ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ

Кафедра інформаційних та комп'ютерних систем

Северин М.В.

МЕТОДИЧНІ ВКАЗІВКИ

з дисципліни «Введення до фаху»

до лабораторних робіт

для здобувачів спеціальності 122(F3) «Комп'ютерні науки»

ОДЕСА
ДУІТЗ
2025

УДК 004:378(07)

Укладач

Северин М.В. старший викладач кафедри інформаційних та комп’ютерних систем Державного університету інтелектуальних технологій і зв’язку

Рецензенти:

Бондар Л. В. – кандидат педагогічних наук, доцент кафедри інноваційних технологій та методики викладання природничих дисциплін, Південноукраїнський національний педагогічний університет імені К. Д. Ушинського (ПНПУ), Одеса, Україна.

Панченко Б.Є., д.ф.-м.н., професор кафедри Інженерії програмного забезпечення Державного університету інтелектуальних технологій і зв'язку

Рекомендовано до друку Навчально-методичною Радою
Державного університету інтелектуальних технологій і зв’язку
(протокол №11 від 25 грудня 2025 р.)

Введення до фаху: методичні вказівки до лабораторних робіт [для здобувачів першого (бакалаврський) рівня вищої освіти спеціальності 122(F3) «Комп'ютерні науки»] / уклад. М. В. Северин. Одеса : ДУІТЗ, (Електр. вид. https://metod.suitt.edu.ua), 2025. 48 с.

Методичний посібник містить комплекс лабораторних робіт з дисципліни «Вступ до фаху» для студентів спеціальності 122(F3) «Комп’ютерні науки». У виданні охоплено ключові аспекти сучасної IT-індустрії: від аналізу технологічного стеку та хмарних обчислень до практичного опанування систем керування версіями Git, методологій Agile, Scrum та основ веб-розробки. Особлива увага приділена питанням кібербезпеки, командній взаємодії у Jira, а також юридичним та кар’єрним аспектам працевлаштування. Посібник спрямований на формування у першокурсників цілісного розуміння життєвого циклу програмного забезпечення та розвиток професійних компетенцій, необхідних для успішного старту в IT-галузі.

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій висуває дедалі вищі вимоги до підготовки фахівців у галузі комп'ютерних наук. Сучасний IT-спеціаліст – це не лише програміст, який володіє певним набором мов програмування, а й інженер, що розуміє архітектуру процесів розробки, вміє працювати у команді, орієнтується у питаннях безпеки та здатен швидко адаптуватися до змін на ринку праці. Дисципліна «Вступ до фаху» є першим кроком у професійному становленні студента, покликаним створити міцний фундамент для подальшого вивчення спеціалізованих дисциплін.

Метою даного методичного посібника є надання студентам теоретичних знань та практичних навичок, що відображають реалії сучасної IT-індустрії. Структура посібника побудована за принципом поступового занурення у професійне середовище. Лабораторний цикл розпочинається з дослідження глобальної інфраструктури – хмарних сервісів, аналізу динаміки технологічного ринку та вивчення критично важливих аспектів кібербезпеки. Ці знання дозволяють студенту зрозуміти контекст, у якому створюються сучасні програмні продукти.

Важливе місце у посібнику посідає вивчення процесів розробки та інструментів командної взаємодії. Студенти на практиці порівнюють класичну методологію Waterfall з гнучкими фреймворками (Scrum), вчаться декомпозувати завдання у системах керування проєктами (Jira, Trello) та опановують роботу з системою керування версіями Git. Такий підхід дозволяє змоделювати умови роботи в реальній продуктовій або аутсорсинговій компанії.

Технічний блок лабораторних робіт знайомить майбутніх розробників з основами веб-технологій (HTML5/CSS3) та проєктуванням ігрових механік. Це дає можливість спробувати себе у різних напрямках розробки та обрати пріоритетну спеціалізацію. Важливою складовою є також розвиток soft skills та юридична грамотність. Завершальні лабораторні роботи присвячені технологіям працевлаштування (створення CV, профілю LinkedIn, підготовка до поведінкових співбесід) та вивченню правових форм співпраці, таких як офер, контракт та NDA.

Виконання завдань, представлених у посібнику, дозволить студенту не лише опанувати інструментарій розробника, а й сформувати професійне мислення, навчитися працювати в команді та свідомо підійти до побудови своєї кар'єрної траєкторії в динамічному світі IT.

Лабораторна робота №1

Тема. Сучасний стек IT-фахівця та хмарні сервіси.

Мета: Ознайомитися з базовим інструментарієм розробника, навчитися конфігурувати робоче середовище та розуміти принципи роботи хмарних моделей (IaaS, PaaS, SaaS).

1. Теоретичні відомості

Сучасний IT-фахівець не працює в ізоляції, він використовую набір технологій ("стек" технологій), який складається не тільки з мови програмування, а й з:

- ***IDE (Integrated Development Environment):*** середовище, де пишеться код (VS Code, JetBrains IntelliJ IDEA/PyCharm).
- ***Термінал (CLI):*** інструмент для керування системою через команди.
- ***Хмарні сервіси:***
  - ***IaaS (Infrastructure as a Service):*** оренда "заліза" (процесор, диск), наприклад, AWS EC2.
  - ***PaaS (Platform as a Service):*** готова платформа для запуску коду без турботи про ОС (Heroku, Google App Engine).
  - ***SaaS (Software as a Service):*** готове програмне забезпечення (ПЗ) доступне через браузер (Google Drive, Slack, Jira).

2. Завдання

Встановлення базового ПЗ: Встановити редактор коду (рекомендовано VS Code) та налаштувати 2-3 професійні розширення (наприклад, Prettier, Bracket Pair Colorizer).

1. ***Робота з CLI:*** Відкрити термінал (PowerShell для Windows або Terminal для macOS/Linux). Виконати базові команди навігації: cd, ls (або dir), mkdir, touch.
2. ***Хмарний аналіз:*** Створити обліковий запис у будь-якому хмарному сховищі для розробників (наприклад, Google Drive або Dropbox для зберігання документації, або скористатися "Sandbox" версіями AWS/Azure).
3. ***Виконання індивідуального завдання:*** Порівняти дві технології з індивідуального завдання за трьома критеріями: ціна, легкість використання, сфера застосування.

3. Контрольні запитання

1. Чим IDE відрізняється від звичайного текстового редактора (наприклад, Notepad)?
2. У чому головна перевага хмарних обчислень перед локальними серверами?
3. Поясніть різницю між терміналами Bash, PowerShell та Zsh.
4. Що таке "менеджери пакетів" (npm, pip) і до якої категорії інструментів вони відносяться?
5. Які хмарні сервіси ви використовуєте у повсякденному житті, не усвідомлюючи, що це SaaS?

4. Індивідуальні завдання

Порівняйте дві технології/сервіси відповідно до вашого номера у списку групи:

1. AWS vs Microsoft Azure (IaaS)
2. Google Cloud Platform vs DigitalOcean
3. Docker vs VirtualBox (Віртуалізація)
4. VS Code vs JetBrains IntelliJ IDEA
5. Sublime Text vs Atom (Legacy vs Modern)
6. GitHub vs Bitbucket (Cloud Git)
7. Slack vs Microsoft Teams (SaaS Collaboration)
8. Heroku vs Netlify (PaaS)
9. Linux Bash vs Windows PowerShell
10. SQL vs NoSQL (Бази даних як сервіс)
11. Vercel vs Firebase Hosting
12. Jenkins vs GitHub Actions (CI/CD tools)
13. PyCharm vs Spyder (IDE для Python)
14. Vim vs Emacs (Консольні редактори)
15. Trello vs Jira (Task Management SaaS)
16. Dropbox vs Google Drive (Storage SaaS)
17. Cloudflare vs Akamai (CDN & Security)
18. Kubernetes vs Docker Swarm
19. Nginx vs Apache (Web Servers)
20. Postman vs Insomnia (API Testing tools)
21. Figma vs Adobe XD (UI Design SaaS)
22. Redis vs Memcached (Caching)

23. Terraform vs Ansible (Infrastructure as Code)
24. MongoDB Atlas vs Amazon DynamoDB
25. Jupyter Notebook vs Google Colab
26. NPM vs Yarn (JS Package Managers)
27. Maven vs Gradle (Java Build Tools)
28. Xcode vs Android Studio
29. Bitwarden vs LastPass (Security SaaS)
30. Zoom vs Google Meet (Video SaaS)

5. Зміст звіту

1. Титульний лист.
2. Скріншот встановленого та налаштованого IDE з відкритим файлом hello.txt.
3. Скріншот терміналу з виконаною командою перегляду поточної директорії.
4. Аналітична записка за індивідуальним завданням (1 сторінка).
5. Відповіді на контрольні запитання.
6. Висновки щодо готовності вашого ПК до навчання за спеціальністю.

Лабораторна робота №2

Тема. Дослідження динаміки IT-ринку та стратегічне планування технологічного стеку.

Мета: Провести порівняльний аналіз популярності мов програмування та технологій, навчитися працювати зі звітами аналітичних агенцій (DOU, Stack Overflow, TIOBE) та сформувати персональний план навчання на основі актуальних вимог ринку.

1. Теоретичні відомості

IT-ринок постійно змінюється. Для вибору актуального стеку фахівці використовують декілька джерел:

- ***Індекси популярності (TIOBE, PYPL):*** показують, скільки людей шукають інформацію про мову в Google.
- ***Опитування розробників (Stack Overflow Survey):*** відображають найпопулярніші та "найбільш оплачувані" технології.
- ***Локальна аналітика (DOU.ua):*** дає розуміння ситуації саме в Україні (зарплати, кількість вакансій).

