ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ЗАТВЕРДЖУЮ
Декан факультету _____ Савенко О.С.
„1” _____ 2023 р.

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Архітектура та проєктування програмного забезпечення

Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Рівень вищої освіти – Перший бакалаврський
Освітньо-професійна програма – Інженерія програмного забезпечення
Обсяг дисципліни –7 кредитів ЄКТС, ***Шифр дисципліни*** – ОПП.01
Статус дисципліни: обов’язкова, ***Мова навчання*** Англійська, українська
Факультет – Інформаційних технологій
Кафедра – Інженерії програмного забезпечення

Форма навчання	Курс	Семестр	Загальне навантаження		Кількість годин						Курсовий проект	Курсова робота	Форма семестрового контролю	
					Аудиторні заняття				Індивідуальна робота студента	Самостійна робота студента в т.ч. ІРС				
			Кредити ЄКТС	Години	Разом	Лекції	Лабораторні роботи	Практичні заняття					Залік	Іспит
Очна (денна)	3	6	7	210	68	34	34			142	+			+
Разом			**7**	**210**	**68**	**34**	**34**			**142**	**1**			**1**

Робоча програма складена на основі Стандарту вищої освіти, освітньої програми підготовки бакалаврів 2023 року та навчального плану.

Програма складена _____ Форкун Ю. В.

_____ Яшина О.М.

Схвалено на засіданні кафедри ІПЗ протокол № 1 від 31 08 2023 р.

Зав. кафедри інженерії програмного забезпечення _____ Л.П. Бедратюк
Робоча програма розглянута та схвалена Вченою радою факультету інформаційних технологій
Голова Вченої ради _____ О.С. Савенко
Підпис Ініціали, прізвище

Хмельницький 2023

АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тип дисципліни	Обов'язкова
Рівень вищої освіти	Перший (бакалаврський)
Мова викладання	Українська
Семестри	шостий
Обсяг кредитів ЄКТС	7
Форми здобуття освіти	Очна (денна)

Результати навчання

Результати навчання. Студент, який успішно завершив вивчення дисципліни, має: Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки. Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення. Проводити передпроєктне обстеження предметної області, системний аналіз об'єкта проєктування. Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення. Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проєктування, тестування, візуалізації, вимірювань та документування програмного забезпечення. Розуміти сутнісні характеристики архітектури програмних застосунків і підходи до їх аналізу та проектування, основні типи архітектур та їх особливості, задачі проектування ПЗ; використовувати мову моделювання UML для проектування програмних застосунків; використовувати стандартні засоби і нотації для документування архітектури та інтерфейсу ПЗ; володіти засобами проектування програмного забезпечення і методами структурного та об'єктно-орієнтованого проектування програмного забезпечення; вміти користуватися архітектурними шаблонами проектування та застосовувати їх при створенні програмних застосунків; створювати ПЗ використовуючи типові архітектурні стилі та рішення з врахуванням особливостей застосування; проектувати корпоративне ПЗ з використанням типової архітектури корпоративних систем та платформ.

Зміст навчальної дисципліни. Вступ до архітектури програм. Моделі, каркаси та зразки проектування. Типи архітектур та їх моделі. Архітектура: нотація, стандарти та інструментальні засоби. Архітектурні шаблони і стилі. Контроль якості при виборі архітектури. Вступ в детальне проектування. Діаграми послідовності та діаграми потоків даних в детальному проектуванні. Специфікація алгоритмів, класів та функцій. Зразки проектування. Бібліотеки стандартних шаблонів. Стандарти, нотація та інструментальні засоби проектування. Вплив детального проектування на проект

Пререквізити – Веб технології, Моделювання та оцінка програмного забезпечення

Кореквізити – Конструювання програмного забезпечення, Безпека програм і даних

Запланована навчальна діяльність: лекції – 34 год, лабораторні заняття – 34 год, самостійна робота – 142 год, разом – 210 год.

Форми (методи) навчання: лекції (з використанням методів візуалізації); лабораторні заняття (з використанням методів комп'ютерного моделювання), самостійна робота

Форми оцінювання результатів навчання: захист лабораторних робі, тестування, іспит, захист курсового проекту

Форма семестрового контролю: іспит, захист курсового проекту

Навчальні ресурси:

1. Форкун Ю. В., Яшина О.М. Методичні вказівки до виконання курсового проекту з дисципліни «Архітектура та проектування програмного забезпечення» для студентів спеціальності 121 «Інженерія програмного забезпечення»/ Ю. В. Форкун, О.М.Яшина. – Хмельницький: ХНУ, 2023. – 42 с.
2. Конспект лекцій з дисципліни «Архітектура та проектування програмного забезпечення» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Інженерія програмного забезпечення» із спеціальності 121 – «Інженерія програмного забезпечення» / Укл. В.В.Завгородній, К.М.Ялова. – Кам'янське: ДДТУ, 2019.– 144с.
3. Мартін Р. Чиста архітектура /Роберт Мартін. – Харків: Фабула, 2019. – 416 с.
4. Фрімен Е. Head First. Патерни проєктування /Ерік Фрімен, Елізабет Робсон. – Харків: Фабула, 2020. – 672 с.

5. Richards M. Fundamentals of Software Architecture: An Engineering Approach / Mark Richards, NealFord. – Sebastopol, California: O'Reilly Media – 1st edition, 2020. – 419 p.
6. Lanciaux R. Modern Front-end Architecture: Optimize Your Front-end Development with Components, Storybook, and Mise en Place Philosophy / Ryan Lanciaux – New York: Apress, 2021 – 144 p.
7. Frighi V. Smart Architecture – A Sustainable Approach for Transparent Building ComponentsDesign / Valentina Frighi – Berlin: Springer – 1st edition, 2022. – 293 p.
8. Модульне середовище для навчання. Доступ до ресурсу: https://msn.khmnu.edu.ua.
9. Електронна бібліотека університету. Доступ до ресурсу: http://library.khmnu.edu.ua
10. Репозитарій ХНУ:https://elar.khmnu.edu.ua/home

Викладачі: кандидат технічних наук, доцент Оксана ЯШИНА О.М., кандидат технічних наук, доцент Юрій ФОРКУН

3. ПОЯСНЮВАЛЬНА ЗАПИСКА

Дисципліна «Архітектура та проектування програмного забезпечення» є однією із дисциплін професійної підготовки і займає провідне місце у підготовці фахівців освітнього рівня "бакалавр" за освітньо-професійною програмою "Інженерія програмного забезпечення".

Мета дисципліни. На основі вивчення теоретичних основ з моделювання, розробки та проектування на основі шаблонів та каркасів програмного забезпечення, а також вивчення основних стандартів, вимог та готових конструкцій, які використовуються при розробці та проектуванні великих програмних додатків, студенти мають набути практичних навичок створення архітектури та детального проектування програмних продуктів з використанням сучасних методів та засобів.

Пререквізити – Веб технології, Моделювання та оцінка програмного забезпечення

Кореквізити – Конструювання програмного забезпечення, Безпека програм і даних

Предмет дисципліни. Теорія і практика застосування базових методів та засобів розробки і проектування архітектури ПЗ.

Завдання дисципліни. Надати студентам знання і практичні навички з основ проектування та розробки архітектури ПЗ.

Відповідно до ***Стандарту вищої освіти*** із та освітньої програми дисципліна сприяє забезпеченню:

компетентностей:

ФК2. Здатність брати участь у проєктуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його струк- тури, поведінки та процесів функціонування.

ФК3. Здатність розробляти архітектури, модулі та компоненти програмних систем.

ФК10. Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя.

ФК11. Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення.

ФК12. Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.

ФК13. Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

програмних результатів навчання:

ПРН1 Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.

ПРН3. Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення.

ПРН10. Проводити передпроєктне обстеження предметної області, системний аналіз об'єкта проєктування.

ПРН11. Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання.

ПР12. Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення.

ПР14. Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проєктування, тестування, візуалізації, вимірювань та документування програмного забезпечення.

Результати навчання. Студент, який успішно завершив вивчення дисципліни, має: Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки. Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення. Проводити передпроєктне обстеження предметної області, системний аналіз об'єкта проєктування. Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення. Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проєктування, тестування, візуалізації, вимірювань та документування програмного забезпечення. Розуміти сутнісні характеритики архітектури програмних застосунків і підходи до їх аналізу та проектування, основні типи архітектур та їх особливості, задачі проектування ПЗ; використовувати мову моделювання UML

для проектування програмних застосунків; використовувати стандартні засоби і нотації для документування архітектури та інтерфейсу ПЗ; володіти засобами проектування програмного забезпечення і методами структурного та об'єктно-орієнтованого проектування програмного забезпечення; вміти користуватися архітектурними шаблонами проектування та застосовувати їх при створенні програмних застосунків; створювати ПЗ використовуючи типові архітектурні стилі та рішення з врахуванням особливостей застосування; проектувати корпоративне ПЗ з використанням типової архітектури корпоративних систем та платформ.

