

Dokumentacja projektu z przedmiotu MAS

System do zarządzania szkołą motocross-ową

Student:

Jakub Redziński vel Rydzyński s16889

Adres email: s16889@pjwstk.edu.pl

Spis treści

1. Dziedzina problemowa:	3
2. Cel	3
3. Zakres odpowiedzialności systemu.....	3
4. Użytkownicy systemu.....	3
5. Wymagania użytkownika	4
6. Diagram przypadków użycia	6
7. Diagram klas – analityczny	7
8. Diagram klas – projektowy.....	8
9. Scenariusz przypadku użycia (jako tekst w punktach)	9
10. Diagram aktywności dla przypadku użycia.....	10
11. Diagram stanu dla klasy	10
12. Diagram interakcji (sekwencji) dla przypadku użycia.....	11
13. Projekt GUI	12
14. Omówienie decyzji projektowych i skutków analizy dynamicznej.....	15
14.1. Decyzje projektowe	15
14.2. Skutki analizy dynamicznej.....	15

1. Dziedzina problemowa:

W dzisiejszych czasach ludzie spędzają coraz więcej czasu na rozwijaniu własnych zainteresowań. Jednym z zainteresowań może być jazda motocyklem typu cross tzw. Motocross. Niestety ciężko jest znaleźć miejsce przeznaczone do tego typu aktywności. W tym celu powstała szkoła motocrossowa, jest to swego rodzaju nisz na rynku przez co brakuje odpowiedniego oprogramowania – oprogramowania do przechowywania informacji związanych z prowadzeniem ww. szkoły motocrossowej.

2. Cel

Powstała więc potrzeba stworzenia systemu informatycznego który umożliwiłby właścicielowi oraz klientowi tzw. Zawodnikowi proces planowania treningów, zawodów oraz zarządzania płatnościami za pozostawiony jednoślad. Dodatkowo system pozwoliłby na przechowywanie danych na temat wydatków związanych z prowadzeniem szkoły.

3. Zakres odpowiedzialności systemu

Głównym zadaniem systemu będzie obsługa procesu treningów i zawodów crossowych. Będzie on również przechowywał informację o crossach znajdujących się w szkole, o ich właścicielach, klientach oraz grupach w szkole crossowej. Dodatkowo system ten będzie zapamiętywał wydatki jakie poniosła szkoła za dany motocykl i umożliwi rozliczenie się z właścicielem.

4. Użytkownicy systemu

- Właściciel szkoły motocross-owej
- Zawodnik
- Właściciel motocykla

5. Wymagania użytkownika

System wspomagający pracę szkoły motocross-owej będzie przechowywać informacje na temat zawodników oraz właścicieli motocykli. Każdy właściciel może być zawodnikiem, a także zawodnik może być właścicielem. Zarówno dla właścicieli jak i zawodników system powinien przechowywać dane takie jak, imię, nazwisko oraz numer telefonu.

Dodatkowo dla zawodnika chcielibyśmy przechowywać informacje dodatkowe takie jak: unikalny numer licencji motocross-owej, ligę w jakiej startuje, przy czym może to być: (KLB – czyli klubowa, REG – czyli regionalna, POL – czyli ogólnopolska oraz GLOBAL, czyli międzynarodowa). Dodatkowo dla zawodnika chcielibyśmy przechowywać aktualne miejsce w lidze, datę urodzenia oraz wyliczalne pole wiek.

Każda osoba może przechowywać motocykl w szkole, niezależnie czy jest to zawodnik czy po prostu właściciel motocykla. Dla właścicieli motocykli chcielibyśmy przechowywać informacje na temat numeru konta. W momencie zabrania motocykla ze szkoły (usunięcia go z bazy danych), wszystkie jego wydatki powinny zostać usunięte. Ważnym aspektem jest, że motocykl może mieć maksymalnie jednego właściciela oraz że właściciel motocykla może zmieniać stan motocykla ze sprawnego na popsuty oraz na odwrót.

Dla każdego motocykla system będzie przechowywał informacje takie jak: marka, model, data produkcji, wyliczalne pole przebieg, moc silnika, datę serwisu oraz czy motocykl jest sprawny, wartość true – kiedy motocykl jest sprawny, wartość false – kiedy motocykl jest popsuty.

Motocykle dzielą się ze względu na typ na dwusuwowe, czterosuwowe oraz elektryczne. Ze względu na bardziej złożoną obsługę motocykli czterosuwowych chcielibyśmy dla nich przechowywać informacje na temat: rodzaju zaworu, rodzaju pompy paliwa oraz datę wymiany układu rozrządu. Dla motocykli dwusuwowych przechowujemy jedynie informacje na temat typu mieszanki paliwowej, natomiast dla motocykli elektrycznych przechowujemy dane o pojemności baterii.

System będzie również przechowywał informacje o kosztach wynikających z przetrzymywania motocykla w szkole. Wydatek będzie zawierał: numer rachunku, datę rozliczenia - dodaną w momencie opłacenia wydatku oraz jego koszt.

Każdy z zawodników może dołączyć do tzw. grupy, jednak trzeba pamiętać, że może on przychodzić indywidualnie i wtedy nie musi należeć do ww. grupy. Dla grup przechowujemy informacje na temat: nazwy grupy, daty założenia, ilości członków oraz poziomu zaawansowania który może być początkujący – „beginner”, średnio-zaawansowany – „intermediate”, zaawansowany – „advanced”. Jeden zawodnik może należeć maksymalnie do jednej grupy.

Jednym z kluczowych wymagań systemu jest planowanie wykonania treningów przez zawodników, w związku z czym należy pamiętać informacje o dacie odbycia treningu, priorytecie, opisie oraz informacje na temat czy został już wykonany. Ważnym aspektem jest, że zawodnik może zaplanować trening z podanym motocyklem.

