

**WYDZIAŁ INŻYNIERII METALI I INFORMATYKI PRZEMYSŁOWEJ**

**Projekt dyplomowy**

**Opracowanie i Implementacja strony internetowej do obliczania i nauki wybranych metod numerycznych**

***Development and implementation of a website for calculating and learning selected numerical methods***

Autor: Szymon Talar

Kierunek studiów: Inżynieria Obliczeniowa

Opiekun pracy: Dr hab. Danuta Szeliga

Kraków, 2025

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

………………………

**Spis treści**

[1. Wstęp 4](#_Toc189755185)

[2. Omówienie projektu 5](#_Toc189755186)

[2.1. Przegląd dostępnych rozwiązań 5](#_Toc189755187)

[2.2. Cel i założenia projektu 11](#_Toc189755188)

[2.3. Wykorzystane technologie 12](#_Toc189755189)

[3. Implementacja 15](#_Toc189755190)

[3.1. Konstrukcja i implementacja algorytmów 15](#_Toc189755191)

[3.2. Komunikacja front-back 20](#_Toc189755192)

[3.3. Stworzenie GUI 26](#_Toc189755193)

[3.4. Automatyzacja zapytań dla przykładowych obliczeń 31](#_Toc189755194)

[3.5. Zapis i odczyt danych 34](#_Toc189755195)

[3.6. Opis widoków w aplikacji 35](#_Toc189755196)

[4. Testy aplikacji 38](#_Toc189755197)

[4.1. Implementacja testów jednostkowych 38](#_Toc189755198)

[4.2. Samodzielne przejście ścieżek procesów 40](#_Toc189755199)

[5. Podsumowanie 42](#_Toc189755200)

[6. Źródła 43](#_Toc189755201)

# Wstęp

Rozwój technologii informacyjnych i dynamiczny postęp w dziedzinie programowania otwierają nowe możliwości w nauce oraz dydaktyce. Współczesne systemy edukacyjne coraz częściej wykorzystują narzędzia internetowe, które nie tylko ułatwiają dostęp do wiedzy, ale także wspierają aktywny proces nauczania i uczenia się. Jednym z obszarów, który może szczególnie skorzystać z tego rodzaju rozwiązań, są metody numeryczne – dziedzina matematyki stosowanej, znajdująca szerokie zastosowanie w inżynierii, fizyce, ekonomii oraz informatyce. Metody numeryczne pozwalają na rozwiązywanie złożonych problemów matematycznych, których analityczne rozwiązanie jest trudne lub wręcz niemożliwe.

Niniejsza praca jest odpowiedzią na rosnące zapotrzebowanie na narzędzia wspierające edukację matematyczną i inżynierską w sposób nowoczesny i przyjazny dla użytkownika. Pomimo istnienia licznych opracowań i aplikacji w tym zakresie, wiele z nich jest trudno dostępnych, ma ograniczoną funkcjonalność lub nie oferuje wystarczająco wysokiej jakości treści edukacyjnych. Celem pracy jest zatem stworzenie działającego narzędzia w postaci aplikacji webowej, która łączy możliwość wykonywania obliczeń numerycznych z nauką działania metod obliczeniowych. Pozwoli to na wykorzystanie takiego narzędzia zarówno przez osoby rozpoczynające naukę zagadnień znajdujących się w aplikacji, jak i ludzi już zaznajomionych z nimi. Praca ta umożliwi również dodawanie do aplikacji kolejnych metod oraz funkcjonalności które dodatkowo ułatwią użytkownikowi poruszanie się po niej, jak i zwiększą jej walory edukacyjne.

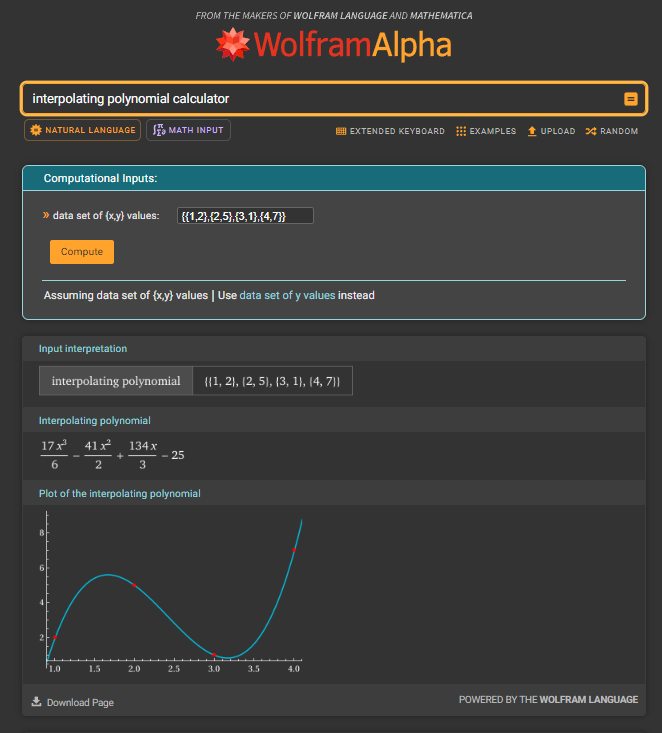
W kolejnych rozdziałach pracy zostaną przedstawione szczegółowe założenia projektu, wybór technologii, proces implementacji oraz analiza wyników. Podjęta zostanie także refleksja nad możliwościami rozwoju stworzonej platformy oraz jej potencjalnym wkładem w edukację matematyczną i naukę metod numerycznych. Tym samym ta praca wpisuje się w nurt wykorzystania nowoczesnych technologii w edukacji oraz promowania nauki przez praktykę i doświadczenie.

# Omówienie projektu

## Przegląd dostępnych rozwiązań

W tym podrozdziale zostaną przeanalizowane obecnie istniejące najpopularniejsze rozwiązania, pozwalające na naukę bądź też obliczenie wybranych metod numerycznych. Uwzględnione zostaną wyłącznie rozwiązania, które są dostępne w wersjach darmowych oraz te, które są szeroko dostępne i nie wymagają żadnych dodatkowych kroków koniecznych do skorzystania z nich jak np. wymagane założenie konta. Pominięte zostaną również aplikacje do pobrania na komputer bądź telefon, ponieważ z racji na swoją specyfikację będą one borykać się innymi problemami nieznanymi stronom internetowym(m.in. utrudniony dostęp z racji na sam fakt wymogu pobrania aplikacji czy też uzależnienie wydajności i dostępności kalkulatora od urządzenia posiadacza oraz rodzaju posiadanego oprogramowania). Wszystkie poniższe przykłady omówiono na podstawie próby wykonania interpolacji, w celu zachowania przejrzystości i spójności przeglądu.

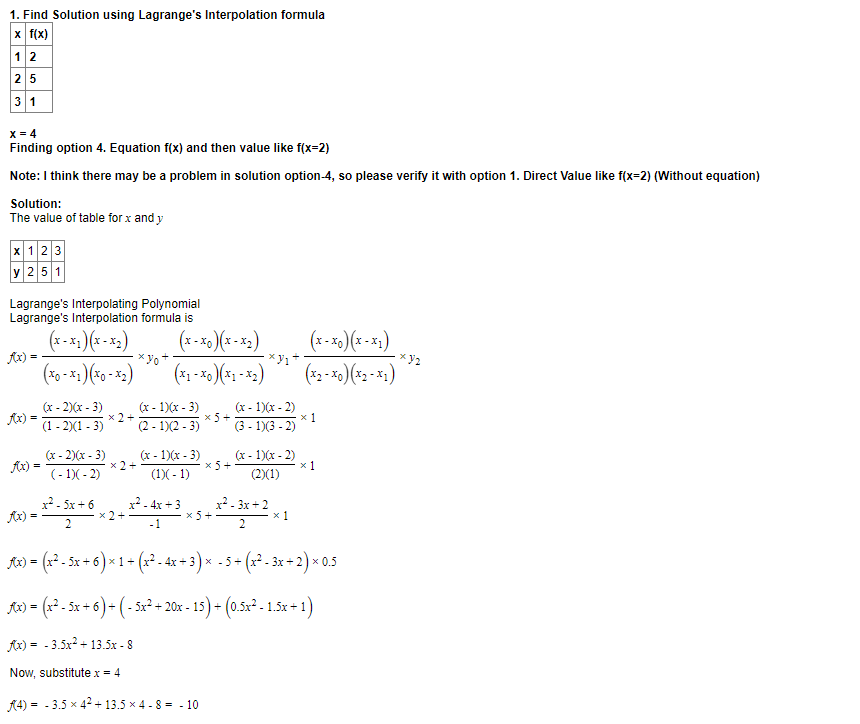
1. **Wolfram Alpha**

Obecnie najpopularniejszy kalkulator naukowy, oferuje szeroki wachlarz możliwości zarówno w kontekście metod numerycznych jak i innych pomocy naukowych. Po wejściu do menu głównego, mamy możliwość wpisania w pasku wyszukiwania interesującego nas zagadnienia.  
  


Rys.1. Zrzut ekranu przedstawiający wynik przykładowej interpolacji wielomianowej wykonanej za pomocą *Wolframu Alpha*[[1]](#footnote-1)

Jak można zauważyć na Rys.1, *Wolfram* umożliwia nam wykonanie interpolacji wielomianowej. Po podaniu danych w postaci listy punktów, użytkownik otrzymuje obliczony wzór funkcji oraz fragment jej wykresu wraz z zaznaczonymi punktami które zostały podane w danych wejściowych. Już na tym etapie można zauważyć jednak kilka wad zaprezentowanego rozwiązania: lista punktów, które są do podania dla interpolacji musi zostać podana w postaci dwóch wartości zamkniętych klamrami, co zważywszy na wstawienie małego okienka bez możliwości jego powiększenia, może powodować problemy z czytelnością danych wejściowych, zwłaszcza przy ich większej ilości. Nie ma również możliwości obliczenia wartości dla konkretnego punktu, a jedynie wzór pozwalający nam na zrobienie tego. Takie rozwiązanie wciąż pozostawia użytkownikowi do wykonania część obliczeń, w postaci podstawienia pożądanej wartości do pokazanego wzoru. Ponadto, brakuje tu również opcji pozwalającej na inne formaty wprowadzania danych. Nie ma również opisu możliwości narzędzia oraz żadnego wytłumaczenia uzyskanego wyniku.

