**Politechnika Wrocławska**

**Wydział Informatyki i Telekomunikacji**

1. Kierunek: **inżynieria systemów (INS)**
   1. PRACA DYPLOMOWA
   2. INŻYNIERSKA

**Narzędzie wspomagające trening układania kostki Rubika**

Bartosz Piotr Krzysztoszek

Opiekun pracy

**Dr Inż. Dariusz Gąsior**

Słowa kluczowe: aplikacja, kostka Rubika, speedcubing, React.js

WROCŁAW (2024)

Streszczenie

Praca ma na celu opracowanie projektu oraz zaimplementowanie narzędzia wspomagającego trening układania kostki Rubika. Narzędzie jest przeznaczone głównie dla osób potrafiących już układać kostkę Rubika i chcących zwiększyć swoje umiejętności układania kostki metodą CFOP. Do wytworzenia narzędzia wykorzystano technologię React.js. Na początku zostało omówione wprowadzenie do dziedziny kostki Rubika oraz dyscypliny „speedcubing” w celu lepszego zrozumienia zamysłu narzędzia, a także zostały omówione istniejące rozwiązania dotyczące tej dziedziny. Projekt narzędzia został podzielony na wymagania, przypadki użycia, omówienie wykorzystanych technologii i architekturę, a następnie została opisana implementacja narzędzia. Po omówieniu implementacji został umieszczony opis samego narzędzia oraz sposoby korzystania z niego a także kroki niezbędne do jego uruchomienia. W zakończeniu zostały wypisane wnioski oraz potencjalne kierunki dalszego rozwoju narzędzia.

Abstract

Treść streszczenia po angielsku

**Spis treści**

[1. Wstęp 1](#_Toc156693678)

[1.1. Cel i zakres pracy 1](#_Toc156693679)

[2. Wprowadzenie do dziedziny 2](#_Toc156693680)

[2.1. Kostka Rubika 2](#_Toc156693681)

[2.1.1. Mechanika kostki Rubika 2](#_Toc156693682)

[2.1.2. Schemat kolorów kostki Rubika 2](#_Toc156693683)

[2.1.3. Układanie kostki Rubika 3](#_Toc156693684)

[2.1.4. Notacja 3](#_Toc156693685)

[2.1.5. Metody układania kostki Rubika 4](#_Toc156693686)

[2.2. Zawody 5](#_Toc156693687)

[2.2.1. Przebieg zawodów 5](#_Toc156693688)

[2.2.2. Procedura układania 6](#_Toc156693689)

[2.2.3. Sekwencje mieszające 6](#_Toc156693690)

[2.3. Opis problemu 7](#_Toc156693691)

[3. Przegląd istniejących rozwiązań 9](#_Toc156693692)

[3.1. Aplikacje typu „timer” 9](#_Toc156693693)

[3.1.1. Aplikacja „csTimer” 9](#_Toc156693694)

[3.1.2. Aplikacja „Cube Timer” 11](#_Toc156693695)

[3.2. Rozwiązania wspomagające trening metodą CFOP 11](#_Toc156693696)

[3.2.1. Strona „jperm.net” 12](#_Toc156693697)

[3.2.2. Aplikacja „csTimer” 13](#_Toc156693698)

[4. Projekt narzędzia 15](#_Toc156693699)

[4.1. Wymagania funkcjonalne 15](#_Toc156693700)

[4.2. Wymagania niefunkcjonalne 16](#_Toc156693701)

[4.3. Przypadki użycia 16](#_Toc156693702)

[4.4. Wybór technologii 20](#_Toc156693703)

[4.4.1. Główne technologie 20](#_Toc156693704)

[4.4.2. Baza danych 21](#_Toc156693705)

[4.4.3. Dane listy algorytmów metody CFOP **Błąd! Nie zdefiniowano zakładki.**](#_Toc156693706)

[4.4.4. Narzędzia wspomagające 21](#_Toc156693707)

[4.5. Architektura narzędzia 22](#_Toc156693708)

[4.5.1. Struktura aplikacji 22](#_Toc156693709)

[4.5.2. Architektura komponentów React 22](#_Toc156693710)

[4.5.3. Struktura bazy danych 23](#_Toc156693711)

[5. Implementacja 24](#_Toc156693712)

[6. Opis narzędzia 25](#_Toc156693713)

[7. Instalacja 26](#_Toc156693714)

[8. Testy 27](#_Toc156693715)

[8.1. Testy manualne 27](#_Toc156693716)

[8.2. Testy użyteczności 27](#_Toc156693717)

[9. Zakończenie 28](#_Toc156693718)

[9.1. Wnioski 28](#_Toc156693719)

[9.2. Kierunek dalszego rozwoju 28](#_Toc156693720)

[Bibliografia 29](#_Toc156693721)

[Spis rysunków 29](#_Toc156693722)

[Spis tabel 29](#_Toc156693723)

[Załącznik 29](#_Toc156693724)

# Wstęp

Kostka Rubika to jedna z najbardziej ikonicznych łamigłówek świata, która od wielu lat fascynuje i inspiruje miliony ludzi na całym świecie. Niezwykłe połączenie kolorów oraz elementów geometrycznych tej układanki stwarza nietypowe jak na zabawkę zależności, w tym potencjał do ogromnej liczby kombinacji, co stanowi wyzwanie będące pasją dla wielu entuzjastów. To co wyjątkowe w kostce Rubika w połączeniu z naturalną u ludzi chęcią rywalizacji sprawiło, iż od 2003 roku regularnie na całym świecie odbywają się zawody w dyscyplinie zwanej „speedcubing”. Obecnie coraz więcej osób, szczególnie młodych ale i nie tylko, zaczyna interesować się dziedziną speedcubingu. Jest to spowodowane rozwojem społeczności speedcuberów, która aktywnie dzieli się wiedzą i doświadczeniem poprzez różnorodne platformy internetowe. Youtube czy różne fora dyskusyjne pełne są poradników, analiz, a także relacji z zawodów, co sprzyja łatwiejszemu wejściu w świat speedcubingu dla nowych osób. Ponadto, organizacja międzynarodowych i lokalnych zawodów speedcubingowych dostarcza uczestnikom okazji do spotkania się, wymiany doświadczeń czy zawarcia nowych znajomości.

Z powodu coraz bardziej rosnącej popularności dyscypliny speedcubingu naturalną koleją rzeczy jest zwiększanie się zapotrzebowania na różnego rodzaju rozwiązania wspomagające. Tak jak każda sfera ludzkiego życia, speedcubing również ma swoje problemy wynikające ze specyfiki poszczególnych jego aspektów, z którymi muszą zmagać się osoby chcące rozwijać się w tej dziedzinie. Istotą speedcubingu jest bowiem stałe dążenie do lepszych wyników poprzez zarówno naukę efektywnych metod rozwiązywania kostki jak i zwiększanie szybkości w poruszaniu ściankami układanki poprzez praktykę. Z tego właśnie powodu, w niniejszej pracy zostanie podjęta próba stworzenia narzędzia wspomagającego proces treningu układania kostki Rubika na czas.

## Cel i zakres pracy

Celem pracy jest stworzenie projektu oraz implementacja narzędzia w formie aplikacji webowej mającej wspomagać trening układania kostki Rubika na czas. Narzędzie będzie dotyczyło jedynie oryginalnej kostki Rubika bez uwzględniania żadnych układanek podobnych. Narzędzie będzie tworzone dla osób, które potrafią już ułożyć kostkę Rubika oraz chcą trenować w celu osiągania lepszych rezultatów czyli w celu skrócenia czasu układania kostki. W szczególności narzędzie jest tworzone głównie dla osób układających lub chcących układać kostkę Rubika metodą CFOP czyli najpopularniejszą metodą zaawansowaną.

W pracy zostanie zawarty opis problemu treningu układania kostki Rubika na czas wraz z istotnymi aspektami dziedziny aby umożliwić pełne zrozumienie celu pracy. Zostaną również przedstawione istniejące rozwiązania warz z ich analizą. W pracy znajdzie się opis procesu powstawania narzędzia zawierający informacje na temat jego projektu, zastosowanych metod rozwiązania problemu, implementacji oraz opis sposobu jego użytkowania. Projekt narzędzia zostanie podzielony na wymagania funkcjonalne oraz niefunkcjonalne, przypadki użycia, omówienie wykorzystanych technologii oraz opis architektury narzędzia.

# Wprowadzenie do dziedziny

W tym rozdziale zostaną wyjaśnione zagadnienia oraz terminy związane z kostką Rubika, dyscypliną speedcubingu oraz oficjalnymi zawodami tej dyscypliny. Zostanie również szczegółowo omówiony problem wynikający ze specyfiki dziedziny, którego próba rozwiązania zostanie podjęta w dalszej części pracy. Należy zaznaczyć, że zostaną przedstawione tylko zagadnienia, które będą istotne z poziomu opisywanego problemu, natomiast pozostałe szczegóły dotyczące kostki Rubika oraz oficjalnych zawodów zostaną pominięte.

## Kostka Rubika

Kostka Rubika (ang. Rubik’s cube) to trójwymiarowa układanka wynaleziona pierwotnie w 1974 roku.

### Mechanika kostki Rubika

Kostka Rubika ma kształt sześcianu składającego się z 26 unikalnych miniaturowych sześcianów. Każdy z miniaturowych sześcianów zawiera ukryte przedłużenie znajdujące się wewnątrz kostki, które umożliwia połączenie się z innymi sześcianami oraz z rdzeniem, dzięki czemu sześciany mogą poruszać się jednocześnie. Środkowy miniaturowy sześcian każdej z sześciu ścian to jedynie pojedyncze kwadratowe pole przymocowane do rdzenia. Rdzeń zapewnia strukturę układanki, do której miniaturowe sześciany mogą się dopasować dzięki czemu mogą się one swobodnie poruszać bez możliwości odpadnięcia od reszty mechanizmu.

Miniaturowe sześciany można podzielić ze względu na liczbę ich zewnętrznych ścian. Są to:

* narożniki, czyli elementy o trzech zewnętrznych ścianach
* krawędzie, czyli elementy o dwóch zewnętrznych ścianach
* środkowe elementy, tak zwane centry, o jednej zewnętrznej ścianie

Każdy z miniaturowych sześcianów jest unikalny ze względu na kolory znajdujące się na jego ścianach. Wartą zauważenia właściwością kostki Rubika jest to, iż ze względu na przymocowanie centrów do rdzenia, centry pozostają względem siebie w tej samej pozycji, co dodatkowo sprawia, iż centry definiują jaki kolor powinien znajdować się na danej ścianie kostki Rubika

### Schemat kolorów kostki Rubika

Każda z sześciu ścian kostki Rubika ma unikalny kolor, przy czym warto wspomnieć , że większość istniejących kostek zachowuje następujący schemat:

* Ściany kostki mają kolory biały, żółty, niebieski, zielony, czerwony oraz pomarańczowy,
* pary kolorów biały i żółty, niebieski i zielony, czerwony i pomarańczowy umieszczone są na przeciwnych ścianach kostki,
* trzymając kostkę ścianką z kolorem białym do góry oraz ścianką z kolorem zielonym do siebie, ścianka znajdująca się po prawej stronie kostki jest czerwona.

### Układanie kostki Rubika

Wewnętrzny mechanizm kostki umożliwia niezależny obrót warstw sześcianów z każdej strony, mieszając w ten sposób kolory znajdujące się na poszczególnych ścianach. Stan kostki, w którym na każdej ścianie znajduje się tylko jeden kolor jest stanem docelowym, który oznacza rozwiązanie lub inaczej mówiąc ułożenie układanki. Trudność kostki Rubika polega na bardzo dużej liczbie możliwych do uzyskania kombinacji kolorów na ścianach przy tylko jednej kombinacji będącej rozwiązaniem. Liczba możliwych kombinacji jest określana na ponad 43 tryliony co oznacza, że losowe wykonywanie obrotów na kostce daje znikome prawdopodobieństwo na rozwiązanie jej. Z tego powodu do rozwiązywania pomieszanej kostki Rubika stosowane są metody o różnym podejściu oraz stopniu zaawansowania.

