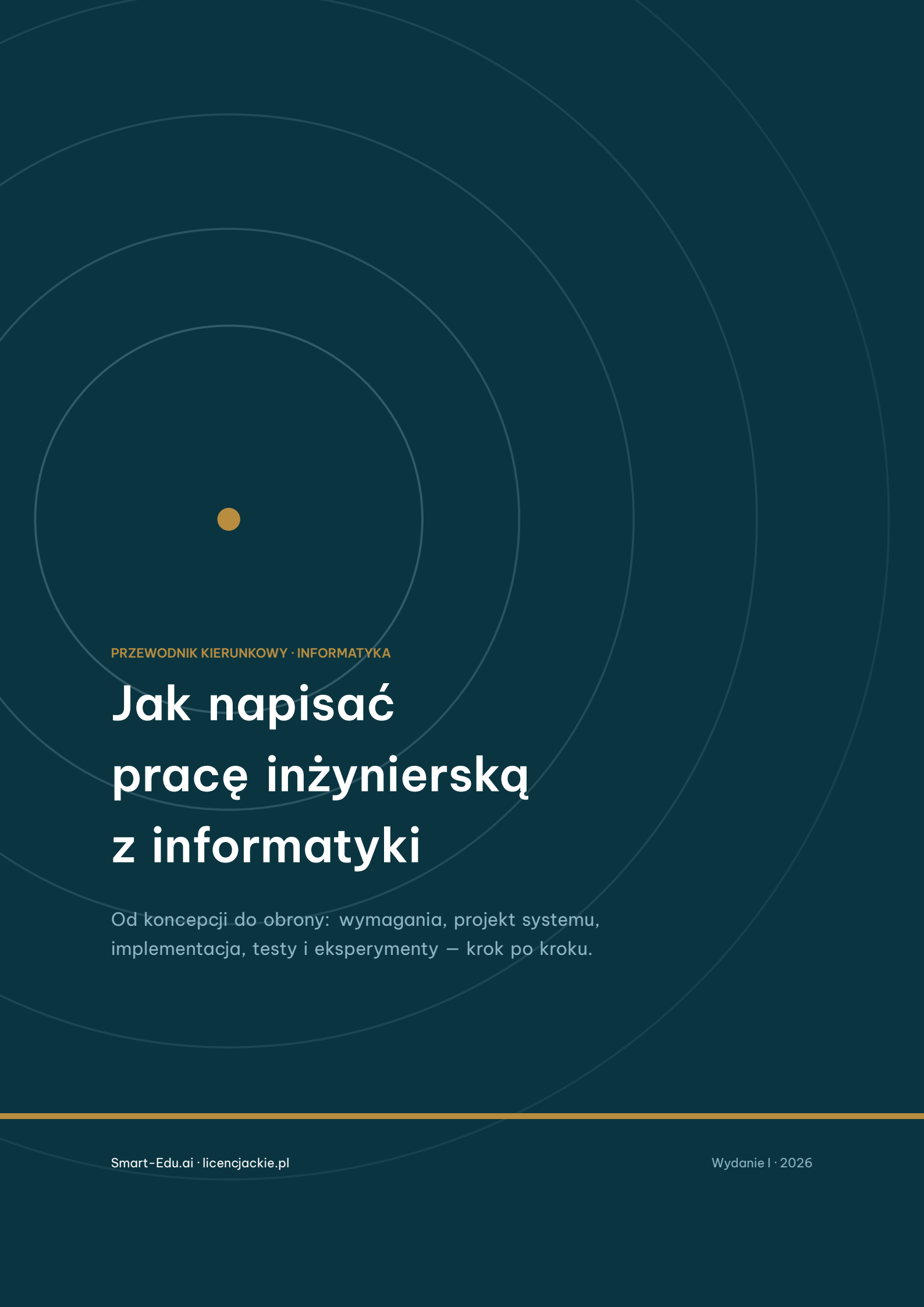




PRZEWODNIK KIERUNKOWY · INFORMATYKA

Jak napisać pracę inżynierską z informatyki

Od koncepcji do obrony: wymagania, projekt systemu,
implementacja, testy i eksperymenty – krok po kroku.

