# Wstęp

## Krótki opis projektu:

Projekt składa się z 4 głównych modułów – Core, Players, Gui i Pieces. W Core znajduje się większość reguł gry oraz działanie szachownicy, w Pieces pozostałe reguły, dotyczące ruchów bierek. Players chroni informację o graczach a Gui odpowiada za interfejs graficzny.

## Podejście do testów:

* Podzielić testy na 2 grupy – systemowe i jednostkowe.
* Sprawdzić działanie testów, przy czym dodawać i sprawdzać nowe testy po każdym dodaniu ważnych module.
* Opisać wszystkie błędy które nie zostały naprawione oraz wskazać przyczynę tego.

# Testy systemowe

## Testy funkcjonalne

Testy funkcjonalne podzieliliśmy na 2 części:

1. Testowanie działania gry w szachy.
2. Testowanie dostosowywania parametrów gry.

### Testy gry

Podstawowe ruchy bierek, np. możliwość przesunięcia bierki poza szachownicę oraz poprawność ruchów każdego typu bierek na szachownicy.

1. Bardziej zaawansowane reguły, [roszada](https://pl.wikipedia.org/wiki/Roszada) oraz [en passant](https://pl.wikipedia.org/wiki/Pion_(szachy)#Bicie_w_przelocie).
2. Poruszanie bierkami pod czas [związania](https://pl.wikipedia.org/wiki/Zwi%C4%85zanie).
3. Wybór nowej figury po dostaniu pionu do ostatniego rzędu szachownicy.

### Testy dostosowywania

Wizualizacja:

1. Poprawność wyboru koloru szachownicy.
2. Ustawienie formatu czasowego.

Notacji:

1. Import/export pozycji w [notacji FEN](https://pl.wikipedia.org/wiki/Notacja_Forsytha-Edwardsa).
2. Import/export gry w [notacji PGN](https://pl.wikipedia.org/wiki/Portable_Game_Notation).

Boty:

1. Działanie bota [Stockfish](https://pl.wikipedia.org/wiki/Stockfish_(szachy)) przeciw gracza.
2. Działanie bota [Stockfish](https://pl.wikipedia.org/wiki/Stockfish_(szachy)) przeciw samego siebie.

## Testy wydajnościowe

Do testowania wydajności użyto różnych „głębokości” bota Stockfish oraz odnalezione optymalne granice, w których bot nie zawiesza aplikację (1-24 poziom).

# Testy jednostkowe

Ze względów na małą liczność zespołu oraz bardzo rozbudowany system aplikacji zdecydowaliśmy nie pokrywać 100% kodu testami UT. Zamiast tego testy były napisane do kilku ważnych klas i funkcji.

Wytestowane były:

1. PlayerClass, HumanPlayer – poprawne tworzenie obiektu klasy gracza.
2. RealMoveTest – sprawdzenie poprawności ruchów bierką.
3. ChessPosition, parsePosition(...) – wczytywanie pozycji w notacji FEN.
4. Perft – zgodność możliwych ruchów dla podanej głębi.

# Znalezione błędy

1. Classa Perft czasem zwraca niepoprawny wynik ilości możliwych ruchów. Głównym problemem w naprawieniu tego błędu jest czas działania metod tej klasy – najdłużej może to trwać do godziny.
2. Przyciski wybory koloru szachownicy mają tekst wybranego koloru na sobie – nie jest on widoczny jeśli nie zaznaczyć jego za pomocą kursora.
3. W kilku przypadkach timer nie zresetował się po zamknięciu okna.

Każdy z tych błędów da się rozwiązać w przyszłości po bardziej dokładnym zbadaniu.s