P O L I T E C H N I K A Ś W I Ę T O K R Z Y S K A
Wydział Elektrotechniki, Automatyki i Informatyki

Programowanie defensywne - projekt		
Temat: System zarządzania kliniką weterynaryjną		
I Kamień milowy	Autorzy: Marek Supierz Andrzej Mysior	Grupa: **1ID24A**
Ocena:		Data oddania: **21.03.2025**

Harmonogram projektu

Tydzień 1 Analiza wymagań i projektowanie architektury

Zadania

- Zebranie wymagań funkcjonalnych (np. rejestracja użytkowników, zarządzanie wizytami, historia medyczna) oraz niefunkcjonalnych (bezpieczeństwo, responsywność).
- Określenie ról użytkowników (lekarz, recepcjonista, właściciel zwierzęcia) oraz ich uprawnień.
- Wybór technologii: Python, FastAPI, SQLAlchemy, PostgreSQL, Web3.py, PyQt lub React.
- Sporządzenie wstępnych diagramów architektury systemu (np. diagram komponentów, diagram UML klas).
- Przygotowanie planu kamieni milowych (podział na moduły: backend, frontend, blockchain, moduł finansowy).
- Konfiguracja repozytorium Git oraz środowiska programistycznego (utworzenie wirtualnego środowiska, instalacja podstawowych pakietów).

Tydzień 2 Projektowanie bazy danych i konfiguracja backendu

Zadania

- Zaprojektowanie szczegółowego schematu bazy danych: tabele dla użytkowników, wizyt, danych medycznych itp.
- Utworzenie projektu w FastAPI i skonfigurowanie podstawowego szkieletu aplikacji.
- Implementacja modeli danych w SQLAlchemy zgodnie z zaprojektowanym schematem.
- Konfiguracja migracji bazy danych (np. przy użyciu Alembic).
- Napisanie pierwszych testów sprawdzających połączenie z bazą danych.
- Dokumentacja projektu bazy danych (diagram ERD, opis tabel).

Tydzień 3 Moduł uwierzytelniania i rejestracji użytkowników

Zadania

- Utworzenie endpointów REST API do rejestracji nowych użytkowników.
- Implementacja logowania użytkowników przy użyciu JWT.
- Integracja z TOTP, korzystając z biblioteki pyotp (generowanie i weryfikacja kodów).
- Przygotowanie testów jednostkowych dla modułu uwierzytelniania (sprawdzenie poprawności rejestracji, logowania, weryfikacji TOTP).
- Konfiguracja dokumentacji API (np. z wykorzystaniem Swagger/OpenAPI).

Tydzień 4 Moduł zarządzania wizytami i danymi zwierząt

Zadania

- Implementacja operacji CRUD dla zarządzania wizytami (dodawanie, edycja, usuwanie, wyświetlanie wizyt).
- Implementacja operacji CRUD dla zarządzania danymi zwierząt.
- Dodanie walidacji danych za pomocą Pydantic.
- Testowanie endpointów przy użyciu narzędzi takich jak Postman lub Insomnia.
- Aktualizacja dokumentacji API, opisująca nowe endpointy.

Tydzień 5 Podstawowy interfejs użytkownika (frontend)

Zadania

- Wybór technologii frontendowej – PyQt dla aplikacji desktop lub React dla aplikacji web.
- Utworzenie pierwszego szkicu interfejsu: ekran logowania, rejestracji oraz panel główny.
- Połączenie interfejsu z backendem poprzez REST API.
- Implementacja podstawowej obsługi błędów (wyświetlanie komunikatów o błędach logowania, walidacji danych).
- Przeprowadzenie pierwszych testów integracji między frontendem a backendem.

Tydzień 6 Moduł medyczny – historia leczenia

Zadania

- Implementacja CRUD dla wpisów medycznych (dodawanie, edycja, usuwanie, przeglądanie historii leczenia).
- Powiązanie danych medycznych z konkretnymi wizytami i zwierzętami.
- Dodanie mechanizmów walidacji danych medycznych.
- Napisanie testów jednostkowych dla operacji CRUD na danych medycznych.
- Stworzenie podstawowego interfejsu do przeglądania historii leczenia (w GUI lub jako widok w aplikacji web).

Tydzień 7 Testy jednostkowe i analiza statyczna kodu

Zadania

- Konfiguracja narzędzi do analizy statycznej (np. SonarQube, flake8, pylint) i analiza kodu.
- Przeprowadzenie refaktoryzacji kodu zgodnie z wynikami analizy.
- Integracja testów jednostkowych w procesie CI/CD (np. GitHub Actions).
- Sporządzenie dokumentacji dotyczącej strategii testowania i wyników analizy.