Spis treści

Jak korzystać z tego przewodnika	1
1 Czym praca z informatyki różni się od innych	2
1.1 Trzy typy prac inżynierskich z informatyki	2
1.2 Anatomia pracy projektowo-implementacyjnej	2
1.3 Kod a tekst: podział ról	3
1.4 Zakres: mniejszy system, głębsza analiza	3
2 Temat i problem: co czyni pracę inżynierską	4
2.1 Skąd brać problemy	4
2.2 Formuła tematu: system + problem	4
2.3 Pytania badawcze w pracy informatycznej	4
2.4 Test wykonalności	5
3 Wymagania i zakres systemu	6
3.1 Wymagania funkcjonalne: konkret i priorytet	6
3.2 Wymagania нефункционалне: tu mieszka „inżynierskość”	6
3.3 Analiza istniejących rozwiązań	6
3.4 Zakres: granice wypowiedziane wprost	7
4 Przegląd technologii: wybór z uzasadnieniem	8
4.1 Zasada: opisujesz wybory, nie technologie	8
4.2 Format decyzji: kryteria porównanie wybór	8
4.3 Źródła: co wolno cytować	8
4.4 Wersje i reprodukowalność	9
5 Projekt systemu: architektura i decyzje	10
5.1 Architektura: widok z lotu ptaka	10
5.2 Model danych i kontrakty	10
5.3 UML: tyle, ile pracuje	10
5.4 Projekt części badawczej	11
6 Implementacja: jak o niej pisać	12
6.1 Selekcja: kryterium nietrywialności	12
6.2 Listingi: krótkie i omówione	12
6.3 „Nie wyszło”: sekcja, która podnosi ocenę	12
6.4 Jakość kodu i repozytorium	13
7 Testowanie systemu	14
7.1 Weryfikacja funkcjonalna: domknięcie kontraktu	14
7.2 Scenariusze brzegowe i negatywne	14
7.3 Testy użyteczności: mały rygor zamiast deklaracji	14

7.4	Wynik testów jako materiał pracy	15
8	Eksperymenty i badania porównawcze	16
8.1	Projekt eksperymentu: co, na czym, czym mierzysz	16
8.2	Środowisko i procedura: reprodukowalność	16
8.3	Prezentacja wyników	16
8.4	Zagrożenia trafności: sekcja dojrzałości	17
9	Dokumentowanie: listingi, rysunki, konwencje	18
9.1	Listingi: konwencje formalne	18
9.2	Diagramy i rysunki	18
9.3	Wzory, jednostki, liczby	18
9.4	Aneksy i repozytorium: co gdzie	19
10	Struktura tekstu pracy	20
10.1	Szkielet pracy projektowo-implementacyjnej	20
10.2	Szkielet pracy badawczo-porównawczej	20
10.3	Narracja: od problemu do rozwiązania, z powrotami	20
10.4	Proporcje i objętość	21
11	Analiza wyników: od tabeli do wniosku	22
11.1	Trzy pytania do każdego wyniku	22
11.2	Porównania: uczciwość arytmetyczna	22
11.3	Wyniki jakościowe i użytecznościowe	22
11.4	Granice wniosków	23
12	Co czyni dobrą pracę inżynierską	24
12.1	Rdzeń inżynierski: proces, nie tylko produkt	24
12.2	Pytanie badawcze jako oś (gdy element badawczy jest)	24
12.3	Wkład pracy: policzalne artefakty	24
12.4	Rzetelność badawcza w wydaniu informatycznym	25
13	Zakończenie: wnioski, ograniczenia, rozwój	26
13.1	Struktura kanoniczna	26
13.2	Wnioski z liczbami, nie z przymiotnikami	26
13.3	Ograniczenia: rama, nie spowiedź	26
14	Redakcja i bibliografia	28
14.1	Styl: rzeczowo, konkretnie, w czasie właściwym	28
14.2	Bibliografia: porządek w trzech rodzinach	28
14.3	Poprawność techniczna tekstu	28
14.4	JSA i oryginalność w pracy z kodem	29
15	Finisz: składanie pracy i projektu	30
15.1	Kolejność ostatnich kroków	30
15.2	Cztery przejścia kontrolne	30
15.3	Harmonogram ostatnich czterech tygodni	31
16	Obrona: ostatnie czterdzieści minut	32
16.1	Autoprezentacja: dziesięć minut, dziesięć slajdów	32
16.2	Demo: pokazuj, co przeciwiczone	32
16.3	Kanon pytań na obronach z informatyki	32
16.4	Czego komisja słucha naprawdę	33
16.5	Po obronie	33

