МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет
імені Івана Пулюя

Кафедра програмної інженерії

Методичні вказівки для самостійної роботи

з дисципліни

«Основи програмної інженерії (SE201)»

CEH ID: **6822**

для здобувачів першого (бакалаврського) рівня вищої освіти

ОПП «Інженерія програмного забезпечення»
всіх форм навчання

Тернопіль — 2025

УДК 004.4(075.8)
ББК 32.973.202я73
Б83

Укладач:
В.М. Бревус, PhD

Рецензент:
В.В. Яцишин, к.т.н., доцент
завідувач кафедри систем штучного інтелекту та аналізу даних ТНТУ ім. І. Пулюя

Розглянуто й затверджено на засіданні кафедри програмної інженерії.
Протокол № 1 від 01.09.2025.

Розглянуто й затверджено на засіданні методичної комісії факультету комп'ютерно-інформаційних систем та програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя.
Протокол № 1 від 01.09.2025.

Б83 Методичні вказівки для самостійної роботи з дисципліни «Основи програмної інженерії (SE201)» для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Інженерія програмного забезпечення» усіх форм навчання / Укладач: Бревус В.М. – Тернопіль: ТНТУ імені Івана Пулюя, 2025 – 26 с.

Відповідальний за випуск: М.Р. Петрик, докт. фіз.-мат. наук, професор

АНОТАЦІЯ

Ці методичні вказівки для самостійної роботи з дисципліни «Основи програмної інженерії (SE201)» формують у здобувачів вищої освіти фахові компетентності та дослідницькі здібності, необхідні для розв'язання задач проєктування та розробки програмного забезпечення.

Розглянуто життєвий цикл програмного забезпечення (SDLC), моделі процесів розробки (Waterfall, Agile, Scrum, Kanban), інженерію вимог, архітектуру та проєктування ПЗ. Висвітлено питання конструювання, забезпечення якості, тестування, управління конфігурацією та супроводу програмних систем. Окрему увагу приділено управлінню проєктами, професійній етиці та стандартам (SWEBOK, ISO/IEC), а також сучасним тенденціям, таким як DevOps, хмарні технології та використання штучного інтелекту в розробці. Матеріал орієнтований на здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 121 «Інженерія програмного забезпечення» та спрямований на формування професійних компетентностей у сфері промислової розробки програмних продуктів.

Ключові слова: програмна інженерія, SDLC, Agile, вимоги, архітектура ПЗ, тестування, якість ПЗ, управління проєктами, DevOps, SWEBOK.

ABSTRACT

These methodological guidelines for independent study of the discipline “Fundamentals of Software Engineering (SE201)” develop professional competencies in students of higher education necessary for solving tasks of designing, developing, and testing software systems.

The guidelines systematize fundamental concepts, principles, and methods of software engineering. They cover the software development life cycle (SDLC), development process models (Waterfall, Agile, Scrum, Kanban), requirements engineering, software architecture and design. Issues of construction, quality assurance, testing, configuration management, and software maintenance are highlighted. Special attention is paid to project management, professional ethics and standards (SWEBOK, ISO/IEC), as well as modern trends such as DevOps, cloud technologies, and the use of artificial intelligence in development.

The document is intended for applicants for the first (bachelor’s) level of higher education in the specialty 121 “Software Engineering” educational and scientific program and is directed at forming professional competencies in the field of industrial software product development. It contains detailed tasks, theoretical background, methodological recommendations, and requirements for presenting results.

Keywords: software engineering, SDLC, Agile, requirements, software architecture, testing, software quality, project management, DevOps, SWEBOK.

Зміст

Вступ **5**

1. Тематичний план навчальної дисципліни **8**

2. Загальні рекомендації до організації самостійної роботи з дисципліни **13**

4. Система поточного й підсумкового контролю знань здобувачів **17**

5. Критерії оцінювання результатів навчання здобувачів **19**

6. Перелік контрольних запитань з дисципліни **22**

7. Рекомендована література **24**

Вступ

Дисципліна «Основи програмної інженерії (SE201)» присвячена розвитку систематизованих знань про програмну інженерію як один із ключових напрямів діяльності під час розроблення програмних проєктів. Ця дисципліна ознайомлює здобувачів з базовими методами й засобами програмної інженерії та сприяє формуванню навичок ефективного аналізу, проєктування, конструювання й тестування програмних систем.

