Черкаський національний університет імені Богдана Хмельницького
ННІ ІНФОТЕХ
Кафедра прикладної математики

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ «АРХІТЕКТУРА ТА РОЗРОБКА ВЕБ-ДОДАТКІВ»

1. Загальна інформація про курс

Мова викладання	Курс викладається українською мовою
Статус дисципліни	Обов'язкова
Викладачі	Дідковський Р.М., д.т.н., доцент, доцент кафедри прикладної математики та інформатики
Код класу /	
Корпоративна пошта / E-mail:	didkovskyirm@vu.cdu.edu.ua
Затвердження та перегляд робочої навчальної програми	Розглянуто та затверджено на засіданні кафедри прикладної математики та інформатики 27.08.2025, протокол № 1

2. Анотація до курсу

Навчальна дисципліна «Архітектура та розробка веб-додатків» є курсом циклу професійної та практичної підготовки фахівця з прикладної математики. Навчальна дисципліна є теоретичною та практичною основою сукупності знань та вмінь, що формують у фахівця в галузі інформаційних технологій навички розробки програмних додатків, орієнтованих на роботу у глобальній мережі Інтернет.

Вивчення навчальної дисципліни рекомендується планувати, починаючи з другого семестру.

3. Мета та цілі курсу

Метою курсу є формування у студентів теоретичних знань та практичних навичок з розробки веб-орієнтованих додатків мовою програмування GoLang з використанням спеціалізованих бібліотек, фреймворків та інструментів, таких як: Vue.js, Docker, Swagger тощо.

Завданнями вивчення навчальної дисципліни «Архітектура та розробка веб-додатків» передбачено:

- набуття теоретичних знань з основ веб-технологій та клієнт-серверної архітектури, архітектурних патернів веб-додатків, мікросервісної архітектура та монолітних систем, використання баз даних у веб-додатках та шаблонів доступу до даних, організації масштабованості, продуктивності та безпеки
- набуття та систематизація студентами знань і практичних навичок з розробки програм мовою програмування GoLang;
- набуття та систематизація студентами знань і практичних навичок з розгортання HTTP-серверів та роутингу в GoLang;
- набуття та систематизація студентами знань і практичних навичок з розробки REST-додатків;
- набуття та систематизація студентами знань і практичних навичок з розробки фронт-енду з використанням бібліотеки Vue.js;
- набуття та систематизація студентами знань і практичних навичок з використання спеціалізованих інструментів для організації неперервної інтеграції та неперервної доставки розроблених додатків на сервер;
- формування відповідних компетентностей, необхідних для виконання вказаних вище задач.

4. Компетентності та очікувані результати навчання

Навчальна дисципліна «Архітектура та розробка веб-додатків» забезпечує формування таких компетентностей, передбачених освітньою програмою підготовки магістрів спеціальності F1 Прикладна математика:

Загальні компетентності:

ЗК03. Здатність оволодівати сучасними знаннями, формулювати та вирішувати проблеми.
ЗК06. Здатність працювати в команді та керувати нею.
ЗК07. Здатність спілкуватися та здійснювати професійну діяльність державною мовою та мовою країн ЄС.

Фахові компетентності *(визначені стандартом та освітньою програмою компетентності, формування яких забезпечує ця навчальна дисципліна)*

СК01. Здатність розв'язувати задачі й проблеми, які можуть бути формалізовані, потребують оновлення й інтеграції знань, зокрема в умовах неповної інформації.

СК07. Здатність проєктувати та розробляти програмне забезпечення для розв'язування формалізованих задач, зокрема систем з великими обсягами даних.

СК09. Здатність до аналізу та моделювання процесів шляхом розробки мережевих застосунків, практичного впровадження застосунків для реалізації функціональних моделей організаційно-економічних і виробничо-технічних систем, задач бізнесу, екологічного моніторингу та інших актуальних сучасних задач.

Згідно з вимогами освітньо-професійної програми, **програмними результатами вивчення** дисципліни «Архітектура та розробка веб-додатків» є такі:

РН03. Логічно, послідовно й точно формулювати свої думки та подавати інформацію у професійному спілкуванні, застосовувати інформаційні і технічні засоби та педагогічні методи для презентації результатів наукових, прикладних й ІТ-проєктів.

РН04. Будувати математичні моделі складних систем в області інформації, природничих наук, медицині тощо і вибирати методи їх дослідження, реалізовувати побудовані моделі програмно та перевіряти їх адекватність за допомогою комп'ютерних технологій.

РН07. Розв'язувати задачі комп'ютерного моделювання шляхом використання і розробки сучасних програмних засобів, зокрема методами розподіленого, паралельного та хмарного програмування.

РН08. Розробляти та програмно реалізовувати алгоритми розв'язування прикладних задач, прикладне програмне забезпечення інформаційних систем і технологій.

РН11. Розробляти на основі структури математичної моделі та алгоритмів функціонування процесів, що моделюються, програмне забезпечення, зокрема – веб-орієнтоване, із застосуванням сучасних технологій програмування та систем комп'ютерної математики, аналізувати отримані результати на адекватність.