Również, zawodnik może brać udział w wielu zawodach, dla każdego zawodów przechowujemy informacje na temat: nazwy, daty startu, daty do, puli nagród w postaci kwoty oraz typu turnieju który może być: „KLB”, dla Regionalnych – „REG”, dla Ogólnopolskich – „POL” a dla międzynarodowych jest to – „GLOBAL”. Przy zapisie na zawody, należy pamiętać o sprawdzeniu uprawnień zawodników. Dodatkowo system umożliwia wyświetlenie uczestników turnieju oraz pozwala wyszukać turniej po podanym id turnieju.

Właściciel szkoły może:

- Dodać wydatek do zapłacenia za motocykl przez właściciela

Zawodnik może:

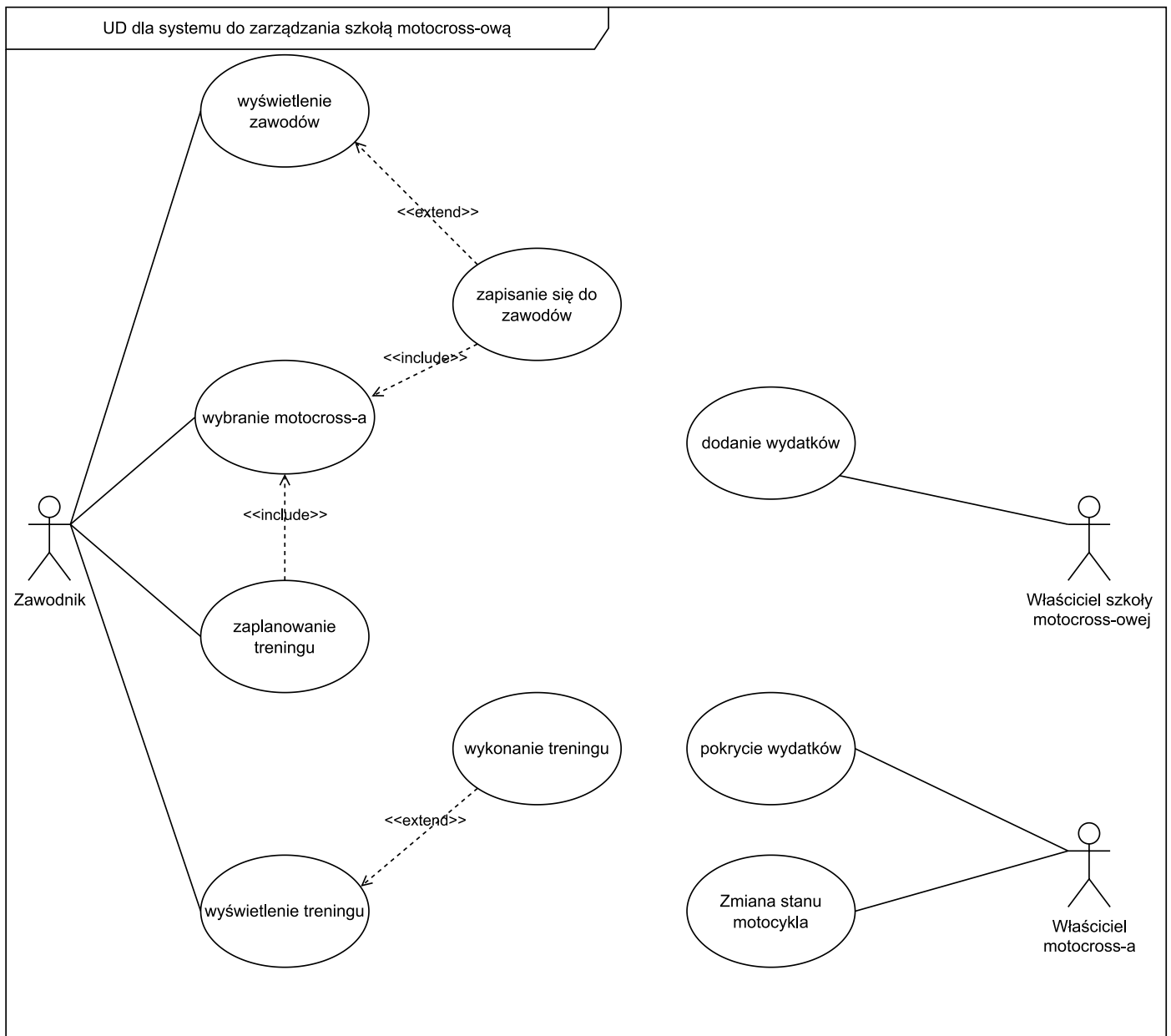
- Wybrać motocross-a
- Zapisać się na zawody z wybranym motocyklem
- Zaplanować trening z wybranym motocyklem
- Oznaczyć trening jako wykonany
- Dołączyć do grupy

Właściciel motocross-a może:

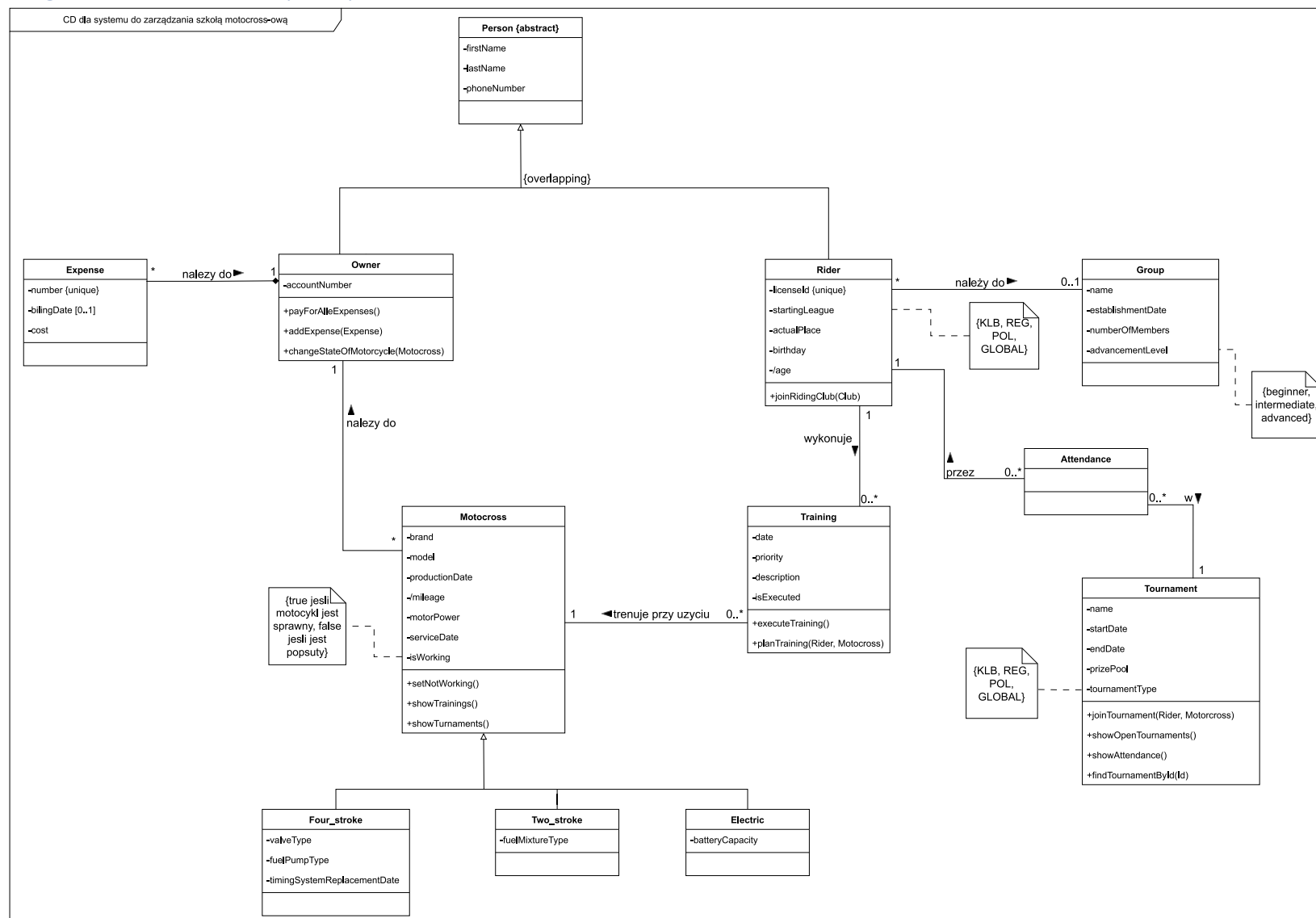
- Pokryć wydatki poniesione w związku z przetrzymywaniem motocykla w szkole motocross-owej
- Zmieniać stan motocykla: sprawny/popsuty

6. Diagram przypadków użycia

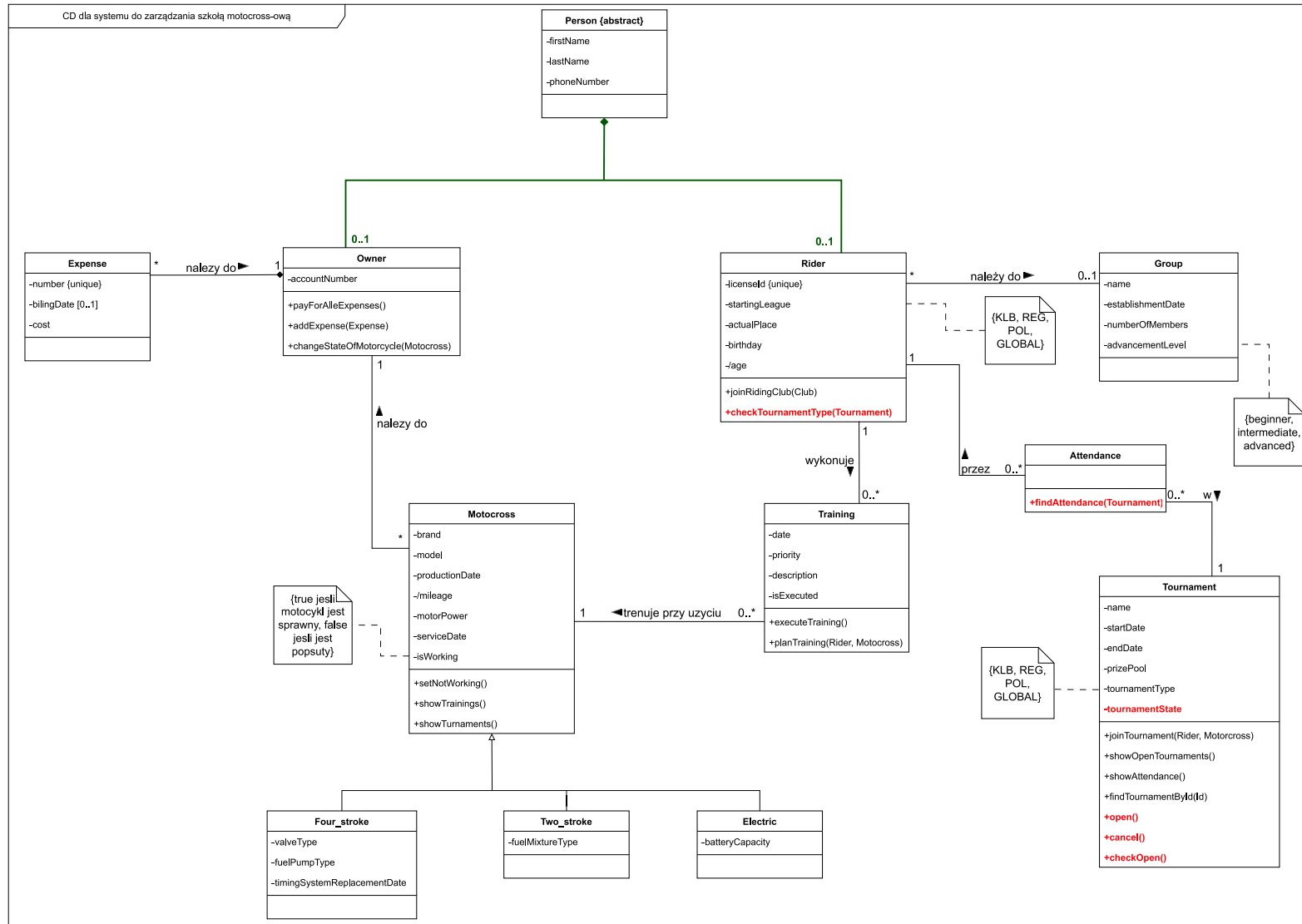
Dla systemu do zarządzania szkołą motocross-ową powstał poniższy diagram przypadków użycia (UD).