Призначення методичних вказівок

Ці методичні вказівки розраховані на здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Інженерія програмного забезпечення» та визначають порядок організації самостійної роботи при виконанні практичних завдань. Вказівки містять детальні завдання, теоретичне підґрунтя, методичні рекомендації та вимоги до оформлення результатів роботи. Матеріали курсу доступні в СЕН ТНТУ ім. І. Пулюя ID 6822.

Усі роботи повинні бути виконані з дотриманням принципів академічної доброчесності, відповідно до Положення про академічну доброчесність учасників освітнього процесу Тернопільського національного технічного університету імені Івана Пулюя.

Мета вивчення навчальної дисципліни «Основи програмної інженерії» (SE201)

Формування у здобувачів систематизованих знань про програмну інженерію як один із ключових напрямів діяльності під час розроблення програмних проєктів, ознайомлення з базовими методами й засобами цієї галузі та сприяння формуванню навичок ефективного аналізу, проєктування, конструювання й тестування програмних систем.

Завдання навчальної дисципліни

За результатами вивчення дисципліни здобувач повинен продемонструвати такі результати навчання:

Завдання навчальної дисципліни

За результатами вивчення дисципліни здобувач повинен продемонструвати такі результати навчання:

Знати:

- етапи життєвого циклу програмного продукту;
- методи проєктування програмних систем;
- методи визначення вимог до програмного забезпечення;

- технології розробки програмного забезпечення;
- інструментальні засоби програмної інженерії;
- кодекс професійної етики.

Вміти:

- аналізувати та будувати моделі предметної області;
- проектувати програмні системи, використовуючи різні підходи;
- створювати артефакти програмного забезпечення з використанням систем контролю версій;
- аналізувати результати побудови та використання програмного забезпечення;
- використовувати інструменти ШІ для ефективного створення програмного забезпечення.

Вивчення навчальної дисципліни передбачає формування та розвиток у здобувачів компетентностей:

- **Інтегральна компетентність.** Здатність розв'язувати складні спеціалізовані завдання або практичні проблеми інженерії програмного забезпечення, що характеризуються комплексністю та невизначеністю умов, із застосуванням теорій та методів інформаційних технологій.
- **К01.** Здатність до абстрактного мислення, аналізу та синтезу.
- **К02.** Здатність застосовувати знання у практичних ситуаціях.
- **К03.** Здатність спілкуватися державною мовою як усно, так і письмово.
- **К04.** Здатність спілкуватися іноземною мовою як усно, так і письмово.
- **К05.** Здатність вчитися і оволодівати сучасними знаннями.
- **К06.** Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
- **К07.** Здатність працювати в команді.
- **К08.** Здатність діяти на основі етичних міркувань.
- **К17.** Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу
- **К22.** Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом усього життя.

- **К23.** Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення.

Програмні результати:

- **ПР01.** Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.
- **ПР02.** Знати кодекс професійної етики, розуміти соціальну значимість та культурні аспекти інженерії програмного забезпечення і дотримуватись їх в професійній діяльності.
- **ПР03.** Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення.
- **ПР04.** Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення.
- **ПР07.** Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.
- **ПР16.** Мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації.
- **ПР23.** Вміти документувати та презентувати результати розробки програмного забезпечення.

Завданням лекційних занять є ознайомлення здобувачів з загальними принципами життєвого циклу програмного забезпечення, методологіями, ключовими процесами та артефактами.

Завдання лабораторних занять полягає у практичному засвоєнні здобувачами навичок по роботі з розподіленими системами контролю версій індивідуально та в команді, досліджені архітектур та методологій розробки програмного забезпечення на прикладі програмних застосунків з відкритим кодом.

1. ТЕМАТИЧНИЙ ПЛАН НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Дисципліна: Основи програмної інженерії (SE201).
Рівень: перший (бакалаврський).
Спеціальність: F2 Інженерія програмного забезпечення.
Форма контролю: залік.
Обсяг: 3 ECTS (90 годин): 36 год. аудиторних (18 лекцій, 18 лабораторних), 54 год. самостійна робота.

При вивченні дисципліни здобувач ознайомлюється з систематизованими знаннями про програмну інженерію як один із ключових напрямів діяльності під час розроблення програмних проєктів. Тематичний план складається з двох модулів, що охоплюють процеси та моделі життєвого циклу ПЗ, методи проєктування та архітектури програмних систем, роботу з вимогами, тестування та управління проєктами, а також професійні стандарти та етику програмної інженерії. Практичні роботи базуються на розробці чат-платформи з використанням систем контролю версій та об'єктно-орієнтованого дизайну.