5. **Обсяг і характеристика курсу**

Найменування показників	**Характеристика навчального курсу**
	денна форма навчання
Освітній ступінь	магістр
Спеціальність, освітня програма	F1 Прикладна математика, Прикладна математика
Рік (курс) навчання	1, 2
Семестр вивчення	2, 3
Кількість кредитів ЄКТС	9
Загальний обсяг годин	270
Кількість годин навчальних занять	90
Лекційні заняття	30
Лабораторні заняття	60
Самостійна та індивідуальна робота	180
Форма підсумкового контролю	екзамен

6. **Пререквізити курсу**

Для вивчення курсу студенти не потребують базових знань з навчальних дисциплін, що викладаються на освітньому ступені магістра, але повинні мати базові знання з алгоритмізації та програмування, мов програмування, об’єктно-орієнтованого програмування, ОС, операційної системи Linux та її адміністрування, комп’ютерних мереж, що викладаються на освітньому ступені бакалавра.

7. **Технічне забезпечення**

Для вивчення курсу потрібно встановити набір для розробки програм мовою GoLang, що вільно розповсюджуваний, інструментальне середовище на зразок вільно розповсюджуваного Visual Studio Code чи Sublime Text, операційну систему Linux на фізичній чи віртуальній машині, програмне забезпечення для створення та запуску веб-сервера. Для виконання домашніх завдань додатково потрібні загальновживані офісні програми та онлайн-сервіси.

8. **Політика курсу**

Академічна доброчесність. Очікується, що роботи студентів будуть їх оригінальними розробками. Фабрикування результатів, списування, втручання у роботу інших студентів можуть бути кваліфіковані як академічна недоброчесність. Виявлення ознак академічної недоброчесності у вихідному коді студента є підставою для незарахування домашнього завдання викладачем, незалежно від масштабів плагіату.

Відвідування занять. Відвідування занять є важливою складовою навчання. Очікується, що всі студенти відвідають усі лекції і практичні заняття курсу. Студенти мають інформувати викладача про неможливість відвідати заняття (за наявності поважної причини), у такому випадку допускається підключення студентів онлайн і їх дистанційна робота. Допускається по 2 пропуски лекцій та лабораторних робіт з поважних причин, які не впливатимуть на систему оцінювання. Студенти зобов'язані дотримуватися усіх строків, визначених для виконання усіх видів робіт, передбачених курсом.

9. **Схема курсу**

Тема, основні питання / завдання	**Розподіл годин за темами та формами занять**	**Форми та методи проведення**	**Література. Ресурси в інтернеті**	**Завдання для самостійної роботи, год**	**Форма контролю, бали**
ЗМІСТОВИЙ МОДУЛЬ 1. АРХІТЕКТУРА ВЕБ-ДОДАТКІВ					
Лекція 1. Основи веб-технологій та архітектури клієнт-сервер 1. Історія та еволюція веб-технологій 2. Протокол HTTP/HTTPS 3. Архітектура клієнт-сервер 4. Типи веб-додатків 5. Тришарова архітектура 6. Компоненти веб-інфраструктури	2	Лекція-візуалізація (з використанням презентації)	1, 3, 31, 42, 43, 44, 46	1. Опрацювання літератури з теми лекції (4 год.)	
Лабораторне заняття 1-2. Аналіз архітектури існуючих веб-додатків ***Мета роботи:*** Навчитися аналізувати та	4	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного	1, 3, 31, 42, 43, 44, 46	1. Вибір веб-додатків для аналізу (обрати 2-3 з різних категорій) 2. Аналіз публічної інформації 3. Дослідження архітектурних аспектів	Звіт, 0-12 балів

документувати архітектуру реальних веб-додатків, розуміти архітектурні рішення та їх обґрунтування		забезпечення, вихідний код програм, діаграми, схеми		4. Створення архітектурних діаграм 5. Аналіз архітектурних рішень 6. Підготовка звіту (8 год.)	
Лекція 2. Архітектурні патерни веб-додатків 1. Model-View-Controller (MVC) 2. Model-View-Presenter (MVP) 3. Model-View-ViewModel (MVVM) 4. Layered (N-Tier) Architecture 5. Hexagonal Architecture (Ports & Adapters) 6. Clean Architecture 7. Порівняння патернів	2	Лекція-візуалізація (з використанням презентації)	4, 5, 23, 24, 45	1. Опрацювання літератури з теми лекції (2 год.)	
Лабораторне заняття 3-4. Проектування архітектури веб-додатку ***Мета роботи:*** Розробити повну архітектуру веб-додатку, застосувавши вивчені архітектурні патерни та принципи	4	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	4, 5, 8, 19, 20, 23, 24, 27, 45	1. Вибір предметної області 2. Збір та аналіз вимог 3. Вибір архітектурного стилю 4. Планування технологічного стеку 5. Створення UML діаграм 6. Архітектурні представлення (модель 4+1) 7. Сценарії атрибутів якості 8. Технологічні рішення 9. Підготовка архітектурної документації (8 год.)	Звіт, 0-12 балів
Лекція 3. Мікросервісна архітектура та монолітні системи 1. Монолітна архітектура 2. Мікросервісна архітектура 3. Domain-Driven Design (DDD) 4. Міжсервісна комунікація 5. Інфраструктурні патерни 6. Service Discovery та Configuration 7. Проблеми мікросервісів	2	Лекція-візуалізація (з використанням презентації)	6, 7, 10, 16, 48	1. Опрацювання літератури з теми лекції (2 год.)	
Лабораторне заняття 5-6. Проектування RESTful API	4	Практична робота. Матеріали: ознайомчі та демонстраційні	1, 6, 7, 10, 16, 29, 30, 31, 48	1. Фундаментальні принципи REST 2. Проектування ресурсів 3. Визначення endpoints	Звіт, 0-12 балів

