Молодченко Дмитро

Модульний контроль

1. 1956 — Дартмутська конференція: народження символічного ШІ (правила, логіка), що дало формальні методи пошуку рішень; 1960-70-ті — перцептрони й експертні системи, перший підйом і «зима ШІ» після критики обмежень; 1986 — відновлення зворотного поширення помилки (backprop), поява багатошарових нейромереж; 1997 — Deep Blue перемагає Каспарова, демонстрація сили вузьких алгоритмів; 2006-2012 — «ренесанс глибокого навчання» (велика кількість даних і GPU), 2012 — AlexNet виграє ImageNet і запускає хвилю DL у комп'ютерному баченні; 2017 — Transformer спрощує навчання на великих текстах; 2018-2020 — великі мовні моделі (BERT, GPT-2/3) змінюють NLP; 2022-2024 — інструкційно налаштовані чат-моделі та мультимодальність роблять ШІ масовим інструментом; прямий ефект кожного етапу — перехід від ручових правил до статистики, далі до масштабованих універсальних моделей, що автоматизують усе більше знаннєвих задач.

2. GitHub Copilot став популярним, бо знімає зменшує кількість ручного написання тексту в щоденному кодингу: він пропонує доречні підказки у реальному часі в IDE, генерує шаблонний код, тести та докстрінги, зменшуючи контекстні перемикання й когнітивне навантаження. Принципи зручності: робота «там, де користувач уже є» (VS, VS Code, Rider), «autocomplete на стероїдах» із урахуванням контексту файлу/проєкту, швидкий зворотний зв'язок і можливість приймати/редагувати пропозиції без відриву від потоку.

3. Попросив ШІ "зробити в .NET MAUI повноекранну сторінку без safe area і сховати статус/навігаційні панелі". Модель згенерувала фрагмент із `WindowCompat.SetDecorFitsSystemWindows(window, false)` та викликами `WindowInsetsControllerCompat`, але без умовної компіляції під Android, без обробки insets та без компенсаційного відступу для контенту. У результаті на Android частина елементів опинилась під статус-баром і стала неклікабельною, жести "назад/домівка" працювали нестабільно; на iOS код взагалі не компілювався через Android-залежності; а для Windows модель порадила "додати ресурси стилів", що призвело до помилки складання на кшталт `PRI277: Conflicting values ... Duplicate Entry`. Причини: змішування платформ в одному фрагменті, застарілий/неперевірений код (орієнтований на іншу версію MAUI),

відсутність логіки для `WindowInsets` і safe-area. Як уникнути: чітко просити платформо-специфічний код із `#if ANDROID / #if WINDOWS / #if IOS` і зазначенням цільової версії .NET MAUI; інтегрувати через `ConfigureLifecycleEvents` у `MauiProgram` замість сирого втручання в `MainActivity`; одразу додавати обробник insets (або `UseWindowSafeArea(false)` + динамічні відступи з фактичних insets), не копіювати вручну системні стилі під Windows; збирати й запускати після кожного кроку на цільовому пристрої, звірятися з офіційними прикладами та не приймати від ШІ зміни “пакетом” без рев’ю.

4. Демонстрація: автоматизація підготовки лабораторних звітів. Задача — з сирих даних побудувати графіки, порахувати метрики (напр., MAD/MSE) і сформувати Word-звіт за шаблоном без ручного форматування. Інструменти — Python (pandas для обчислень, matplotlib для графіків, python-docx для підстановки в шаблон), плюс чат-модель для стислого резюме висновків. Результат — стабільний 60–70% виграш у часі на кожен звіт, менше механічних помилок і відтворюваність (змінив дані — отримав оновлений звіт одним запуском).

5. Ризики «цифрового рабства»: атрофія навичок і критичного мислення через надмірну делегацію; упереджені або непрозорі рішення, що непомітно спрямовують вибір.

Заходи: регулярна перевірка навичок без ШІ; обмеження часу/сфер, де дозволена автоматизація.