

Nazwa przedmiotu (projekt)  
tu Tytuł Twojej Pracy

Imię Nazwisko  
nr albumu: 123456  
kierunek: Informatyka  
rok: 2025/2026

3 lutego 2026

**Kierunek studiów:** [Wpisz kierunek, np. Zarządzanie]

**Specjalność:** [Wpisz specjalność]

**Poziom studiów:** [I stopnia / II stopnia]

**Profil studiów:** [praktyczny / ogólnoakademiczny]

**Forma studiów:** [stacjonarne / niestacjonarne]

## ABSTRAKT PRACY DYPLOMOWEJ

**Imię i nazwisko studenta:** [Twoje Imię i Nazwisko]

**Numer albumu:** [Twój Numer Albumu]

**Tytuł pracy w języku polskim:** [Tytuł Pracy]

**Tytuł pracy w języku angielskim:** [Title of the Thesis]

**Promotor:** [Tytuł/Stopień naukowy, Imię i Nazwisko]

**Słowa kluczowe (w języku polskim):**

Słowo 1, słowo 2, słowo 3, słowo 4, słowo 5

**Abstrakt (w języku polskim):**

[Tutaj wpisz treść abstraktu w języku polskim. Powinien on zawierać cel pracy, krótki opis metodologii oraz najważniejsze wnioski. Zazwyczaj tekst ten nie powinien przekraczać jednej strony A4 wraz z pozostałymi elementami.]

**Keywords (in English):**

Keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

**Abstract (in English):**

[Place the English translation of your abstract here. Ensure that it matches the Polish version in terms of content and professional terminology.]

---

(podpis studenta)

---

(podpis promotora)

# Spis treści

Wstęp . . . . .	4
1. Cel i zakres pracy . . . . .	7
1.1. Cel pracy . . . . .	7
1.2. Cele szczegółowe . . . . .	7
1.3. Zakres pracy . . . . .	7
1.4. Struktura pracy . . . . .	8
2. Problematyka projektu . . . . .	9
2.1. Opis problemu . . . . .	9
2.2. Cel i zakres pracy . . . . .	9
2.3. Uzasadnienie wyboru tematu . . . . .	9
2.4. Pytania badawcze . . . . .	9
3. Analiza wymagań i projekt rozwiązania . . . . .	10
3.1. Wymagania funkcjonalne . . . . .	10
3.2. Wymagania niefunkcjonalne . . . . .	10
3.3. Architektura rozwiązania . . . . .	10
3.4. Projekt procesu komplikacji . . . . .	10
3.5. Struktura projektu . . . . .	11
4. Implementacja i uruchomienie . . . . .	12
4.1. Środowisko i narzędzia . . . . .	12
4.2. Struktura projektu . . . . .	12
4.3. Proces uruchomienia . . . . .	12
4.4. Obsługa błędów . . . . .	12
5. Testy i weryfikacja . . . . .	13
5.1. Strategia testów . . . . .	13
5.2. Scenariusze weryfikacyjne . . . . .	13
5.3. Wyniki testów . . . . .	13
5.4. Ograniczenia i możliwości rozwoju . . . . .	13
Podsumowanie . . . . .	14
Spis rysunków . . . . .	15
Spis tabel . . . . .	16
Spis listingów . . . . .	17

# Wstęp

Wstęp powinien zawierać krótki opis tematu pracy, jej cel oraz zakres. Możesz również wspomnieć o strukturze dokumentu i metodach badawczych, które zostaną użyte.

Poniżej znajduje się przykładowy fragment tekstu zawierający cytowanie oraz listing kodu źródłowego. Wzory tabel, grafik itp. Stanowią one jedynie wzór do dalszego rozwinięcia pracy.