Мета роботи: Спроектувати повноцінний RESTful API для веб-додатку, дотримуючись кращих практик та стандартів		версії програмного забезпечення, вихідний код програм, діаграми, схеми		4. HTTP методи та семантика 5. Request/Response design 6. Створення OpenAPI (Swagger) специфікації 7. Додаткові функції API 8. Реалізація автентифікації та авторизації 9. Документація 10. Тестування API (8 год.)	
Лекція 4. Бази даних та шаблони доступу до даних 1. Типи баз даних 2. SQL та NoSQL 3. Шаблони доступу до даних 4. Порівняння ORM, Query Builder та Raw SQL 5. Кращі методики розробки баз даних 6. Масштабування баз даних 7. Транзакції та ізоляція	2	Лекція-візуалізація (з використанням презентації)	2, 17, 21, 22	1. Опрацювання літератури з теми лекції (2 год.)	
Лабораторне заняття 7-8. Проектування бази даних ***Мета роботи:*** Спроектувати ефективну схему бази даних для веб-додатку з урахуванням нормалізації, індексування та продуктивності	4	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	2, 17, 21, 22	1. Аналіз предметної області 2. Створення концептуальної моделі 3. Логічна модель даних 4. Нормалізація 5. Вибір типу бази даних 6. Фізична модель БД (для SQL) 7. Індексування 8. Партиціонування (для великих таблиць) 9. Оптимізація запитів 10. Транзакції та цілісність 11. Стратегія перенесення даних 12. Розробка схеми NoSQL БД (якщо обрано NoSQL) (8 год.)	Звіт, 0-12 балів
Лекція 5. Масштабованість, продуктивність та безпека 1. Принципи масштабованості	2	Лекція-візуалізація (з використанням презентації)	3, 8, 18, 25, 26, 30, 42, 43, 44, 47	1. Опрацювання літератури з теми лекції (2 год.)	

2. Балансування завантаження 3. Стратегії кешування 4. Продуктивність веб-додатків 5. Обмеження швидкості та регулювання 6. Безпека веб-додатків 7. Автентифікація та авторизація 8. Захист від атак 9. HTTPS та криптографія					
Лабораторне заняття 9-10. Документування архітектури системи ***Мета роботи:*** Створити повну архітектурну документацію веб-додатку, використовуючи сучасні підходи та нотації	4	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	3, 8, 9, 18, 26, 28, 42, 43, 44	1. Модель C4 - Рівень 1: Системний контекст 2. Модель C4 - Рівень 2: Діаграма контейнерів 3. Модель C4 - Рівень 3: Діаграми компонентів 4. Модель C4 - Рівень 4: Код (опціонально) 5. Записи архітектурних рішень (ADR) 6. Атрибути якості 7. Нефункціональні вимоги 8. Технологічний радар 9. Архітектура розгортання 10. Моніторинг та спостережуваність 11. Архітектура безпеки 12. Архітектура даних 13. Архітектура інтеграції 14. Настанови з розробки 15. Ризики та технічний борг 16. Глосарій (8 год.)	Звіт, 0-12 балів
Всього балів за змістовим модулем 1					**60**
Лекцій	**10**				
Лабораторних занять	**20**				
Самостійна робота	**60**				
Підсумковий контроль: екзамен		Тестування			**40**
ЗМІСТОВИЙ МОДУЛЬ 2. BACKEND-РОЗРОБКА З GO					
Лекція 1. Основи Go та налаштування	2	Лекція-візуалізація (з	11, 12, 13	1. Встановити Go на локальну машину та	Звіт, 0-2 бали

середовища 1. Історія та філософія мови Go 2. Встановлення Go SDK та налаштування GOPATH/Go modules 3. Базовий синтаксис: змінні, константи, типи даних 4. Функції, методи та інтерфейси 5. Пакети та організація коду 6. Goroutines та concurrency 7. Channels та синхронізація 8. Обробка помилок в Go		використанням презентації)		налаштувати IDE (VS Code/GoLand) 2. Написати програму для обчислення чисел Фібоначчі з використанням goroutines 3. Створити власний пакет з утилітарними функціями 4. Реалізувати producer-consumer pattern з використанням channels 5. Вивчити документацію стандартної бібліотеки Go (4 год.)	
Лекція 2. HTTP-сервери та роутинг в Go 1. Пакет net/http: основи веб-серверів 2. Handler та HandlerFunc 3. Multiplexer та роутинг 4. Фреймворк Gin: установка та базові концепції 5. Групування роутів та параметри URL 6. Query parameters та path variables 7. Обробка різних HTTP методів (GET, POST, PUT, DELETE) 8. Структурування веб-додатку	2	Лекція-візуалізація (з використанням презентації)	11, 12, 13, 32	1. Створити простий HTTP-сервер без фреймворків 2. Переписати сервер з використанням Gin 3. Реалізувати роутинг для blog API (статті, коментарі) 4. Додати обробку JSON запитів та відповідей 5. Порівняти продуктивність різних роутерів (benchmark) (4 год.)	Звіт, 0-2 бали
Лекція 3. Робота з базами даних 1. SQL та NoSQL: вибір БД для проєкту 2. Пакет database/sql та драйвери 3. Connection pooling та управління з'єднаннями 4. GORM: ORM для Go 5. Визначення моделей та міграції 6. CRUD операції через ORM 7. Відносини між таблицями (one-to-one, one-to-many, many-to-many) 8. Транзакції та оптимізація запитів	2	Лекція-візуалізація (з використанням презентації)	11, 12, 33	1. Встановити PostgreSQL та підключитися з Go 2. Створити схему БД для e-commerce системи 3. Реалізувати моделі з використанням GORM 4. Написати міграції для створення таблиць 5. Виконати складні запити з JOIN операціями (4 год.)	Звіт, 0-2 бали
Лекція 4. RESTful API та middleware	2	Лекція-візуалізація (з використанням	11, 12, 34, 35	1. Спроектувати REST API для системи управління завданнями	Звіт, 0-2 бали