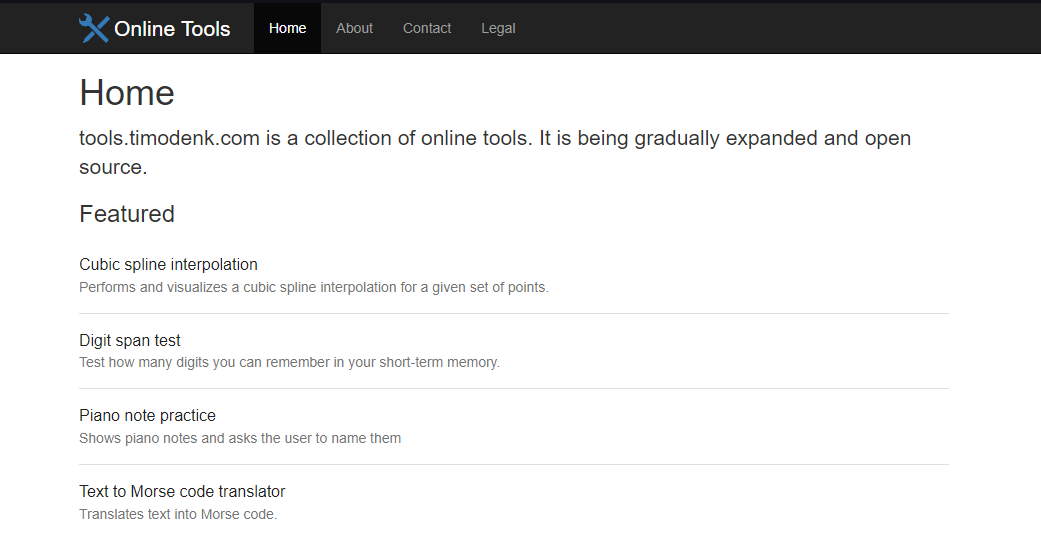
1. **AtoZmath**

Strona specjalizująca się rozwiązywaniu wszelakich problemów obliczeniowych. Tak jak w przypadku *Wolframu*, jest tutaj również możliwość wyszukania interesującej nas metody w celu wykonania obliczeń z jej pomocą. Po wejściu na podstronę przeznaczoną obliczaniu interpolacji *Lagrange’a* jest możliwość podania ilości punktów, ich umiejscowienia na osi oraz typ rozwiązania.  
  


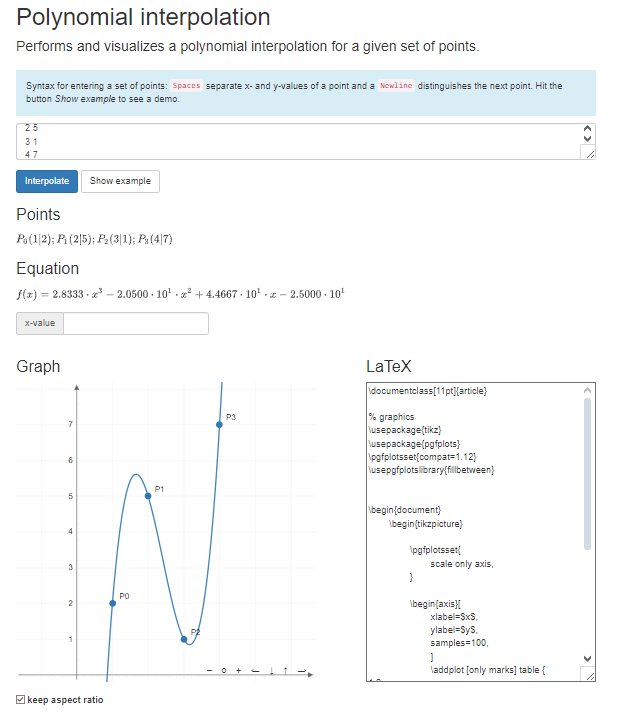
Rys.2. Zrzut ekranu przedstawiający wynik przykładowej interpolacji wielomianowej wykonanej za pomocą AtoZmath[[2]](#footnote-2)

Na Rys.2 jest widoczny rezultat takiego przykładowego zapytania o rozwiązanie dla trzech punktów. Pierwsze co przykuwa uwagę to skrupulatne rozpisanie algorytmu wykonującego obliczenia, które to stanowi podstawę do uzyskania zaprezentowanego rezultatu. Ponadto, pochwalić można również czytelność strony: w przeciwieństwie do rozwiązania dostępnego w *Wolframie*, tutaj zarówno wpisywanie danych wejściowych jak i odczytywanie wyniku jest znacznie łatwiejsze i przystępniejsze, zwłaszcza dla mniej doświadczonych użytkowników. Jednakże strona ta również ma pewne wady: Pojawiające się regularnie reklamy utrudniają poruszanie się w jej obrębie, a do uzyskania pełnego rozwiązania należy odczekać kilkanaście sekund dla pojedynczego przypadku, a okres ten dodatkowo zwiększa się z każdym kolejnym obliczeniem w krótkim odstępie czasu, przez co wykonanie większej serii operacji trwa o wiele dłużej w porównaniu do innych rozwiązań. Dodatkowo, nawet pełne rozwiązanie pozbawione jest wizualnego zobrazowania wyniku oraz opcji porównania z alternatywną metodą obliczeniową. Brakuje również możliwości zapisania wyniku obliczeń do dalszej analizy uzyskanego wyniku i działania metody.

1. **Online Tools**

Jak sama nazwa wskazuje, jest to witryna obejmujące wiele różnorakich narzędzi online, zarówno z zakresu metod obliczeniowych, jak i tych w żaden sposób nie związanych z matematyką jak np. ćwiczenia z zapamiętywania notacji muzycznej bądź też konwersje dowolnego znaku na jej kod *UTF-16*. Pomimo iż nie jest to strona oferująca tak szeroki wachlarz możliwości, umieszczono ją tutaj jako prezentację możliwości trochę mniej znanych i rozbudowywanych stron.   


Rys.3. Zrzut ekranu przedstawiający stronę główną witryny Online Tools[[3]](#footnote-3)

Pierwszy problem jak napotykamy po wejściu na samą stronę zaczyna się jeszcze przed próbą rozwiązania jakiegokolwiek zadania, ponieważ strona główna widoczna na Rys.3 nie pozwala na wyszukanie w żaden sposób funkcjonalności, z której użytkownik mógłby chcieć skorzystać. W przypadku liczby funkcjonalności oferowanej przez tę stronę, brak takiej możliwości już na początku utrudnia użytkownikowi nawigację po niej. Gdy już uda się wyszukać stronę odpowiadającą za interpolację wielomianową, pojawia się możliwość wpisania punktów w polu tekstowym, przy czym każdy punkt musi być podany jako dwie wartości oddzielone spacją, a każdy punkt zaczynać się od nowej linii.  
  
  


Rys.4. Zrzut ekranu przedstawiający stronę wynikową Online Tools po rozwiązaniu interpolacji wielomianowej

Ekran wynikowy zaprezentowany na Rys.4 składa się ze wzoru funkcji, obliczonej na podstawie podanych punktów, oraz wykresu tej funkcji. Ponadto jest możliwość swobodnego poruszania się po wygenerowanym wykresie, a także wyświetlany zostaje kod w *LaTeXie* który da się następnie zapisać w celu ponownego odtworzenia tej funkcji. Niestety, analogicznie do poprzednich rozwiązań, brakuje tutaj opcji porównującej wyniki z innymi metodami, jak i brakuje tutaj możliwości obliczenia wartości funkcji w podanym przez użytkownika miejscu.

Powyższe przykłady rozwiązań dostępnych w internecie oddają w znacznym stopniu możliwości oferowane przez powszechnie dostępne strony. Wszelkie konkurencyjne domeny dostarczają możliwości zbliżone do tych już omówionych na poszczególnych przykładach.

Podsumowując, wśród najczęściej występujących wad stron służących do obliczania

metod numerycznych i ich nauki znajdują się:

**a)** ubogi opis możliwości oraz metod oferowanych na stronach, brak możliwości samodzielnego zrozumienia występujących funkcji.

**b)** wąski zakres możliwych do wprowadzenia danych obliczeniowych, jak i samych wykonywanych obliczeń

**c)** brak opcji zapisu wyników w celu późniejszego odtworzenia lub analizy

**d)** wysoka czasochłonność korzystania ze stron wynikająca z blokad twórców, bądź też niskiej czytelności interfejsu.

Ten projekt będzie skupiał się na rozwiązaniu wyżej wymienionych problemów.

## Cel i założenia projektu

Celem pracy jest stworzenie narzędzia, pozwalającego na swodobny i darmowy dostęp do sekcji edukacyjnej oraz kalkulatora obliczeniowego. Zakres metod wchodzących w skład obu tych elementów zostanie ograniczony do kilku wybranych problemów matematycznych, dla których zaimplementowane algorytmy będą wyłącznie przykładem możliwości oferowanych przez to narzędzie. Oprócz implementacji algorytmów, dla każdego z nich dołączone zostaną dodatkowe funkcjonalności pomagające użytkownikowi na pełne możliwości wykorzystanie narzędzia oraz pomagające w zrozumieniu działania wybranego problemu. Architektura projektu pozwoli na dalsze rozbudowywanie aplikacji poprzez dodawanie kolejnych metod numerycznych oraz funkcjonalności ułatwiających użytkownikowi pozyskiwanie wiedzy. Wśród najistotniejszych założeń można wymienić m.in. :

1. Ułatwienie użytkownikowi samodzielnej edukacji pozostając w aplikacji, poprzez wykonanie widoku teoretycznego, prezentującego dogłębne wyjaśnienie każdej dostępnej metody wraz z prezentacją przykładu.
2. Dodanie opisu zakresu użycia każdej metody i danych potrzebnych do wykonania obliczeń.
3. Poszerzenie zakresu dostępnych problemów do rozwiązania w ramach wybranej metody względem już istniejących rozwiązań.
4. Wprowadzenie alternatywnych możliwości wprowadzania danych dla użytkownika, pomagających w sprawnym poruszaniu się po aplikacji i przyspieszających korzystanie z niej.
5. Stworzenie rozbudowanego ekranu wyświetlającego rezultat obliczeń, wraz z dodaniem elementów pomagających w zrozumieniu istoty problemu i przeprowadzających użytkownika przez kolejne etapy otrzymywania rozwiązania.
6. Udostępnienie użytkownikowi historii wyników pozwalającą na ponowne wykonanie obliczeń, a także eksport danych wejściowych i wyjściowych w celu dalszej analizy zagadnienia.

