**Uniwersytet Warmińsko-Mazurski w Olsztynie**

**Wydział Matematyki i Informatyki**

**Zespół TSR:**

**Damian Tomczak**

**Dominik Słodkowski**

**Bartosz Rybiński**

**Block-based programming language**

Praca zaliczeniowa napisana

napisana pod kierunkiem

mgr inż. Tomasza Żmijewskiego

Olsztyn 2022

1. **Opis firmy:**

TSR School jest franczyzodawcą o zasięgu ogólnokrajowym, który swoje placówki ma zlokalizowane w średnio zaludnionych polskich aglomeracjach powyżej 50 tyś mieszkańców, oraz większych miastach. Poza tym firma dostarcza narzędzia do nauczania zdalnego w języku angielskim oraz kursy online, jak również ‘bootcampy’ programistyczne i szkolenia z zakresu kompetencji cyfrowych.

Przedsiębiorstwo jest akademią IT z prawie 10-letnim doświadczeniem proponującym nieszablonowane rozwiązania w zakładaniu innowacyjnego biznesu w sektorze edukacyjnym. Korzystając z doświadczenia, „know-how”, materiałów edukacyjnych i platform do zarządzania.

Główna siedziba firmy zlokalizowana jest w Olsztynie na ulicy Słonecznej 54, 10-561 i otwarta jest w godzinach 8-16 (pomoc telefoniczna dla franczyzobiorców oraz student jest otwarta w godzinach pracy placówki). Preferowana metoda komunikacji ze szkołą odbywa się za pośrednictwem poczty elektronicznej pod adresem [contact@tsr.edu.pl](mailto:contact@tsr.edu.pl).

Firma zajmuje się nauką programowania i posiada system zarządzający przydzielaniem i oceną zadań. Umożliwia on autoryzację użytkownika oraz komunikację ze studentami i nauczycielami. Zadania są realizowane przez uczniów w postaci napisanego kodu, który jest wykonywany na urządzeniu użytkownika a informacja o wykonaniu zadania zostaje przekazana do systemu zarządzającego gdzie jest to zapisywane po stronie serwera firmy. Efekty działania oceniane są poprzez system rankingowy, posiadający publicznie dostępną tablicę wyników z najlepszymi studentami w danej grupie.

System to cała platforma nauczania, która oprócz przydzielania i oceny zadań dostarcza narzędzia do nauczania dla poszczególnych grup w zależności od zaznajomienia się z informatyką.

Istnieje luka w systemie nauczania którą TSR School chce załatać. Nauka programowania za pomocą kodu jest przystępna dla osób, które miały już styczność z komputerami oraz znają język angielski. Problemem są dzieci i osoby o niskich zdolnościach cyfrowych, w którym wypadku nauczanie przebiegało za pomocą pisania i analizowania skryptów w pythonie. Osoby te, które dopiero uczą się pisać na klawiaturze, nie są w stanie efektywnie pisać kodu.

Z komentarzy osób prowadzących oraz uczniów pilotażowej grupy uczącej się poprzez pisanie skryptów pythona wynikało, że materiał jest zbyt skomplikowany. Pierwsze 3 zajęcia poświęcali na naukę wprowadzania danych do systemu, wizualizacja za pomocą rysunków nie była wystarczająca dla uczniów, a język angielski wprowadzał dodatkowe obciążenie dla studentów.

Firma podejmowała się, również prób nauki poprzez fizyczne obiekty służące za analogi elementów wykorzystywanych do stworzenia programu:

* drewniane bloczki o różnych kształtach – podstawowe operacje warunkowe oraz arytmetyczne, pętle.
* karteczki samoprzylepne – zmienne
* deska do krojenia – podstawa programu, funkcji.

Ta metoda nauki, wymagała dużego zaangażowania ze strony wysoko płatnych prowadzących. Wykonywali oni kompilacje w głowie żmudnie analizując krok po kroku program napisany przez ucznia w trakcie zajęć. Brakowało również ujednoliconej składni, oraz informacji o błędach które normalnie świadczą o złym działaniu programu. To wszystko przy jednoczesnym pilnowaniu uczniów którzy ze względu na młody wiek, zachowywali się często niestosownie.

Forma nauki poprzez drewniane klocki analogiczne do elementów programu była atrakcyjna dla uczniów, rozwiązanie to jednak było nieefektywne. Wymagania wobec prowadzących początkowo skutkowały wymogiem posiadania więcej niż jednego nauczyciela przy realizacji zajęć grup najsłabszych pod względem umiejętności cyfrowych i programistycznych.

