КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Кафедра моделювання та програмного забезпечення

ЗАТВЕРДЖУЮ
Перший проректор
_____ Владислав ЧУБАРОВ
“_____”_____2023 р.

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Основи WEB-програмування
(шифр і назва навчальної дисципліни)

спеціальність 121 – «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки)

факультет Інформаційних технологій
(назва інституту, факультету, відділення)

Форма навчання	Курс	Семестр	Всього годин за планом	Кількість національних кредитів	Всього аудиторних годин	Аудиторних годин, (у тому числі КЗ)		Самостійна робота (год.)	Контрольно-модульні роботи	Залік (сем.)	Екзамен (сем.)
						Лекції	Лабораторні				
Денна	1	2	150	5	54	18	36	96	1		*
Заочна	1	2	150	5	20	10	10	130	-		*

Кривий Ріг – 2023 рік

Робочу програму навчальної дисципліни «Основи WEB-програмування» для здобувачів першого (бакалаврського) рівня вищої освіти за освітньою програмою «Інженерія програмного забезпечення» розроблено згідно з ОПП галузі знань 12 «Інформаційні технології» зі спеціальності 121 «Інженерія програмного забезпечення».

Розробник: доцент кафедри МПЗ Трачук А.А.

Робоча програма затверджена на засіданні кафедри моделювання та програмного забезпечення

Протокол від “____”_____2023 року № ___

Завідувач кафедри МПЗ, доцент, к.п.н. _____ Андрій СТРЮК

Схвалено вченою радою факультету інформаційних технологій

Протокол від “____”_____2023 року № ___

Голова вченої ради _____ Іван МУЗИКА

Схвалено групою забезпечення ОПП

Протокол від “____”_____2023 року № ___

Гарант ОПП _____ Андрій СТРЮК

ЗМІСТ

1. ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

<table>
<tr><td rowspan="2">Найменування показників</td><td rowspan="2">Галузь знань, спеціальність, освітній рівень</td><td colspan="2">Характеристика навчальної дисципліни</td></tr>
<tr><td>денна форма навчання</td><td>заочна форма навчання</td></tr>
<tr><td>Кількість кредитів – 5</td><td>Галузь знань
Інформаційні технології</td><td colspan="2">Нормативна</td></tr>
<tr><td>Модулів – 2</td><td rowspan="5">Спеціальність:
121 Інженерія програмного забезпечення</td><td colspan="2">Рік підготовки:</td></tr>
<tr><td>Змістових модулів – 2</td><td>1-й</td><td>1-й</td></tr>
<tr><td rowspan="3">Загальна кількість годин – 150</td><td colspan="2">Семестр</td></tr>
<tr><td>2-й</td><td>2-й</td></tr>
<tr><td colspan="2">Лекції</td></tr>
<tr><td rowspan="10">Тижневих годин для денної форми навчання:
аудиторних – 3
самостійної роботи студента – 5,3</td><td rowspan="10">Ступінь вищої освіти:
бакалавр</td><td>18 год.</td><td>10 год.</td></tr>
<tr><td colspan="2">Практичні, семінарські</td></tr>
<tr><td>-</td><td>-</td></tr>
<tr><td colspan="2">Лабораторні</td></tr>
<tr><td>36 год.</td><td>10 год.</td></tr>
<tr><td colspan="2">Самостійна робота</td></tr>
<tr><td>96 год.</td><td>130 год.</td></tr>
<tr><td colspan="2">Вид контролю: екзамен</td></tr>
</table>

Примітка.

Співвідношення кількості годин аудиторних занять до самостійної роботи становить:

для денної форми навчання – 0,56

для заочної форми навчання – 0,154

2. МЕТА ТА ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

2.1. **Метою** викладання навчальної дисципліни «Основи Web-програмування» є формування у здобувачів теоретичного базису щодо сучасних підходів і методик розробки веб-сторінок, а також формування практичних навичок з роботи з інформаційним ресурсом в глобальній комп'ютерній мережі Internet та вмінь програмування web-сторінок у подальшій фаховій практиці.

2.2. **Завданнями** вивчення дисципліни «Основи Web-програмування» є

- набуття теоретичних знань та практичних умінь з формування базового уявлення про галузі застосування сучасних технологій Internet-програмування;
- набуття вмінь і навичок проектування програмного забезпечення;
- знайомство з теорією при програмуванні основних сервісів Інтернет,
- опанування синтаксису і семантики мов HTML, CSS та JavaScript;
- знайомство з програмуванням змісту www–ресурсів, з використанням при цьому відповідних інтегрованих середовищ розробки.

2.3. Відповідно до освітньої програми дисципліна забезпечує наступні **компетентності**:

Загальні компетентності

ЗК01 Здатність до абстрактного мислення, аналізу та синтезу.

ЗК02 Здатність застосовувати знання у практичних ситуаціях.

ЗК05 Здатність вчитися і оволодівати сучасними знаннями.

ЗК06 Здатність до пошуку, оброблення та аналізу інформації з різних джерел.

Фахові компетентності

СК01 Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.

СК02 Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.

СК03 Здатність розробляти архітектури, модулі та компоненти програмних систем.

СК05 Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу.

СК07 Володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних.

СК11 Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення.

СК12 Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.

СК13 Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

СК14 Здатність до алгоритмічного та логічного мислення.

Програмні результати навчання освітньої програми, яким відповідає дисципліна:

ПР01 Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.

ПР03 Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення.

ПР04 Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення.

