Kamila Skorupka **Justyna Stępień** **Michał Pałucki**

Kompletny projekt systemu

Spis treści:

1. Wymagania:
   a. Funkcjonalne
   b. Niefunkcjonalne

2. Funkcje

3. Diagram ER

4. Architektury systemu:
   a. Moduły systemu
   b. Diagram połączeń

5. Opis interfejsów

6. Lista wykorzystywanych technologii

7. Instrukcja użytkownika

1. Wymagania

a) Funkcjonalne

Funkcje:

(1) Rejestracja:

- Opis: Rejestracja nowego użytkownika w systemie.
- Dane wejściowe: E-mail, imię, nazwisko, hasło.
- Źródło danych wejściowych: Pochodzą od użytkownika.
- Dane wyjściowe: Informacja o poprawnej lub niepoprawnej rejestracji.
- Przeznaczenie: Dodanie danych nowego użytkownika do bazy danych użytkowników oraz umożliwienie późniejszego logowania.
- Wymagania: Dane użytkownika nie mogą istnieć w bazie danych użytkowników oraz dane muszą spełniać poniższe kryteria:
  - Długość adresu e-mail powinna być nie mniejsza niż 4 oraz zawierać symbol ‘@’.
  - Długość imienia oraz nazwiska powinna być nie mniejsza niż 3 znaki oraz nie powinny zawierać cyfr.
  - Długość hasła powinna być nie mniejsza niż 8.
  - Numer telefonu nie zawiera znaków innych niż cyfry.
- Warunek początkowy: Poprawne połączenie ze stroną.
- Warunek końcowy: Poprawność wprowadzanych danych
- Efekty uboczne: Istnieje ryzyko nadmiernych rejestracji tych samych użytkowników
- Uwagi: Użytkownik będzie proszony o powtórzenie hasła.

(2) Logowanie:

- Opis: Loguje użytkownika do systemu.
- Dane wejściowe: E-mail, hasło.
- Źródło danych wejściowych: Zarówno e-mail jak i hasło pochodzą od użytkownika.
- Dane wyjściowe: Informacja o poprawnym lub niepoprawnym zalogowaniu. System sprawdza w bazie danych, czy podany użytkownik istnieje (jeśli nie, pojawia się odpowiednia informacja o braku istnienia danego konta), a następnie sprawdza poprawność hasła.

- Przeznaczenie: Po zalogowaniu się użytkownik ma możliwość korzystania z panelu użytkownika (tj. dokonania rezerwacji, przeglądania historii rezerwacji, zgłoszenia problemu, itp.).
- Wymagania: Poprawne połączenie się użytkownika z aplikacją (brak zakłóceń).
- Warunek początkowy: Poprawne połączenie ze stroną.
- Warunek końcowy: Poprawność wprowadzonych danych.
- Efekty uboczne: Brak.
- Uwagi: Brak.

(3) Dostępne:

- Opis: Wyświetlenie dostępnych (czyli takich, które nie są wypożyczone/zarezerwowane/zepsute) do wypożyczenia urządzeń dla każdego z istniejących w bazie danych punktów wypożyczeń.
- Dane wejściowe: Brak.
- Dane wyjściowe: Lista wszystkich urządzeń.
- Przeznaczenie: Późniejsza możliwość rezerwacji wybranego urządzenia z listy dostępnych urządzeń.
- Wymagania: Poprawne połączenie się użytkownika z aplikacją (brak zakłóceń).
- Warunek początkowy: Uprzednie zalogowanie.
- Warunek końcowy: Istnienie dostępnych urządzeń.
- Efekty uboczne: Brak.
- Uwagi: Brak.

(4) Wypożycz:

- Opis: Umożliwia wypożyczenie konkretnego sprzętu z listy aktualnie dostępnych.
- Dane wejściowe: Sprzęt, lokalizacja.
- Dane wyjściowe: Informacja o rozpoczęciu wypożyczenia.
- Przeznaczenie: Wypożycza sprzęt do momentu oddania, usuwa sprzęt z dostępnych, jeżeli w nich występuje.
- Wymagania: Poprawne połączenie się użytkownika z aplikacją (brak zakłóceń).
- Warunek początkowy: Uprzednie zalogowanie.
- Warunek końcowy: Wybór dostępnego sprzętu.
- Efekty uboczne: Brak.
- Uwagi: Brak.

