Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра
з теми: «Модель автоматизованої системи вибору та аналізу вибіркових освітніх компонент»

Виконав: здобувач вищої освіти
групи KN1-B21
спеціальності 122 Комп’ютерні науки
Білоус Максим Васильович

Керівник:
Мястковська Марина Олександрівна,
кандидат педагогічних наук, старший
викладач кафедри комп’ютерних наук

Рецензент: Чорна Оксана Григорівна,
кандидат педагогічних наук, старший
викладач кафедри фізики, керівник
навчального відділу університету

Кам’янець-Подільський – 2025 р.

АНОТАЦІЯ

Білоус М.В. Модель автоматизованої системи вибору та аналізу вибіркових освітніх компонент. − Кваліфікаційна робота бакалавра, 2025.

У кваліфікаційній роботі розроблено та реалізовано модель автоматизованої інформаційної системи, призначеної для підтримки процесу вибору студентами вибіркових освітніх компонент. Система забезпечує зручний інтерфейс для студентів з можливістю авторизації через корпоративну пошту, вибору дисциплін відповідно до спеціальності, форми навчання, ступеня освіти та факультету.

Функціонал адміністратора реалізовано на основі скриптів Google Apps Script, що автоматизують аналіз результатів вибору. Дані студентів і дисциплін зберігаються в Google Таблицях, що забезпечує простоту розгортання та використання без додаткового серверного забезпечення. Для представлення та фільтрації даних реалізовано функції сортування студентів за факультетами та дисциплінами.

Аналітична частина дипломної роботи була використана навчальним відділом Кам’янець-Подільського національного університету імені Івана Огієнка для підготовки звітів щодо вибору студентами вибіркових дисциплін, що підтверджує її прикладну цінність та актуальність.

Ключові слова: вибіркові освітні компоненти, автоматизована система, Google Sheets, Google Apps Script, Angular, аналіз даних, сортування студентів.

ANNOTATION

Bilous M. V. Model of an Automated System for the Selection and Analysis of Elective Educational Components. – Bachelor's Qualification Thesis, 2025.

This bachelor’s thesis presents the development and implementation of a model of an automated information system designed to support the process of selecting elective educational components by students. The system provides a convenient user interface with the ability to log in via corporate email and select disciplines based on specialty, form of study, educational degree, and faculty.

The administrator’s functionality is implemented using Google Apps Script, which automates the analysis of selection results. Student and discipline data are stored in Google Sheets, ensuring ease of deployment and use without additional server infrastructure. The system includes functions for filtering and sorting students by faculties, groups, and disciplines for analytical purposes.

The analytical part of the thesis was used by the Academic Department of Kamianets-Podilskyi Ivan Ohiienko National University to prepare reports on students’ elective course selections, confirming its practical value and relevance.

Keywords: elective educational components, automated system, Google Sheets, Google Apps Script, Angular, data analysis, student sorting.

ЗМІСТ

ВСТУП

У сучасних умовах діджиталізації освітнього процесу дедалі більшої актуальності набуває проблема автоматизації рутинних та організаційно складних процедур в університетах. Однією з таких процедур є вибір здобувачами вищої освіти вибіркових освітніх компонентів. Вибірковість є невід'ємною частиною академічної автономії та академічної свободи студентів, а також однією з ключових вимог до формування індивідуальних освітніх траєкторій згідно з Законом України «Про вищу освіту». Водночас на практиці реалізація цього принципу часто супроводжується низкою організаційних труднощів, особливо коли відсутні належні інструменти для збору, обробки та аналізу даних.

Актуальність теми зумовлена потребою у створенні ефективної, гнучкої та доступної системи, яка дозволить студентам обирати дисципліни у зручному форматі, а адміністрації – оперативно отримувати структуровану інформацію про ці вибори. Існуючі рішення, як правило, є громіздкими або платними, що створює бар'єри для їх широкого впровадження. Тому розробка інформаційної системи, заснованої на відкритих вебтехнологіях і доступних сервісах, є актуальною як для окремих факультетів, так і для всіх закладів вищої освіти загалом.

Об'єкт дослідження – процес вибору студентами вибіркових освітніх компонентів в закладах вищої освіти.

Предмет дослідження – модель автоматизованої системи вибору та аналізу вибіркових освітніх компонент.

Метою дипломної роботи є розробка та впровадження моделі автоматизованої системи вибору вибіркових дисциплін для закладів вищої освіти.

Для досягнення поставленої мети необхідно розв'язати такі **завдання**:

1. Проаналізувати нормативно-правову базу та існуючі підходи до вибору вибіркових компонентів.

2. Дослідити проблеми, що виникають у закладах вищої освіти під час цього процесу.
3. Розробити вебзастосунок для студентів з функціоналом вибору дисциплін.
4. Реалізувати збереження та оновлення даних за допомогою інтеграції з Google Sheets.
5. Створити інструменти сортування та аналізу даних за допомогою Google Apps Script.
6. Впровадити інтерфейсні рішення, що забезпечують зручність використання як для студентів, так і для адміністрації.

У процесі виконання дипломної роботи застосовувались такі **методи дослідження**: аналіз нормативних документів і практик вищої освіти; методи структурного та об’єктно-орієнтованого проектування програмного забезпечення; методи автоматизації обробки даних у середовищі Google Workspace; емпіричне тестування застосунку з подальшим удосконаленням його функціональності.

Практичне значення одержаних результатів полягає у створенні повнофункціонального застосунку, який може бути впроваджений в освітній процес для спрощення процедури вибору дисциплін. Аналітична частина дипломної роботи була використана навчальним відділом Кам’янець-Подільського національного університету імені Івана Огієнка для підготовки звітів щодо вибору студентами вибіркових дисциплін, що підтверджує її прикладну цінність та актуальність. Завдяки використанню відкритих технологій система є доступною для адаптації в інших навчальних закладах без потреби у складній серверній інфраструктурі.

Апробація результатів дослідження: виступ на науковій конференції студентів і магістрантів за підсумками НДР у 2024-2025 навчальному році в Кам’янець-Подільському національному університеті імені Івана Огієнка (9-10 квітня 2025 року). Результати роботи опубліковані в 1 тезах [7].

Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків. У першому розділі розглянуто

нормативну базу, подібні системи та загальні підходи до автоматизації вибору дисциплін. У другому розділі детально описано розроблений застосунок, його інтерфейс і функціональність зі сторони студента. Третій розділ присвячено інструментам обробки та аналізу вибору дисциплін за допомогою Google Apps Script у копії таблиці. Список використаних джерел складається із 14 джерел.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Нормативно-правове регулювання вибору вибіркових компонентів

В Україні право студента на індивідуальний вибір частини навчальних дисциплін закріплене на законодавчому рівні. Закон України «Про вищу освіту» (ст. 62) гарантує здобувачам вищої освіти можливість вільно обирати освітні компоненти (дисципліни) в обсязі, що становить не менш як 25 відсотків від загальної кількості кредитів ЄКТС за освітньою програмою. Для спеціальностей, що передбачають доступ до професій зі запровадженим додатковим регулюванням, кількість кредитів становить не менше ніж 10 відсотків. Ці вибіркові компоненти включаються до індивідуального навчального плану студента і стають обов'язковими для нього після затвердження вибору. Закон також передбачає, що здобувачі певного рівня вищої освіти можуть обирати дисципліни не лише зі списку, запропонованого їхньою освітньою програмою, але й з інших програм та рівнів підготовки відповідно до умов, визначених закладом (відповідно до положення про організацію освітнього процесу даного закладу вищої освіти). Таким чином, законодавство встановлює мінімальну частку вибіркового компоненту в освітніх програмах і надає студентам інструмент формування індивідуальної освітньої траєкторії [9].