2. Завдання

1. ***Глобальний аналіз:*** Зайти на сайт *Stack Overflow Survey (https://survey.stackoverflow.co/)*. Знайти розділ "Technology" за останній рік та виписати топ-5 мов програмування.
2. ***Локальний аналіз:*** На порталі DOU *(dou.ua)* для звіту за останній рік знайти медіанну зарплату для вашого напрямку (згідно з варіантом).
3. ***Порівняння:*** Знайти на Djinni *(https://djinni.co)* кількість вакансій для рівня Junior та для рівня Senior за вашим стеком. Обчислити коефіцієнт конкуренції (кількість відгуків на одну вакансію).
4. ***Складання Roadmap:*** Використовуючи сервіс *https://roadmap.sh*, знайти дорожню карту для свого варіанта та виписати 5 ключових технологій, які потрібно вивчити після опанування базової мови програмування.

3. Контрольні запитання

1. Що таке "медіанна зарплата" і чому вона краще відображає реальність, ніж "середня"?

2. Чому високий рейтинг мови в індексі TIOBE не завжди означає велику кількість вакансій для початківців?
3. Яка різниця між Hard Skills (технічними навичками) та Domain Knowledge (знанням предметної області, наприклад, фінтех або медицина)?
4. Як криза в IT-індустрії впливає на вимоги до Junior-фахівців?
5. Чому важливо знати англійську мову незалежно від обраного технологічного стеку?

4. Індивідуальні завдання

Провести дослідження конкретного сегмента ринку відповідно до варіанту:

1. Web-Frontend: Динаміка React vs Angular vs Vue за останні 3 роки.
2. Web-Backend (Enterprise): Ринок Java-розробки (Spring Framework) в Україні.
3. Data Science: Популярність Python vs R та попит на бібліотеки AI/ML.
4. Mobile (Native): Попит на Swift-розробників (iOS) порівняно з Kotlin (Android).
5. Mobile (Cross-platform): Ринок Flutter vs React Native.
6. GameDev (High-end): Аналіз вакансій Unreal Engine (C++) та вимоги до знань математики.
7. GameDev (Indie/Mobile): Попит на Unity-розробників (C#).
8. Cybersecurity: Аналіз вакансій Security Auditor та Pentester.
9. DevOps: Попит на знання AWS порівняно з Azure та Google Cloud.
10. Embedded: Стек технологій для розробки мікроконтролерів та IoT.
11. QA Automation: Порівняння популярності Selenium vs Playwright vs Cypress.
12. Big Data: Стек технологій Apache Spark, Hadoop та попит на Data Engineers.
13. Blockchain: Ринок Solidity-розробників та Web3 технологій.
14. System Programming: Сучасний стан та перспективи мови Rust.
15. Backend (High Load): Ринок Go (Golang) розробки в продуктових компаніях.
16. E-commerce: Попит на розробників платформ Shopify, Magento, Salesforce.
17. Desktop Dev: Стан ринку розробки на C# (.NET) для Windows.
18. Cloud-native: Попит на фахівців із Kubernetes та Docker.

19. CRM/ERP: Аналіз ринку впровадження SAP та Microsoft Dynamics.
20. AI Implementation: Вакансії AI Prompt Engineer та AI Integration specialist.
21. No-Code/Low-Code: Попит на фахівців Bubble, Webflow та Zapier.
22. Fintech: Специфічні вимоги до розробників у банківській сфері.
23. Healthcare IT: Стек технологій для медичних інформаційних систем.
24. Database Admin: Порівняння попиту на адміністраторів PostgreSQL vs Oracle.
25. Linux Systems: Вакансії системних адміністраторів та Linux-інженерів.
26. Frontend (Design Systems): Попит на розробників, що спеціалізуються на UI-кітах.
27. Fullstack (JS): Аналіз вакансій зі стеком MERN (Mongo, Express, React, Node).
28. Hardware Dev: Ринок розробки друкованих плат (PCB Design) та FPGA.
29. Python for Web: Ринок Django vs FastAPI розробників.
30. Legacy Systems: Попит на підтримку старих систем (COBOL, Fortran) – де та скільки платять.

5. Зміст звіту

1. Титульний лист.
2. Аналітична таблиця з даними (Джерело – Показник – Значення).
3. Графіки, що ілюструють різницю зарплат Junior/Middle/Senior у вашому варіанті.
4. Скріншот персонального Roadmap з обраного напрямку.
5. Відповіді на контрольні запитання.
6. Висновок: "Чи варто починати кар'єру в цьому напрямку?" (аргументуйте свою думку на основі цифр).

Лабораторна робота №3

Тема. Система безпеки в IT: аналіз кіберзагроз та методи захисту.

Мета: Вивчити основні типи кіберзагроз, зробити аналіз вразливостей програмного забезпечення та навчитися застосовувати базові інструменти захисту даних на рівні розробника та користувача.

1. Теоретичні відомості

Кібербезпека базується на тріаді ***CIA***:

- ***Confidentiality (Конфіденційність)*** – доступ до даних мають лише авторизовані особи.
- ***Integrity (Цілісність)*** – дані не були змінені або пошкоджені під час передачі чи зберігання.
- ***Availability (Доступність)*** – сервіси та дані доступні користувачам у будь-який момент.

Основними векторами атак сьогодні є: соціальна інженерія (фішинг), експлуатація вразливостей коду (SQL-ін'єкції, XSS), атаки на відмову в обслуговуванні (DDoS) та шкідливе ПЗ (Ransomware).

Рекомендовані інструменти для роботи

- ***Password Managers:*** Bitwarden, KeePass.
- ***2FA Apps:*** Google Authenticator, Authy.
- ***Security Check:*** *https://haveibeenpwned.com*.
- ***SSH Generation:*** ssh-keygen (у терміналі).

2. Завдання

1. ***Дослідження:*** Вивчити опис атаки згідно з вашим варіантом.
2. ***Аналіз:*** Знайти реальний історичний приклад такої атаки (кейс) за останні 5 років.
3. ***Практика:***
   - Створити надійний пароль (мінімум 12 символів, різні регістри, цифри, спецсимволи) та перевірити його на сервісах типу *Have I Been Pwned (*наприклад, *https://haveibeenpwned.com/Passwords)*.
   - Налаштувати двофакторну автентифікацію (2FA) у будь-якому професійному сервісі (GitHub, GitLab, Google).

– Згенерувати пару SSH-ключів для безпечного доступу до репозиторіїв.

4. ***Моделювання:*** Описати 3 кроки, які б допомогли запобігти атаці з вашого варіанта.

3. Контрольні запитання

1. Що таке багатофакторна автентифікація (MFA) і чому SMS-коди вважаються найменш надійним методом?
2. Поясніть різницю між симетричним та асиметричним шифруванням.
3. Що таке "соціальна інженерія" і чому вона є найнебезпечнішим вектором атаки?
4. Яка роль файлу .env у веб-розробці з точки зору безпеки?
5. Що таке принцип "Zero Trust" (нульова довіра) в архітектурі мереж?

4. Індивідуальні завдання

Підготувати доповідь-аналіз та план захисту від конкретної загрози:

1. Phishing (Фішинг): Викрадення облікових даних через підроблені сайти.
2. SQL Injection: Впровадження шкідливого коду в запити до бази даних.
3. DDoS-атака: Перевантаження сервера масовими запитами.
4. Ransomware (Шифрувальник): Блокування файлів з вимогою викупу.
5. Man-in-the-Middle (MitM): Перехоплення даних у відкритих Wi-Fi мережах.
6. Cross-Site Scripting (XSS): Впровадження скриптів у сторінки, які переглядають інші користувачі.
7. Brute Force: Метод автоматичного підбору паролів.
8. Social Engineering: Психологічне маніпулювання співробітниками компанії.
9. Keylogging: Запис натискань клавіш для крадіжки паролів.
10. Zero-day Vulnerability: Використання помилок у програмному забезпеченні, про які ще не знає розробник.
11. Insider Threat: Навмисний витік даних від співробітника компанії.
12. Spyware: Програми для прихованого стеження за діями користувача.
13. Supply Chain Attack: Атака через оновлення легітимного ПЗ (кейс MeDoc/NotPetya).
14. DNS Spoofing: Перенаправлення користувача на фейковий IP-адрес.
15. Credential Stuffing: Використання бази вкрадених паролів з одного сайту для доступу до інших.

16. Session Hijacking: Викрадення сесії користувача (cookies).
17. IoT Botnets: Використання "розумних" пристроїв для проведення масштабних атак.
18. Dictionary Attack: Підбір паролів за словником популярних слів.
19. Drive-by Download: Приховане завантаження вірусів при простому відвідуванні сайту.
20. Trojan Horse: Шкідливе ПЗ, масковане під корисну програму.
21. Adware: Нав'язлива реклама, що веде до зараження системи.
22. Wi-Fi Eavesdropping: Прослуховування мережевого трафіку.
23. Rootkit: ПЗ для отримання повного контролю над ОС, яке важко виявити.
24. Clickjacking: Обман користувача з метою натискання на невидимі елементи сторінки.
25. SIM Swapping: Перехоплення мобільного номера для доступу до банкінгу.
26. Data Exfiltration: Переміщення даних за межі безпечного периметра компанії.
27. Bluejacking/Bluesnarfing: Атаки на пристрої через протокол Bluetooth.
28. Cryptojacking: Використання чужих потужностей для майнінгу криптовалют.
29. Buffer Overflow: Переповнення буфера для виконання довільного коду.
30. Eavesdropping (Прослуховування): Аналіз незашифрованого HTTP-трафіку.

5. Зміст звіту

1. Титульний лист.
2. Результати виконання завдання роботи.
3. Відповіді на контрольні запитання.
4. Висновки.

Лабораторна робота №4

Тема. Порівняльний аналіз методологій управління ІТ-проєктами: від каскадної моделі (Waterfall) до гнучкого фреймворку (Scrum).

Мета: Ознайомитися з основними принципами методологій Waterfall та Scrum. Отримати практичні навички планування розробки ПЗ за допомогою діаграми Ганта (Waterfall) та формування беклогу проєкту з розподілом на спринти (Scrum). Навчитися обирати оптимальну модель управління залежно від вимог проєкту.

