

**Dokumentacja projektu**

Przedmiot: **Programowanie**

Tytuł projektu: Aplikacja rozwiązująca zagadkę Sudoku.

Prowadzący: Wykonawca:

*Dr Marek Jaszuk Krzysztof Bigos W60156*

*Konrad Haduch W61513*

*Dawid Hamerla W60174*

Semestr i symbol kierunku:4IIZ

Grupa: GP01, GP02

Grupa na platformie BB:

GP01/15

Rzeszów, 2020

1. Opis założeń projektu (wymagania biznesowe)
2. Specyfikacja wymagań – tutaj napisać listę wymagań z podziałem na wymagania funkcjonalne i niefunkcjonalne
3. Diagram przypadków użycia z wyróżnieniem aktorów projektu, oraz przypadków użycia odzwierciedlających wymagania funkcjonalne. Do rysowania diagramu należy wykorzystać dedykowane do tego celu oprogramowanie.
4. Harmonogram realizacji projektu (diagram Gantta). W przypadku projektu realizowanego zespołowo należy przypisać poszczególne zadania do członków zespołu.
5. Opis techniczny projektu – w jaki sposób projekt został zrealizowany (struktura kodu programu).
6. Prezentacja warstwy użytkowej projektu (widoki ekranu z opisem ich użytkowania).
7. Projekt powinien być realizowany z wykorzystaniem wybranego systemu kontroli wersji. Należy podać link do repozytorium
8. Raporty z testów – powinny być przeprowadzone testy jednostkowe (dwa przykładowe testy). Moduł testowy powinien być w repozytorium. Tutaj umieścić zrzut ekranu z pozytywnie przeprowadzonymi testami.
9. Materiały źródłowe – wskazanie literatury i materiałów źródłowych wykorzystanych przy realizacji projektu (w tym tutoriale internetowe).
10. Do kodu programu powinna być stworzona dokumentacja poprzez system komentarzy dokumentujących. Załącznikiem dokumentacji projektu powinna być dokumentacja kodu w postaci HTML (wygenerowana przy pomocy programu Doxygen). Wygenerowaną dokumentację należy udostępnić poprzez dysk internetowy podając link w niniejszym dokumencie.

# Opis założeń projektu(wymagania biznesowe)

Aplikacja w założeniu ma pomagać w rozwiązywaniu zagadki logicznej Sudoku. Użytkownik otrzyma możliwość skorzystania z pomocy przy rozwiązywaniu łamigłówki, po uprzednim wprowadzeniu przez niego stanu faktycznego, aplikacja sprawdzi czy wprowadzone dane są poprawne a następnie, jeśli nie napotka błędu w otrzymanych danych rozwiąże problem. Wyżej wymieniony program, może być wykorzystany do uczenia się z jego pomocą rozwiązywania Sudoku o różnej skali trudności. Może być również wykorzystany w miejscach gdzie, wymagana jest szybka analiza wyników, np. różnego rodzaju turnieje, w których sędziowie przygotowując zagadki, potrzebują szybko poprawnych wyników, by następnie móc porównać je z wynikami uzyskanymi przez zawodników. Za pomocą aplikacji można też testować czy wymyślony przez nas układ będzie w ogóle możliwy do rozwiązania.