1.1 Структура змістових модулів

Модуль 1 формує базу: вступ до програмної інженерії, процеси та моделі життєвого циклу ПЗ, роботу з вимогами, основи архітектури та дизайну ПЗ, конструювання та реалізацію ПЗ. Модуль 2 поглиблює: тестування програмного забезпечення, управління проєктами, професійні стандарти та етику, кар'єру в програмній інженерії та майбутнє галузі.

1.2 Тематичний план (лекції)

№	Тема заняття та короткий зміст	ДФН	ЗФН
	Модуль 1		
1	**Лекція 1. Вступ до програмної інженерії.** Поняття програмної інженерії, її місце в життєвому циклі розробки ПЗ. Історичний огляд та сучасні тенденції.	2	0.5
2	**Лекція 2. Процеси та моделі життєвого циклу ПЗ.** Огляд основних моделей (водоспадна, ітеративна, спіральна) та гнучких методологій (Agile, Scrum, Kanban).	2	0.5
3	**Лекція 3. Робота з вимогами.** Збір, аналіз, специфікація та валідація вимог. Створення програмної документації.	2	0.5
4	**Лекція 4. Основи архітектури та дизайну ПЗ.** Архітектурні стилі, шаблони проєктування (Design Patterns). Принципи SOLID.	2	0.5
5	**Лекція 5. Конструювання та реалізація ПЗ.** Інструментальні засоби, мови програмування, якість коду, рефакторинг.	2	
	Модуль 2		
6	**Лекція 6. Тестування програмного забезпечення.** Види та рівні тестування. Стратегії та методи тестування. Автоматизація тестування.	2	
7	**Лекція 7. Управління проєктами в програмній інженерії.** Планування, оцінка трудомісткості, управління ризиками та командна робота.	2	
8	**Лекція 8. Професійні стандарти та етика.** Огляд стандартів (IEEE, ISO). Кодекс професійної етики інженера-програміста. Соціальна відповідальність.	2	1
9	**Лекція 9. Кар'єра в програмній інженерії та майбутнє галузі.** Ролі в команді, необхідні навички, важливість навчання протягом життя. Вплив AI на розробку ПЗ.	2	1
	Усього годин	**18**	**4**

1.3 Лабораторні заняття (прикладний наскрізний проєкт)

Лабораторні роботи організовані навколо наскрізного проєкту **«Розробка чат-платформи MVP»**, що базується на комплексному застосуванні методів програмної інженерії від аналізу вимог до тестування, з використанням систем контролю версій та об'єктно-орієнтованого дизайну.

№	Тема заняття та короткий зміст	ДФН	ЗФН
1	**Лабораторна робота №1. Налаштування професійного середовища розробки.** Робота з системою контролю версій Git та GitHub для створення командного проєкту.	4	1
2	**Лабораторна робота №2. Аналіз вимог та проєктування архітектури чат-платформи.** Створення специфікації вимог (User Stories) та проєктування за допомогою UML-діаграм (Use Case, класів).	6	1
3	**Лабораторна робота №3. Конструювання та тестування ключового модуля.** Реалізація основної функціональності та покриття його unit-тестами.	4	1
4	**Лабораторна робота №4. Аналіз професійних стандартів та етики.** Створення професійного профілю розробника та аналіз ринку праці.	4	1
	Усього годин	**18**	**4**

1.4 Самостійна робота

Рекомендовані напрями самостійного опрацювання узгоджені з логікою поглиблення матеріалу та формування компетентностей.

№	Найменування робіт	ДФН	ЗФН
1	Розподілені системи контролю версій	3	10
2	Підходи до планування та організації проекту розробки ПЗ	2	10
3	Підходи до розробки ПЗ з використанням AI	3	10
4	Актуалізація профілю у GitHub та Linkedin	2	10
5	Опрацювання теоретичного матеріалу	10	15
6	Підготовка до лабораторних занять	9	2
7	Підготовка до заліку	25	25
	Усього годин	**54**	**82**

1.5 Очікувані результати опанування тематичного плану

Після завершення дисципліни здобувач повинен:

- розуміти етапи життєвого циклу програмного продукту та основні моделі розробки ПЗ;
- аналізувати та будувати моделі предметної області з використанням UML та інших методів;
- проектувати програмні системи, використовуючи різні архітектурні стилі та шаблони проєктування;
- створювати артефакти програмного забезпечення з використанням систем контролю версій;
- застосовувати методи тестування та забезпечувати якість коду;
- управляти програмними проєктами та командною роботою;
- дотримуватись професійних стандартів та етики програмної інженерії;
- використовувати інструменти ШІ для ефективного створення ПЗ;
- аналізувати результати розробки та використання програмного забезпечення.