Spowodowało to znaczący wzrost kosztów operacyjnych firmy, która w ostatnim kwartale zanotowała niższy niż przewidywany wzrost zysków. Zmusiło to do wycofania się z przystępnej formy nauki na rzecz mniej intuicyjnych klasycznych rozwiązań programistycznych. Firma jest jednak świadoma luki w obszarze nauczania programowania i przewiduje wzrost konkurencji w następujących latach. Sprawia to, że jest zainteresowana rozwojem nowatorskich narzędzi nauczania.

Po konsultacjach z firmą okazało się, że bardzo dobrym rozwiązaniem, który pokryłby lukę w nauczaniu w prosperującej gałęzi nauczania niezaznajomionych z technologią, jest narzędzie, które przeniosłyby poprzednie doświadczenie z grupą testową do systemu informatycznego oraz bazując na badaniach uniknęłoby wcześniej wymienionych wzrostów kosztów operacyjnych lecz nadal przybliżałoby te osoby do obsługi komputera. Okazało się również, że pomimo początkowej łatwości w poruszaniu się podczas zabawy w języku ojczystym dla studentów, podczas dalszych etapów nauki stanowiło to problem, dlatego interfejs użytkownika powinien być w języku angielskim.

**Słownik Pojęć**

Użytkownik = Osoba korzystająca z systemu zapisana w bazie danych, posiada login i hasło.

Student = Użytkownik zapisany w rankingu posiadający możliwość budowania schematów oraz wybierania i rozwiązywania przydzielonych zadań.

Nauczyciel = Użytkownik posiadający możliwość przydzielania zadań dla studentów, mający dostęp do schematów studentów oraz możliwość monitorowania postępu ich nauki.

Blok = Podstawowa część schematu, mogąca realizować tworzenie i zmianę zmiennych, operacje warunkowe i arytmetyczne oraz pętle warunkowe, mogący wchodzić z interakcję z innymi blokami.

Schemat = Algorytm stworzony za pomocą bloków i relacji między nimi, którego wynik wypisywany jest na konsole.

Ranking = Lista prezentująca postęp nauki studenta z porównaniem do innych studentów, na podstawie ukończenia zadań dla nich przydzielonych.

Zadanie = Przydzielony przez prowadzącego zajęcia problem algorytmiczny, który jest rozwiązywany budując schemat.

Konsola = Standardowe wyjście tekstowe w dostępnym systemie, na którym wyświetlane są wyniki działania schematu.

System = Należące do firmy multifunkcjonalne oprogramowanie dostarczające programowi możliwość autoryzacji, informacji o studentach jak przydzielonych do nich zadaniach oraz zestaw narzędzi do nauki programowania. Przechowuje też informacje o postępie nauki studentów.

Program = Jedno z narzędzi do nauki programowania z którego korzysta student. Do jego działania niezbędne są informacje dostarczane przez system.

* Organizacja – TSR/TSR School zwany również franczyzodawca zajmujący się organizacją usług w dziedzinie edukacji w branży IT
* Otoczenie organizacji – firmy rekrutujące specjalistów IT, inne szkoły programowania
* Intersariusze organizacji - dzieci i nastolatkowie, human reasources teams
* Dziedzina problemowa – nowoczesne technologie
* Intersariusze dziedziny problemowej - firmy technologiczne
* Pracownicy biznesowi – główna siedziba firmy, nauczyciele

Zakres organizacji, dziedziny problemowej, aktorzy

Firma TSR School

Organizacja = cała firma

Struktura firmy = nauczyciel, placówka dydaktyczna, główna siedziba firmy

Dziedzina problemowa = nauka programowania i problem solving

Aktorzy biznesowi: student, nowy student

Pracownicy biznesowi: nauczyciele

PU:

- ~~Utwórz studenta~~

- ~~Przydziel zadanie~~ -> Udostępnianie danych

- ~~Autoryzuj użytkownika~~ -> Autoryzowanie

- ~~Wybierz zadanie~~ -> Nauka w programie

- ~~Buduj program~~ -> Nauka w programie

- ~~Kompiluj program~~ -> Nauka w programie

- ~~Oceń zadanie~~ -> Ranking

- ~~Dostosuj program~~ -> Nauka w programie

Aktorzy biznesowi: student, ~~nowy student~~

Pracownicy biznesowi: nauczyciel

PU:

- Administrowanie systemem

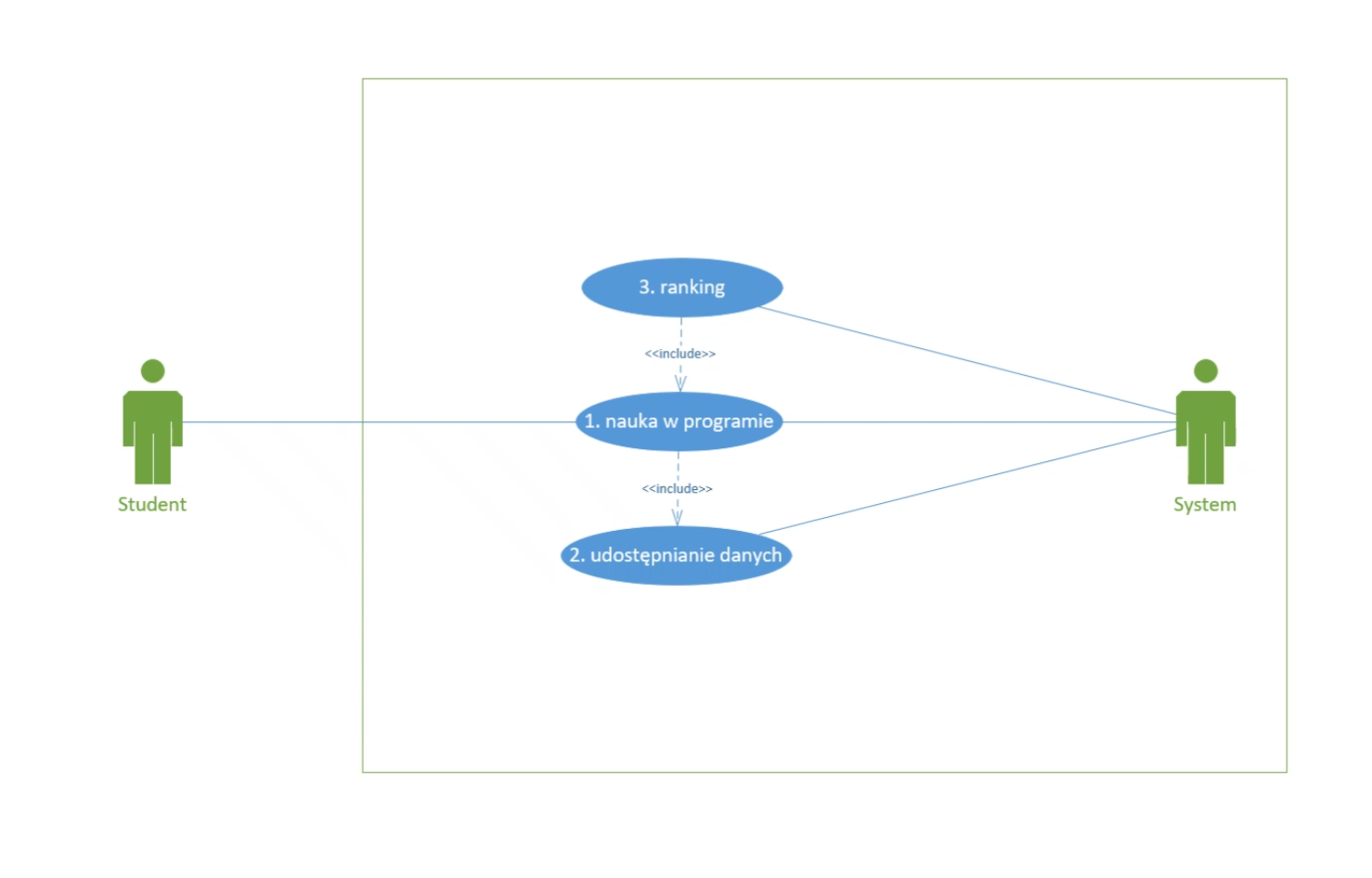
- Ranking

- Nauka w programie

Aktorzy biznesowi: student

Pracownicy biznesowi: nauczyciele

**Biznesowy Diagram Przypadków Użycia**



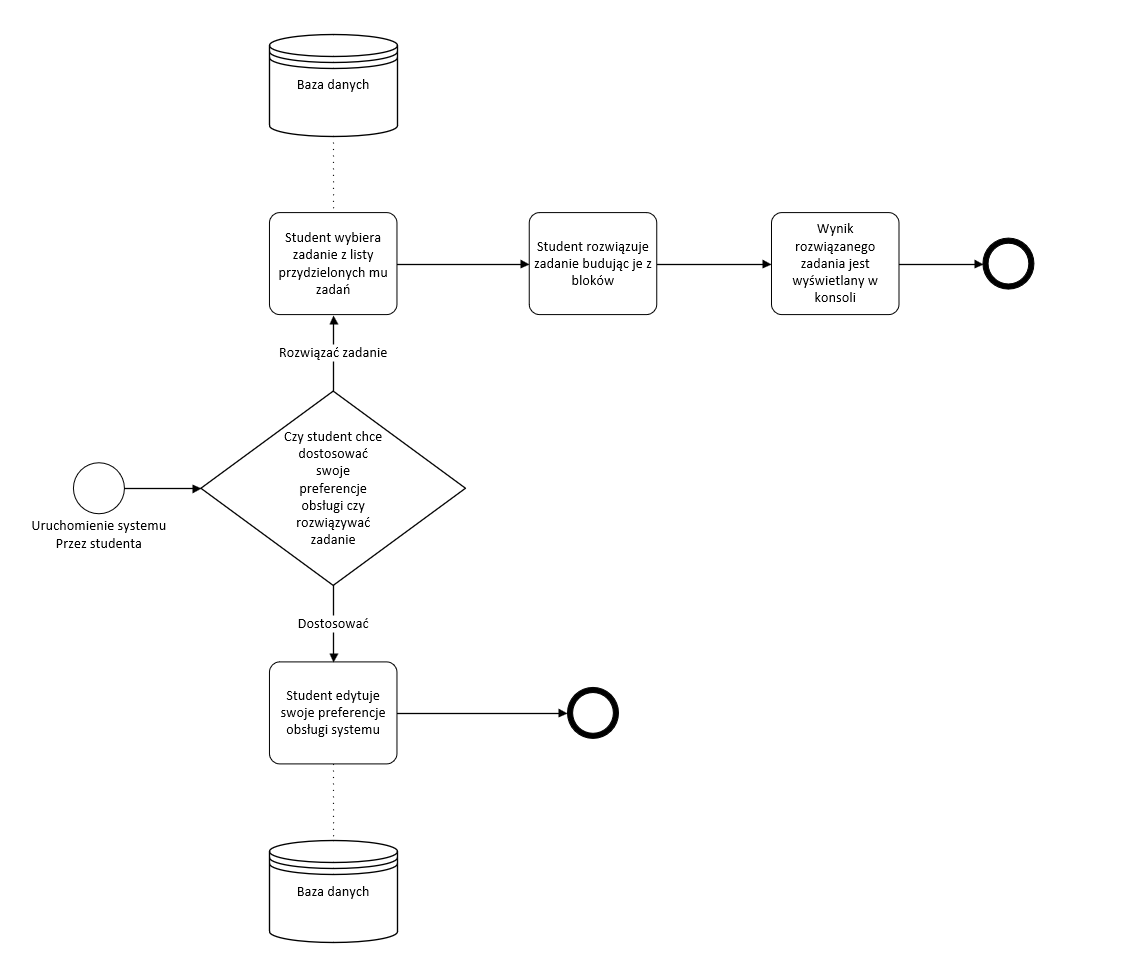
**Aktorzy**

* Student
* System