Політика дисципліни Організація освітнього процесу з дисципліни відповідає вимогам положень про організаційне і навчально-методичне забезпечення освітнього процесу, освітній програмі та навчальному плану. Студент зобов’язаний відвідувати лекції, практичні заняття, лабораторні роботи, тощо, згідно з розкладом, не запізнюватися на заняття, виконувати усі завдання та контрольні точки відповідно до графіка. Пропущені практичні заняття і лабораторні роботи студент зобов’язаний опрацювати самостійно у повному обсязі і відзвітувати перед викладачем не пізніше, ніж за тиждень до чергової атестації. До практичних занять і лабораторних робіт студент має підготуватися за відповідною темою і проявляти активність. Набутті особою знання з дисципліни або її окремих розділів у неформальній освіті зараховуються відповідно до Положення про порядок перезарахування результатів навчання та визначення академічної різниці у ХНУ.

4. СТРУКТУРА ЗАЛІКОВИХ КРЕДИТІВ ДИСЦИПЛІНИ

Тема	Лекції	Лаб. роб.	Сам. роб. Повна
Денна форма навчання (3 курс, 2 семестр)			
Тема 1. Вступ до архітектури програмного забезпечення	2	4	6
Тема 2. Моделі, каркаси та зразки проектування	4		8
Тема 3. Типи архітектур та їх моделі	2	4	12
Тема 4. Архітектура: нотація, стандарти та інструментальні засоби проектування	2	8	12
Тема 5. Архітектурні шаблони і стилі	2		12
Тема 6. Контроль якості при виборі архітектури	2		12
Тема 7. Вступ в детальне проектування	4	8	14
Тема 8. Діаграми послідовності та діаграми потоків даних в проектуванні програмного забезпечення	2	4	12
Тема 9. Специфікація алгоритмів, класів та функцій	4		14
Тема 10. Зразки проектування	4	4	14
Тема 11. Бібліотеки стандартних шаблонів	4	2	14
Тема 12. Вплив детального проектування на проект	2		12
Всього за рік:	**34**	**34**	**142**

5. ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

5.1. Зміст лекційного курсу

№ Лекції	Перелік тем лекцій	Кількість Годин
1	*2*	*3*
1	Вступ до архітектури програмного забезпечення. Огляд технології розробки. Поняття архітектури програмного забезпечення. Цілі вибору архітектури. Декомпозиція [1]; [2];[3];[4];[5]	2
2	Моделі, каркаси та зразки проектування. Використання моделей. Уніфікована мова моделювання (UML). Каркаси. [1]; [2]; [3]	2
3	Моделі, каркаси та зразки проектування. Класифікація архітектур. Зразки проектування. Компоненти.[1]; [4]	2
4	Типи архітектур та їх моделі. Архітектури, засновані на потоках даних. Незалежні компоненти. Віртуальні машині. Репозиторні архітектури. Рівневі архітектури. Додатки зі змішаною архітектурою. Процедура вибору архітектури [1]; [2] ;[4];[5]	2
5.	Архітектура: нотація, стандарти та інструментальні засоби. Нотація. Інструментальні засоби. Стандарт IEEE/ANSI для опису проекта [1]; [2] ;[3];[4]	2
6.	Архітектурні шаблони і стилі. Основні архітектурні стилі. Поєднання архітектурних стилів. Архітектура клієнт/сервер. Компонентна архітектура. Проектування на основі предметної області. Багатошарова архітектура. Об'єктно-орієнтована архітектура. Архітектура, заснована на шині повідомлень. Сервісно-орієнтована архітектура. [1]; [2];[3] ;[4];[6]	2
7	Контроль якості при виборі архітектури. Якість та вибір архітектури. Вибір з альтернативних архітектур. Перевірка архітектури з допомогою варіантів використання. Інспектування вибору архітектури. Вплив вибору архітектури на SPMP. [1]; [2]; [3];[6]	2
8	Вступ в детальне проектування. Поняття детального проектування. Співвідношення варіантів використання, архітектури та детального проектування. Типова схема процесу детального проектування. [1]; [2] ; [3] ;[4];[5]	2
9	Вступ в детальне проектування. Проектування по схемі USDP – вимоги, аналіз, проектування, реалізація, тестування. Проектування та інтерфейси. Повторно використовувані елементи.[1]; [2] ; [3] ; [5]	2
10	Діаграми послідовності та діаграми потоків даних в детальному проектуванні. Детальні діаграми послідовності. Детальні діаграми потоків даних [1]; [2] ; [3] ;[4];[6]	2
11	Специфікація алгоритмів, класів та функцій. Інваріанти класу. Інваріанти, передумови та післяумови функцій. Блок-схеми. Псевдокод. Використання блок-схем та псевдокоду [1]; [2]; [3];[4]	2
12	Специфікація алгоритмів, класів та функцій. Інваріанти класу. Інваріанти, передумови та післяумови функцій. Блок-схеми. Псевдокод. Використання блок-схем та псевдокоду [1];[2];[3];[4];][5];[6]	2
13	Зразки проектування. Прийоми детального проектування. Креаційні зразки проектування. Структурні зразки проектування. Зразки проектування, засновані на поведінці додатка [1];[2]; [3];[4];[6]	2

14	Зразки проектування. Патерни поведінки. Антипатерни проектування [3];[4];[6]	2
15	Бібліотеки стандартних шаблонів. Стандарти, нотація та інструментальні засоби проектування. Бібліотека стандартних шаблонів (STL) C++. [3];[4];[6]	2
16	Бібліотеки стандартних шаблонів. Стандарт IEEE 890. Мова UML. Інструменти, що використовують вихідний код: Javadoc [1] ; [3] ;[4];[6]	2
17	Вплив детального проектування на проект. Оцінка об'єму робіт з допомогою детального проектування. Якість і метрики в детальному проектуванні. Інспектування для детального проектування [1]; [2] ; [3] ;[4];[5]	2
	Всього за семестр:	34

5.2 ЗМІСТ ЛАБОРАТОРНИХ ЗАНЯТЬ

№ лаб. заняття	Теми лабораторних занять та їх зміст	Кіль-кість годин
1	*2*	*3*
1.	Лабораторна робота №1 Оцінка та вибір компонентів програмного забезпечення	4
2.	Лабораторна робота №2 Використання різних типів архітектур програмного забезпечення. Шаблони проектування	4
3.	Лабораторна робота №3 Робота додатків із змішаною архітектурою	4
4.	Лабораторна робота №4 Використання стандартів та інструментальних засобів при розробці архітектури програмного забезпечення	4
5.	Лабораторна робота №5 Початковий етап проектування. дослідження предметної області	4
6.	Лабораторна робота №: Розробка архітектури програмної системи: вибір типу архітектури та опис декомпозиції	4
7.	Лабораторна робота №7 Розробка архітектури програмної системи: опис залежностей та опис інтерфейсу	4
8.	Лабораторна робота №8 Детальне проектування програмної системи: модулі та дані	6
	Всього:	34

5.3 ЗМІСТ САМОСТІЙНОЇ РОБОТИ

Самостійна робота студентів ***денної*** форми навчання полягає у систематичному опрацюванні програмного матеріалу, підготовці до виконання і захисту лабораторних робіт, тестування з теоретичного матеріалу, виконанні індивідуальних завдань, в тому числі курсового проекту, тощо.