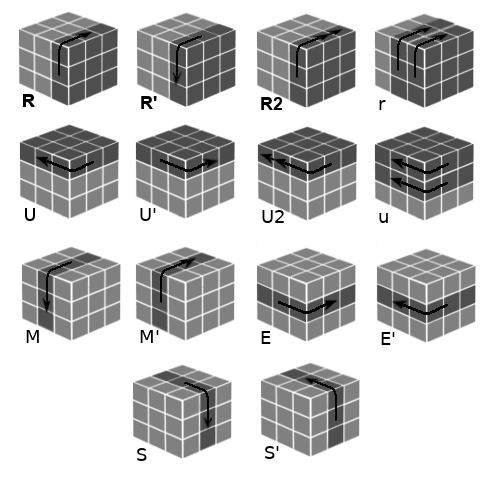
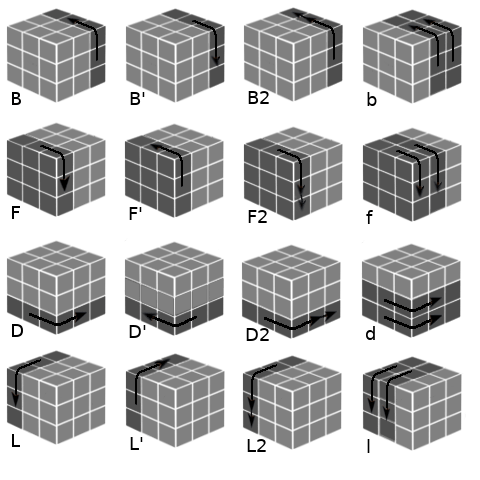
W kontekście metod układania kostki Rubika istotne jest pojęcie tak zwanego „algorytmu”. W żargonie osób układających kostki Rubika algorytmem określa się stałą sekwencję obrotów wykonywanych na kostce, które mają pożądany wpływ. Terminologia ta wywodzi się z matematycznego zastosowania algorytmu, czyli listy instrukcji wykonania zadania od danego stanu początkowego, poprzez dobrze określone kolejne stany, aż do pożądanego stanu końcowego. W tym momencie chciałbym zaznaczyć, że w dalszej części pracy określenie „algorytm” będzie wykorzystywane właśnie w znaczeniu sekwencji ruchów wykonywanych na kostce Rubika. Każda metoda układania kostki wykorzystuje własny zestaw algorytmów wraz z opisem, jaki efekt ma algorytm i kiedy można go zastosować, aby przybliżyć kostkę do rozwiązania. Szczególną cechą wielu algorytmów jest przeznaczenie do przekształcania tylko małej części kostki bez ingerencji w pozostałe części. Niektóre metody układania kostki, które polegają na stopniowym układaniu kolejnych części kostki mają również algorytmy, które wpływają na dużą liczbę elementów, jednak algorytmy te są stosowane w początkowych etapach układania, kiedy należy skupić się na ułożeniu mniejszej liczby elementów bez zwracania uwagi na położenie pozostałych elementów. Ważną kwestią dotyczącą algorytmów, o której należy wspomnieć jest to, iż może istnieć kilka algorytmów mających ten sam efekt. W praktyce algorytmy takie mogą różnić się pod względem liczby obrotów ale i wygody wykonywania co może przekładać się na szybkość ich wykonywania.

### Notacja

Najpopularniejszym sposobem opisu sekwencji obrotów kostki Rubika jest opracowana przez matematyka Davida Singmastera tak zwana „notacja Singmastera”. Notacja pozwala na zapis algorytmów w taki sposób, że możliwe jest ich stosowanie niezależnie od orientacji w jakiej trzymana jest kostka. Każdej ze ścian kostki przypisana jest duża litera:

* F – ściana przednia
* B – ściana tylna
* U – ściana górna
* D – ściana dolna
* L – ściana po lewej stronie
* R – ściana po prawej stronie

Litery użyte do oznaczeń ścian pochodzą od pierwszych liter angielskich słów oznaczających położenie (front, back, up, down, left, right). Zapis danej litery z notacji oznacza obrót jedną warstwą danej ściany o dziewięćdziesiąt stopni zgodnie z ruchem wskazówek zegara. Zapisanie symbolu apostrofu (′) po danej literze z notacji oznacza obrót jedną warstwą danej ściany o dziewięćdziesiąt stopni przeciwnie do ruchu wskazówek zegara. Zapisanie cyfry „2” po danej literze z notacji oznacza obrót jedną warstwą danej ściany o sto osiemdziesiąt stopni. Występują również obroty dwoma warstwami ścian zamiast jedną warstwą oznaczane małymi odpowiednikami liter notacji, dla których stosują się takie same reguły dotyczące kierunku obrotu. Dodatkowo w notacji występują obroty całą kostką wokół jednej z jej osi, czyli zmiany orientacji w jakiej trzymana jest kostka lub ujmując inaczej, obroty trzema warstwami naraz. Są one oznaczane małymi literami „x”, „y” oraz „z” i odpowiadają odpowiednio obrotom ściany prawej (R), obrotom warstwą ściany górnej (U) oraz obrotom warstwą ściany przedniej (F). Dla obrotów całą kostką stosuje się takie same reguły dotyczące kierunku obrotu co do obrotów jedną warstwą.



Rysunek . Graficzne przedstawienie notacji kostki Rubika

Istnieje również rozszerzenie notacji Singmastera, które zostało ogólnie przyjęte przez osoby układające kostki Rubika. Polega ono na dodaniu opisów dla obrotów środkowymi warstwami kostki. Każdej warstwie środkowej przypisana jest duża litera:

* M – warstwa pomiędzy warstwami ściany prawej (R) i lewej (L)
* E – warstwa pomiędzy warstwami ściany górnej (U) i dolnej (D)
* S – warstwa pomiędzy warstwami ściany przedniej (F) i tylnej (B)

Litery użyte do oznaczeń warstw pochodzą od pierwszych liter angielskich słów „middle”, „equator” oraz „standing”. Zapis litery „M” oznacza obrót warstwą o dziewięćdziesiąt stopni z góry na dół. Zapis litery „E” oznacza obrót warstwą o dziewięćdziesiąt stopni w prawo. Zapis litery „S” oznacza obrót warstwą o dziewięćdziesiąt stopni zgodnie z ruchem ściany przedniej (F). Zapisanie symbolu apostrofu (′) po danej literze z notacji oznacza obrót warstwą o dziewięćdziesiąt stopni w przeciwnym kierunku. Zapisanie cyfry „2” po danej literze z notacji oznacza obrót warstwą o sto osiemdziesiąt stopni.

Notacja Singmastera oraz jej rozszerzenie są zaprezentowane w formie graficznej na rysunku 2.1.

### Metody układania kostki Rubika

Najpopularniejszą metodą układania kostki Rubika dla początkujących jest metoda „LBL”. W skrócie polega ona na ułożeniu każdej warstwy kostki po kolei zaczynając od dolnej warstwy i kończąc na górnej warstwie. Metoda ta wymaga nauczenia się zaledwie kilku prostych algorytmów oraz dobrze opanowana pozwala zrozumieć ogólne podstawowe aspekty układania kostki. Pomimo że metoda nie jest efektywna pod względem liczby wymaganych do ułożenia kostki obrotów, pozwala ona na stopniowe wdrażanie większej liczby algorytmów co przekłada się na skrócenie liczby wymaganych obrotów.

Najpopularniejszym rozwinięciem metody LBL jest metoda zaawansowana o nazwie „CFOP” (można się również spotkać z nazwą „metoda Fridrich”). Nazwa metody bierze się z czterech etapów wymaganych do ułożenia kostki. Pierwszym etapem jest ułożenie krzyża dolnej warstwy. Drugi etap, tak zwany „F2L” (First Two Layers), polega na ułożeniu dolnej i środkowej warstwy układając w tym samym momencie narożnik dolnej warstwy i krawędź należącą do środkowej warstwy. Trzeci etap, tak zwany „OLL” (Orientation of the Last Layer) polega na zorientowaniu górnej warstwy za pomocą jednego algorytmu, tak aby górna ściana kostki była pokryta tylko jednym kolorem poprzez odpowiednie obrócenie wszystkich miniaturowych sześcianów. Czwarty etap, tak zwany „PLL” (Permutation of the Last Layer) polega na permutowaniu górnej warstwy za pomocą jednego algorytmu, tak aby górna warstwa została w całości ułożona. Metoda ta jest bardziej efektywna od metody podstawowej, jednakże ze względu na liczbę możliwych sytuacji, które mogą wystąpić w kolejnych etapach układania, wymaga nauczenia się ponad stu algorytmów. Dobra znajomość wszystkich wymaganych algorytmów oraz umiejętność szybkiego stosowania ich podczas układania pozwala na uzyskiwanie czasów ułożenia kostki nawet poniżej dziesięciu sekund.

Istotnym aspektem metod układania kostki Rubika, o którym warto wspomnieć jest orientacja kostki Rubika. Kostka Rubika może być bowiem trzymana w jednej z 24 orientacji, które można określać za pomocą koloru znajdującego się na górnej ścianie ułożonej kostki Rubika oraz koloru znajdującego się na przedniej ścianie kostki Rubika. Znając kolory górnej i przedniej ściany kostki Rubika jesteśmy w stanie określić jej orientację. Podczas rozpoczynania ułożenia kostki Rubika pierwszym krokiem jest zdecydowanie w jakiej orientacji lub grupie orientacji będziemy układać kostkę Rubika. Dla przykładu, używając metody LBL albo CFOP należy zdecydować, na której ścianie zostanie ułożony początkowy krzyż. W celu przyzwyczajenia się do kostki Rubika, poradniki dla osób początkujących zazwyczaj tłumaczą aby rozpoczynać ułożenia od białego krzyża co określa, że kostka Rubika będzie układana w grupie czterech orientacji, których kolor górnej ściany kostki Rubika to żółty. Jednakże możliwe jest ułożenie kostki Rubika za pomocą każdej metody w dowolnej orientacji, co sprawia, że wybór orientacji może zależeć od preferencji danej osoby.

## Zawody

„Speedcubing” to dyscyplina polegającą na jak najszybszym układaniu kostki Rubika i innych układanek logicznych z jej rodziny. Istotą tej dziedziny jest to, że aby układać układanki w najkrótszym czasie trzeba umieć szybko reagować na losowo natrafiane sytuacje. Każde pomieszanie układanki jest bowiem unikalne i liczba możliwych kombinacji jest na tyle duża, że nie jest możliwym nauczenie się każdej kombinacji na pamięć.

Organizowaniem oficjalnych zawodów w tej dyscyplinie zajmuje się organizacja międzynarodowa WCA. Oficjalne zawody przebiegają według zasad oraz regulacji ustalanych przez WCA.

### Przebieg zawodów

Na każde oficjalne zawody są określane układanki w jakich odbywa się rywalizacja, przy czym obecnie mogą być to tylko układanki wybrane z siedemnastu oficjalnych konkurencji. Warto wspomnieć, że standardowa kostka Rubika nie jest obowiązkową układanką, w której odbywa się rywalizacja na każdych zawodach, ale jest ona zdecydowanie najczęściej pojawiającą się układanką na zawodach. Zawody są podzielone na rundy. W każdej rundzie odbywa się rywalizacja w danej układance, przy czym rywalizacja w jednej układance może obejmować więcej niż jedną rundę.

Przebieg każdej rundy rozpoczyna się od zaniesienia swoich układanek przez zawodników do miejsca mieszania. Następnie zawodnicy udają się do poczekalni. Układanki są mieszane według ustalonych sekwencji mieszających, przy czym każdy zawodnik otrzymuje swoją układankę pomieszaną w ten sam sposób, jednakże każda próba rundy ma swoją sekwencje mieszającą. Pomieszane układanki są zabierane na bieżąco przez wyznaczonych do tego sędziów. Sędzia z układanką wywołuje odpowiedniego zawodnika z poczekalni oraz udaje się z nim do stanowiska startowego składającego się ze stołu na którym rozłożona jest mata wraz z profesjonalnym miernikiem czasu określanym jako „timer”. Następuje procedura układania układanki, po której zawodnik wraca do poczekalni i oczekuje na kolejne wywołanie, a sędzia zabiera układankę z powrotem do miejsca mieszania. Proces powtarza się do momentu, kiedy zawodnik wykona wszystkie próby.

Zajęte przez zawodnika miejsce w danej rundzie na oficjalnych zawodach dla większości układanek jest ustalane na podstawie średniego czasu trzech środkowych ułożeń kostki z pięciu prób.