1. Теоретичні відомості

Waterfall (Водоспадна модель) – це послідовний підхід до розробки ПЗ, де кожна наступна фаза починається лише після повного завершення попередньої.

Переваги: Чіткий план, фіксований бюджет та терміни.

Інструмент: Діаграма Ганта – візуальний графік робіт, що відображає послідовність завдань, їх тривалість та залежності.

Основні етапи:

1. ***Аналіз вимог (Requirements):*** Збір усіх побажань замовника та написання ТЗ.
2. ***Проєктування (Design):*** Створення архітектури системи та дизайну інтерфейсів.
3. ***Розробка (Implementation):*** Написання коду.
4. ***Тестування (Verification):*** Перевірка на баги.
5. ***Експлуатація та підтримка (Maintenance):*** Реліз та виправлення помилок у користувачів.

Методологія Agile та фреймворк Scrum – це ітеративний та інкрементальний підхід, орієнтований на зміни та швидку доставку цінності. Процес розбивається на короткі цикли – спринти (зазвичай 1-4 тижні), наприкінці кожного з яких замовник отримує робочий продукт (інкремент). Основні ролі: Product Owner (Власник продукту), Scrum Master (Скрам-майстер), Development Team (Команда розробки). Відрізняється гнучкістю, постійною комунікацією та адаптивністю до змін вимог.

Основні елементи:

- ***Product Backlog:*** Список усіх функцій проєкту.

- ***Sprint:*** Короткий цикл розробки (1-4 тижні), результатом якого є готовий інкремент продукту.
- ***Scrum-ролі:*** Product Owner (визначає пріоритети), Scrum Master (допомагає команді), Development Team (виконує роботу).

2. Загальне завдання та покрокове виконання

Завдання: Спланувати розробку інформаційної системи "Онлайн-бібліотека" двома способами.

Крок 1. Аналіз вимог (WBS - Work Breakdown Structure)

Складіть перелік основних завдань та задайте умовно кількість робочих днів на виконання:

1. Розробка ТЗ та архітектури (10 днів).
2. Створення бази даних (5 днів).
3. Розробка Front-end (інтерфейс) (15 днів).
4. Розробка Back-end (логіка) (15 днів).
5. Інтеграція та тестування (10 днів).

Крок 2. Побудова діаграми Ганта (Waterfall)

Завдання: Розподілити виконання проєкту за класичними етапами каскадної моделі. Для кожного етапу записати 1-2 конкретні дії та очікуваний результат (артефакт).

Пояснення: Визначте залежності. Наприклад, тестування не почнеться без розробки.

Виконання: Побудуйте діаграму Ганта використавши MS Project, Excel або онлайн-сервіс (наприклад, GanttPRO/GanttProject). Розмістіть завдання послідовно ("драбинкою"). Позначте критичний шлях.

Крок 3. Формування Product Backlog (Scrum)

Завдання: Сформувати Product Backlog.

Пояснення: Перетворіть завдання з *Крок 1* на User Stories (Користувацькі історії).

Приклад: "Я, як користувач, хочу реєструватися, щоб мати доступ до книг".

Виконання: Складіть список із мінімум 5 User Stories та оцініть їх у Story Points (відносна складність).

Крок 4. Планування спринтів

Завдання: Розділіть беклог на 2–3 спринти.

Виконання: Сформуйте таблицю "Sprint 1", "Sprint 2". У перший спринт винесіть найкритичніші функції (MVP — Minimum Viable Product), наприклад: авторизація та пошук книг.

Крок 4. Порівняльний аналіз

Завдання: Заповнити таблицю порівняння двох підходів.

Критерій оцінки	Waterfall	Scrum
Реакція на зміну вимог	Жорстка, потребує перегляду ТЗ	Гнучка, додавання до Backlog
Швидкість першого релізу	В кінці всього циклу розробки	В кінці 1-го або 2-го спринту
Залученість клієнта	Тільки на початку та в кінці	Постійна (на демо-зустрічах)
Ризик невдачі	Високий (через довгий фідбек)	Знижений (регулярна перевірка)

3. Контрольні запитання

1. У чому полягає принципова різниця між ітеративним (Scrum) та послідовним (Waterfall) підходами до розробки?
2. Які артефакти створюються на кожному етапі каскадної моделі?
3. Опишіть основні ролі у фреймворку Scrum та їхні зони відповідальності.
4. Що таке Product Backlog та Sprint Backlog? У чому їхня відмінність?
5. В яких випадках використання методології Waterfall є більш виправданим, ніж Scrum?

4. Індивідуальні завдання

За зразком загального завдання виконайте індивідуальне завдання відповідно до варіанту:

1. Побудувати діаграму Ганта (тривалість проєкту – 3-4 місяці).
2. Сформувати Product Backlog (мінімум 12 історій).
3. Розподілити завдання мінімум на 3 спринти.

Перелік тем для індивідуального проєкту:

1. Система обліку пацієнтів лікарні.
2. Інтернет-магазин електроніки.
3. Автоматизація складського обліку.
4. Система бронювання готельних номерів.
5. Мобільний додаток для фітнес-тренувань.

6. Платформа для пошуку репетиторів.
7. Система доставки їжі з ресторанів.
8. Портал для обліку комунальних послуг.
9. Веб-сайт туристичної агенції.
10. Система управління задачами (To-Do).
11. Сервіс онлайн-запису до барбершопу.
12. Система обліку запчастин СТО.
13. Портал для внутрішньої комунікації компанії.
14. Сервіс порівняння характеристик авто.
15. Додаток для обліку особистих фінансів.
16. Додаток для замовлення таксі.
17. Платформа для вебінарів та онлайн-курсів.
18. Соціальна мережа для фотографів.
19. Сервіс оренди житла (подобово).
20. Система моніторингу цін конкурентів.
21. Електронний журнал для школи.
22. Чат-бот для технічної підтримки банку.
23. CRM-система для малого бізнесу.
24. Додаток для відстеження курсу валют.
25. Платформа для фріланс-біржі.
26. Мобільна гра "Вікторина".
27. Інтернет-аптека з доставкою.
28. Сайт для притулку тварин (збір коштів).
29. Система електронного голосування.
30. Платформа для онлайн-аукціонів.

5. Зміст звіту

1. Титульний лист.
2. Скріншот діаграми Ганта.
3. Скріншот Product Backlog (мінімум 12 історій).
4. Скріншот розподілу завдань мінімум по 3 спринтам.
5. Відповіді на контрольні запитання.
6. Висновок.

Лабораторна робота №5

Тема. Система управління проєктами Jira (або Trello). Візуалізація робочого процесу.

Мета: Навчитися використовувати цифрові інструменти для управління розробкою ПЗ, створювати дошки завдань, керувати беклогом та налаштовувати робочі процеси (Workflows).

1. Теоретичні відомості

Управління проєктом в IT базується на візуалізації.

- ***Backlog:*** Список усіх задач, які потрібно зробити.
- ***Issue (Тікет):*** Одиниця роботи. В Jira це може бути *Story* (функція), *Task* (технічна задача) або *Bug* (помилка).
- ***Board (Дошка):*** Візуалізація стану задач. Зазвичай має колонки: To Do (Зробити), In Progress (В роботі), Review/Testing (Перевірка) та Done (Готово).
- ***Workflow:*** Шлях, який проходить задача від створення до завершення.

2. Завдання

1. ***Налаштування:*** Створити безкоштовний акаунт на *Atlassian Jira* або *Trello*. Створити проєкт (тип: "Software Development", шаблон: "Scrum" або "Kanban").
2. ***Наповнення беклогу:*** Перенести 10 User Stories, розроблених у Лабораторній №4, у систему як окремі тікети (Issues).
3. ***Деталізація:*** Для 3-х обраних задач додати:
   - Детальний опис (Description).
   - Пріоритет (High, Medium, Low).
   - Оцінку (Story Points або Label).
   - Чек-ліст підзадач.
4. ***Управління спринтом:***
   - Створити "Sprint 1".
   - Перетягнути вибрані задачі з Backlog у Sprint.
   - "Запустити" спринт.

5. ***Візуалізація прогресу:*** Перемістити декілька задач по колонках (наприклад, одну в "In Progress", одну в "Done").

3. Контрольні запитання

1. Чим відрізняється Scrum-дошка від Kanban-дошки в інтерфейсі Jira?
2. Що таке "WIP-ліміти" (Work In Progress) і для чого вони потрібні в Kanban?
3. Яку інформацію має містити ідеально оформлений "Bug report" у Jira?
4. Для чого потрібна функція "Assignee" (Виконавець)?
5. Що таке "Burn-down chart" і яку інформацію він надає менеджеру?

4. Індивідуальні завдання

Виконати налаштування в Jira компоненту або розширити систему для свого проєкту (теми з Лабораторної роботи №4):

1. Створити спеціальну мітку (Label) "Urgent" і відфільтрувати задачі за нею.
2. Налаштувати автоматичне призначення (Assignee) себе на всі створені задачі.
3. Додати кастомне поле "Risk Level" у налаштування тікета.
4. Створити Bug-тікет, який описує помилку реєстрації, і зв'язати його з відповідною User Story (Link issues).
5. Налаштувати колонку "Blocked" для задач, які не можуть рухатися далі.
6. Прикріпити макет (будь-яке зображення) до тікета з розробки інтерфейсу.
7. Створити Epic (Епік) "Реліз 1.0" і прив'язати до нього всі задачі.
8. Налаштувати дедлайн (Due Date) для трьох найважливіших задач.
9. Прокоментувати задачу від імені "тестувальника", вказавши на помилку.
10. Створити підзадачі (Sub-tasks) для задачі "Авторизація користувача".
11. Використати фільтр JQL (Jira Query Language) для пошуку всіх задач з пріоритетом High.
12. Налаштувати автоматичне надсилання повідомлення про завершення задачі (якщо дозволяє версія).
13. Створити задачу типу "Design" і додати лінк на зовнішній ресурс (наприклад, Google Drive).
14. Налаштувати версії продукту (Versions/Releases) і призначити задачі до "v1.0".