# Specyfikacja wymagań

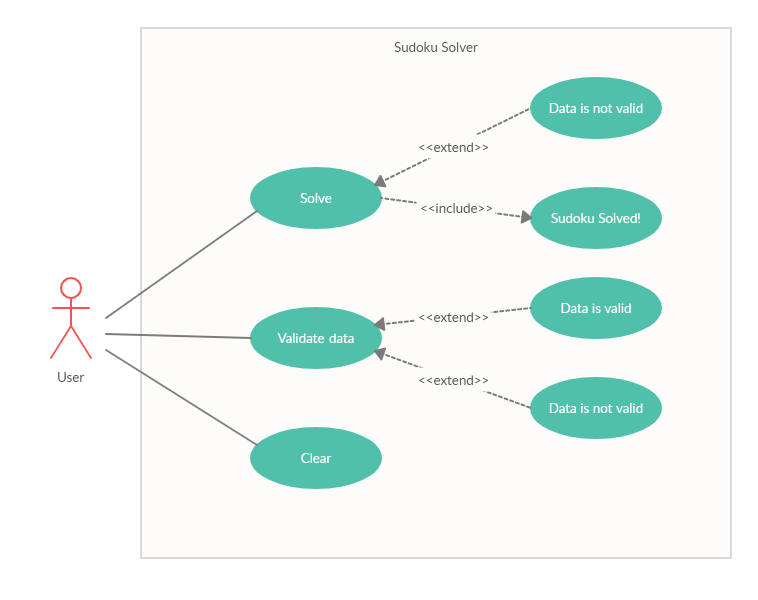
## Wymagania funkcjonalne

* + 1. Aplikacja posiada graficzny interfejs pozwalający zawierający planszę do gry.
    2. Plansza podzielona jest na dziewięć kwadratów, które z kolei zawierają po dziewięć kwadratów w których umieszczane są cyfry.
    3. W prawym górnym rogu znajduje się przycisk „Validate data” służący do sprawdzenia poprawności danych wpisanych przez użytkownika.
    4. Poniżej przycisku „Validate data” znajduje się przycisk „Solve”. Po jego kliknięciu zagadka powinna zostać rozwiązana.
    5. Pod przyciskiem „Solve” znajduje się pole tekstowe służące do wyświetlania komunikatów z aplikacji. Jeżeli klikniemy przycisk „Solve” to wyświetli się zielony napis „Sudoku solved!”, Jeżeli klikniemy przycisk „Validate data” to wyświetli się komunikat „Data is valid” jeśli dane są poprawne lub „Data is not valid” jeśli dane są niepoprawne.
    6. Na dole ekranu po prawej stronie znajduje się przycisk „Clear” służący do czyszczenia planszy.

## Wymagania niefunkcjonalne

* + 1. Każde pole ma ograniczoną możliwość wprowadzania danych do jednego znaku na pole tekstowe,
    2. Każde dane wprowadzane do aplikacji są typu integer
    3. Aplikacja działa bez opóźnień
    4. Wymiary poszczególnych pól danych powinny mieć wymiar 20px na 20px.