Для реалізації цього права кожен заклад вищої освіти розробляє власні нормативні акти – насамперед положення про порядок вибору вибіркових освітніх компонент у вигляді окремого документа або розділу у загальному положенні про організацію освітнього процесу. Міністерство освіти і науки України (МОН) у своїх рекомендаціях наголошує на автономії закладів вищої освіти в організації цієї процедури: заклади самостійно визначають механізми, за якими здобувачі вищої освіти здійснюють вибір, закріплюючи їх у внутрішніх нормативних документах. Освітні програми та стандарти вищої освіти передбачають поділ навчального плану на обов'язкову та вибіркову

компоненти. Зокрема, стандарт освітньої програми має відповідати вимогам закону щодо частки вибіркових кредитів, а навчальні плани містять варіативну частину, з якої студент обирає дисципліни на свій розсуд. Вибіркові освітні компоненти визначаються у документах як дисципліни, що пропонуються закладом для більш повного задоволення освітніх та кваліфікаційних потреб студента, поглиблення підготовки за спеціальністю та ефективного використання можливостей закладу. При цьому вибіркові дисципліни можуть бути структуровані у тематичні блоки (модулі): у такому разі студент обирає не окремий курс, а цілий блок дисциплін, які після вибору автоматично включаються до його навчального плану як обов'язкові для вивчення. Окрім того, навчальні плани можуть передбачати різні категорії вибіркових компонентів – наприклад, дисципліни загальної підготовки (міждисциплінарні чи гуманітарні курси за вибором університету) та професійно орієнтовані дисципліни за спеціальністю. Такий підхід забезпечує широку варіативність траєкторій навчання, одночасно гарантуючи відповідність підготовки стандартам спеціальності.

Контроль за дотриманням права студентів на вибір здійснюється в рамках внутрішньої та зовнішньої систем забезпечення якості освіти. Зокрема, Національне агентство із забезпечення якості вищої освіти (НАЗЯВО) у критеріях акредитації вимагає, щоб заклад забезпечував реальну можливість вибору: експертна група під час оцінювання освітньої програми аналізує кількість і різноманітність вибіркових дисциплін, запропонованих студентам, а також процедури, визначені у закладі вищої освіти для реалізації цього вибору. Якщо «вибір» зводиться до формальності (наприклад, пропонується надто мало альтернатив або всі студенти фактично змушені слухати один і той самий курс), це розглядається як недолік освітньої програми. Таким чином, актуальна нормативна база зобов'язує заклади вищої освіти не лише включити до програм необхідний варіативний компонент, але й створити дієвий механізм його впровадження – прозорий, зручний для студентів і підконтрольний з точки зору якості освіти.

1.2 Автоматизовані системи вибору вибіркових дисциплін: досвід університетів

Впровадження права на вибір значної частини дисциплін вимагає налагодження чіткої організаційної процедури, особливо у великих університетах. Щоб ефективно поінформувати всіх студентів про доступні курси, зібрати їхні побажання та сформувати розклад і групи, більшість закладів вищої освіти використовують автоматизовані інформаційні системи. Сучасна практика українських університетів показує різноманітні підходи до такого електронного вибору дисциплін. За результатами порівняльного аналізу, проведеного у 2022–2023 рр., різні заклади вищої освіти України впровадили різні платформи для вибору освітніх компонентів: одні розробили власні веб-системи для цієї мети, інші адаптували наявні платформи, зокрема Moodle, або придбали сторонні рішення [14]. Наприклад, Національний авіаційний університет (НАУ) та НТУУ «КПІ ім. І. Сікорського» створили внутрішні онлайн-системи для вибору дисциплін, інтегровані зі своїми порталами (у КПІ функцію вибору реалізовано в системі «Електронний кампус (МуКРІ)»). Львівський національний університет ім. І. Франка використовує комерційну систему «Деканат», що була розроблена на замовлення університету приватною фірмою. Вона надає кожному студентові персональний електронний кабінет, через який здійснюється запис на курси. Зокрема, вибір загальноуніверситетських дисциплін у ЛНУ відбувається онлайн на платформі «Деканат» у два етапи, з можливістю зміни вибору протягом першої хвилі та автоматичним розподілом тих, хто не визначився, після завершення другої хвилі. Така поетапна процедура (дві хвилі вибору з моніторингом наповнення груп, а далі третій етап – рандомізований розподіл студентів, які не обрали вчасно) дозволяє сформувати навчальні групи оптимального розміру (в ЛНУ встановлено мінімум 25 і максимум 200 осіб на одну вибіркову дисципліну) та забезпечити, щоб кожен студент був зарахований на один із запропонованих курсів. Після цього обрані дисципліни

фіксуються як частина індивідуального плану і не підлягають заміні, що спонукає здобувачів вищої освіти відповідально ставитися до вибору [8].

Іншим прикладом є Запорізький національний університет (ЗНУ), де процес вибору повністю інтегровано в середовище електронного навчання Moodle. Студенти ЗНУ здійснюють вибір шляхом персонального електронного голосування в системі Moodle – спеціальний онлайн-опитувальник дозволяє відмітити бажані дисципліни, після чого адміністрація збирає результати та формує групи. Такий підхід мінімізує навантаження на адміністраторів, використовуючи вже існуючу LMS-платформу. В інших університетах створені або впроваджені аналогічні системи вибору: так, Запорізький національний технічний університет розробив модуль «Smart University», що діє як електронний кабінет студента, у функціонал якого також входить і можливість обрати вибіркові компоненти, а у Київському національному університеті ім. Т. Шевченка функціонує система «Тритон», через яку студенти реєструються на вибіркові курси. Деякі заклади, не маючи власного програмного забезпечення, використовують інтеграції з хмарними сервісами – найпростіший варіант це збір заявок через онлайн-форми. Наприклад, в окремих університетах вибір дисциплін проводився шляхом голосування через Google Forms – студенти підтверджують вибір дисциплін через форму Google, після чого навчальний відділ опрацьовує результати. Аналогічно можуть застосовуватися інструменти Microsoft (Forms, Excel Online тощо) в рамках корпоративних порталів Office 365.

Таким чином, українська практика наразі охоплює весь спектр рішень: від модулів у системах управління навчанням (LMS) і власних CRM-платформ університетів до використання сторонніх сервісів. У всіх випадках мета таких систем – надати студентам зручний інтерфейс для ознайомлення з каталогом доступних вибіркових дисциплін та подачі свого вибору онлайн, а адміністрації – засоби автоматично підсумувати результати і прийняти управлінські рішення (сформувати групи, затвердити індивідуальні плани тощо). Важливо відзначити, що зарубіжні університети вже давно

застосовують подібні інформаційні системи для реєстрації на курси. Як правило, це комплексні Student Information Systems або модулі в кампус-порталах, часто з підтримкою мобільних застосунків. Студенти очікують можливості здійснювати всі академічні операції онлайн у централізованому «one-stop-shop» середовищі – від оплати навчання і перегляду оцінок до реєстрації на дисципліни. Тому новітні системи вибору курсів усе частіше пропонують мобільні застосунки або адаптований під смартфони вебінтерфейс, що підвищує доступність і оперативність процесу. Наприклад, у багатьох університетах США вибір/реєстрація на курси здійснюється через веб-портал студентського кабінету, який також доступний у вигляді мобільного застосунку; компанії-розробники освітніх систем (Ellucian, Oracle та ін.) відзначають, що реєстрація на курси є однією із найзатребуваніших функцій у структурі мобільного кампус-застосунку [5].

1.3 Переваги впровадження інформаційних систем вибору дисциплін

Автоматизація процесу вибору вибіркових освітніх компонентів дає низку суттєвих переваг для всіх учасників освітнього процесу:

Прозорість і підконтрольність. Електронна система забезпечує чіткі правила та однакові умови вибору для усіх здобувачів вищої освіти, мінімізує ризики суб’єктивного впливу або помилок при ручному збиранні заяв. Усі кроки (від подання заявки до формування груп) фіксуються в системі, що підвищує довіру до результатів і відповідає принципам академічної прозорості. Студенти можуть самостійно відстежувати статус обраних курсів (кількість записаних, сформовані групи тощо) у режимі реального часу, що робить процес більш відкритим [8].

Зручність для студентів. Можливість обирати дисципліни онлайн через зручний інтерфейс (веб-портал або мобільний застосунок) значно підвищує задоволеність здобувачів вищої освіти. Вони можуть ознайомитися з силабусами вибіркових компонент, порівняти варіанти і зробити усвідомлений вибір у комфортний для себе час, без необхідності відвідувати

деканат чи заповнювати паперові заяви. Сучасні студенти очікують, що всі основні університетські сервіси (в тому числі вибір вибіркових освітніх компонент) будуть доступні їм з телефону чи ноутбука в будь-який час. Таким чином, автоматизована система вибору дисциплін покращує загальний досвід взаємодії студентів з університетом і сприяє реалізації їхніх освітніх прав у повному обсязі.