### Procedura układania

Cała procedura układania jest nadzorowana przez sędziego, tak aby uniemożliwić potencjalne oszustwa. Układanka zakryta pudełkiem ustawiana jest przed zawodnikiem na macie przez sędziego. W momencie kiedy zawodnik zgłosi gotowość, sędzia odkrywa układankę. Rozpoczyna się tak zwana „inspekcja”, w której zawodnik ma piętnaście sekund na obejrzenie pomieszanej układanki przed rozpoczęciem układania. Czas inspekcji jest sprawdzany przez sędziego za pomocą osobnego stopera. Dodatkowo sędzia informuje zawodnika w momencie upływu ośmiu oraz dwunastu sekund czasu inspekcji. Zawodnik powinien rozpocząć układanie poprzez uruchomienie miernika czasu przed upływem czasu przeznaczonego na inspekcję. W przypadku, gdy zawodnik przekroczy limit piętnastu sekund inspekcji, zostaje nałożona kara w postaci dwóch sekund dodanych do końcowego wyniku ułożenia. Jeżeli zawodnik po przekroczeniu limitu piętnastu sekund inspekcji nie rozpocznie ułożenia przez kolejne dwie sekundy, ułożenie zostaje uznane za nieważne co jest określane poprzez skrót „DNF”, który rozwija się do angielskiej frazy „did not finished”. Zawodnik, który rozpoczął ułożenie powinien rozwiązać układankę, a następnie zatrzymać miernik czasu. Po ułożeniu sędzia zapisuje czas z miernika czasu. Jeżeli miernik czasu został zatrzymany, ale układanka nie została rozwiązana poprawnie, ułożenie zostaje uznane za nieważne co ponownie jest określane poprzez skrót „DNF”. Jeżeli miernik czasu został zatrzymany, a układanka znajduje się w stanie, w którym brakuje wykonania jednego ruchu aby została ona rozwiązana, zostaje nałożona kara w postaci dwóch sekund dodanych do końcowego wyniku. Warto wspomnieć, że kary dwóch sekund są kumulowane.

### Sekwencje mieszające

Sekwencje mieszające to sekwencje ruchów wykonywanych na układance w celu jej pomieszania zapisane za pomocą odpowiedniej dla danej układanki notacji. Dla standardowej kostki Rubika sekwencje mieszające wykorzystują notację Singmastera. Sekwencje mieszające są stosowane na zawodach w celu mieszania układanki w sposób obiektywny i całkowicie losowy oraz w celu uzyskania tego samego pomieszania układanki dla każdego zawodnika, tak by wyniki zawodników były porównywane sprawiedliwie. Według organizacji WCA aby układanka była pomieszana sprawiedliwie, powinna ona reprezentować jeden z wszystkich możliwych do uzyskania na układance stanów z wyłączeniem stanów, które wymagają liczby posunięć do ułożenia mniejszej niż dwa ruchy. Należy zaznaczyć, że aby pomieszane za pomocą sekwencji mieszającej układanki były identyczne, sekwencje mieszającą rozpoczyna się trzymając układankę w określonej orientacji. Dla standardowej kostki Rubika jest to orientacja trzymając białą ścianę skierowaną do góry oraz zieloną ścianę skierowaną do siebie.

Sekwencje mieszające są generowane losowo przed zawodami za pomocą oprogramowania komputerowego przez osobę odpowiedzialną za poprawny przebieg oficjalnych zawodów, czyli przez delegata WCA. Delegat WCA odpowiada za bezpieczne przechowanie sekwencji mieszających do momentu rozpoczęcia danej rundy zawodów, tak aby żaden z zawodników nie miał możliwości wcześniejszego zapoznania się ze sposobem pomieszania układanki.

## Opis problemu

W celu przygotowania się na zawody zawodnicy muszą trenować układanie kostki Rubika poza zawodami. Tworzy to potrzebę narzędzia umożliwiającego mierzenie czasu układania kostki Rubika. Profesjonalne mierniki czasu używane na zawodach są powszechnie dostępne do zakupu, jednakże ich ceny są często całkiem wysokie. Urządzenia te są bardzo dokładne w mierzeniu czasu, aczkolwiek jest to ich jedyna funkcja. Ze względu na koszt oraz ograniczoną funkcjonalność rodzi się potrzeba alternatywnego rozwiązania.

Istotnym aspektem treningu w każdej dziedzinie są uzyskiwane wyniki. W celu zwiększania efektywności treningu niezbędne jest podejmowanie odpowiednich decyzji o jego dalszym przebiegu. Aby decyzje były odpowiednie konieczne jest branie pod uwagę uzyskiwanych do tej pory wyników. Uzyskiwane wyniki bowiem obrazują to jak przebiegał dotychczasowy trening i stanowią jego ocenę. Tak samo w dziedzinie speedcubingu jest ważnym to aby dostosowywać trening do zawodnika odpowiednio, a informacja o tym jak to zrobić znajduje się w jego wynikach. Rodzi to potrzebę analizy uzyskiwanych przez zawodników czasów układania kostki Rubika.

Tak jak zostało to opisane w poprzednim podrozdziale, kostki Rubika są mieszane na zawodach za pomocą sekwencji mieszających. Powstaje jednak tutaj problem sprawiedliwego mieszania układanki do celów treningu. Mieszanie układanki samodzielnie przez zawodnika może prowadzić do pewnych tendencji lub powtarzających się schematów uzyskiwanych na kostce Rubika. Tego typu podejścia nie można uznać za obiektywne, a wyniki zawodnika nie byłby porównywalne względem wyników innych zawodników.

Wcześniej opisana metoda CFOP jest obecnie zdecydowanie najpopularniejszą metodą układania kostki Rubika wśród zawodników zawodów speedcubingu. Dwa ostatnie etapy metody CFOP, czyli OLL oraz PLL obejmują łącznie 78 przypadków, z czego każdy przypadek to inna kombinacja kolorów na ostatniej górnej warstwie. Nauczenie się algorytmów OLL oraz PLL pozwala na znaczne skrócenie czasów układania kostki Rubika, jednakże uczenie się ich nie jest proste. Każdy algorytm to sekwencja kilkunastu ruchów co sprawia, że informacji do przyswojenia jest dużo. Ważnym aspektem jest również to, iż nie wystarczy zapamiętać algorytmu zapisanego za pomocą notacji. Aby móc wykorzystać pełny potencjał i szybkość metody CFOP niezbędne jest wykorzystanie pamięci mięśniowej podczas nauki algorytmów. Oznacza to, że osoba jest nauczona danego algorytmu dopiero wtedy kiedy jest w stanie wykonać go automatycznie lub spontanicznie bez zastanawiania się nad nim, co wymaga pewnej ilości praktyki w stosowaniu algorytmu. Dodatkowo ważne jest też poprawne rozpoznawanie dla którego przypadku należy użyć którego algorytmu.

Biorąc pod uwagę wyżej rozpisane problemy można uświadomić sobie jak złożonym procesem może być trening układania kostki Rubika na czas. Rodzi się potrzeba narzędzia wspomagającego trening układania kostki Rubika, które umożliwiałoby generowanie sekwencji mieszających, mierzenie i zapisywanie rezultatów, podsumowywanie i analizę treningu, ale i również narzędzia skupionego na wspomaganiu nauki i ćwiczenia algorytmów.

# Przegląd istniejących rozwiązań

W tym rozdziale zostały przedstawione najpopularniejsze rodzaje istniejących rozwiązań w formie narzędzi informatycznych dotyczących wspomagania ogólnie pojętego treningu układania kostki Rubika oraz w szczególności treningu metody CFOP. Zostały przedstawione zarówno typowe cechy tych narzędzi wynikające ze specyfiki oraz zwyczajów dyscypliny jak i cechy najpopularniejszych przykładów narzędzi, dzięki którym się wyróżniają.

Obecnie w Internecie można znaleźć dużą liczbę aplikacji służących do wspomagania treningu układania kostki Rubika w różny sposób. Są to aplikacje w formie zarówno aplikacji desktopowych, stron internetowych jak i aplikacji mobilnych. Aplikacje te w zróżnicowany sposób podchodzą do samego procesu trenowania układania kostki Rubika oraz oferują różnego rodzaju funkcje.

## Aplikacje typu „timer”

Najbardziej powszechnym typem narzędzia wspomagającego trening układania kostki Rubika są aplikacje typu „timer”. Powszechnym założeniem tego typu aplikacji jest odwzorowanie funkcjonalności oraz doświadczeń jakie oferują fizyczne mierniki przeznaczone do mierzenia czasu układania układanek, w szczególności mierniki czasu używane na zawodach. Do głównych funkcjonalności aplikacji typu „timer” należą:

* uruchamianie i zatrzymywanie mierzenia za pomocą klawiszy klawiatury lub ekranu dotykowego smartphone’a,
* zapisywanie czasów ułożeń układanki,
* przedstawianie prostych danych statystycznych w celu podsumowania uzyskiwanych wyników,

Dodatkowo często pojawiającą się funkcją jest możliwość edycji i usuwania uzyskiwanych wyników.

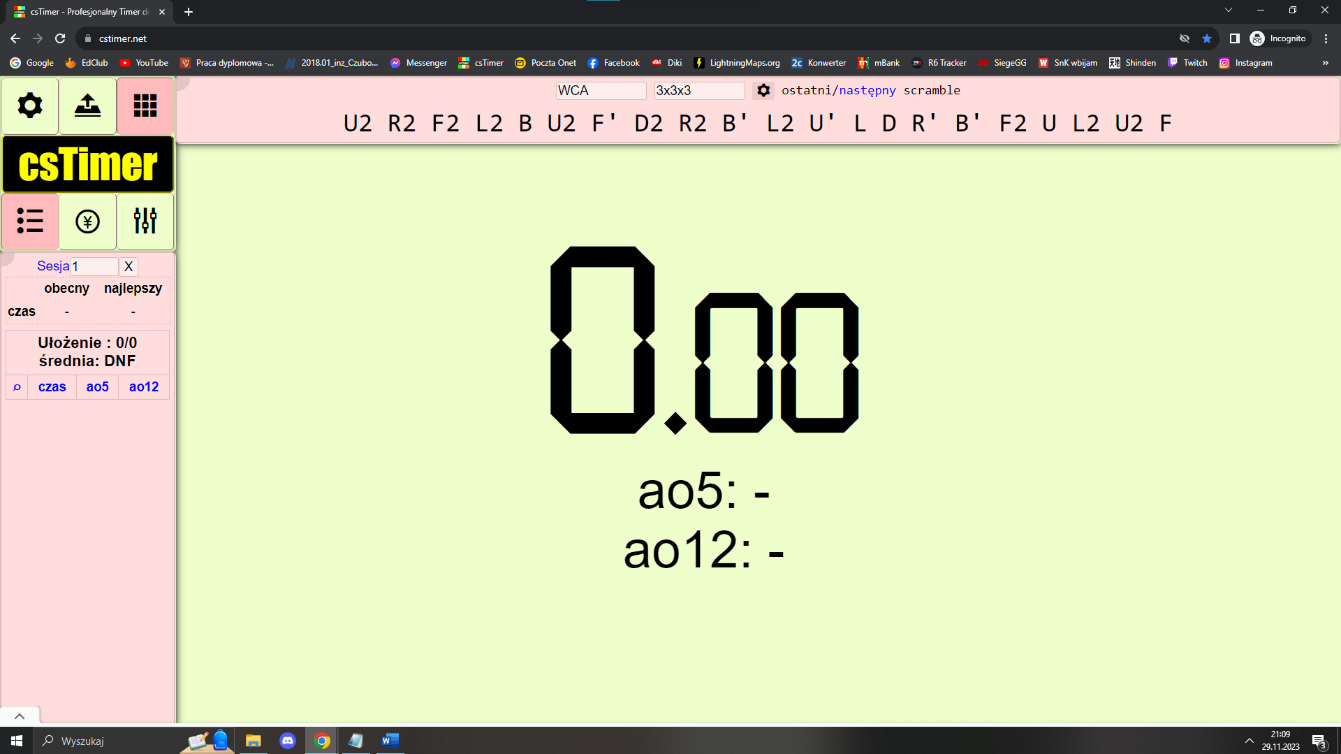
W wielu aplikacjach typu „timer” znajdują się również funkcjonalności mające na celu odwzorowanie procedury układania układanek na oficjalnych zawodach. Chodzi tutaj o funkcjonalności takie jak:

* wyświetlanie czasu inspekcji wraz z komunikatami o upływie ośmiu i dwunastu sekund,
* kary dwóch sekund dodanych do wyniku w przypadku przedłużenia czasu inspekcji lub w przypadku brakującego jednego ruchu do ułożenia układanki,
* oznaczenie ułożenia jako nieważne poprzez skrót „DNF”,
* generowanie i wyświetlanie sekwencji mieszających.