Jak korzystać z tego przewodnika

Ten przewodnik powstał dla jednej osoby: studentki lub studenta informatyki, który ma przed sobą pracę inżynierską i chce ją zrobić dobrze — bez błędzenia, bez poprawek „na trzeci raz” i bez paniki na finiszu. Nie znajdziesz tu ogólników o „pracach dyplomowych w ogóle”: każdy rozdział, każdy przykład i każda checklista dotyczą informatyki — jej projektów, kodu, eksperymentów i jej recenzentów.

Praca z informatyki jest inna niż na większości kierunków: zwykle coś **budujesz** (system, aplikację, model) albo coś **mierzysz** (porównanie algorytmów, technologii, architektur) — a tekst pracy dokumentuje koncepcję, decyzje i wyniki. To rodzi specyficzne pytania: ile kodu wolno wkleić, czym praca inżynierska różni się od projektu inżynierskiego, jak przeprowadzić eksperyment, którego nie zmiażdży pierwsze pytanie o metodykę. Ten przewodnik odpowiada na nie po kolei.

Po drodze spotkasz pięć rodzajów ramek. **Definicje** porządkują pojęcia warsztatu. **Przykłady z praktyki** pokazują, jak zasada wygląda w realnym tekście. **Pułapki** ostrzegają przed błędami z co drugiej pracy. **Checklisty** pozwalają odhaczyć etap przed oddaniem rozdziału promotorowi. A ramki **Okiem recenzenta** zdradzają, na co naprawdę patrzy osoba, która wystawi Ci ocenę.

Przewodnik czyta się liniowo, ale nie musisz: masz już temat i technologie — wejdź od razu w projekt systemu i implementację; planujesz badanie porównawcze — do rozdziału o eksperymentach; składasz tekst — do rozdziałów redakcyjnych. Spis treści prowadzi do konkretnych problemów, nie do ogólnych haseł.

Powodzenia. Dobrze poprowadzona praca inżynierska z informatyki to pierwszy poważny projekt, w którym musisz nie tylko zbudować coś działającego, lecz także udokumentować decyzje i obronić je pomiarami — czyli dokładnie to, co odróżnia inżyniera od seniora.

01

Czym praca z informatyki różni się od innych

Praca inżynierska z informatyki ma dwa artefakty zamiast jednego: **system lub badanie** oraz **tekst, który je dokumentuje**. Komisja ocenia oba — i większość rozczarowań bierze się z niedoceny drugiego: świetny projekt opisany na kolanie przegrywa z przeciętnym projektem opisanym rzetelnie. Ten rozdział ustawia proporcje i pokazuje typy prac.

1.1 Trzy typy prac inżynierskich z informatyki

Praca projektowo-implementacyjna — dominujący typ: projektujesz i budujesz system (aplikację, usługę, moduł, narzędzie), a praca dokumentuje wymagania, decyzje projektowe, implementację i testy. Uwaga kluczowa: rdzeniem pracy inżynierskiej jest **działający, rzetelnie udokumentowany i przetestowany system** — to on podlega ocenie. Element ponadstandardowy (porównanie podejść, pomiar właściwości) nie jest wymogiem, ale podnosi ocenę wyraźnie i przygotowuje grunt pod przyszłe magisterium.

Praca badawczo-porównawcza — mierzysz i porównujesz: wydajność algorytmów, frameworków, architektur, jakość modeli uczenia maszynowego. Rdzeniem jest poprawnie zaprojektowany eksperyment (rozdział ósmy) i uczciwa analiza wyników.

Praca analityczno-przeglądowa — systematyczny przegląd obszaru (metod, narzędzi, podatności) z własną syntezą, taksonomią lub oceną. Najradsza i najtrudniejsza do zrobienia dobrze — wymaga metodycznego przeglądu literatury, nie „opisania paru technologii”.