Ефективність і швидкість адміністрування. Для адміністрації та навчально-методичних підрозділів використання ІТ-систем значно спрощує збір і обробку інформації про вибір здобувачів вищої освіти. Раніше цю роботу могли виконувати вручну (шляхом опитування в групах або збору паперових форм), що займало багато часу і було більш схильне до помилок. Натомість автоматизована система одразу підсумовує кількість виборів по кожному курсу, попереджає перевищення лімітів або недостатню наповненість, може автоматично призначити студентів, які не обрали вчасно, до тих груп, які сформувалися. Це прискорює процес складання розкладів і формування індивідуальних навчальних планів. До того ж, керівництво отримує оперативні дані для прийняття рішень – наприклад, які курси користуються найбільшим попитом і потребують відкриття додаткових груп, а які майже не обрані (і можливо, потребують перегляду або заміни в майбутньому).

Гнучкість та відповідність освітнім потребам. Завдяки систематизації вибіркового компоненту, університет може пропонувати ширший каталог дисциплін, у тому числі міжфакультетські курси. Студенти реально формують власні траєкторії навчання, комбінуючи курси за інтересами та потребами. Автоматизована система дозволяє забезпечити цю гнучкість без втрати керованості: навіть якщо тисячі студентів одночасно роблять різний вибір, система обробляє його та гарантує, що кожен отримає певний набір курсів. Таким чином, заклад виконує вимоги закону щодо мінімального 25% вибіркового компоненту і сприяє розвиткові індивідуальних компетентностей студентів. Це відповідає сучасній концепції індивідуальної освітньої траєкторії, роль якої посилена останніми змінами в законодавстві (розвиток

індивідуальних траєкторій згадується серед пріоритетів освітнього процесу) та критеріями якості освіти [9].

1.4 Труднощі та проблеми впровадження автоматизованих систем

Попри очевидні переваги, впровадження автоматизованих систем вибору дисциплін може супроводжуватися низкою викликів:

Організаційні та технічні складнощі. Розробка або адаптація відповідного програмного забезпечення потребує часу, фінансових ресурсів і наявності ІТ-фахівців. Не всі університети мають змогу самостійно створити якісну систему – доводиться або закуповувати готові рішення, або використовувати тимчасові інструменти (на кшталт електронних форм). Інтеграція нової системи з існуючою ІТ-інфраструктурою закладу вищої освіти (єдиною базою студентів, розкладом, бухгалтерією тощо) – непросте завдання, яке потребує узгодження різних підрозділів. Відсутність єдиного державного програмного продукту для підтримки вибору дисциплін змушує кожен заклад шукати власне рішення, що призводить до різної якості та функціоналу систем по країні.

Питання культури та навчання користувачів. Впровадження нової цифрової системи вимагає навчання як студентів, так і співробітників (деканатів, кураторів). Необхідно підготувати інструкції, провести роз’яснення, забезпечити підтримку користувачів під час перших циклів вибору. Якщо цього не зробити належним чином, можливі помилки або пасивність: частина студентів може не виконати вибір вчасно через необізнаність або технічні труднощі. Як наслідок, адміністрації доводиться вручну вибирати дисципліни за таких студентів, що створює напруженість. Досвід показує, що перші роки впровадження можуть бути експериментальними – університети стикаються з необхідністю вдосконалення правил та інтерфейсу системи, враховуючи зворотний зв’язок від користувачів.

Ризики нерівномірного розподілу та задоволеності вибором. Одне з практичних питань – як справедливо розподілити студентів на популярні

курси, де число охочих перевищує максимально допустиму кількість. Системи вирішують це по-різному (принцип «перший прийшов – перший обслуговований», багатократні хвилі вибору, жеребкування тощо), але кожен підхід має недоліки і може викликати нарікання з боку студентів, які не потрапили на бажаний курс. Необхідно продумати механізми пріоритезації або пропонувати еквівалентні альтернативи, інакше частина студентів відчуватиме незадоволення. Крім того, у випадку недостатньої кількості заявок на певні дисципліни виникає потреба їх скасування або об'єднання груп, що також впливає на навчальні плани окремих студентів. Управління цими ситуаціями додає складності та потребує гнучкості від адміністратора системи.

Матеріально-технічні та соціальні аспекти. Запуск електронної системи вимагає стабільної роботи серверів, особливо в пікові періоди (коли одночасно заходять тисячі користувачів, щоб здійснити вибір). Потрібно заздалегідь подбати про навантаження, інакше можливі збої, які підривають довіру до системи. З соціальної точки зору, важливо забезпечити рівний доступ: хоча більшість студентів мають інтернет і гаджети, завжди можуть бути окремі випадки відсутності доступу у визначений час, технічних проблем на боці користувача тощо. Університет мусить передбачити резервні варіанти (наприклад, можливість звернутися в деканат офлайн) на випадок форс-мажору, інакше право на вибір окремих осіб може бути обмежене технічними причинами.

Інституційна інерція та нормативні невизначеності. Деякі заклади повільно впроваджують нововведення: навіть після прийняття нового закону не всі відразу забезпечили 25% вибіркових компонентів і прозорий механізм їх вибору. Опитування показували, що ще кілька років тому частина університетів фактично обмежувала свободу вибору студентів (пропонували мінімум альтернатив або формально включали вибіркові дисципліни без реального вибору). Подолання такої інерції вимагало зусиль з боку МОН та НАЗЯВО (через роз'яснення, рекомендації та вимоги акредитації). Додатково,

існують нюанси нормативного характеру, наприклад: як враховувати вибір дисциплін іншої освітньої програми, чи можна дозволити вибір кредитів понад встановлений мінімум, як оформлювати результати вибору документально. Відповіді на ці питання мають бути відображені у внутрішніх положеннях закладу вищої освіти і погоджені з чинними стандартами. Розробка таких положень також є нетривіальним завданням, що потребує методичної підтримки.

Висновки до розділу 1

Вибіркові освітні компоненти є невід'ємною частиною сучасних освітніх програм у закладах вищої освіти, а їх ефективна реалізація значною мірою залежить від налагодження нормативної бази та впровадження зручних інформаційних систем. Нормативні акти України (закон, стандарти, рекомендації МОН) встановили необхідні рамки та гарантії для вільного вибору дисциплін студентами. Практика університетів показує, що автоматизовані системи здатні успішно забезпечити цей процес, пропонуючи широкий функціонал – від електронного каталогу курсів до алгоритмів розподілу на них. Разом з тим, впровадження таких систем потребує продуманої стратегії, ресурсів та підтримки, але у підсумку дає значний виграш у якості освітнього процесу, підвищує гнучкість навчання і задоволеність студентів. За умов подальшого вдосконалення (наприклад, використання елементів штучного інтелекту для рекомендацій студентам, як пропонують дослідники), інформаційні системи вибору вибіркових дисциплін стануть ще ефективнішими інструментами персоналізації вищої освіти в Україні.

РОЗДІЛ 2. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИБОРУ ВИБІРКОВИХ ОСВІТНІХ КОМПОНЕНТ

2.1 Підготовка до розробки застосунку

Для розробки застосунку вибору вибіркових освітніх компонент спочатку необхідно здійснити підготовку необхідних середовища та інструментів. Основними технологіями, обраними для фронтенду, стали фреймворк Angular (з мовою програмування TypeScript) та платформа Node.js як середовище виконання JavaScript-коду. Встановлення Node.js є необхідним кроком, оскільки Angular CLI (Command Line Interface) працює як пакет Node.js і використовує менеджер пакетів npm для встановлення залежностей. Після інсталяції Node.js було глобально встановлено Angular CLI командою nmp install –g @angular/cli. Це дає можливість створювати та керувати Angular-проектами з командного рядка.

Далі, за допомогою Angular CLI, ініціалізовано новий проект за шаблоном типового Angular-застосунку. Зокрема, виконано команду ng new із зазначенням назви проекту, яка автоматично згенерувала базову структуру директорій і файлів. Ця структура містить кореневу директорію проекту з конфігураційними файлами (angular.json, package.json тощо), а також папку src/ із початковим вихідним кодом. У папці src знаходиться підпапка app з стандартним початковим компонентом AppComponent (файли app.component.ts, app.component.html та app.component.scss). Також створено директорію assets/ для статичних ресурсів і папку environments/ з файлами налаштувань середовища (рис. 2.1).