1. Принципи REST архітектури 2. Проектування RESTful endpoints 3. HTTP статус-коди та їх використання 4. Middleware: концепція та застосування 5. Логування запитів 6. CORS та його налаштування 7. JWT автентифікація та авторизація 8. Обмеження швидкості та найкращі практики безпеки		презентації)		2. Реалізувати middleware для логування 3. Додати JWT автентифікацію 4. Створити middleware для перевірки ролей користувачів 5. Документувати API з використанням Swagger (4 год.)	
Лекція 5. Тестування та розгортання 1. Тестування в Go: пакет testing 2. Модульні тести для функцій та методів 3. Табличні тести 4. Імітація та тестування з БД 5. Інтеграційні тести для API 6. Docker: контейнеризація Go додатків 7. Docker Compose для multi-container setup 8. CI/CD конвеєр з GitHub Actions 9. Стратегії розгортання	2	Лекція-візуалізація (з використанням презентації)	11, 12, 36, 37	1. Написати unit тести для бізнес-логіки 2. Створити integration тести для API endpoints 3. Досягти 80%+ покриття коду 4. Створити Dockerfile для додатку 5. Налаштувати CI/CD pipeline (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 1. Встановлення Go та створення простого "Hello World" сервера 1. Встановлення Go SDK 2. Налаштування середовища розробки 3. Створення першого Go проєкту 4. Базовий HTTP сервер	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 13	1. Встановити Go версії 1.21 або новішої 2. Налаштувати VS Code з Go extension 3. Створити новий Go module з назвою "webserver" 4. Реалізувати HTTP сервер, який: 5. Запустити сервер на порту 8080 6. Протестувати за допомогою curl або Postman (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 2. Розробка REST API з базовим роутингом 1. Установка та налаштування Gin framework 2. Створення роутів для різних HTTP методів 3. Обробка JSON даних 4. Структурування проєкту	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 14, 15	1. Встановити Gin framework 2. Створити API для управління книгами 3. Використовувати in-memory slice як тимчасове сховище 4. Додати валідацію вхідних даних 5. Реалізувати правильні HTTP статус-коди (4 год.)	Звіт, 0-2 бали

Лабораторне заняття 3. Підключення PostgreSQL та створення моделей даних 1. Встановлення PostgreSQL 2. Підключення до БД з Go 3. Використання GORM 4. Створення моделей	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 32, 33	1. Встановити PostgreSQL та створити БД "bookstore" 2. Встановити GORM та драйвер для PostgreSQL 3. Створити моделі 4. Встановити зв'язки між моделями 5. Реалізувати автоматичні міграції 6. Заповнити БД тестовими даними (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 4. Реалізація CRUD операцій через API 1. Інтеграція GORM з Gin 2. Repository pattern 3. Обробка помилок БД 4. Пагінація результатів	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 32, 34	1. Створити repository layer для роботи з БД 2. Переписати API з лабораторної 2.2 для використання PostgreSQL 3. Реалізувати повний CRUD функціонал: 4. Додати пагінацію (limit, offset) 5. Реалізувати сортування результатів 6. Обробляти всі можливі помилки БД (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 5. Додавання middleware для логування та обробки помилок 1. Створення користувацького middleware 2. Логування HTTP запитів 3. Централізована обробка помилок 4. Recovery middleware	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 15, 35	1. Створити middleware для логування 2. Реалізувати middleware для обробки помилок 3. Додати recovery middleware для panic situations 4. Створити middleware для встановлення request ID 5. Інтегрувати logrus або zap для структурованого логування 6. Застосувати middleware до всіх роутів (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 6. Імплементація JWT автентифікації 1. JWT токени: структура та принципи 2. Реєстрація та логін користувачів 3. Хешування паролів 4. Захист endpoints	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 34, 35, 38	1. Створити модель User (ID, Email, Password, Role) 2. Реалізувати endpoints 3. Використовувати bcrypt для хешування паролів 4. Генерувати JWT токени з використанням jwt-go 5. Створити middleware для перевірки JWT 6. Додати refresh token механізм 7. Реалізувати logout функціонал	Звіт, 0-2 бали