ПР05 Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об’єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення.

ПР06 Уміння вибирати та використовувати відповідні задачі методології створення програмного забезпечення.

ПР07 Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

ПР09 Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення.

ПР11 Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання.

ПР13 Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.

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

ПР15 Мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення.

ПР16 Мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації.

ПР17 Вміти застосовувати методи компонентної розробки програмного забезпечення.

ПР18 Знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних.

ПР23 Вміти документувати та презентувати результати розробки програмного забезпечення.

В результаті вивчення дисципліни здобувач повинен:

*знати***:**

- базові методи створення Web-сторінок
- принципи проектування логічної структури Web-сторінок;
- синтаксис і семантику мов програмування клієнт-серверних застосувань HTML, CSS, JavaScript;
- компонентний підхід до програмування;
- базові принципи проектування з урахуванням таких якостей, як продуктивність, безпека, захищеність, можливість повторного використання, надійність.

*вміти***:**

- обґрунтовувати необхідність використання об’єктно-орієнтованої технології або компонентного підходу для вирішення конкретної задачі;
- використовувати в професійній діяльності сучасні інтегровані середовища програмування;
- використовувати основні синтаксичні та семантичні конструкції мов HTML, CSS, JavaScript;
- створювати консольні прикладні застосування мовами HTML, CSS, JavaScript на основі сучасних методів об'єктно-орієнтованого та компонентного програмування;
- самостійно опановувати нові методи та технології розробки програм.

2.4. Міждисциплінарні зв’язки

При вивченні дисципліни використовуються знання здобувачів з дисциплін «Алгоритмізація обчислювальних процесів», «Вища математика», «Основи інженерії програмного забезпечення», «Основи програмування».

Знання, одержані здобувачами при вивченні дисципліни, використовуються при вивченні дисциплін «Архітектура та проектування програмного забезпечення», «Організація комп`ютерних мереж», «Сучасні технології Internet-програмування» з курсовою роботою, «Програмна інженерія розподілених інтернет-застосувань» з курсовою роботою, «Програмування мобільних пристроїв».

Вимоги до знань та умінь визначаються галузевими стандартами вищої освіти України.

3. ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Змістовий модуль 1. Мови гіпертекстової розмітки.

Тема 1. Загальні відомості про Інтернет технології.

Мережа Інтернет: сервіси, протоколи,домени, адресація ресурсів в Інтернет. Всесвітня павутина WWW. Сучасний стан Інтернет технологій, перспективи розвитку. Основи WEB: засоби створення web - сторінки та їх види (HTML, CSS, JavaScript, Flash). Поняття сайту. Типи web – сайтів і оцінка їх якості . Будова сайтів

Тема 2. Вивчення мови розмітки гіпертекстових документів HTML.

Мови гіпертекстової розмітки HTML, XML, XHTML. Елементи мови HTML і DTD цих елементів. Структура гіпертекстового документу. Елементи текстового і блокового рівнів. Таблична розмітка і узагальнена розмітка.

Тема 3. Створення Web – сторінок засобами HTML.

Створення посилань. Вставка зовнішніх об'єктів в гіпертекстові документи. Створення діалогових форм. Правила побудови HTML – документів. Структурування тексту в HTML – документі за допомогою списків.

Тема 4. Форматування web-документів за допомогою каскадних таблиць стилів CSS (Cascading Style Sheets).

Основні синтаксичні конструкції мови. Поняття селектора і види селекторів. CSS правила, властивості і їх можливі значення. Зв'язування CSS правил і гіпертекстових документів. Методи використання таблиць, групуючи елементи DIV та SPAN. Вбудований та впроваджений стилі CSS.

Змістовий модуль 2. Засоби створення інтерактивних WEB-сайтів.

Тема 5. Вступ в JavaScript.

Динамічний вміст web-сторінки. Реалізація програмної взаємодії JavaScript з HTML документами на основі DOM API. Вимоги до технологій розробки клієнтських обробників. Огляд сучасних технологій. JavaScript, призначення і принципи роботи. Основні типи даних, синтаксис і вбудовані об'єкти мови. Ієрархія класів, що описує браузер в JavaScript.

Тема 6. Система подій і виконання функцій JavaScript.

Призначення та використання подій і функцій JavaScript. Розміщення коду на html-сторінці. Ієрархія класів. Створення типових додатків на JavaScript.

Тема 7. Об’єктна модель JavaScript.

Об’єкти JavaScript: поняття об’єкту, прототипи, властивості та методи об’єкта Object. Створення меню web – сторінки мовою JavaScript.

Тема 8. Стандартні об'єкти JavaScript.

Управління таблицями стилів засобами JavaScript. Програмування гіпертекстових переходів на JavaScript. Взаємодія JavaScript – програм з Flash – презентаціями. Захист коду JavaScript.

