити спробу або вибрати інший метод оплати.

**Postconditions**:

* Платіж успішно завершений, і гроші надійшли до реабілітаційного центру або волонтерської організації.

**3.4 Classes**

**3.4.1 Class: Animal**

Цей клас представляє тварину, зареєстровану в системі.

**3.4.1.1 Attributes:**

* animalID: Унікальний ідентифікатор тварини (int).
* name: Ім'я тварини (string).
* species: Вид тварини (string).
* breed: Порода (string).
* age: Вік тварини (int).
* status: Поточний статус тварини (наприклад, "знайдено", "у клініці", "усиновлено") (string).
* location: Місцезнаходження тварини (string).
* healthHistory: Історія хвороб та лікування (list).
* photo: Посилання на фото тварини (string).
* volunteerID: Ідентифікатор волонтера, відповідального за тварину (int).

**3.4.1.2 Functions:**

* updateStatus(newStatus): Оновлює статус тварини.
* addHealthRecord(record): Додає запис до історії хвороб.
* updateLocation(newLocation): Оновлює місцезнаходження тварини.

**3.4.2 Class: User**

**3.4.2.1 Attributes:**

* userID: Унікальний ідентифікатор користувача (int).
* username: Ім'я користувача (string).
* email: Електронна пошта (string).
* passwordHash: Хеш пароля (string).
* role: Роль користувача (наприклад, "волонтер", "ветеринар", "адміністратор") (string).
* registrationDate: Дата реєстрації (datetime).

**3.4.2.2 Functions:**

* authenticate(password): Перевіряє правильність введеного пароля.
* updateEmail(newEmail): Оновлює електронну пошту.
* changePassword(newPassword): Змінює пароль користувача.

**3.4.3 Class: Volunteer**

Цей клас є підкласом User і представляє волонтера.

**3.4.3.1 Attributes:**

* assignedAnimals: Список тварин, за які відповідає волонтер (list).

**3.4.3.2 Functions:**

* addAnimal(animalID): Додає тварину під опіку волонтера.
* removeAnimal(animalID): Видаляє тварину з опіки волонтера.

**3.4.4 Class: Veterinarian**

Цей клас є підкласом User і представляє ветеринара.

**3.4.4.1 Attributes:**

* clinicID: Ідентифікатор клініки, де працює ветеринар (int).

**3.4.4.2 Functions:**

* addHealthRecord(animalID, record): Додає запис до історії хвороб тварини.

**3.4.5 Class: Admin**

* Цей клас є підкласом User і представляє адміністратора.

**3.4.5.1 Attributes:**

* permissions: Рівень прав адміністратора (string).

**3.4.5.2 Functions:**

* manageUsers(userID, action): Додає, змінює або видаляє користувачів.
* assignRole(userID, role): Призначає користувачеві нову роль.

**3.4.6 Class: Payment**

Цей клас представляє платіж у системі.

**3.4.6.1 Attributes:**

* paymentID: Унікальний ідентифікатор платежу (int).
* userID: Ідентифікатор користувача, який зробив платіж (int).
* amount: Сума платежу (float).
* currency: Валюта платежу (string).
* paymentMethod: Використаний метод оплати (наприклад, "PayPal", "кредитна картка") (string).
* status: Статус платежу ("успішний", "відхилений") (string).
* timestamp: Час здійснення платежу (datetime).

**3.4.6.2 Functions:**

* processPayment(): Виконує платіж через інтегрований платіжний сервіс.
* refund(): Ініціює повернення коштів.
* getPaymentStatus(): Отримує поточний статус платежу.

**3.4.7 Class: Marker**

Цей клас представляє мітку на карті (наприклад, місцезнаходження знайденої тварини, притулку, ветеринарної клініки або волонтера).

**3.4.7.1 Attributes:**

* markerID: Унікальний ідентифікатор мітки (int).
* latitude: Широта (float).
* longitude: Довгота (float).
* markerType: Тип мітки (наприклад, "тварина", "притулок", "ветеринар") (string).
* description: Опис мітки (string).
* relatedEntityID: Ідентифікатор об'єкта, пов'язаного з міткою (наприклад, animalID або shelterID) (int).
* timestamp: Час останнього оновлення мітки (datetime).

**3.4.7.2 Functions:**

* updateLocation(newLatitude, newLongitude): Оновлює координати мітки.
* updateDescription(newDescription): Оновлює опис мітки.
* setMarkerType(newType): Змінює тип мітки.

**3.4.8 Class: Shelter**

Цей клас представляє притулок або реабілітаційний центр для тварин.

**3.4.8.1 Attributes:**

* shelterID: Унікальний ідентифікатор притулку (int).
* name: Назва притулку (string).
* location: Координати притулку (tuple: latitude, longitude).
* capacity: Максимальна кількість тварин у притулку (int).
* currentAnimals: Список тварин, які перебувають у притулку (list).
* contactInfo: Контактна інформація (string).

**3.4.8.2 Functions:**

* addAnimal(animalID): Додає тварину до притулку.
* removeAnimal(animalID): Видаляє тварину з притулку.
* updateCapacity(newCapacity): Оновлює максимальну місткість притулку.

**3.5 Non-Functional Requirements**

**3.5.1 Продуктивність та Масштабованість**

* **NFR-1**: Система повинна обробляти не менше 1000 одночасних запитів без значного зниження продуктивності.
* **NFR-2**: Час відповіді сервера не повинен перевищувати 2 секунд при стандартних запитах користувача.
* **NFR-3**: База даних повинна підтримувати зберігання інформації про мінімум 1 мільйон тварин і пов’язаних записів.
* **NFR-4**: Система повинна бути здатною до горизонтального масштабування через контейнеризацію в Docker.