## Przykład cytowania

To jest przykład cytowania Lamport, 1994. Poniżej kod:

## Przykład kodu źródłowego w CPP

Listing 1. Program sumujący dwie liczby w Pythonie

```
1 def main():
2     liczba1 = float(input("Podaj pierwszą liczbę:"))
3     liczba2 = float(input("Podaj drugą liczbę:"))
4     suma = liczba1 + liczba2
5     print(f"Suma: {suma}")
6
7 if __name__ == "__main__":
8     main()
```

## Przykład kodu źródłowego w JAVA

Listing 2. Program sumujący dwie liczby w Javie

```
1 import java.util.Scanner;
2
3 public class Suma {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Podaj pierwszą liczbę:");
7         float liczba1 = scanner.nextFloat();
8         System.out.print("Podaj drugą liczbę:");
9         float liczba2 = scanner.nextFloat();
10        float suma = liczba1 + liczba2;
11        System.out.println("Suma: " + suma);
12        scanner.close();
13    }
14}
```

## Przykład kodu źródłowego w CPP

Listing 3. Program sumujący dwie liczby w C++

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     float liczba1, liczba2;
6     cout << "Podaj pierwszą liczbę:" ;
7     cin >> liczba1;
8     cout << "Podaj drugą liczbę:" ;
9     cin >> liczba2;
10    float suma = liczba1 + liczba2;
11    cout << "Suma:" << suma << endl;
12    return 0;
13 }

```

## Przykład zapytania SQL

Listing 4. Zapytanie SQL wybierające wszystkie rekordy z tabeli 'uzytkownicy'