4. СТРУКТУРА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Назви змістових модулів і тем	Кількість годин							
	денна форма				заочна форма			
	разом	у тому числі			разом	у тому числі		
		лекц.	лаб.	с.р.		лекц.	лаб.	с.р.
1	2	3	4	5	6	7	8	9
Змістовий модуль 1 Мови Інтернет-програмуваня								
Тема 1. Загальні відомості про Інтернет технології	18	2	4	12	18	1	1	16
Тема 2. Вивчення мови розмітки гіпертекстових документів HTML	18	2	4	12	18	1	1	16
Тема 3. Створення Web – сторінок засобами HTML	22	4	6	12	22	2	2	18
Тема 4. Вивчення Cascading Style Sheets	20	2	4	14	20	1	1	18
Разом за змістовим модулем 1	**78**	**10**	**18**	**50**	**78**	**5**	**5**	**68**
Змістовий модуль 2 Засоби створення інтерактивних WEB-сайтів								
Тема 5. Вступ в JavaScript	18	2	4	12	18	1	1	16
Тема 6. Система подій і виконання функцій JavaScript	20	2	6	12	20	2	2	16
Тема 7. Об’єктна модель JavaScript	16	2	4	10	16	1	1	14
Тема 8. Стандартні об'єкти JavaScript	18	2	4	12	18	1	1	16
Разом за змістовим модулем 2	**72**	**8**	**18**	**46**	**72**	**5**	**5**	**62**
Разом за семестр	**150**	**18**	**36**	**96**	**150**	**10**	**10**	**130**

5. ТЕМИ ПРАКТИЧНИХ ЗАНЯТЬ

Не передбачено.

6. ТЕМИ ЛАБОРАТОРНИХ ЗАНЯТЬ

№ з/п	Назва теми	Кількість годин	
		Денна ф. н.	Заочна ф. н.
1.	Лабораторна робота №1. Створення алгоритму розробки web-сайту	4	1
2.	Лабораторна робота №2. Аналіз сильних та слабких сторін web-сайту	4	1
3.	Лабораторна робота №3. Створення HTML-сторінок	6	2
4.	Лабораторна робота №4. Структурування інформації за допомогою списків та форм	4	1
5.	Лабораторна робота №5. Таблиця стилів CSS	6	1
6.	Лабораторна робота №6. Використання шарів і зображень-мап	4	1
7.	Лабораторна робота №7. Реалізація програмної взаємодії JavaScript з HTML документами	4	2
8.	Лабораторна робота №8. Робота із стандартними об'єктами JavaScript	4	1
Усього годин		**36**	**10**

7. САМОСТІЙНА РОБОТА

На самостійну роботу здобувачам денної форми навчання відведено 96 години, заочної - 130 годин.

Самостійна робота здобувачів при вивченні дисципліни «Основи WEB-програмування» залучає такі складові:

- опрацювання лекційного матеріалу з кожної теми;
- опрацювання додаткової літератури по темі;
- вивчення окремих тем або питань, що передбачені для самостійного опрацювання;
- підготовка до виконання, а також до захисту лабораторних робіт;
- виконання індивідуального завдання, зокрема підготовка матеріалу до розробки та захисту проекту;
- підготовка до проведення контрольних заходів.

Розподіл годин самостійної роботи

№ з/п	Назва теми	Кількість годин	
		денна	заочна
1.	Типи web – сайтів і оцінка їх якості.	15	17
2.	Поняття кроссбраузерной розмітки. Редактори HTML.	15	18
3.	Пожвавлення web-сторінки за допомогою графіки: теги IMG, OBJECT. Фрейми..	10	16
4.	Методи використання та синтаксис таблиці, групуючи елементи DIV та SPAN.	10	17
Разом за змістовим модулем 1:		**50**	**68**
5.	Ієрархія класів, що описує броузер в JavaScript.	12	11
6.	Програмування гіпертекстових переходів на JavaScript.	11	19
7.	Об’єктна модель JavaScript.	11	16
8.	Взаємодія JavaScript – програм з Flash – презентаціями.	12	16
Разом за змістовим модулем 2:		**46**	**62**
Разом:		**96**	**130**

8. МЕТОДИ НАВЧАННЯ

Використовуються наступні методи навчання**:** лекції, лабораторні заняття, самостійна робота.

Навчальна лекція – це логічне, послідовне викладання змісту навчання, яке характеризується судженнями, висновками, підсумком. Вона охоплює основний теоретичний матеріал однієї або кількох тем навчальної дисципліни. Призначенням лекції є формування у здобувачів фундаментальних знань з дисципліни, а також визначає основний зміст і характер усіх інших навчальних занять та самостійної роботи здобувачів із цієї дисципліни.

Лабораторне заняття - форма організації навчання, яку проводять за завданням і під керівництвом НПП. Основні дидактичні цілі – експериментальне підтвердження вивчених теоретичних положень навчальної дисципліни та формування вмінь й навичок їх практичного застосування. Проведення лабораторного заняття ґрунтується на попередньо підготовлених наборах завдань різної складності для розв’язання на занятті. Лабораторне заняття проводиться у навчальних лабораторіях з використанням пристосованого до умов навчального процесу устаткування.

Самостійна робота здобувача є основним способом оволодіння навчальним матеріалом у час, вільний від обов’язкових аудиторних занять. Мета виконання самостійної роботи – поглиблення, узагальнення й закріплення теоретичних знань і практичних умінь здобувачів із дисципліни шляхом вироблення вміння самостійної роботи з навчальною і фаховою літературою та інформацією в мережі Інтернет.

Самостійна робота здобувачів здійснюється у формі: підготовки до лекцій і лабораторних занять, виконанні самостійних проектів. Самостійну роботу здобувач може виконувати у бібліотеці, комп’ютерних класах, а також у домашніх умовах.

Підготовка до лекцій передбачає самостійне опрацювання теоретичного матеріалу. При цьому необхідно звернути увагу на необхідність чіткого засвоєння основних термінів та визначень, розуміння їх змісту, обов’язкового аналізу використання теоретичних положень для розв’язання наданих прикладів.