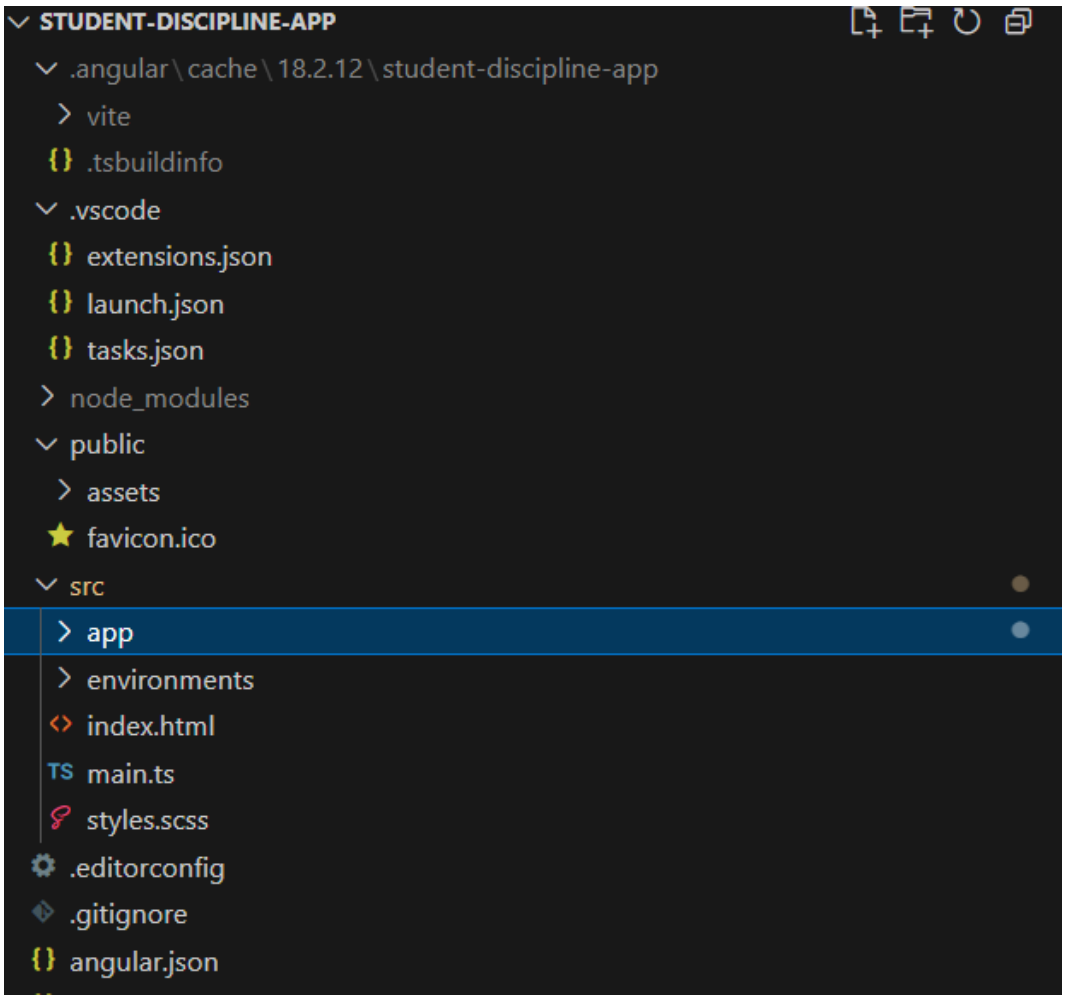


Рис. 2.1. Структура Angular-застосунку у редакторі Visual Studio Code.

Для розробки студентського веб-застосунку Angular було обрано невипадково. Адже цей фреймворк надає структурований підхід до розробки односторінкових застосунків і має розвинену екосистему. Angular підтримує модульність та компонентно-орієнтований дизайн, що полегшує розподіл функціоналу на логічні частини й повторне використання компонентів. Крім того, Angular пропонує вбудовані засоби для роботи з формами (Reactive Forms і Template-driven Forms), маршрутизації та сервіси для HTTP-запитів – усе це важливо для інтеграції з бекенд-сервісами (такими як Firebase чи Google Apps Script у нашому випадку). Використання TypeScript у Angular підвищує надійність коду завдяки статичній типізації, що особливо корисно в проектах з довготривалою підтримкою. Таким чином, Angular забезпечує впорядковану структуру коду, високу швидкість розробки й зручність підтримки, а також

добре підходить для створення динамічного інтерфейсу, необхідного для системи вибору вибіркових дисциплін.

У процесі підготовки до розробки було налаштовано й зовнішні служби, які взаємодіятимуть із Angular-застосунком. По-перше, для керування автентифікацією користувачів (студентів) створено проект у сервісі Google Firebase. У консолі Firebase активовано модуль Authentication і налаштовано Sign-in providers (увімкнення авторизації за допомогою електронної пошти та пароля для облікових записів студентів) (рис. 2.2).

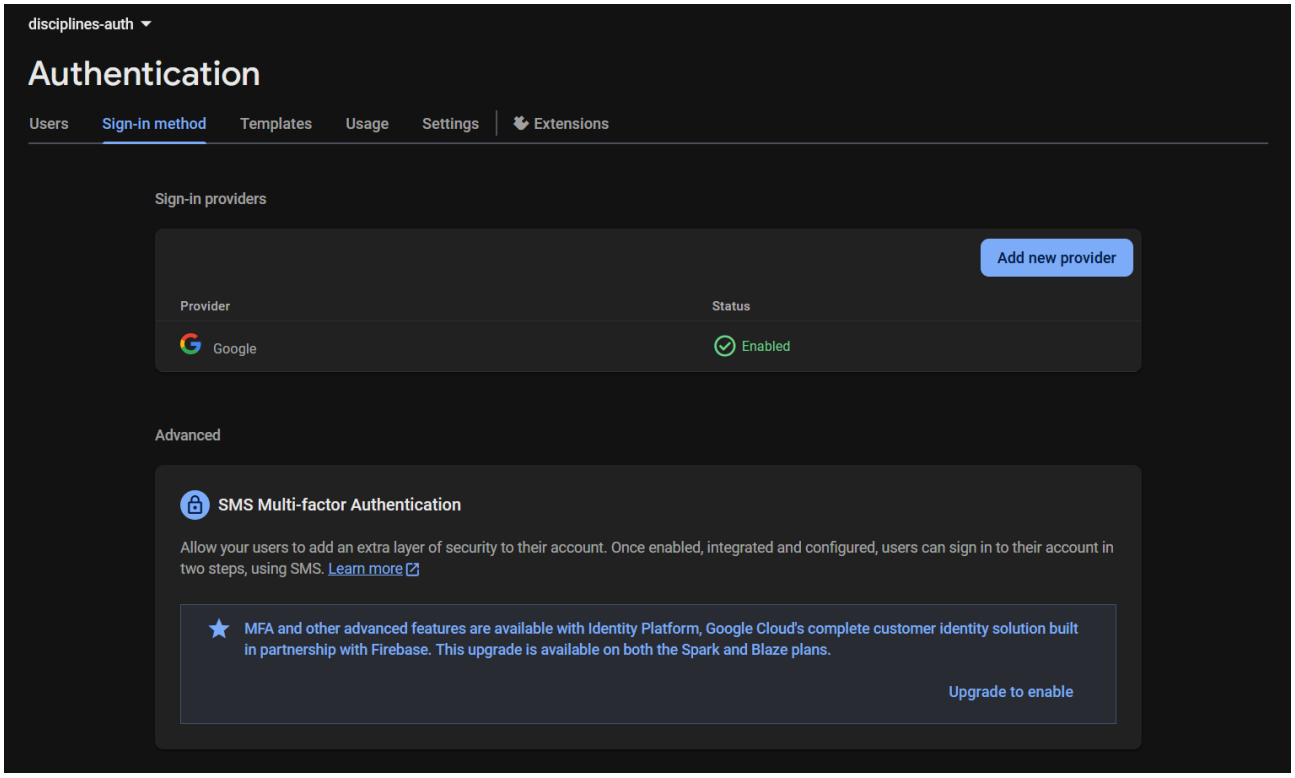
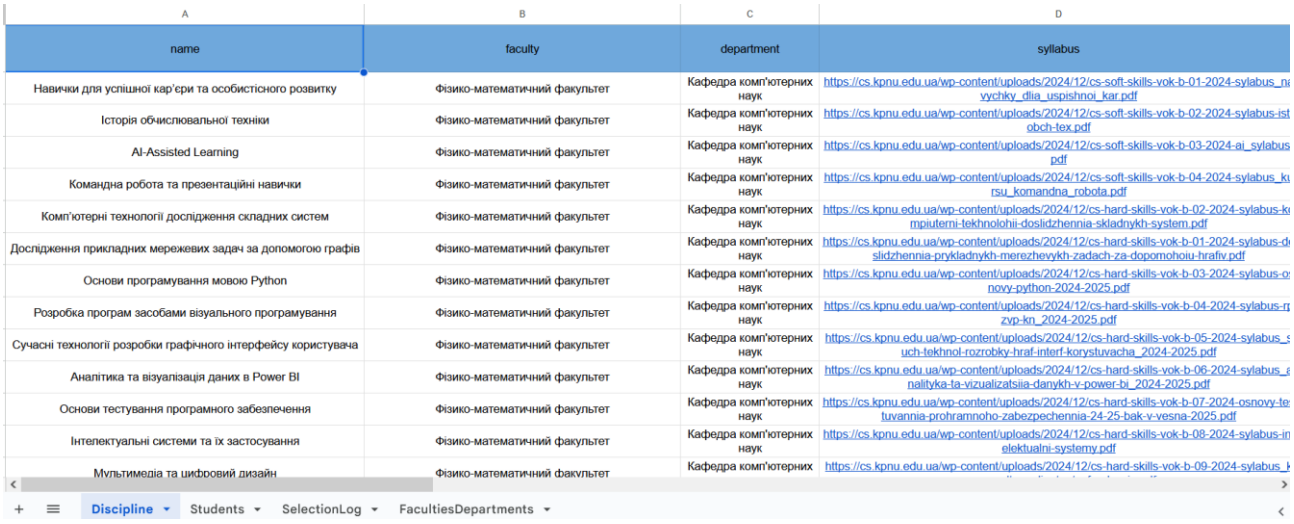