**Przypadki użycia**

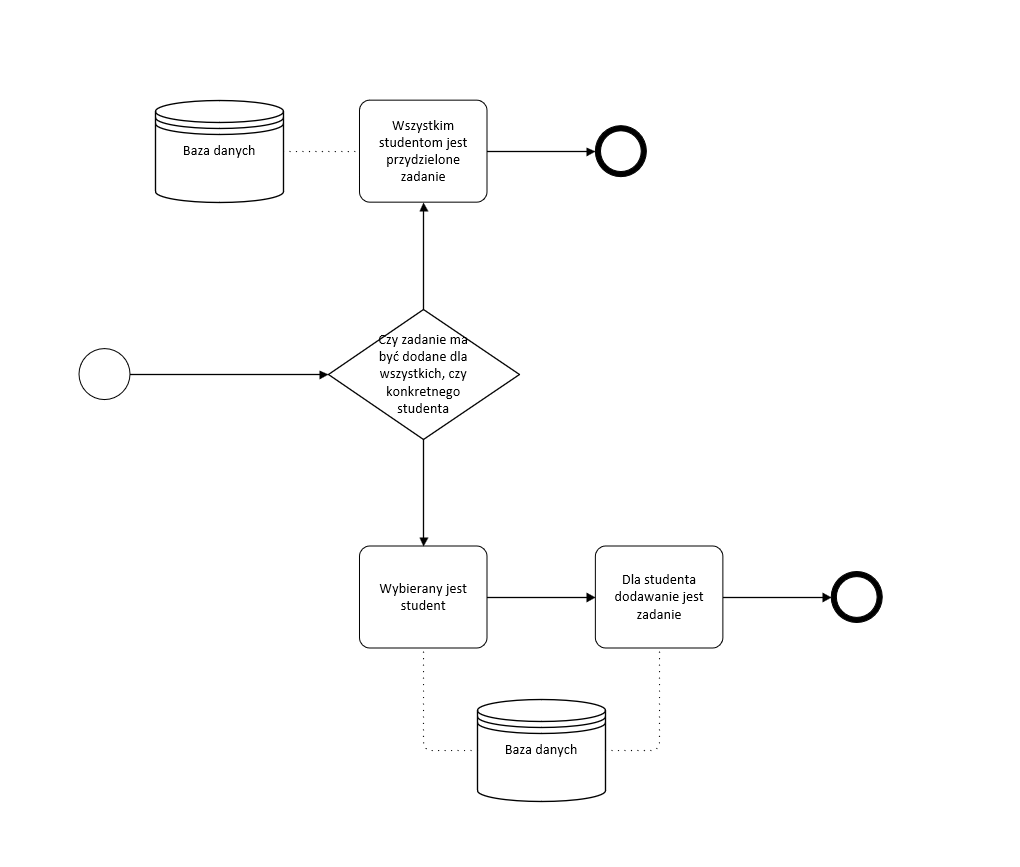
1. Nauka w programie

Student nabywa nowych umiejętności w programie, a jego obsługa sprowadza się do: dostosowaniu preferencji obsługi, wybieraniu przydzielanych zadań oraz tworzeniu skryptu za pośrednictwem bloków oraz sprawdzaniu i analizy wyników.