15. Створити чек-ліст "Definition of Done" в описі проєкту.
16. Оформити тікет на "Refactoring" і пояснити його технічну необхідність.
17. Змінити порядок колонок на дошці, додавши етап "QA Testing".
18. Проставити "Estimate" (оцінку часу) у годинах замість Story Points для порівняння.
19. Додати Story, яка була "відхилена" (статус Rejected/Cancelled).
20. Створити залежність: задача Б не може бути почата, поки не завершена задача А.
21. Налаштувати Dashboard (панель приладів) з графіком залишку задач.
22. Додати в опис тікета форматування (таблицю або шматок коду).
23. Створити тікет для проведення мітингу (Meeting Task) з порядком денним.
24. Зробити скріншот "Log work" (облік часу), витраченого на задачу.
25. Створити задачу типу "Improvement" для вже існуючої функції.
26. Налаштувати фільтр "My Tasks" для відображення тільки своїх задач.
27. Додати в опис тікета User Persona (портрет користувача).
28. Змоделювати ситуацію: перенести задачу з "Done" назад в "In Progress" через баг.
29. Додати компонент (Component) "Frontend" та "Backend" до відповідних задач.
30. Налаштувати пріоритезацію задач методом "Drag and Drop" у беклозі.

5. Зміст звіту

1. Титульний лист.
2. Назва та посилання на проєкт у Jira/Trello (якщо він публічний) або скріншоти.
3. Скріншот Backlog із 10-ма задачами.
4. Скріншот Active Sprint / Board, де видно рух задач по колонках.
5. Скріншот одного детально оформленого тікета (з описом, пріоритетом та підзадачами).
6. Відповіді на контрольні запитання.
7. Висновок: наскільки зручніше керувати проєктом у системі порівняно з паперовим списком?

Лабораторна робота №6

Тема. Планування та розробка проєкту в команді. Розподіл ролей та організація процесів.

Мета: Навчитися формувати команду для IT-проєкту, розподіляти функціональні обов'язки, створювати командний статут та опанувати інструменти для синхронізації спільної роботи.

1. Теоретичні відомості

Ефективна команда базується на чіткому розподілі ролей:

- ***Project Manager (PM):*** стежить за дедлайнами та комунікацією.
- ***Business Analyst (BA):*** формулює вимоги.
- ***Frontend/Backend Developers:*** реалізують технічну частину.
- ***QA Engineer:*** перевіряє якість.
- ***UI/UX Designer:*** відповідає за зручність та вигляд.

Для успіху важливо визначити *Team Agreement* (правила гри): де відбувається спілкування, як часто проводяться збори (мітинги) та як вирішують конфлікти.

2. Завдання

1. ***Формування команди:*** Об'єднатися у групи по 3-5 осіб (або змоделювати команду самостійно згідно з варіантом).
2. ***Розподіл ролей:*** Призначити кожному учаснику роль. Написати короткий список обов'язків для кожного (що саме ця людина робить у проєкті).
3. ***Створення Team Agreement:*** Оформити текстовий документ, який містить:
   - Канали зв'язку (напр., Microsoft Teams, Discord, Slack, Telegram).
   - Графік зустрічей (напр., Daily Sync о 10:00).
   - Процедура Code Review (хто перевіряє код перед злиттям у main).
4. ***Командний Backlog:*** Вибрати один із проєктів Лабораторної роботи №4, який належить учаснику команди та розподілити перші 5 задач між учасниками команди в Jira/Trello.
5. ***Вирішення конфлікту (Кейс):*** Описати алгоритм дій команди у ситуації, вказаній у варіанті завдання.

3. Контрольні запитання

1. Що таке "Bus Factor" і чому команді важливо його підвищувати?
2. Чим роль Scrum-майстра відрізняється від ролі Project Manager?
3. Що таке "крос-функціональна команда"?
4. З якою метою створюється Team Agreement?
5. Як "Ретроспектива" допомагає команді ставати кращою після кожного спринту?

4. Індивідуальні завдання

Вирішення командних конфліктів. Опишіть (письмово), як ви, як команда, вирішите наступну ситуацію:

1. Розробник зник зі зв'язку за 2 дні до дедлайну, його код не завантажений.
2. Дизайнер та програміст не можуть домовитися про колір кнопок.
3. QA знайшов критичний баг, але розробник каже "на моєму комп'ютері все працює".
4. Замовник вимагає додати нову функцію посеред спринту.
5. Двоє учасників команди хочуть бути PM-ами одночасно.
6. PM призначив занадто багато задач розробнику, він не встигає.
7. Команда постійно запізнюється на щоденні мітинги.
8. Один з учасників робить коміти прямо в гілку main, ламаючи код іншим.
9. Дизайнер малює занадто складні елементи, які неможливо реалізувати швидко.
10. Команда використовує різні IDE, що спричиняє помилки в кодуванні файлів.
11. В одного з учасників зламався ноутбук, він випадає з процесу на тиждень.
12. Виявилося, що дві людини робили одну і ту саму задачу одночасно.
13. Команда не розуміє вимог, які написав Business Analyst.
14. Розробник Backend змінив API, не попередивши Frontend-розробника.
15. Учасник команди постійно критикує ідеї інших, не пропонуючи своїх.
16. Команда вирішила змінити мову програмування посеред проєкту.
17. PM вимагає працювати у вихідні, команда проти.
18. QA постійно пропускає дрібні баги, які потім бачить замовник.
19. Виник конфлікт при злитті гілок (Merge Conflict), який ніхто не може розв'язати.

20. Один учасник робить 90% всієї роботи, інші "відпочивають".
21. Команда не може вибрати назву для продукту вже тиждень.
22. Розробник використовує бібліотеку з платною ліцензією без дозволу.
23. З'ясувалося, що обраний стек технологій не підходить для реалізації проєкту.
24. Учасники команди живуть у різних часових поясах і не можуть знайти час для зборів.
25. Дизайнер зробив макети тільки для ПК, хоча додаток мобільний.
26. Команда витратила весь бюджет на дизайн, не залишивши на розробку.
27. Виникла суперечка: використовувати SQL чи NoSQL базу даних.
28. Один з розробників написав код, який ніхто інший не може зрозуміти (заплутаний код).
29. Команда забула зробити бекап, і частина коду була втрачена.
30. Замовник незадоволений швидкістю роботи і погрожує скасувати контракт.

5. Зміст звіту

1. Титульна сторінка.
2. Склад команди та розподіл ролей (ПІБ – Роль).
3. Текст Team Agreement.
4. Скріншот дошки в Jira/Trello, де видно, на кого призначені (Assigned) поточні задачі.
5. Письмовий аналіз ситуації з індивідуального завдання (план дій команди).
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №7

Тема. Система керування версіями Git: робота з локальним репозиторієм.

Мета: Опанувати базові команди Git, навчитися відстежувати зміни у файлах, створювати точки збереження (коміти) та навчитися переміщатися по історії проєкту.

1. Теоретичні відомості

Git не просто копіює файли, він зберігає знімки (snapshots) стану всього проєкту. Життєвий цикл файлу в Git складається з трьох основних зон:

1. ***Working Directory:*** Тека на диску, де ви редагуєте файли.
2. ***Staging Area (Index):*** Тимчасова зона, куди ви додаєте зміни, які готові потрапити в наступний коміт.
3. ***Local Repository (.git):*** База даних, де Git зберігає всю історію та метадані.

2. Завдання

1. ***Конфігурація:*** Налаштувати глобальні параметри користувача (ім'я та email).
2. ***Ініціалізація:*** Створити нову робочу директорію та перетворити її на Git-репозиторій за допомогою git init.
3. ***Фіксація змін:***
   - Створити файл відповідно до варіанта індивідуального завдання.
   - Додати його до індексу (git add).
   - Зробити перший коміт (git commit).
4. ***Історія та статус:*** Змінити вміст файлу, переглянути різницю (git diff), переглянути статус репозиторію (git status) та лог комітів (git log).
5. ***Ігнорування:*** Створити файл .gitignore і налаштувати його так, щоб Git "не бачив" тимчасові файли (наприклад, .log або .tmp).

3. Контрольні запитання

1. Чим git status відрізняється від git log?
2. Що станеться, якщо змінити файл після git add, але перед git commit?
3. Яка команда дозволяє побачити різницю між робочою директорією та останній комітом?
4. Навіщо кожному коміту потрібен унікальний SHA-1 хеш?

5. Як видалити файл з репозиторію, але залишити його на диску?

4. Індивідуальні завдання

Створіть локальний репозиторій для міні-проєкту. Виконайте мінімум 3 коміти, кожен з яких відображає етап розробки (створення структури, наповнення контентом, виправлення помилки).

