Міністерство освіти і науки, молоді та спорту України
Криворізький технічний університет
Факультет інформаційних технологій
Кафедра комп’ютерних систем та мереж

План роботи гуртка «Байт»

керівник: к. пед.н., доц. кафедри КСМ О. Маркова

Кривий Ріг
2025

Зміст

План роботи студентського гуртка "Байт"

Мета: Розширення знань у сфері IT, розвиток практичних навичок розробки та командної роботи.

№ засідання	Тема засідання	Формат та основні питання
1	**Kick-off meeting: Вектори розвитку в IT**	Знайомство, визначення інтересів учасників. Огляд трендів 2025-2026 років. Поділ на команди для майбутніх міні-проєктів.
2	**Алгоритми та структури даних: "Спортивне програмування"**	Розбір класичних алгоритмів сортування та пошуку. Проведення міні-змагання на швидкість розв’язання задач (на платформі LeetCode або Codeforces).
3	**Магія Clean Code: Як писати код, який не хочеться спалити**	Принципи SOLID, DRY, KISS. Практикум з рефакторингу "брудного" коду. Огляд інструментів статичного аналізу.
4	**Git & GitHub: Командна розробка без конфліктів**	Робота з гілками (branching strategies), Pull Requests, Code Review. Вирішення Merge Conflicts на практиці.
5	**Штучний інтелект та Prompt Engineering**	Як ефективно використовувати LLM (ChatGPT, Claude, Gemini) для розробки. Створення власного міні-бота через API.
6	**Кібербезпека: Основи цифрової гігієни та тестування на проникнення**	Поширені вразливості (OWASP Top 10). Етичний хакінг: як захистити свій веб-ресурс від атак.
7	**UI/UX Дизайн для розробників**	Основи роботи у Figma. Чому інтерфейс має бути інтуїтивним. Створення прототипу мобільного застосунку за 60 хвилин.
8	**Backend vs Frontend: Битва технологій**	Огляд сучасних стеків (React/Next.js vs Python/Node.js/Go). Побудова простої архітектури клієнт-серверної взаємодії.
9	**Soft Skills та підготовка до Interview**	Як скласти резюме, яке помітить рекрутер. Симуляція технічної та поведінкової співбесіди (Mock Interview).
10	**Demo Day: Презентація проєктів**	Фінальна зустріч, де команди презентують свої напрацювання. Підбиття підсумків, нагородження активних учасників та піца-паті.

План засідання №1

Тема: «Kick-off meeting та вектори розвитку в IT»

Мета: Знайомство учасників, визначення очікувань, огляд актуальних технологій та формування команд.

Час проведення: 90 хвилин.

1. Блок «Знайомство» (20 хв)

- **Презентація гуртка:** Коротка історія назви «Байт», місія гуртка (спільне навчання, pet-проєкти, підготовка до олімпіад/хакатонів).
- **Вправа «Elevator Pitch»:** Кожен студент за 30 секунд має розповісти:
  1. Хто він/вона.
  2. Яку мову програмування любить (або хоче вивчити).
  3. Який крутий продукт мріє створити.
- **Створення ком'юніті:** Додавання всіх у закритий чат (Telegram/Discord) та спільний GitHub-орг.

2. Блок «Тренди IT 2025-2026» (25 хв)

Інтерактивна лекція-дискусія про те, куди рухається ринок. Основні тези:

- **AI-Driven Development:** Як ШІ змінює роботу програміста (від написання коду до архітектури).
- **Cybersecurity & Privacy:** Чому безпека стає частиною розробки (DevSecOps).
- **Edge Computing та IoT:** Чому обробка даних «на місцях» витісняє чисті хмарні рішення.
- **Green IT:** Енергоефективність коду як новий стандарт якості.

3. Блок «Проєктування траєкторії» (25 хв)

- **Анкетування (Mentimeter або Google Forms):** Студенти в реальному часі голосують за технології, які їм цікаві найбільше (наприклад: Python, Rust, React, AI, CyberSec).
- **Презентація «Мапи засідань»:** Демонстрація плану з 10 зустрічей, який ми склали раніше, з можливістю внести корективи на основі анкетування.
- **Вибір командних проєктів:** Оголошення ідеї, що до кінця курсу кожна команда (3-4 особи) має презентувати свій міні-проєкт (Demo Day).

4. Організаційний блок (10 хв)

- **Графік:** Узгодження дня та часу (щоб не накладалося на пари).
- **Ролі:** Хто хоче бути контент-мейкером гуртка (вести соцмережі), хто – техлідом (допомагати іншим з кодом).