## Wykorzystane technologie

Technologie wykorzystywane do wykonania tego projektu można podzielić na trzy kategorie:

1. **Front-end**

W tej grupie technologii zawierają się te podstawowe, niemalże niezbędne do utworzenia jakiejkolwiek strony internetowej, tj. *HTML* oraz *CSS*, natomiast całość programistyczna została stworzona za pomocą jednego z najpopularniejszych frameworków dla *JavaScripta*, czyli *Angularze*. Wybór takich narzędzi do pracy został podyktowany szerokim wachlarzem możliwości oferowanym przez ten właśnie framework: pozwala on zarówno na stworzenie pełnych stron od podstaw, co daje twórcy bardzo duży wpływ na jej finalny wygląd oraz komunikację pomiędzy poszczególnymi segmentami, natomiast z drugiej strony pozwala zachować przy tym przejrzystość w kreowanym projekcie i zapewnia wsparcie dla nowoczesnych rozwiązań, pozwalających w o wiele prostszy sposób wykonać pewne zabiegi na stronie, które to byłyby znacznie trudniejsze do wykonania posługując się zwykłym *JavaScriptem*.

1. **Back-end**

Językiem, który został wykorzystany w ramach pracy jest *Java*. W porównaniu z jej głównym konkurentem w tej kategorii, tzn. *C#,* oferuje ona o wiele większą stabilność w wydajności dla różnych systemów operacyjnych na których będzie uruchamiana, co z kolei powinno wpłynąć na zwiększoną liczbę potencjalnych użytkowników takiej strony, a także zapewnia łatwiejsze zarządzanie projektami tworzonymi na szeroką skalę, zawierających w sobie wiele współpracujących ze sobą podzespołów. Ewentualne wady takiego rozwiązania, wśród których głównym z nich z pewnością może być mniejsza wydajność *Javy* od *C#,* mogą zostać zniwelowane poprzez zarówno optymalne tworzenie algorytmów obliczeniowych, jak i w przypadku nadal występujących problemów z wydajnością, zastosowaniem wielowątkowości.[[4]](#footnote-4) [[5]](#footnote-5)

1. **Bazy danych**

Dla zapisania danych rejestrujących się użytkowników oraz danych wejściowych obliczeń przez nich wykonywanych, wykorzystano w projekcie dwie osobne bazy danych, kolejno: *H2* oraz *MongoDB*. Pierwsza z nich, choć nie jest obecnie powszechnie wykorzystywana jako pełnoprawna baza danych i pełni głównie funkcje testowe, została uznana za odpowiednią dla tego zastosowania ze względu na:

- Niski wkład pracy potrzebny w utworzenie oraz zarządzanie taką bazą danych

- Niewielki rozmiar tabeli oraz przewidywana liczba rekordów, dzięki czemu kwestie wydajności nie są dla tego zastosowania zbyt istotne

- Projekt nie zostaje na chwilę obecną wdrażany do powszechnego użytku, dzięki czemu nie ma konieczności przykładania tak dużej troski o względy bezpieczeństwa, jak dla przypadku w którym baza ta miałaby funkcjonować na środowisku produkcyjnym.

W przypadku gdyby któraś z tych zależności uległa zmianie w trakcie rozbudowywania projektu, da się również w dość prosty sposób zamienić tę bazę na inną, stworzoną w bardziej sprawdzonym środowisku jak np. *MySQL*.

Drugie zastosowanie, czyli zapis danych wejściowych niestety już nie jest możliwy do obsługi tą bazą danych w tak prosty sposób, przez co potrzebne było dołączenie osobnej bazy danych w postaci bazy *MongoDB*. Jej wybór z kolei został podyktowany długą historią powszechnego użytku, co zapewnia wsparcie dla wielkoskalowych projektów, oraz ze względu na swoją specyfikację pozwala na łatwe zapisywanie znacznie różniących się od siebie rekordów w jednej tabeli, co jest dla tego projektu kluczowe ze względu na mnogość potencjalnych metod wybranych przez użytkowników. W tym względzie jest to wybór nieporównywalnie lepszy od relacyjnych baz danych takich jak *MySQL*. Jak możemy przeczytać na stronie samych twórców: *„The schemaless design of MongoDB documents makes it extremely easy to build and enhance applications over time, without needing to run complex and expensive schema migration processes as you would with a relational database.”[[6]](#footnote-6)* [pl. Bezschematowa konstrukcja dokumentów w *MongoDB* sprawia, że niezwykle łatwo jest budować i rozwijać aplikacje w miarę upływu czasu, bez konieczności przeprowadzania skomplikowanych i kosztownych procesów migracji schematów, jak miałoby to miejsce w przypadku relacyjnej bazy danych.], co idealnie odzwierciedla istniejący w tym projekcie przypadek zastosowania, w którym do bazy danych z danymi obliczeń w miarę upływu czasu mogłyby dochodzić kolejne metody obliczeniowe, co oznaczałoby jednocześnie kolejne, różne od wszystkich poprzednich, struktury rekordów do zapisania w tabeli.

# Implementacja

## Konstrukcja i implementacja algorytmów

Na potrzebny projektu zaimplementowano algorytmy pozwalające obliczyć trzy różne zagadnienia: interpolację, całkowanie oraz układ równań. Dla każdego z nich dodano co najmniej dwa różne sposoby na ich obliczenie, aby dać użytkownikom możliwość porównania skuteczności każdej z metod. Dla interpolacji były to interpolacje: wielomianowa i trygonometryczna. Dla całkowania były to metody: punktu środkowego, trapezów, Simpsona oraz Gaussa-Kronroda. Dla układów równań zaimplementowałem metodę wykorzystującą wzory Kramera i metodą wielosiatkową. Łącznie pozwala to na wykorzystanie ośmiu różnych algorytmów, zaimplementowanych jako funkcje testowe w celu demonstracji możliwości, jakie oferuje strona.



Fragment kodu 1. Implementacja modelu punktu. Źródło: opracowanie własne



Fragment Kodu 2. Implementacja modelu odpowiadającego danym wejściowym dla interpolacji.

Źródło: opracowanie własne

****Fragment Kodu 3. Implementacja funkcji służącej do obliczania wartości interpolacji trygonometrycznej.

Źródło: opracowanie własne

Fragmenty Kodu 1 i 2 przedstawiają klasy odpowiadające modelom wejściowym dla danego problemu, natomiast w załączonym Fragmencie Kodu 3 zamieszczono przykładowy algorytm zastosowany w projekcie. W celu przyszłego rozwinięcia aplikacji o kolejne metody obliczeniowe, należałoby na pierwszym miejscu napisać klasę analogiczną do niej, implementującą inną wybraną metodę numeryczną wraz z modelami dla danych wejściowych. W tym przypadku, jako że celem jest obliczenie wyniku interpolacji trygonometrycznej, oprócz zmiennej *searchedX* odpowiadającej za wartość *x* dla której potrzeba jest obliczyć wartość, konieczna jest także zmiana skali w której znajdują się nasze punkty w taki sposób, aby wszystkie podane przez użytkownika wartości zawierały się w przedziale [0, π)[[7]](#footnote-7). Aby tego dokonać, dla każdego podanego punktu niezbędne jest rozwiązanie równania podanego w Równaniu (1)

|  |  |  |
| --- | --- | --- |
|  |  |  |

Równanie 1. Równanie przekształcające dowolną podaną wartość *x* na jej odpowiednik w zbiorze [0;2π)

Gdzie:

*normalizedX* – wartość po dokonaniu przekształcenia

*minX* – dolna granica zbioru określonego na podstawie podanych w danych wejściowych punków

*rangeX* – różnica pomiędzy górną i dolną granicą zbioru wejściowego

*x* – wartość wejściowa, podana przez użytkownika

Pozostała część kodu, skupia się na obliczeniu *szeregu Fouriera*, dzięki której mamy możliwość przedstawić funkcję okresową w postaci sumy funkcji trygonometrycznych. W tym przypadku, funkcja ta będzie przyjmowała postać zawartą we Wzorze (2)

|  |  |  |
| --- | --- | --- |
|  |  | **(2)** |

Równanie 2. *Szereg Fouriera*, pozwalający na obliczenie wartości interpolacji dla dowolnego podanego *x*

Gdzie:

*F(x)*- wartość funkcji w punkcie *x*, gdzie w tym przypadku będzie on przyjmował wartość podaną przez użytkownika jako wartość do obliczenia  
, , – współczynniki wyliczane na podstawie znanych dotychczas punktów

*N* – liczba podanych punktów

Wartość zwracana przez tę funkcję, tj. pojedyńcza zmienna typu *double* przedstawiająca wartość funkcji w konkretnym punkcie, jest następnie przekazywana do front-endu który wyświetla ją na ekranie z pozostałymi danymi. W analogiczny sposób zostały napisane pozostałe algorytmy obliczające inne metody bądź wykorzystujące inne sposoby realizacji obliczeń, tak samo jak ich obsługa w pozostałych częściach kodu.

Niektóre kalkulatory wymagały również możliwości wprowadzenia własnoręcznie napisanej funkcji wykraczającej poza wielomiany, a dokładniej kalkulatory całkowania. Dla nich, została dodana dodatkowa biblioteka importowana w *pom.xml*, dająca możliwość parsowania surowego tekstu na funkcję, a następnie obliczenia jej wartości w wybranym punkcie. Fragmentu Kodu 4 i 5 prezentują dodanie biblioteki do kodu i jej przykładowe wykorzystanie w praktyce.



Fragment kodu 4. Zależności dla zewnętrznej biblioteki pozwalającej na wydobycie funkcji zawartej w zmiennej *String*. Źródło: opracowanie własne



Fragment Kodu 5. Przykładowy fragment funkcji obliczającej całkę metodą trapezów, wykorzystujący zewnętrzną bibliotekę do obliczania dowolnych funkcji, zawartych w zmiennej *function*.

Źródło: opracowanie własne

Istotnym problemem, który został rozwiązany dzięki zastosowaniu takiego podejścia jest spora elastyczność w możliwościach wprowadzania danych – wszystkie modele zostały utworzone w sposób pozwalający im na przechowywanie dowolnie dużej liczby danych, a także podanie ich w kilku różnych formach, co daje użytkownikowi szerszy zakres możliwych zastosowań takiej aplikacji.

## Komunikacja front-back

Ten segment zawiera opis procesu, w którym to dane otrzymane w wyniku wywołania kalkulatora są przekazywane na ekran widokowy.



Fragment Kodu 6. Część kodu serwisu odpowiadającego za komunikację pomiędzy kontrolerem a kalkulatorem. Źródło: opracowanie własne



Fragment Kodu 7. Część kodu kontrolera a Javie odpowiedzialna za obsługę danych dot. Interpolacji trygonometrycznej, i przekazanie rezultatu do widoku Żródło: opracowanie własne



Fragment Kodu 8. Obiekt zawierający informacji o odpowiedzi zwróconej przez kalkulator.