# Diagram przypadków użycia

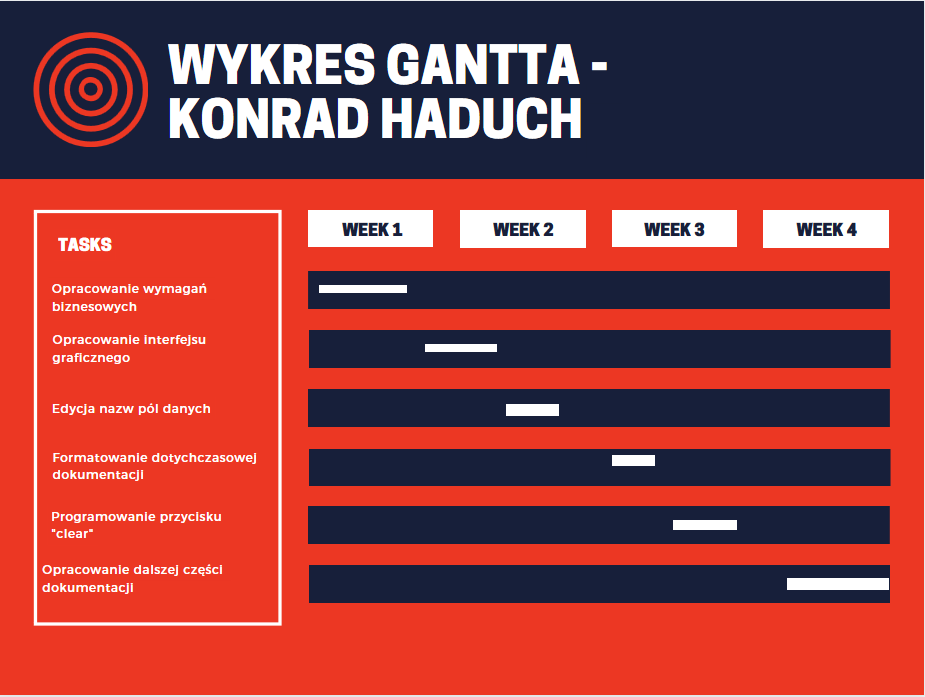


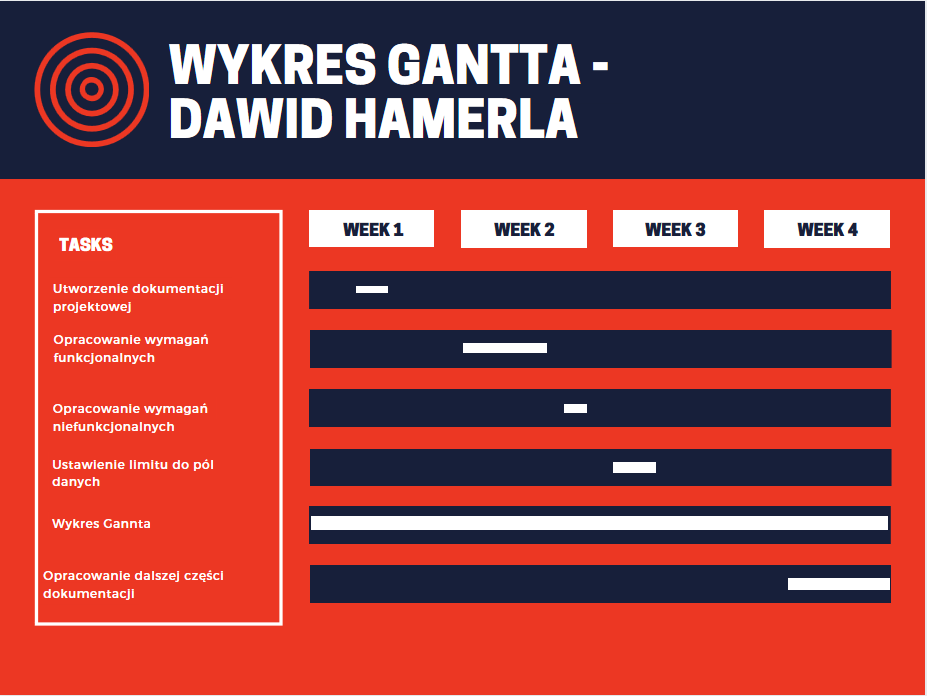
Rys. 1 Diagram przypadków użycia

W powyższym diagramie można zaobserwować przypadki użycia wszystkich założeń funkcjonalnych, których z poziomi aplikacji może dokonać użytkownik. Za pomocą „Solve” rozwiązuje sudoku z wcześniej wprowadzonymi danymi. Przy użyciu „Validate data” sprawdza czy dane przez niego wprowadzone są poprawne. Natomiast za wykorzystaniem „Clear” może wyczyścić wprowadzone własnoręcznie dane lub rozwiązaną przez program łamigłówkę.

# Harmonogram prac nad projektem







# Raport z testów jednostkowych

Testowaniu zostały poddane dwie funkcjonalności. Przetestowano funkcję walidacji danych oraz funkcję rozwiązywania zagadki. Obie funkcje przeszły testy pozytywnie bez poprawiania kodu źródłowego. Funkcję walidacji danych przetestowano na 3 różne sposoby i każdy dał wynik pozytywny. *ValidTest* testuję metodę *IsDataValid* zawartą w klasie Engine, która jest silnikiem aplikacji. *ValidTest* tworzy tablicę poprawnych danych które następnie implementuje do metody *IsDataValid*.. Test zwrócił poprawny wynik zatem test uznano za pozytywny. Następnie ValidTest2 posiada jednowymiarową tablicę z niepoprawnymi danymi. Test miał na celu sprawdzenie czy niepoprawne dane zostaną wykryte przez metodę i czy zostanie zwrócona wartość „*false”*. Po przeprowadzeniu testu wynik spełnił oczekiwania i została zwrócona wartość „*false”* zatem test drugi również zakończono pozytywnie. ValidTest3 również posiada niepoprawne dane. Test pokazał zwrócił poprawny wynik zatem ValidTest3 przeszedł również pozytywnie. Czasy wykonywania się testów przedstawiono na rysunku poniżej. *SolvingTest* zawiera dwa zbiory danych. Pierwszy zawiera dane przygotowane do rozwiązania zagadki. Test miał na celu sprawdzenie czy metoda uzupełni poprawnie brakujące dane. Wynik był zadowalający i uznano test za pozytywny. Drugi zbiór danych to gotowe rozwiązanie zagadki. Test miał za zadanie sprawdzić czy dane oczekiwane będą się pokrywały z danymi wyświetlonymi przez metodę. Test wykonany dał wynik pozytywny.