				(4 год.)	
Лабораторне заняття 7. Валідація вхідних даних та обробка файлів 1. Валідація з використанням validator 2. Користувацькі правила валідації 3. Завантаження файлів 4. Зберігання файлів на сервері	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 15, 39	1. Встановити go-playground/validator 2. Додати validation tags до моделей 3. Реалізувати endpoint POST /books/:id/cover для завантаження обкладинки 4. Додати валідацію типів файлів (jpg, png) 5. Обмежити розмір файлів (max 5MB) 6. Зберігати файли у структурованих директоріях 7. Генерувати унікальні імена файлів 8. Повертати URL завантаженого файлу (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 8. Написання unit-тестів для API endpoints 1. Структура тестів в Go 2. HTTP тестування 3. Імітація БД 4. Табличні тести	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 36, 40	1. Створити тести для всіх CRUD операцій 2. Використовувати httptest для тестування handlers 3. Створити mock repository з testify/mock 4. Написати table-driven tests для валідації 5. Тестувати edge cases та error scenarios 6. Досягти мінімум 70% покриття коду 7. Створити setup та teardown функції 8. Запустити тести з генерацією звіту покриття (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 9. Контейнеризація додатку з Docker 1. Основи Docker 2. Створення Dockerfile 3. Multi-stage builds 4. Docker Compose	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	11, 12, 37, 41	1. Створити Dockerfile з multi-stage build 2. Створити .dockerignore файл 3. Створити docker-compose.yml з сервісами 4. Налаштувати змінні оточення 5. Налаштувати томи для персистентності даних 6. Налаштувати мережу між контейнерами 7. Запустити весь стек однією командою 8. Перевірити роботу API в контейнері (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 10. Розгортання на хмарну платформу 1. Підготовка додатку до production	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного	11, 12, 37	1. Підготувати додаток до розгортання 2. Вибрати платформу (Railway, Render або Fly.io) 3. Створити production БД на платформі	Звіт, 0-2 бали

2. Змінні середовища 3. Міграції бази даних у production 4. Моніторинг та логування		забезпечення, вихідний код програм, діаграми, схеми		4. Виконати міграції на production БД 5. Розгорнути додаток 6. Налаштувати HTTPS 7. Протестувати API в production 8. Налаштувати базовий моніторинг (4 год.)	
Захист програмного проєкту		Презентація, огляд програмного коду			
Всього балів за змістовим модулем 2					**30**
Лекцій	**10**				
Лабораторних занять	**20**				
Самостійна робота	**60**				
ЗМІСТОВИЙ МОДУЛЬ 3. FRONTEND-РОЗРОБКА З VUE.JS ТА ІНТЕГРАЦІЯ					
Лекція 1. Основи Vue.js 3 1. Еволюція Vue.js: Options API vs Composition API 2. Встановлення та налаштування Vue CLI / Vite 3. Структура Vue компонента 4. Реактивність: ref, reactive, computed 5. Синтаксис шаблонів та директиви 6. Обробка подій та двостороннє зв'язування 7. Хуки життєвого циклу 8. Props та користувацькі події 9. Слоти та компонентна композиція	2	Лекція-візуалізація (з використанням презентації)	12, 13, 14, 15	1. Встановити Node.js та npm 2. Створити новий Vue 3 проєкт з Vite 3. Створити компонент Todo List з додаванням/видаленням 4. Реалізувати фільтрацію списку 5. Вивчити Vue DevTools (4 год.)	Звіт, 0-2 бали
Лекція 2. Vue Router та управління станом 1. Концепція односторінкових додатків 2. Встановлення та налаштування Vue Router 3. Визначення маршрутів та вкладених маршрутів 4. Охоронці навігації	2	Лекція-візуалізація (з використанням презентації)	12, 14, 15	1. Створити multi-page додаток з роутингом 2. Реалізувати navigation guards для захисту роутів 3. Створити Pinia store для користувача 4. Реалізувати глобальний стан корзини покупок 5. Додати збереження стану в localStorage (4 год.)	Звіт, 0-2 бали

5. Динамічні маршрути та параметри 6. Pinia: сучасне управління станом 7. Stores, state, getters, actions 8. Композиція stores 9. Персистентність стану					
Лекція 3. Інтеграція frontend та backend 1. HTTP клієнти: Axios та Fetch API 2. Налаштування екземплярів Axios 3. Перехоплювачі для запитів та відповідей 4. Обробка помилок HTTP 5. CORS: причини та рішення 6. Axios з TypeScript 7. Async/await патерни у Vue 8. Стани завантаження та обробка помилок 9. Персистентність стану	2	Лекція-візуалізація (з використанням презентації)	12, 14, 34, 35	1. Встановити та налаштувати Axios 2. Створити сервісний рівень API 3. Реалізувати перехоплювач для додавання JWT 4. Створити composable для HTTP запитів 5. Коректно обробляти мережеві помилки (4 год.)	Звіт, 0-2 бали
Лекція 4. UI/UX та компонентні бібліотеки 1. Основи UI/UX дизайну для веб 2. Vuetify: Material Design для Vue 3. Element Plus: альтернативна бібліотека 4. Робота з формами у Vue 5. Валідація форм з VeeValidate 6. Адаптивний дизайн з Vuetify grid 7. Теми та налаштування 8. Найкращі практики доступності	2	Лекція-візуалізація (з використанням презентації)	12, 14, 15, 39	1. Встановити Vuetify або Element Plus 2. Створити responsive layout 3. Реалізувати складну форму з валідацією 4. Створити власну тему 5. Додати темний режим (4 год.)	Звіт, 0-2 бали
Лекція 5. Build та production deployment 1. Процес збірки Vite 2. Змінні середовища у Vue 3. Розділення коду та ліниве завантаження 4. Оптимізація розміру пакету 5. Можливості PWA 6. Генерація статичних сайтів з Vite 7. Розгортання на Netlify/Vercel 8. Налаштування Nginx для SPA 9. Оптимізація продуктивності	2	Лекція-візуалізація (з використанням презентації)	12, 14, 37, 41	1. Налаштувати змінні середовища 2. Оптимізувати пакет з розділенням коду 3. Створити production збірку 4. Проаналізувати розмір пакету 5. Розгорнути на Netlify або Vercel (4 год.)	Звіт, 0-2 бали