5. Q&A та Рефлексія (10 хв)

- Відповіді на запитання.
- **«One-word checkout»:** Учасники одним словом описують свої враження від першої зустрічі.

Методичні матеріали:

1. **Презентація (5-7 слайдів):** З візуалізацією трендів та структурою гуртка.
2. **QR-коди:** На вступ у чат гуртка та на анкету інтересів.

3. **Стікери/Мерч (опціонально):** Невеликі наклейки з логотипом «Байт» для ноутбуків — це дуже зближує.

Результат засідання:

У кожного учасника є чітке розуміння, чим займається гурток, він доданий до всіх ресурсів та знає, хто його потенційні партнери по команді.

Зміст анкети, реалізованої через **Google Forms**:

Анкета учасника гуртка "Байт"

І. Загальна інформація

1. **Прізвище та ім’я**
2. **Курс та спеціальність**
3. **Твій рівень в програмуванні:**
   - *Початківець (знаю синтаксис, але важко писати код).*
   - *Середній (робив власні невеликі проєкти/лабораторні самостійно).*
   - *Просунутий (маю досвід роботи або глибокі знання в конкретному стеку).*

ІІ. Технологічні вподобання

4. **Які мови програмування тебе цікавлять найбільше?** (Оберіть до 3-х)
   - *Python / JavaScript (TypeScript) / C++ / Java / Rust / Go / Swift / Інше.*
5. **Який напрямок ІТ тобі найближчий?**
   - *Web Development (Frontend/Backend).*
   - *Mobile Development (iOS/Android).*
   - *Artificial Intelligence & Data Science.*
   - *Cybersecurity (Кібербезпека).*
   - *Game Development.*
   - *DevOps & Cloud Technologies.*
6. **Яку операційну систему ти використовуєш як основну для навчання?**
   - *Windows / macOS / Linux.*

ІІІ. Формат роботи та очікування

7. **Чого ти очікуєш від гуртка найбільше?** (Пріоритезуй: 1 — найважливіше)
   - *Отримання практичних навичок (воркшопи).*
   - *Робота над командним проєктом для портфоліо.*
   - *Підготовка до технічних співбесід.*
   - *Участь у хакатонах та олімпіадах.*
   - *Просто спілкування з однодумцями (нетворкінг).*
8. **Чи готовий/готова ти приділяти гуртку 2-3 години на тиждень поза засіданнями для роботи над проєктом?**
   - *Так / Можливо / Ні, тільки під час зустрічей.*

9. **Який формат засідань тобі подобається більше?**
   - *Live-coding (дивимось і повторюємо).*
   - *Міні-лекція + самостійна практика.*
   - *Групові обговорення та брейнсторми.*

IV. Творчий блок

10. **Чи є у тебе ідея проєкту, яку б ти хотів/хотіла реалізувати в межах гуртка?** (Якщо так — коротко опиши).
11. **Яку тему ти б міг/могла сам(а) розказати іншим учасникам (якщо є така експертиза)?**

План засідання №2

Тема: Алгоритми та структури даних: "Спортивне програмування"

Мета: Ознайомити з платформою LeetCode/Codeforces, навчити оцінювати складність коду та розв'язати перші змагальні задачі.

Тривалість: 90 хвилин.

1. Теоретичний блок: «Код, що літає» (20 хв)

- **Складність алгоритмів (O-нотація):** На пальцях пояснюємо різницю між $O(1)$, $O(n)$, $O(n^2)$ та $O(\log n)$. Чому ваш код може «впасти» на великих даних (Time Limit Exceeded).
- **Структури даних, що рятують життя:** Короткий огляд: Масиви vs Зв'язні списки, Стеки/Черги та Хеш-таблиці (Dictionary в Python / Map в JS).
- **Чек-лист спортивного програміста:** Як читати умову задачі, щоб не пропустити крайні випадки (Edge cases).

2. Воркшоп: «Розбір польотів» (25 хв)

Спільне розв'язання класичної задачі (наприклад, **"Two Sum"** або **"Valid Parentheses"**):

- **Крок 1:** "Брутальна сила" (Brute force) — пишемо найочевидніше рішення.
- **Крок 2:** Аналіз — чому це повільно?
- **Крок 3:** Оптимізація — використовуємо хеш-таблицю або два вказівники (Two Pointers), щоб пришвидшити код у рази.

3. Практика: Міні-турнір «Байт-Контест» (35 хв)

Студенти реєструються на LeetCode або Codeforces.