1. Проєкт "RecipeBook": створення файлів рецептів у форматі .txt.
2. Проєкт "MyPlaylist": список пісень та авторів.
3. Проєкт "Inventory": облік техніки в офісі.
4. Проєкт "Contacts": телефонна книга.
5. Проєкт "Timetable": розклад занять на тиждень.
6. Проєкт "BudgetTracker": витрати за місяць.
7. Проєкт "WeatherStation": лог температури за тиждень.
8. Проєкт "BookCatalog": список прочитаних книг.
9. Проєкт "GymJournal": план тренувань.
10. Проєкт "MovieWatchlist": фільми до перегляду.
11. Проєкт "CodeSnippets": колекція корисних функцій.
12. Проєкт "GameCharacters": опис персонажів гри.
13. Проєкт "TravelLog": список країн та міст.
14. Проєкт "StudyNotes": конспект з дискретної математики.
15. Проєкт "HardwareSpec": характеристики вашого ПК.
16. Проєкт "TodoAdvanced": список задач з пріоритетами.
17. Проєкт "CoffeeMenu": асортимент кав'ярні.
18. Проєкт "CarService": журнал обслуговування авто.
19. Проєкт "SmartHome": перелік датчиків та їх станів.
20. Проєкт "LanguageLearning": словник нових іноземних слів.
21. Проєкт "PetHealth": календар щеплень тварини.
22. Проєкт "PlantCare": графік поливу квітів.
23. Проєкт "ProjectIdeas": банк ідей для стартапів.
24. Проєкт "Wishlist": список бажаних покупок.
25. Проєкт "CryptoPortfolio": перелік монет та їх закупівлі.
26. Проєкт "MeetingNotes": протоколи зборів групи.
27. Проєкт "E-library": посилання на PDF-підручники.
28. Проєкт "ChefSecrets": поради з кулінарії.
29. Проєкт "OldArchives": опис старих сімейних фото.
30. Проєкт "MusicTheory": таблиця акордів та гам.

5. Зміст звіту

1. Титульний лист.
2. Скріншот консолі з історією команд (використовуйте history або просто зробіть знімок вікна терміналу).
3. Результат команди git log --oneline --graph.
4. Вміст файлу .gitignore.
5. Короткий опис того, що саме було зроблено в кожному коміті.
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №8

Тема. Робота з Git. Віддалені репозиторії, розгалуження та розв'язання конфліктів.

Мета: Навчитися працювати з віддаленими репозиторіями (GitHub), опанувати стратегію розгалуження (Branching) та отримати практичні навички вирішення конфліктів під час злиття коду (Merge Conflicts).

1. Теоретичні відомості

- ***Remote (Віддалений репозиторій):*** копія проєкту, що зберігається на сервері (GitHub/GitLab). Команда push відправляє зміни на сервер, pull – забирає їх.
- ***Branch (Гілка):*** паралельна версія репозиторію. Вона дозволяє розробляти нові функції, не псуючи основний код (гілку main або master).
- ***Merge Conflict (Конфлікт злиття):*** ситуація, коли в різних гілках змінили один і той самий рядок в одному файлі. Git не знає, яку версію залишити, і просить розробника зробити вибір вручну.

2. Завдання

1. ***Створення зв'язку*****:** Створити публічний репозиторій на **GitHub**. Пов'язати локальний репозиторій з Лабораторної роботи №7 з віддаленим репозиторієм (git remote add origin ...).
2. ***Розгалуження:*** Створити нову гілку з назвою feature-updates (git checkout -b feature-updates).
3. ***Робота в гілці:*** У новій гілці внести зміни у файл проєкту (згідно з варіантом індивідуального завдання) та зробити коміт. Відправити гілку на GitHub (git push origin feature-updates).
4. ***Моделювання конфлікту:***
   - Перемкнутися назад у main. Змінити той самий рядок у тому самому файлі. Зробити коміт.
   - Спробувати злити гілку feature-updates у main (git merge feature-updates).
   - Отримати конфлікт.

5. ***Вирішення конфлікту:*** Відкрити файл, обрати фінальний варіант коду, видалити маркери конфлікту (<<<<<, =====, >>>>>), завершити злиття комітом.
6. ***Pull Request:*** Створити Pull Request на GitHub (симуляція пропозиції змін).

3. Контрольні запитання

1. Чим git fetch відрізняється від git pull?
2. Для чого потрібні гілки в реальних комерційних проєктах?
3. Як виглядають маркери конфлікту у файлі під час невдалого злиття?
4. Що таке origin у команді git push origin main?
5. Як видалити гілку локально та на віддаленому сервері?

4. Індивідуальні завдання

Кожен варіант передбачає специфічний сценарій конфлікту у файлах проєкту, розпочатого в Лабораторній роботі №7:

1. Конфлікт у заголовку: Зміна назви проєкту в main та feature.
2. Конфлікт у списку: Одночасне додавання різних пунктів на 5-й рядок.
3. Конфлікт у контактних даних: Різні номери телефону для однієї особи.
4. Конфлікт у README: Різні описи мети проєкту.
5. Конфлікт у розкладі: Різні аудиторії для однієї пари.
6. Конфлікт у бюджеті: Різні суми витрат за один і той самий день.
7. Конфлікт дат: Різні формати дати в логах (DD.MM vs MM.DD).
8. Конфлікт цін: Різні ціни на один товар у меню/каталозі.
9. Конфлікт метаданих: Зміна версії проєкту (v1.0 vs v1.1).
10. Конфлікт логіки: Різні умови в if-елементі.
11. Конфлікт коментарів: Різні пояснення до однієї функції.
12. Конфлікт у CSS-стилях: Різні кольори фону для одного елемента.
13. Конфлікт посилань: Різні URL-адреси для одного ресурсу.
14. Конфлікт у списку авторів: Різний порядок імен.
15. Конфлікт у конфігурації: Різні налаштування порту (8080 vs 3000).
16. Конфлікт пріоритетів: Завдання позначено як "High" та "Low".
17. Конфлікт інгредієнтів: Зміна кількості цукру в рецепті.
18. Конфлікт координат: Різні значення широти/довготи для локації.
19. Конфлікт імен файлів: Спроба перейменувати файл у різні назви.
20. Конфлікт ліцензії: MIT vs Apache у файлі LICENSE.

21. Конфлікт перекладу: Одне слово перекладено по-різному.
22. Конфлікт видалення: Один видалив рядок, інший його відредагував.
23. Конфлікт табуляції: Змішування пробілів та табів (у чутливих мовах).
24. Конфлікт у SQL-схемі: Різні типи даних для одного поля.
25. Конфлікт паролів: Зміна пароля до тестової бази даних у різних гілках.
26. Конфлікт структури JSON: Додавання різних полів у один об'єкт.
27. Конфлікт кодування: UTF-8 vs ASCII (симуляція).
28. Конфлікт у хедері сайту: Різне меню навігації.
29. Конфлікт скриптів: Різні версії підключених бібліотек.
30. Конфлікт у звіті: Різні висновки в підсумковому файлі.

5. Зміст звіту

1. Титульний лист.
2. Посилання на публічний GitHub-репозиторій.
3. Скріншот терміналу в момент виникнення конфлікту (повідомлення CONFLICT (content): Merge conflict in...).
4. Скріншот файлу з маркерами конфлікту до його вирішення.
5. Опис: як саме ви вирішили конфлікт (яку версію обрали та чому).
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №9

Тема. Основи розробки вебсайтів. Семантична розмітка за допомогою HTML5.

Мета: Ознайомитися зі структурою HTML-документа, вивчити основні теги розмітки та навчитися створювати семантично правильні вебсторінки.

1. Теоретичні відомості

HTML (HyperText Markup Language) – це мова розмітки, що використовується для створення структури та вмісту вебсторінок. Вона визначає, як браузер відображає контент (текст, зображення, посилання), використовуючи систему тегів.

- ***Теги:*** Ключові слова, взяті в кутові дужки (наприклад, <h1>). Більшість тегів є парними.
- ***Семантика:*** Використання тегів відповідно до їхнього змісту (<header>, <nav>, <main>, <footer>). Це важливо для пошукових систем (SEO) та людей з обмеженими можливостями (Accessibility).
- ***Атрибути:*** Додаткова інформація про тег (наприклад, <img src="logo.png">, де src – атрибут).

2. Завдання

1. ***Підготовка:*** Створити каталог проєкту та додати в нього файл index.html. Відкрити його в редакторі коду (наприклад, VS Code).
2. ***Базова структура:*** Створити стандартний "скелет" документа. Для прискорення роботи в VS Code можна згенерувати базовий шаблон HTML5 скориставшись Emmet-скороченням: введіть знак оклику (!) та натисніть Tab.
3. ***Розмітка:*** Використовуючи семантичні теги, створити структуру сторінки згідно з варіантом. Вона повинна містити:
   - Заголовок першого рівня (<h1>).
   - Навігаційне меню (<nav>) зі списком посилань (<ul>, <li>, <a>).
   - Основний текстовий блок (<p>, ,).
   - Зображення (<img>) та таблицю (<table>) або форму (<form>).
4. ***Валідація:*** Перевірити свій код на наявність помилок через W3C Markup Validation Service *(https://validator.w3.org)*.

3. Контрольні запитання

1. У чому різниця між тегами та <section>?
2. Навіщо потрібен атрибут alt у тегу <img>?
3. Яка роль тегу <head> і чи бачить користувач його вміст на сторінці?
4. Чим відрізняється нумерований список (<ol>) від маркованого (<ul>)?
5. Що таке "абсолютний" та "відносний" шлях до файлу?

4. Індивідуальні завдання

Створити структуру сторінки (без оформлення стилями) для наступних об'єктів:

1. Персональна візитка: ПІБ, фото, короткий опис "Про себе", посилання на соцмережі.
2. Рецепт страви: Назва, інгредієнти (таблиця), кроки приготування (нумерований список).
3. Розклад занять: Заголовок та таблиця на 6 днів тижня.
4. Картка товару: Зображення, ціна, характеристики (список), кнопка "Купити".
5. Форма реєстрації: Поля для імені, пошти, пароля та вибору статі (radio buttons).
6. Новинна стаття: Заголовок, дата, автор, текст із декількох абзаців та фото.
7. Список улюблених фільмів: Таблиця з назвою, роком та рейтингом.
8. Прайс-лист послуг: Опис послуг та ціни у форматі списку визначень (<dl>).
9. Сторінка "Контакти": Адреса, карта (фрейм або посилання) та форма зворотного зв'язку.
10. Галерея фотографій: Сітка з мініатюр з підписами (<figure>, <figcaption>).
11. Сторінка відгуків: Список коментарів з іменем користувача та текстом відгуку.
12. FAQ (Питання-відповіді): Використання тегів <details> та <summary>.
13. Плейлист пісень: Таблиця з назвою треку, тривалістю та посиланням на прослуховування.
14. Анкета опитування: Питання з варіантами вибору (checkboxes) та полем для розгорнутої відповіді.
15. Технічні характеристики ноутбука: Детальна таблиця порівняння параметрів.