(5) Zwróć sprzęt:

- Opis: Umożliwia oddanie wypożyczonego sprzętu w wybranym punkcie.

- Dane wejściowe: Punkt oddania.
- Dane wyjściowe: Informacja o poprawnym/ niepoprawnym oddaniu sprzętu do punktu.
- Przeznaczenie: Zakończenie wypożyczenia, utworzenie nowego rekordu w tabeli „Historia”
- Wymagania: Poprawne połączenie się użytkownika z aplikacją (brak zakłóceń).
- Warunek początkowy: Uprzednie zalogowanie.
- Warunek końcowy: posiadanie wypożyczonego urządzenia.
- Efekty uboczne: Brak.
- Uwagi: Brak.

(6) Zgłoś usterkę:

- Opis: Umożliwia zgłoszenie usterki. Poprzez usterkę rozumiemy awarię sprzętu, brak sprzętu na punkcie lub błąd działania aplikacji.
- Dane wejściowe: Opis usterki.
- Źródło danych: Dane pochodzą od użytkownika.
- Dane wyjściowe: Informacja o zgłoszeniu usterki.
- Przeznaczenie: Funkcja umożliwia administracji kontrolę nad aplikacją oraz sprzętem.
- Wymagania: Poprawne połączenie się użytkownika z aplikacją (brak zakłóceń).
- Warunek początkowy: Uprzednie zalogowanie.
- Warunek końcowy: Formularz zgłoszenia nie może być pusty.
- Efekty uboczne: Istnieje ryzyko zgłaszania fałszywych usterek.
- Uwagi: Brak.

(7) Ogłoszenia:

- Opis: Wyświetla wszystkie ogłoszenia (posortowane wg daty publikacji).
- Dane wejściowe: Brak.
- Dane wyjściowe: Lista wszystkich ogłoszeń.
- Przeznaczenie: Użytkownik ma możliwość sprawdzenia najnowszych aktualności.
- Wymaga: Poprawne połączenie się użytkownika z aplikacją (brak zakłóceń).
- Warunek początkowy: Uprzednie zalogowanie.
- Warunek końcowy: Brak.
- Efekty uboczne: Brak.
- Uwagi: Brak.