Mimo że wiele aplikacji typu „timer” skupia się głównie na wspomaganiu treningu zwykłej kostki Rubika to jednak wiele aplikacji jest również przeznaczona do treningu układania innych układanek, w szczególności układanek, w których odbywa się rywalizacja na oficjalnych zawodach, jednakże istnieją również aplikacje obsługujące szersze pule różnych układanek. Działa to w ten sposób, iż użytkownik wybiera, którą układankę chce trenować, a aplikacja generuje odpowiednią dla wybranej układanki sekwencję mieszającą oraz zapisuje uzyskiwane wyniki oddzielnie.

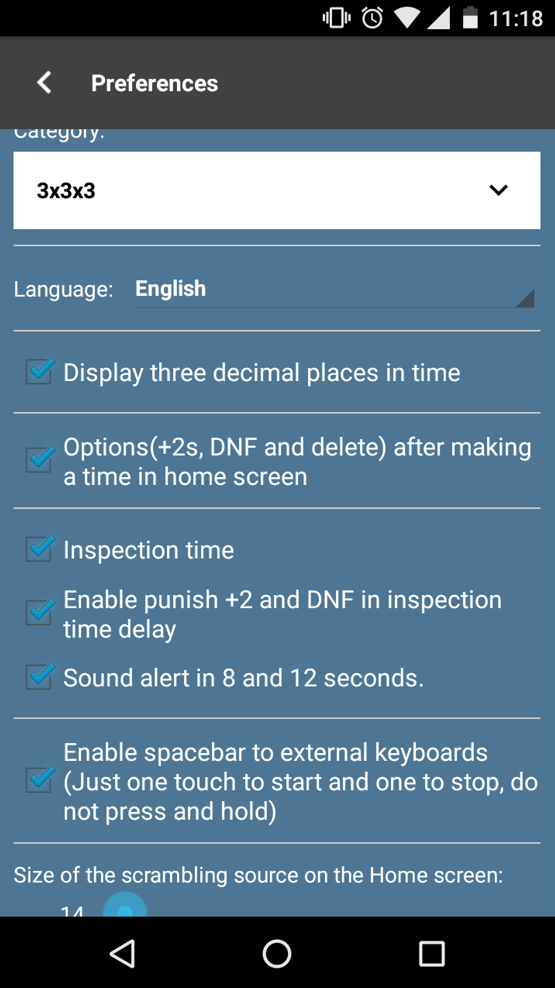
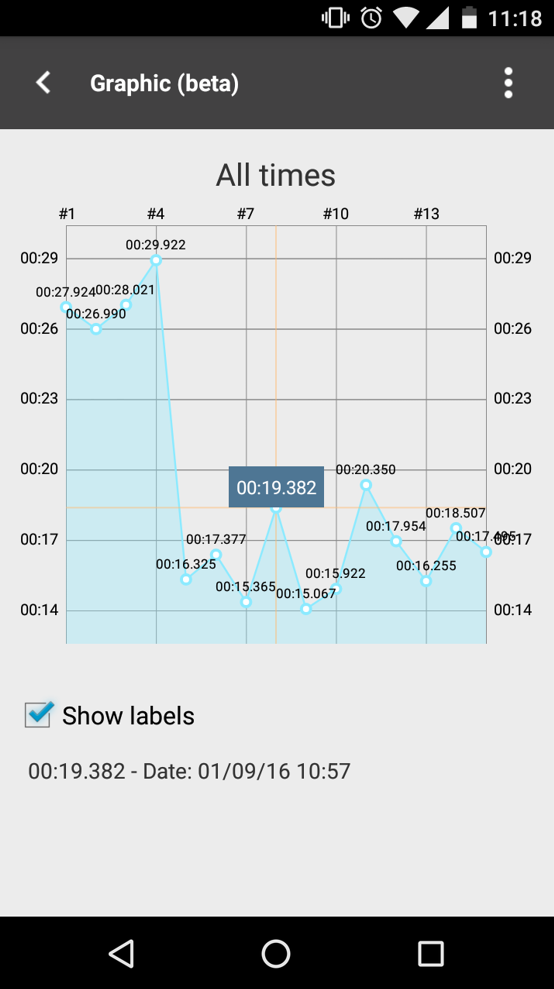
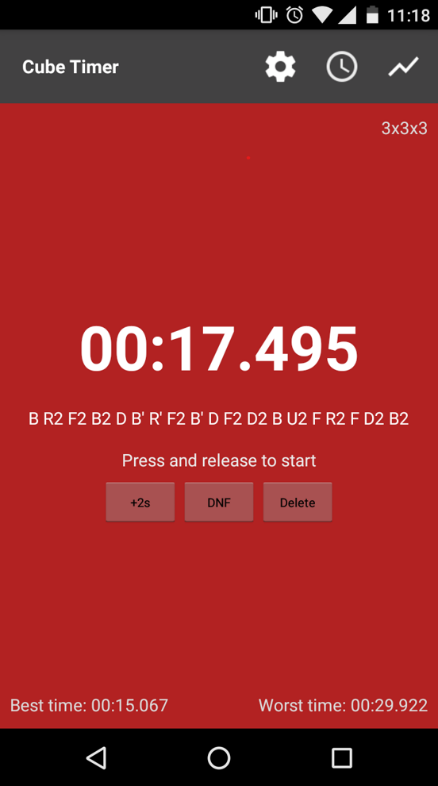
### Aplikacja „csTimer”

Najpopularniejszą aplikacją typu „timer” jest „csTimer” (rys. 3.1). Mimo, że aplikacja jest w formie aplikacji webowej, możliwe jest używanie jej również na urządzeniach mobilnych jako aplikacji natywnej. Aplikacja zawiera wszystkie wymienione wcześniej główne funkcjonalności aplikacji typu „timer” dotyczące zarówno mierzenia oraz zapisywania czasu z wyświetlaniem szeregu informacji statystycznych na temat uzyskiwanych wyników jak i funkcjonalności dotyczące procesu układania układanek na oficjalnych zawodach. Aplikacja obsługuje generowanie sekwencji mieszających dla kostki Rubika, dla układanek, w których odbywa się rywalizacja na oficjalnych zawodach, a także dla wielu innych mniej znanych oraz nietypowych układanek. Istotnym jest fakt tego, że aplikacja daje możliwość dostosowywania każdego aspektu jej działania. Oglądając interfejs graficzny aplikacji po raz pierwszy można odnieść wrażenie, że nie spełnia on dzisiejszych standardów o czym świadczy chociażby domyślny dobór kolorystyki, jednakże pomimo, że interfejs jest dość surowy, aplikacja daje użytkownikowi szereg opcji dostosowywania interfejsu takich jak wybieranie kolorów poszczególnych komponentów i dostosowywanie ich rozmiarów czy możliwość wybrania, które sekcje mają być wyświetlane w danym momencie. Dodatkowo aplikacja posiada dużą liczbę wbudowanych dodatkowych narzędzi wspomagających trening pod różnymi względami, takich jak wyświetlanie docelowego stanu układanki po pomieszaniu, możliwość połączenia z własnym fizycznym urządzeniem mierzącym czas, możliwość połączenia z elektroniczną kostką bluetooth, metronom i wiele więcej. Można śmiało stwierdzić, że aplikacja jest prosta ale jednocześnie pozwala na wysokie spersonalizowanie oraz dostarcza wiele dodatkowych funkcjonalności pomagających w treningu układania.



Rysunek . Aplikacja „csTimer”

Warto zwrócić dodatkową uwagę na rozmieszczenie elementów interfejsu na ekranie. Na samym środku znajduje się duże miejsce na uzyskiwany czas. Pozostałe elementy jak przyciski ustawień, sekwencja mieszająca czy lista wyników znajdują się dookoła centralnego obszaru. Tego typu układ pozytywnie wpływa na przejrzystość interfejsu oraz ułatwia szybkie i wygodne odczytanie rezultatu, a jednocześnie użytkownik ma łatwy dostęp do wszystkich dodatkowych opcji w każdym momencie. Dodatkowo, podczas mierzenia czasu ułożenia, wszystkie elementy poza wyświetlanym czasem są chowane. Jest to pomocne w minimalizowaniu zakłóceń i dodatkowym skupianiu uwagi na ułożeniu. Ogólnie rzecz biorąc, rozmieszczenie elementów w interfejsie zostało zaplanowane tak, aby dostarczyć intuicyjne i przejrzyste środowisko do pomiaru czasu, co pozytywnie wpływa na doświadczenie użytkowania z aplikacji.



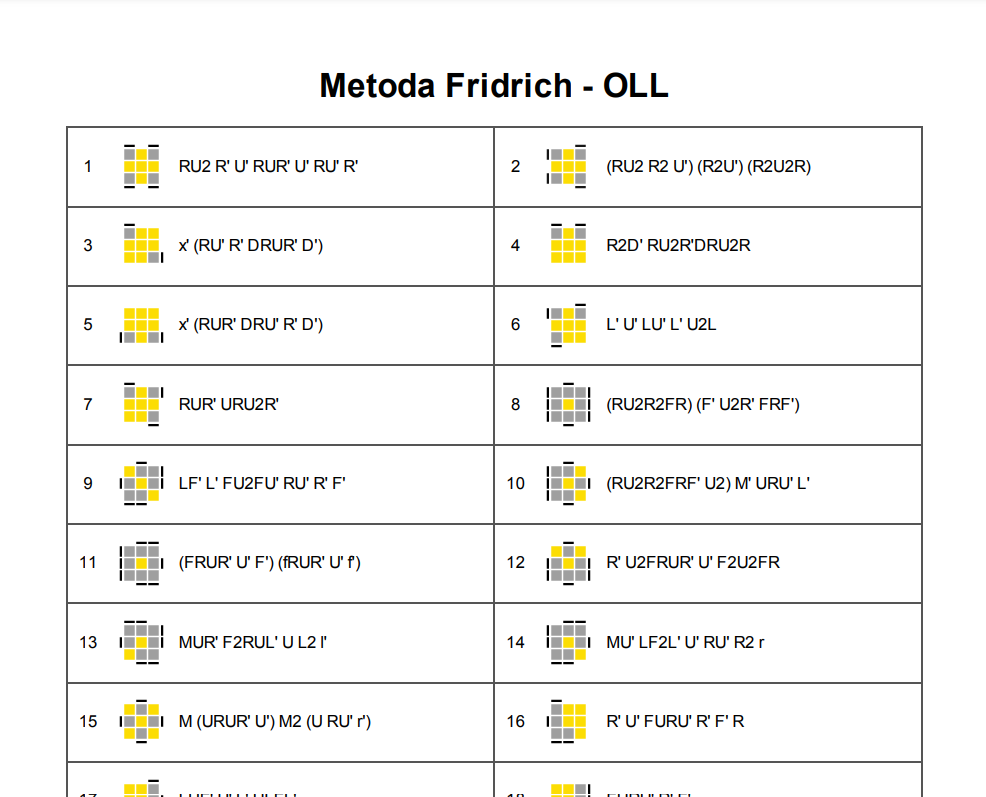
Rysunek . Aplikacja „Cube Timer”

### Aplikacja „Cube Timer”

Inną, jedną z popularnych aplikacji mobilnych typu „timer” jest aplikacja „Cube Timer” (rys. 3.2). Aplikacja jest dostępna do pobrania w sklepie Google Play. Tak samo jak poprzednia omawiana aplikacja, zawiera ona wszystkie główne funkcjonalności aplikacji typu „timer”. Aplikacja jednak nie jest tak rozbudowana i nie zawiera dodatkowych narzędzi. Działanie aplikacji ogranicza się do mierzenia oraz zapisywania czasu, generowania sekwencji mieszających tylko dla układanek w których odbywa się rywalizacja na oficjalnych zawodach oraz wyświetlania prostych statystyk wraz z wykresem przedstawiającym uzyskane kolejno wyniki. Aplikacja dodatkowo oferuje panel opcji, gdzie użytkownik może dokonać niewielkiej personalizacji. Wyróżniającym się elementem aplikacji jest panel, na którym wyświetlone są dane pobierane ze strony organizacji WCA na temat zbliżających się oficjalnych zawodów. Panel ten jednak ogranicza się tylko do wyświetlania informacji. Można stwierdzić, że aplikacja jest prosta oraz nie daje wielu możliwości użytkownikowi, natomiast dobrze zapewnia podstawowe funkcje.

