Nazwa przedmiotu (projekt)
tu Tytuł Twojej Pracy

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

2 lutego 2026

Spis treści

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

```
def main():
    liczba1 = float(input("Podaj␣pierwszą␣liczbę:␣"))
    liczba2 = float(input("Podaj␣drugą␣liczbę:␣"))
    suma = liczba1 + liczba2
    print(f"Suma:␣{suma}")

if __name__ == "__main__":
    main()
```

Przykład kodu źródłowego w JAVA

Listing 2. Program sumujący dwie liczby w Javie

```
import java.util.Scanner;

public class Suma {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Podaj␣pierwszą␣liczbę:␣");
        float liczba1 = scanner.nextFloat();
        System.out.print("Podaj␣drugą␣liczbę:␣");
        float liczba2 = scanner.nextFloat();
        float suma = liczba1 + liczba2;
        System.out.println("Suma:␣" + suma);
        scanner.close();
    }
}
```

Przykład kodu źródłowego w CPP

Listing 3. Program sumujący dwie liczby w C++

```cpp
#include <iostream>
using namespace std;

int main() {
    float liczba1, liczba2;
    cout << "Podaj pierwszą liczbę: ";
    cin >> liczba1;
    cout << "Podaj drugą liczbę: ";
    cin >> liczba2;
    float suma = liczba1 + liczba2;
    cout << "Suma: " << suma << endl;
    return 0;
}
```

Przykład zapytania SQL

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

```sql
SELECT * FROM uzytkownicy;
```

Przykład tabeli

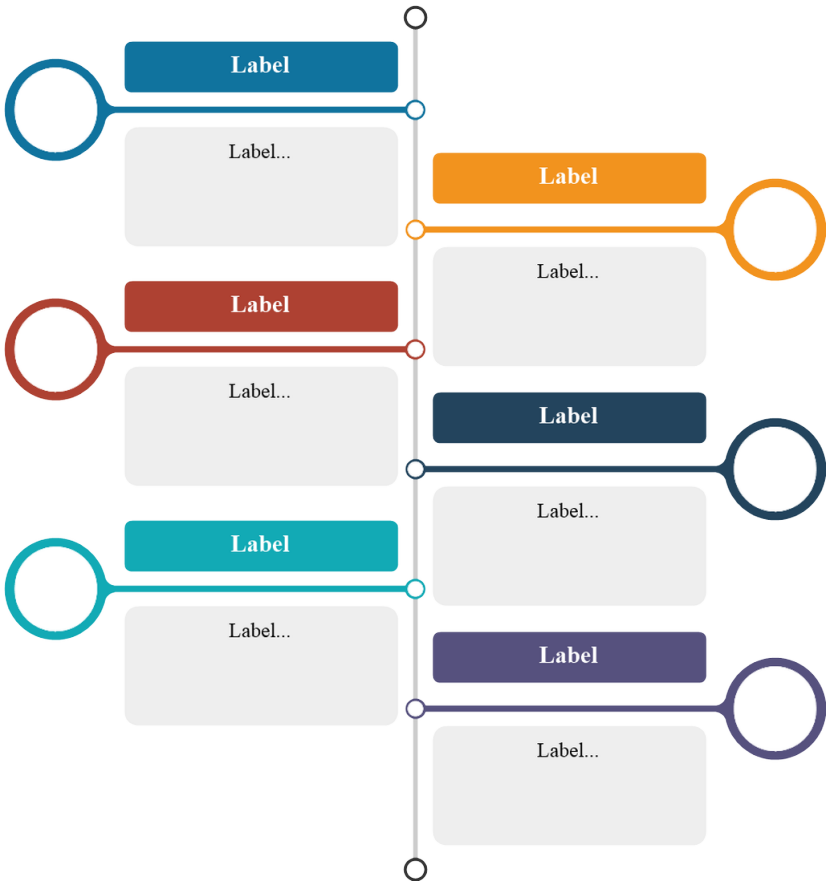
Tabela 1
Przykładowa tabela z trzema kolumnami sformatowana zgodnie z zasadami

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 LaTeX oraz automatyzującego proces kompilacji 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 kompilacji dokumentów,
- implementacja rozwiązania wraz z mechanizmami automatyzacji uruchomienia,
- przygotowanie scenariuszy weryfikacyjnych i ocena rezultatów.