1.6 Основні компоненти наскрізного проєкту

Артефакти розробки: специфікація вимог, UML-діаграми, архітектурні рішення, код з unit-тестами, документація.
Проміжні результати: Git-репозиторій з комітами, User Stories, Use Case діаграми, діаграми класів, реалізована функціональність, професійний профіль розробника.
Методичні акценти: колективна розробка, системне мислення, якість артефактів, дотримання стандартів, професійна комунікація.

1.7 Короткі критерії оцінювання

- Повнота виконання лабораторних робіт (артефакти + документація).
- Якість специфікації вимог та архітектурних рішень.
- Коректність реалізації функціональності та покриття unit-тестами.
- Комунікація в команді та використання систем контролю версій.
- Розуміння професійних стандартів та етики програмної інженерії.
- Академічна доброчесність (оригінальність роботи, коректні посилання).

1.8 Рекомендовані інструменти

Git та GitHub (контроль версій), Visual Studio Code (IDE), Python або Java (для реалізації), PlantUML або draw.io (для діаграм), JUnit або PyTest (для тестування), GitHub Copilot (для допомоги AI), уціль других інструментів — за рішенням кафедри.

1.9 Базові та допоміжні джерела

Перелік літератури й ресурсів наведено у відповідному розділі видання (див. повну робочу програму SE201 2025). Основні джерела: Guide to the Software Engineering Body of Knowledge (SWEBOK) v4.0, Sommerville I. «Software Engineering» (10-е видання), Martin R.C. «Clean Code», Pro Git book (українська версія).

1.10 Примітки щодо адаптації плану

Тематичний план може коригуватися (оновлення кейсів розробки ПЗ, модернізація інструментарію, інтеграція AI-підтримки) за рішенням кафедри з фіксацією змін у протоколах засідань кафедри програмної інженерії.

2. ЗАГАЛЬНІ РЕКОМЕНДАЦІЇ ДО ОРГАНІЗАЦІЇ САМОСТІЙНОЇ РОБОТИ З ДИСЦИПЛІНИ

Обов'язковим елементом успішного засвоєння навчального матеріалу дисципліни «Основи програмної інженерії (SE201)» є самостійна робота здобувачів з вітчизняною і зарубіжною літературою.

Самостійна робота є основним засобом оволодіння навчальним матеріалом у час, вільний від нормованих навчальних занять, тобто лекційних і лабораторних. Особливо важливою є самостійна робота з дослідження сучасних методів програмної інженерії, розроблення програмного забезпечення та професійних стандартів.

2.1 Основні види самостійної роботи

На які повинні звертати увагу здобувачи:

- вивчення лекційного матеріалу з основ програмної інженерії та життєвого циклу програмного забезпечення;
- опрацювання та вивчення рекомендованої наукової літератури та актуальних статей з програмної інженерії;
- підготовка до лабораторних занять з Git, GitHub та систем контролю версій;
- виконання завдань з аналізу вимог, проєктування архітектури та тестування;
- дослідження методологій розробки (Waterfall, Agile, Scrum) та архітектурних патернів;
- аналіз професійних стандартів та етики в програмній інженерії;
- підготовка до поточного та підсумкового контролю (залік).

2.2 Опрацювання лекційного матеріалу

У системі різних форм навчально-виховної роботи особливе місце належить лекції, де викладач надає здобувачу основну інформацію, навчає розмірковувати, аналізувати, допомагає опанувати ключові знання, а також спрямовує самостійну роботу здобувача.

Зв'язок лекції і самостійної роботи здобувача розглядається в таких напрямах:

- лекція як головна початкова ланка, що визначає зміст і обсяг самостійної роботи здобувача;
- методичні прийоми читання лекцій, що активізують самостійну роботу здобувачів;
- самостійна робота, яка сприяє поглибленому засвоєнню теми на базі прослуханої лекції.

Перший етап самостійної роботи починається з процесу слухання і записування лекції. Правильно складений конспект лекції — найефективніший засіб стимулювання подальшої самостійної роботи здобувачів. Здобувач повинен чітко усвідомити, що конспект — це короткий тезовий запис головних положень навчального матеріалу. Складання і вивчення конспекту — перший етап самостійної роботи здобувача над вивченням теми чи розділу. Конспект допомагає в раціональній підготовці до практичних занять, заліку, у визначенні напряму і обсягу подальшої роботи з літературними джерелами.