Рис. 2.2. Сторінка налаштувань із вибраним методом автентифікації.

По-друге, для зберігання даних було підготовлено електронну таблицю в сервісі Google Sheets. Ця таблиця містить два ключових аркуші: Discipline для переліку вибіркових освітніх компонентів та Students для даних про студентів та обрані ними дисципліни. Структура цього документа передбачає, що в аркуші Discipline розміщено колонки з інформацією про навчальні дисципліни (назва дисципліни, факультет, кафедра та посилання на силабус) (рис. 2.3), а в аркуші Students – колонки для збереження інформації про студента (ім’я з прізвищем, електронна пошта, факультет, форма навчання,

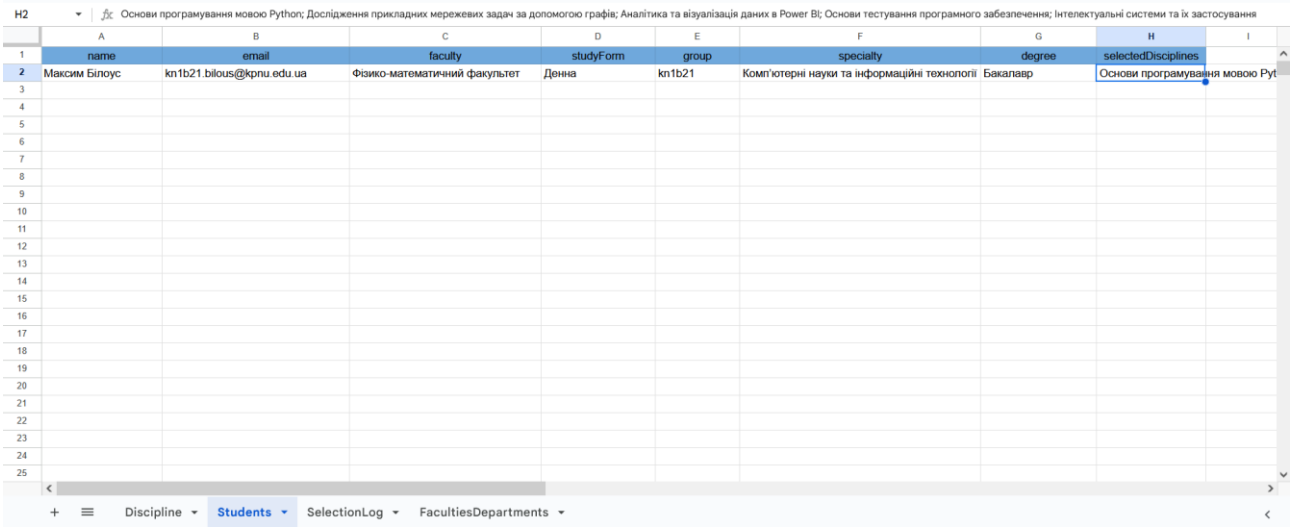
шифр групи, спеціальність, ступінь освіти) та перелік обраних ним дисциплін (рис. 2.4).



name	faculty	department	syllabus
Навички для успішної кар'єри та особистісного розвитку	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-soft-skills-vok-b-01-2024-syllabus_n... vychky_dlia_uspishnoi_kar.pdf
Історія обчислювальної техніки	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-soft-skills-vok-b-02-2024-syllabus-ist... obch-tex.pdf
AI-Assisted Learning	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-soft-skills-vok-b-03-2024-ai_sylabus... pdf
Командна робота та презентаційні навички	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-soft-skills-vok-b-04-2024-syllabus_k... rsu_komandna_robota.pdf
Комп'ютерні технології дослідження складних систем	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-02-2024-syllabus-k... mpiuterni-tekhnolohii-doslidzhennia-skladnykh-system.pdf
Дослідження прикладних мережевих задач за допомогою графів	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-01-2024-syllabus-d... slidzhennia-prykladnykh-merezhevykh-zadach-za-dopomohoiu-hrafiv.pdf
Основи програмування мовою Python	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-03-2024-syllabus-o... novy-python-2024-2025.pdf
Розробка програм засобами візуального програмування	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-04-2024-syllabus-r... zvp-kn_2024-2025.pdf
Сучасні технології розробки графічного інтерфейсу користувача	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-05-2024-syllabus_s... uch-tekhnol-rozrobky-hraf-interf-korystuvacha_2024-2025.pdf
Аналітика та візуалізація даних в Power BI	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-06-2024-syllabus_a... nalityka-ta-vizualizatsiia-danykh-v-power-bi_2024-2025.pdf
Основи тестування програмного забезпечення	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-07-2024-osnovy-te... tuvannia-prohramnoho-zabezpechennia-24-25-bak-v-vesna-2025.pdf
Інтелектуальні системи та їх застосування	Фізико-математичний факультет	Кафедра комп'ютерних наук	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-08-2024-syllabus-in... elektualni-systemy.pdf
Мультимедіа та цифровий дизайн	Фізико-математичний факультет	Кафедра комп'ютерних	https://cs.kpnu.edu.ua/wp-content/uploads/2024/12/cs-hard-skills-vok-b-09-2024-syllabus_k...

Discipline | Students | SelectionLog | FacultiesDepartments

Рис. 2.3. Структура аркуша Discipline таблиці Google Sheets.



H2: Основи програмування мовою Python; Дослідження прикладних мережевих задач за допомогою графів; Аналітика та візуалізація даних в Power BI; Основи тестування програмного забезпечення; Інтелектуальні системи та їх застосування

name	email	faculty	studyForm	group	specialty	degree	selectedDisciplines
Максим Білоус	kn1b21.bilous@kpnu.edu.ua	Фізико-математичний факультет	Денна	kn1b21	Комп'ютерні науки та інформаційні технології	Бакалавр	Основи програмування мовою Pyt

Discipline | Students | SelectionLog | FacultiesDepartments

Рис. 2.4 Структура аркуша Students таблиці Google Sheets.

Підготувавши таким чином необхідне середовище (фреймворк Angular, службу автентифікації Firebase та таблиці даних Google Sheets), можна переходити безпосередньо до реалізації функціональних можливостей студентського застосунку для вибору вибіркових компонент.

2.2 Реалізація авторизації студента

У розробленій інформаційній системі авторизація студентів реалізована за допомогою Firebase Authentication із використанням сервісу Google Sign-In. Для цього в структурі застосунку було створено спеціальний Angular-сервіс

AuthService (додаток А), який відповідає за весь життєвий цикл аутентифікації користувача – від входу до збереження стану сесії.