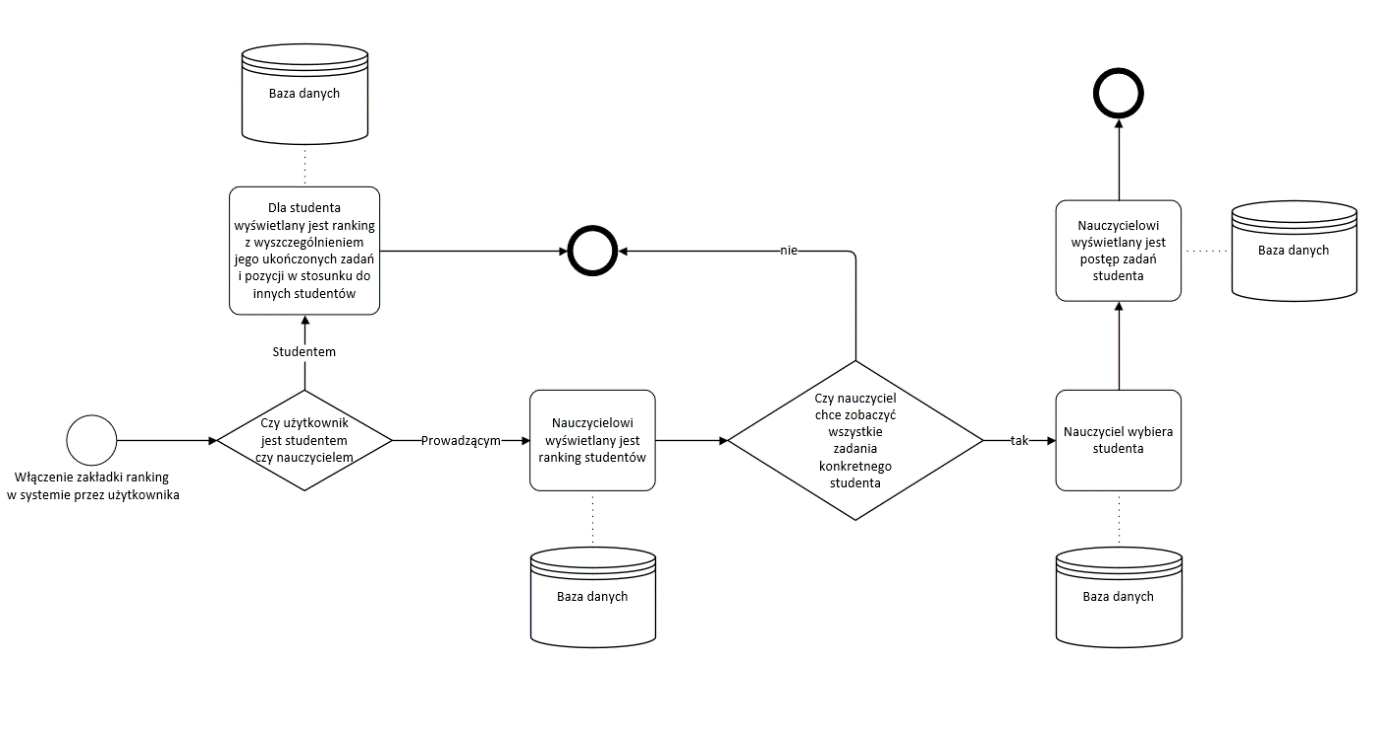
1. Udostępnianie danych

Program do swojego działania potrzebuje możliwości autoryzacji użytkownika, pobierania zadań dla zalogowanego użytkownika.



1. Ranking

Student w systemie ma możliwość monitorowania swoich osiągów oraz motywowania się dysponując możliwością porównaniu swoich wyników z rówieśnikami z grupy. Rola nauczyciel w rankingu sprowadza się do analizy pracy jego studentów. Dlatego program powinien dostarczyć informacji o wyniku zadania, aby system mógł potem sprawdzić te dane i odnotować to studentowi i rankingowi.



1)

Przykład:

Student Milena Samsel wybiera dostępne dla niej zadanie o nazwie „dzielenie z resztą”a następnie rozwija bazowy schemat składających się z dwóch bloków typu zmienna o wartościach 2 i 3 o odpowiednie bloki, aby osiągnąć wymieniony cel – „osiągnij resztę z dzielenia 5”. Przyciskiem Start student prosi system o wykonanie schematu, następnie program prezentuje na konsoli wynik działania schematu.

Specyfikacja:

- student wybiera zadanie

- student rozwija schemat

- program wykonuje schemat

- program prezentuje wynik działania schematu

2)

Przykład:

Student Milena Samsel próbuje się zalogować w programie. Dane uwierzytelniania (milena.samsel1 password123) są przetwarzane przez system. Zalogowanie kończy się prawidłowo, program następnie prosi system o dostarczenie zadań, które zostały przypisanego do wykonania dla studenta: *https://www.gov.pl/tasks/task1.*json (nazwa zadania to *dzielnie i mnożenie*) #id0 *https://www.gov.pl/tasks/task2.json* (nazwa zadania to *dodawanie i odejmowanie*) #id1.