Самоперевірку засвоєння навчального матеріалу здобувач здійснює за контрольними запитаннями, що надано після кожної теми у конспекті лекцій та іншій літературі, та після кожного лабораторного заняття у відповідних методичних вказівках. Якщо на деякі запитання здобувач не може надати відповіді, то необхідно повторити вивчення навчального матеріалу, або визначити правильну відповідь з викладачем на консультації.

Під час вивчення даної дисципліни використовуються:

− мультимедійні освітні технології: інтерактивні лекції (презентації) із використанням програми MS Power Point у поєднанні з анімацією та звуковим супроводом; перегляд відеороликів за окремими пунктами тем занять, використання електронних посібників;

− діалогові технології: організація групових обговорень, використання «мозкового штурму».

Лекції проводяться з використанням технічних засобів навчання й супроводжуються демонстрацією презентацій за допомогою проектора.

У разі виникнення необхідності забезпечення навчального процесу в дистанційному режимі супровід та контроль знань реалізовується за допомогою дистанційного курсу, розробленого в Google ClassRoom. Онлайн лекції, консультації та усні відповіді на питання, захист проєктів проводиться за допомогою Google Meet або Zoom.

9. МЕТОДИ КОНТРОЛЮ

Основними завданнями контролю знань здобувачів вищої освіти з дисципліни є оцінювання засвоєння теоретичних знань і практичних навичок, отриманих під час навчання.

Контрольні заходи мають виконувати наступні функції:

- стимулювати систематичну самостійну роботу над навчальним матеріалом;
- забезпечувати закріплення та реалізацію набутих теоретичних знань при підготовці до практичних занять;
- прищеплювати навички відповідального ставлення до своїх обов’язків, самостійного цілеспрямованого пошуку потрібної інформації, чіткої організації свого робочого дня.

Оцінювання знань здобувачів складається з поточного та підсумкового контролю.

Поточний контроль знань здобувачів вищої освіти передбачає оцінювання за наступними основними напрямами:

- перевірка теоретичних знань;
- перевірка підготовки до лабораторних занять;
- перевірка виконання індивідуального проекту.

З даних компонентів складаються загальні бали, які фіксуються в журналі викладача.

Оцінювання рівня засвоєння теоретичних знань здобувачів вищої освіти проводиться під час усної співбесіди зі здобувачами по теоретичним матеріалам, за результатами захисту проекту й виконання самостійних робіт. Підсумковим контролем є залік.

10. РОЗПОДІЛ БАЛІВ, ЯКІ ОТРИМУЮТЬ ЗДОБУВАЧІ

Використовується модульно-рейтингова система оцінювання, яка передбачає розподіл балів за виконання всіх запланованих видів робіт. Сума складається з балів, що накопичив студент у ході поточного контролю. При цьому максимальна кількість балів за семестр, при

умові його бездоганного виконання, дорівнює 100 балів, а мінімальна – 50 балів. Успішність студентів-заочників оцінюється аналогічно.

Лабораторні роботи відображують оволодіння навичками та вміння застосовувати знання на практиці і складаються з завдань різного рівня. Базовий рівень відповідає виконанню 60% завдань лабораторної роботи, достатній рівень – 80% і високий – 100%. При зниженні якості виконання тієї чи іншої лабораторної роботи, знижується і кількість балів, якою вона оцінюється.

Розподіл балів за видами робіт

№ модуля	№ зан	Вид роботи	Максимальна кількість балів (для усіх форм навчання)		
			Базовий рівень	Достатній рівень	Високий рівень
1	1	Лабораторна робота № 1	7	10	12
	2	Лабораторна робота № 2	8	10	13
	3	Лабораторна робота № 3	8	10	13
	4	Лабораторна робота № 4	7	10	12
2	5	Лабораторна робота № 5	7	10	12
	6	Лабораторна робота № 6	8	10	13
2	7	Лабораторна робота № 7	8	10	13
	8	Лабораторна робота № 8	7	10	12
Разом			**60**	**80**	**100**

Оцінювання кожної лабораторної роботи та індивідуального завдання ведеться за показниками, наведеними в таблиці:

Оцінювання кожної лабораторної роботи ведеться за показниками, наведеними в таблицях відповідно до рівня:

№ з/п /Критерій оцінювання	Максимальна кількість балів, *високий рівень* (денна форма/заочна форма)			
	Своєчасність виконання	Правильність виконання	Захист роботи	Всього за роботу
Лабораторна робота № 1	3/-	5/13	4/-	12/13
Лабораторна робота № 2	3/-	5/13	4/-	12/13
Лабораторна робота № 3	3/-	5/14	5/-	13/14
Лабораторна робота № 4	3/-	5/13	4/-	12/13

Лабораторна робота № 5	3/-	5/13	5/-	13/13
Лабораторна робота № 6	3/-	5/13	5/-	13/13
Лабораторна робота № 7	3/-	5/13	4/-	12/13
Лабораторна робота № 8	3/-	5/13	5/-	13/13
Всього	**24/-**	**40/100**	**36/-**	**100/100**

№ з/п /Критерій оцінювання	Максимальна кількість балів, *достатній рівень* (денна форма/заочна форма)			
	Своєчасність виконання	Правильність виконання	Захист роботи	Всього за роботу
Лабораторна робота № 1	3/-	4/10	3/-	10/10
Лабораторна робота № 2	3/-	4/10	3/-	10/10
Лабораторна робота № 3	3/-	4/10	3/-	10/10
Лабораторна робота № 4	3/-	4/10	3/-	10/10
Лабораторна робота № 5	3/-	4/10	3/-	10/10
Лабораторна робота № 6	3/-	4/10	3/-	10/10
Лабораторна робота № 7	3/-	4/10	3/-	10/10
Лабораторна робота № 8	3/-	4/10	3/-	10/10
Всього	**24/-**	**32/80**	**24/-**	**80/80**