Під час підготовки до лекції здобувач повинен опрацювати матеріал попередної лекції з використанням підручників та інших джерел літератури. На лекціях висвітлюють тільки основні теоретичні положення та найбільш актуальні проблеми, тому більшість питань виноситься на самостійне опрацювання.

2.3 Підготовка до лабораторних занять

Підготовка до лабораторних занять розпочинається з опрацювання лекційного та методичного матеріалу до заданої роботи. Здобувач повинен самостійно ознайомитися з відповідним розділом робочої програми SE201, ознайомитися з методичними вказівками, підготувати відповіді на контрольні запитання, які подані в програмі та методичних матеріалах у певній послідовності згідно з логікою засвоєння навчального матеріалу.

Лабораторні роботи збагачують і закріплюють теоретичні знання здобувачів щодо методів програмної інженерії, розвиваючи їхню творчу активність, допомагають у набутті практичних навичок роботи з системами контролю версій (Git, GitHub), аналізу вимог, проєктування архітектури та тестування.

У процесі підготовки до лабораторних робіт самостійна робота здобувачів є обов'язковою частиною навчальної роботи, без якої успішне і якісне засвоєння навчального матеріалу неможливе. Це свідчить про необхідність керування самостійною роботою здобувачів з боку викладача завдяки проведенню цілеспрямованих організаційних і контрольних заходів.

Викладач у вступній лекції рекомендує здобувачам основну і додаткову літературу з робочої програми SE201, а також методичні рекомендації до самостійної роботи та до організації лабораторних робіт з дисципліни. У методичних вказівках з кожної лабораторної роботи наведено перелік питань для теоретичної підготовки та практичних завдань.

У разі, коли здобувач не може самостійно розібратися в якомусь питанні, він може отримати консультацію у викладача (згідно з графіком проведення консультацій викладачами кафедри). Добре організовані консультації дозволяють спрямувати самостійну роботу в потрібному напрямі, зробити раціональною і підвищити її ефективність.

2.4 Структурована програма самостійної роботи

№	Найменування розділів самостійної роботи	ДФН	ЗФН
1	Розподілені системи контролю версій (Git, GitHub)	3	10
2	Підходи до планування та організації проекту розробки ПЗ (Agile, Scrum)	2	10
3	Підходи до розробки ПЗ з використанням AI (GitHub Copilot)	3	10
4	Актуалізація профілю у GitHub та LinkedIn	2	10
5	Опрацювання теоретичного матеріалу з рекомендованої літератури	20	20
6	Підготовка до лабораторних занять	9	2
7	Підготовка до заліку	15	20
	Разом	**54**	**82**

2.5 Рекомендована послідовність роботи

Для кожного модуля:

1. **Етап 1 — Підготовка до лекцій.** Прочитайте стислий огляд теми з рекомендованої літератури перед лекцією, щоб отримати загальне розуміння матеріалу програмної інженерії.
2. **Етап 2 — Слухання лекції та конспектування.** Активно слухайте лекцію, записуйте ключові концепції, приклади та висновки викладача про методи розробки ПЗ.
3. **Етап 3 — Опрацювання конспекту.** Відразу після лекції переглянете конспект, уточніть незрозумілі місця, доповніть конспект інформацією з посилань та методичних матеріалів.
4. **Етап 4 — Поглиблене вивчення теми.** Прочитайте відповідні розділи підручників та матеріалів з рекомендованого списку літератури, особливу увагу приділіть актуальним статтям з програмної інженерії та професійних стандартів.
5. **Етап 5 — Підготовка до лабораторного заняття.** Встановіть необхідне програмне забезпечення (Git, GitHub, IDE), ознайомтеся з документацією, підготуйте відповіді на контрольні запитання.
6. **Етап 6 — Виконання лабораторної роботи.** Активно беріть участь у лабораторному занятті, виконуйте завдання з аналізу вимог, проєктування та тестування, задавайте питання.

7. **Етап 7 — Закріплення матеріалу.** Експериментуйте з різними підходами до розв'язання задач, досліджуйте додаткові інструменти та методи для розширення розуміння теми програмної інженерії.

8. **Етап 8 — Підготовка до модульного контролю.** Повторіть весь матеріал модуля, розв'яжіть тестові завдання, готуйтеся до залік.

2.6 Поради щодо ефективної самостійної роботи

- **Планування часу.** Розподіліть самостійну роботу рівномірно протягом семестру, уникайте «штурму» перед залік.

- **Організація робочого простору.** Створіть умови для концентрації уваги: чистий стіл, мінімум відволікань, необхідні матеріали та доступ до комп'ютера.