Żródło: opracowanie własne

We Fragmentach Kodu 6, 7 oraz 8 został zaprezentowany przykład, jak dane są odbierane i przekazywane z powrotem do kontrolera obługującego dany typ obliczeń. W tym przypadku, otrzymywane są dane w postaci z Fragmentu Kodu 2 na adres kończący się ciągiem znaków *„/trigonometric\_interpolation”*, a następnie utrworzony zostaje obiekt *InterpolationResult* przechowujący dane które będą później potrzebne. Wśród nich znajdują się:

*result* – zmienna przechowująca wynik interpolacji

*coefficients* – obliczone współczynniki funkcji interpolującej

*prompt, explanation* – dwie zmienne do przechowywania opisu wykonanych kroków. To, która z nich zostanie wykorzystana zależne jest od tego skąd będzie pochodziło źródło informacji: *prompt* przechowuje informacje wygenerowane przez sztuczną inteligencję i zostaje utworzony poprzez osobne zapytanie do zewnętrznego *API*(ang. Application Programming Interface), natomiast *explanation* zawiera opis pochodzący z kalkulatora który docelowo znajduje się wewnątrz Fragmentu Kodu 3, natomiast dla prezentacji kodu został tymczasowo usunięty. Istotne w pokazanych częściach kodu jest wdrożenie *Service-Repository Pattern*.

Następnie, wszystkie niezbędne dane zostają uzupełnione podczas wywołania kalkulatora i ewentualnie osobnego serwisu dla prompta, i w takiej formie zostają zwrócone do kontrolera. Ten z kolei, zapisuje wszystkie dane do rekordu w bazie danych, a samą odpowiedź zwraca przy pomocy *ResponseEntity*.



Fragment Kodu 9. Klasa odpowiadająca za zapewnienie możliwości komunikacji do części napisanej w Angularze. Źródło: opracowanie własne

Kluczowe dla komunikacji jest również zapewnienie łączności pomiędzy programami napisanymi w *Javie* oraz *Angularze*. Sama budowa projektu wymusza na nas zastosowanie różnych portów dla obu tych części, tak więc w przypadku gdy jest potrzeba przekazania danych z *Angulara* do *Javy* bądź też odwrotnie, potrzebna jest odpowiednia konfiguracja projektu po obu stronach w sposób umożliwiający pożądaną komunikację. Dla *Javy*, taką konfigurację zapewnia klasa *CorsConfig* zamieszczona we Fragmencie Kodu 9., w której to nadpisana jest metoda *addCorsMappings*. Zmiany zawarte w niej krok po kroku dotyczą:  
- Kluczową rzeczą potrzebną do tego celu jest uzyskanie dostępu do obiektu *CorsRegistry*, w którym przechowywane są m.in. ustawienia dotyczące zasad jakimi kieruje się program komunikując się ze wszystkimi innymi punktami końcowymi (ang. *endpoints*).

- Kolejna linia (*addMapping(„/\*\*”)*, dodaje wymienione ustawienia na cały projekt, tj. dla wszystkich występujących w nim punktów końcowych  
- *.allowedOrigins(…)* pozwala na zdefiniowanie zewnętrznego adresu, z którym będziemy mogli się komunikować. W tym przypadku, adres pozostaje taki sam, gdyż cały projekt jest umieszczony na serwerze lokalnym(ang. *localhost)*, natomiast wymagany jest dostęp konkretnie do portu 4200, na którym uruchamia się część napisana w *Angularze*

- *.allowedMethods(…)* pozwala na zdefiniowanie metod, którymi będziemy mogli się komunikować z podanymi punktami końcowymi. Wymienione zostały tutaj cztery najczęściej wykorzystywane, choć w praktyce zdecydowana większość wykorzystywana w projekcie jest metodą *POST*, i to ona odgrywa dla nas najważniejszą rolę. Jej wykorzystanie umożliwia przesyłanie danych pomiędzy punktami, takie jak np. dane dla algorytmów, a także tworzyć i modyfikować istniejące już zasoby np. modyfikować dane w bazach danych.

- *.allowedHeaders* pozwala na dołączanie dowolnych nagłówków wraz z resztą informacji

- *.allowedCredentials(true)* pozwala na stosowanie autentykacji w kodzie, co może się okazać przydatne w przypadku potrzeby sprawdzenia w wybranym miejscu w kodzie czy użytkownik jest zalogowany na swoim koncie, bądź też korzysta ze strony jako gość.



Fragment Kodu 10. Zawartość pliku *proxy.conf.json* zawierający konfigurację łączności dla Angulara

Źródło: opracowanie własne

We Fragmencie Kodu 10 została pokazana konfiguracja ze strony front-endu. Podobnie jak w *Javie*, tutaj również potrzebne jest zdefiniowanie ip oraz portu punktów docelowych, z którymi będziemy chcieli w przyszłości się łączyć w celu wymiany danych. W tym przypadku taką funkcję pełni plik *proxy.conf.json*, który zawiera podstawowe informacje na temat przyszłych połączeń. Dane możliwe do odczytania zawierają:  
- obszar działania wybranych ustawień, przy pomocy obiektu *„/api/\*”*, co oznacza wybranie ustawień dla całego projektu

- ip i docelowy port z którym umożliwiana jest komunikacja, czyli *localhost:8081*

- flagę odpowiadającę za włączenie zabezpieczeń występujących w kodzie

- poziom na którym będą wyświetlane wszystkie logi programu.

Zastosowanie właśnie takiego podejścia we Fragmentach Kodu 9 i 10, rozwiązuje wszystkie problemy dotyczące komunikacji pomiędzy komponentami *front-end* i kontrolerami *back-end*.  


Fragment Kodu 11. Wysyłanie danych we front-endzie na opisany w Fragmencie Kodu 5. Kontroler

Źródło: opracowanie własne



Fragment Kodu 12. Własnoręcznie napisany parser danych.Źródło: opracowanie własne

Zastosowanie tych ustawie, umożliwia zastosowanie funkcji *this.http.post<>(…)* w dowolnym miejscu w kodzie, która wykonuje połączenie tak jak zostało to zademonstrowane we Fragmencie Kodu 11. W celu wysłania zapytania, najpierw zmieniane są ewentualne wartości tekstowe dla punktów (takie jak. *ln(7)*) na wartości liczbowe przy pomocy funkcji *parseInput()* oraz inicjowany zostaje model dla danych wejściowych. Funkcja *parseInput()* została zaprezentowana we Fragmencie Kodu 12, i jej celem jest zamiana tekstu napisanego przez użytkownika w danych wejściowych na wartości liczbowe. W obecnej postaci, pozwala na konwersję stałych *π*, *e* oraz obliczenie wartości z wyrażeń *sqrt(x)*  oraz *ln(x)*, co daje większą możliwość dla użytkownika w kontekście wprowadzania wartości do kalkulatora. Jej konstrukcja pozwala na proste dodanie kolejnych wyrażeń do parsowania, co w dalszym stopniu rozszerzyłoby jej zakres.

Po wykonaniu parsowania, w nawiasach klamrowych podawany jest typ zmiennej oczekiwane w rezultacie wykonania połączenia, natomiast nawiasy okrągłe zawierają w sobie zmienne dotyczące dokładnego punktu końcowego, z którym zostaje wykonana próba połączenia, oraz dane wejściowe do przekazania dla algorytmu. W przypadku pomyślnego transferu danych, funkcja zapisuje zmienne otrzymane w wyniku działania algorytmu i przekierowuje użytkownika na ekran wynikowy, natomiast w momencie wystąpienia błędu, jego treść jest wypisywana na ekranie.