- **Завдання:** Дається набір із 3-х задач (Easy, Medium, Medium+).
- **Формат:** Можна працювати поодинці або в парах.
- **Азарт:** Виводимо на екран таблицю лідерів (якщо використовуєте внутрішній контест) або просто слідкуємо, хто першим здасть усі задачі.

4. Ретроспектива та лайфхаки (10 хв)

- **Code Review:** Розбір найкращого (найкоротшого або найшвидшого) розв'язку однієї з задач турніру.
- **Ресурсна база:** Поради, де тренуватися щодня (Advent of Code, Codewars).

- **Анонс:** Як алгоритми допоможуть нам у наступних темах (наприклад, у криптографії).

Обладнання:

1. Ноутбук із встановленим середовищем розробки (VS Code, PyCharm тощо) або доступ до онлайн-компіляторів.
2. Обліковий запис на LeetCode (бажано створити до початку).

План засідання №3

Тема: Магія Clean Code: Як писати код, який не хочеться спалити

Мета: Навчити студентів бачити різницю між "працюючим" та "якісним" кодом, опанувати базові принципи рефакторингу. **Тривалість:** 90 хвилин.

1. Інтерактив: Естетика коду (15 хв)

- **Гра "Вгадай, що я роблю":** На екрані показується шматок коду з жахливими назвами змінних (наприклад, data1, temp_list, a, b12). Студенти мають вгадати логіку функції.
- **Висновок:** Код читається набагато частіше, ніж пишеться. Код має бути "прозорим".

2. Теорія: Золоті правила (25 хв)

Короткий розгляд ключових концепцій (без занурення в академічну нудьгу):

- **Назви, що говорять:** Замість d = 86400 пишемо SECONDS_IN_A_DAY.
- **Правило бойскаута:** Залишай код після себе трохи кращим, ніж він був до тебе.
- **Принципи KISS та DRY:** * *KISS (Keep It Simple, Stupid):* Не ускладнюй там, де можна зробити просто.
    - *DRY (Don’t Repeat Yourself):* Бачиш дублювання — винось у функцію.
- **Функції-егоїсти:** Функція має робити лише одну річ (Single Responsibility). Якщо в назві функції є слово "And" — її треба розділити.

3. Практикум: "Code Clinic" (35 хв)

Це основна частина. Студенти отримують посилання на репозиторій (або файл) із "брудним" кодом.

- **Завдання:** Провести рефакторинг.
- **Етапи:**

1. Виправити іменування.
2. Розбити велику функцію (на 50+ рядків) на декілька маленьких.
3. Замінити складні конструкції if-else на більш зрозумілу логіку або match/switch.
4. Видалити мертвий код та зайві коментарі (гарний код не потребує коментарів, він сам себе пояснює).

4. Дискусія: Коли "Clean" стає "Too Clean"? (10 хв)

- Обговорення межі: коли прагнення до ідеального коду починає шкодити швидкості розробки (Overengineering).

- Чи завжди потрібно слідувати всім правилам SOLID?

5. Підбиття підсумків та Demo (5 хв)

- Порівняння коду "До" та "Після" на екрані.
- **Анонс:** Наступного разу ми дізнаємося, як зберігати цей чистий код у Git.

Методичні матеріали:

1. Книга **Роберта Мартіна "Чистий код"** (must-read для кожного учасника).
2. Сайт **Refactoring.Guru** (чудові візуалізації принципів та патернів).
3. Плагіни для IDE: **SonarLint** або **Prettier** (показати, як вони автоматизують чистоту).

«Брудний» код (Code to Refactor)

Ця програма розраховує знижки для покупців в інтернет-магазині та виводить фінальну ціну.

Python

```
def p(d, t):
    # d - це список товарів, t - тип клієнта
    res = 0
    for i in d:
        res = res + i['p']

    if t == 1: # VIP клієнт
        s = res * 0.2
        res = res - s
    else:
        if res > 1000: # Оптова знижка
            res = res - 50

    print("Фінальна сума: " + str(res))

    # Записуємо в лог
    f = open("log.txt", "a")
     f.write("User " + str(t) + " bought stuff for " +
str(res) + "\n")
    f.close()

items = [{'n': 'Phone', 'p': 1200}, {'n': 'Case', 'p':
50}]
p(items, 1)
```

Що тут «брудного» (Аналіз для студентів):

1. **Жахливі назви:** p, d, t, res, s, i — нічого не зрозуміло без коментарів.
2. **Magic Numbers:** Що таке 0.2? Що таке 1? (Це мають бути константи або Enum).
3. **Порушення Single Responsibility:** Функція і рахує, і виводить на екран, і пише у файл.