- **Активне читання.** Не просто читайте текст, але й робіть нотатки, складайте схеми, створюйте опорні конспекти та діаграми.

- **Практичне застосування.** Намагайтеся застосовувати теоретичні знання до розв'язання практичних завдань, експериментуйте з інструментами програмної інженерії, створюйте власні проєкти.

- **Взаємодія з однолітками.** Обговорюйте складні питання програмної інженерії з одногрупниками, створюйте спільні проєкти з Git та GitHub, діліться досвідом розробки та вирішення проблем.

- **Консультації з викладачем.** Не стримуйтеся звертатися за допомогою, коли виникають труднощі в розумінні матеріалу чи виконанні завдань.

- **Регулярне повторення.** Повторюйте раніше вивчений матеріал, щоб закріпити знання та запобігти їх забуванню.

- **Використання ресурсів.** Користуйтеся рекомендованою літературою, електронним курсом на СЕН, офіційною документацією та ресурсами GitHub для поглиблення знань.

4. СИСТЕМА ПОТОЧНОГО Й ПІДСУМКОВОГО КОНТРОЛЮ ЗНАНЬ ЗДОБУВАЧІВ

4.1 Форми контролю

Оцінювання знань, вмінь і навичок здобувачів включає ті види занять, які згідно з програмою навчальної дисципліни «Основи програмної інженерії (SE201)» передбачають лекційні, лабораторні роботи, самостійну роботу.

Перевірку і оцінювання знань здобувачів проводять в наступних формах:

- оцінювання роботи і знань здобувачів під час лабораторних занять;
- оцінювання виконання і захист лабораторних робіт;
- складання проміжного контролю знань за змістовими модулями (модульний контроль);
- здача залікового тестування.

4.2 Модульний контроль

Для кожного змістовного модуля передбачено певну форму поточного контролю. Результати поточного контролю автоматично, без участі здобувача, зараховуються при модульному контролі.

Максимальна оцінка при I модульному контролі — 40 балів:

- Теоретичний курс (тестування): 15 балів
- Лабораторна робота: 20 балів
- Самостійна робота: 5 балів

Максимальна оцінка при II модульному контролі — 35 балів:

- Теоретичний курс (тестування): 15 балів
- Лабораторна робота: 15 балів
- Самостійна робота: 5 балів

4.3 Підсумковий контроль

Підсумковий контроль здійснюється у формі залік з максимальною оцінкою 25 балів.

Максимальна оцінка навчальної дисципліни — 100 балів.

Розподіл балів між модулями:

- Модуль 1: 40 балів
- Модуль 2: 35 балів
- Залік: 25 балів

4.4 Критерії оцінювання за компонентами

Теоретичний курс (тестування): Оцінюється глибина розуміння теоретичного матеріалу, знання основних концепцій програмної інженерії, методів аналізу вимог, проєктування та тестування.

Лабораторні роботи: Оцінюється якість виконання практичних завдань з Git, GitHub, аналізу вимог, проєктування архітектури та тестування, прикладне застосування теоретичних знань, коректність результатів та обґрунтованість висновків.

5. КРИТЕРІЇ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ ЗДОБУВАЧІВ

5.1 Шкала оцінювання

Сума балів за навчальну діяльність	Оцінка ECTS	Оцінка за національною шкалою
90 – 100	A	Відмінно
82–89	B	Добре
74–81	C	Добре
64–73	D	Задовільно
60–63	E	Задовільно
35–59	FX	Незадовільно з можливістю повторного складання
0–34	F	Незадовільно з обов'язковим повторним вивченням дисципліни

5.2 Таблиця розподілу балів

Модуль 1		**Модуль 2**	
Аудиторна та самостійна робота		Аудиторна та самостійна робота	
Теоретичний курс (тестування)	Лабораторна робота	Теоретичний курс (тестування)	Лабораторна робота
15	**20**	**15**	**15**
Підсумковий контроль (залік): 25			
Разом з дисципліни: 100			

5.3 Деталізація оцінювання за модулями

Модуль 1			**Модуль 2**		
Тема	Вид робіт	К-ть балів	Тема	Вид робіт	К-ть балів
Тема 1-3	ЛР№1	10	Тема 6, 7	ЛР№3	10
Тема 4, 5	ЛР№2	10	Тема 8, 9	ЛР№4	5
Тестування: 15 балів			Тестування: 15 балів		

5.4 Критерії виставлення оцінок

Оцінка «A» (90–100 балів):