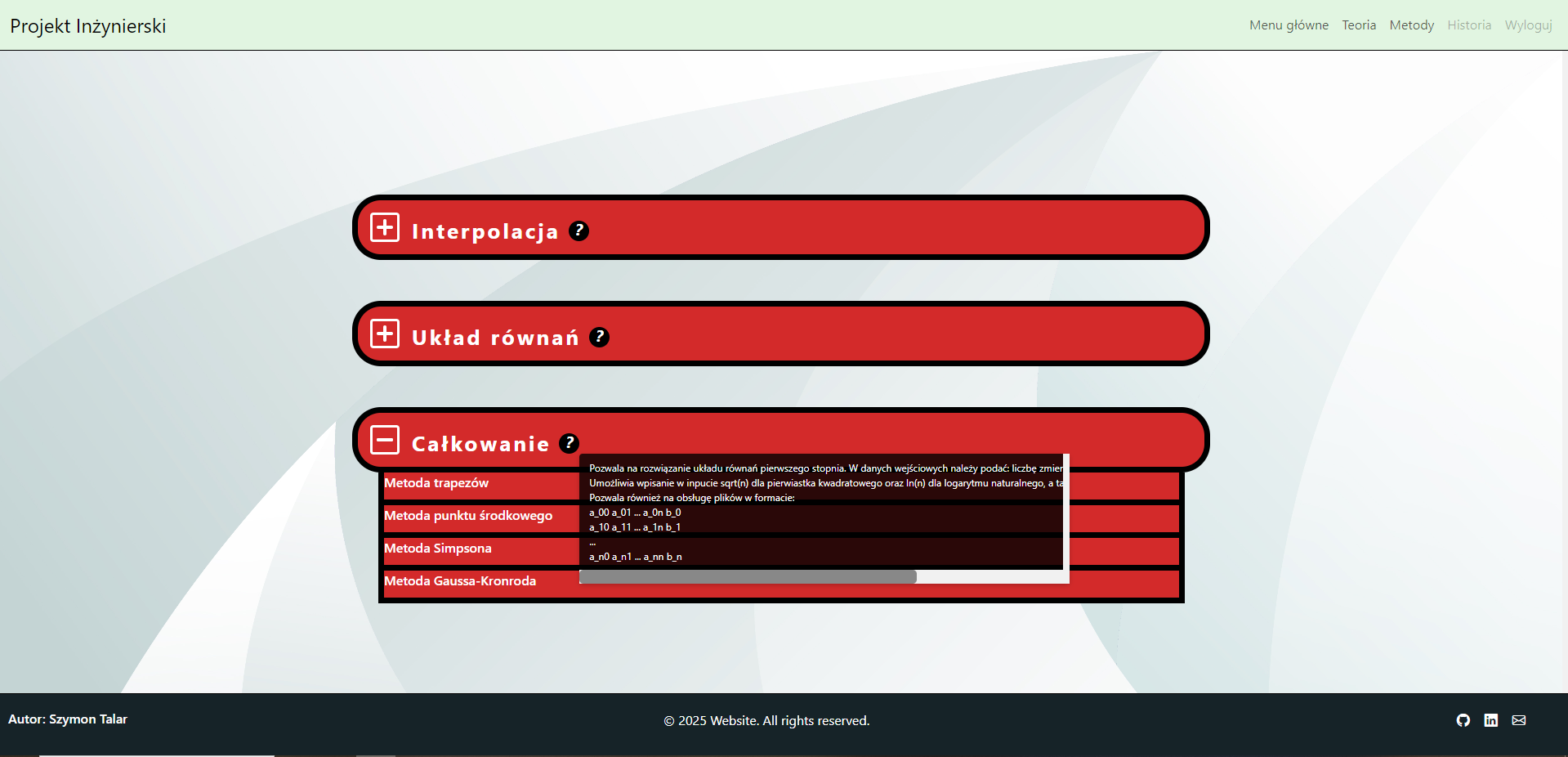
## Stworzenie GUI

*GUI* (ang. *Graphical User Interface*) jest formą prezentacji danych i zestawem narzędzi graficznych umożliwiających użytkownikowi na wchodzenie w interakcję z danym programem. W tym projekcie przybiera on formę wszystkich ekranów dostępnych na stronie, wraz z dobranymi do niej stylami mającym za zadanie ułatwienie poruszanie się po jej poszczególnych ekranach. Ideą, która przyświecała przy wykonywaniu elementów graficznych dla tego projektu była prostota interfejsu graficznego, pozwalająca na proste lokalizowanie kluczowych dla użytkownika miejsc na stronie, a także czytelność znajdującej się tam zawartości. Każda strona znajdująca się w projekcie składa się z trzech części:

*-* paska górnego, na którym znajduje się nazwa projektu (w przypadku wydania tej strony do użytku publicznego byłoby to miejsce na umieszczenie jej nazwy i/lub logo) oraz pasek nawigacji, który umożliwia przełączanie się pomiędzy poszczególnymi kartami. Na Rys. 5 zaprezentowany został wraz z ekranem wyszczególniającym dostepne metody do użytku. W momencie jego tworzenia użytkownik nie był zalogowany na stronie i korzystał z niej jako gość, więc część podstron została dla niego zablokowana.

*-* sekcji głównej, której zawartość będzie się zmieniała w zależności od podstrony na którą wejdzie użytkownik. Niezależnie od jego położenia, zapisane style zapewniają ustawienie minimalnej wysokości tak, aby strona obejmowała cały ekran, oraz domyślne tło.

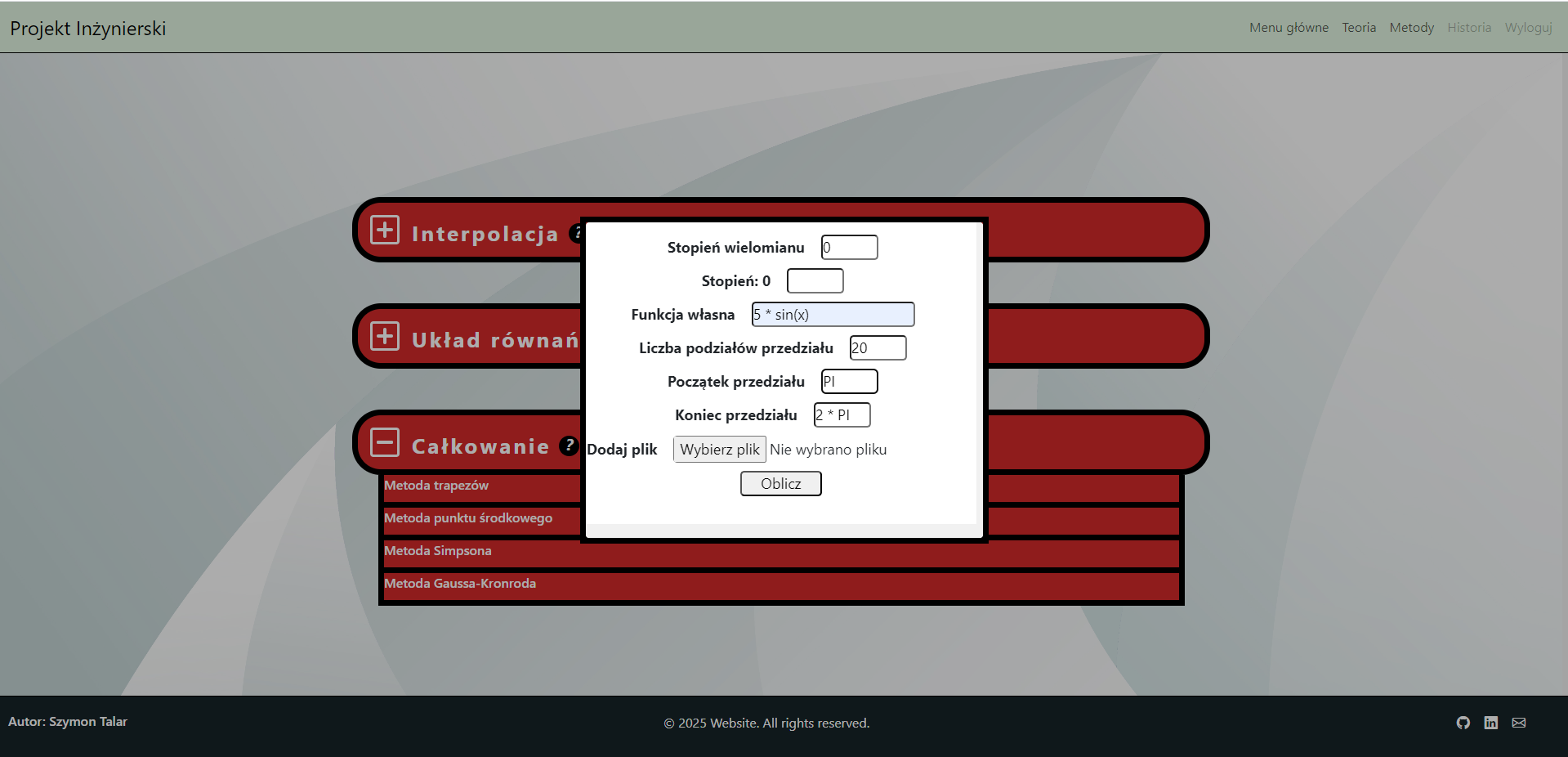
*-* stopki, na której umieszczono możliwości kontaktu z autorem strony oraz krótki napis o zastrzeżeniu praw autorskich. W przypadku umieszczenia strony do użytku publicznego, ta sekcja mogłaby również zawierać informacje nt. polityki prywatności czy też zmiany szaty graficznej strony, jak możliwość przełączania się pomiędzy trybem jasnym i ciemnym.



Rys. 5. Ekran przedstawiający listę metod możliwych do wybrania dla dalszego wykonania obliczeń. Źródło: opracowanie własne

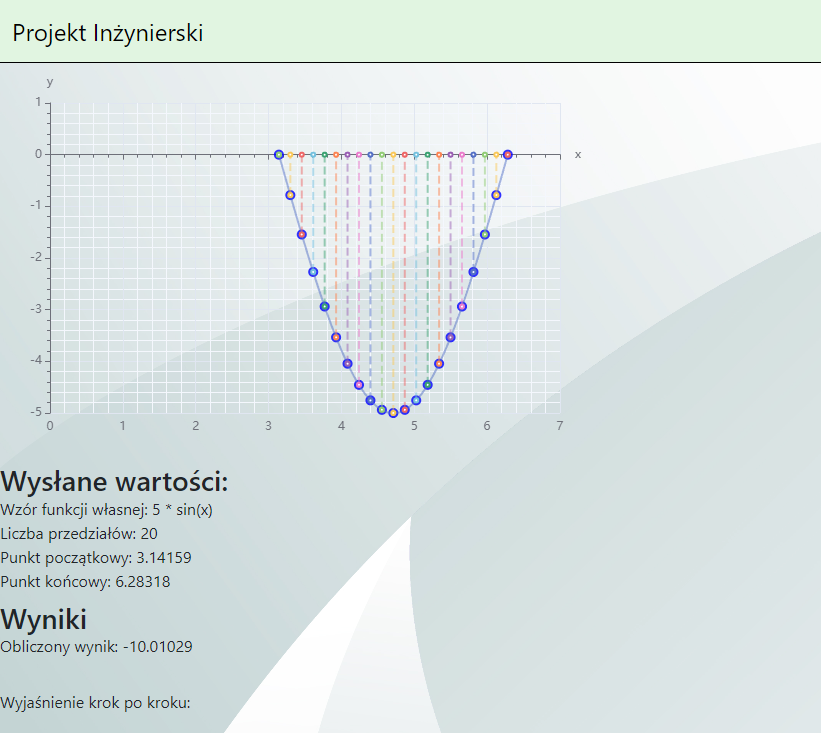
Dla każdej istniejącej metody został stworzony tooltip podpowiadający użytkownikowi jaki zakres możliwości oferuje wybrana przez niego metoda, a także informuje o formacie w jakim należy wprowadzić dane w celu wykonania obliczeń.

Po wybraniu dowolnej metody, na środku ekranu pojawi się dodatkowe okienko pozwalające wpisać dane potrzebne do wywołania pożądanego algorytmu.



Rys 6. Ekran z dodatkowym oknem pomocniczym dla danych wejściowych. Źródło: opracowanie własne

Jak widać, ekran pomocniczy zawiera okienka umożliwiające wpisanie niezbędnych zmiennych: liczby znanych punktów, wartości *x* oraz *y* tych punktów, a także wartość potrzebną do znalezienia.

Ponadto, możliwe są również alternatywne metody wprowadzenia danych. Pierwszą z nich jest ‘funkcja własna’ która pozwala użytkownikowi na podanie dowolnie wybranej funkcji w postaci tekstu, w przypadku gdyby nie dało się zapisać jej w postaci wielomianu. W przypadku podania własnej funkcji, pola ‘Stopień wielomianu’ i kolejne stopnie nie są już konieczne, co również podane jest w tooltipie opisującym korzystanie z metody. Drugą funkcjonalnością podaną użytkownikowi jest możliwość wprowadzenia danych z pliku tekstowego. Po wybraniu go, i uzupełnieniu w odpowiedni sposób pola automatycznie zostają uzupełnione. Po uzupełnieniu takiego formularza w dowolny sposób i wciśnięciu przycisku ‘Oblicz’, dane te są przekazywane do odpowiedniego kontrolera, w tym przypadku odpowiadająca całkowaniu metodą punktu środkowego. Zastosowanie powyższego rozwiązania ma na celu zwiększenie komfortu użytkownika, który przy pomocy podanych podpowiedzi i alternatywnych metod wprowadzania danych jest w stanie w prostszy sposób wykonać pożądane przez niego operacje. Po wykonaniu obliczeń, program przenosi nas na stronę zawierającą rezultat.  
  