7. Diagram klas – analityczny



8. Diagram klas – projektowy



9. Scenariusz przypadku użycia (jako tekst w punktach)

Przypadek użycia rozpoczyna Aktor – Zawodnik.

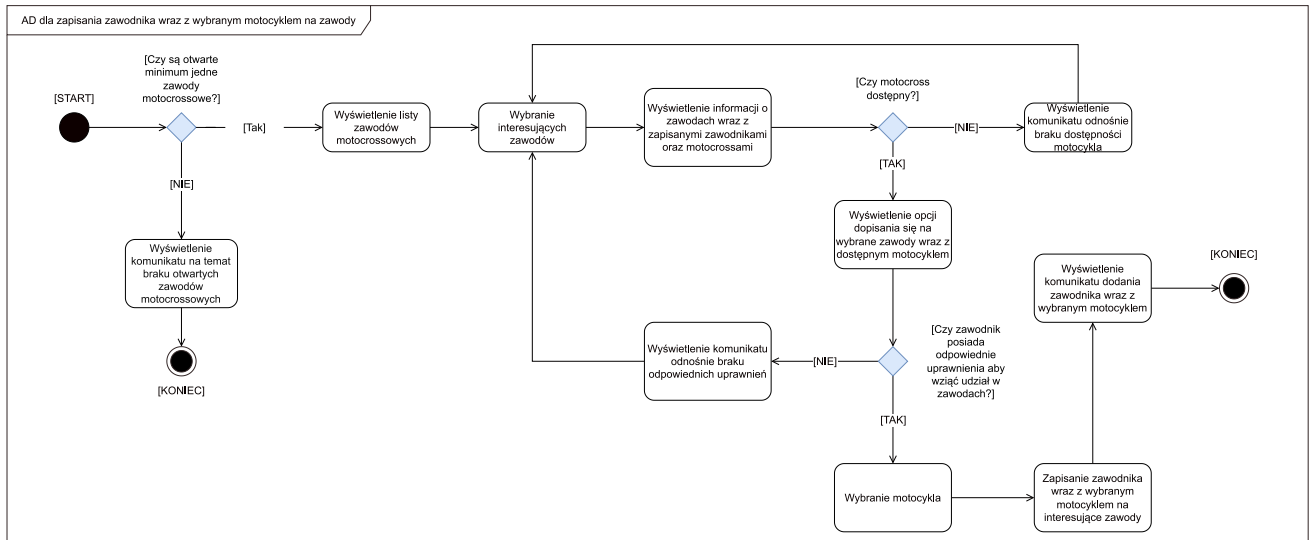
Warunek początkowy: Zawodnik musi być zalogowany w systemie oraz w bazie danych istnieją zawody motocrossowe.

1. System weryfikuje czy są w bazie danych otwarte zawody w których zawodnik mógłby wziąć udział.
2. System zwraca listę zawierającą dostępne zawody.
 - 2a. Brak zawodów. System wyświetla komunikat o braku dostępnych zawodów i kończy przypadek użycia.
3. Zawodnik wybiera interesujące go zawody.
4. System zwraca informacje na temat zawodników i ich motocrossach zapisanych do wybranych zawodów.
5. System wyświetla informacje o możliwości dopisania się na wybrane zawody wraz z dostępnym motocyklem.
 - 5a. Brak dostępnych motocykli do dołączenia w wybranych zawodach. System wyświetla komunikat o braku dostępnych motocykli do dołączenia w wybranych zawodach i następuje powrót do scenariusza głównego w punkcie 4.
6. System sprawdza czy zawodnik może wziąć udział w wybranych zawodach.
 - 6a. Aktor nie ma uprawnień, aby wziąć udział w ww. zawodach. System wyświetla komunikat z informacją: „Nie masz odpowiedniej licencji, aby wziąć udział w tych zawodach”. Następuje powrót do scenariusza głównego w punkcie 4.
7. Zawodnik wybiera motocykl, na którym chce startować w zawodach motocrossowych.
8. System zapisuje zawodnika wraz z wybranym motocyklem na wybrane zawody oraz system wyświetla komunikat o pozytywnym zapisie zawodnika i kończy przypadek użycia.

Warunek końcowy: Aktor został dodany do zawodów wraz z wybranym motocross-em.