## Rozwiązania wspomagające trening metodą CFOP

Próbując znaleźć rozwiązanie wspomagające naukę lub trening metody CFOP w Internecie, bardzo łatwo natrafić w pierwszej kolejności na strony internetowe zawierające listy algorytmów do poszczególnych etapów tej metody (rys. 3.3). Takie listy zazwyczaj ograniczone są jedynie do wyświetlania przedstawionej sytuacji na kostce Rubika, nazwy danego przypadku oraz samego algorytmu czyli sekwencji ruchów. Niektóre z takich stron służą także jako poradnik poszczególnych etapów metody CFOP. Rozwiązanie takie jest jednak interaktywne w bardzo niewielkim stopniu. Dodatkowo można też natrafić na analogiczne rozwiązania w formie aplikacji mobilnych.

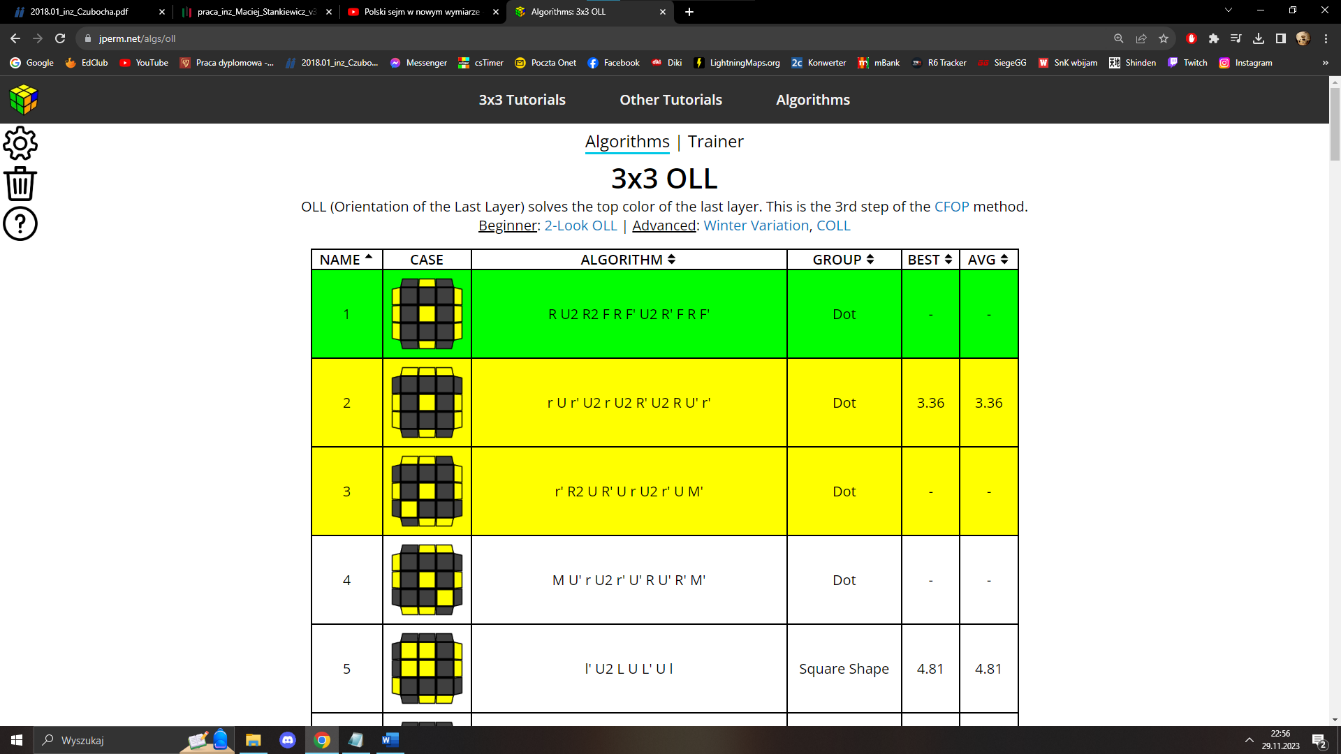


Rysunek . Lista algorytmów OLL, strona speedcubing.pl

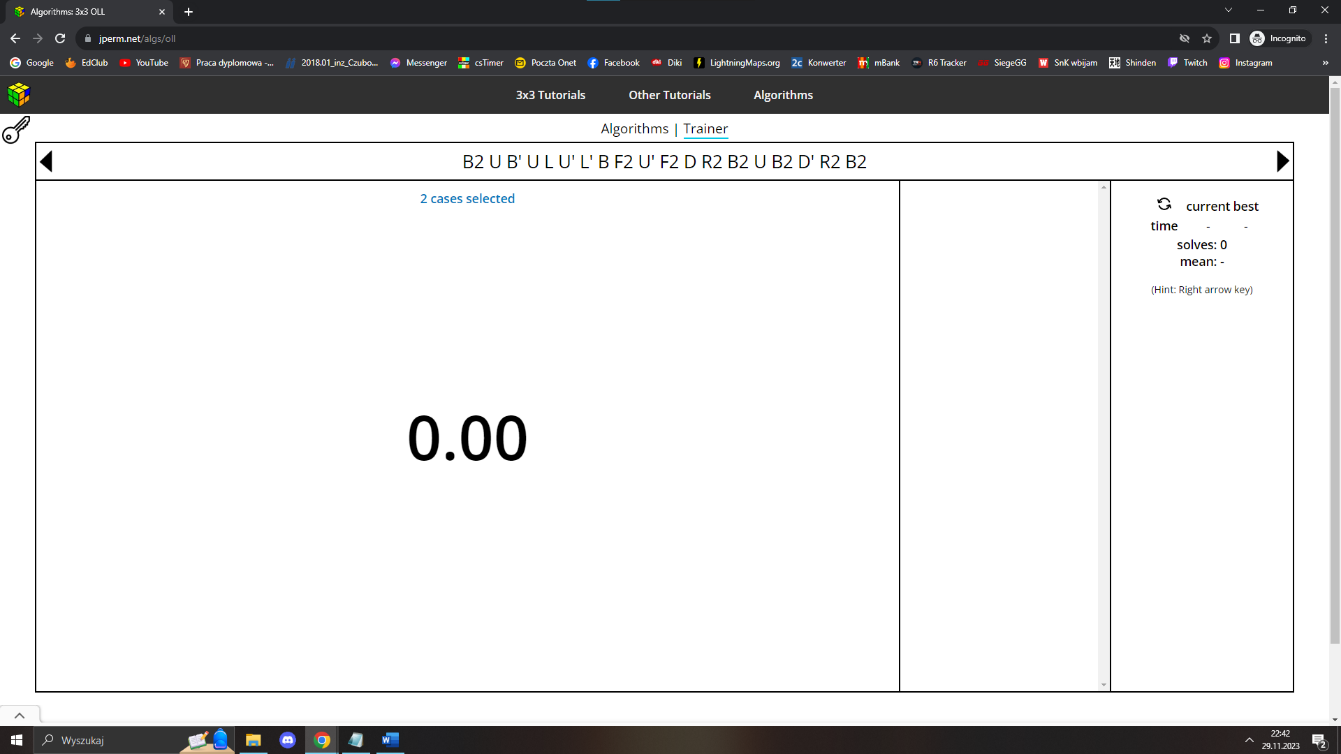
### Strona „jperm.net”

Bardziej interaktywne podejście oferuje strona „jperm.net”. Na stronie można znaleźć zarówno poradniki do różnych metod układania układanek, w tym poradnik do metody CFOP, jednak bardziej istotnym elementem są przedstawione tutaj listy algorytmów (rys. 3.4). Listy algorytmów znajdujące się na stronie dotyczą etapów metody CFOP ale również innych metod oraz układanek. Listy te pozwalają na kilka szczególnych interakcji. Użytkownik może oznaczać wybrane algorytmy jako „nieznane”, „obecnie uczone” lub „nauczone” poprzez kliknięcie i zmianę koloru wiersza listy z danym przypadkiem, co pozwala na śledzenie postępów w ich nauce. Użytkownik może również decydować jakiej sekwencji ruchów używać dla danego przypadku poprzez wybór sekwencji ruchów z proponowanych sekwencji lub wprowadzenie własnej sekwencji, co umożliwia personalizacje listy i możliwość pełnej modyfikacji, zamiast przedstawiania „gotowej” listy algorytmów bez możliwości zmian.

Oprócz samej listy algorytmów jest również zakładka „Trainer” (rys. 3.5), w której użytkownik może ćwiczyć wybrane przypadki na takiej samej zasadzie jak odbywa się to w aplikacjach typu „timer”. Narzędzie losuje jeden z przypadków, który został oznaczony jako „obecnie uczony”, generuje i wyświetla sekwencję mieszającą mającą doprowadzić kostkę Rubika do stanu z wylosowaną sytuacją. Użytkownik uruchamia i zatrzymuje mierzenie czasu za pomocą klawiszy klawiatury w celu zmierzenia czasu wykonywania algorytmu dla wylosowanej sytuacji, oraz wyświetlane są dane statystyczne w celu podsumowania dotychczasowego treningu. W ten sposób użytkownik może trenować wykonywanie wybranych algorytmów pod względem szybkości ich wykonywania, a dostarczane użytkownikowi dane statystyczne stanowią podsumowanie uzyskiwanych wyników treningu, dzięki któremu użytkownik może bardziej efektywnie dostosować dalszy trening.



Rysunek . Strona jperm.net, lista algorytmów



Rysunek . Strona jperm.net, zakładka "Trainer"

### Aplikacja „csTimer”

Inne interaktywne podejścia do uczenia się i trenowania metody CFOP można znaleźć również we wspomnianej już aplikacji „csTimer”. W tej aplikacji możliwe jest generowanie sekwencji mieszających, które mają doprowadzić kostkę Rubika do jednej z możliwych sytuacji, które mogą pojawić się w poszczególnych etapach metody CFOP, jednakże w tej aplikacji nie ma żadnej możliwości wyboru, które sytuacje użytkownik chce w danym momencie trenować. Poza tym ta aplikacja nie zawiera funkcjonalności dotyczących listy algorytmów.

## Wnioski

Analizując przegląd istniejących rozwiązań można dojść do kilku wniosków. Przede wszystkim, do tej pory powstało bardzo niewiele narzędzi, które są przeznaczone do wspomagania nauki metody CFOP, a żadne z nich nie obejmuje jednocześnie wspomagania klasycznego treningu układania kostki Rubika.

Innym aspektem jest to, iż każda lista algorytmów metody CFOP, którą można znaleźć w Internecie wykorzystuje wizualizacje sytuacji na kostce Rubika dla algorytmów w celu pokazania tego, kiedy należy stosować dany algorytm. Istotny jest fakt, że owe wizualizacje znajdują się w formie grafik co sprawia, że nie ma możliwości na dostosowanie orientacji wizualizowanych kostek Rubika, a wyświetlane są jedynie stany kostki Rubika tylko w jednej wybranej wcześniej orientacji. Umożliwienie zmiany orientacji kostek Rubika będących wizualizacjami przypadków mogłoby pozytywnie wpłynąć na efektywność poznawania i uczenia się kolejnych algorytmów przez użytkowników. Każdy użytkownik mógłby bowiem wybrać orientację w jakiej aktualnie znajduje się jego kostka w celu łatwego znalezienia wymaganego algorytmu do rozwiązania natrafionej sytuacji. Dodatkowo każdy użytkownik mógłby wybrać orientację pasującą do jego preferowanego koloru.

Kolejnym wnioskiem, który należy wziąć pod uwagę jest charakterystyczne rozmieszczenie elementów interfejsu na ekranie. Ze względu na specyfikę dziedziny speedcubingu, której istotą jest osiąganie jak najkrótszych czasów, interfejsy rozpatrywanych narzędzi poświęcają dużo przestrzeni środkowej części ekranu właśnie na liczony czas, podczas gdy reszta narzędzi znajduje się na obrzeżach ekrany dookoła czasu.

# Projekt narzędzia

W tym rozdziale zostanie przedstawiony projekt narzędzia tak, by możliwe było określenie kluczowych wymagań co do narzędzia oraz sprecyzowanie struktury jego działania. Dzięki temu, proces implementacji i rozwoju narzędzia będzie mógł przebiegać w konkretny sposób, a żaden istotny z poziomu planu narzędzia wcześniej określony aspekt nie zostanie pominięty. Zostaną sprecyzowane zarówno wymagania funkcjonalne jak i wymagania niefunkcjonalne w celu określenia najważniejszych elementów narzędzia oraz najważniejszych aspektów. Zostaną przedstawione przypadki użycia narzędzia, tak aby tworzenie go mogło odbywać się w kierunku zapewnienia praktycznej użyteczności. Określone zostaną technologie niezbędne do implementacji zarówno poszczególnych elementów narzędzia jak i jego całości, a także technologie, które będą wykorzystane jedynie w procesie implementacji jako rozwiązania wspierające sam proces realizacji. Zostanie również przedstawiona architektura narzędzia.