```

1 SELECT * FROM uzytkownicy;

```

## Przykład tabeli

Tabela 1

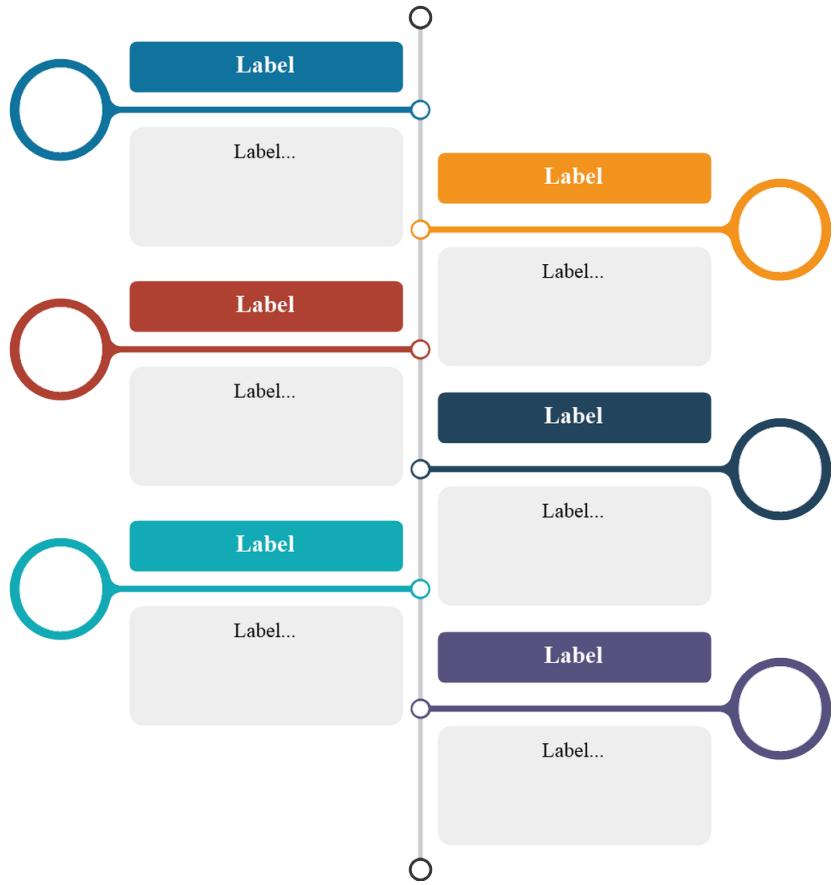
Przykładowa tabela z trzema kolumnami sformatowana zgodnie z zasadami

Nazwa elementu	Opis funkcjonalności	Wartość [j.]
Element A	Opis szczegółowy A	10
Element B	Opis szczegółowy B	20
Element C	Opis szczegółowy C	30

Źródło: opracowanie własne

## Przykład obrazów

Jak pokazaono na rysunku 1, poniżej znajduje się przykładowy schemat systemu.



Rysunek 1. Dodatkowy schemat systemu (import z pliku PNG)

*Źródło: opracowanie własne*

Na rysunku 1 zaprezentowano strukturę systemu w ujęciu ogólnym.

### Przykład odwołania do bibliografii

W literaturze przedmiotu (Kowalski i Nowak, 2026) omówiono zagadnienia związane z...

### Przykład odwołania do netografii

W Internecie dostępne są zasoby dotyczące... (Goossens, Mittelbach i Samarin, 1997)

# **1. Cel i zakres pracy**

## **1.1. Cel pracy**

Celem niniejszej pracy inżynierskiej jest zaprojektowanie i wykonanie rozwiązania informatycznego wspierającego przygotowanie dokumentów w L<sup>A</sup>T<sub>E</sub>X oraz automatyzującego proces komplikacji w ujednoliconym środowisku uruchomieniowym. Projektowy charakter pracy obejmuje analizę wymagań, opracowanie architektury, implementację oraz weryfikację poprawności działania przygotowanego rozwiązania.

## **1.2. Cele szczegółowe**

W ramach realizacji celu głównego wyznaczono następujące cele szczegółowe:

- analiza problemu i zdefiniowanie wymagań funkcjonalnych i niefunkcjonalnych,
- zaprojektowanie struktury systemu oraz procesu komplikacji dokumentów,
- implementacja rozwiązania wraz z mechanizmami automatyzacji uruchomienia,
- przygotowanie scenariuszy weryfikacyjnych i ocena rezultatów.

Zgodnie z ujęciem przedstawionym w literaturze, nowoczesne systemy informacyjne powinny być projektowane z myślą o powtarzalności procesów oraz możliwości automatyzacji, co bezpośrednio wpływa na jakość i niezawodność wytworzonych rezultatów (Kowalski i Nowak, 2026).

## **1.3. Zakres pracy**

Zakres pracy obejmuje:

- opracowanie koncepcji systemu do tworzenia dokumentów oraz jego architektury,
- dobór technologii wspierających komplikację w kontenerach Docker,
- implementację skryptów uruchomieniowych i konfiguracji projektu,
- weryfikację poprawności generowania dokumentu wynikowego.

Poza zakresem pracy pozostają: rozbudowane badania wydajnościowe, tworzenie alternatywnych silników składu oraz pełna integracja z zewnętrznymi systemami CI/CD.

## **1.4. Struktura pracy**

Praca została podzielona na pięć rozdziałów. Rozdział pierwszy przedstawia cel oraz zakres. Rozdział drugi opisuje problematykę projektu i uzasadnia wybór tematu. Rozdział trzeci zawiera analizę wymagań oraz projekt systemu. Rozdział czwarty prezentuje implementację i sposób uruchamiania rozwiązania. Rozdział piąty obejmuje testy i ocenę rezultatów.

## **2. Problematyka projektu**

Rozdział przedstawia problem projektowy związanego z przygotowaniem dokumentów w L<sup>A</sup>T<sub>E</sub>X oraz trudnościami w zapewnieniu powtarzalnego procesu komplikacji na różnych stanowiskach. Kluczowym wyzwaniem jest ujednolicenie środowiska narzędziowego oraz uproszczenie procesu tworzenia pliku wynikowego dla użytkownika końcowego.

### **2.1. Opis problemu**

W praktyce akademickiej często występują rozbieżności w konfiguracji narzędzi L<sup>A</sup>T<sub>E</sub>X, co skutkuje błędami komplikacji, różnicami w wyniku składu lub brakiem możliwości odtworzenia wyników na innym komputerze. Problemem jest również złożoność procesu komplikacji, wymagającego znajomości wielu parametrów i narzędzi pomocniczych. Projekt ma na celu uproszczenie tego procesu oraz zapewnienie powtarzalności wyników.