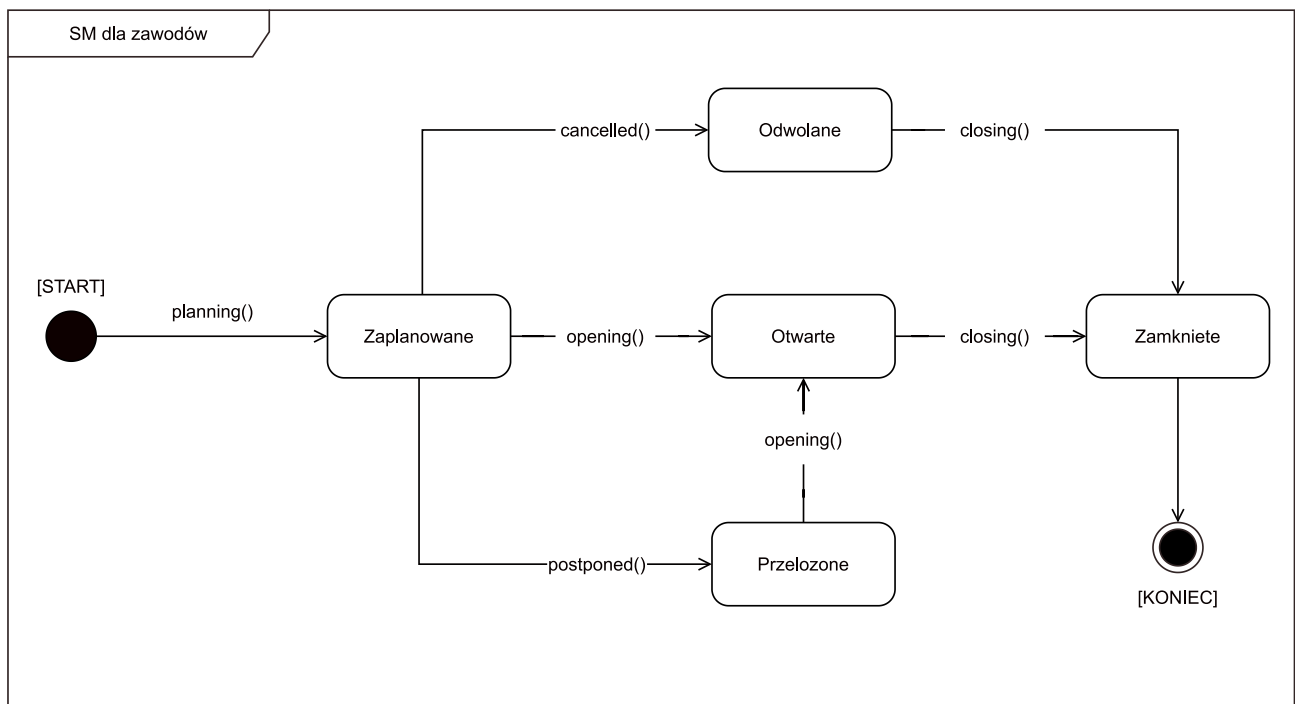
10. Diagram aktywności dla przypadku użycia

Dla systemu do zarządzania szkołą motocross-ową a dokładnie dla zapisania zawodnika wraz z wybranym motocyklem na zawody, powstał poniższy diagram aktywności dla przypadku użycia (AD).