Номер теми	**Зміст самостійної роботи**	**Кількість годин**
1	*2*	*3*
1	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 1. Визначення тематики курсового проекту.	8
2	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 1. Визначення завдань до КП. Аналіз останніх публікацій, досліджень та існуючих рішень	8
3	Опрацювання лекційного матеріалу, підготовка до виконання лабораторних робіт № 2. Постановка задачі та перелік задач для реалізації відповідно до тематики КП.	8
4	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 2. Постановка задачі та перелік задач для реалізації відповідно до тематики КП.	8
5	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 3. Вибір типу архітектури програмного забезпечення відповідно до завдань КП	8
6	Опрацювання лекційного матеріалу, підготовка до захисту лабораторних робіт № 3. Вибір типу архітектури програмного забезпечення	8
7	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 4. Опис декомпозиції відповідно до завдань КП	8
8	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 4. Опис декомпозиції відповідно до завдань КП	8
9	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 5. Опис залежностей	8
10	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 5. Опис залежностей відповідно до завдань КП	10
11	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 6. Опис інтерфейсу відповідно до завдань КП	8
12	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 6. Опис інтерфейсу відповідно до завдань КП	10
13	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 7. Детальне проектування модулів відповідно до завдань КП	8
14	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 7. Детальне проектування модулів відповідно до завдань КП	10

15	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 8. Детальне проектування даних відповідно до завдань КП	8
16	Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи № 8. Оформлення пояснювальної записки КП та перевірка на антиплагіат. Підготовка до захисту КП.	8
17	Опрацювання лекційного матеріалу, підготовка до захисту лабораторної роботи № 8. Підготовка до захисту КП.	8
	Разом за семестр	**142**

5.4 КУРСОВЕ ПРОЕКТУВАННЯ

На індивідуальну роботу студентам повної форми навчання в 6 семестрі заплановано виконання курсового проекту з дисципліни „Архітектура та проектування програмного забезпечення”, який захищається на шістнадцятому тижні VI семестру для денної форми навчання.

Метою курсового проектування є поглиблення та закріплення теоретичних знань, одержаних студентом при вивченні даної дисциплін, застосування їх для комплексного вирішення питань, які виникають при розробці архітектури та проектуванні програмного забезпечення.

У процесі роботи над курсовим проектом студент повинен проявити ініціативу та самостійність при обґрунтуванні та виборі конструктивних і технологічних рішень, що будуть застосовані при проектуванні програмного продукту, використовуючи при цьому знання та практичні навички з дисциплін, які вивчалися раніше, зокрема об’єктно-орієнтованого програмування, моделювання програмного забезпечення, бази даних, аналізу вимог до програмного забезпечення.

Приблизний обсяг розрахунково-пояснювальної записки від 30 до 50 сторінок. При виконанні курсового проекту студенти користуються відповідними методичними рекомендаціями. Тематика курсових проектів затверджується на засіданні кафедри.

Консультації з курсового проекту здійснює керівник проекту – викладач кафедри відповідно до графіку, затвердженого на засіданні кафедри. Захист курсового проекту здійснюється після його перевірки керівником перед комісією, яка складається з 2-3 викладачів кафедри. До складу комісії обов’язково входить керівник проекту.

Критерії оцінювання. Оцінювання курсового проекту здійснюється за видами робіт згідно з розподілом і ваговими коефіцієнтами. Оцінка «відмінно/A» виставляється за високоякісно виконані програмний продукт і пояснювальну записку без помилок, оригінальність програмного продукту, дотримання вимог оформлення. Доповідь і захист роботи обґрунтовані, виявлені комплексні знання зі спеціальних дисциплін стосовно теми курсового проекту.

Оцінка «добре/B» виставляється за якісне виконання курсового проекту при одній-двох незначних помилках чи недосить впевнені відповіді на одне-два питання комісії. Оцінка «добре/C» виставляється за якісно виконаний проект, дотриманні усіх вимог, що пред’являються до курсового проекту, за дві-три незначні помилки при реалізації поставленої задачі чи пояснювальній записці, не чіткі відповіді на два-три питання комісії.

Оцінка «задовільно/D» виставляється, якщо в пояснювальній записці чи програмному продукті виявлені помилки, є незначні порушення вимог до оформлення проекту, невпевнені відповіді на основні питання з теми проекту.

Оцінка «задовільно/E» виставляється, якщо в проекті виявлені суттєві помилки як в пояснювальній записці, так і в розробленому продукті, неправильно обґрунтовані прийняті конструкторські рішення, грубі помилки у відповідях на запитання членів комісії, невпевненому захисті в цілому.

Оцінка «незадовільно/FX/F» виставляється, якщо в проекті вибрані неправильні методи проектування, або за невідповідність змісту і лістингу програми затвердженій темі курсового проекту, коли студент не орієнтується в тому, що виконав. У цьому випадку студент представляє виправлену роботу на повторний захист, або йому видається нова тема проекту і призначається термін його виконання і захисту.

Приблизна тематика курсових проектів

1. Розробка архітектури і компонентів програмно-інформаційного забезпечення інформаційної системи підтримки ухвалення рішень по формуванню виробничо-операційної стратегії підприємств.

2. Розробка архітектури і компонентів системи планування завдань, що використовує алгоритми економічних моделей.

3. Розробка архітектури і компонентів програмного комплексу для аналізу біометричних сигналів на прикладі системи аналізу клавіатурного почерку.

4. Розробка архітектури і компонентів інтелектуального агента на основі моделі життєздатної системи.

5. Розробка архітектури і компонентів системи безпеки веб-застосунку.

6. Розробка архітектури і компонентів підсистеми інформаційної безпеки.

7. Розробка архітектури і компонентів розподіленої пошукової системи.

8. Розробка архітектури і компонентів інструментальних засобів підтримки систем вирішенню завдань.

9. Розробка архітектури і компонентів системи розпізнавання жестів.

10. Розробка архітектури і компонентів програмного забезпечення для електронної бібліотеки.

11. Розробка архітектури і компонентів автоматизованої системи тестування.

В індивідуальному завданні на курсове проектування конкретизуються конструктивні особливості створюваного програмного продукту, та їх поетапне виконання. Завдання видається студенту на першому тижні семестру, а захист курсового – на 16 тижні. Курсовий проект студенти виконують згідно з методичними вказівками до курсового проектування з дисципліни «Архітектура та проектування програмного забезпечення»

6. МЕТОДИ НАВЧАННЯ

Процес навчання з дисципліни ґрунтується на використанні традиційних та сучасних методів. Зокрема, лекції проводяться в основному словесними методами, а лабораторні заняття проводяться з використанням інформаційних технологій, майстер-класів, практикумів і мають за мету – набуття студентами практичних навичок з проектування та розробки архітектури програмного забезпечення за сучасними методиками, користування спеціальними інструментами проектування тощо.

7. ФОРМИ І МЕТОДИ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ

Кожний вид роботи з дисципліни оцінюється за чотирибальною шкалою. Семестрова підсумкова оцінка визначається як середньозважена з усіх видів навчальної роботи, виконаних і зданих позитивно з врахуванням коефіцієнта вагомості. Вагові коефіцієнти змінюються залежно від структури дисципліни і важливості окремих її видів робіт.. Студент, який набрав позитивний середньозважений бал за поточну роботу і не здав підсумковий контрольний захід (іспит), вважається невстигаючим.

При оцінюванні знань студентів використовуються різні засоби контролю, зокрема: усне опитування перед допуском до виконання лабораторної роботи – здійснюється на її початку; засвоєння теоретичного матеріалу з тем перевіряється тестовим контролем; якість виконання, набуття теоретичних знань і практичних навичок перевіряється шляхом захисту кожної лабораторної роботи та індивідуального завдання згідно з робочою програмою дисципліни і робочим навчальним планом.

Оцінка, яка виставляється за лабораторне заняття, складається з таких елементів: усне опитування студентів перед допуском до виконання лабораторної роботи; знання теоретичного матеріалу з теми; якість оформлення протоколу і графічної частини; вміння студента обґрунтувати прийняті конструктивні рішення.

Пропущене лабораторне заняття студент повинен відпрацювати в лабораторіях кафедри не пізніше, ніж за два тижні до кінця теоретичних занять у семестрі.

При оцінюванні знань студентів викладач керується такими критеріями.

Оцінка «відмінно» виставляється студенту, який глибоко засвоїв матеріал та вміє його раціонально застосувати, знає методики та вміє ними користуватися при проектуванні та розробки архітектури ПЗ. Відмінна оцінка передбачає грамотний, логічний виклад відповіді (як в усній, так і в письмовій формі), якісне зовнішнє оформлення. Студент повинен набути практичних навичок з розробки та проектування різних типів ПЗ. Студент не повинен вагатися при видозміні запитання, повинен робити детальні та узагальнюючі висновки.

Оцінку «добре» отримує студент за повне засвоєння навчального матеріалу, володіння понятійним апаратом, орієнтування в вивченому матеріалі, свідоме використання знань для вирішення практичних завдань, грамотний виклад відповіді, але у змісті і формі відповіді мали місце окремі неточності (похибки), нечіткі формулювання закономірностей тощо. Відповідь студента має будуватись на основі самостійного мислення.