### **2.2. Cel i zakres pracy**

Główym celem projektu jest przygotowanie rozwiązania, które umożliwia komplikację dokumentów L<sup>A</sup>T<sub>E</sub>X w izolowanym środowisku kontenerowym oraz dostarcza prosty mechanizm uruchomienia procesu komplikacji. Zakres obejmuje analizę wymagań, projekt architektury, implementację i weryfikację działania rozwiązania.

### **2.3. Uzasadnienie wyboru tematu**

Wybór tematu wynika z praktycznych potrzeb studentów i promotorów w zakresie tworzenia prac dyplomowych oraz rosnącej popularności narzędzi kontenerowych w inżynierii oprogramowania. W literaturze podkreśla się znaczenie automatyzacji procesu składu i jego niezawodności (Goossens, Mittelbach i Samarin, 1997).

### **2.4. Pytania badawcze**

W ramach projektu sformułowano następujące pytania:

- W jaki sposób ujednolicić środowisko komplikacji dokumentów L<sup>A</sup>T<sub>E</sub>X na różnych systemach?
- Jak zaprojektować proces komplikacji, aby był prosty w użyciu i możliwy do automatyzacji?
- Jakie wymagania funkcjonalne i niefunkcjonalne są kluczowe dla użytkownika końcowego?

### **3. Analiza wymagań i projekt rozwiązania**

Rozdział przedstawia wymagania oraz projekt rozwiązania, zgodny z charakterem pracy inżynierskiej o profilu projektowym. Opis obejmuje zarówno aspekty funkcjonalne, jak i niefunkcjonalne, a także strukturę systemu i proces komplikacji dokumentów.

#### **3.1. Wymagania funkcjonalne**

Do kluczowych wymagań funkcjonalnych należą:

- możliwość komplikacji dokumentu L<sup>A</sup>T<sub>E</sub>X do postaci PDF w środowisku kontenerowym,
- automatyzacja wielokrotnego przebiegu komplikacji (np. dla spisu treści),
- czyszczenie plików tymczasowych po zakończeniu procesu,
- czytelna informacja o wyniku komplikacji dla użytkownika.

#### **3.2. Wymagania niefunkcjonalne**

Rozwiązanie powinno spełniać następujące wymagania jakościowe:

- przenośność pomiędzy systemami operacyjnymi,
- powtarzalność wyników komplikacji niezależnie od środowiska hosta,
- łatwość uruchomienia przez użytkownika nietechnicznego,
- bezpieczeństwo wynikające z izolacji procesu komplikacji.

#### **3.3. Architektura rozwiązania**

Architektura przyjmuje układ warstwowy: warstwa dokumentu (źródła L<sup>A</sup>T<sub>E</sub>X), warstwa automatyzacji (skrypt uruchomieniowy) oraz warstwa środowiska uruchomieniowego (kontener Docker z zainstalowanym L<sup>A</sup>T<sub>E</sub>X). Taki podział umożliwia niezależne zarządzanie treścią dokumentu i procesem budowania.

#### **3.4. Projekt procesu komplikacji**

Proces komplikacji obejmuje:

- przygotowanie katalogu roboczego i usunięcie plików pomocniczych,
- uruchomienie kontenera i wykonanie komplikacji `latexmk` w trybie PDF,
- ponowny przebieg komplikacji w celu odświeżenia spisów,
- weryfikację istnienia pliku wynikowego i przekazanie komunikatu użytkownikowi.

### **3.5. Struktura projektu**

Projekt składa się z pliku głównego dokumentu oraz rozdziałów tematycznych, uzupełnionych o pliki bibliografii i zasoby graficzne. Struktura jest zgodna z przyjętymi standardami prac inżynierskich i ułatwia dalszą rozbudowę.

## **4. Implementacja i uruchomienie**

Rozdział opisuje realizację zaprojektowanego rozwiązania oraz sposób jego uruchomienia z punktu widzenia użytkownika końcowego. Przedstawiono środowisko wykonawcze, strukturę plików i podstawowe kroki pracy z systemem.