Сервіс AuthService підключає Firebase через об’єкт Auth з бібліотеки @angular/fire/auth та ініціалізує Google-автентифікацію через GoogleAuthProvider. Метод LoginWithGoogle() викликає функцію signInWithPopup(), яка відкриває вікно входу через Google (рис. 2.5).

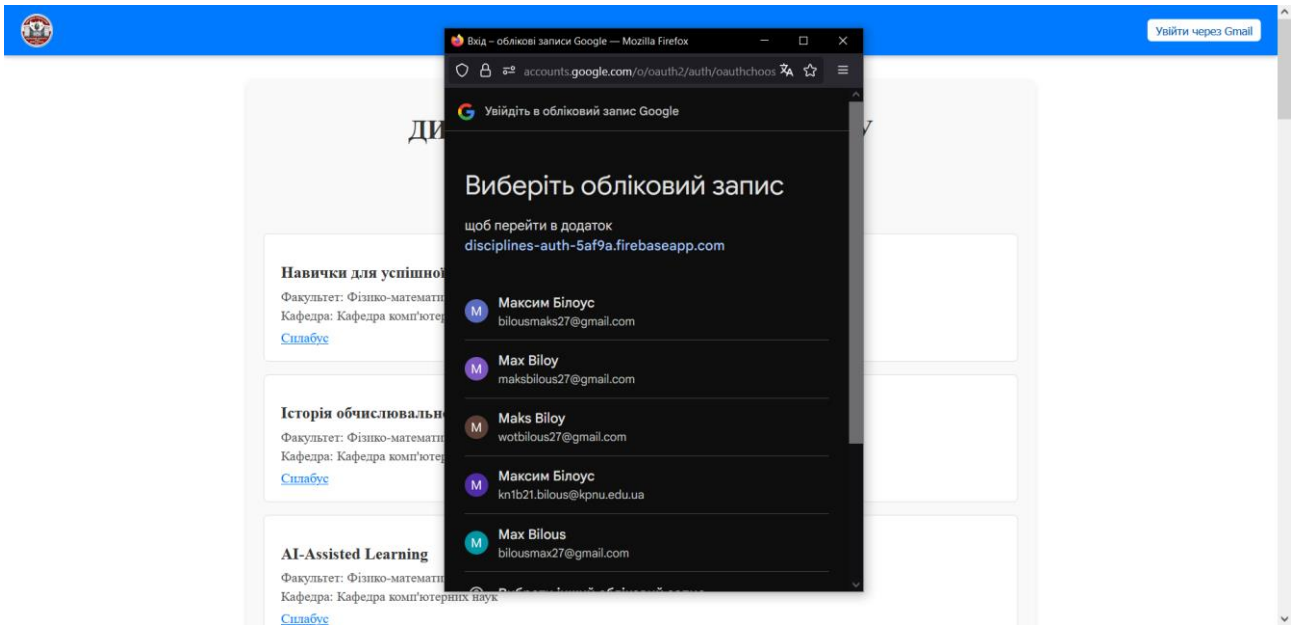


Рис. 2.5. Вікно входу через Google.

Після успішної авторизації виконується перевірка, чи email користувача належить до домену університету @kpnu.edu.ua. Якщо домен не відповідає заданому, вхід скасовується, і студент бачить повідомлення про відмову в доступі (рис. 2.6).

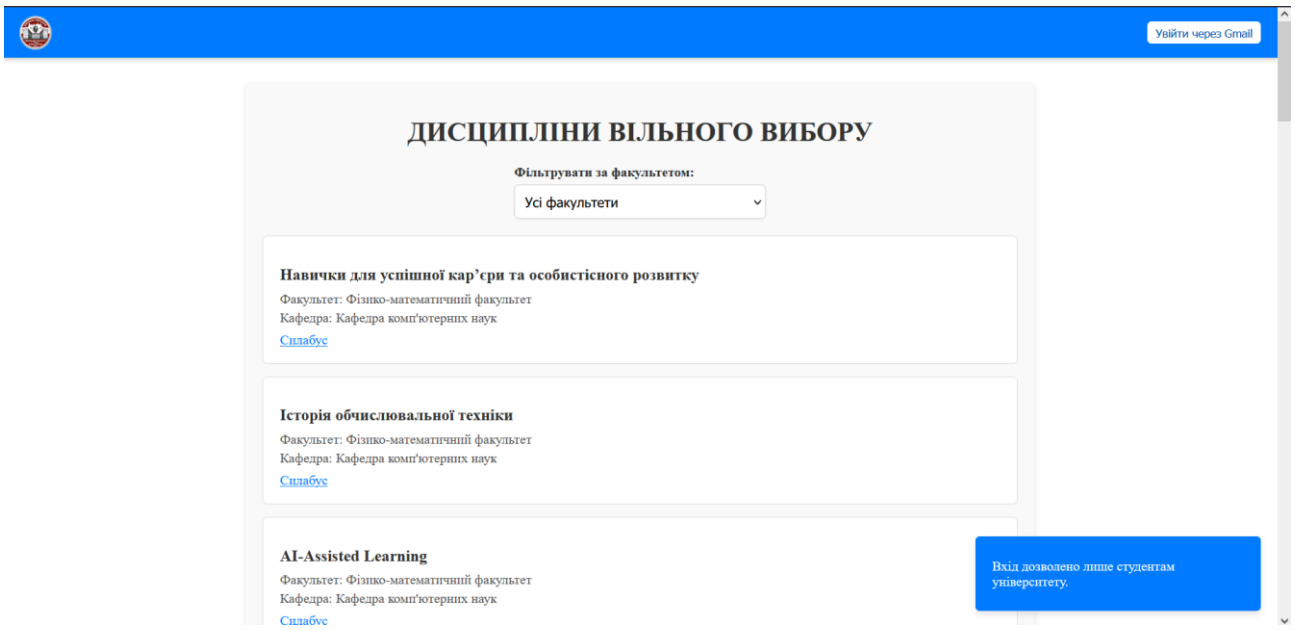


Рис. 2.6. Повідомлення про відмову в доступі.

Повідомлення про успішний або невдалий вхід виводяться через інший Angular-сервіс – MessageService, що реалізує систему коротких інформативних повідомлень для користувача.

Інформація про авторизованого студента зберігається у вигляді об'єкта {name, email} у поточному стані застосунку через BehaviorSubject. Це дає змогу підписатися на зміни стану автентифікації з будь-якого компонента або сервісу в застосунку. Крім того, при кожній зміні стану авторизації використовується спостерігач onAuthStateChanged(), який визначає, чи відбувся вхід або вихід користувача, й автоматично оновлює поточний стан userSubject. Таким чином, система завжди має актуальні дані про авторизованого студента.

Крім email та імені, сервіс AuthService додатково визначає шифр групи на основі структури адреси електронної пошти. Для цього використовується регулярний вираз, який знаходить підрядок у форматі, що відповідає типовому шифру групи. Результат зберігається як частина об'єкта користувача у методі getCurrentUser(), який повертає {name, email, group}. Після успішного входу в систему студент автоматично отримує доступ до функціоналу вибору дисциплін (рис. 2.7).