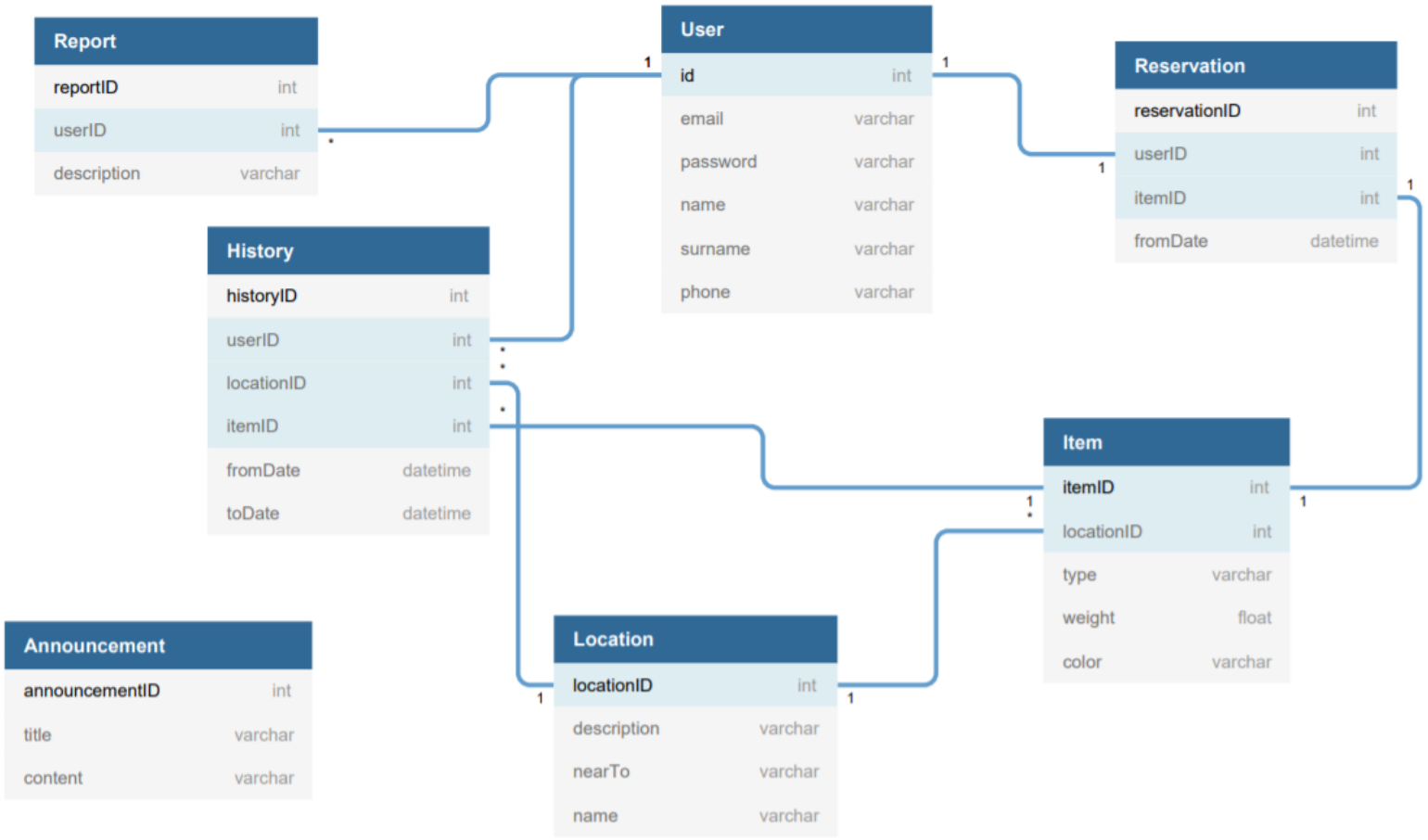
b) Niefunkcjonalne:

- Szybkość jest jednym z najznaczniejszych wymogów.
- System powinien przetwarzać dane w możliwie minimalnym czasie.
- System powinien być bezpieczny.
- Dostęp do danych powinni mieć tylko upoważnieni użytkownicy.
- System powinien być łatwy w utrzymaniu.
- Oprogramowanie powinno być działającym i użytecznym produktem.
- Dane powinny być wiarygodne i dostępne w razie potrzeby.

2. Funkcje:

(1) Rejestracja – założenie konta
(2) Logowanie – zalogowanie się do istniejącego konta
(3) Dostępne – wyświetlenie dostępnych pojazdów
(4) Wypożycz – wypożyczenie pojazdu
(5) Zwróć sprzęt – oddanie pojazdu po wypożyczeniu
(6) Zgłoś usterkę – zgłoszenie usterki przed/ w trakcie/ po - wypożyczeniu
(7) Ogłoszenia – wyświetlenie bieżących ogłoszeń

3. Diagram ER:



4. Architektury systemu:

a) Moduły systemu**.**

W tworzonym systemie wykorzystany został popularny wzorzec projektowy wyodrębniający 3 komponenty aplikacji.