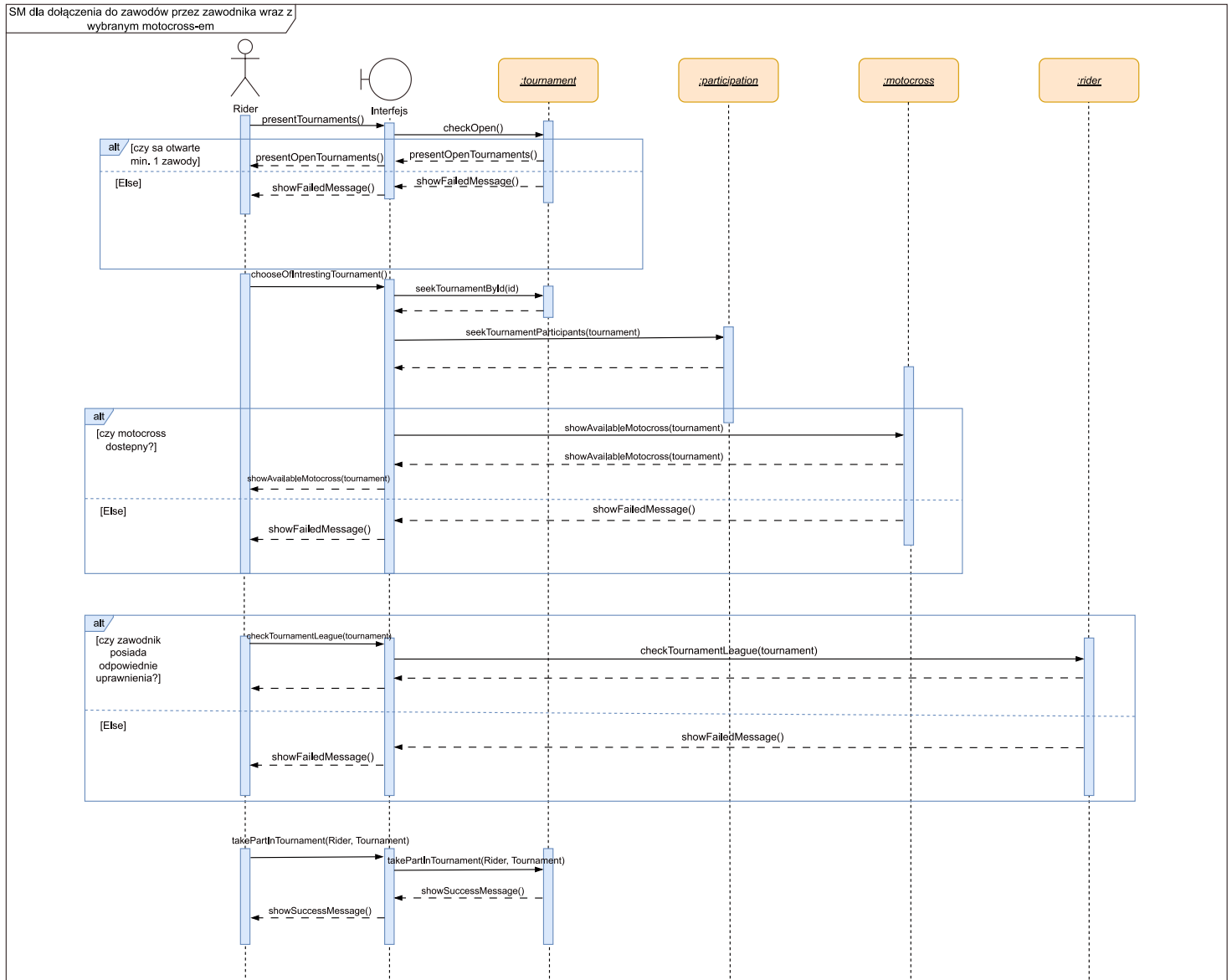
11. Diagram stanu dla klasy

Dla systemu do zarządzania szkołą motocross-ową powstał poniższy diagram stanu dla klasy Tournaments (SM).



12. Diagram interakcji (sekwencji) dla przypadku użycia

Dla systemu do zarządzania szkołą motocross-ową powstał poniższy diagram interakcji (sekwencji) dla przypadku użycia (SM). Diagram ten dotyczy dołączenia do zawodów przez zawodnika wraz z wybranym motocross-em.



13. Projekt GUI

Dla systemu do zarządzania szkołą motocross-ową powstał poniższy projekt GUI.

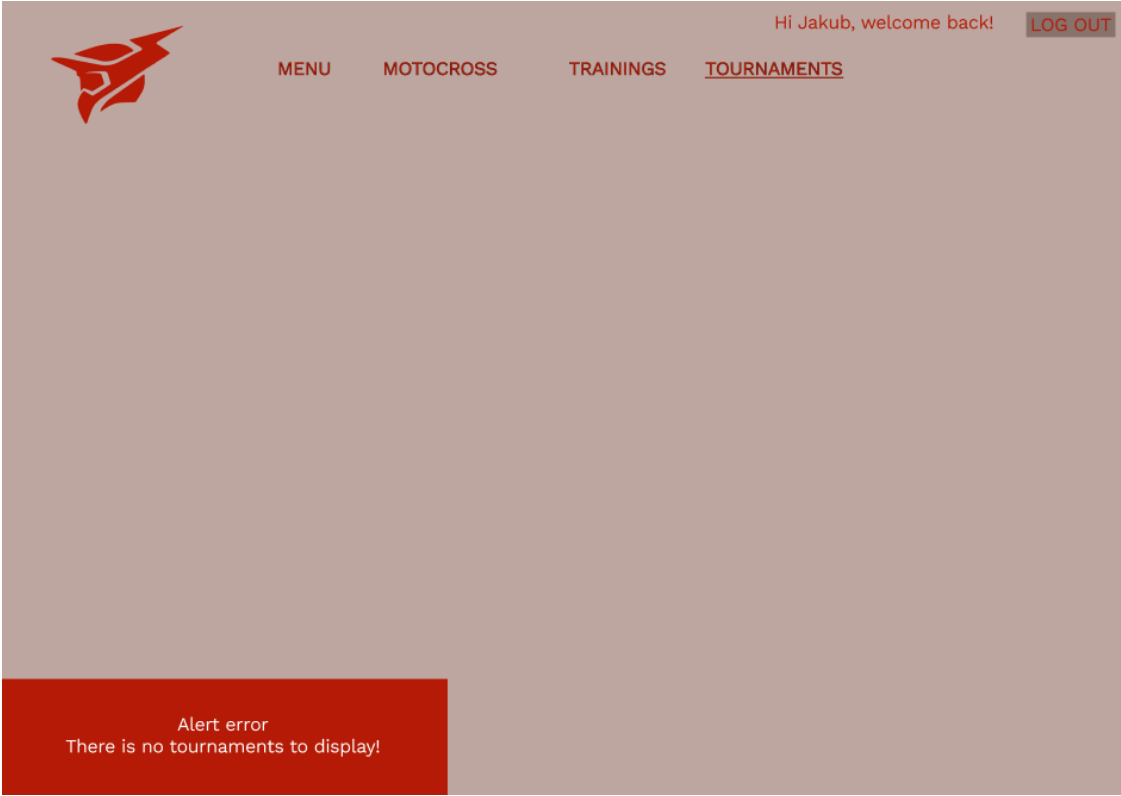
1. Ekran logowania.

[illegible]