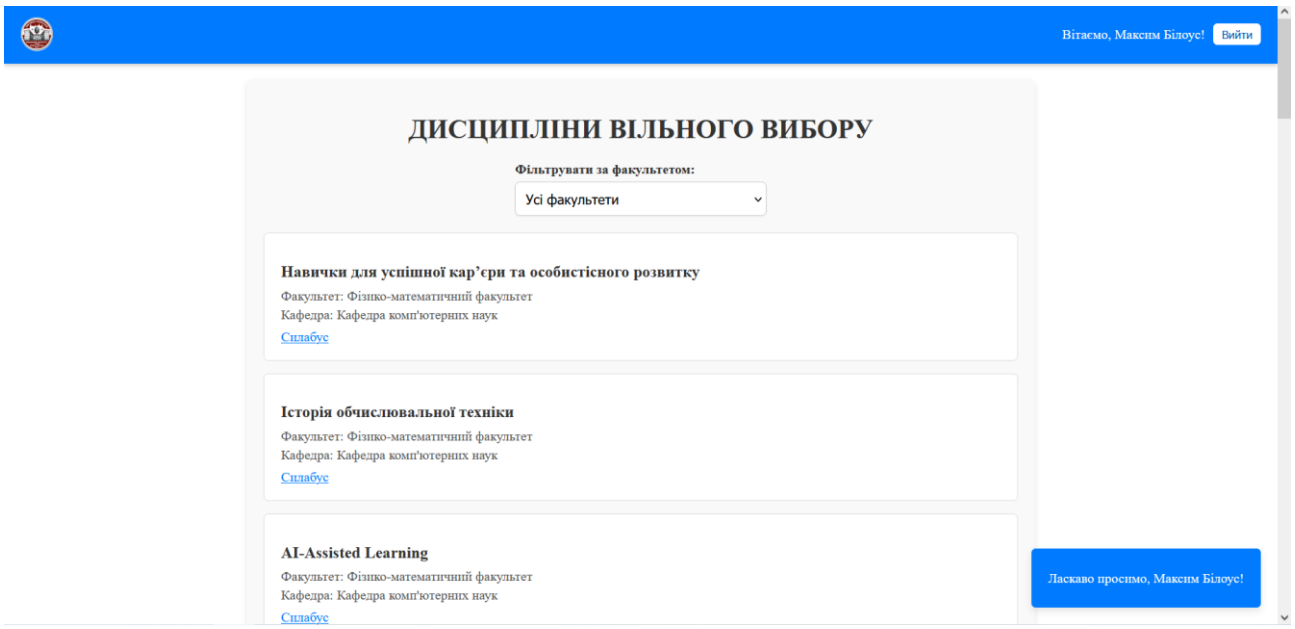


Рис. 2.7. Повідомлення про успішний вхід в систему.

2.3 Завантаження та фільтрація переліку дисциплін

Після авторизації студент отримує доступ до головного інтерфейсу системи, де відображається перелік вибіркових освітніх компонентів. Завантаження цього переліку реалізовано через окремий Angular-сервіс GoogleSheetsService, що відповідає за взаємодію з Google Sheets API (додаток Б). Застосунок звертається напряму до таблиці Google Sheets через REST API, використовуючи ключ доступу apiKey.

У сервісі реалізовано метод getDisciplines(), який формує HTTP GET-запит до конкретного аркуша Google Sheets з назвою Discipline. Запит має ось такий вигляд: https://sheets.googleapis.com/v4/spreadsheets/${this.sheetId}/values/Discipline?key=${this.apiKey}

У відповіді API повертає двовимірний масив рядків (матрицю), де перший рядок містить заголовки колонок, а наступні – записи про дисципліни. У таблиці Discipline зберігається базова інформація про дисципліни: назва дисципліни, факультет, кафедра та посилання на силабус. Отримані дані обробляються у сервісі та перетворюються у зручний для Angular-інтерфейсу

формат (масив об’єктів). Далі ці дані передаються компоненту DisciplineComponent, який є відповідальним за відображення списку дисциплін. З огляду на те, що загальна кількість вибіркових дисциплін може бути значною, у застосунку реалізовано механізм фільтрації дисциплін за факультетом і кафедрою. В компоненті DisciplineComponent було створено інтерфейс із двома випадаючими списками (dropdown), які дозволяють студенту обрати факультет та кафедру. При виборі факультету перелік кафедр автоматично оновлюється відповідно до наявних значень для цього факультету. Фільтрація виконується на клієнтському рівні без додаткових запитів до API: при зміні значення фільтра застосунок просто відображає підмножину дисциплін, які відповідають вибраним критеріям.

На рис. 2.8 зображено інтерфейс фільтрації дисциплін: студент бачить фільтр за факультетом та кафедрою, а під ним перелік вибіркових дисциплін, який динамічно оновлюється залежно від вибраних параметрів. Передбачено також можливість скидання фільтра, що дозволяє швидко повернутись до повного переліку дисциплін.

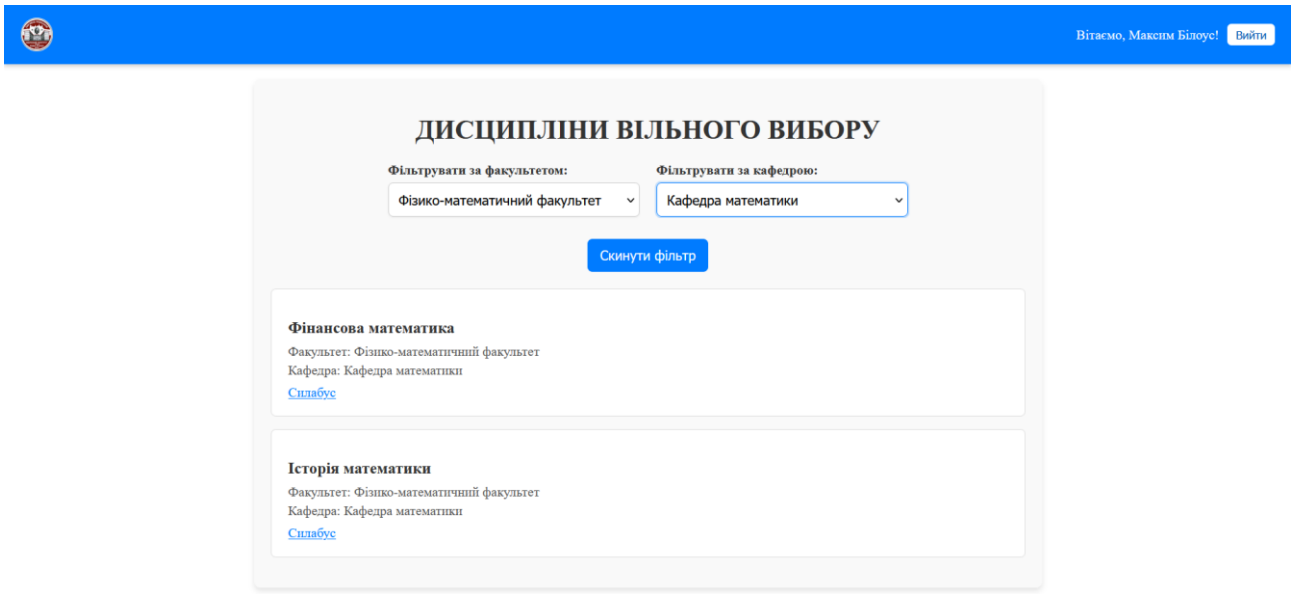


Рис. 2.8. Відфільтрований список дисциплін.

Таким чином, завдяки сервісу GoogleSheetsService реалізовано гнучке та ефективне завантаження даних із Google Sheets, а фільтрація забезпечує студенту зручну навігацію по доступних вибіркових дисциплінах. Такий

підхід дозволяє підтримувати просту, зрозумілу й адаптивну архітектуру фронтенду, не перевантажуючи бекенд або інтерфейс зайвими запитами.

2.4 Процес вибору дисциплін студентом

Процес вибору вибіркових освітніх компонент студентом реалізовано у два послідовні етапи: вибір дисциплін із переліку та заповнення персональних даних через форму. Цей підхід забезпечує поетапне збирання даних та підвищує зручність користування застосунком.

Усі взаємодії користувача в межах цього процесу реалізовані через два окремі Angular-компоненти – DisciplineComponent та StudentFormComponent. Перший відповідає за завантаження, фільтрацію та вибір дисциплін, а другий – за заповнення академічної інформації студента та остаточне збереження даних у таблицю.

2.4.1 Вибір дисциплін

У компоненті DisciplineComponent (додаток В) студенту відображається список доступних вибіркових дисциплін, завантажених з Google Sheets через GoogleSheetsService. Дисципліни обираються натисканням на їхні назви. Для цього використовується метод toggleDiscipline(), який додає назву дисципліни в список обраних, або її з нього (рис. 2.9).