- Глибоке розуміння фундаментальних принципів програмної інженерії та життєвого циклу розробки ПЗ
- Безпомилково виконані всі лабораторні роботи з Git, GitHub, аналізу вимог, проєктування та тестування
- Логічне та обґрунтоване застосування методів програмної інженерії до розв'язання складних задач
- Демонстрація аналітичних здібностей та творчого підходу до проєктування та розробки ПЗ

Оцінка «B» (82–89 балів):

- Добре розуміння методологій розробки ПЗ та принципів проєктування систем
- Якісне виконання лабораторних робіт з незначними помилками
- Коректне застосування методів аналізу вимог та проєктування архітектури
- Достатня глибина та точність у розробці програмних систем та артефактів

Оцінка «C» (74–81 балів):

- Задовільне розуміння основних тем курсу програмної інженерії
- Виконання лабораторних робіт з деякими помилками
- Розв'язання стандартних завдань з аналізу та проєктування без істотних проблем
- Задовільне обґрунтування виконаних робіт

Оцінка «D–E» (60–73 балів):

- Мінімальне розуміння основних концепцій програмної інженерії
- Лабораторні роботи виконані з значними прогалинами
- Розв'язання простих завдань з помилками
- Слабке обґрунтування результатів

Оцінка «FX» (35–59 балів):

- Неповне розуміння теоретичного матеріалу з програмної інженерії
- Суттєві недоліки у виконанні лабораторних робіт

- Помилки при розв’язанні базових завдань з аналізу та проєктування
- Можливість повторного складання або доопрацювання

Оцінка «F» (0–34 балів):

- Неадекватне розуміння основних концепцій програмної інженерії
- Неякісне виконання лабораторних робіт
- Неспроможність розв’язати навіть прості завдання з проєктування ПЗ
- Вимагає обов’язкового повторного вивчення дисципліни

6. ПЕРЕЛІК КОНТРОЛЬНИХ ЗАПИТАНЬ З ДИСЦИПЛІНИ

6.1 Запитання з Модулю 1 (Теми 1-5)

1. Дайте визначення програмної інженерії та поясніть її місце в життєвому циклі розробки ПЗ.
2. Назвіть і опишіть основні моделі життєвого циклу ПЗ: водоспадна, ітеративна, спіральна.
3. Які гнучкі методології розробки ви знаєте? Опишіть Agile, Scrum та Kanban.
4. Які основні етапи збору та аналізу вимог? Поясніть різницю між функціональними та нефункціональними вимогами.
5. Що таке User Stories і як їх використовувати при аналізі вимог?
6. Назвіть основні архітектурні стилі та поясніть їх особливості (MVC, REST, мікросервіси тощо).
7. Що таке Design Patterns? Назвіть кілька прикладів (Singleton, Factory, Observer тощо).
8. Сформулюйте принципи SOLID та поясніть кожен з них.
9. Як здійснюється проєктування архітектури програмної системи?
10. Що таке UML-діаграми? Назвіть їх типи та поясніть їх застосування.

6.2 Запитання з Модулю 2 (Теми 6-9)

11. Які основні рівні та типи тестування ви знаєте?
12. Що таке Unit тестування і як його реалізувати?
13. Поясніть різницю між функціональним та нефункціональним тестуванням.
14. Як здійснюється автоматизація тестування? Назвіть інструменти.
15. Що таке тест-кейси та як їх розробляти?
16. Що входить до управління проєктом розробки ПЗ?
17. Як проводити планування та оцінку трудомісткості проєкту?
18. Назвіть основні ризики при розробці ПЗ та методи їх управління.
19. Що таке командна робота у розробці ПЗ? Які ролі в команді?
20. Назвіть професійні стандарти в програмній інженерії (IEEE, ISO).
21. Що таке кодекс професійної етики інженера-програміста?

22. Поясніть соціальну відповідальність у розробці ПЗ.
23. Які ролі та навички необхідні для кар'єри в програмній інженерії?
24. Як важливість навчання протягом життя в професійному розвитку?
25. Який вплив AI на розробку ПЗ? Назвіть приклади (GitHub Copilot, код-генерація).

6.3 Спеціальні питання для лабораторних робіт

26. Як налаштувати професійне середовище розробки з Git та GitHub?
27. Назвіть основні команди Git для роботи з репозиторіями.
28. Що таке гілки (branches) у Git і як ними користуватися?
29. Як здійснювати злиття гілок (merge) та розв'язувати конфлікти?
30. Як аналізувати вимоги та створювати User Stories для чат-платформи?
31. Як проєктувати архітектуру за допомогою UML-діаграм (Use Case, Class)?
32. Як реалізувати основну функціональність (напр., відправка повідомлень)?
33. Як писати Unit тести для модулів програмної системи?
34. Як проводити аналіз професійних стандартів та етики у розробці?
35. Як створити професійний профіль розробника на GitHub та LinkedIn?