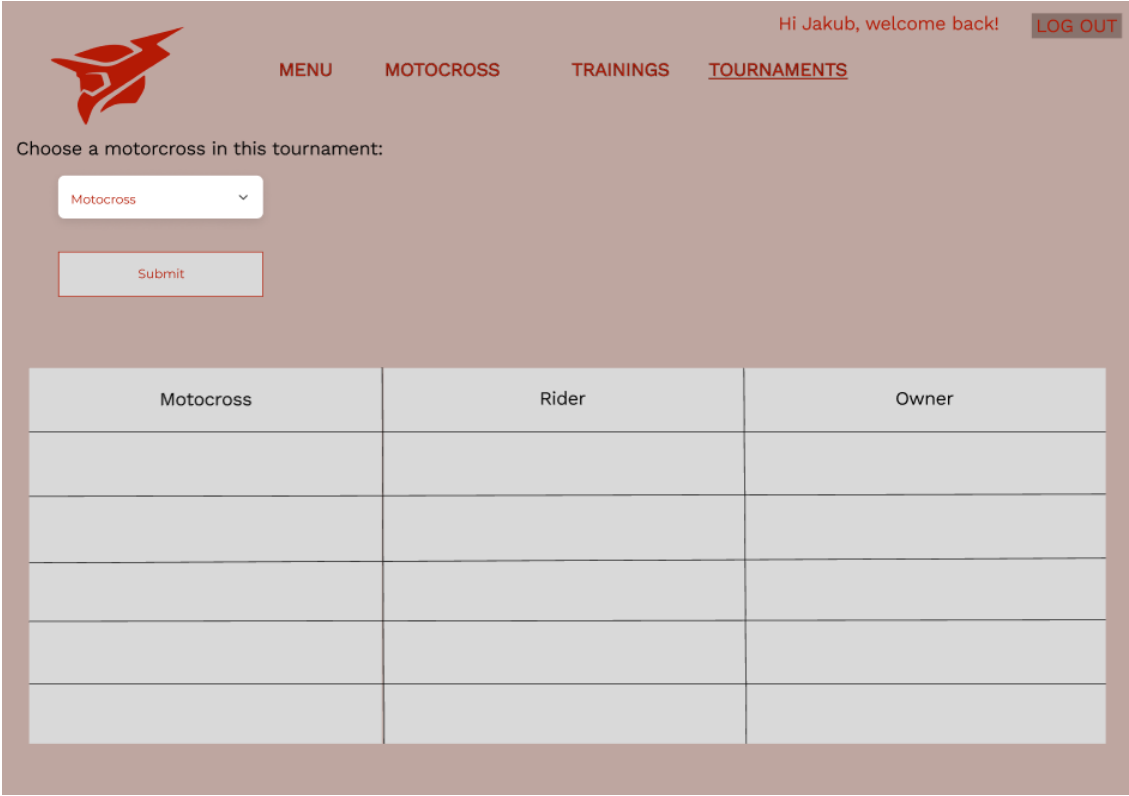
2. Sekcja „Tournament” wraz z listą aktualnych turniejów.

[illegible]

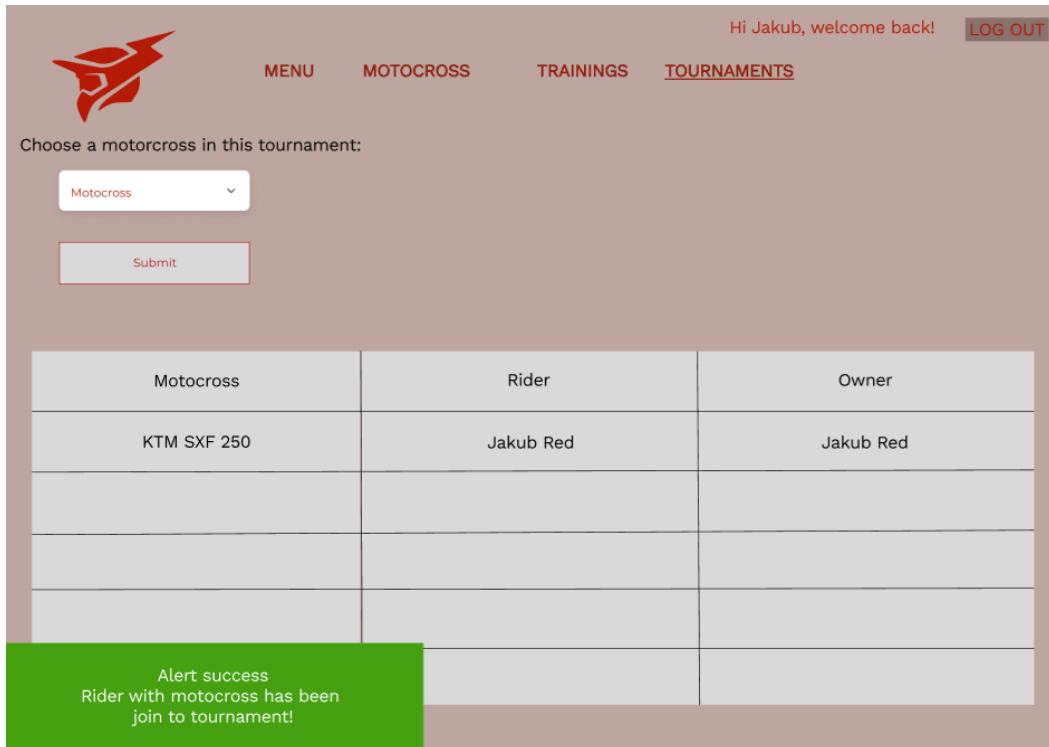
3. Ścieżka alternatywna: Przejście do sekcji „Tournaments”, brak turnieju do wyświetlenia. Alert error z informacją zostaje wyświetlony.



4. Sekcja „Join Tournament” po wcisnięciu przycisku w kolumnie „Join”.



5. Wybranie motocrossa na którym zawodnik pojedzie w turnieju. W przypadku dodania zawodnika z wybranym motocyklem do listy, system wyświetli alert: „success” wraz z informacją.



The screenshot shows a web interface for a motocross tournament. At the top, there is a navigation bar with a logo on the left and links for MENU, MOTOCROSS, TRAININGS, and TOURNAMENTS on the right. A user greeting "Hi Jakub, welcome back!" and a "LOG OUT" button are also present. Below the navigation bar, the text "Choose a motorcross in this tournament:" is displayed. A dropdown menu labeled "Motocross" is open, showing a list of options. Below the dropdown is a "Submit" button. A table with three columns: "Motocross", "Rider", and "Owner" is shown. The first row contains the data "KTM SXF 250", "Jakub Red", and "Jakub Red". Below the table, a green alert box displays the message: "Alert success Rider with motocross has been join to tournament!".