№ з/п /Критерій оцінювання	Максимальна кількість балів, *базовий рівень* (денна форма/заочна форма)			
	Своєчасність виконання	Правильність виконання	Захист роботи	Всього за роботу
Лабораторна робота № 1	2/-	3/7	2/-	7/7
Лабораторна робота № 2	2/-	3/8	3/-	8/8
Лабораторна робота № 3	2/-	3/8	3/-	8/8
Лабораторна робота № 4	2/-	3/8	3/-	8/8
Лабораторна робота № 5	2/-	3/8	3/-	8/8
Лабораторна робота № 6	2/-	3/7	2/-	7/7
Лабораторна робота № 7	2/-	3/7	2/-	7/7
Лабораторна робота № 8	2/-	3/7	2/-	7/7
Всього	**16/-**	**24/60**	**20/-**	**60/60**

Під своєчасністю виконання та своєчасністю захисту лабораторної роботи розуміється виконання та захист у тиждень згідно із графіком робіт.

Правильність виконання роботи оцінюється наступним чином:

- робота виконана без зауважень - максимальний бал;
- робота виконана достатньо повно з деякими зауваженнями – дві третини від максимального балу;
- при перевірці роботи виявлені грубі помилки – 0 балів.

Захист лабораторної роботи передбачає відповіді на контрольні питання, які представлені у методичних вказівках до виконання лабораторних робіт відповідно до кожної теми.

Для допуску до підсумкового контролю студент повинен виконати графік навчального процесу, усі види запланованих завдань і протягом семестру отримати в сумі не менше 50 балів.

Семестровий контроль здійснюється у формі екзамену.

Бали, набрані студентом за результатами поточного контролю складають максимум 60 відсотків, а екзаменаційне завдання – 40 відсотків оцінки за семестр.

Бали конвертуються у відсотки наступним чином:

бали	відсотки
<65	50
65-80	55
>80	60

Результати екзамену оцінюються за 100-бальною шкалою. У відомість оцінка проставляється як у балах національної шкали, так і за шкалою ECTS:

Шкала оцінювання

Національна шкала успішності	**Оцінка ECTS**	**Визначення ECTS**	**100-бальна система оцінювання**
відмінно/ зараховано	**A**	ВІДМІННО - відмінне виконання лише з незначними помилками	90...100
добре/ зараховано	**B**	ДУЖЕ ДОБРЕ - вище середнього рівня з кількома помилками	80...89
	C	ДОБРЕ - у цілому правильно робота з певною кількістю помилок і недоліків	71...79
задовільно/ зараховано	**D**	ЗАДОВІЛЬНО - непогано, але зі значною кількістю грубих помилок	61...70
	E	ДОСТАТНЬО - виконання задовольняє мінімальні потреби	50...60
незадовільно/ не зараховано	**FX**	НЕЗАДОВІЛЬНО - із можливістю повторного складання	30...49

	F	НЕЗАДОВІЛЬНО - з обов'язковим повторним вивчення дисципліни	0...29

При наявності у здобувачів **результатів неформального навчання** за освітнім компонентом «Основи WEB-програмування» у повному обсязі, визнання та оцінювання результатів здійснюється відповідно до «Положення про порядок визнання у Криворізькому національному університеті результатів навчання, отриманих в умовах неформальної освіти». У випадку, якщо за підсумками визнання результатів неформального навчання визнається тільки частина результатів навчання, заявнику зараховуються окремі види навчальної роботи за освітнім компонентом «Основи WEB-програмування».

Нижче наведені окремі види навчальної роботи, які можуть бути зараховані здобувачеві при наявності сертифікату про успішне проходження рекомендованих онлайн курсів.

Тема	**Курси**
Загальні відомості про Інтернет технології	https://prometheus.org.ua/prometheus-plus/front-end/
Вивчення мови розмітки гіпертекстових документів HTML.	https://training.epam.ua/Training/Details/3474?lang=ua
Вивчення Cascading Style Sheets	https://training.epam.ua/Training/Details/3474?lang=ua
Головні технології front-end розробника: HTML, CSS та JavaScript;	https://prometheus.org.ua/prometheus-plus/front-end/
Вступ в JavaScript	https://prometheus.org.ua/prometheus-plus/front-end/

11 ІНСТРУМЕНТИ, ОБЛАДНАННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

- Аудиторія персональних комп’ютерів класу Celeron, AMD, Pentium, Core 2 Duo, Core i5, i7 (або вище) з операційною системою типу Windows 7, 8 або 10. Забезпеченість комп’ютерами – 12 шт. на 25 студентів.
- Підключення до мережі Інтернет.
- Програмне забезпечення: Visual Studio Code.