4. **Відсутність обробки помилок:** Робота з файлом без with або try-except.
5. **Дублювання логіки:** Змішування бізнес-логіки зі звітами.

Як має виглядати цей код після «Магії Clean Code»:

Покажіть **цей варіант після того, як студенти запропонують свої рішення.**

Python

```
from enum import Enum

class CustomerType(Enum):
    REGULAR = 0
    VIP = 1

VIP_DISCOUNT = 0.2
BULK_DISCOUNT_THRESHOLD = 1000
BULK_DISCOUNT_AMOUNT = 50

def calculate_total_price(cart_items):
    return sum(item['price'] for item in cart_items)

def apply_discounts(total, customer_type):
    if customer_type == CustomerType.VIP:
        return total * (1 - VIP_DISCOUNT)

    if total > BULK_DISCOUNT_THRESHOLD:
        return total - BULK_DISCOUNT_AMOUNT

    return total

def log_transaction(customer_type, final_amount):
    try:
                with open("transactions.log", "a") as
log_file:
                                log_file.write(f"Customer:
{customer_type.name}, Amount: {final_amount}\n")
    except IOError as e:
        print(f"Logging failed: {e}")

# Головна логіка
cart = [
    {'name': 'Phone', 'price': 1200},
    {'name': 'Case', 'price': 50}
]

raw_total = calculate_total_price(cart)
```

```
    final_price = apply_discounts(raw_total,
CustomerType.VIP)

    print(f"Фінальна сума до сплати: {final_price} грн")
    log_transaction(CustomerType.VIP, final_price)
```

Завдання для засідання:

1. **Роздати "брудний" код** (скинути в чат або GitHub Gist).
2. **Дати 20 хвилин** на виправлення назв та розділення функцій.
3. **Обговорити:** Чому варіант з Enum та константами кращий, ніж if t == 1.

План засідання №4

Тема: Git & GitHub: Командна розробка без конфліктів

Мета: Опанувати основні команди Git, зрозуміти робочий процес (Workflow) у команді та навчитися вирішувати конфлікти злиття.

Тривалість: 90 хвилин.

1. Теорія: Git – це машина часу для коду (15 хв)

- **Навіщо це треба:** Історія версій, можливість відкотитися назад та паралельна розробка фіч.
- **Три стани файлу:** Working Directory, Staging Area, Local Repository.
- **Централізація:** Різниця між Git (інструмент) та GitHub (соціальна мережа для коду).

2. Воркшоп: Базовий цикл (20 хв)

Студенти виконують у терміналі (або через GUI) базовий ланцюжок команд:

1. git init – створюємо репозиторій.
2. git add . – готуємо файли (Stage).
3. git commit -m "feat: add login logic" – фіксуємо зміни.
   - *Важливо:* Вчимо писати змістовні повідомлення (Conventional Commits).
4. git remote add origin ... та git push – відправляємо в хмару.

3. Командна гра: "The Conflict Challenge" (40 хв)

Це ключова практична частина. Студенти діляться на пари (А і Б).

- **Крок 1:** Студент А створює репозиторій на GitHub і додає студента Б як співавтора (Collaborator).
- **Крок 2 (Branching):** Кожен створює свою гілку: git checkout -b feature-alpha.
- **Крок 3 (Conflict):** Обидва студенти змінюють **один і той самий рядок** у файлі README.md або в коді, роблять комміт та пуш.
- **Крок 4 (Pull Request):** Студент А мержить свій код. Студент Б намагається мержити свій і отримує **Merge Conflict**.
- **Крок 5 (Resolution):** Разом вчимося відкривати редактор, обирати потрібний варіант (Current vs Incoming change) та завершувати злиття.

4. Огляд GitHub Flow (10 хв)

- Що таке **Pull Request (PR)** і чому не можна пушити прямо в main.
- Як проводити **Code Review**: залишаємо коментарі до коду колеги, просимо виправити помилки.
- Використання **GitHub Issues** для планування завдань.

5. Підсумки та ДЗ (5 хв)

- **Домашнє завдання:** Створити репозиторій для свого майбутнього командного проєкту, додати туди учасників своєї команди та файл .gitignore.

Поради для модератора:

- **Візуалізація:** Використовуємо команду git log --graph --oneline для демонстрації того, як гілки розходяться і сходяться.
- **Інструменти:** Показуємо, як вбудований інструмент VS Code допомагає візуально вирішувати конфлікти – це знімає страх перед ними.
- **Курйози:** Розказуємо історію про те, як хтось випадково видалив main або завантажив у публічний репозиторій паролі (і нагадайте про .gitignore).