## Wymagania funkcjonalne

Poniżej znajdują się wymagania funkcjonalne. Każdemu wymaganiu został nadany priorytet określony liczbą w skali od 1 do 3, gdzie liczba 3 oznacza najwyższy priorytet.

1. Mierzenie czasu rozwiązywania kostki Rubika przez użytkownika – 3
2. Wyświetlanie na bieżąco czasu rozwiązywania kostki Rubika przez użytkownika – 2
3. Wyświetlenie uzyskanego czasu rozwiązywania kostki Rubika przez użytkownika – 3
4. Zapisywanie danych dotyczących każdego rezultatu rozwiązywania kostki Rubika przez użytkownika – 3
5. Możliwość edycji oraz usuwania danych dotyczących każdego rezultatu przez użytkownika – 3
6. Obsługiwanie mierzenia czasu za pomocą klawiszy klawiatury przez użytkownika – 3
7. Możliwość zatwierdzenia lub odrzucenia rezultatu przez użytkownika po wykonaniu ułożenia – 2
8. Generowanie sekwencji mieszającej dla kostki Rubika – 3
9. Generowanie sekwencji mieszającej dla losowo wybranego „algorytmu” - 3
10. Wyświetlanie sekwencji mieszającej – 3
11. Wyświetlanie prostych danych statystycznych dotyczących uzyskiwanych rezultatów użytkownika – 3
12. Możliwość dodawania kar dwóch sekund dla rezultatów przez użytkownika – 2
13. Możliwość oznaczenia rezultatu jako „dnf” – 2
14. Możliwość włączenia czasu piętnastu sekund „inspekcji” przez użytkownika – 3
15. Wyświetlanie ostrzeżeń podczas „inspekcji” dla ośmiu oraz dwunastu sekund – 2
16. Sygnał dźwiękowy podczas „inspekcji” dla ośmiu oraz dwunastu sekund – 2
17. Wyświetlanie listy „algorytmów” metody „CFOP” dla etapów „OLL” i „PLL” wraz z wizualizacją stanu kostki Rubika – 3
18. Możliwość dostosowania orientacji kostki Rubika dla jakiej są wizualizowane stany dla każdego „algorytmu” – 1
19. Możliwość zmiany i zapisywanie stanu każdego „algorytmu” dotyczącego nauki przez użytkownika – 2
20. Możliwość wyboru trenowanych „algorytmów” – 3
21. Możliwość dostosowania wyświetlanych informacji dla użytkownika – 1
22. Sekcja pomocy wyjaśniająca podstawy korzystania z narzędzia – 1
23. Zapisywanie wybranych przez użytkownika ustawień narzędzia - 1

## Wymagania niefunkcjonalne

Poniżej znajdują się wymagania niefunkcjonalne określone ogólnie wraz z opisem wyjaśniającym wymaganie w odniesieniu do narzędzia.

1. **Użyteczność**

Interfejs graficzny narzędzia powinien być intuicyjny i pozwalający na łatwe nawigowanie oraz korzystanie. Podczas korzystania z narzędzia powinno być jasne, które elementy tylko wyświetlają informacje, a które pozwalają na interakcję. Funkcje narzędzia powinny być nietrudne w zrozumieniu dla osób zaznajomionych z dziedziną układania kostki Rubika na czas, a zapoznawanie się z nimi nie powinno wymagać większego wysiłku. Nieoczywiste interakcje, takie jak korzystanie z klawiszy klawiatury podczas korzystania z narzędzia powinny być dodatkowo opisane w narzędziu, tak by użytkownik mógł łatwo się o nich dowiedzieć i przyswoić ich używanie.

1. **Dostępność**

Narzędzie powinno być możliwe do wykorzystania poprzez wszystkie najpopularniejsze obecnie przeglądarki internetowe.

1. **Niezawodność**

Narzędzie powinno być odporne na błędy.

1. **Wydajność**

Narzędzie powinno być wydajne. Wszystkie operacje odbywające się w narzędziu powinny natychmiastowo zwracać wynik, a użytkownik nie powinien oczekiwać w żadnym momencie korzystania z narzędzia po jego uruchomieniu.

1. **Łatwość utrzymania**

Struktura narzędzia powinna pozwalać na łatwe dodawanie nowych funkcji i rozwijanie istniejących. Kod narzędzia powinien być czytelny oraz podzielony na odpowiednie części.

## Przypadki użycia

Poniżej zostały przedstawione przypadki użycia narzędzia przez użytkownika w celu określenia możliwości wykorzystania narzędzia. Dla każdego przypadku została podana jego nazwa, warunek wstępny, scenariusz oraz warunek końcowy.

1. **Wykonanie ułożenia kostki Rubika**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie oraz ma ze sobą ułożoną kostkę Rubika. Narzędzie wyświetla ekran stopera w zakładce „3x3”.

Scenariusz główny:

1. Użytkownik miesza kostkę Rubika stosując sekwencję mieszającą wyświetloną na ekranie narzędzia.
2. Użytkownik przytrzymuje klawisz spacji na klawiaturze.
3. Czas stopera zmienia kolor na zielony oraz pozostałe elementy interfejsu zostają ukryte.
4. Użytkownik puszcza klawisz spacji na klawiaturze i rozpoczyna układanie kostki Rubika.
5. Mierzenie czasu zostaje uruchomione.
6. Użytkownik kończy układanie kostki Rubika po czym naciska klawisz spacji na klawiaturze.
7. Mierzenie czasu zostaje zatrzymane oraz zostają wyświetlone przyciski odpowiadające za określenie poprawności ułożenia.
8. Użytkownik ponownie naciska klawisz spacji na klawiaturze lub wybiera opcję „OK”.
9. Rezultat zostaje zapisany do bazy danych jako ułożenie poprawne.
10. Przyciski odpowiadające za określenie poprawności ułożenia zostają ukryte, a pozostałe elementy interfejsu ponownie się pojawiają

Alternatywny przebieg:

3.A.1. Czas stopera zmienia kolor na żółty oraz pozostałe elementy interfejsu zostają ukryte.

3.A.2. Użytkownik puszcza klawisz spacji na klawiaturze.

3.A.3. Czas stopera zmienia kolor na czerwony, pozostałe elementy interfejsu zostają ukryte oraz rozpoczyna się odliczanie 15 sekund czasu inspekcji.

3.A.4. Użytkownik przytrzymuje klawisz spacji na klawiaturze.

3.A.5. Czas stopera zmienia kolor na zielony.

3.A.6. Użytkownik puszcza klawisz spacji na klawiaturze i rozpoczyna układanie kostki Rubika.

3.A.4.A.1 Użytkownik naciska klawisz „esc” na klawiaturze.

3.A.4.A.2 Elementy interfejsu ponownie się pojawiają oraz inspekcja zostaje anulowana.

6.A.1. Użytkownik naciska klawisz „esc” na klawiaturze.

6.A.2. Rezultat zostaje zapisany do bazy danych jako ułożenie nieukończone.

6.A.3. Elementy interfejsu ponownie się pojawiają.

8.A.1. Użytkownik wybiera opcję „+2”.

8.A.2. Rezultat zostaje zapisany do bazy danych jako ułożenie z karą dwóch sekund.

8.A.3. Przyciski odpowiadające za określenie poprawności ułożenia zostają ukryte, a pozostałe elementy interfejsu ponownie się pojawiają

8.B.1. Użytkownik naciska klawisz „esc” na klawiaturze lub wybiera opcję „DNF”.

8.B.2. Rezultat zostaje zapisany do bazy danych jako ułożenie nieukończone.

8.B.3. Przyciski odpowiadające za określenie poprawności ułożenia zostają ukryte, a pozostałe elementy interfejsu ponownie się pojawiają

Warunek końcowy: Rezultat jest zapisany i jest widoczny na liście wyników.

1. **Wykonanie treningu algorytmów**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie oraz ma ze sobą ułożoną kostkę Rubika. Narzędzie wyświetla ekran stopera w zakładce „PLL” lub „OLL”.

Scenariusz główny:

1. Użytkownik miesza kostkę Rubika stosując sekwencję mieszającą wyświetloną na ekranie narzędzia.
2. Użytkownik przytrzymuje klawisz spacji na klawiaturze.
3. Czas stopera zmienia kolor na zielony oraz pozostałe elementy interfejsu zostają ukryte.
4. Użytkownik puszcza klawisz spacji na klawiaturze i rozpoczyna wykonywanie algorytmu.
5. Mierzenie czasu zostaje uruchomione.
6. Użytkownik kończy wykonywanie algorytmu na kostce Rubika po czym naciska klawisz spacji na klawiaturze.
7. Mierzenie czasu zostaje zatrzymane, a rezultat zostaje zapisany do bazy danych.
8. Pozostałe elementy interfejsu ponownie się pojawiają

Alternatywny przebieg:

6.A.1. Użytkownik naciska klawisz „esc” na klawiaturze.

6.A.2. Mierzenie czasu zostaje anulowane.

6.A.3. Elementy interfejsu ponownie się pojawiają.

Warunek końcowy: Rezultat jest zapisany i jest widoczny na liście wyników.

1. **Usunięcie wszystkich wyników**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran stopera w jednej z zakładek. Opcja wyświetlania okna wyników jest włączona.

Scenariusz główny:

1. Użytkownik wybiera opcję „Delete all results” w oknie wyników.
2. Zostaje wyświetlone okienko z komunikatem z prośbą o potwierdzenie.
3. Użytkownik wybiera opcję „delete all results”.
4. Okienko znika, a wszystkie wyniki dla danej zakładki stopera zostają usunięte z bazy danych.

Alternatywny przebieg:

3.A.1. Użytkownik wybiera opcję „close” dla okienka.

3.A.2. Okienko znika.

Warunek końcowy: Wszystkie wyniki dla danej zakładki stopera są usunięte i nie są widoczne na liście wyników.

1. **Usunięcie jednego wyniku**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran stopera w jednej z zakładek. Opcja wyświetlania okna wyników jest włączona.

Scenariusz główny:

1. Użytkownik wybiera jeden z wyników w oknie wyników.
2. Zostaje wyświetlone okienko z informacjami na temat wyniku.
3. Użytkownik wybiera opcję „delete result”.
4. Okienko znika, a wybrany wynik zostaje usunięty z bazy danych.

Alternatywny przebieg:

3.A.1. Użytkownik wybiera opcję „close” dla okienka.

3.A.2. Okienko znika.

Warunek końcowy: Wybrany wynik jest usunięty i nie jest widoczny na liście wyników.

1. **Zmiana szczegółów wyniku**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran stopera w jednej z zakładek. Opcja wyświetlania okna wyników jest włączona.

Scenariusz główny:

1. Użytkownik wybiera jeden z wyników w oknie wyników.
2. Zostaje wyświetlone okienko z informacjami na temat wyniku.
3. Użytkownik zaznacza lub odznacza jedną z opcji „+2 (inspection)”, „+2 (one turn)” lub „DNF”.
4. Zmiana dotycząca wyniku zostaje zapisana w bazie danych.
5. Użytkownik wybiera opcję „close” dla okienka.
6. Okienko znika.

Alternatywny przebieg:

3.A.1. Użytkownik wybiera opcję „close” dla okienka bez dokonywania zmiany.

3.A.2. Okienko znika.

Warunek końcowy: Zmiana dotycząca wyniku jest zapisana oraz widoczna na liście wyników.

1. **Wyświetlenie informacji na temat wyniku**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran stopera w jednej z zakładek. Opcja wyświetlania okna wyników jest włączona.

Scenariusz główny:

1. Użytkownik wybiera jeden z wyników w oknie wyników.
2. Zostaje wyświetlone okienko z informacjami na temat wyniku.

Warunek końcowy: Okienko z informacjami na temat wyniku jest wyświetlone.

1. **Zmiana stanu uczenia się algorytmu**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran algorytmów w jednej z zakładek „PLL” lub „OLL”.

Scenariusz główny:

1. Użytkownik wybiera jeden z algorytmów na liście algorytmów.
2. Użytkownik klika w wizualizację kostki Rubika dla wybranego algorytmu.
3. Stan uczenia się algorytmu zostaje zmieniony na kolejny stan zaawansowania oznaczany przez kolor tła algorytmu, czyli z „nieznane”, na „obecnie uczone”, z „obecnie uczone” na „nauczone” oraz z „nauczone” z powrotem na „nieznane”.
4. Zmiana stanu uczenia się algorytmu zostaje zapisana w bazie danych.