12. ПЕРЕЛІК ПИТАНЬ ДЛЯ ПІДСУМКОВОГО КОНТРОЛЮ ЗНАНЬ

1. Початкові етапи розробки сайту (тех. завдання, визначення бізнес-вимог, структура сайту).
2. Типи сайтів. Модель взаємодії комп'ютерів в мережі.
3. Модульна сітка сайту, варіанти її побудови.
4. Мови розмітки. HTTP - протоколи. Web – редактори.
5. Структура і правила складання тегів. Контейнер. Атрибути. Основні групи тегів.
6. Теги для створення заголовків тексту. Фізичне і логічне виділення тексту.
7. Гіперпосилання і якір.
8. Невпорядковані списки. Впорядковані списки.
9. Створення таблиць. Атрибути тегів створення таблиць.
10. Вкладені таблиці. Атрибути тегів створення таблиць.
11. Структура HTML – документу.
12. Тег <body> і його атрибути. Теги
 і <p>. Способи завдання абзацного відступу. Вирівнювання абзацу.
13. Спеціальні символи і їх додавання в HTML документ.
14. Способи представлення кольорів в Web – документі.
15. Вставка малюнків в HTML – документ. Підготовка зображень перед вставкою в HTML – документ. Властивості картинок в HTML - документе
16. HTML – форми.
17. Обробник Html форм. Атрибути тегу форми.
18. Методи передачі даних з форми, їх відмінності, переваги.
19. Тег Input, його атрибути.
20. Створення текстового поля форми, поля введення пароля, багаторядкового поля.
21. Поля з радіокнопками, прапорці.
22. Типи кнопок на формі.
23. Випадний список на формі, приховані поля.
24. Створення фреймів.
25. Параметри тегу <frame>. Переваги і недоліки фреймів.
26. Каскадні таблиці стилів. Сенс каскадности.
27. У чому полягає каскадність таблиці стилів.
28. Методи застосування таблиці стилів до документу HTML.
29. Синтаксис і пріоритети таблиць стилів.
30. Селектори та класи в CSS. Вага селектора. Складні селектори.
31. Контекстні селектори. Псевдокласи CSS.
32. Принципи вибору адреси для сайту.
33. Реєстрація сайту в пошукових системах і каталогах. Підготовка сторінки до реєстрації в пошукових системах.
34. Мета-теги.
35. Навігаційні мапи. Серверна та клієнтська форма.
36. Типи областей на навігаційних мапах.
37. Історія створення JavaScript. Орфографія мови. Варіанти підключення скріпта.
38. Типи даних JavaScript.
39. Види модальних вікон.
40. Функції, аргументи функції JavaScript. Область видимості змінної.
41. Оператори порівняння JavaScript, умовні оператори.
42. Масиви і робота з ними, цикли в JavaScript.
43. Об’єкти JavaScript.
44. Об’єкт Объект Window та його методи.
45. Управління подіями в JavaScript.

46. Робота с математичними функціями.
47. Робота зі строками, парсінг.
48. Асоціативні массиви.
49. Робота з об’єктом Date.
50. Установка часу та таймеру.

13. НАВЧАЛЬНО-МЕТОДИЧНІ МАТЕРІАЛИ З ДИСЦИПЛІНИ

13.1 Навчальна та довідкова література

1. Джамса К., Кінг К., Андерсон Е. Креативний WEB-дизайн. HTML, XHTML, CSS, JavaScript, PHP, ASP, ActiveX. Текст, графіка, звук та анімація. ДіаСофт, 499 стор.-2016.
2. Матвєєва Л.Є., Волков В.А. Процес розробки програмного забезпечення. Від теорії до практики. – К., 2018. – 117 с.
3. Крамер Ерік. HTML: наочний курс веб-дизайну. : Пров. з англ. : Навч. Пос. - М.: Видавничий дім "Вільямс", 2017. - 304 с.
4. Д. Крокфорд "JavaScript: Сильні Сторони".
5. David Flanagan "JavaScript: The Definitive Guide".
6. Стоян Стефанов “JavaScript. Шаблони”.
7. Бер Бібо, Єгуда Кац “jQuery in Action”.
8. Дакетт, Джон. JavaScript та jQuery. Інтерактивна веб-розробка / Джон Дакетт;(Пер. з англ. М.А. Райтмана). - М.: Видавництво «Е»217. - 640 с.
9. Джамса К. Ефективний самовчитель з креативного Web-дизайну. HTML, XHTML, CSS, JavaScript, PHP, ASP, ActiveX. Текст, графіка, звук та анімація. М: "ТОВ ДіаСофтЮП", 2015.
10. Ніколенко Д.В. Практичні заняття з JavaScript. М.: Наука та техніка, 2010. - 128 с.

13.2. Методична література

1. Конспект лекцій з курсу «Основи Web-програмування» [Текст] : Метод. посібн. / Уклад. А.А.Трачук, Кривий Ріг, КНУ, 2022 р. — 84 с.
2. Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи Web-програмування» для усіх форм навчання спеціальності 121 «Інженерія програмного забезпечення». / Уклад. А.А.Трачук – Кривий Ріг, КНУ, 2021 р. — 72 с.
3. Презентації лекцій з дисципліни «Основи Web-програмування» для студентів всіх форм навчання зі спеціальності 121 – Інженерія програмного забезпечення.

14. ІНФОРМАЦІЙНІ РЕСУРСИ

До складу інформаційних ресурсів навчальної дисципліни входять:

1. Бібліотека Криворізького національного університету (м. Кривий Ріг, вул. Пушкіна, 37). – Режим доступу: http://lib.knu.edu.ua/

Internet-ресурси:

2. Google Classroom. [Електронний ресурс] – Режим доступу: https://classroom.google.com/u/0/c/NTg3NzQ5NjEyMzIy

3. JavaScript. [Електронний ресурс] - Режим доступу: http://pluralsight.com/training/Courses#javascript.

4. Кишеньковий довідник з JavaScript. [Електронний ресурс] - Режим доступу: - http://bonsaiden.github.io/JavaScript-Garden.