Specyfikacja:

- autoryzacja użytkownika

- przypisanie zadań

3)

Przykład:

Student Milena Samsel uruchamia wykonanie schematu dla zadania *dzielenie i mnożenie.* Program przekazuje wynik działania schematu (5) do Systemu, gdzie jest on porównywany z pożądanym wynikiem 5. Wynik jest prawidłowy, dlatego informacja o prawidłowym wykonaniu zadania wraca do programu. A prawidłowe wykonywanie zadania dla studenta jest zapisywane w rankingu.

Specyfikacja:

- sprawdzenie wyniku

- aktualizacja rankingu

**Systemowy diagram przypadków użycia**



**Aktorzy**

* Student
* System

**Przypadki Użycia**

1. Wybierz Zadanie

Student po zalogowaniu i pobraniu przez program informacji o podzielonych do niego zadań widzi ekran z możliwymi do wyboru zadaniami przypisanymi mu przez nauczyciela. Wybiera konkretne zadanie i przechodzi do budowania programu.

* + - 1. Wykonaj 6. Autoryzuj użytkownika
      2. Wykonaj 7. Przydziel zadania
      3. System wyświetla listę zadań dostępnych dla studenta

2a. System wyświetla komunikat o braku dostępnych zadań

Zakończ

* + - 1. Student wciska przycisk „wybierz”
      2. Wykonaj 2. Buduj Program

Studentowi Karolowi Tuskowi po zalogowaniu system prezentuje wertykalną listę przypisanych do jego zadań w postaci identyfikator oraz nazwa (#1 Dzielenie z resztą, #2 Dodawanie, #3 Równanie kwadratowe). Karol zaznacza zadanie o identyfikatorze #2 i wybiera wciska przycisk „wybierz”.