Warunek końcowy: Zmiana dotycząca stanu uczenia się algorytmu jest zapisana oraz widoczna na liście algorytmów.

1. **Zmiana ustawień**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran stopera lub algorytmów.

Scenariusz główny:

1. Użytkownik wybiera opcję „settings”.
2. Zostaje otworzone okienko z listą dostępnych ustawień.
3. Użytkownik zaznacza lub odznacza wybrane ustawienia.
4. Zmiana dotycząca ustawień zostaje zapisana w bazie danych.
5. Użytkownik wybiera opcję „close” dla okienka.
6. Okienko znika, a wybrany wynik zostaje usunięty z bazy danych.

Alternatywny przebieg:

3.A.1. Użytkownik wybiera opcję „close” dla okienka bez dokonywania zmian.

3.A.2. Okienko znika.

Warunek końcowy: Zmiana dotycząca ustawień jest zapisana oraz zastosowana.

1. **Wyświetlenie sekcji pomocy**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran stopera lub algorytmów.

Scenariusz główny:

1. Użytkownik wybiera opcję „help”.
2. Zostaje wyświetlone okienko z sekcją pomocy.

Warunek końcowy: Okienko z sekcją pomocy jest wyświetlone.

1. **Zmiana orientacji wyświetlanych kostkek Rubika na liście algorytmów**

Aktor: Użytkownik

Warunki wstępne: Użytkownik ma uruchomione narzędzie. Narzędzie wyświetla ekran algorytmów.

Scenariusz główny:

1. Użytkownik wybiera jeden z sześciu kolorowych przycisków pod sekcją „top color”.
2. Orientacja wyświetlanych kostek Rubika na liście algorytmów zostaje dostosowana do wybranego koloru górnej ściany, a kolorowe przyciski pod sekcją „front color” zostają ograniczone do czterech dostępnych.
3. Użytkownik wybiera jeden z czterech kolorowych przycisków pod sekcją „front color”.
4. Orientacja wyświetlanych kostek na liście algorytmów zostaje dostosowana do wybranego koloru górnej i przedniej ściany.

Warunek końcowy: Orientacja wyświetlanych kostek Rubika na liście algorytmów jest dostosowana do wybranych kolorów górnej i przedniej ściany.

## Wybór technologii

### Główne technologie

W trakcie procesu tworzenia projektu narzędzia kluczowym jest wybór głównych technologii, na których oparte będzie narzędzie. Docelowo narzędzie ma funkcjonować jako aplikacja webowa tak by możliwe było uruchomienie jej w szybki sposób jedynie poprzez przejście pod odpowiedni adres url strony. Biorąc pod uwagę dotychczasowe założenia na temat narzędzia, sensownym wydaje się być wybranie technologii służącej do tworzenia tak zwanych „Single Page Applications” czyli aplikacji jednostronnicowych, opartych tylko o jeden plik HTML i wymagających tylko jednego załadowania podczas uruchamiania aplikacji. Z tych powodów zdecydowano w pracy na wybranie technologii React.js.

React.js jest darmową biblioteką języka Java Script rozwijaną głównie przez firmę Meta. Biblioteka ta służy do budowania dynamicznych i interaktywnych interfejsów, w tym między innymi właśnie do budowy jednostronicowych aplikacji. React.js jest obecnie uważany za najpopularniejszy framework front-ednowy wśród deweloperów co zawdzięcza swojej wydajności, modularności oraz deklaratywnej składni. Istotnym aspektem tego frameworku jest koncepcja komponentów. Komponent jest odizolowanym elementem interfejsu, w którym mogą być umieszczane inne komponenty co pozwala na łatwe zarządzanie strukturą interfejsu. Każdy komponent może być wykorzystywany wielokrotnie, a przepływ danych między komponentami odbywa się tylko od komponentów nadrzędnych do podrzędnych co ułatwia kontrolowanie przepływu danych i ich śledzenie. Komponenty można podzielić na komponenty stanowe i bezstanowe. Komponenty bezstanowe służą do renderowania interfejsu użytkownika na podstawie przekazanych im informacji, natomiast komponenty stanowe posiadają stan i są odpowiedzialne za zarządzanie nim. Każdy komponent jest definiowany w kodzie w postaci funkcji języka Java Script, która przyjmuje pojedynczy argument „props” oraz zwraca kod komponentu, zazwyczaj w postaci kodu JSX. JSX to rozszerzenie języka Java Script pozwalające na definiowanie znaczników HTML w kodzie Java Script co znacznie upraszcza proces tworzenia elementów interfejsu. Kolejnym istotnym aspektem jest wykorzystanie przez React.js wirtualnej wersji obiektowego modelu dokumentu (ang. Document Object Model, DOM) do skutecznego zarządzania aktualizowaniem interfejsu. Zamiast bezpośrednio zarządzać rzeczywistym drzewem DOM, React.js porównuje wirtualny DOM utworzony w swojej pamięci z rzeczywistym i wprowadza zmiany tylko w niezbędnych miejscach, co w efekcie daje zwiększoną wydajność.

Ze względu na użycie frameworka React.js należy również wspomnieć o konieczności wykorzystania języka HTML, dzięki któremu możliwe będzie zdefiniowanie struktury strony dla aplikacji.

W celu zapewnienia obecnych standardów w narzędziu zostanie wykorzystany język arkuszy stylów CSS. Jest to język pozwalający na określanie właściwości znaczników HTML, takich jak kolor, rozmiar, położenie i wiele innych. Zastosowanie go umożliwia oddzielenie treści od prezentacji i zdefiniowanie wyglądu strony w jednym miejscu.

### Baza danych

Biorąc pod uwagę przyjęte założenia na temat narzędzia można stwierdzić, że nie jest koniecznym aby dane były zbierane na serwerze aplikacji webowej, a dobrym wyjściem jest zbieranie ich lokalnie na urządzeniu użytkownika bez konieczności ciągłego przesyłania, co dodatkowo może wpłynąć na zwiększenie wydajności oraz uniezależni działanie aplikacji od połączenia z siecią. Istnieje kilka rozwiązań stosowanych powszechnie w różnych aplikacjach w takich właśnie sytuacjach. Jednym z nich jest „local storage”.

Local storage to określenie pamięci używanej przez przeglądarki internetowe wykorzystywanej przez Java Script. Zaletą local storage jest przechowywanie pamięci nawet po zamknięciu przeglądarki w formie par „klucz - wartość”. Dzięki temu możliwe jest zapisywanie różnych informacji dotyczących aplikacji webowych. Local storage jest dosyć prostym narzędziem, jednakże posiada ono kilka znaczących ograniczeń. Rozmiar pamięci jaka może być przechowana za pomocą local storage dla większości współczesnych przeglądarek wynosi 5 MB na domenę. Dodatkowo jedynym typem danych jaki może być przechowywany przez local storage jest string, co wymaga dodatkowych konwersji typów danych za każdym razem kiedy chcemy zapisać dane innego typu niż tekstowe, co w pewnym stopniu może tworzyć utrudnienia. Ze względu na swoje cechy local storage jest idealnym wyborem na zapisywanie informacji, których liczba jest stała i nieduża, takich jak ustawienia.

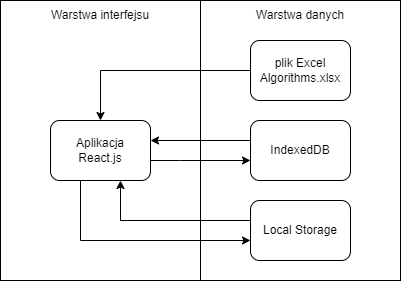
Z perspektywy tworzonego w pracy narzędzia, w którym zapisywane rekordy uzyskiwanych rezultatów będą się stale zwiększać, limit rozmiaru pamięci może stanowić barierę. Z tego powodu kolejnym wybranym rozwiązaniem będzie IndexedDB. Jest to przeglądarkowa baza danych NoSQL. Pozwala ona na przechowywanie większych ilości danych i umożliwia bardziej zaawansowane operacje niż local storage.

### Narzędzia wspomagające

W celu zapewnienia wydajności pracy nad tworzeniem narzędzia konieczne jest zastosowanie rozwiązań wspomagających, które usprawnią pracę z kodem.

Jako zintegrowane środowisko programistyczne (and. Integrated Development Environment, IDE) zostanie wykorzystane narzędzie WebStorm 2023 od firmy JetBrains. Narzędzie to ułatwia pracę z kodem poprzez między innymi dynamiczne oznaczanie poszczególnych części kodu odpowiednim kolorem, dynamiczne sugerowanie wpisywanych nazw wraz z autouzupełnianiem, zaawansowane narzędzia do debugowania czy integrację z popularnymi systemami kontroli wersji. Jest to jedno z wielu IDE firmy JetBrains przeznaczone głównie do pracy w języku Java Script oraz w innych powiązanych z Java Scipt technologii, w tym w React.js.

Jako system kontroli wersji zostanie zastosowany Git. Jest to obecnie najpopularniejsze rozwiązanie umożliwiające zapisywanie postępów pracy tworzenia oprogramowania. Dzięki Git’owi praca nad tworzeniem narzędzia będzie przebiegać bardziej efektywnie. Git pozwoli na bardziej bezpieczne wprowadzanie zmian oraz kolejnych elementów aplikacji, bez obawy o utratę działającej wersji aplikacji. Możliwe również będzie łatwe zapisywanie postępów pracy w zdalnym repozytorium GitHub co da możliwość pracy nad aplikacją z dowolnego urządzenia.



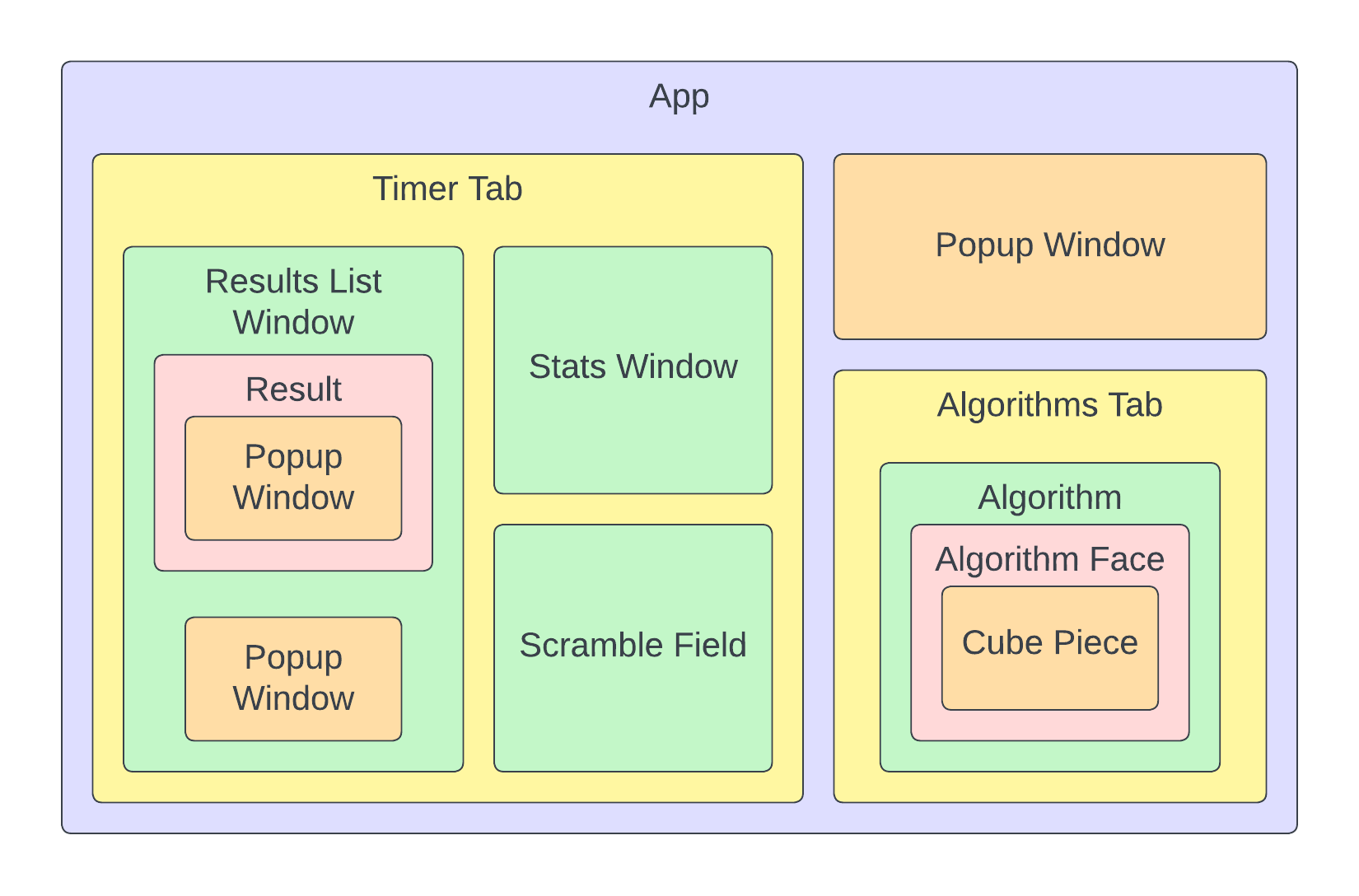
Rysunek . Architektura narzędzia

## Architektura narzędzia

### Struktura aplikacji

Elementy składające się na narzędzie można podzielić na dwie warstwy struktury: warstwę interfejsu oraz warstwę danych. Diagram obrazujący architekturę narzędzia został przedstawiony na rysunku 4.1. Poniżej znajduje się opis każdego przepływu danych między warstwami narzędzia.