Tydzień 8 Wdrożenie zabezpieczeń i optymalizacja backendu

Zadania

- Implementacja zabezpieczeń przed atakami typu SQL Injection i XSS.
- Wprowadzenie mechanizmów ograniczania prób logowania (np. blokada po kilku nieudanych próbach).
- Przeprowadzenie testów penetracyjnych (przy użyciu narzędzi OWASP ZAP lub podobnych).
- Usprawnienie konfiguracji FastAPI pod kątem wydajności i bezpieczeństwa.
- Dokumentacja wdrożonych zabezpieczeń oraz przeprowadzonych testów.

Tydzień 9 Przygotowanie środowiska blockchain

Zadania

- Wybór platformy blockchain (np. Ethereum) oraz konfiguracja lokalnej sieci testowej (np. Ganache).
- Instalacja Web3.py i konfiguracja połączenia z lokalnym blockchainem.
- Przeprowadzenie pierwszych testów połączenia i zapoznanie się z dokumentacją Web3.py.
- Przygotowanie przykładowego smart kontraktu (np. prosty rejestr historii medycznej).
- Dokumentacja konfiguracji środowiska blockchain.

Tydzień 10 Implementacja smart kontraktu i integracja z backendem

Zadania

- Napisanie smart kontraktu w Solidity, np. do rejestracji wpisów medycznych.
- Konfiguracja kompilacji i wdrożenia smart kontraktu na lokalnej sieci testowej.
- Przeprowadzenie testów funkcjonalnych smart kontraktu.
- Implementacja integracji z backendem przy użyciu Web3.py.
- Aktualizacja dokumentacji dotyczącej interakcji z blockchainem.

Tydzień 11 Rozszerzenie funkcjonalności blockchain i integracja z modułem medycznym

Zadania

- Rozbudowa smart kontraktu o funkcje wyszukiwania i filtrowania rekordów.
- Implementacja mechanizmu hashowania wpisów medycznych przed ich zapisaniem w blockchainie.
- Integracja modułu medycznego z blockchainem – zapisanie hashy wpisów medycznych na blockchainie przy ich dodawaniu.
- Przeprowadzenie testów integracyjnych między backendem a smart kontraktem.
- Aktualizacja dokumentacji API o nowe funkcje związane z blockchainem.

Tydzień 12 Implementacja modułu finansowego i automatyzacja płatności

Zadania

- Utworzenie podstawowego modułu do wystawiania faktur i rozliczeń (CRUD faktur).
- Integracja systemu z testowym API płatności (np. Stripe, PayU).
- Analiza możliwości zastosowania smart kontraktów do automatycznych rozliczeń (np. weryfikacja płatności, rozliczenia za pomocą kontraktów).
- Implementacja testów funkcjonalnych dla modułu finansowego.
- Dokumentacja procesu wystawiania faktur oraz integracji z systemem płatności.

Tydzień 13 Integracja interfejsu użytkownika z funkcjami blockchain i finansowymi

Zadania

- Aktualizacja interfejsu użytkownika, umożliwiająca przeglądanie historii medycznej zapisanej na blockchainie.
- Dodanie widoków prezentujących status płatności i historię faktur.
- Połączenie funkcjonalności modułów medycznego, finansowego i blockchain w interfejsie użytkownika (np. dashboard z informacjami).
- Przeprowadzenie testów integracyjnych między frontendem, backendem i blockchainem.
- Zbieranie opinii od użytkowników/testujących i wprowadzanie niezbędnych usprawnień.

Tydzień 14 Testy systemowe i optymalizacja wydajności

Zadania

- Uzupełnienie testów jednostkowych dla wszystkich wcześniej zaimplementowanych modułów, dążąc do minimum 75% pokrycia.
- Przeprowadzenie pełnych testów systemowych – testy integracyjne, obciążeniowe i end-to-end.
- Identyfikacja potencjalnych wąskich gardeł oraz optymalizacja zapytań do bazy danych i operacji blockchain.
- Przegląd zabezpieczeń i przeprowadzenie dodatkowych testów penetracyjnych.
- Refaktoryzacja kodu na podstawie wyników testów i analizy wydajności.
- Aktualizacja dokumentacji testów oraz sporządzenie raportu z optymalizacji.

Tydzień 15 Finalizacja, dokumentacja i przygotowanie prezentacji

Zadania

- Opracowanie pełnej dokumentacji projektowej, obejmującej opis systemu, diagramy, instrukcje użytkowania i konfiguracji.
- Uporządkowanie kodu, przeprowadzenie ostatecznych poprawek i weryfikacja, czy wszystkie testy przechodzą pomyślnie.
- Przygotowanie prezentacji systemu, w tym demonstracji działania oraz omówienia kluczowych funkcjonalności i integracji blockchain.
- Spakowanie repozytorium kodu wraz z dokumentacją (README.md, PDF z pełną dokumentacją projektową).
- Finalna weryfikacja i testy systemu w celu zapewnienia jego stabilności i poprawnego działania.