7. РЕКОМЕНДОВАНА ЛІТЕРАТУРА

7.1 Базова література

1. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). Version 4.0 / ed. H. Washizaki. IEEE Computer Society, 2024. URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering

2. Sommerville I. Software Engineering. 10th ed. Pearson, 2016. URL: https://software-engineering-book.com/

3. Левус Є.В., Мельник Н.Б. Вступ до інженерії програмного забезпечення: навч. посібник. Львів: Видавництво Львівської політехніки, 2017. 280 с.

4. Петрик М.Р., Петрик О.Ю. Моделювання програмного забезпечення. Тернопіль: Вид-во ТНТУ, 2015. 200 с.

5. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

7.2 Допоміжна література

6. Pro Git book. URL: https://git-scm.com/book/uk/v2

7. GitHub Foundations Part 1 of 2. URL: https://learn.microsoft.com/en-us/training/paths/github-foundations/

8. GitHub Copilot Fundamentals Part 1 of 2. URL: https://learn.microsoft.com/en-us/training/paths/copilot/

9. EU AI Act Fundamentals. URL: https://app.datacamp.com/learn/skill-tracks/eu-ai-act-fundamentals

10. Datacamp. URL: https://app.datacamp.com/

11. Martin Fowler's blog. URL: https://martinfowler.com/

12. Visual Studio Code. URL: https://code.visualstudio.com/

13. Git downloads. URL: https://git-scm.com/downloads

7.3 Інформаційні ресурси та електронні джерела

14. Основи програмної інженерії (SE201): електронний навчальний курс. URL: https://dl.tntu.edu.ua/bounce.php?course=6822

15. TNTU-F2-Software-Engineering: GitHub repository. URL: https://github.com/TNTU-F2-Software-Engineering

16. IEEE Standards. URL: https://www.ieee.org/standards/

17. ISO/IEC Standards. URL: https://www.iso.org/

18. UML Documentation. URL: https://www.uml.org/

19. GitHub Guides. URL: https://guides.github.com/

20. Markdown Guide. URL: https://www.markdownguide.org/

7.4 Матеріали дисципліни

- Основи програмної інженерії (SE201): електронний навчальний курс доступний у СЕН ТНТУ ім. І. Пулюя.
- Методичні вказівки до лабораторних робіт з дисципліни «Основи програмної інженерії (SE201)»: інструктивно-методичні матеріали / уклад. В. М. Бревус. Тернопіль: Вид-во ТНТУ ім. Івана Пулюя, 2025. 33 с.
- Конспект лекцій з дисципліни «Основи програмної інженерії (SE201)» для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Інженерія програмного забезпечення» всіх форм навчання / Укладач: Бревус В.М., Бревус Г.Б. – Тернопіль: ТНТУ імені Івана Пулюя, 2025 – 127 с.

7.5 Рекомендований список для поглибленого вивчення

- **Для теоретичної підготовки:** Матеріали конспекту лекцій та класичні підручники з програмної інженерії, SWEBOK Guide, стандарти IEEE та ISO.
- **Для практичного застосування:** Репозиторії на GitHub з прикладами проєктів з відкритим кодом, case studies з розробки ПЗ, матеріали про Design Patterns та SOLID принципи.
- **Для розробки проєктів:** Шаблони проєктів на GitHub, документація для Git та GitHub, інструменти для UML-моделювання (Lucidchart, Draw.io).
- **Для актуальної інформації:** Наукові журнали та конференції з програмної інженерії (IEEE, ACM, USENIX), блоги Martin Fowler та інших видатних фахівців у галузі.
- **Для професійного розвитку:** Ресурси для навчання Git та GitHub, матеріали з agile методологій, освітні курси на Coursera, Udemy та DataCamp.
- **Для використання AI у розробці:** GitHub Copilot документація, промпт-інженерія, приклади використання ШІ для генерації коду.

Зазначені матеріали можна знайти у:

- Бібліотеці ТНТУ ім. І. Пулюя;
- Електронних базах даних Інтернету;
- Науковому каталогу УДКон (UNILIB);
- Світовій системі цитування Google Scholar;
- Офіційних сайтах та документації стандартизаційних організацій (IEEE, ISO).