(1) interfejs użytkownika
(2) logika sterowania
(3) model danych

Zaletą tego wzorca jest odseparowanie części wizualizacyjnej programu (GUI) od logiki systemu odpowiedzialnej za kontrolę i przesyłanie danych oraz od tego, w jaki sposób tego jak te dane są przechowywane. Popularną nazwą tego wzorca jest MVC (Model-View-Controler):

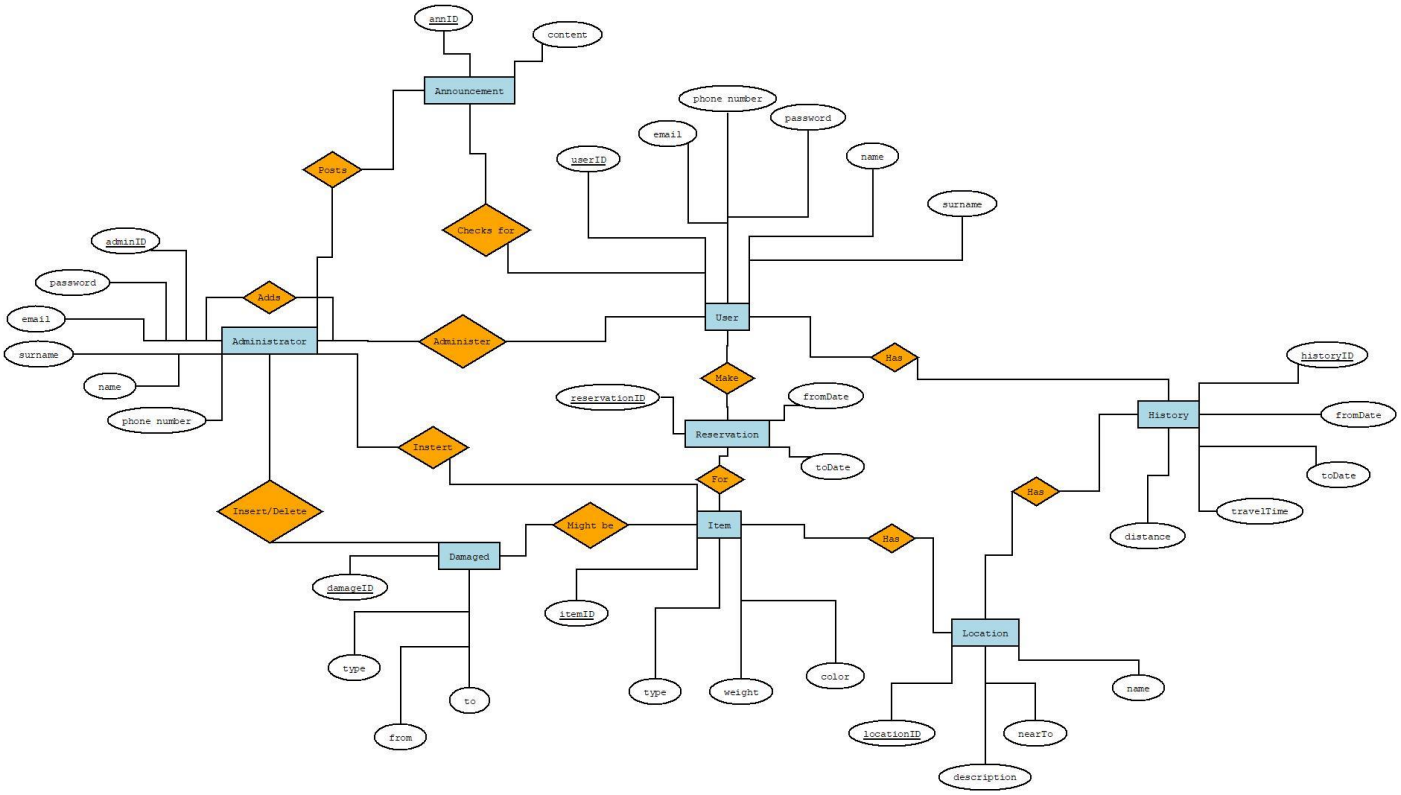
(1) Model – opisuje dane przechowywane przez system oraz steruje ich tworzeniem, modyfikacją i usuwaniem
(2) Widok – wyświetla dane przechowywane w modelu użytkownikowi w przejrzysty

(3) Kontroler – odpowiada za przekształcenie danych z widoku i zapytań użytkownika na polecenia zrozumiałe dla modelu, oraz na przekształcanie danych

Specyfikacja zależności w modelu MVC wygląda następująco:

(1) Widok wysyła zapytania do kontrolera i odświeża informacje stosownie do aktualnych danych
(2) kontroler reaguje na zapytania widoku pobierając lub modyfikując dane w modelu
(3) model stosownie do polecę kontrolera zwraca aktualne dane bądź dokonuje w nich zmiany

b) Diagram połączeń



5. Opis interfejsów:

1. <u>Opis interfejsu użytkownika</u>

Panel użytkownika umożliwia skorzystanie z funkcji:

- Dostępne
- Rezerwuj
- Wypożycz

- Oddaj
- Historia rezerwacji
- Zgłoś usterkę
- Ogłoszenia

Każda z powyższych funkcji została opisana w wymaganiach.

2. Opis połączenia aplikacji z GUI
   Do stworzenia graficznego interfejsu wykorzystaliśmy Flaska. Komunikacja z aplikacją polega m.in. na obsłudze metod "GET/POST", dzięki którym aplikacja obsługuje działania użytkownika.
   Wewnątrz aplikacji stworzyliśmy szablony stron html przy użyciu funkcji bootsrap'a.

3. Opis połączenia aplikacji z bazą danych

   Baza danych została połączona z aplikacją przy użyciu Flaska oraz biblioteki SQLAlchemy. Wykorzystaliśmy bazę SQLite. Dzięki takiemu połączeniu w prosty sposób stworzyliśmy całą bazę danych (tabele oraz funkcje zmieniające je) wewnątrz aplikacji.

6. Lista wykorzystywanych technologii + uzasadnienie:

a) Python – Prosty w użyciu, łatwy w zrozumieniu
b) Flask + SQL-Alchemy - SQL + programowanie obiektowe = SQLAlchemy
c) Jinja - Szablony Jinja oferują podstawowe funkcje programistyczne, takie jak zastępowanie zmiennych, pętle, wywołania funkcji, filtry, a także możliwość rozszerzania podstawowych komponentów. Szablony Jinja są tradycyjnie używane w programowaniu HTML / WWW do tworzenia widoków przy użyciu Flaska jako struktury internetowej.
d) Discord – Prosty i wielofunkcyjny komunikator wspomagający komunikację między członkami zespołu
e) Github – najlepsze dostępne narzędzie do pracy w grupie nad jednym kodem

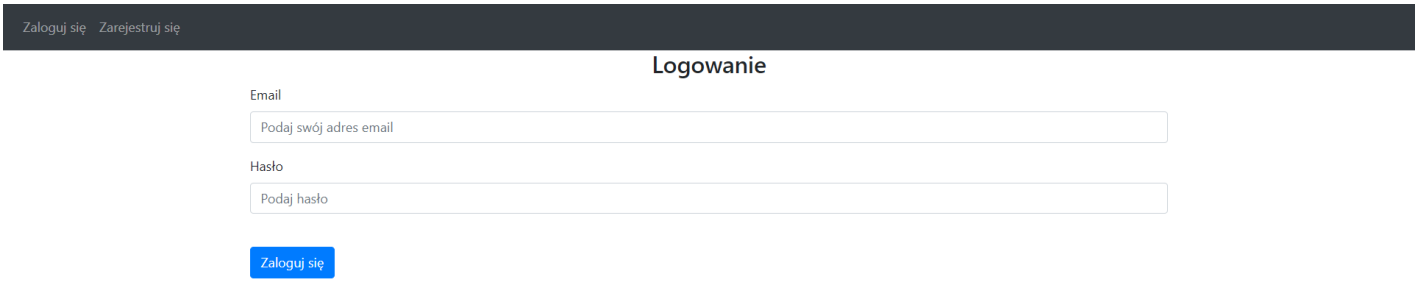
7. Instrukcja użytkownika

1. Uruchomienie aplikacji

Aktualnie aplikacja uruchamiana jest lokalnie. Po uruchomieniu terminala należy wpisać „python […]/main.py" (gdzie […] to lokalizacja pliku main.py (docelowo w plikach projektu)) w celu uruchomienia aplikacji. Następnie pojawi się komunikat, w którym będzie widniał link do skopiowania: http://127.0.0.1:5000/. Należy skopiować go i wkleić w przeglądarce internetowej. Użytkownik zostanie przekierowany na stronę aplikacji.