16. Програма конференції: Час, назва доповіді та спікер (структурований список).
17. Сторінка "Команда": Блоки з фото співробітників, їх посадами та біографією.
18. Портфоліо проєктів: Список посилань на проєкти з короткими описами.
19. Замовлення піци: Форма з вибором розміру, додатків та адреси доставки.
20. Блог-пост про подорож: Заголовок, цитати (<blockquote>) та маркований список порад.
21. Сторінка підписки: Опис тарифних планів (Basic, Pro, Enterprise) у вигляді колонок.
22. Календар подій: Список подій на місяць з використанням тегу <time>.
23. Меню ресторану: Розділи (Сніданки, Обіди) з описом страв та калорійністю.
24. Сторінка завантаження ПЗ: Кнопки для різних ОС (Windows, Mac, Linux) та опис версії.
25. Шаблон листа (Email): Проста структура з логотипом, привітанням та підвалом.
26. Опис гри: Жанр, системні вимоги (список) та опис сюжету.
27. Сторінка "404 Помилка": Повідомлення про помилку та посилання на головну сторінку.
28. Інструкція користувача: Кроки з використанням вкладених списків.
29. Порівняння двох гаджетів: Таблиця з двома стовпчиками характеристик.
30. Пошукова сторінка: Поле введення та імітація списку результатів пошуку.

5. Зміст звіту

1. Титульний лист
2. Скріншот коду у редакторі VS Code.
3. Скріншот відображення сторінки у браузері.
4. Посилання на валідатор з результатом перевірки (скріншот "Document checking completed. No errors or warnings to show").
5. Коротке пояснення: які семантичні теги були використані та чому.
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №10

Тема. Стилізація вебсторінок за допомогою CSS3. Робота з Flexbox та блоковою моделлю.

Мета: Ознайомитися з підключенням стилів до HTML-документа, опанувати роботу з селекторами, кольорами, шрифтами та навчитися будувати структуру простого макету за допомогою технології Flexbox.

1. Теоретичні відомості

- ***Селектори:*** Вказують, до яких елементів застосувати стиль (використовують селектор тегу, класу або ідентифікатору).
- ***Блокова модель (Box Model):*** Кожен елемент – це контейнер прямокутної форми, що має вміст (content), внутрішні відступи (padding), рамку (border) та зовнішні відступи (margin).
- ***Flexbox:*** Модель верстки, яка дозволяє легко вирівнювати елементи в ряд або стовпчик та розподіляти простір між ними.

2. Завдання до виконання

1. ***Підключення:*** В каталог проєкту лабораторної роботи №9 додати файл style.css та підключити його до файлу index.html за допомогою тегу <link>.
2. ***Шрифти та кольори:***
   - Підключити шрифт із Google Fonts.
   - Налаштувати основний колір фону та тексту (використовуйте HEX або RGB коди).
3. ***Блокова модель:*** Для основних блоків сторінки налаштувати padding, margin та border. Додати ефект при наведенні (:hover) на кнопки або посилання.
4. ***Flexbox верстка:*** Використати display: flex, щоб:
   - Вирівняти пункти меню в один рядок.
   - Розмістити контент та бокову панель (при її наявності) поруч.
   - Центрувати елементи всередині контейнера.

3. Контрольні запитання

1. У чому різниця між margin та padding?
2. Який селектор має вищий пріоритет: за класом чи за ідентифікатором (id)?
3. Що робить властивість display: flex;?
4. Навіщо потрібна властивість box-sizing: border-box;?
5. Як змінити колір посилання тільки тоді, коли на нього наведений курсор миші?

4. Індивідуальні завдання

Кожен варіант передбачає стилізацію структури вебсторінки з Лабораторної роботи №9 за певною колірною схемою та стилістикою:

1. Стиль "Dark Mode": Темно-сірий фон, неонові зелені акценти, шрифти Monospace.
2. Стиль "Minimalism": Білий фон, багато простору (великі margin), тонкі рамки, шрифт Sans-serif.
3. Стиль "Retro": Бежевий фон, коричневий текст, закруглені кути (border-radius), шрифт Serif.
4. Стиль "Business": Темно-синій (Navy) та білий кольори, чіткі лінії, шрифт Roboto.
5. Стиль "Nature": Відтінки зеленого та коричневого, м'які тіні (box-shadow).
6. Стиль "High-Tech": Скляний ефект (напівпрозорість rgba), градієнтні фони.
7. Стиль "Pastel": Ніжні кольори (рожевий, блакитний), відсутність гострих кутів.
8. Стиль "Cyberpunk": Фіолетовий фон, яскраво-рожевий текст, світіння елементів.
9. Стиль "Classic News": Білий фон, засічки у шрифтів, чорні роздільні лінії.
10. Стиль "Mobile First": Вертикальне розташування всіх блоків, великі кнопки.
11. Стиль "Ocean": Блакитні та бірюзові кольори, хвилясті межі (якщо можливо).
12. Стиль "Eco": Салатовий колір, використання іконок (за бажанням), шрифт Open Sans.

13. Стиль "Gold & Luxury": Чорний фон, золоті рамки та шрифти.
14. Стиль "Kids": Яскраві контрастні кольори (червоний, жовтий), великі шрифти.
15. Стиль "Autumn": Помаранчевий, бордовий кольори, теплі відтінки.
16. Стиль "Winter": Світло-блакитний, білий кольори рамки.
17. Стиль "Coffee Shop": Кавові відтінки, текстурні фони.
18. Стиль "Space": Дуже темний фон, білі "крапки" (зірки), футуристичні шрифти.
19. Стиль "Paper": Ефект паперу (легка жовтизна), шрифт "друкарська машинка".
20. Стиль "Sunset": Градієнт від помаранчевого до фіолетового.
21. Стиль "Flat Design 2.0": Яскраві кольори, але без тіней та градієнтів.
22. Стиль "Material Design": Виражені тіні для створення глибини.
23. Стиль "Neon Night": Чорний фон, сині та червоні неонові рамки.
24. Стиль "Art Gallery": Тонкі лінії, акцент на великих зображеннях.
25. Стиль "Sports": Агресивні кольори (червоний/чорний), нахилені шрифти (italic).
26. Стиль "University": Класичний стиль: бордовий колір, строгі таблиці.
27. Стиль "Startup": Енергійний фіолетовий, заокруглені кнопки, великі заголовки.
28. Стиль "Medical": Світло-блакитний, білий, відчуття стерильності.
29. Стиль "Candy": Рожевий, м'ятний кольори, "солодкий" дизайн.
30. Стиль "Concrete": Сірі відтінки, грубі форми, індустріальний стиль.

5. Зміст звіту

1. Титульний лист.
2. Скріншот HTML-коду з підключеним CSS.
3. Скріншот повного файлу style.css.
4. Скріншот підсумкової сторінки в браузері (порівняйте "до" та "після" стилізації).
5. Опис: які саме властивості Flexbox ви використали для вирівнювання елементів.
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №11

Тема. Вступ до GameDev. Проєктування ігрових механік та створення концепт-документа (GDD).

Мета: Ознайомитися з циклом розробки відеоігор, основними жанрами та рушіями. Навчитися формулювати ігрові механіки та створювати базовий документ ігрового дизайну (Game Design Document).

1. Теоретичні відомості

Розробка гри починається з ідеї, яка фіксується у GDD. Основні складові гри:

- ***Core Loop (Основний ігровий цикл):*** Повторювана серія дій гравця (наприклад: знайти ворога → перемогти → отримати досвід → покращити героя → знайти сильнішого ворога).
- ***Механіки:*** Правила, за якими гравець взаємодіє зі світом (стрибки, стрільба, торгівля).
- ***Сетинг:*** Світ, де відбувається дія (фентезі, кіберпанк, космос).
- ***Game Engine (Ігровий рушій):*** Платформа для розробки (Unity, Unreal Engine, Godot).

2. Завдання

1. ***Концепція:*** Згідно з варіантом індивідуального завдання, визначити назву гри, її жанр та основну мету.
2. ***Проєктування Core Loop:*** Намалювати або описати схему основного ігрового циклу.
3. ***Опис механік:*** Визначити 3 основні механіки (як гравець перемагає, як програє, як прогресує).
4. ***Монетизація та платформа:*** Вибрати, де буде випущена гра (PC, Mobile, Console) та як вона приноситиме дохід (Free-to-play, Premium, Ads).
5. ***Візуалізація (Прототип):*** Створити текстовий опис або ескіз (схематичний малюнок) головного екрана гри.

3. Контрольні запитання

1. Чим ігровий рушій Unity відрізняється від Unreal Engine з точки зору мов програмування?
2. Що таке "геймплей" і чим він відрізняється від сюжету?
3. Яка роль ігрового дизайнера в команді розробки?
4. Що означає термін "MVP" (Minimum Viable Product) у контексті ігор?
5. Навіщо грі потрібен баланс і як його перевірити?