5. Стандарт HTML. [Електронний ресурс] - Режим доступу: https://html.spec.whatwg.org/multipage/window-object.html#thewindow-object.

6. W3schoolsEN.TheBest. [Електронний ресурс] - Режим доступу: https://w3schoolsua.github.io/index_en.html#gsc.tab=0 .

15. ТЕРМІНОЛОГІЧНИЙ СЛОВНИК

Гіпертекстові Web-сторінки (Hypertext Web pages) are stored in ordinary text files with the extension NTM or HTML. HTML is a markup language for hypertext documents, designed for creating and writing hypertexts. HTML is based on the standard generalized markup language (SGML — Standard Generalized Markup Language) and is intended for describing interconnected documents in the distributed network information system WWW. HTML describes not only the structure of documents, but also the relationship between them. In general, HTML is a set of styles (tags) that distinguish different components of WWW documents.

The appearance of the document on the user's screen is determined by the browser. Browsers are graphic and text, in each browser the document will look in its own way, but its structure will remain unchanged, as it is specified by the html format.

Мова HTML (language HTML) - HTML is a language that describes how a document should look. It can also be called a markup language.

The main element of HTML are descriptors or tags. The document is formatted by adding a descriptor that specifies exactly how the text should look. HTML descriptors are enclosed in angle brackets. Conventionally, descriptors can be divided into three parts: Descriptors that inform the browser that the document is an HTML document and comment descriptors. HTML document header descriptors. HTML document body descriptors.

There are two types of descriptors: container descriptors and simple descriptors.

Атрибути HTML (HTML attributes) – Many HTML descriptors also accept parameters called attributes. Attributes are used to indicate additional information to the Web browser about how to use this descriptor. If you add a handle that creates a link, the attributes will be used to specify the URL of the linked Web page. Attributes are names that are assigned specific values through an equal sign (=).

Успадкування (Inheritance) - Inheritance is a mechanism that allows a new class, called the derived class or subclass, to be based on an existing class, called the base class or superclass. The derived class inherits the properties and behavior of the base class, which it can then modify, extend, or

override as needed. This enables code reuse and allows for a more organized and modular code structure, as common properties and behavior can be defined in the base class and reused in the derived classes.

Поліморфізм (Polymorphism) - Polymorphism refers specifically to the ability of an object or function to behave in different ways based on the context in which it is used. This can include subtype polymorphism, as in OOP, but can also refer to other forms of polymorphism such as parametric polymorphism, where a function or data type can operate on values of different types, or ad-hoc polymorphism, where a function can behave differently based on the types of its arguments.

Абстракція (Abstraction) - Abstraction is a fundamental concept in computer science and programming that involves focusing on essential features or properties of an object or system, while ignoring details that are not relevant to its function or behavior. In programming, abstraction can refer to the process of defining and using abstract classes or interfaces that specify the essential features or behaviors of a group of related objects, without specifying their implementation details. Abstraction is important for creating reusable, modular, and maintainable code, as it allows programmers to work with higher-level concepts and hide implementation details that may change over time.

Властивість (Property) - Property is a member of a class or struct that provides access to a private field or other underlying data source, and allows for reading and/or writing of the value through getter and/or setter methods. Properties are often used as an abstraction layer to provide controlled access to an object's state or data, and can enforce validation, calculation, or other behaviors when the value is accessed or modified. Properties are declared with a get and/or set accessor, which define the logic for getting or setting the value, and can be declared with access modifiers such as public, private, or protected, to control their visibility and accessibility.

Метод (Method) - Method is a collection of statements or instructions that perform a specific task or action. Methods are associated with a class or object and can be called to perform a specific action on that class or object. Methods can take input parameters and return output values. Methods are often used to encapsulate reusable code and provide a way to perform a specific operation on an object or system, as part of the abstraction and modularity principles of OOP.

Конструктор (Constructor) - constructor is a special method that is called when an object of a class is created or instantiated. The constructor is used to initialize the object's state or data, and can take parameters or use default values. Constructors are typically used to set the initial values of an object's fields or properties, and can perform initialization tasks such as memory allocation or resource acquisition. Constructors are declared with the same name as the class, and can have different signatures to support different parameter lists or overloading. The use of constructors is a fundamental aspect of object-oriented programming and supports the encapsulation and abstraction principles.

Абстрактний клас (abstract class) - abstract class is a class that cannot be instantiated directly, but serves as a template or blueprint for other classes to derive from. An abstract class may contain one or more abstract methods, which are declared but not implemented in the abstract class itself, but must be implemented in derived classes. Abstract classes are used to define common behavior or characteristics that can be shared by multiple subclasses, while allowing for variation and

customization in the implementation of specific methods or properties. Abstract classes are often used to implement the abstraction and inheritance principles of OOP, and base behavior, that must be extended by subclasses.

Інтерфейс (Interface) - an interface is a collection of abstract methods and properties that define a contract or set of behaviors that a class or object must implement. An interface defines a set of rules or guidelines for implementing a specific feature or behavior, but does not contain any implementation details itself. Interfaces are often used to enable polymorphism and provide a way for objects of different types to interact with each other in a consistent way. In C#, interfaces are declared using the interface keyword and may contain methods, properties, indexers, and events. Implementing an interface requires a class or struct to provide concrete implementations for all the methods and properties defined in the interface.