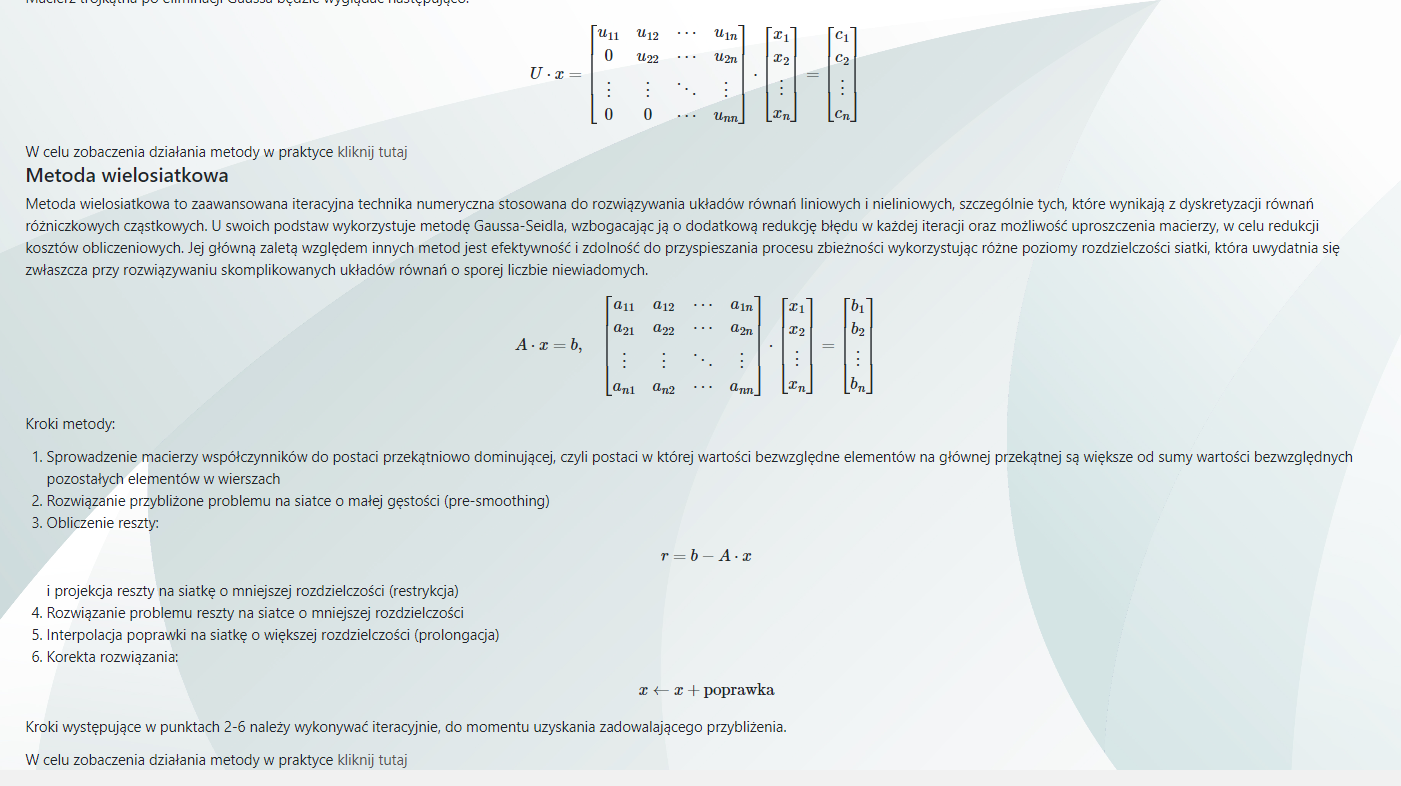
Rys 7. Ekran wynikowy dla danych wysłanych przy pomocy ekranu z Rys 6. Źródło: opracowanie własne

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

Rys 8. Wyjaśnienie obliczeń wykonanych przez kalkulator. Źródło: opracowanie własne

Ekran wynikowy, oprócz obliczonego rezultatu, zawsze będzie wyświetlał wszystkie dane podane przez użytkownika. Dla wybranego zagadnienia całkowania, dostępne są również: precyzyjny wykres demonstrujący podaną funkcję w granicach uzyskanych z danych wejściowych, a także proste objaśnienie wykonanych obliczeń których całość została pokazana na Rys 8. Narzędzia te mają na celu pomóc użytkownikowi w lepszym zrozumieniu rozwiązywanego zagadnienia, natomiast sama treść wyświetlanych odpowiedzi i jej zakres zależeć będzie od wybranej metody.

Dla użytkowników, którzy nie są zaznajomieni z prezentowanymi tutaj metodami i którzy mogliby mieć problemy ze skonkretyzowaniem problemu do rozwiązania, bądź też potrzebowaliby dodatkowych informacji odnośnie wykorzystywanych na stronie metod, dostępna jest również strona zawierająca teoretyczne omówienie dostępnych metod.  


Rys 9. Fragment strony teoretycznej, wraz z możliwością prezentacji przykładowych obliczeń.

Źródło: opracowanie własne

Z tego miejsca, oprócz przeczytania informacji na temat każdej z dostępnych metod, osoba wchodząca na stronę może odtworzyć przygotowany odgórnie przykład obliczeń oferowanych na tej stronie przez tę właśnie metodą.

## Automatyzacja zapytań dla przykładowych obliczeń

Po skorzystaniu z przycisku ‘kliknij tutaj’ użytkownik wykonuje zapytanie w dokładnie taki sam sposób jak poprzez skorzystanie z jednego z ekranów podobnych do widocznego na Rys 6., z jedyną różnicą będącą w ustawieniu dodatkowej flagi w danych wejściowych o nazwie *isTest*. W przypadku gdy jej wartość wynosi *true*, algorytm wykonuje zapytanie poprzez wywołanie funkcji *GetChatGptResponse(String prompt)*.  

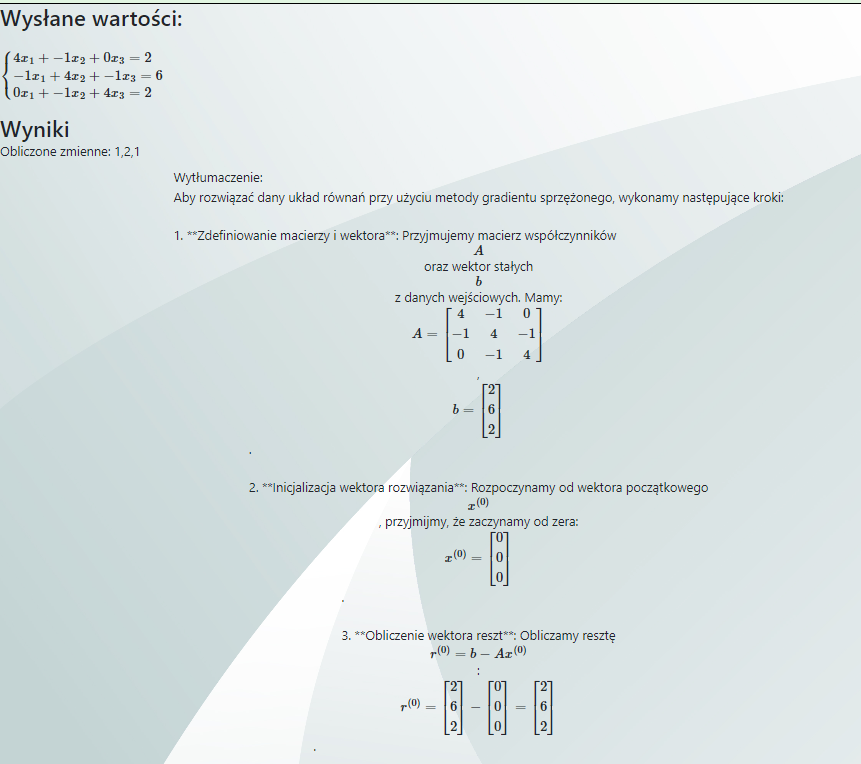

Fragment Kodu 13. Funkcja wysyłająca zapytanie do zewnętrznego API OpenAI.

Źródło: opracowanie własne

Ta funkcja składa się z:  
- zmiennej *client*, która w tym przypadku służy wyłącznie do ustawienia maksymalnego czasu oczekiwania na odpowiedź zwrotną. W przypadku przekroczenia limitu, ekran wynikowy pokazuje wszystkie pozostałe informacje, ignorując brak dostępnego wytłumaczenia.

- zmiennej *requestBody* do której przekazywane są dane niezbędne dla wykonania zapytania, takie jak wykorzystywany model sztucznej inteligencji i wiadomość przekazywana do zapytania jako prompt.

- zmiennej *request*, tworzącej zapytanie na podstawie promptu i dokonującej autentykacji użytkownika przy pomocy wygenerowanego klucza na stronie *OpenAI*

W przypadku udanego zapytania otrzymywana jest wygenerowana odpowiedź zagnieżdżona wewnątrz *responseObject*, natomiast jeśli wystąpi jakikolwiek błąd zapytania wyrzucany jest wyjątek *RuntimeException*. Na Rys 10. zaprezentowany został fragment przykładowej odpowiedzi na ekranie wynikowym po wykonaniu zapytania dla układu równań metodą wielosiatkową.  


Rys 10. Fragment odpowiedzi otrzymanej po testowym zapytaniu dla układu równań metodą wielosiatkową.

Źródło: opracowanie własne

Zaimplementowane rozwiązanie, choć w sporym stopniu poprawia doświadczenia użytkownika poprzez wgląd do szczegółowego opisu rozwiązania, postawiło również wyzwanie pod kątem implementacji. Albowiem, aby móc skorzystać z tej możliwości, należy:

1. Najpierw uzyskać dostęp do zewnętrznego *API* pozwalającego na wykonywanie takich zapytań. Taki dostęp jest zapewniany wyłącznie po dokonaniu pewnej opłaty, a następnie wprowadzenia dokładnego adresu *URL* na który mają zostać kierowane zapytania wraz z kluczem służący do walidacji osoby korzystającej. W ramach tej pracy nie wzięto pod uwagę innych modeli sztucznej inteligencji niż ten zaproponowany przez *OpenAI*, tak więc niemożliwe jest porównanie kosztów, wydajności czy też poprawności odpowiedzi generowanych na podstawie zapytań wbudowanych w aplikacji.
2. Mając dostęp do *API,* należało także poznać dokładną nazwę modelu do którego kierowane ma być zapytanie. W tym celu został stworzony osobny poboczny projekt, który na podstawie danych z prywatnego konta jest w stanie wypisać do pliku wszystkie dostępne wersje modelu. Nazwa tego modelu została następnie wykorzystana w tej pracy w postaci literału łańcuchowego podczas inicjowania zmiennej *requestBody* we Fragmencie Kodu 13.