Лабораторне заняття 1. Створення Vue.js проєкту та базових компонентів 1. Ініціалізація Vue 3 проєкту 2. Структура проєкту 3. Створення компонентів 4. Props та події	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 13, 14, 15	1. Встановити Node.js (версія 18+) та npm 2. Створити новий Vue 3 проєкт з Vite: 3. Налаштувати ESLint та Prettier 4. Створити компоненти: header з навігацією, footer, картка книги з props, список карток 5. Використовувати Composition API 6. Додати базові стилі з scoped CSS 7. Реалізувати emit events від дочірніх компонентів (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 2. Розробка системи роутингу для SPA 1. Встановлення Vue Router 2. Конфігурація роутів 3. Вкладені роути 4. Охоронці навігації	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 15	1. Встановити Vue Router 4 2. Створити задані сторінки 3. Налаштувати роути з параметрами 4. Реалізувати вкладені роути для профілю 5. Додати охоронець навігації для захищених роутів 6. Створити 404 сторінку 7. Реалізувати програмну навігацію (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 3. Налаштування Pinia store для управління станом 1. Установка Pinia 2. Створення stores 3. State, getters, actions 4. Використання stores в компонентах	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 15	1. Встановити Pinia 2. Створити stores 3. У useAuthStore, useBooksStore та useCartStore реалізувати задані стани та дії 4. Інтегрувати stores з компонентами (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 4. Підключення до Go API через Axios 1. Встановлення Axios 2. Створення API сервісу 3. Перехоплювачі 4. Обробка помилок	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 34, 35	1. Встановити Axios 2. Створити `/src/services/api.js` з налаштованим Axios instance 3. Створити request interceptor для додавання JWT токена 4. Створити response interceptor для обробки помилок 5. Створити окремі сервіси 6. Інтегрувати сервіси з Pinia stores 7. Додати стани завантаження	Звіт, 0-2 бали

				8. Обробляти мережеві помилки з toast сповіщеннями (4 год.)	
Лабораторне заняття 5. Реалізація форм авторизації та реєстрації 1. Створення форм у Vue 2. Валідація форм 3. VeeValidate 4. Користувацький досвід	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 15, 39	1. Встановити VeeValidate та yup 2. Створити форму реєстрації 3. Створити форму логіну 4. Додати валідацію на стороні клієнта 5. Показувати помилки валідації 6. Блокувати кнопку submit під час запиту 7. Показувати loading spinner 8. Перенаправляти на головну після успішного логіну 9. Зберігати токен в localStorage 10. Додати "Forgot password?" функціонал (UI) (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 6. Створення CRUD інтерфейсу для роботи з даними 1. Vuetify таблиці даних 2. Діалогові вікна 3. CRUD операції через UI 4. Діалоги підтвердження	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 15, 39	1. Встановити Vuetify 2. Створити сторінку управління книгами (admin panel) 3. Реалізувати таблицю з заданими колонками 4. Додати кнопки дій: Edit, Delete 5. Створити діалог для додавання/редагування книги з заданою формою 6. Додати валідацію форми 7. Реалізувати підтвердження видалення 8. Додати пошук по таблиці 9. Реалізувати сортування колонок 10. Додати пагінацію (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 7. Додавання валідації форм на frontend 1. Валідація на стороні клієнта 2. Користувацькі правила валідації 3. Асинхронна валідація 4. Зворотний зв'язок з користувачем	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми,	12, 14, 15, 39	1. Розширити валідацію з VeeValidate 2. Додати користувацькі правила валідації 3. Створити composable `useFormValidation` 4. Додати валідацію в реальному часі (on blur, on input) 5. Показувати помилки біля полів 6. Додати стан завантаження для	Звіт, 0-2 бали

		схеми		асинхронної валідації 7. Створити компонент FormField з вбудованою валідацією 8. Додати атрибути доступності (aria-invalid, aria-describedby) (4 год.)	
Лабораторне заняття 8. Реалізація захищених роутів 1. Охоронці навігації 2. Потік автентифікації 3. Оновлення токена 4. Перенаправлення	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 15, 34, 35	1. Створити middleware для перевірки автентифікації 2. Додати `beforeEach` охоронця в router 3. Створити захищені роути 4. Реалізувати логіку перенаправлення після логіну (returnUrl) 5. Додати автоматичний вихід при 401 помилці 6. Реалізувати механізм оновлення токена 7. Додати екран завантаження під час перевірки автентифікації 8. Показувати різне меню для авторизованих/неавторизованих користувачів (4 год.)	Звіт, 0-2 бали
Лабораторне заняття 9. Оптимізація та збірка production версії 1. Конфігурація збірки Vite 2. Розділення коду 3. Ліниве завантаження 4. Аналіз пакету	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 37, 41	1. Налаштувати змінні середовища 2. Реалізувати ліниве завантаження для роутів 3. Додати розділення коду для великих бібліотек 4. Оптимізувати зображення (webp формат, компресія) 5. Додати meta теги для SEO 6. Створити production збірку 7. Проаналізувати пакет з `rollup-plugin-visualizer` 8. Оптимізувати розмір пакету (видалити невикористані залежності) 9. Налаштувати стиснення (gzip) 10. Протестувати production збірку локально (4 год.)	Звіт, 0-2 бали