Zgodnie z ujęciem przedstawionym w literaturze, nowoczesne systemy informatyczne 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 kompilację 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 związany z przygotowaniem dokumentów w LaTeX oraz trudnościami w zapewnieniu powtarzalnego procesu kompilacji 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 LaTeX, co skutkuje błędami kompilacji, różnicami w wyniku składu lub brakiem możliwości odtworzenia wyników na innym komputerze. Problemem jest również złożoność procesu kompilacji, wymagającego znajomości wielu parametrów i narzędzi pomocniczych. Projekt ma na celu uproszczenie tego procesu oraz zapewnienie powtarzalności wyników.

2.2. Cel i zakres pracy

Głównym celem projektu jest przygotowanie rozwiązania, które umożliwia kompilację dokumentów LaTeX w izolowanym środowisku kontenerowym oraz dostarcza prosty mechanizm uruchomienia procesu kompilacji. 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 kompilacji dokumentów LaTeX na różnych systemach?
- Jak zaprojektować proces kompilacji, 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 kompilacji dokumentów.

3.1. Wymagania funkcjonalne

Do kluczowych wymagań funkcjonalnych należą:

- możliwość kompilacji dokumentu LaTeX do postaci PDF w środowisku kontenerowym,
- automatyzacja wielokrotnego przebiegu kompilacji (np. dla spisu treści),
- czyszczenie plików tymczasowych po zakończeniu procesu,
- czytelna informacja o wyniku kompilacji dla użytkownika.

3.2. Wymagania niefunkcjonalne

Rozwiązanie powinno spełniać następujące wymagania jakościowe:

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

3.3. Architektura rozwiązania

Architektura przyjmuje układ warstwowy: warstwa dokumentu (źródła LaTeX), warstwa automatyzacji (skrypt uruchomieniowy) oraz warstwa środowiska uruchomieniowego (kontener Docker z zainstalowanym LaTeX). Taki podział umożliwia niezależne zarządzanie treścią dokumentu i procesem budowania.

3.4. Projekt procesu kompilacji

Proces kompilacji obejmuje:

- przygotowanie katalogu roboczego i usunięcie plików pomocniczych,
- uruchomienie kontenera i wykonanie kompilacji `latexmk` w trybie PDF,
- ponowny przebieg kompilacji 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 LaTeX oraz narzędziu `latexmk` uruchamianym w kontenerze Docker. Takie podejście eliminuje konieczność instalacji pełnego pakietu 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 LaTeX,
- uruchomienie skryptu automatyzującego kompilację,
- wygenerowanie pliku PDF w katalogu głównym projektu,
- opcjonalne czyszczenie plików tymczasowych po zakończeniu pracy.

4.4. Obsługa błędów

W przypadku problemów z kompilacją użytkownik otrzymuje czytelny komunikat. Najczęstszymi przyczynami są: brak uruchomionego Dockera, błędy składni w plikach 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 kompilacji.

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:

- kompilacja dokumentu bez błędów i wygenerowanie pliku PDF,
- ponowna kompilacja z uwzględnieniem spisu treści i elementów pomocniczych,
- obsługa sytuacji braku uruchomionego Dockera,
- reakcja systemu na błędny zapis w pliku LaTeX.

5.3. Wyniki testów

Wyniki potwierdziły poprawność działania procesu kompilacji oraz czytelność komunikatów z punktu widzenia użytkownika. Zidentyfikowano typowe źródła błędów związane z treścią LaTeX 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 kompilacji oraz konfigurację zależną od profilu użytkownika.

Podsumowanie

Tu znajduje 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 LaTeX Companion (informacje online)*. Ang. URL: `https://latex-project.org/` (dostęp 31.01.2026).

Spis rysunków

Spis tabel

Spis listingów