1. Buduj Program

Student za pomocą bloków(zestaw instrukcji bądź instrukcja do wykonania przez procesor) które są układane przez niego w wybranej kolejności tworzy algorytm po czym może wcisnąć przycisk Start.

* + - 1. System wyświetla planszę budowania
      2. Student wyświetla listę bloków

2a Student postanawia opuścić zadanie

Wykonaj 1. Wybierz zadanie

* + - 1. Student wybiera blok z listy
      2. System umieszcza blok na planszy
      3. Student modyfikuje blok

4a. Student modyfikuje wartość bloku

4b. Student modyfikuje wyjście/a bloku

4c. Student modyfikuje wejście/a bloku

* + - 1. Student podejmuje decyzję

5a. Student dalej buduje program

Wróć do 5.1. System wyświetla planszę budowania

5b. Student wciska przycisk „start”

Wykonaj 3. Kompiluj program

Student Karol Tusk po zalogowaniu i wybraniu zadania „#2 Dodawanie” wyświetlana jest przez system plansza do budowania. Użytkownik zapoznaje się z istniejącymi blokami (blok typu zmienna int z wartością 1, blok typu zmienna int z wartością 2 oraz blok output) oraz celem działania programu „uzyskaj wynik na wyjściu 3”. karol.tusk1 następnie otwiera menu wyboru bloków i z kategorii działania arytmetyczne wybiera addition. Kolejną czynnością studenta jest połączenie wejść bloku addition do zmiennych typu int wartością 2 oraz 1, wyjście bloku addition łączy z blokiem output. Aby sprawdzić rezultat student wciska przycisk „start”.

1. Kompiluj program

Po wciśnięciu przycisku start program blokowy jest parse’owany do pythona i jest interpretowany przez interpreter. Jeżeli program jest napisany poprawnie jego wynik jest wyświetlany studentowi na konsoli, jeżeli nie zwracany jest błąd.

* + - 1. System parsuje schemat blokowy na kod pythona
      2. System wykonuje kod pythona

2a. System zwrócił błąd podczas interpretowania kodu

2a.1. System wyświetla błąd w konsoli

2a.2. Wykonaj 2. Buduj program

* + - 1. System wyświetla wynik działania kodu na konsoli
      2. Wykonaj 4. Oceń zadanie

System otrzymuje listę bloków w aplikacji wraz z ich stanem i relacjami miedzy nimi. Bloki to zmienna typu int z wartością 2, zmienna typu int z wartością 1, operacja arytmetyczna addition, blok output. Wejścia bloku addition zawierają relacje z blokami zmiennych z wartościami 1 oraz 2, natomiast wyjście zawiera relacje z outputem. System parsuje schemat na kod pythona. Wynik jego działania jest prawidłowy i wyświetla komunikat 3. Komunikat z wartością 3 jest prezentowany na konsoli studenta i przekazywany do oceny przez system.

1. Oceń zadanie

System po wykonaniu schematu porównuje rezultat z oczekiwanym rezultatem potem wyświetla stosowny komunikat w zależności od wyniku porównania, a uzyskanie prawidłowego zapisuje do bazy.

* + - 1. System odbiera wynik
      2. System sprawdza wynik

2a. Jeśli wynik prawidłowy

2a.1. System aktualizuje wykonanie zadanie

2a.2 Program wyświetla komunikat o prawidłowym wykonaniu na konsoli

2a.3 Wykonaj 2. Buduj program

2b. Jeśli wynik nieprawidłowy

2b.1 Program wyświetla komunikat o nieprawidłowym wykonaniu na konsoli

2b.2 Wykonaj 2. Buduj program

System odbiera wynik działania schematu studenta Karola Tuska w postaci stringu „3”, jego wartość jest porównywana z oczekiwaną wartością przypisaną do zadania „3”. Wynik to prawda, dlatego na konsoli studenta pojawia się komunikat o prawidłowym wykonaniu zadania, a wykonanie zadania rejestrowane jest w bazie danych.

1. Dostosuj program

Użytkownik i prowadzący ma możliwość podstawowej możliwości dostosowania programu do własnych preferencji za pośrednictwem paska zadań (m.in. rozdzielczość).

* + - 1. System odbiera żądanie personalizacji
      2. System obsługuje żądanie personalizacji

Użytkownik Jarosław Kaczyński naciska przycisk „inne” na pasku zadań, system rozwija opcje ukryte spod tej opcji, które składają się z informacji o programie i pomocy. Następnie użytkownik naciska opcję „pomoc”. System prezentuje mu pomoc w obsłudze programu. Użytkownik zapoznaje się i zamyka okno.