Лабораторне заняття 10. Розгортання повного full-stack додатку 1. Розгортання frontend 2. Розгортання backend 3. Налаштування CORS 4. SSL сертифікати	2	Практична робота. Матеріали: ознайомчі та демонстраційні версії програмного забезпечення, вихідний код програм, діаграми, схеми	12, 14, 37, 41	1. Підготувати backend для production 2. Розгорнути backend на Railway або Render 3. Розгорнути frontend на Vercel або Netlify 4. Налаштувати custom domain (опціонально) 5. Налаштувати SSL/HTTPS 6. Перевірити роботу повного додатку 7. Додати Google Analytics (опціонально) 8. Створити документацію процесу розгортання 9. Налаштувати неперервне розгортання з GitHub (4 год.)	Звіт, 0-2 бали
Захист програмного проєкту		Презентація, огляд програмного коду			
Всього балів за змістовим модулем 3					**30**
Лекцій	**10**				
Лабораторних занять	**20**				
Самостійна робота	**60**				
Підсумковий контроль: екзамен		Тестування			**40**

10. **Система оцінювання та вимоги**

Навчальні досягнення студентів оцінюються за 100-бальною шкалою Університету, чотирибальною шкалою (5 «відмінно», 4 «добре», 3 «задовільно», 2 «незадовільно»), і шкалою оцінок ЄКТС. На поточний контроль відводиться 60 балів.

Оцінювання поточної успішності студентів на окремих навчальних заняттях та за виконання завдань самостійної роботи визначається диференційовано, відповідно до рівня складності завдань, та встановлюється в межах від 0 до 4 балів.

Кількість балів за роботу з теоретичним матеріалом та на лабораторних заняттях залежить від дотримання таких вимог:

- своєчасність виконання навчальних завдань;
- повний обсяг їх виконання;
- якість виконання навчальних завдань;
- оригінальність виконання у групах;
- творчий підхід у виконанні завдань.

Під час роботи над матеріалом змістових модулів 2 та 3 студенти можуть об'єднуватись у групи по 2-4 чоловіки й виконувати завдання разом, розділяючись по ролям розробників.

Виконання завдань самостійної роботи є обов'язковим. До їх виконання допускаються всі студенти. Студент, який не виконав поточних завдань, не підготувався до лабораторних занять, отримує 0 балів. Поточну заборгованість, пов'язану з непідготовленістю або недостатньою підготовленістю до навчальних занять, студент повинен ліквідувати шляхом виконання у визначений термін завдань, передбачених програмою. За виконані завдання нараховуються від 0 до 6 балів.

Студенти, які за результатами поточного контролю набрали менше 20 балів, вважаються такими, що мають академічну заборгованість, ліквідація якої є обов'язковою. Студенти, які не мають академічної заборгованості за результатами поточного контролю, допускаються до екзамену.

11. **Критерії оцінювання успішності навчання**

1. Завданням **поточного контролю** є систематична перевірка розуміння та засвоєння програмного матеріалу шляхом тестування, аналіз виконання завдань самостійної роботи, умінь у електронному форматі представляти певний матеріал.

Критеріями оцінювання у ході поточного контролю є:

а) під час поточної аудиторної роботи на лекційних та семінарських заняттях:
- активна участь у дискусіях та пропонованих формах роботи на лекційних та лабораторних заняттях;
- доповнення та запитання на лекційних та лабораторних заняттях;

б) при виконанні завдань для самостійної та індивідуальної роботи:
- повнота виконання завдання;
- творчість та самостійність виконання.

Завданням **підсумкового контролю** (екзамену) є комплексна діагностика результатів навчання, глибини засвоєння студентом програмного матеріалу з навчальної дисципліни, логіки та взаємозв'язків між окремими його змістовими модулями, здатності до творчого використання набутих знань.

Екзамен містить 30 тестових завдань, кожне з яких оцінюється 1-2 балами.

12. **Список рекомендованої літератури / інтернет-ресурси / нормативні документи**

Основна

1. Mass M., Biehl M. API Design Patterns. – Manning Publications, 2021. – 480 p.
2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – O'Reilly Media, 2017. – 614 p.