План засідання №5

Тема: «Штучний інтелект та Prompt Engineering»

Тривалість: 90 хвилин.

Обладнання: Ноутбуки, доступ до інтернету, облікові записи (ChatGPT, Gemini або Claude).

1. Теорія: Як «думає» нейромережа (15 хв)

- **Архітектура коротко:** Що таке токени та чому модель «передбачає» наступне слово, а не знає відповідь.
- **Обмеження:** Проблема галюцинацій та актуальність знань (cutoff date).
- **Чому контекст важливий:** Поняття «context window».

2. Воркшоп: Мистецтво промптингу (30 хв)

Розгляд ключових технік створення запитів. Кожен студент має спробувати ці підходи в реальному часі:

- **Формула RTF (Role, Task, Format):**
  - *Role:* «Дій як Senior Python Developer...»
  - *Task:* «Напиши скрипт для парсингу сайту...»
  - *Format:* «Виведи результат у вигляді таблиці Markdown...»
- **Техніка Few-Shot Prompting:** Надання моделі кількох прикладів «вхідні дані -> вихідні дані» перед основним питанням.
- **Chain-of-Thought (Ланцюжок думок):** Використання фрази «Давай думати крок за кроком», що змушує ШІ вибудовувати логічну послідовність (особливо важливо для коду та математики).

3. Практичне завдання: «Pair Programming з ШІ» (35 хв)

Студенти діляться на пари. Завдання: написати невеликий застосунок (наприклад, «Калькулятор валют» або «Телеграм-бот для розкладу») за допомогою ШІ.

- **Етап 1:** Генерація структури проєкту.

- **Етап 2:** Написання окремих функцій.
- **Етап 3:** Виправлення помилок (студенти навмисно вставляють помилку в код і просять ШІ її знайти та пояснити).

4. Дискусія та етика (10 хв)

- Де межа між «допомогою» та «плагіатом»?
- Як перевіряти код, згенерований ШІ, на безпеку та вразливості.

Корисні ресурси для засідання:

1. **Learn Prompting** (безоплатний курс).
2. **GitHub Copilot / Cursor –** огляд інструментів, що інтегруються безпосередньо в редактор коду.
3. **Prompt Engineering Guide** (dair-ai).

Домашнє завдання:

Створити «Perfect Prompt» для вирішення конкретної навчальної задачі (наприклад, пояснення складної теми з дискретної математики п'ятирічній дитині) та поділитися результатом у чаті гуртка.

Це засідання присвячене безпеці. Робимо акцент із «хакерства як злочину» на «етичний хакінг» та розуміння того, як захистити власні розробки.

План засідання №6

Тема: Кібербезпека: Основи цифрової гігієни та тестування на проникнення

Мета: Ознайомити з основними векторами атак на вебдодатки, принципами безпечної розробки та методами захисту персональних даних.

Тривалість: 90 хвилин.

1. Теорія: Світ очима хакера (15 хв)

- **Хто такий Ethical Hacker:** Чим відрізняються "White Hat" від "Black Hat".
- **OWASP Top 10:** Короткий огляд десяти найкритичніших ризиків безпеки вебдодатків (SQL-ін'єкції, XSS, зламаний контроль доступу тощо).
- **Соціальна інженерія:** Чому найслабша ланка в безпеці – це людина, а не код.

2. Демонстрація: «Анатомія атаки» (25 хв)

- **SQL Injection:** Показ того, як через звичайне поле логіна можна отримати доступ до всієї бази даних (на прикладі безпечного локального стенду або тренажера).
- **XSS (Cross-Site Scripting):** Як через коментар на сайті можна вкрасти Cookies користувача.
- **Брутфорс:** Чому пароль 123456 або password зламується за мілісекунди.

3. Практикум: «Safe Code & Pentest» (40 хв)

Студенти працюють із тренувальними платформами (наприклад, **TryHackMe**, **Hack The Box** або локальна **DVWA –** Damn Vulnerable Web Application).

- **Завдання 1:** Знайти вразливість у наданому шматочку коду (наприклад, де дані від користувача потрапляють прямо в SQL-запит).

- **Завдання 2:** Виправити цей код (використання Prepared Statements або валідації).
- **Завдання 3:** Налаштування двофакторної автентифікації (2FA) для свого GitHub-акаунта, якщо це ще не зроблено.