Виключення (Exception) - an exception is an object that represents an error or unexpected condition that occurs during the execution of a program. Exceptions can be thrown by a method or statement when an error occurs, and are used to signal that an operation has failed or could not be completed. Exceptions can contain information about the type and location of the error, as well as any relevant data or context. Exception handling is the process of detecting, responding to, and recovering from exceptions, and is an important aspect of robust and reliable software design. In C#, exceptions are typically handled using try-catch blocks or other structured error handling mechanisms.

Колекція (Collection) - a collection is an object that groups together other objects or values into a single unit, typically for the purpose of storing, organizing, or processing them in a convenient way. Collections can be used to represent a wide range of data structures, from simple arrays and lists to more complex structures like sets, maps, and trees. Collections can be generic or non-generic, depending on whether they are strongly typed or not, and can provide a variety of methods and properties for manipulating and querying the contents of the collection. Collections are an essential part of many programming tasks, from simple data processing to complex algorithms and data analysis. In C#, collections are often implemented using the ICollection or IEnumerable interfaces (but not only them), and may be provided by the standard library or custom data structures.

Узагальнення (Generic) - generic refers to a feature that allows classes, methods, and other constructs to be parameterized by one or more types, rather than being limited to a fixed set of types. Generics provide a way to write reusable and type-safe code that can work with a wide variety of data types, without having to create separate implementations for each type. By defining classes or methods that can accept any type of data, generics enable greater flexibility and code reuse, while reducing the potential for errors and inefficiencies. Generics are commonly used for collections, algorithms, and other data structures that need to be generic and type-safe. In C#, generics are implemented using type parameters and are declared using angle brackets (< >). Also, generics can filter types in it by using word “where”.

Подія (Event) - an event is a mechanism that allows objects to notify other objects or components about changes or actions that occur within them. Events are commonly used for communication and coordination between objects, and can be used to trigger specific behaviors or

reactions in response to changes in the system or user interactions. Events are typically implemented using a publisher-subscriber model, where one object (the publisher) raises an event, and one or more other objects (the subscribers) register to receive and handle the event. Events can be used to implement a wide range of behaviors, from simple notifications to complex workflows and state machines. In C#, events are declared using the event keyword and are typically implemented using delegates or event handlers.

Типи значень (Value types) - In C#, a value type is a data type that directly stores its value in memory, rather than storing a reference to an object that contains the value. Examples of value types in C# include integral types (such as int, long, and byte), floating-point types (such as float and double), and the bool, char, and decimal types. Value types are typically smaller and more efficient than reference types, as they do not require memory allocation on the heap and do not need to be garbage collected. In C#, value types are passed by value, meaning that a copy of the value is passed to a method or assigned to a variable, rather than a reference to the original value. However, value types can also be boxed, which means that they can be wrapped in a reference type and treated as objects.

Типи посилань (reference type) - In C#, a reference type is a data type that stores a reference to an object in memory, rather than storing the object directly. Examples of reference types in C# include class types, interface types, delegate types, and array types. Reference types are allocated on the heap and are garbage collected by the CLR (Common Language Runtime), which means that they can be dynamically allocated and deallocated at runtime. In C#, reference types are passed by reference, meaning that a reference to the original object is passed to a method or assigned to a variable, rather than a copy of the object's value. This allows multiple variables or methods to access and modify the same object. However, this also means that reference types can lead to issues such as memory leaks and unintended modifications if not used carefully.

Нульові значення (null) - In programming, null is a special value that represents the absence of a value or the lack of an object reference. It is typically used to indicate that a variable or object does not currently hold a valid value or reference. In C#, null is the default value for reference types, meaning that a variable that has not been initialized will hold a null reference. Attempting to access a member or method of a null reference will result in a NullReferenceException at runtime. In contrast, value types cannot hold null as a value, but can be made nullable using the ? operator or the Nullable<T> structure. Handling null values is an important consideration when writing robust and reliable code.

Computer Vision: The field of computer science that deals with enabling computers to "see" and interpret visual information.

Knowledge Engineering: The process of acquiring, organizing, and representing knowledge within intelligent information systems.

Semantic Web: An extension of the World Wide Web that enables machines to understand and process information more effectively.

16 ЗМІНИ ТА ДОПОВНЕННЯ ДО РОБОЧОЇ ПРОГРАМИ

№ з/п	Дата внесення змін	Зміст змін та доповнень	Підстава до внесення змін (№ і дата наказу, рішення вченої ради, засідання кафедри)
1	2	3	4

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____20 р.

Завідувач кафедри _____

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____20 р.

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____20 р.

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____20 р.

Завідувач кафедри _____ Завідувач кафедри _____

ДОДАТОК 1

Робочий план з дисципліни «Основи WEB-програмування»

Семестр 1

Вид навчальної роботи	Годин у семестрі / кредити			Тиждень							Вид підсум-кового контро-лю
		1	3	5	7	9	11	13	15	17	
Лекційні заняття	18	2ПК	2ПК	2ПК	2ПК	2ПК	2ПК	2ПК	2ПК	2ПК	
Лабораторні заняття	36	4ПК	4ПК	4ПК	4ПК	4ПК	4ПК	4ПК	4ПК	4ПК	
Самостійна робота	96	10	10	10	10	10	10	10	13	13	
Всього годин/кредитів	150/5	16	16	16	16	16	16	16	19	19	екзамен

Позначки: ПК – поточний контроль

Викладач А.А.Трачук