3. Indrasiri K., Siriwardena P. Design Patterns for Cloud Native Applications: Patterns in Practice Using APIs, Data, Events, and Streams. – O'Reilly Media, 2021. – 298 p.
4. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. – O'Reilly Media, 2020. – 419 p.
5. Ford N., Parsons R., Kua P. Building Evolutionary Architectures: Support Constant Change. – 2nd ed. – O'Reilly Media, 2021. – 314 p.
6. Newman S. Building Microservices: Designing Fine-Grained Systems. – 2nd ed. – O'Reilly Media, 2021. – 612 p.
7. Newman S. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. – O'Reilly Media, 2019. – 272 p.
8. Nygard M. T. Release It! Design and Deploy Production-Ready Software. – 2nd ed. – Pragmatic Bookshelf, 2018. – 378 p.
9. Stopford B. Designing Event-Driven Systems: Concepts and Patterns for Streaming Services with Apache Kafka. – O'Reilly Media, 2018. – 156 p.
10. Vernon V. Domain-Driven Design Distilled. – Addison-Wesley Professional, 2016. – 176 p.
11. Tolaram N., Glynn N. Full-Stack Web Development with Go: Build your web applications quickly using the Go programming language and Vue.js. – Packt Publishing, 2023. – 302 p.
12. Bodner J. Learning Go: An Idiomatic Approach to Real-World Go Programming. – O'Reilly Media, 2021. – 376 p.
13. Donovan A., Kernighan B. The Go Programming Language. – Addison-Wesley Professional, 2015. – 400 p.
14. Macrae C. Vue.js 3 Design Patterns and Best Practices: Develop scalable and robust applications with Vite, Pinia, and Vue Router. – Packt Publishing, 2023. – 296 p.
15. Hanchett E., Listwon B. The Majesty of Vue.js 3. – Leanpub, 2022. – 250 p.

Додаткова

16. Khononov V. Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy. – O'Reilly Media, 2021. – 341 p.
17. Brookshear J. G., Smith D., Brylow D. Computer Science: An Overview. – 14th ed. – Pearson, 2023. – 672 p.
18. Burns B. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. – O'Reilly Media, 2018. – 166 p.
19. Beyer B., Jones C., Petoff J., Murphy N. R. Site Reliability Engineering: How Google Runs Production Systems. – O'Reilly Media, 2016. – 552 p.
20. Beyer B., Murphy N. R., Rensin D. K., Kawahara K., Thorne S. The Site Reliability Workbook: Practical Ways to Implement SRE. – O'Reilly Media, 2018. – 512 p.
21. Winand M. SQL Performance Explained: Everything Developers Need to Know about SQL Performance. – Markus Winand, 2020. – 204 p.
22. Harrison G., Gromoll D. Database Reliability Engineering: Designing and Operating Resilient Database Systems. – O'Reilly Media, 2020. – 330 p.
23. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley Professional, 2020. – 416 p.
24. Freeman E., Robson E., Sierra K., Bates B. Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software. – 2nd ed. – O'Reilly Media, 2020. – 694 p.
25. Adkins H., Beyer B., Blankinship P., Lewandowski P., Oprea A., Stubblefield A. Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems. – O'Reilly Media, 2020. – 558 p.
26. Kane D. Cloud Native Patterns: Designing Change-tolerant Software. – Manning Publications, 2021. – 400 p.
27. Majors C., Fong-Jones L., Miranda G. Observability Engineering: Achieving Production Excellence. – O'Reilly Media, 2022. – 316 p.
28. Percival H., Gregory B. Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven

Microservices. – O'Reilly Media, 2020. – 304 p.
29. Richardson M., Rymer J. R. Web API Design: The Missing Link. – Manning Publications, 2023. – 328 p.
30. Gupta A. Mastering API Architecture: Design, Operate, and Evolve API-Based Systems. – O'Reilly Media, 2023. – 482 p.
31. Tilkov S., Eigenbrodt M. Web Architecture: Design Principles and Patterns for Modern Web Applications. – O'Reilly Media, 2024. – 392 p.32. Yermalovich A. Practical Go: Building Scalable Network and Non-Network Applications. – Wiley, 2022. – 560 p.
33. Conery R. A Curious Moon: A Functional Approach to Database Design with PostgreSQL. – Big Machine, 2017. – 234 p.
34. Fielding R.T., Taylor R.N. Architectural Styles and the Design of Network-based Software Architectures. – University of California, Irvine, 2015. – 162 p.
35. Richardson C., Smith F. Microservices Patterns: With examples in Java. – Manning Publications, 2018. – 520 p.
36. Kennedy W., Ketelsen B., Martin E.S. Go in Action. – Manning Publications, 2015. – 264 p.
37. Poulton N. Docker Deep Dive: Zero to Docker in a single book. – Independently published, 2023. – 370 p.
38. Siriwardena P. Advanced API Security: OAuth 2.0 and Beyond. – Apress, 2020. – 400 p.
39. Osmani A. Learning JavaScript Design Patterns: A JavaScript and React Developer's Guide. – O'Reilly Media, 2023. – 310 p.
40. Winters T., Manshreck T., Wright H. Software Engineering at Google: Lessons Learned from Programming Over Time. – O'Reilly Media, 2020. – 602 p.
41. Kim G., Debois P., Willis J., Humble J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. – IT Revolution Press, 2016. – 480 p.

Посилання на інтернет-ресурси

42. Microsoft Azure Architecture Center (2024): https://learn.microsoft.com/en-us/azure/architecture/
43. AWS Well-Architected Framework (2024): https://aws.amazon.com/architecture/well-architected/
44. Google Cloud Architecture Framework (2024): https://cloud.google.com/architecture/framework
45. Martin Fowler's Blog (2024): https://martinfowler.com/
46. The Twelve-Factor App (2024): https://12factor.net/
47. OWASP Top 10 (2021, оновлення 2024): https://owasp.org/www-project-top-ten/
48. Microservices.io by Chris Richardson (2024): https://microservices.io/