4. Цифрова гігієна для розробника (10 хв)

- **Менеджери паролів:** Чому не можна використовувати один пароль всюди.
- **Безпека в Git:** Як не «запушити» випадково API-ключі чи паролі в публічний репозиторій (використання .env та інструментів на кшталт git-secrets).
- **VPN та публічний Wi-Fi:** Ризики роботи в кафе.

Ресурси та інструменти:

1. **OWASP Juice Shop:** Найсучасніший і найскладніший проєкт для тренувань із безпеки (Open Source).
2. **Google Gruyere:** Старий, але простий тренажер від Google для вивчення веб-вразливостей.
3. **Have I Been Pwned:** Сервіс для перевірки, чи були ваші дані злиті в мережу.

Порада для модератора:

Будьте обережні: наголошуємо, що будь-яке тестування чужих систем без дозволу є незаконним.

План засідання №7

Тема: UI/UX Дизайн для розробників

Мета: Навчитися проектувати зручні інтерфейси, зрозуміти різницю між UX та UI, та опанувати Figma на базовому рівні.

Тривалість: 90 хвилин.

1. Теорія: UX — це не "красиві кнопки" (15 хв)

- **UX (User Experience):** Логіка роботи, шлях користувача від точки А до точки Б. Чому "зручно" важливіше, ніж "красиво".
- **UI (User Interface):** Візуальний стиль (кольори, шрифти, відступи).
- **Закони дизайну для технарів:**
  - **Закон Фіттса:** Чим більша ціль і чим ближче вона, тим швидше її досягти (чому кнопка "Купити" велика).
  - **Закон Хіка:** Час на прийняття рішення зростає з кількістю варіантів (чому меню не повинно мати 50 пунктів).
  - **Принцип контрасту:** Головна дія має виділятися.

2. Огляд інструменту: Figma за 15 хвилин (15 хв)

- **Робоча область:** Frames (фрейми), Layers (шари).
- **Auto Layout:** Секретна зброя для створення адаптивних елементів (аналог Flexbox у вебі).

- **Компоненти:** Як створити одну кнопку і перевикористовувати її всюди (аналог класів/функцій у коді).

3. Практикум: "Спринт-дизайн мобільного застосунку" (45 хв)

Студенти працюють у Figma (можна в парах).

- **Завдання:** Створити 3 екрани для простого застосунку (наприклад, "Трекер звичок" або "Замовлення кави").
- **Етапи:**

1. **Wireframing (низька деталізація):** Малюємо "скелет" застосунку сірими прямокутниками (фокус на логіці).
2. **Visual Design:** Додаємо кольори, типографіку та іконки.
3. **Prototyping:** З'єднуємо екрани стрілочками, щоб зробити клікабельний макет.

4. Demo та Feedback: "Критика дизайнера" (15 хв)

- Випадкові пари обмінюються посиланнями на свої Figma-проєкти.
- **Завдання:** Спробувати "пройти" шлях користувача в прототипі партнера і знайти одне місце, де можна заплутатися.
- **Обговорення:** Чому розробнику важливо вміти читати Figma-макети (Inspect mode, копіювання CSS-параметрів).

Що знадобиться:

1. Попередньо створений акаунт у **Figma** (працює в браузері).
2. Встановлені шрифти або використання стандартних (Inter, Roboto).
3. Набір іконок (наприклад, плагін **Iconify** у Figma).

Порада для модератора:

Акцентуємо увагу на тому, що **дизайн – це теж ітеративний процес**. Не обов'язково бути художником, щоб зробити зручний інтерфейс. Використовуємо готові UI-кити (набори елементів), щоб студенти не витрачали час на малювання кожної лінії з нуля.