2. Logowanie oraz rejestracja

W celu zalogowania się, użytkownik powinien wpisać poprawnie dane lub zarejestrować się.



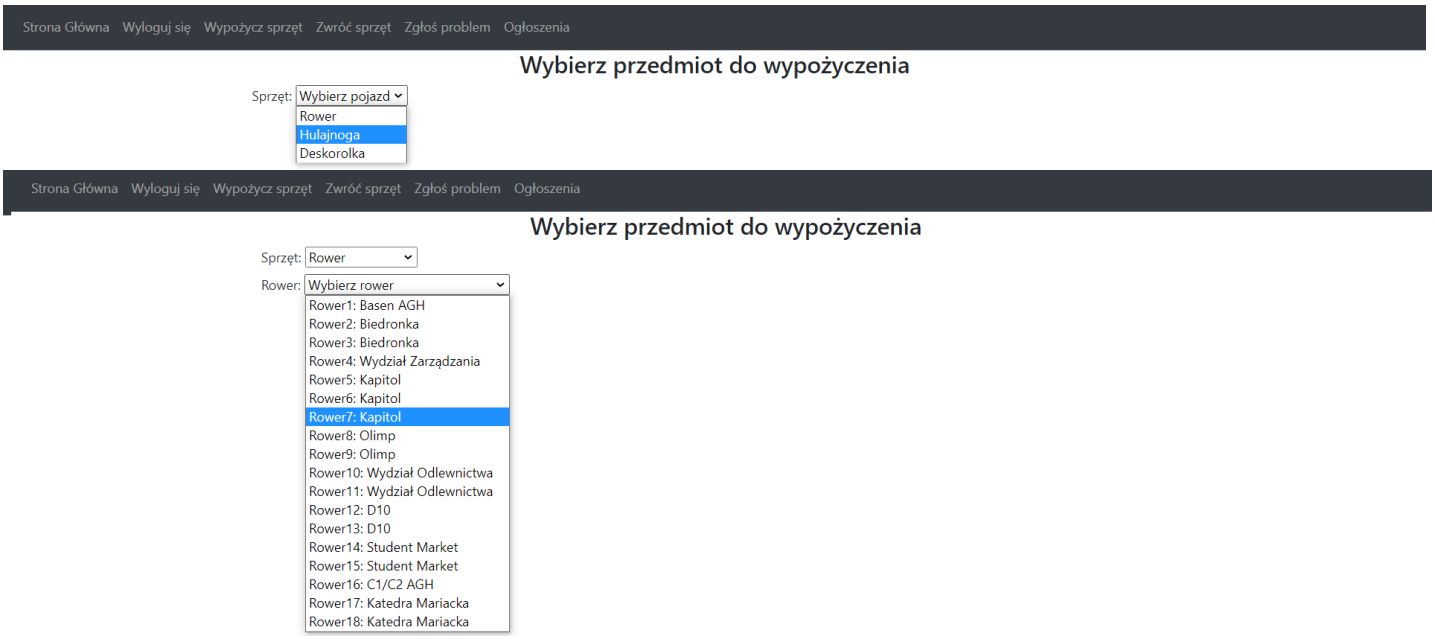
Rejestracja wymaga poprawności wprowadzanych danych.

3. Strona główna

Po zalogowaniu się użytkownik zostanie przekierowany na stronę główną.