**3.5.2 Безпека**

* **NFR-5**: Усі дані користувачів, зокрема особиста інформація та паролі, повинні зберігатися в зашифрованому вигляді (наприклад, використання bcrypt для паролів).
* **NFR-6**: Доступ до адміністративних функцій повинен бути обмежений ролями та перевірятися через систему аутентифікації та авторизації (OAuth 2.0 або JWT).
* **NFR-7**: Передача даних між клієнтом та сервером повинна здійснюватися через HTTPS.
* **NFR-8**: Усі запити до API повинні перевірятися на наявність потенційних атак, таких як SQL-ін’єкції, XSS і CSRF.

**3.5.3 Юзабіліті (Зручність Використання)**

* **NFR-9**: Інтерфейс користувача повинен бути інтуїтивно зрозумілим і відповідати сучасним стандартам UI/UX.
* **NFR-10**: Веб-версія застосунку повинна підтримувати адаптивний дизайн для коректного відображення на мобільних пристроях та планшетах.
* **NFR-11**: Основні функції (додавання інформації про тварин, перегляд мапи, пошук волонтерів) повинні бути доступними в не більше ніж 3 кліки від головної сторінки.

**3.5.4 Надійність та Відновлення після Збоїв**

* **NFR-12**: Система повинна забезпечувати 99.9% часу безвідмовної роботи (uptime).
* **NFR-13**: У разі збою система повинна мати механізми автоматичного резервного копіювання бази даних з інтервалом у 24 години.
* **NFR-14**: Усі критичні процеси (авторизація, фінансові транзакції) повинні мати механізми аварійного відновлення та логування.
* **NFR-15**: У разі втрати з’єднання з сервером користувач повинен отримати відповідне повідомлення і мати змогу повторити запит.
* **3.5.5 Сумісність**
* **NFR-16**: Система повинна коректно працювати в актуальних версіях браузерів Google Chrome, Mozilla Firefox, Safari та Microsoft Edge.
* **NFR-17**: Бекенд-система повинна підтримувати API для інтеграції з іншими сервісами через REST.
* **NFR-18**: Програма повинна працювати на будь-якій платформі, яка підтримує Docker.
* **NFR-19**: Використання бази даних MariaDB, кешування через Redis і веб-сервера Nginx повинно бути оптимізовано для роботи з великим навантаженням.

**3.6 Зворотні Вимоги (Inverse Requirements)**

Цей розділ описує функціональність, яка не буде реалізована у системі.

* **INV-1**: Система не підтримуватиме офлайн-режим роботи без доступу до Інтернету.
* **INV-2**: Не передбачена можливість прямого спілкування користувачів у вигляді чату чи форуму.
* **INV-3**: Система не братиме на себе логістику перевезення тварин між локаціями.
* **INV-4**: Не буде підтримки телефонного зв’язку чи SMS-повідомлень для повідомлень про тварин.
* **INV-5**: Розширені функції аналізу стану здоров’я тварин (на основі штучного інтелекту або медичних алгоритмів) не будуть реалізовані на початковому етапі.
* **INV-6**: Автоматичне визначення місцеперебування тварини за зображенням або відео не передбачене.

**3.7 Обмеження Дизайну (Design Constraints)**

Цей розділ визначає обмеження, які впливають на архітектуру та реалізацію системи.

* **DC-1**: Система має використовувати Python 3.11 для серверної частини.
* **DC-2**: База даних повинна бути реалізована на MariaDB з Redis для кешування.
* **DC-3**: Усі серверні компоненти повинні працювати в Docker-контейнерах.
* **DC-4**: Інтерфейс повинен бути адаптивним та відповідати сучасним UI/UX стандартам.
* **DC-5**: Використання HTTPS є обов’язковим для всіх зовнішніх запитів.
* **DC-6**: Взаємодія між модулями повинна відбуватися через REST API.

**3.8 Логічні Вимоги до Бази Даних (Logical Database Requirements)**

Опис вимог до бази даних, її структури та операцій.

* **LDB-1**: Усі дані про тварин (ім’я, вид, стан здоров’я, місцеперебування) мають зберігатися в реляційній базі даних (MariaDB).
* **LDB-2**: Дані користувачів (ім’я, контактна інформація, права доступу) повинні бути зберігатися в зашифрованому вигляді.
* **LDB-3**: Історія змін профілів тварин повинна логуватися для аудиту.
* **LDB-4**: Доступ до чутливих даних повинен регулюватися на рівні ролей користувачів (ветеринари, волонтери, адміністратори).
* **LDB-5**: Логування всіх запитів до бази даних для безпеки та аналізу (Redis може використовуватися для кешування).
* **LDB-6**: Дані про транзакції та фінансову допомогу повинні відповідати вимогам бухгалтерського обліку.
* **LDB-7**: Інформація про тварин, додана волонтерами, повинна бути перевірена адміністраторами перед публікацією.
* **LDB-8**: База повинна підтримувати резервне копіювання кожні 24 години для запобігання втраті даних.

**3.9 Інші Вимоги (Other Requirements)**

Додаткові вимоги, які не потрапили в інші категорії.

* **OR-1**: Платформа повинна мати можливість розширення функціоналу шляхом додавання нових модулів (наприклад, підтримка нових видів тварин або автоматичне визначення місцеперебування).
* **OR-2**: Система повинна підтримувати багатомовність (українська, англійська, можливо, інші мови).
* **OR-3**: Повідомлення користувачам (про новини, оновлення статусу тварин) мають надсилатися через електронну пошту та push-сповіщення.
* **OR-4**: У разі виходу нових оновлень користувачі повинні отримувати інформацію про нові можливості та зміни в інтерфейсі.
* **OR-5**: Документація API повинна бути доступна для розробників через OpenAPI або Swagger.