4. Індивідуальні завдання

Розробити концепцію (GDD) для гри за заданою комбінацією жанру та сеттингу:

1. Platformer: Постапокаліпсис, гра за робота-збирача сміття.
2. RPG: Стародавній Єгипет з елементами магії.
3. Racing: Перегони на підводних човнах у майбутньому.
4. Strategy: Керування колонією на Марсі.
5. Horror: Покинута космічна станція, де зник зв'язок.
6. Simulation: Життя студента-айтішника (симулятор виживання).
7. Puzzle: Маніпуляція часом для проходження лабіринтів.
8. Action/Slasher: Середньовіччя, де головний герой – привид.
9. Tower Defense: Захист імунної системи людини від вірусів.
10. Adventure: Детектив у стилі нуар у місті антропоморфних тварин.
11. Survival: Виживання на безлюдному острові з динозаврами.
12. Fighting: Битви міфічних істот різних народів світу.
13. Stealth: Гра за ніндзя в умовах сучасного мегаполіса.
14. Idle/Clicker: Будівництво власної IT-корпорації.
15. Battle Royale: Змагання магічних академій на літаючому острові.
16. First Person Shooter: Полювання на монстрів у паралельному вимірі.
17. Sports Sim: Футбол у невагомості.
18. Visual Novel: Романтична історія про мандрівників у часі.
19. Roguelike: Дослідження генерованих підземель алхіміком.
20. City Builder: Побудова міста під водою.
21. Card Game: Битви за допомогою карт, що представляють закони фізики.
22. Music/Rhythm: Гра, де бої відбуваються в ритм музики.
23. Hidden Object: Пошук доказів у квартирі подорожуючого між світами.
24. Open World: Дослідження джунглів на іншій планеті.
25. Cyberpunk Stealth: Злом систем безпеки мегакорпорацій.

26. Farm Sim: Вирощування фантастичних істот на фермі.
27. Tycoon: Керування мережею міжгалактичних готелів.
28. Bullet Hell: Космічний корабель проти нескінченних хвиль астероїдів.
29. Educational: Навчання математиці через алхімічні експерименти.
30. Text Quest: Психологічний трилер, де вибір впливає на фінал.

5. Зміст звіту

1. Титульний лист.
2. Назва та жанр гри.
3. Короткий опис сюжету та цілі гравця (1-2 абзаци).
4. Опис 3-х ключових механік.
5. Схема Core Loop.
6. Скетч або опис інтерфейсу (де розташовані кнопки, HP, карта тощо).
7. Відповіді на контрольні запитання.
8. Висновки.

Лабораторна робота №12

Тема. Технології працевлаштування: створення професійного CV та профілю LinkedIn.

Мета: Навчитися структурувати власний професійний досвід та навички, опанувати інструменти для створення резюме (CV) та принципи побудови професійної мережі в LinkedIn.

1. Теоретичні відомості

CV (Curriculum Vitae) для IT-фахівця має бути лаконічним (1-2 сторінки) та містити:

- ***Contact Info:*** Пошта, GitHub, LinkedIn, місто.
- ***Summary/Objective:*** Хто ви і що шукаєте.
- ***Tech Stack (Hard Skills):*** Мови програмування, інструменти, фреймворки.
- ***Projects/Experience:*** Що ви зробили (навіть навчальні проєкти).
- ***Education & Languages:*** Рівень англійської є критичним.

ATS (Applicant Tracking System) – це програма, яка автоматично сканує ваше CV на наявність ключових слів. Якщо їх немає – рекрутер навіть не побачить ваше резюме.

2. Завдання

1. ***Підготовка контенту:*** Виписати свої Hard Skills (що вже знаєте і що вивчите за семестр) та Soft Skills.
2. ***Створення CV:*** Використовуючи сервіси Canva, Standard Resume або Novoresume, створити своє перше професійне резюме.
3. ***Оформлення LinkedIn:***
   - Заповнити заголовок (Headline) за формулою: [Role] | [Key Technology] | [Interest/Goal].
   - Написати розділ "About", використовуючи ключові слова вашого варіанта.
   - Додати навички (Skills) та отримати/дати підтвердження (Endorsements) одногрупникам.

4. ***GitHub Link:*** Додати у CV посилання на свій GitHub, де вже є результати лабораторних робіт.

3. Контрольні запитання

1. Чому не варто додавати в CV шкалу відсотків володіння мовою (наприклад, "Java – 70%")?
2. У чому різниця між Hard Skills та Soft Skills? Наведіть приклади.
3. Що таке "Summary" у резюме і чим воно відрізняється від "Cover Letter"?
4. Навіщо в LinkedIn потрібні рекомендації (Recommendations)?
5. Як адаптувати резюме під конкретну вакансію?

4. Індивідуальні завдання

Адаптувати своє резюме під "вакансію мрії", зробивши акцент на відповідному стеку технологій та ключових словах:

1. Junior Frontend (React): Акцент на HTML, CSS, JS, React, Git.
2. Junior Java Developer: Spring Boot, SQL, JUnit.
3. Junior Python Dev: Django/Flask, REST API, Linux.
4. Junior QA Manual: Bug Tracking, Jira, Test Cases.
5. Junior Data Analyst: Python, SQL, Excel, Tableau.
6. Junior UI/UX Designer: Figma, Adobe Suite, Prototyping.
7. Junior Android Dev: Kotlin, Android Studio, API.
8. Junior iOS Dev: Swift, Xcode, CocoaPods.
9. Junior Game Dev: Unity, C#, Vector math.
10. Junior DevOps: Docker, Bash, Cloud basics.
11. Junior Cybersecurity: Network security, Kali Linux, OWASP.
12. Junior .NET Dev: C#, ASP.NET, Entity Framework.
13. Junior Node.js Dev: Express, MongoDB, WebSockets.
14. Junior PHP Dev: Laravel, MySQL, Composer.
15. Junior Flutter Dev: Dart, Cross-platform UI.
16. Junior Ruby Dev: Ruby on Rails, PostgreSQL.
17. Junior Golang Dev: Concurrency, Microservices.
18. Junior Business Analyst: BPMN, Agile, Requirements.
19. Junior Project Manager: Scrum, Trello, Soft Skills.
20. Junior Support Engineer: Troubleshooting, Communication.
21. Junior Embedded Dev: C, Microcontrollers, RTOS.
22. Junior Rust Dev: Memory safety, Cargo.

23. Junior Salesforce Dev: Apex, Visualforce.
24. Junior Machine Learning Eng: NumPy, Scikit-learn, Statistics.
25. Junior SEO Specialist: Google Analytics, Keyword research.
26. Junior Hardware Eng: Circuit design, Soldering, PCB.
27. Junior Technical Writer: Documentation, Markdown.
28. Junior Database Admin: Backup/Recovery, Query optimization.
29. Junior Fullstack (JS): MERN stack, Deployment.
30. Junior Cloud Architect: AWS/Azure/GCP certifications.

5. Зміст звіту

1. Титульний лист.
2. Файл резюме у форматі PDF.
3. Посилання на профіль у LinkedIn (перевірте, щоб профіль був публічним).
4. Скріншот блоку "Skills" у LinkedIn.
5. Список із 5 ключових слів (Keywords).
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №13

Тема. Етапи співбесід в IT-компаніях. Методика STAR для відповіді на поведінкові питання.

Мета: Ознайомитися з типовим процесом рекрутингу в IT, навчитися готувати презентацію «Elevator Pitch» та опанувати техніку STAR для проходження поведінкових (behavioral) інтерв'ю.

1. Теоретичні відомості

Типові етапи відбору в IT:

1. ***Screening call:*** 15-20 хв розмови з рекрутером (перевірка Soft Skills та англійської).
2. ***Technical Interview:*** Перевірка Hard Skills, знання алгоритмів та стеку.
3. ***Managerial/Team Fit Interview:*** Перевірка, чи підходите ви команді за цінностями.
4. ***Offer:*** Фінальна пропозиція роботи.

Метод STAR – це модель відповіді на питання типу «Розкажіть про випадок, коли…»:

- ***S (Situation):*** Опис ситуації.
- ***T (Task):*** Опис вашого завдання.
- ***A (Action):*** Опис дій, які ви здійснили.
- **R (Result):** Опис результату (бажано з конкретними показниками або досягненнями).

2. Завдання

1. ***Elevator Pitch:*** Підготувати коротку розповідь про себе (до 1 хвилини), яка відповідає на питання «Tell me about yourself».
2. ***Підготовка до технічних питань:*** Згідно з вашим стеком (індивідуальний варіант з Лабораторної роботи №12) знайти та виписати 5 найпопулярніших технічних питань для рівня Junior.
3. ***Застосування STAR:*** Вибрати ситуацію зі свого студентського або проєктного життя (згідно з варіантом) і розписати відповідь за схемою STAR.
4. ***Reverse Interviewing:*** Підготувати 3 питання, які ви поставите роботодавцю в кінці співбесіди (щоб показати свою зацікавленість).

3. Контрольні запитання

1. Що таке «Culture Fit» і чому він іноді важливіший за технічні навички?
2. Як правильно відповідати на питання «Які ваші слабкі сторони?»
3. Чим відрізняється технічна співбесіда від «скринінгу»?
4. Що робити, якщо ви не знаєте відповіді на технічне питання під час інтерв'ю?
5. Навіщо потрібно надсилати «Thank you note» після співбесіди?

4. Індивідуальні завдання

Завдання передбачає моделювання відповідей на запитання співбесіди, виходячи з припущення, що ви вже маєте професійний досвід. Підготуйте відповіді за методикою STAR (Situation – Task – Action – Result) на наведені запитання:

1. Розкажіть про випадок, коли ви припустилися помилки в проєкті.
2. Опишіть ситуацію, коли у вас був конфлікт з колегою/одногрупником.
3. Розкажіть про момент, коли вам довелося вивчати нову технологію в стислі терміни.
4. Опишіть свій досвід роботи над складним завданням, де не було чіткої інструкції.
5. Розкажіть про випадок, коли ви взяли на себе лідерську роль.
6. Опишіть ситуацію, коли ви не встигали до дедлайну.
7. Розкажіть про ваше найбільше технічне досягнення на сьогодні.
8. Опишіть випадок, коли вам довелося переконувати інших у своїй правоті.
9. Розкажіть, як ви справлялися з критикою вашої роботи.
10. Опишіть ситуацію, де вам довелося працювати з «важкою» людиною.
11. Розкажіть про випадок, коли ви зробили більше, ніж від вас очікували.
12. Опишіть ситуацію, коли ви допомогли іншому учаснику команди.
13. Розкажіть про момент, коли ви запропонували покращення робочого процесу.
14. Опишіть випадок, коли ви успішно розв'язали технічну проблему (баг), яку довго не могли знайти.
15. Розкажіть про ситуацію, коли вам довелося працювати в умовах багатозадачності.
16. Опишіть ваш досвід публічного виступу або презентації проєкту.