Diagram

Description automatically generated

1. Autoryzuj użytkownika

Student wpisuje login I hasło w aplikacji. Jeżeli wprowadził prawidłowe dane przechodzi do przeglądu zadań. Jeśli wpisał złe informacje proszony jest o ponowne wpisanie hasła i loginu. Wprowadzone dane są później wykorzystywane w celu weryfikacji dostępu do danych.

1. Przydziel Zadania

Nauczyciel tworzy nowe zadanie dla studentów w systemie z przykładowymi niekompletnymi rozwiązaniami, po czym umieszcza je w bazie danych i przypisuje wybrane z bazy danych zadania wybranym studentom. Następnie o te dane prosi program podczas pracy.

* + - 1. Program prosi o dane przypisane dla zalogowanego studenta
      2. System zwraca przypisane zadania dla zalogowanego studenta

Program z zalogowanym studentem Stanisławem Kowalskim o identyfikatorze #666 prosi o przypisane dla niego zadania. System zwraca, że przypisane dla niego zadania znajdują się pod adresami *https://www.gov.pl/tasks/task1.*json (nazwa zadania to *dzielnie i mnożenie*) *https://www.gov.pl/tasks/task2.json* (nazwa zadania to *dodawanie i odejmowanie*).

**Dokumenty występujące w ramach dziedziny problemowej**

**Graphical user interface, text, application

Description automatically generatedformularz\_rejestracji** = rodzaj\_kursu + { dane\_uczestnika } + { dane\_opiekuna } + skad\_wiesz + uwagi

**rodzaj\_kursu** = [on-site | remote]

**dane\_uczestnika** = email + rok\_urodzenia + imie\_i\_nazwisko

**dane\_opiekuna** = email + nr\_telefonu + imie\_nazwisko + kod\_pocztowy

Graphical user interface, text, application

Description automatically generated

**dyplom** = imie\_nazwisko\_uczestnika + prowadzacy\_zajecia + data

Text

Description automatically generated

**zgoda\_przetwarzania\_danych\_osobowych** = data + podpis



**faktura\_za\_serwer** = data\_wystawienia + data\_wystawienia + nr\_faktury + nazwa\_sprzedawcy + adres\_sprzedawcy + nip\_sprzedawcy + vat\_ue + nabywca + adres\_nabywcy + płatnosc + zamowienie + pelna\_cena + uwagi

**platnosc** = [ karta | przelew | przelewy24 ]

**zamówienie** = { usluga }

**usluga** = nazwa\_serwera + ilosc + cena\_netto + vat + warto\_brutto

**Lista klas, atrybutów, związków**

Klasy:

**Użytkownik**

* Metody komunikacji

**Student, dziedziczy po użytkowniku**

* Postęp nauki studenta

**Zadanie**

* Określony cel
* Zadane warunki początkowe
* Nazwa

**Blok**

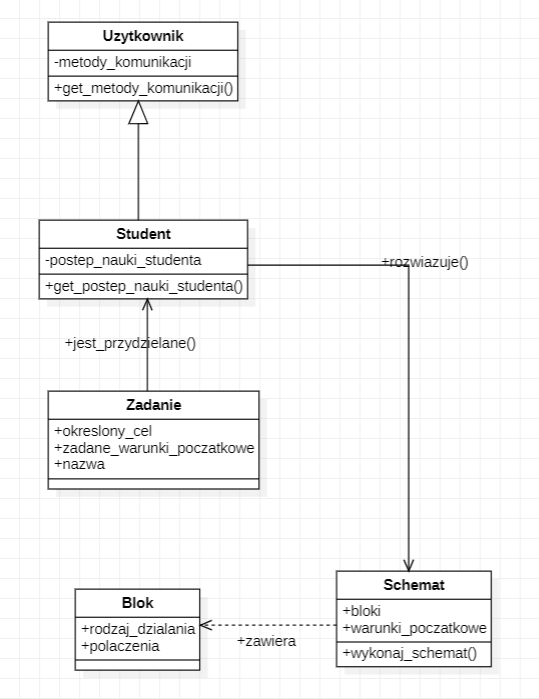
* Rodzaj działania
* Połączenia

**Schemat**

* Bloki
* Warunki początkowe

**Diagram klas - konceptualny**

**Diagram Klas - implementacyjny**



**Diagram obiektów**