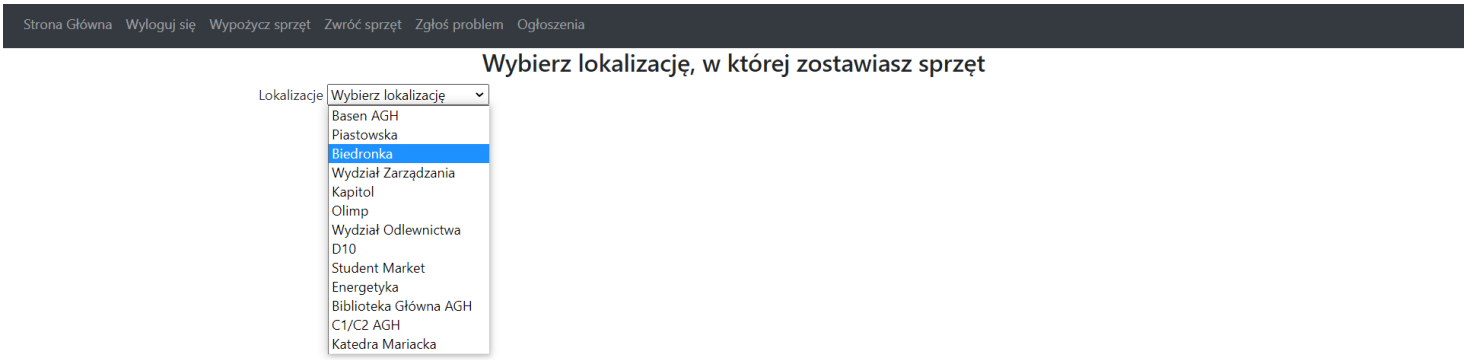
4. Wypożyczenie sprzętu

W celu wypożyczenia sprzętu należy wcisnąć przycisk "Wypożycz sprzęt". Następnie należy wybrać rodzaj sprzętu oraz lokalizację.



5. Zwrot sprzętu

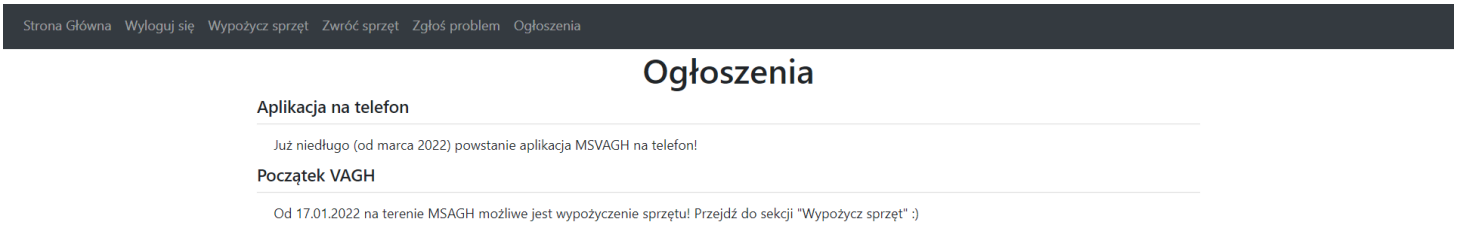
W celu zwrócenia sprzętu należy wcisnąć przycisk "Zwróć sprzęt", a następnie wybrać lokalizację, w ktorej sprzęt zostaje pozostawiony.



Nie można zwrócić sprzętu, jeśli żadnego się nie wypożyczyło.

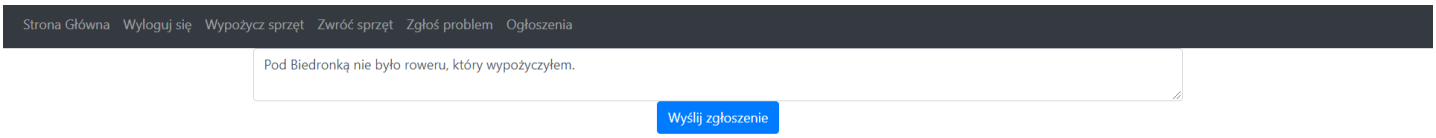
6. Ogłoszenia

W celu sprawdzenia ogłoszeń należy wcisnąć przycisk "Ogłoszenia".



7. Zgłaszanie problemu

W celu zgłoszenia usterki należy wcisnąć przycisk "Zgłoś problem". Należy wpisać zgłoszenie a następnie kliknąć "Wyślij zgłoszenie". Uwaga, zgłoszenie nie może być puste.



8. Wylogowanie

Aby wylogować się należy wcisnąć przycisk "Wyloguj się".