Hi Jakub, welcome back! LOG OUT

MENU MOTOCROSS TRAININGS TOURNAMENTS

Choose a motorcross in this tournament:

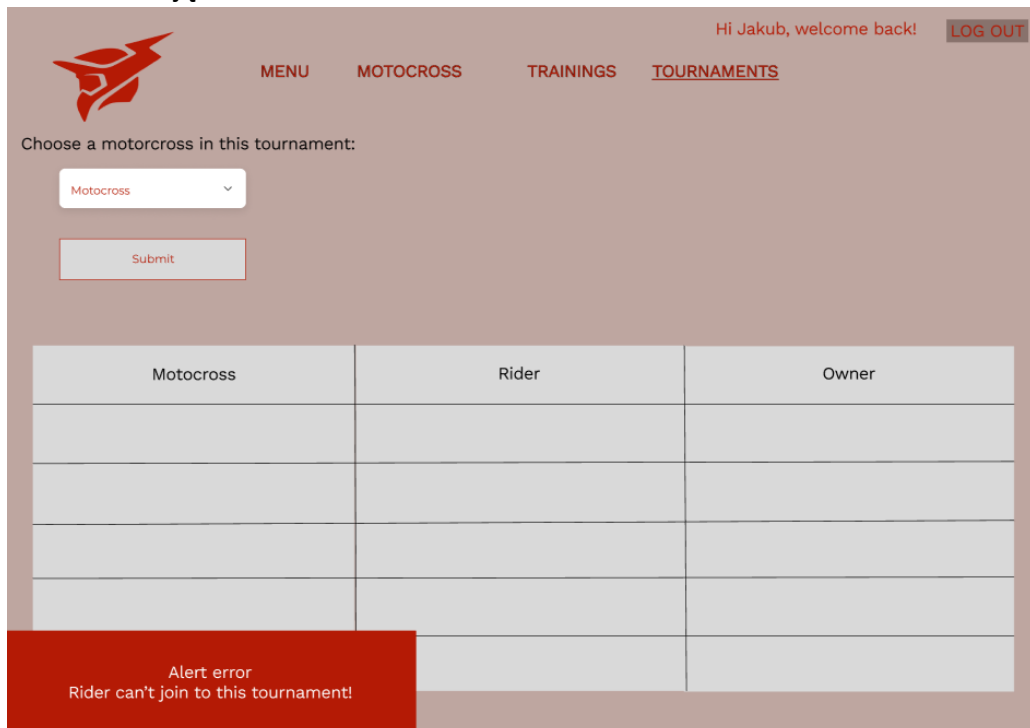
Motocross

Submit

Motocross	Rider	Owner
KTM SXF 250	Jakub Red	Jakub Red

Alert success
Rider with motocross has been
join to tournament!

6. Ścieżka alternatywna: Wybranie motocrossa na którym zawodnik pojedzie w turnieju. W przypadku posiadania nieprawidłowych uprawnień lub motocykla który jest nieprawidłowej kategorii dla danego turnieju, system wyświetli alert: „error” wraz z informacją.



The screenshot shows the same web interface as the previous one, but with an error alert. The navigation bar and the "Choose a motorcross in this tournament:" section are identical. However, the table is empty. A red alert box at the bottom displays the message: "Alert error Rider can't join to this tournament!".

Hi Jakub, welcome back! LOG OUT

MENU MOTOCROSS TRAININGS TOURNAMENTS

Choose a motorcross in this tournament:

Motocross

Submit

Motocross	Rider	Owner

Alert error
Rider can't join to this tournament!

14. Omówienie decyzji projektowych i skutków analizy dynamicznej

14.1. Decyzje projektowe

W powyższym projekcie zostały podjęte poniższe decyzje projektowe:

- Ekstensja będzie realizowana za pomocą bazy danych MySQL.
- Asocjacje będą realizowane za pomocą ORM Hibernate z adnotacjami @ManyToMany, @ManyToOne, @OneToMany.
- Ograniczenie unique oraz optional będą realizowane za pomocą ORM Hibernate z adnotacjami.
- Kompozycja będzie zaimplementowana jak relacja przy użyciu typowych adnotacji.
- Wybrana strategia zapisu w bazie danych to single_table.

```
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
```

- W projekcie problem dziedziczenia łącznego (overlapping) zostanie rozwiązany za pomocą kompozycji.
- W projekcie zostało zastosowane dziedziczenie rozłączne (disjoint). Dzięki temu, możemy utworzyć podklasy „Rider” oraz „Owner” na podstawie nadklasy „Person”.
- Atrybuty pochodne takie jak: age, mileage będą wyliczane za pomocą metod.

14.2. Skutki analizy dynamicznej

Dzięki przeprowadzeniu analizy dynamicznej zostały znalezione poniższe odchylenia:

- W klasie „Tournament” brakowało pola, dzięki któremu moglibyśmy zarządzać stanem zawodów, więc zostało dodane pole: „tournamentState”. Dodatkowo brakował metody open() otwierającej zawody oraz metody cancel() zamykającej zawody. Również w klasie „Tournament” należało dodać metodę checkOpen() która będzie wyświetlała informacje na temat obecnie otwartych zawodów. Dodatkowo zostało dodany atrybut w klasie „Tournament” o nazwie: „tournamentState”. Atrybut ten będzie odpowiadało za przechowanie informacji o stanie zawodów.

- W klasie „Rider” brakowało metody, która sprawdzałaby typ turnieju. Dzięki dodaniu metody: checkTournamentType(Tournament) jesteśmy w stanie sprawdzić czy Rider jest w stanie wziąć udział w zawodach pod kątem posiadanej licencji.

-W klasie „Attendance” brakowało metody, która wyświetliłaby nam udział w turnieju przez zawodników. Dzięki implementacji metody: findAttendance(Tournament) jesteśmy w stanie sprawdzić udział z powiązanymi zawodnikami.

Jakub Redziński vel Rydzyński s16889

Adres email: s16889@pjawst.edu.pl