17. Розкажіть про випадок, коли ви змінили свою думку під впливом аргументів команди.
18. Опишіть ситуацію, коли вам довелося відмовити комусь у запиті через зайнятість.
19. Розкажіть про досвід роботи з негативним фідбеком від викладача або замовника.
20. Опишіть момент, коли ви проявили ініціативу, не чекаючи вказівки.
21. Розкажіть про ситуацію, коли ви працювали в команді, де не всі були вмотивовані.
22. Опишіть досвід адаптації до змін у проєкті на фінальній стадії.
23. Розкажіть про випадок, коли ви самостійно розібралися в чужому коді.
24. Опишіть ситуацію, де вам довелося пояснювати технічні речі нетехнічній людині.
25. Розкажіть про випадок, коли ви відчували професійне вигорання або сильну втому, і як з цим впоралися.
26. Опишіть ситуацію, коли ви були не згодні з рішенням менеджера/лідера.
27. Розкажіть про ваш досвід віддаленої роботи або навчання: як ви організували дисципліну.
28. Опишіть випадок, коли ви знайшли помилку в роботі старшого колеги.
29. Розкажіть про свій найнапруженіший робочий/навчальний день і як ви його пережили.
30. Опишіть ситуацію, коли результат проєкту був невдалим, і які висновки ви зробили.

5. Зміст звіту

1. Титульний лист.
2. Текст вашого **Elevator Pitch** (у письмовій формі).
3. Список з 5 технічних питань та коротких відповідей на них.
4. Опис ситуації за схемою STAR (4 чіткі блоки).
5. Список з 3-х ваших питань до роботодавця.
6. Відповіді на контрольні запитання.
7. Висновки.

Лабораторна робота №14

Тема. Юридичні аспекти в IT: офер, контракт, NDA та інтелектуальна власність.

Мета: Ознайомитися з основними типами документів, які підписує IT-фахівець при працевлаштуванні, навчитися аналізувати умови контрактів та розуміти відповідальність за нерозголошення інформації.

1. Теоретичні відомості

- ***Job Offer (Офер):*** Офіційна пропозиція роботи, де вказані позиція, зарплата, бонуси та дата виходу. Це ще не контракт, але юридично значущий документ.
- ***NDA (Non-Disclosure Agreement):*** Договір про нерозголошення конфіденційної інформації (коду, клієнтів, планів компанії).
- ***NCA (Non-Compete Agreement):*** Угода про неконкуренцію (обмеження на роботу в компаніях-конкурентах протягом певного часу).
- ***IP Rights (Intellectual Property):*** Пункт договору, який зазвичай вказує, що весь код, написаний вами в робочий час, належить компанії.
- ***ФОП (Фізична особа-підприємець):*** Популярніша форма співпраці в українському IT (модель B2B – бізнес для бізнесу).

2. Завдання

1. ***Аналіз оферу:*** Ознайомитися з шаблоном Job Offer та перевірити його на відповідність вашим очікуванням (згідно з варіантом індивідуального завдання).
2. ***Робота з NDA:*** Прочитати запропонований зразок договору про нерозголошення. Виділити 3 речі, які категорично заборонено робити після його підписання.
3. ***Порівняння моделей:*** Скласти коротку таблицю переваг та недоліків роботи за КЗпП (штатний працівник) та через ФОП (контрактор).
4. ***Практичний кейс:*** Виконати індивідуальне завдання щодо аналізу специфічного пункту контракту.

3. Контрольні запитання

1. Чи є Job Offer юридично зобов'язуючим документом в Україні?

2. Протягом якого терміну зазвичай діє NDA після звільнення з компанії?
3. Хто володіє правами на код, який ви написали вдома на вихідних, якщо ви працюєте в компанії?
4. Що таке "Notice period" (період попередження про звільнення) і яка його стандартна тривалість в IT?
5. Які основні податки сплачує IT-спеціаліст, що працює як ФОП 3-ї групи?

4. Індивідуальні завдання

Аналіз пунктів контракту. Проаналізуйте пункт договору та поясніть своїми словами: чи є він безпечним для вас, чи потребує обговорення з юристом?

1. Пункт про штраф у $5000 за публікацію скріншоту робочого чату в соцмережах.
2. Пункт про заборону працювати на конкурентів протягом 2 років після звільнення.
3. Пункт, де вказано, що роботодавець може змінювати графік роботи без попередження.
4. Пункт про повну матеріальну відповідальність за робочий ноутбук.
5. Пункт про передачу прав на всі ідеї, які виникли у вас під час терміну дії контракту.
6. Пункт про "випробувальний термін" тривалістю 6 місяців без оплати.
7. Пункт про неможливість розірвання контракту раніше ніж через рік під загрозою штрафу.
8. Пункт про автоматичне подовження контракту на той самий термін (Prolongation).
9. Пункт про дозвіл компанії використовувати ваше обличчя в рекламних матеріалах вічно.
10. Пункт про те, що зарплата виплачується в національній валюті без прив'язки до курсу.
11. Пункт про право компанії перевіряти ваш особистий комп'ютер, якщо ви працюєте Remote.
12. Пункт про заборону обговорювати рівень своєї зарплати з колегами.
13. Пункт про те, що всі відпустки мають бути погоджені за 3 місяці.
14. Пункт про штраф за запізнення на Daily Scrum більше ніж на 5 хвилин.
15. Пункт про відмову від права на інтелектуальну власність навіть на сторонні проєкти.

16. Пункт про територіальну юрисдикцію (наприклад, всі спори вирішуються в суді штату Делавер, США).
17. Пункт про обов'язкове використання трекерів активності (скріншоти екрана кожні 10 хв).
18. Пункт про заборону займатися фрілансом у вільний від основної роботи час.
19. Пункт про відшкодування компанії витрат на ваше навчання у разі звільнення раніше ніж через пів року.
20. Пункт про те, що Bonus (премія) виплачується лише на розсуд директора.
21. Пункт про нерозголошення факту самої співпраці з компанією (White Label).
22. Пункт про заборону переманювання (Poaching) колег у свій стартап.
23. Пункт про право компанії на одностороннє зменшення винагороди при форс-мажорі.
24. Пункт про необхідність повідомляти про звільнення за 3 місяці.
25. Пункт про те, що робота у вихідні дні вже включена у вартість контракту.
26. Пункт про передачу паролів від особистих акаунтів, створених для роботи.
27. Пункт про конфіденційність технічного стеку, що використовується в проєкті.
28. Пункт про заборону публічних виступів щодо досвіду роботи в компанії без згоди PR-відділу.
29. Пункт про те, що виплата за останній місяць проводиться через 60 днів після звільнення.
30. Пункт про згоду на відрядження тривалістю до 50% робочого часу.

5. Зміст звіту

1. Короткий конспект: 3 головні загрози в NDA, які ви знайшли.
2. Порівняльна таблиця: КЗпП з ФОП.
3. Аналіз індивідуального пункту (варіанта) з обґрунтуванням вашої позиції.
4. Відповіді на контрольні запитання.
5. Висновок: на що ви звернете увагу в першу чергу, коли отримаєте свій справжній перший офер.

Список рекомендованої літератури

1. Данилюк Н. М. Віртуальна організаційна структура іт-компанії: маркетинговий та управлінський аспекти. Наукові записки Національного університету «Острозька академія». Серія «Економіка» : науковий журнал. Острог : Вид-во НаУОА, березень 2022. № 24(52). С. 20–25. Режим доступу: https://eprints.oa.edu.ua/id/eprint/8704/1/5.pdf
2. Кібербезпека: актуальні загрози та методи захисту. – [Електронний ресурс]. Режим доступу: https://lemon.school/blog/kiberbezpeka-aktualni-zagrozy-ta-metody-zahystu
3. Відмінності та переваги agile, scrum та kanban в управлінні проєктами – [Електронний ресурс]. Режим доступу: https://online.novaposhta.education/blog/vidminnosti-ta-perevagi-agile-scrum-ta-kanban-v-upravlinni-proektami
4. Скрам – це ефективне управління проєктами. – [Електронний ресурс]. Режим доступу: https://brainrain.com.ua/uk/scrum-upravlinnya-proektom/
5. Канбан (Kanban) – [Електронний ресурс]. Режим доступу: https://www.maxzosim.com/kanban/
6. Що таке Jira і як з нею працювати. – [Електронний ресурс]. Режим доступу: https://iampm.club/ua/blog/shho-take-jira-i-yak-z-neyu-praczyuvati/
7. Як налаштувати Jira для управління беклогом: покрокова інструкція. – [Електронний ресурс]. Режим доступу: https://www.ba.in.ua/2023/02/24/yak-nalashtuvaty-jira-dlya-upravlinnya-beklogom-pokrokova-instrukcziya/
8. Git – [Електронний ресурс]. Режим доступу: https://git-scm.com/book/uk/v2
9. HTML Підручник – [Електронний ресурс]. Режим доступу: https://w3schoolsua.github.io/html/index.html#gsc.tab=0
10. CSS Підручник – [Електронний ресурс]. Режим доступу: https://w3schoolsua.github.io/css/index.html#gsc.tab=0
11. Unity Documentation – [Електронний ресурс]. Режим доступу: https://docs.unity3d.com/Manual/index.html
12. Основи Unity (Unity Essentials) – [Електронний ресурс]. Режим доступу: https://www.youtube.com/watch?v=uABmbFD6Of8&list=PL1BiB2ltf2GsPsa28Y3ckDf4Z0xEIt25s