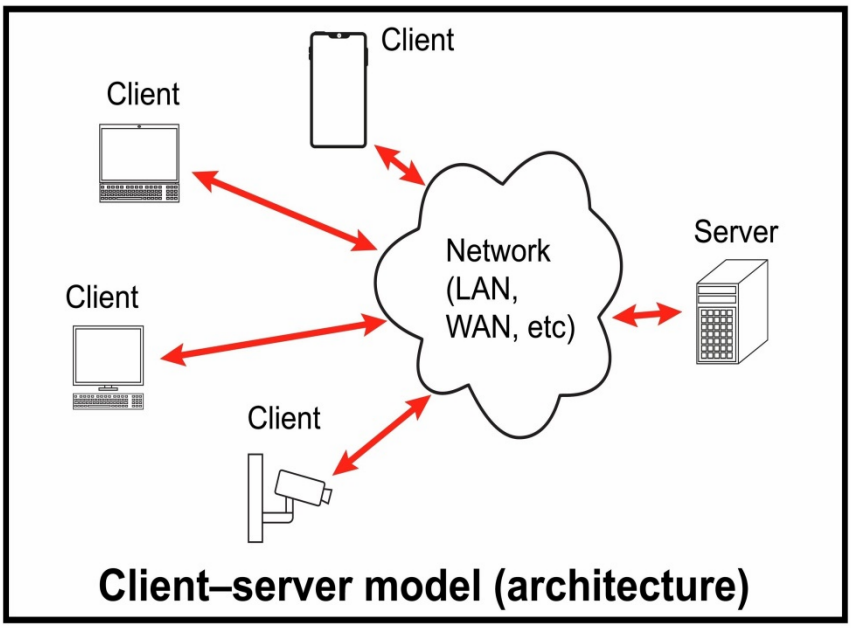
План засідання №8

Тема: Backend vs Frontend: Битва технологій

Мета: Розібрати ролі фронтенд- та бекенд-розробників, зрозуміти архітектуру клієнт-серверної взаємодії та спробувати «подружити» їх через API. **Тривалість:** 90 хвилин.

1. Теорія: Дві сторони однієї медалі (20 хв)

- **Frontend (Клієнт):** Те, що бачить користувач. Стек: HTML, CSS, JS + фреймворки (React, Vue, Angular).
- **Backend (Сервер):** Те, що працює «під капотом». Логіка, автентифікація, робота з БД. Стек: Python (Django/FastAPI), Node.js, Go, PHP, Java.
- **Міст ок — API (REST, GraphQL):** Як вони передають дані. Аналогія з рестораном: Клієнт (Frontend) замовляє страву через офіціанта (API), а кухар (Backend) її готує.



Shutterstock

2. Архітектурний розбір: JSON та HTTP (15 хв)

- **HTTP методи:** GET (дай дані), POST (створи), PUT (онови), DELETE (видали).
- **Формат JSON:** Чому він став стандартом обміну даними.
- **Статуси відповідей:** 200 (OK), 404 (Not Found), 500 (Internal Server Error).

3. Практикум: «З'єднуємо деталі» (45 хв)

Студенти працюють із готовим міні-проєктом (наприклад, "Список справ").

- **Завдання 1 (Backend):** Запустити простий сервер (наприклад, на FastAPI або Express), який повертає список завдань у форматі JSON.
- **Завдання 2 (Frontend):** Використати команду fetch у JavaScript, щоб отримати ці дані з сервера та відобразити їх у браузері.
- **Завдання 3 (Interaction):** Реалізувати додавання нового завдання через форму, яке б зберігалося на сервері (POST запит).

4. Дискусія: Fullstack чи спеціалізація? (10 хв)

- Хто такий **Fullstack розробник** і чи реально знати все.
- Як обрати свій шлях на початку кар'єри.
- Поняття **BaaS (Backend as a Service)**, наприклад, Firebase або Supabase — коли не хочеться писати бекенд з нуля.

Що знадобиться учасникам:

1. Встановлений **Postman** або розширення **Thunder Client** у VS Code для тестування API-запитів.
2. Базові знання JavaScript (ES6+).
3. Для бекенду можна підготувати короткий шаблон на Python або Node.js, щоб студенти не витрачали час на налаштування середовища.

Порада для модератора:

Найцікавіший момент засідання – коли студент вперше бачить, як дані, введені у форму на сторінці, реально з'являються в базі даних або логах сервера. Зробіть на цьому акцент!

Це засідання – перехід від технічних навичок (Hard Skills) до кар'єрного старту. Тут ми вчимося «продавати» свій код та свою особистість роботодавцю.

План засідання №9

Тема: Soft Skills та підготовка до Interview

Мета: Навчитися презентувати свій досвід, підготувати якісне резюме (CV) та відпрацювати навички проходження співбесіди.

Тривалість: 90 хвилин.

1. Теорія: Чому "геніїв-одинаків" більше не наймають (15 хв)

- **Soft Skills, що рятують кар'єру:** Комунікація, вміння приймати критику (Code Review), тайм-менеджмент та емпатія в команді.
- **Як працює рекрутинг:** Хто такі HR, рекрутери, техліди. Що таке ATS (системи автоматичного сканування резюме) і як їх пройти.

2. Майстер-клас: Ідеальне CV та LinkedIn (20 хв)

- **Структура резюме:** Як описати проєкти, над якими працювали в гуртку «Байт».
- **GitHub як візитівка:** Чому порожній профіль — це погано, і що має бути в README.md вашого проєкту.
- **LinkedIn:** Поради щодо оформлення профілю, щоб рекрутери самі писали вам.