## Zapis i odczyt danych

Wszystkie obliczenia wykonywane przez zalogowanego użytkownika, z pominięciem tych testowych, są zapisywane w bazie danych. Ma to na celu umożliwić późniejsze powtórzenie tego samego zapytania bez ponownego wprowadzania danych, a także daje możliwość eksportu do pliku tekstowego uzyskanej odpowiedzi. Jest to możliwe dzięki wykorzystaniu prostych repozytoriów, których celem jest jedynie zapis danych poprzez wbudowaną domyślnie metodę *save()* oraz utworzoną na potrzeby aplikacji metodę *findByUsername()*.

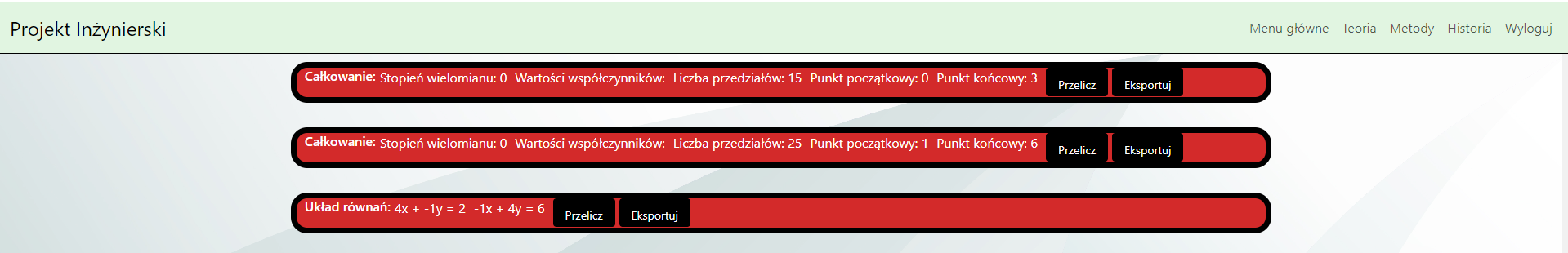


Fragment Kodu 14. Przykład repozytorium obsługującego dane z obliczanych metod.

Źródło: opracowanie własne

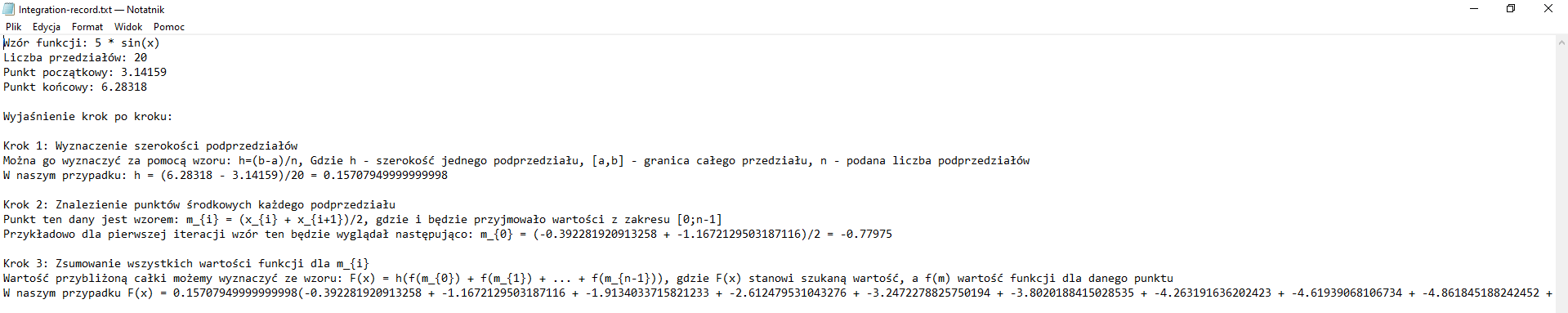
Wywoływana jest ona za każdym razem, gdy użytkownik wykonuje obliczenia z ekranu z listą metod. Każdy rodzaj danych wejściowych, dla każdej metody numerycznej posiada wyodrębione repozytorium do wykonywania na nim operacji zapisu oraz odczytu rekordów, stąd też analogicznie w przypadku chęci dodania nowej metody do aplikacji – musiałoby zostać utworzone kolejne tego typu repozytorium. Warte uwagi jest tutaj zastosowanie osobnego interfejsu o nazwie *Savable*. Jest ono zupełnie puste w środku, i jego jedynym celem jest uzyskanie prostego dostępu do zmiennych zapisywanych wcześniej w bazie danych. Implementują go wszystkie klasy modeki, utworzone dla danych wejściowych, co pozwala na późniejsze przechowanie tych zmiennych w pojedyńczej liście.

Aby obejrzeć historię, wystarczy wejść w zakładkę ‘Historia’



Rys 11. Ekran z historią obliczeń. Źródło: opracowanie własne

Opracowany z myślą o dalszej pracy użytkownika nad wykonywanymi obliczeniami, umożliwia przeliczenie metody oraz eksport danych do pliku tekstowego. Daje to dodatkowe możliwości edukacyjne, poprzez łatwiejsze ponawianie kolejnych obliczeń przy tych samych danych i innych metodach, bądź też podobnych danych w celu zobaczenia subtelnych różnic występujących z racji na zmianę danych wejściowych.



Rys 12. Plik tekstowy wygenerowany przez aplikację. Źródło: opracowanie własne

## Opis widoków w aplikacji

Cała aplikacja składa się z następujących widoków:

1. **Menu główne**

Jest to pierwszy widok który zobaczy użytkownik zaraz po otworzeniu strony.   
Z tego ekranu użytkownik ma możliwość: zarejestrować swoje konto na stronie, zalogować się podając dane do już istniejącego konta, bądź też wejść na stronę jako użytkownik niezarejestrowany i przejść od razu do ekranu z metodami. Celem tego ekranu jest danie użytkownikowi możliwości rejestracji i logowania się na stronę,   
co z kolei pozwala na swobodę w wyborze jak będzie chciał z niej korzystać. Podczas korzystania z niej jako zalogowany użytkownik, dane podawane w postaci parametrów będą zapisywane w bazie danych, natomiast użytkownik bez logowania zostanie pozbawiony tej możliwości.

1. **Strona powitalna**

Zaraz po zalogowaniu się albo wejściu na stronę bez logowania, na ekranie pojawi się strona powitalna. Jej zadaniem jest krótkie przedstawienie użytkownikowi możliwości przez nią oferowanych w postaci opisu każdej podstrony. Z tego miejsca, użytkownik może się udać do ekranu przedstawiającego teoretyczny opis metod znajdujących się na stronie, do ekranu z samymi metodami możliwymi do obliczenia oraz do historii wyników z obliczeń już wykonanych, jeśli takowe już miały miejsce.

1. **Metody**

W tym miejscu jest dostępny podgląd wszystkich dostępnych metod do wykorzystania. Obok każdej metody zawartej na stronie umieszczony został tooltip z podpowiedzią dla użytkownika, jaki zakres możliwości jest oferowany przez nią oraz w jaki sposób będzie mógł wprowadzić dane w celu wykonania obliczeń. Każda metoda po kliknięciu na nią, rozwija osobne menu zawierające wszystkie przewidziane sposoby na rozwiązanie tej metody, a z kolei każdy ze sposobów,  po naciśnięciu wyświetla okienko z danymi potrzebnymi do wprowadzenia aby wykonać obliczenia. To okno zmienia się w zależności od wybranej metody, tak, aby zawrzeć wszystkie potrzebne dane.

1. **Teoria**

Na tym ekranie jest widoczne teoretyczne omówienie metod dostępnych na stronie.

Oprócz krótkiego opisu teoretycznego, zawierają wybrane grafiki/animacje ilustrujące działanie danej metody, oraz krótki opis wzoru pomagający lepiej zrozumieć działanie wybranego przez użytkownika zagadnienia. Ponadto, każda sekcja zawiera na końcu odnośnik do testowego przykładu w kalkulatorze w jaki sposób wykonywane są obliczenia wewnątrz.

1. **Wyniki**

Ta podstrona wyświetla rezultaty obliczeń wcześniej zaznaczonej metody.  
W zależności od tego, z jakiej skorzystamy, strona będzie dostosowywała wyświetlaną zawartość. Da się dostać do niej zarówno poprzez wybranie własnoręcznie metody z menu i wprowadzenie niezbędnych danych, jak i wybranie przykładu znajdującego się w sekcji teoretycznej. Do każdego przykładu z sekcji teoretycznej generuje się również tekst przy pomocy sztucznej inteligencji opisujący w szczegółowy sposób działanie metody krok po kroku, co pozwala stopniowo śledzić w jaki sposób został uzyskany ostateczny wynik. Jeśli użytkownik zamiast tego samemu wprowadza dane, opis generowany automatycznie zostanie zamieniony na krótszy, wykonany już wewnątrz kalkulatora przedstawiający pokrótce kolejne kroki. W ten sposób, użytkownik będzie mógł zarówno zobaczyć wnikliwy opis danej metody wraz z jej szczegółowym wytłumaczeniem, oraz unikniemy sytuacji w której tekst wygenerowany automatycznie wprowadzi użytkownika w błąd, co mogłoby mieć hipotetycznie miejsce, zwłaszcza w przypadku bardziej złożonych danych wejściowych. Wśród wyświetlanej zawartości, zawsze będą znajdowały się wszystkie dane które zostały wprowadzone do programu, a także obliczony wynik algorytmu. Ponadto, jeśli to będzie możliwe, zostanie zaprezentowany wykres obrazujący w lepszy sposób dane oraz wynik tak, aby użytkownik mógł łatwiej zrozumieć działanie danej metody, a także wzór ogólny ukazujący istotę rozwiązywanego problemu.