Оцінку «добре» отримує студент за правильну відповідь з двома-трьома суттєвими помилками.

Оцінки «задовільно» заслуговує студент за неповне опанування програмного матеріалу, але отримані знання і набуті практичні навички з розроблення та проектування ПЗ відповідають мінімальним критеріям оцінювання.

Оцінка „незадовільно” виставляється, коли студент має розрізнені, безсистемні знання, не вміє виділяти головне і другорядне, допускається помилок у визначенні понять, перекручує їх зміст, хаотично і невпевнено викладає матеріал, не може використовувати знання при вирішенні практичних завдань. Як правило, оцінка "незадовільно" виставляється студенту, який не може продовжити навчання без додаткових знань з курсу.

На основі результатів поточного контролю і підсумкового контрольного заходу виставляється підсумкова семестрова оцінка. На основі аналізу контролю знань викладач удосконалює курс лекцій, звертаючи особливу увагу на ті розділи, чи теми, з яких було найбільше неточних відповідей, що свідчить про методичні чи інші недоліки при висвітленні

вказаних тем або розділів.

Аналогічно вносяться корективи в методичні посібники з лабораторного практикуму та курсового проектування, детальніше розглядаються принципові питання з проектування ПЗ при виконанні лабораторних робіт, курсового проекту та їх захисті.

Структурування дисципліни за видами робіт і оцінювання результатів навчання студентів за ваговими коефіцієнтами

<table>
<tr><th colspan="8">Аудиторна робота</th><th>Самостійна робота</th><th>Підсумковий контроль</th></tr>
<tr><td colspan="8">Лабораторні роботи</td><td>Тестовий контроль</td><td>ІСПИТ</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>7</td><td></td><td></td></tr>
<tr><td colspan="8">0,4</td><td>0,2</td><td>0,4</td></tr>
<tr><td colspan="10">VI семестр (курсовий проект)</td></tr>
<tr><td colspan="2">1
розділ</td><td colspan="2">2
розділ</td><td colspan="4">3
розділ</td><td>Оформлення
КП</td><td>Захист</td></tr>
<tr><td colspan="2">0,2</td><td colspan="2">0,25</td><td colspan="4">0,25</td><td>0,1</td><td>0,2</td></tr>
</table>

Перехід від вітчизняної шкали оцінювання до європейської (ECTS)

Оцінка ECTS	Бали	Вітчизняна система	
A	4,75-5,00	5	ВІДМІННО - глибоке і повне опанування навчального матеріалу і виявлення відповідних умінь та навичок
B	4,25-4,74	4	ДОБРЕ - повне знання навчального матеріалу з кількома незначними помилками
C	3,75-4,24	4	ДОБРЕ - в загальному правильна відповідь з двома-трьома суттєвими помилками
D	3,25-3,74	3	ЗАДОВІЛЬНО - неповне опанування програмного матеріалу, але достатнє для практичної діяльності за професією
E	3,00-3,24	3	ЗАДОВІЛЬНО - неповне опанування програмного матеріалу, що задовольняє мінімальні критерії оцінювання
FX	2,00-2,99	2	НЕЗАДОВІЛЬНО - безсистемність одержаних знань і неможливість продовжити навчання без додаткових знань з дисципліни
F	0,00-1,99	2	НЕЗАДОВІЛЬНО - необхідна серйозна подальша робота і повторне вивчення дисципліни

8. ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Поняття системної розробки.
2. Поняття програмного забезпечення.
3. Поняття створення архітектури.
4. Цілі вибору архітектури.
5. Поняття декомпозиції.
6. Поняття детального проектування.
7. Використання моделей.
8. Каркаси.
9. Класифікація архітектур.
10. Зразки проектування.
11. Компоненти.
12. Типи архітектур та їх моделі.
13. Архітектури засновані на потоках даних.
14. Незалежні компоненти.
15. Віртуальні машини.
16. Репозиторні архітектури
17. Рівневі архітектури.
18. Клієнт-серверна архітектура.
19. Архітектура заснована на предметній області.
20. Процедура вибору архітектури.
21. Інструментальні засоби вибору архітектури.
22. Контроль якості при виборі архітектури.
23. Перевірка архітектури з допомогою варіантів використання.
24. Інспектування вибору архітектури.
25. Поняття детального проектування.
26. Співвідношення варіантів використання архітектури та детального проектування.
27. Типова схема детального проектування.
28. Проектування та інтерфейси.
29. Повторно використовувані компоненти.
30. Детальні діаграми послідовності.
31. Детальні діаграми потоків даних.
32. Специфікація класів та функцій.
33. Інваріанти класів.
34. Інваріанти, передумови та післяумови.
35. Специфікація алгоритмів.
36. Блок-схеми.
37. Псевдокод.
38. Порівняння використання блок-схеми та псевдокоду.
39. Зразки проектування: прийоми детального проектування.
40. Креаційні зразки проектування.
41. Структурні зразки проектування.

42. Зразки проектування засновані на поведінці додатків.
43. Що таке шаблон проектування?
44. Які типи шаблонів проектування вам відомі?
45. Шаблон проектування «Стратегія».
46. Шаблон проектування «Будівельник».
47. Шаблон проектування «Фабрика», «Абстрактна фабрика».
48. Шаблон проектування «Адаптер».
49. Шаблон проектування «Міст».
50. Шаблон проектування «Команда».
51. Шаблон проектування «Компонувальник».
52. Шаблон проектування «Ітератор».
53. Шаблон проектування «Медіатор».
54. Шаблон проектування «Ланцюг відповідальності».
55. Шаблон проектування «Оглядач».
56. Шаблон проектування «Посередник».
57. Шаблон проектування «Одинак».
58. Шаблон проектування «Стан».
59. Шаблон проектування «Зразок».
60. Що таке аналіз якості програмного дизайну?
61. Основні оцінки програмного дизайну.
62. Які нотації та засоби підтримки проектування відомі?
63. Методи аналізу компромісних архітектурних рішень.
64. Основні етапи методу аналізу компромісних архітектурних рішень.
65. Метод аналізу вартості та ефективності архітектурних рішень та програмноих застосунків.

9. МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ

Навчальний процес з дисципліни «Архітектура та проектування програмного забезпечення» повністю і в достатній кількості забезпечений необхідною навчально-методичною літературою. Всі необхідні навчально-методичні матеріали розміщені у Модульному навчальному середовищі. Доступ до ресурсу: https://msn.khmnu.edu.ua/course/view.php?id=4908.

10. РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна:

1. Конспект лекцій з дисципліни «Архітектура та проектування програмного забезпечення» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Інженерія програмного забезпечення» із спеціальності 121 – «Інженерія програмного забезпечення» / Укл. В.В.Завгородній, К.М.Ялова. – Кам'янське: ДДТУ, 2019.– 144с.
2. Мартін Р. Чиста архітектура /Роберт Мартін. – Харків: Фабула, 2019. – 416 с.
3. Фрімен Е. Head First. Патерни проєктування /Ерік Фрімен, Елізабет Робсон. – Харків: Фабула, 2020. – 672 с.
4. Richards M. Fundamentals of Software Architecture: An Engineering Approach / Mark Richards, NealFord. – Sebastopol, California: O'Reilly Media – 1st edition, 2020. – 419 p.
5. Lanciaux R. Modern Front-end Architecture: Optimize Your Front-end Development with Components, Storybook, and Mise en Place Philosophy / Ryan Lanciaux – New York: Apress, 2021 – 144 p.
6. Frighi V. Smart Architecture – A Sustainable Approach for Transparent Building ComponentsDesign / Valentina Frighi – Berlin: Springer – 1st edition, 2022. – 293 p.
7. Кудрявцев В. В., Форкун Ю. В. Аналіз та застосування методів оптимізації швидкодії та відмовостійкості програмних продуктів // Зб. наук. пр. наукової конференції «АПКН- 89 2021». Хмельницький ХНУ. – 2021. – с.338-339
8. Форкун, Ю., Форкун, І., Яшина, О., & Праворська, Н. (2023). Архітектурні методи оптимізації швидкодії та відмовостійкості програмних застосунків . measuring and computing devices in technological processes, (2), 196–201. https://doi.org/10.31891/2219-9365-2023-74-27
9. Ivan Lopatto, Mykyta Lebiga, Yurii Forkun and Artem Boyarchuk. Method for Determining the Informativeness of the Software Requirements Specifications. IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine. http://ceur-ws.org/Vol-2853/paper15.pdf