1. Dane z pliku arkusza kalkulacyjnego Algorithms.xlsx są importowane do aplikacji w momencie jej uruchamiania.
2. W momencie uruchamiania aplikacji dane dotyczące uzyskiwanych wyników z bazy danych IndexedDB są przesyłane do aplikacji, a także są one przesyłane później w każdym momencie wystąpienia zmiany w bazie danych.
3. Kiedy użytkownik zatrzymuje czas ułożenia kostki Rubika lub wykonania algorytmu, wynik jest zapisywany do bazy danych IndexedDB
4. W momencie uruchamiania aplikacji dane dotyczące ustawień, a także dane dotyczące stanu listy algorytmów są przesyłane do aplikacji z narzędzia local storage
5. Kiedy użytkownik dokona zmian ustawień lub dokona zmiany dotyczącej listy algorytmów, zmiana jest zapisywana w narzędziu local storage



Rysunek . Diagram komponentów React

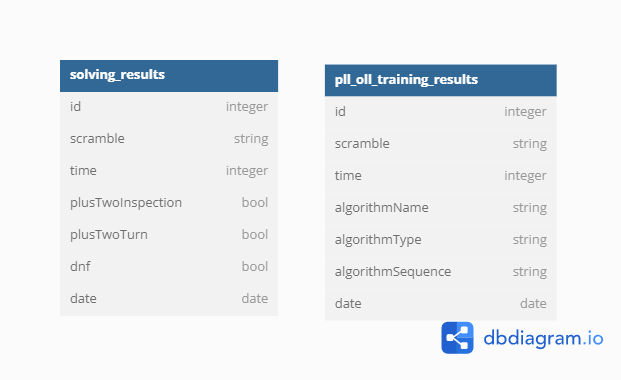
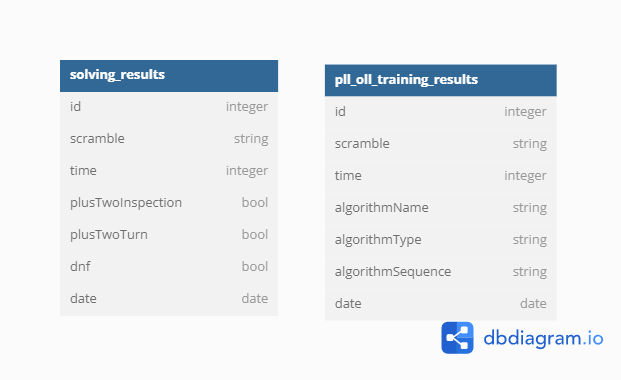
### Architektura komponentów React

Na rysunku 4.2 został przedstawiony diagram komponentów React. Głównym komponentem jest komponent aplikacji App. Ten komponent będzie pozostawał wyrenderowany przez cały czas działania aplikacji od momentu jej uruchomienia co sprawi, iż będzie mógł on pełnić funkcję miejsca zarządzania globalnym stanem aplikacji. Dzięki temu, zewnętrzne dane będą mogły być w łatwy sposób udostępniane do innych komponentów z jednego miejsca. Wpływa to na poprawę elastyczności aplikacji w zakresie wprowadzania zmian. Komponentem podrzędnym do komponentu App jest Popup Window. Komponent ten jest wykorzystany wielokrotnie. Jego działanie będzie polegało na wyświetlaniu podanej mu zawartości w formie wyskakującego okna. To użycie komponentu jako komponent podrzędny do komponentu App będzie odpowiedzialne za wyświetlanie okna ustawień oraz za wyświetlanie okna pomocy.

Kolejnymi komponentami są Timer Tab oraz Algorithms Tab. Będą to dwie oddzielne zakładki w formie ekranów aplikacji.

W zakładce Timer Tab użytkownik będzie mógł trenować układanie oraz wykonywanie algorytmów. Komponent Timer Tab zawiera podrzędne komponenty takie jak Stats Window, który będzie wyświetlał statystyki dotyczące uzyskiwanych wyników, Scramble Field, który będzie wyświetlał sekwencje mieszającą oraz Results List Window, który będzie wyświetlał listę uzyskanych wyników. Komponent Results List Window będzie zawierał komponent Popup Window przeznaczone do zatwierdzania operacji usuwania wszystkich uzyskanych wyników, tak by użytkownik nie mógł wykonać tej operacji przypadkowo. Lista uzyskanych wyników będzie tak naprawdę składała się z wielu kopii komponentu Result. Każdy komponent Result będzie jednym wynikiem na liście, który po kliknięciu otworzy komponent Popup Window zawierający szczegóły danego wyniku.

W zakładce Algorithms Tab będzie wyświetlona lista algorytmów metody CFOP. Każdy algorytm na liście będzie kopią komponentu Algorithm. Komponent Algorithm będzie wyświetlał więc dane na temat algorytmu. Za wyświetlanie wizualizacji stanu kostki będzie odpowiadał komponent Algorithm Face, a ten będzie złożony z wielu podrzędnych komponentów Cube Piece.



Rysunek 4.3 Schemat bazy danych IndexedDB

### Struktura bazy danych

Baza danych IndexedDB będzie przeznaczona do przechowywania danych dotyczących uzyskiwanych przez użytkownika rezultatów. Rezultaty jakie będzie uzyskiwał użytkownik można podzielić na dwa typy: rezultaty zwykłego trenowania układania kostki Rubika oraz rezultaty trenowania algorytmów. Te typy rezultatów różnią się pod względem informacji jakie mogą być istotne, dlatego struktura bazy danych będzie oparta o dwie różne tabele faktów. Atrybuty obu tabel są widoczne na rysunku 4.3.

Tabela solving\_results zawiera informacje na temat klasycznych ułożeń kostki Rubika. W każdym istniejącym rozwiązaniu kary nakładane na ułożenie były ograniczone jedynie do kary dwóch sekund oraz kary dnf, jednakże podjąłem decyzję aby umieścić w narzędziu możliwość dodawania oddzielnie kary dwóch sekund za przedłużenie czasu inspekcji oraz kary dwóch sekund z powodu brakującego ruchu do ukończenia ułożenia. Możliwe będzie nałożenie obu kar na jeden wynik, a wtedy dodatkowe sekundy będą się kumulowały. Dzięki temu narzędzie będzie w większym stopniu odzwierciedlać procedurę układania kostki Rubika na zawodach. Atrybuty tabeli:

* id – identyfikator rekordu
* scramble – wygenerowana sekwencja mieszająca
* time – uzyskany czas w jednostce 10 milisekund
* plusTwoInspection – wartość logiczna oznaczająca czy na wynik została nałożona kara dwóch sekund z powodu przedłużenia czasu inspekcji
* plusTwoTurn – wartość logiczna oznaczająca czy na wynik została nałożona kara dwóch sekund z powodu brakującego ruchu do ukończenia ułożenia
* dnf – wartość logiczna oznaczająca czy wynik został uznany za nieważny z powodu nieułożenia kostki Rubika
* date – data uzyskania wyniku

Tabela pll\_oll\_training\_results zawiera informacje na temat wyników wykonanych algorytmów. Atrybuty algorithmName, algorithmType oraz algorithmSequence zawierają informacje na temat algorytmu, który był ćwiczony. Jednym z rozpatrywanych przeze mnie rozwiązań było zamienienie tych atrybutów na pojedynczy identyfikator algorytmu, dzięki któremu można by uzyskać te same informacje z danych otrzymanych z pliku excel jednocześnie nie powielając ich niepotrzebnie, jednakże zdecydowałem pozostawić ową tabelę w obecnej formie. Jest to spowodowane chęcią oddzielenia uzyskanych już wyników od innych danych. Dzięki temu, możliwa będzie modyfikacja danych na temat algorytmów, na przykład w przyszłości przy dalszym rozwijaniu narzędzia, bez obawy o naruszenie danych uzyskanych wyników. Atrybuty tabeli pll\_oll\_training\_results:

* id – identyfikator rekordu
* scramble – wygenerowana sekwencja mieszająca
* time – uzyskany czas w jednostce 10 milisekund
* algorithmName – nazwa wykonanego algorytmu
* algorithmType – typ wykonanego algorytmu, pll lub oll
* algorithmSequence – sekwencja rozwiązująca algorytmu
* date – data uzyskania wyniku

# Implementacja

Dwa ostatnie etapy metody układania kostki Rubika CFOP, czyli OLL oraz PLL obejmują łącznie 78 przypadków, z czego każdy przypadek to inna kombinacja kolorów na kwadracikach znajdująca się na ostatniej górnej warstwie. Stan ostatniej warstwy kostki Rubika musi zostać zwizualizowany dla każdego przypadku aby osoba ucząca się kolejnych algorytmów mogła rozpoznawać przypadki i wiedzieć w jakiej sytuacji należy stosować dany algorytm. Wizualizacje te mogłyby zostać umieszczone w tworzonym narzędziu w formie grafik tak jak są one umieszczone w istniejących rozwiązaniach, jednakże nie dawałoby to możliwości na dostosowanie orientacji wizualizowanych kostek Rubika, a wyświetlało stan kostki tylko w jednej wybranej wcześniej orientacji. W celu umożliwienia zmiany orientacji wizualizowanej kostki Rubika, dane dotyczące wizualizacji kostki Rubika powinny zostać zapisane w formie liczbowej, tak by każdy z kolorów był reprezentowany przez liczbę. Pozwoli to na manipulację i zmianę kolorów na wizualizacjach jedynie poprzez odpowiednią zmianę koloru przypisanego do liczby.

Ze względu na konieczność zapisania danych na temat algorytmów tj. sekwencji rozwiązujących oraz stanu kostki Rubika samodzielnie, zdecydowano na wykorzystanie technologii programu Microsoft Excel. Dane zostaną umieszczone w pojedynczym pliku .xlsx w formie tabelki. Zawartość pliku .xlsx będzie wczytywana do aplikacji i przetwarzana. Ręczne wpisywanie kolejnych liczb reprezentujących wygląd kostki dla każdego z 78 przypadków może zostać uproszczone dzięki wykorzystaniu funkcji formatowania warunkowego. Komórki w tabeli będą zmieniały kolor swojego tła w zależności od liczby jaka zostanie w nich wpisana, co ułatwi weryfikację poprawności wpisanych danych oraz znajdywanie błędów. Kolory komórek będą odpowiadały jednej z możliwych orientacji na kostce Rubika.

# Opis narzędzia

# Instalacja

# Testy

## Testy manualne

## Testy użyteczności

# Zakończenie

Cel prazy można uznać za osiągnięty. Pomimo natrafiania na różnego rodzaju problemy podczas procesu implementacji, zaimplementowane narzędzie spełnia wszystkie wcześniej postawione założenia i wymagania. Dodatkowo podczas tworzenia pracy zdołałem poszerzyć swoją wiedzę oraz umiejętności tworzenia aplikacji webowych w technologii React.js.

Bibliografia

|  |  |
| --- | --- |
| [1] |  |
| [2] |  |
| [3] |  |
| [4] | Graficzne przedstawienie notacji kostki Rubika https://rcube.pl/pelna-notacja dostęp 29.11.2023. |

Spis rysunków

Spis tabel

Załącznik