1. **Historia wyników**

Tam znajduje się lista ostatnich obliczeń wykonywanych przez użytkownika. Dzięki temu, użytkownik ma możliwość ponownej analizy konkretnych przykładów algorytmów, z których najczęściej korzysta wraz z już podstawionymi wcześniej wartościami. Po wybraniu danego zapisu, strona przekierowuje nas ponownie na ekran wyników, tak, jakbyśmy wybrali tę metodę i dane z menu głównego. Ponadto, zostaje również udostępniona możliwość wygenerowania pliku tekstowego zawierającego pełen zapis danych wejściowych oraz wyjściowych danego przykładu, co może pomóc użytkownikowi w jego dalszej samodzielnej analizie.

# Testy aplikacji

W celu przetestowania sprawnego działania wszystkich komponentów, zdecydowano się na wykorzystanie dwóch technik:

## Implementacja testów jednostkowych

Ten sposób sprawdzenia działania kodu był niezwykle przydatny dla pisania algorymów metod, które zostały wykorzystane. Dzięki nim całość sprawdzenia działania danej metody sprowadzała się zaledwie do utworzenia kilku przypadków testowych, dla których wyniki były weryfikowane poza stroną korzystając z innych źródeł internetowych bądź też korzystając z samodzielnie wykonanych obliczeń, a następnie wprowadzenie utworzonego scenariusza jako obiekt z danymi dla algorytmu. Po tym następowało wywołanie funkcji, która jeśli zwróciła przewidywane rezultaty dla wszystkich przypadków – była uznawana za sprawną. Warto jednak w tym miejscu zaznaczyć, że przejście założonych testów nie daje gwarancji pełni sprawności algorytmów. Wynika to z faktu, iż testy jednostkowe które zostały zaimplementowane zawsze ograniczają się wyłącznie do pewnej puli przypadków, która niezależnie od ich doboru zawsze może pozostawiać ryzyko pominięcia pewnych specyficznych okoliczności dla których algorytm popełni błąd. W tym projekcie, ograniczono się do funkcji trzeciego rzędu jako tych najwyższego rzędu w testach jednostkowych, przez co błędy występujące wyłącznie dla wyższych rzędów mogły zostać niezauważone. Ponadto, nawet dla funkcji które uwzględniono w testach, z powodu błędów wynikających z braku odpowiedniej precyzji zmiennych oraz błędów wynikających z samej natury metod, których celem często jest wyłącznie przybliżenie poprawnego wyniku, a nie podanie jego dokładnej wartości, wymuszone było użycie porównania obejmującego błąd pomiarowy. We Fragmencie Kodu 15 ten błąd jest widoczny w wywołaniu funkcji *assertEquals* jako trzeci argument. Dla niektórych metod, takich jak interpolacja trygonometryczna, dodatkowym problemem był brak zewnętrznych źródeł pomocniczych podczas testowania algorytmu. Wymuszało to ręczne testowanie stosunkowo prostych przypadków, w celu wykrycia potencjalnych błędów algorytmu.   
  
****

Fragment Kodu 15. Przykładowy test jednostkowy sprawdzający działanie metody trapezów dla całkowania w przypadku funkcji liniowych. Źródło: opracowanie własne

Obraz zawierający tekst, Czcionka, zrzut ekranu, typografia

Opis wygenerowany automatycznie

Rys 13. Fragment wyników klasy testowej dla Fragmentu Kodu 15. Źródło: opracowanie własne

Na podstawie Fragmentu Kodu 15 został zaprezentowany fragment zaimplementowanych testów. Przykładowo, dane wejściowe oznaczone jako *integrationData1* oznaczają wielomian o współczynnikach 3 oraz 1, czyli funkcję , która to jest funkcją 1 stopnia, liczoną na przedziale [0;5] i dzieloną w trakcie całkowania na 100 podprzedziałów. W analogiczny sposób zostały wprowadzone pozostałe testy. Następnie funkcja testowa wypisuje wyniki na ekran, oraz porównuje otrzymane rezultaty z oczekiwaną wartością. Na Rys 13. widać uzyskane w ten sposób wyniki, które pokazują jednocześnie błędy pomiarowe występujące we wdrożonych algorytmach, jak i konieczność zastosowania wcześniej wspomnianego błędu pomiarowego. Gdyby dla pierwszego przypadku, czyli *result1*, został zastosowany błąd o wartości 0.1 zamiast 1 jak to ma miejsce dla późniejszych przypadków, test nie przeszedłby ze względu na zbyt duży odchył wartości uzyskanej od wartości oczekiwanej.

## Samodzielne przejście ścieżek procesów

Z racji na obecnie niewielką skalę projektu, zamiast automatyzować bardziej zaawansowane testy jak testy integracyjne lub testy *E2E* (ang. *end-to-end*), część testów została wykonana manualnie. Wśród wykonywanych w ten sposób testów funkcjonalności znajdują się między innymi:

- testy obejmujące zakres działania bazy danych *H2* potrzebnej do zapisywania użytkowników. Funkcjonalność tego segmentu była niezbędna do kontynuowania dalszych prac ponieważ jest to pierwszy ekran znajdujący się na stronie, a także część możliwości strony jest uzależniona od prawidłowego logowania się na stronę, tak więc w przypadku braku możliwości logowania strona utraciłaby sporą część dostępnych funkcjonalności

- testy prawidłowego wyświetlania się wyników algorytmów, które obejmowały przejście dla wszystkich algorytmów do ekranu wynikowego, takiego jak na Rys. 7. Z racji na zastosowanie osobnych punktów końcowych dla każdej metody, test ten również musiał być powtarzany osobny dla każdego istniejącego przypadku. Wraz z tym testem, sprawdzana była również możliwość wyświetlenia alternatywnej metody rozwiązania problemu.

- testy bazy *MongoDB*, zarówno przy pomocy zewnętrznego *GUI*(*MongoDb Compass*), jak i poprzez sprawdzanie rekordów znajdujących się w historii wyników po wykonaniu kilku różnych algorytmów.

Pomyślne przejście wyżej wymienionych testów definiowało pomyślne ukończenie projektu.

# Podsumowanie

Strona, która została zaimplementowana na rzecz projektu, przeszła niezbędne testy w celu potwierdzenia sprawności funkcjonalności wymienionych we wcześniejszych rozdziałach. Pomimo braku możliwości pełnego przetestowania wszystich wdrożonych metod i możliwości istnienia błędów obliczeniowych, obecny stan umożliwia już na korzystanie ze strony i wykonywanie obliczeń, które przynajmniej dla prostych scenariuszy wyświetlą na ekranie wynik bliski temu rzeczywistemu. W połączeniu z wprowadzenymi dodatkowymi funkcjonalnościami, tworzy to projekt który potencjalnie daje możliwości obecnie niedostępne na innych stronach w ich darmowych wersjach, i które jednocześnie mogą przynieść dodatkowe korzyści w kontekście powszechnego użytkowania przez osoby, które chcą się nauczyć więcej o wybranych metodach numerycznych. Pomimo to, strona nadal pozostawia wiele możliwości do dalszego rozwoju. Pomijając ewentualną naprawę istniejących błędów i dalsze usprawnienie algorytmów w celu zwiększenia precyzji obliczeń, wśród potencjalnych nowych treści jakie dałoby się wprowadzić i które powinny jeszcze bardziej zwiększyć jej wartość edukacyjną wymieniłbym:

- dodanie kolejnych algorytmów, urozmajcając tym samym dostępną obecnie ofertę

- zwiększenie dostępności dla użytkowników mających określone trudności w korzystaniu ze standardowych stron, poprzez dodanie takich elemntów jak wdrożenie *WCAG*(ang. *Web Content Accessibility Guidlines*), dodanie atrybutów *ARIA*(ang. *Accessible Rich Internet Applications*), możliwość głosowego komunikowania się ze stroną, wprowadzenie możliwości dostosowania wielkości czcionki oraz przycisków na stronie, czy też uwydajnienie obliczeń wykonywanych a *Javie* np. poprzez zastosowanie wielowątkowości dla użytkowników dysponujących mniej zaawansowanymi sprzętami elektronicznymi.

- możliwość większej interakcji ze stroną, np. poprzez możliwość natychmiastowej zmiany danych w celu usprawnienia wykonywania wielu podonych obliczeń, a także wbudowanie interaktywnego wykresu, na którym użytkownik mógłby samemu dodawać/usuwać punkty wprowadzone na wejściu do programu.

- dodanie rozbudowanego opisu wyniku, zarówno przy pomocy kroków przejściowych podczas wykonywania obliczeń, jak i opisu słownego wygenerowanego przez sztuczną inteligencję, np. poprzez wykonywanie zapytań do udostępnionego przez *OpenAI API*.

# Źródła

1. Wolfram Alpha, *Interpolating polynomial calculator,* <https://www.wolframalpha.com> [dostęp: 06.01.2025r.]
2. AtoZmath, *Numerical interpolation using Lagrange's Interpolation formula*, <https://atozmath.com> [dostęp: 06.01.2025r.]
3. Online Tools, https://tools.timodenk.com [dostęp: 06.01.2025r.]
4. Shivani Makwana, *C# vs Java: Key Differences Explained*, <https://www.ifourtechnolab.com/> [dostęp: 07.01.2025r.]
5. Anastasia Grishina, Roman Ogolikhin, *Java vs. C#: What is the Right Choice for Your Project?*, <https://softteco.com/> [dostęp: 07.01.2025r.]
6. *MongoDB vs. MySQL Differences*, <https://www.mongodb.com/> [dostęp: 07.01.2025r.]
7. Anthony P. Austin, *On trigonometric interpolation in an even number of points*, Kent State University 2023

1. Wolfram Alpha, *Interpolating polynomial calculator,* <https://www.wolframalpha.com> [dostęp: 06.01.2025r.] [↑](#footnote-ref-1)
2. AtoZmath, *Numerical interpolation using Lagrange's Interpolation formula*, <https://atozmath.com> [dostęp: 06.01.2025r.] [↑](#footnote-ref-2)
3. Online Tools, https://tools.timodenk.com [dostęp: 06.01.2025r.] [↑](#footnote-ref-3)
4. Shivani Makwana, *C# vs Java: Key Differences Explained*, <https://www.ifourtechnolab.com/> [dostęp: 07.01.2025r.] [↑](#footnote-ref-4)
5. Anastasia Grishina, Roman Ogolikhin, Java vs. C#: What is the Right Choice for Your Project?, <https://softteco.com/> [dostęp: 07.01.2025r.]

   [↑](#footnote-ref-5)
6. *MongoDB vs. MySQL Differences*, <https://www.mongodb.com/> [dostęp: 07.01.2025r.] [↑](#footnote-ref-6)
7. Anthony P. Austin, *On trigonometric interpolation in an even number of points*, Kent State University 2023 [↑](#footnote-ref-7)