Додаткова

1. Бородкіна, І.Л. Інженерія програмного забезпечення: навч. посібник / І. Л.Бородкіна, Г. О. Бородкін ; НУБіП. — Київ : Центр учбової літ., 2020. — 204 с.
2. Lanciaux R. Modern Front-end Architecture: Optimize Your Front-end Development with Components, Storybook, and Mise en Place Philosophy / Ryan Lanciaux – New York: Apress, 2021 – 144 p.
3. Frighi V. Smart Architecture – A Sustainable Approach for Transparent Building ComponentsDesign / Valentina Frighi – Berlin: Springer – 1st edition, 2022. – 293 p.
4. Роберт Мартін. Чиста архітектура. Харків:Фабула. – 2019. – 368с. ISBN : 978-617-09-5286-8
5. Роберт Мартін. Чистий код. Харків:Фабула. – 2019. – 416с. ISBN : 978-617-09-5285-1
6. Олексій Васильєв. Програмування мовою Python. Тернопіль: НАВЧАЛЬНА КНИГА – БОГДАН. - 2019. – 504 с. ISBN : 9789661056113
7. Елізабет Робсон , Ерік Фрімен. Head First. Патерни проєктування. Харків:Фабула. – 2020. – 672с. ISBN : 978-617-09-6159-4
8. Елізабет Робсон , Ерік Фрімен. Head First. Програмування на JavaScript. Харків:Фабула. – 2022. – 672с. ISBN : 978-617-522-047-4
9. Пол Беррі.Head First. Python. Харків:Фабула. – 2021. – 624с. ISBN : 978-617-522-019-1

10. Роб Коул. Блискучий Agile. Харків:Фабула. – 2020. – 192с. ISBN : 978-617-09-6381-9
11. Берт Бейтс , Кеті Сьєрра. Head First. Java. Харків:Фабула. – 2022. – 720 с. ISBN : 978-617-522-033-7
12. Юрій Рамський, Василь Олексюк, Анатолій. Адміністрування комп'ютерних мереж і систем. Тернопіль: НАВЧАЛЬНА КНИГА – БОГДАН. - 2020. – 196 с. ISBN : 9789661015615
13. Емі Вебб. Велика дев'ятка. Як IT-гіганти та їхні розумні машини можуть змінити людство. Харків:Vivat. – 2020. – 352 с. ISBN : 9789669822185
14. Денис Каплунов. Королі соціальних мереж. .Київ: BookChef – 2022. – 432 с. ISBN: 9786175480922

11. ІНФОРМАЦІЙНІ РЕСУРСИ

1. Модульне середовище для навчання. Доступ до ресурсу: https://msn.khmnu.edu.ua/
2. Електронна бібліотека університету. Доступ до ресурсу: http://lib.khmnu.edu.ua/asp/php_f/p1age_lib.php
3. Репозитарій ХНУ. Доступ до ресурсу: http://elar.khmnu.edu.ua/jspui/

KHMELNYTSKYI NATIONAL UNIVERSITY

APPROVED
Dean of IT Faculty
_____ Savenko O.S.
"_1_" _____ 2023.

COURSE PROGRAM

Software Architecture and Design

Field of study: 12 - Information Technologies
Major: 121 – Software Engineering
Level of Higher Education: First Level (Bachelor)
Educational program: Software Engineering
Discipline status: Compulsory
Faculty: Information Technologies
Department: Software Engineering

Study mode	Year	Semester	Total Credits	Number of hours						Course project	Coursework	Semester control form	
				Classwork hours				Seminar classes	Independent work, including individual				
			ECTS credits	Total	Lectures	Laboratory works	Practical classes					pass/ fail test	Exam
Full-time (Daytime)	3	6	7	210	34	34			142	+			+
Total			**7**	**210**	**34**	**34**			**142**	**1**			**1**

The course program is based on the Higher Education Standard, the 2023 Bachelor's degree educational program, and the curriculum.

Program's author _____ Yu. Forkun _____ O. Yashyna

Approved at the staff meeting of the Software Engineering Department

Minutes from 31.08.2023 No. 1

Head of the Software Engineering Department _____ L. Bedratyuk

The course program is approved by the Academic Board of the Faculty of Information technologies

Head of the Academic Board _____ O.S. Savenko

Khmelnytskyi 2023

SOFTWARE ARCHITECTURE AND DESIGN

Type of discipline	Mandatory
Level of higher education	First (bachelor's)
Language of instruction	Ukrainian
Semesters	sixth
ECTS credits	7
Course study mode	Full-time (full-time)

Learning outcomes. According to the Standard of Higher Education and Educational Program, the discipline must ensure:

Integral Competence (IC): Ability to solve complex, specialised tasks or practical problems in software engineering, characterised by complexity and uncertainty of conditions, using information technology theories and methods.

Special (Professional) Competencies (PC): Ability to participate in software design, including modelling (formal description) of its structure, behaviour, and operational processes. Ability to develop architectures, modules, and components of software systems. Ability to accumulate, process, and systematise professional knowledge regarding the creation and maintenance of software and recognise the importance of lifelong learning. Ability to implement phases and iterations of the life cycle of software systems and information technologies based on relevant software development models and approaches. Ability to execute the system integration process and apply standards and change management procedures to maintain the integrity, overall functionality, and reliability of the software. Ability to reasonably choose and master the toolkit for software development and maintenance.

Program learning outcomes. To analyse, purposefully search for, and select the necessary information, reference resources, and knowledge for solving professional tasks, considering modern scientific and technical achievements. To understand the software lifecycle's leading processes, phases, and iterations. To conduct a pre-project survey of the subject area and system analysis of the design object. To select initial data for design, guided by formal methods of requirement descriptions and modelling. To apply effective software design approaches in practice. To use instrumental software tools in practice for domain analysis, design, testing, visualisation, measurement, and software documentation

The content of the discipline. Introduction to Application Architecture. Models, wireframes, and design samples. Types of architectures and their models. Architecture: Notation, Standards, and Tools. Architectural templates and styles. Quality control when choosing an architecture. Introduction to detailed design. Sequence diagrams and data flow diagrams in detailed design. Specification of algorithms, classes, and functions. Design samples. Libraries of standard templates. Standards, notation, and design tools. The Impact of Detailed Design on the Project.

Planned educational activities: lectures – 34 hours, laboratory classes – 34 hours, individual work – 142 hours, total – 210 hours.

Forms (methods) of training: lectures (using visualization methods); laboratory classes (using computer modeling methods), independent work

Forms of assessment of learning outcomes: defense of laboratory work, testing, exam, defense of a course project

Formi of semester control: exam, defense of the course project

Learning Resources:

1. Форкун Ю. В., Яшина О.М. Методичні вказівки до виконання курсового проекту з дисципліни «Архітектура та проектування програмного забезпечення» для студентів спеціальності 121 «Інженерія програмного забезпечення»/ Ю. В. Форкун, О.М.Яшина. – Хмельницький: ХНУ, 2023. – 42 с.
2. Конспект лекцій з дисципліни «Архітектура та проектування програмного забезпечення» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Інженерія програмного забезпечення» із спеціальності 121 – «Інженерія програмного забезпечення» / Укл. В.В.Завгородній, К.М.Ялова. – Кам'янське: ДДТУ, 2019.– 144с.
3. Мартін Р. Чиста архітектура /Роберт Мартін. – Харків: Фабула, 2019. – 416 с.
4. Фрімен Е. Head First. Патерни проєктування /Ерік Фрімен, Елізабет Робсон. – Харків: Фабула, 2020. – 672 с.

5. Richards M. Fundamentals of Software Architecture: An Engineering Approach / Mark Richards, NealFord. – Sebastopol, California: O'Reilly Media – 1st edition, 2020. – 419 p.
6. Lanciaux R. Modern Front-end Architecture: Optimize Your Front-end Development with Components, Storybook, and Mise en Place Philosophy / Ryan Lanciaux – New York: Apress, 2021 – 144 p.
7. Frighi V. Smart Architecture – A Sustainable Approach for Transparent Building ComponentsDesign / Valentina Frighi – Berlin: Springer – 1st edition, 2022. – 293 p.
8. Modular learning environment MOODLE. Access to the resource: https:/msn.khmnu.edu.ua
9. Electronic Library of the University Access to the resource: https:/lib.khnu.km.ua
10. KhNU Repository. Access to the resource: http://elar.khmnu.edu.ua/jspui/?locate=uk

Lecturers: Ph.D. Tech. Doctor of Physical Sciences, Associate Professor Yashina O.M., Ph.D. Tech. Doctor of Physical Sciences, Associate Professor Forkun Y.V.