### **4.1. Środowisko i narzędzia**

Implementacja opiera się na  $\text{\LaTeX}$  oraz narzędziu `latexmk` uruchamianym w kontenerze Docker. Takie podejście eliminuje konieczność instalacji pełnego pakietu  $\text{\LaTeX}$  na komputerze użytkownika oraz zapewnia spójność wyników.

### **4.2. Struktura projektu**

Projekt zawiera plik główny dokumentu, rozdziały tematyczne, pliki bibliografii oraz katalog z zasobami graficznymi. Taka organizacja ułatwia zarządzanie treścią i rozwój dokumentu w kolejnych iteracjach.

### **4.3. Proces uruchomienia**

Proces uruchomienia składa się z następujących kroków:

- przygotowanie treści w plikach  $\text{\LaTeX}$ ,
- uruchomienie skryptu automatyzującego komplikację,
- wygenerowanie pliku PDF w katalogu głównym projektu,
- opcjonalne czyszczenie plików tymczasowych po zakończeniu pracy.

### **4.4. Obsługa błędów**

W przypadku problemów z komplikacją użytkownik otrzymuje czytelny komunikat. Najczęstszymi przyczynami są: brak uruchomionego Dockera, błędy składni w plikach  $\text{\LaTeX}$  lub brak wymaganych zasobów graficznych. Dzięki temu możliwe jest szybkie zidentyfikowanie problemu i jego korekta.

## **5. Testy i weryfikacja**

Rozdział przedstawia sposób sprawdzenia poprawności działania rozwiązania oraz ocenę uzyskanych rezultatów. Testy skupiają się na weryfikacji kluczowych funkcji i jakości procesu komplikacji.

### **5.1. Strategia testów**

Przyjęto podejście oparte na scenariuszach użycia. Każdy scenariusz odpowiada konkretnemu wymaganiu funkcjonalnemu lub niefunkcjonalnemu, co pozwala jednoznacznie ocenić, czy wymaganie zostało spełnione.

### **5.2. Scenariusze weryfikacyjne**

Zastosowano między innymi następujące scenariusze:

- komplikacja dokumentu bez błędów i wygenerowanie pliku PDF,
- ponowna komplikacja z uwzględnieniem spisu treści i elementów pomocniczych,
- obsługa sytuacji braku uruchomionego Dockera,
- reakcja systemu na błędny zapis w pliku L<sup>A</sup>T<sub>E</sub>X.

### **5.3. Wyniki testów**

Wyniki potwierdziły poprawność działania procesu komplikacji oraz czytelność komunikatów z punktu widzenia użytkownika. Zidentyfikowano typowe źródła błędów związane z treścią L<sup>A</sup>T<sub>E</sub>X oraz dostępnością środowiska kontenerowego, co uwzględniono w instrukcjach uruchomieniowych.

### **5.4. Ograniczenia i możliwości rozwoju**

Testy nie obejmowały pełnych badań wydajnościowych ani integracji z systemami ciągłej integracji. W przyszłości możliwe jest rozszerzenie rozwiązania o automatyczną walidację szablonu, raporty z komplikacji oraz konfigurację zależną od profilu użytkownika.

## **Podsumowanie**

Tu znajdują się krótkie podsumowanie pracy — przedstaw główne wnioski, ograniczenia oraz propozycje dalszych badań. ...

## Bibliografia

Kowalski, Jan i Adam Nowak (2026). *Nowoczesne systemy informatyczne*. Warszawa:  
Wydawnictwo Naukowe.

Lamport, Leslie (1994). *LaTeX: A Document Preparation System*. Ang. 2 wyd. Reading,  
MA: Addison-Wesley.

## **Netografia**

Goossens, Michel, Frank Mittelbach i Alexander Samarin (1997). *The L<sup>A</sup>T<sub>E</sub>X Companion (informacje online)*. Ang. URL: <https://latex-project.org/> (dostęp 31.01.2026).

## **Spis rysunków**

1	Dodatkowy schemat systemu (import z pliku PNG) . . . . .	6
---	--	---

## **Spis tabel**

- 1 Przykładowa tabela z trzema kolumnami sformatowana zgodnie z zasadami 5

## **Spis listingów**

1	Program sumujący dwie liczby w Pythonie . . . . .	4
2	Program sumujący dwie liczby w Javie . . . . .	4
3	Program sumujący dwie liczby w C++ . . . . .	4
4	Zapytanie SQL wybierające wszystkie rekordy z tabeli 'uzytkownicy' . .	5