3. Практикум: "Mock Interview" (Симуляція співбесіди) (40 хв)

Учасники діляться на пари: «Інтерв'юер» та «Кандидат». Потім міняються ролями.

- **Етап 1: Behavioral Interview (15 хв):** Відповіді на запитання за методикою **STAR** (Situation, Task, Action, Result).
  - *Приклад:* "Розкажи про ситуацію, коли ти не встигав зробити завдання вчасно".
- **Етап 2: Technical Screen (15 хв):** Бліц-опитування з базових знань (що таке об'єкт, як працює інтернет, різниця між SQL та NoSQL).
- **Етап 3: Фідбек (10 хв):** Обговорення в парах: що сподобалося, а що викликало труднощі.

4. Блок: "Запитай у профі" (10 хв)

- Огляд типових помилок джуніорів на першій співбесіді.
- Як відповідати на запитання: "Які ваші очікування щодо зарплати?".
- Чому відмова після співбесіди — це теж корисний досвід.

5. Підготовка до фіналу (5 хв)

- Останні настанови перед **Demo Day** (Засідання №10).
- Поради щодо публічного виступу: як презентувати свій проєкт за 5 хвилин.

Що підготувати заздалегідь:

1. **Шаблон резюме** (наприклад, посилання на Canva або Google Docs).
2. **Список з 20 типових питань** на співбесіду для роздруківки учасникам.
3. Запрошення (якщо можливо) **реального рекрутера** або розробника з досвідом для короткого спічу онлайн.

Порада для модератора:

Створюємо атмосферу безпеки. Багато студентів бояться співбесід. Ваша мета – показати, що це звичайна розмова двох спеціалістів про те, як вони можуть бути корисні один одному.

Це фінальна точка подорожі гуртка **«Байт»**. Це не просто звіт, а свято, де студенти перетворюються з учнів на творців, які презентують власні продукти.

План засідання №10

Тема: Demo Day: Презентація проєктів

Мета: Публічний захист командних проєктів, отримання зворотного зв'язку та неформальне святкування завершення сезону.

Тривалість: 120 хвилин (залежить від кількості команд).

1. Відкриття та Welcome Drink (10 хв)

- Вітальне слово керівника гуртка.
- Представлення «журі» або гостей (викладачі, представники IT-компаній або старшокурсники).
- Коротке нагадування про шлях, який пройшов гурток за ці 10 занять.

2. Pitch Session: Презентації (60 хв)

Кожна команда має **5-7 хвилин** на виступ + **3 хвилини** на запитання. **Структура презентації:**

1. **Проблема:** Яку задачу вирішує ваш проєкт?
2. **Демонстрація (Live Demo):** Показ працюючого коду/інтерфейсу (не просто скріншоти!).
3. **Технологічний стек:** Що використовували (Git, Clean Code, API, AI-інструменти).
4. **Команда:** Хто за що відповідав.

3. Голосування та фідбек (15 хв)

- Поки журі радиться, учасники можуть проголосувати в номінації **"People's Choice Award"** (Приз глядацьких симпатій) через онлайн-форму.
- Гості дають короткий конструктивний фідбек: що сподобалося в технічних рішеннях команд.

4. Нагородження (15 хв)

- Вручення сертифікатів про участь у роботі гуртка.
- Оголошення переможців у номінаціях:
  - *«Натистіший код»* (найкращий рефакторинг).
  - *«Накращий UX/UI»*.
  - *«Найтехнологічніше рішення»*.
- Невеликі подарунки (мерч, книги з програмування або промокоди на курси).

5. Pizza Party та Networking (20 хв)

- Неформальне спілкування.
- Колективне фото.
- **Збір фідбеку про гурток:** Що студентам сподобалося найбільше і які теми вони хочуть розглянути в наступному семестрі.

Що підготувати заздалегідь:

1. **Технічне забезпечення:** Проєктор, клікер, стабільний інтернет.
2. **Сертифікати та призи:** Надруковані грамоти (можна додати QR-код на спільний репозиторій гуртка).
3. **Кейтеринг:** Піца, напої — це створює атмосферу справжнього IT-мітапу.

Порада для модератора:

Підтримуємо студентів, у яких під час демо «щось пішло не так» (класичний ефект демо). Наголошуємо, що навіть у великих компаній презентації інколи провалюються, і головне – як розробник вміє це пояснити.