3. EXPLANATORY NOTE

The discipline "Software Architecture and Design" is one of the special specialized disciplines that occupies a leading place in the training of bachelors in the specialty "Software Engineering"

The purpose of the discipline. Based on the study of the theoretical foundations of modeling, development and design based on software templates and frameworks, as well as the study of the basic standards, requirements and ready-made designs used in the development and design of large software applications, students should acquire practical skills in creating architecture and detailed design of software products using modern methods and tools.

Subject of discipline. Theory and practice of application of basic methods and tools for the development and design of software architecture.

The tasks of the discipline. To provide students with knowledge and practical skills in the basics of software architecture design and development.

Integral competence

Ability to solve complex, specialised tasks or practical problems in software engineering, characterised by complexity and uncertainty of conditions, using information technology theories and methods.

Special (Professional) Competencies (PC)

PC2. Ability to participate in software design, including modelling (formal description) of its structure, behaviour, and operational processes.

PC3. Ability to develop architectures, modules, and components of software systems..

PC10. Ability to accumulate, process, and systematise professional knowledge regarding the creation and maintenance of software and recognise the importance of lifelong learning.

PC11. Ability to implement phases and iterations of the life cycle of software systems and information technologies based on relevant software development models and approaches.

PC12. Ability to execute the system integration process and apply standards and change management procedures to maintain the integrity, overall functionality, and reliability of the software.

PC13. Ability to reasonably choose and master the toolkit for software development and maintenance.

Programmatic Learning Outcomes (PLO)

PLO 1 To analyse, purposefully search for, and select the necessary information, reference resources, and knowledge for solving professional tasks, considering modern scientific and technical achievements.

PLO3. To understand the software lifecycle's leading processes, phases, and iterations.

PLO10. To conduct a pre-project survey of the subject area and system analysis of the design object.

PLO11. To select initial data for design, guided by formal methods of requirement descriptions and modelling.

PLO12. To apply effective software design approaches in practice.

PLO14. To use instrumental software tools in practice for domain analysis, design, testing, visualisation, measurement, and software documentation.

Learning outcomes. A student who has successfully completed the course should understand the essential characteristics of the architecture of software applications and approaches to their analysis and design, the main types of architectures and their features, software design tasks; use the UML modeling language to design software applications; use standard tools and notations to document the architecture and interface of the software; possess software design tools and methods of structural and object-oriented software design; be able to use architectural design patterns and apply them when creating software applications; create software using typical architectural styles and solutions, taking into account the specifics of the application; design enterprise software using the typical architecture of corporate systems and platforms.

Discipline Policy. The organization of the educational process for the discipline complies with the requirements of the provisions on organizational and instructional-methodological support of the educational process, the educational program, and the curriculum. Students are required to attend lectures, practical classes, laboratory work, etc., according to the schedule, not to be late for classes, and to complete all tasks and checkpoints according to the schedule. Missed practical classes and laboratory work must be independently completed by the student in full and reported to the instructor no later than one week before the next assessment. For practical classes and laboratory work, students must prepare on the relevant topic and demonstrate active participation. Knowledge acquired by an individual in the discipline or its specific sections through informal education is credited according to the Regulation on the procedure for transferring learning outcomes and determining academic differences at KhNU.

4. STRUCTURE OF CREDIT CREDITS OF THE DISCIPLINE

Theme	Lecture	Lab. Rob.	By oneself. Rob. Full
Full-time (3 year, 2 semester)			
Topic 1. Introduction to Software Architecture	2	4	6
Topic 2. Models, wireframes and design samples	4		8
Topic 3. Types of Architectures and Their Models	2	4	12
Topic 4. Architecture: Notation, Standards, and Design Tools	2	8	12
Topic 5. Architectural templates and styles	2		12
Topic 6. Quality control when choosing an architecture	2		12
Topic 7. Introduction to Detailed Design	4	8	14
Topic 8. Sequence Diagrams and Data Flow Diagrams in Software Design	2	4	12
Topic 9. Specification of Algorithms, Classes, and Functions	4		14
Topic 10. Design Samples	4	4	14
Topic 11. Standard Template Libraries	4	2	14
Topic 12. The Impact of Detailed Design on the Project	2		12
In just one year:	**34**	**34**	**142**

5. PROGRAM OF THE DISCIPLINE
5.1. THE CONTENT OF THE LECTURE COURSE

№ Lecture	List of lecture topics	Quantity Hours
1	*2*	*3*
1	Introduction to Software Architecture. Overview of development technology. The concept of software architecture. Goals of architecture selection. Decomposition [1]; [2]; [3]; [4]; [5]	2
2	Models, wireframes and design samples. Use of models. Unified Modeling Language (UML). Wireframes. [1]; [2]; [3]	2
3	Models, wireframes and design samples. Classification of architectures. Design samples. Components.[1]; [4]	2
4	Types of architectures and their models. Architectures based on data flows. Independent components. Virtual machines. Repository architectures. Layered architectures. Mixed-architecture applications. Architecture selection procedure [1]; [2] ; [4]; [5]	2
5.	Architecture: Notation, Standards, and Tools. Notation. Tools. IEEE/ANSI standard for project description [1]; [2] ; [3]; [4]	2
6.	Architectural templates and styles. Main architectural styles. Combination of architectural styles. Client/server architecture. Component architecture. Domain-based design. Multi-layered architecture. Object-oriented architecture. Message bus-based architecture. Service-oriented architecture. [1]; [2]; [3] ; [4]; [6]	2
7	Quality control when choosing an architecture. Quality and choice of architecture. Choose from alternative architectures. Validate the architecture with use cases. Inspecting the choice of architecture. The Impact of Architecture Choice on SPMP. [1]; [2]; [3]; [6]	2
8	Introduction to detailed design. The concept of detailed design. Correlation of use cases, architecture, and detailed design. A typical diagram of the detailed design process. [1]; [2] ; [3] ; [4]; [5]	2
9	Introduction to detailed design. USDP design – requirements, analysis, design, implementation, testing. Design and interfaces. Reusable items. [1]; [2] ; [3] ; [5]	2
10	Sequence diagrams and data flow diagrams in detailed design. Detailed sequence diagrams. Detailed data flow diagrams [1]; [2] ; [3] ; [4]; [6]	2
11	Specification of algorithms, classes, and functions. Class invariants. Invariants, prerequisites and afterconditions of functions. Flowcharts. Pseudocode. Use of flowcharts and pseudocode [1]; [2]; [3]; [4]	2
12	Specification of algorithms, classes, and functions. Class invariants. Invariants, prerequisites and afterconditions of functions. Flowcharts. Pseudocode. Using flowcharts and pseudocode [1];[2]; [3]; [4];] [5]; [6]	2
13	Design samples. Detailed design techniques. Creation Design Patterns. Structural design samples. Design samples based on the behavior of the application [1];[2]; [3]; [4]; [6]	2
14	Design samples. Patterns of behavior. Anti-patterns of design [3]; [4]; [6]	2
15	Libraries of standard templates. Standards, notation, and design tools. C++ Standard Template Library (STL). [3]; [4]; [6]	2

16	Libraries of standard templates. IEEE 890 standard. UML language. Tools that use source code: Javadoc [1] ; [3] ; [4]; [6]	2
17	The impact of detailed design on the project. Estimation of the scope of work with the help of detailed design. Quality and metrics in detailed design. Inspection for detailed design [1]; [2] ; [3] ; [4]; [5]	2
	Total per semester:	34

5.2 CONTENT OF LABORATORY CLASSES

№ Lab. Classes	Topics of laboratory classes and their content	Quantity Hours
1	*2*	*3*
1.	Laboratory work No1 Evaluation and selection of software components	4
2.	Laboratory work No2 Use of different types of software architectures. Design Patterns	4
3.	Laboratory work No3 Work of applications with mixed architecture	4
4.	Laboratory work No4 Use of standards and tools in the development of software architecture	4
5.	Laboratory work No5 Initial stage of design. Subject Area Research	4
6.	Laboratory work No: Software System Architecture Development: Choosing the Type of Architecture and Describing the Decomposition	4
7.	Laboratory work No7 Development of software system architecture: description of dependencies and description of the interface	4
8.	Laboratory work No8 DReference design of a software system: modules and data	6
	Just:	34

5.3 CONTENT OF INDEPENDENT WORK

Independent work of ***full-time*** students consists in the systematic study of program material, preparation for the performance and defense of laboratory work, testing on theoretical material, performance of individual tasks, including a course project, etc.