W praktyce najlepsze prace łączą pierwszy typ z drugim: budujesz system i mierzysz go albo porównujesz z alternatywą. Taka hybryda ma wbudowaną odpowiedź na pytanie „co w tej pracy jest inżynierskiego?”.

Co widzę w pierwszych pięciu minutach

*Spis treści: czy jest rozdział z testami/eksperymentem, czy praca kończy się na „zaimplementowano”. Rozdział o wymaganiach: czy są konkretne, czy marketingowe. Losowy listing: czy fragment kodu jest omówiony, czy wklejony na trzy strony bez słowa. I wnioski: czy zawierają liczby z badań, czy tylko „system działa poprawnie”. Prace z informatyki dzielą się dla mnie na te, które coś **zmierzyły**, i te, które tylko coś **napisały** — pierwsza kategoria zaczyna dwa stopnie wyżej.*

1.2 Anatomia pracy projektowo-implementacyjnej

Tabela 1.1: Szkielet pracy projektowo-implementacyjnej

Część	Zawartość	Typowa objętość
Wstęp	problem, cel, zakres, struktura	2–4 strony
Analiza	kontekst: istniejące rozwiązania, technologie, wymagania	12–18 stron

Część	Zawartość	Typowa objętość
Projekt	architektura, model danych, kluczowe decyzje z uzasadnieniem	12–18 stron
Implementacja	wybrane, nietrywialne aspekty realizacji	10–15 stron
Testy / badanie	metodyka, środowisko, wyniki, analiza	10–15 stron
Zakończenie	wnioski, ograniczenia, kierunki rozwoju	3–4 strony

Dwie proporcje ważne dla oceny: rozdział projektowy z **uzasadnieniami decyzji** (nie tylko diagramami) oraz rozdział testowo-badawczy **nie krótszy niż implementacyjny** — to on czyni pracę inżynierską.

1.3 Kod a tekst: podział ról

Tekst pracy nie jest wydrukiem repozytorium. Kod żyje w repozytorium (link lub płyta wg wymogów wydziału), a do pracy trafiają **fragmenty ilustrujące decyzje**: nietrywialny algorytm, ciekawy wzorzec, sedno rozwiązania problemu — zawsze omówione w tekście (co robi, dlaczego tak). Reguła kciuka: listing dłuższy niż strona to prawie zawsze błąd selekcji; szczegóły w rozdziale dziewiątym.

⚠ Praca-dokumentacja użytkownika

Rozdziały „Instalacja”, „Logowanie do systemu”, „Opis każdego okna z zrzutami ekranu” — praca zamienia się w instrukcję obsługi, a recenzent nie znajduje ani jednej decyzji inżynierskiej. Zrzuty ekranu mają w pracy rolę służebną (2–4 sztuki ilustrujące kluczowe funkcje), a treścią rozdziałów są **problemy i rozwiązania**: dlaczego taka architektura, jak rozwiązano X, co się nie udało i czemu. Manual, jeśli wymagany, idzie do załącznika.

1.4 Zakres: mniejszy system, głębsza analiza

Najczęstszy błąd planistyczny: system zakrojony jak startup (aplikacja + panel + API + aplikacja mobilna), z którego w połowie semestru zostaje wydumuszka. Zasada inżynierska jest odwrotna: **mniejszy zakres funkcjonalny, głębszy wymiar analityczny** — jeden dobrze zbadany problem (np. wydajność synchronizacji offline) wart jest więcej niż dziesięć CRUD-owych ekranów. Definicję zakresu i granic (co świadomie poza) opisuje rozdział trzeci.

☑ Zanim przejdziesz dalej

- ☐ Wiesz, jaki typ pracy piszesz (projektowa / porównawcza / przeglądowa / hybryda)
- ☐ Wiesz, co jest elementem ponadstandardowym Twojej pracy
- ☐ Plan zakłada rozdział testowo-badawczy z pomiarami
- ☐ Rozumiesz podział ról: kod w repo, w tekście decyzje i fragmenty
- ☐ Zakres: mniej funkcji, więcej głębi

W kolejnych rozdziałach przechodzimy trasę w kolejności projektu: temat, wymagania i zakres, przegląd technologii, projekt systemu, implementacja, testy i eksperymenty, dokumentowanie, struktura tekstu, analiza wyników — aż po finisz i obronę.