Topic Number	**The content of independent work**	**Number of hours**
1	*2*	*3*
1	Elaboration of lecture material, preparation for laboratory work No 1.	8
2	Processing of lecture material, preparation for the defense of laboratory work No 1.	8
3	Processing of lecture material, preparation for laboratory work No 2.	8
4	Processing of lecture material, preparation for the defense of laboratory work No 2.	8
5	Elaboration of lecture material, preparation for laboratory work No 3.	8
6	Elaboration of lecture material, preparation for the defense of laboratory work No 3.	8
7	Processing of lecture material, preparation for laboratory work No 4.	8
8	Elaboration of lecture material, preparation for the defense of laboratory work No 4.	8
9	Processing of lecture material, preparation for laboratory work No 5.	8
10	Processing of lecture material, preparation for the defense of laboratory work No 5.	10
11	Processing of lecture material, preparation for laboratory work No 6.	8
12	Elaboration of lecture material, preparation for the defense of laboratory work No 6.	10
13	Processing of lecture material, preparation for laboratory work No 7.	8
14	Elaboration of lecture material, preparation for the defense of laboratory work No 7.	10
15	Processing of lecture material, preparation for laboratory work No 8.	8
16	Processing of lecture material, preparation for laboratory work No 8.	8
17	Processing of lecture material, preparation for the defense of laboratory work No 8.	8
	Total per semester	**142**

5.4 COURSE DESIGN

For individual work of full-time students in the 6th semester, it is planned to complete a course project in the discipline "Software Architecture and Design", which is defended in the sixteenth week of the VI semester for full-time education.

The purpose of course design is to deepen and consolidate the theoretical knowledge gained by the student in the study of this discipline, to apply it for a comprehensive solution of issues that arise in the development of architecture and software design.

In the process of working on a course project, the student must show initiative and independence in substantiating and choosing constructive and technological solutions that will be used in the design of a software product, using knowledge and practical skills in the disciplines studied earlier, in particular object-oriented programming, software modeling, object-oriented analysis and design, analysis of software requirements.

The approximate volume of the explanatory note (with attachments) is from 30 to 50 pages and the graphic part. When completing a course project, students use the appropriate methodological recommendations. The topics of course projects are developed by the supervisor and approved at a meeting of the department.

Consultations on the course project are carried out by the project manager - a teacher of the department in accordance with the schedule approved at the meeting of the department. The defense of the course project is carried out after its verification by the supervisor before the commission, which consists of 2-3 teachers of the department. The commission must include a project manager.

Evaluation criteria. Evaluation of the course project is carried out by types of work according to the distribution and weighting coefficients. The grade "excellent/A" is given for a high-quality software product and an explanatory note without errors, originality of the software product, compliance with the requirements of methodological recommendations for course design. The report and defense of the work are substantiated, comprehensive knowledge of special disciplines regarding the topic of the course project is revealed.

The grade "good/B" is given for the quality of the course project with one or two minor errors or insufficiently confident answers to one or two questions of the commission. The grade "good/C" is given for a well-executed project, compliance with all the requirements for the course project, for two or three minor mistakes in the implementation of the task or explanatory note, unclear answers to two or three questions of the commission.

A grade of "satisfactory/D" is given if errors are found in the explanatory note or software product, there are minor violations of the requirements for the design of the project, uncertain answers to the main questions on the topic of the project.

The grade "satisfactory/E" is given if the project contains significant errors both in the explanatory note and in the developed product, incorrectly justified design decisions, gross errors in answers to questions from commission members, uncertain defense in general.

A grade of "unsatisfactory/FX/F" is given if incorrect design methods are chosen in the project, or for the inconsistency of the content and listing of the program with the approved topic of the course project, when the student is not oriented in what he has done. In this case, the student submits the corrected work for re-defense, or he is given a new topic of the project and is assigned a deadline for its implementation and defense.

Approximate topics of course projects

1. Development of architecture and components of software and information support of the information system for supporting decision-making on the formation of production and operational strategy of enterprises.

2. Development of the architecture and components of the task planning system using algorithms of economic models.

3. Development of the architecture and components of the software package for the analysis of biometric signals on the example of the keyboard handwriting analysis system.

4. Development of the architecture and components of the intelligent agent based on the model of a viable system.

5. Development of the architecture and components of the portal security system.
6. Development of the architecture and components of the information security subsystem.
7. Development of architecture and components of a distributed search engine.
8. Development of architecture and components of tools to support systems for solving problems.
9. Development of the architecture and components of the gesture recognition system.
10. Development of architecture and software components for the electronic library.
11. Development of architecture and components of an automated testing system.

In the individual assignment for course design, the design features of the created software product and their step-by-step implementation are specified. The assignment is given to the student in the first week of the semester, and the defense of the coursework is given in the 16th week. Students carry out the course project in accordance with the methodological guidelines for the course design in the discipline "Software Architecture and Design"

6. TEACHING METHODS

The learning process in the discipline is based on the use of traditional and modern methods. In particular, lectures are conducted mainly by verbal methods, and laboratory classes are conducted using information technology, master classes, workshops and are aimed at acquiring practical skills in designing and developing software architecture according to modern methods, using special design tools, etc.

7. FORMS AND METHODS OF ASSESSMENT OF LEARNING OUTCOMES

Each type of work in the discipline is evaluated on a four-point scale. The semester final grade is determined as the weighted average of all types of academic work, completed and passed positively, taking into account the weight coefficient. Weighting coefficients vary depending on the structure of the discipline and the importance of its individual types of work. A student who has scored a positive weighted average score for the current work and has not passed the final control measure (exam) is considered to be underachieved.

When assessing students' knowledge, various means of control are used, in particular: oral examination before admission to laboratory work is carried out at its beginning; assimilation of theoretical material on topics is checked by test control; The quality of performance, acquisition of theoretical knowledge and practical skills is checked by defending each laboratory work and individual task in accordance with the work program of the discipline and the working curriculum.

The grade given for the laboratory lesson consists of the following elements: oral questioning of students before admission to laboratory work; knowledge of theoretical material on the topic; the quality of the protocol and graphic part; the student's ability to justify the constructive decisions made.

The student must work out the missed laboratory lesson in the laboratories of the department no later than two weeks before the end of theoretical classes in the semester.

When assessing students' knowledge, the teacher is guided by the following criteria.

An "excellent" grade is given to a student who has deeply mastered the material and is able to apply it rationally, knows the techniques and knows how to use them in the design and development of software architecture. An excellent grade involves a competent, logical presentation of the answer (both orally and in writing), high-quality external design. The student must acquire practical skills in the development and design of various types of software. The student should not hesitate when modifying the question, must draw detailed and generalizing conclusions.

The grade "good" is given to the student for the complete assimilation of the educational material, mastery of the conceptual apparatus, orientation in the studied material, conscious use of knowledge to solve practical problems, competent presentation of the answer, but in the content and form of the answer there were some inaccuracies (errors), vague formulations of patterns, etc. The student's answer should be based on independent thinking.

A student receives a "good" mark for a correct answer with two or three significant errors.

A student deserves a "satisfactory" mark for incomplete mastery of the program material, but the acquired knowledge and practical skills in software development and design meet the minimum evaluation criteria.

The grade "unsatisfactory" is given when the student has scattered, unsystematic knowledge, does not know how to distinguish the main and secondary, makes mistakes in the definition of concepts, distorts their content, chaotically and hesitantly presents the material, cannot use knowledge in solving practical problems. As a rule, the grade "unsatisfactory" is given to a student who cannot continue his studies without additional knowledge from the course.

Based on the results of the current control and the final control measure, the final semester grade is set. Based on the analysis of knowledge control, the teacher improves the course of lectures, paying special attention to those sections or topics from which there were the most inaccurate answers, which indicates methodological or other shortcomings in the coverage of these topics or sections.

Similarly, adjustments are made to the methodological manuals for laboratory practicum and course design, the fundamental issues of software design during laboratory work, course projects and their defense are considered in more detail.

Structuring the discipline by types of work and assessing the learning outcomes of students by weighting coefficients

Classroom work								**Independent work**	**Final control**
Laboratory work								Test control	EXAM
1	2	3	4	5	6	7	7	1	1
0,4								0,2	0,4
Semester VI (course project)									
1 розділ		2 розділ		3 розділ				Graphic part	Defense of the course project
0,2		0,2		0,25				0,15	0,2

Transition from the national assessment scale to the European one (ECTS)

ECTS assessment	Points	Domestic system	
And	4,75-5,00	5	EXCELLENT - deep and complete mastery of the educational material and identification of relevant skills and abilities
Into	4,25-4,74	4	GOOD – complete knowledge of the study material with a few minor errors
C	3,75-4,24	4	GOOD - generally correct answer with two or three significant errors
D	3,25-3,74	3	SATISFACTORY - incomplete mastery of the program material, but sufficient for practical activity in the profession
E	3,00-3,24	3	SATISFACTORY - incomplete mastery of the program material that satisfies the minimum evaluation criteria
FX	2,00-2,99	2	UNSATISFACTORY - unsystematic knowledge and inability to continue training without additional knowledge of the discipline
F	0,00-1,99	2	UNSATISFACTORY - serious further work and re-study of the discipline are required

8. SELF-ASSESSMENT QUESTIONS

1. The concept of system development.
2. The concept of software.
3. The concept of creating an architecture.
4. Architecture selection goals.
5. The concept of decomposition.
6. The concept of detailed design.
7. Use of models.
8. Frameworks.
9. Classification of architectures.
10. Design samples.
11. Components.
12. Types of architectures and their models.
13. Architectures are based on data flows.
14. Independent components.
15. Virtual machines.
16. Repository architectures
17. Tiered architectures.
18. Client-server architecture.
19. The architecture is domain-based.
20. Architecture selection procedure.
21. Architecture selection tools.
22. Quality control when choosing an architecture.
23. Validate the architecture with use cases.
24. Inspecting the choice of architecture.
25. The concept of detailed design.
26. Correlation of use cases for architecture and detailed design.
27. Typical detailed design scheme.
28. Design and interfaces.
29. Reusable components.
30. Detailed sequence charts.
31. Detailed diagrams of data flows.
32. Specification of classes and functions.
33. Class invariants.
34. Invariants, prerequisites and afterconditions.
35. Specification of algorithms.
36. Flowcharts.
37. Pseudocode.
38. Comparison of the use of flowchart and pseudocode.
39. Design Samples: Detailed Design Techniques.
40. Creation Patterns of Design.
41. Structural design samples.

42. Design samples are based on application behavior.
43. What is a design pattern?
44. What types of design patterns are you familiar with?
45. Design template "Strategy".
46. Design template "Builder".
47. Design template "Factory", "Abstract factory".
48. The Adapter design pattern
49. The "Bridge" design pattern.
50. The "Team" design pattern
51. The Composer design pattern
52. Iterator design pattern.
53. The "Mediator" design template
54. Chain of Responsibility design pattern
55. The Browser design pattern
56. The "Intermediary" design pattern
57. The "Loner" design pattern
58. The "State" design pattern
59. Design pattern "Sample".
60. What is Software Design Quality Analysis?
61. Basic Software Design Evaluations.
62. What notations and design support tools are known?
63. Methods of Analysis of Compromise Architectural Solutions.
64. The main stages of the method of analysis of compromise architectural solutions.
65. A method for analyzing the cost and efficiency of architectural solutions and software applications.

9. METHODOLOGICAL SUPPORT

The educational process in the discipline "Software Architecture and Design" is fully and in sufficient quantities provided with the necessary educational and methodological literature. All the necessary teaching materials are placed in the Modular Learning Environment. Access to the resource: https://msn.khmnu.edu.ua/course/view.php?id=4908.

10. RECOMMENDED READING

Main:

1. Lecture notes on the discipline "Software Architecture and Design" for applicants for higher education of the first (bachelor's) level in the educational and professional program "Software Engineering" in the specialty 121 – "Software Engineering" / Ucl. V.V.Zavgorodniy, K.M.Yalova. – Kamianske: DGTU, 2019.– 144 p.
2. Martin R. Pure Architecture / Robert Martin. – Kharkiv: Fabula, 2019. – 416 c.
3. Freeman E. Head First. Design patterns / Eric Freeman, Elizabeth Robson. – Kharkiv: Fabula, 2020. – 672 c.
4. Richards M. Fundamentals of Software Architecture: An Engineering Approach / Mark Richards, NealFord. – Sebastopol, California: O'Reilly Media – 1st edition, 2020. – 419 p.
5. Lanciaux R. Modern Front-end Architecture: Optimize Your Front-end Development with Components, Storybook, and Mise en Place Philosophy / Ryan Lanciaux – New York: Apress, 2021 – 144 p.
6. Frighi V. Smart Architecture – A Sustainable Approach for Transparent Building ComponentsDesign / Valentina Frighi – Berlin: Springer – 1st edition, 2022. – 293 p.
7. Kudryavtsev, V. V., and Forkun, Y. V. "Analysis and application of methods for optimizing speed and fault tolerance of software products." Sciences. etc. scientific conference "APKN-89 2021". Khmelnitsky KhNU. – 2021. – p.338-339
8. Forkun, Y., Forkun, I., Yashina, O., & Pravorska, N. (2023). Architectural Methods for Optimizing the Performance and Fault Tolerance of Software Applications. measuring and computing devices in technological processes, (2), 196–201. https://doi.org/10.31891/2219-9365-2023-74-27
9. Ivan Lopatto, Mykyta Lebiga, Yurii Forkun and Artem Boyarchuk. Method for Determining the Informativeness of the Software Requirements Specifications. IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine. http://ceur-ws.org/Vol-2853/paper15.pdf

Additional

1. Borodkina, I.L. Software Engineering. manual / I. L. Borodkina, G. O. Borodkin; NUBiP. — Kyiv: Tsentr uchbovoi lit., 2020. 204 p. (in Russian).
2. Lanciaux R. Modern Front-end Architecture: Optimize Your Front-end Development with Components, Storybook, and Mise en Place Philosophy / Ryan Lanciaux – New York: Apress, 2021 – 144 p.
3. Frighi V. Smart Architecture – A Sustainable Approach for Transparent Building ComponentsDesign / Valentina Frighi – Berlin: Springer – 1st edition, 2022. – 293 p.
4. Robert Martin. Clean architecture. Kharkiv: Fabula. – 2019. – 368 p. (in Russian). ISBN : 978-617-09-5286-8
5. Robert Martin. Clean code. Kharkiv: Fabula. – 2019. – 416 p. (in Russian). ISBN : 978-617-09-5285-1
6. Alexey Vasiliev. Python programming. Ternopil: EDUCATIONAL BOOK – BOGDAN. - 2019. – 504 p. ISBN : 9789661056113
7. Elizabeth Robson, Eric Freeman. Head First. Design patterns. Kharkiv: Fabula. – 2020. – 672 p. (in Russian). ISBN : 978-617-09-6159-4
8. Elizabeth Robson, Eric Freeman. Head First. JavaScript programming. Kharkiv: Fabula. – 2022. – 672 p. (in Russian). ISBN : 978-617-522-047-4
9. Paul Berry. Head First. Python. Kharkiv: Fabula. – 2021. – 624 p. (in Russian). ISBN : 978-617-522-

019-1
10. Rob Cole. Brilliant Agile. Kharkiv: Fabula. – 2020. – 192 p. (in Russian). ISBN : 978-617-09-6381-9
11. Burt Bates, Kathy Sierra. Head First. Java. Kharkiv: Fabula. – 2022. – 720 p. ISBN : 978-617-522-033-7
12. Yuriy Ramsky, Vasyl Oleksyuk, Anatoly. Administration of Computer Networks and Systems. Ternopil: EDUCATIONAL BOOK – BOGDAN. - 2020. – 196 p. ISBN : 9789661015615
13. Amy Webb. The Big Nine. How IT Giants and Their Smart Machines Can Change Humanity. Kharkiv: Vivat. – 2020. – 352 p. ISBN : 9789669822185
14. Denis Kaplunov. Kings of social media. .Kyiv: BookChef – 2022. – 432 p. ISBN: 9786175480922

11. INFORMATION RESOURCES

1. Modular learning environment. Access to the resource: https://msn.khmnu.edu.ua/
2. Electronic Library of the University. Access to the resource: http://lib.khmnu.edu.ua/asp/php_f/p1age_lib.php
3. Repository of KhNU. Access to the resource: http://elar.khmnu.edu.ua/jspui/