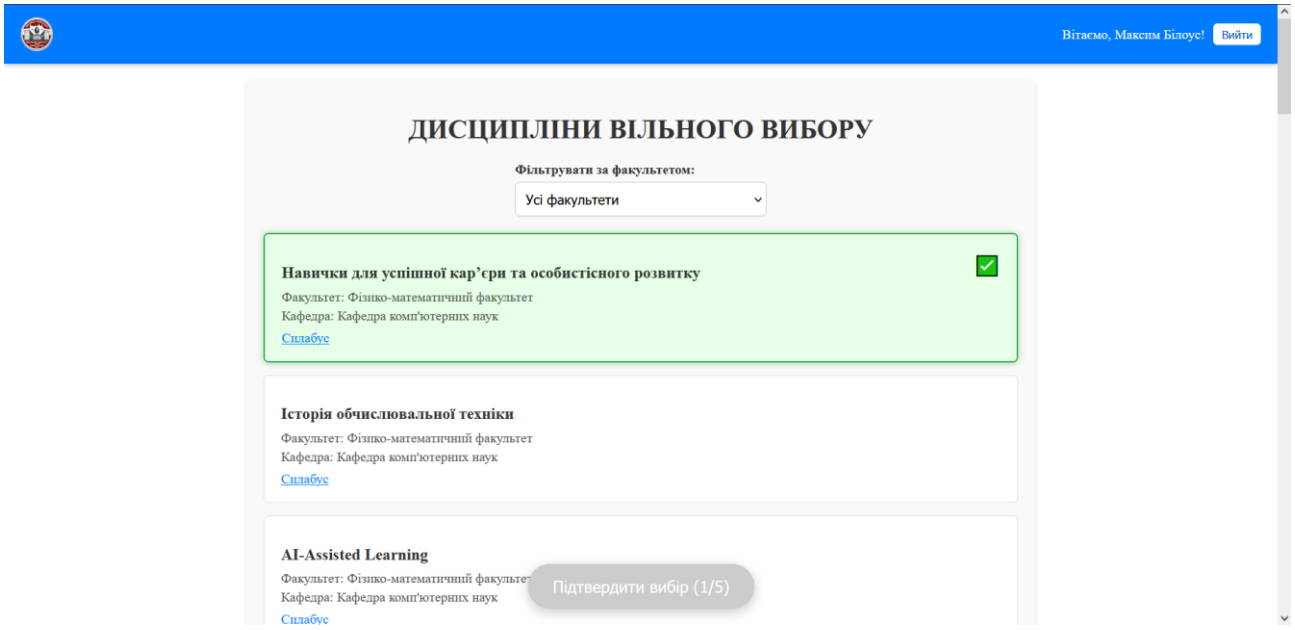


Рис. 2.9. Вибрана дисципліна.

Також було встановлено обмеження на максимальну кількість вибраних дисциплін (maxSelection), яке за замовчуванням становить 5 дисциплін, однак студент може змінити це обмеження через відповідне модальне вікно. Початок вибору відбувається вручну. Тобто при першій взаємодії система відкриває модальне вікно, яке дозволяє підтвердити (рис. 2.10 або змінити ліміт вибору (рис. 2.11).

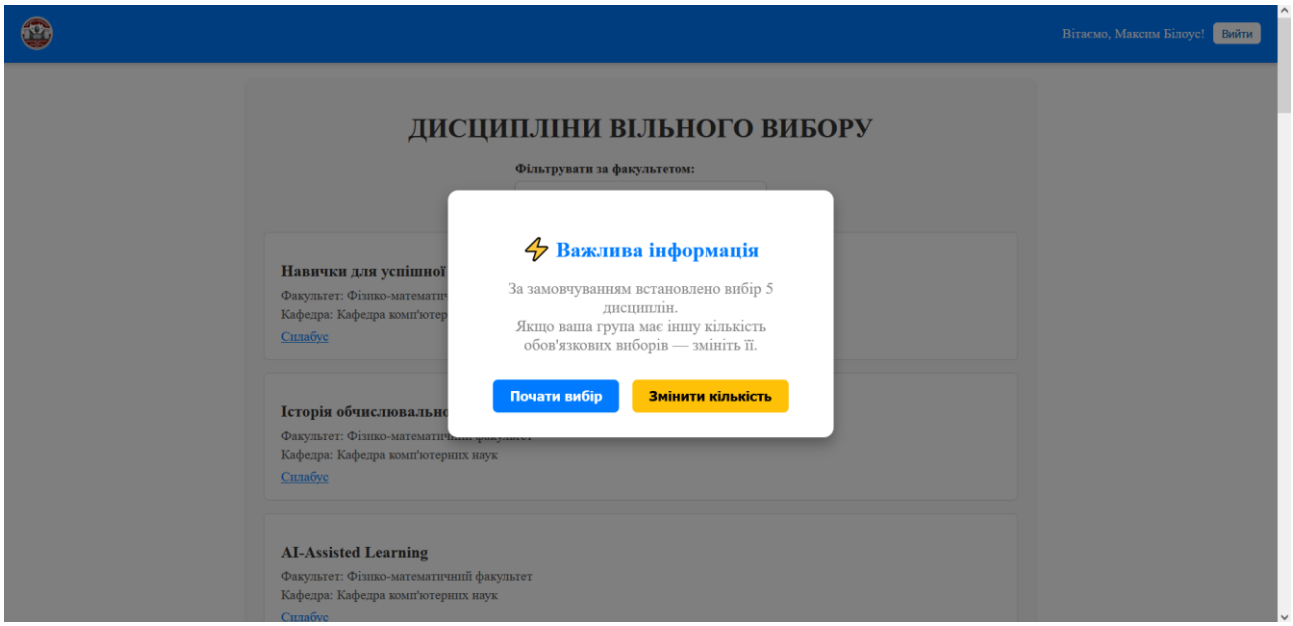


Рис. 2.10. Вікно підтвердження вибору за замовчуванням.

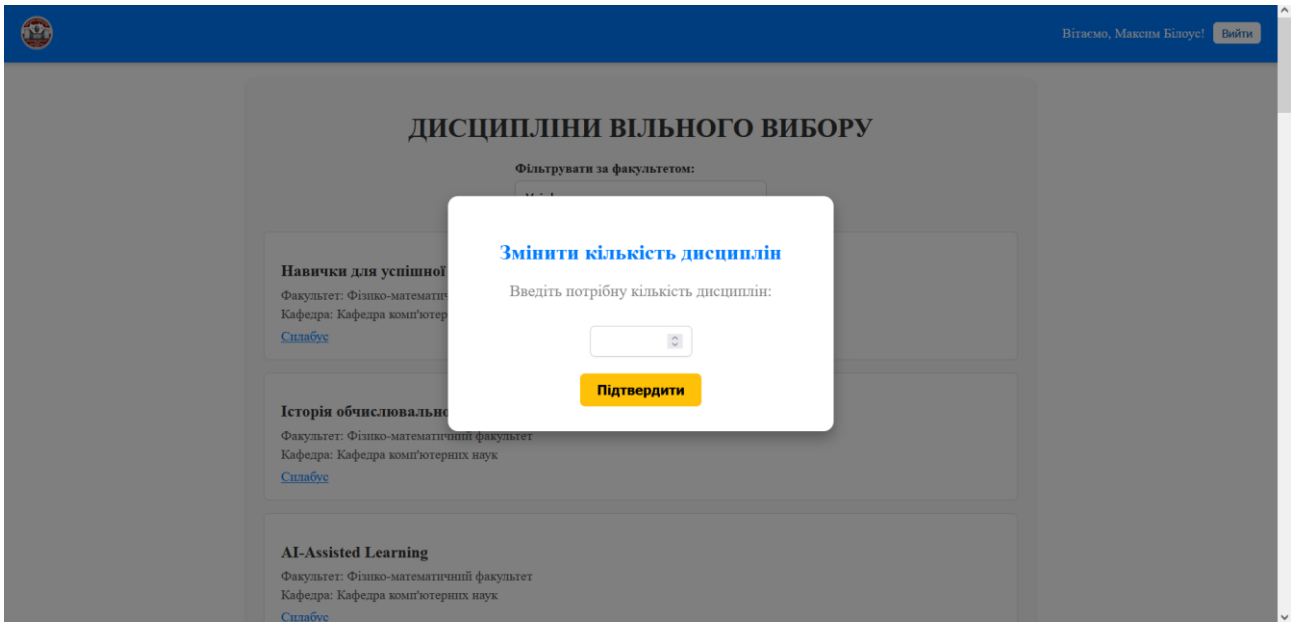


Рис. 2.11. Вікно зміни ліміту вибору.

Після формування списку вибраних дисциплін студент натискає кнопку підтвердження, яка активує метод confirmSelection(). У цьому методі

перевіряється, чи студент авторизований і чи вибрано хоча б одну дисципліну. Якщо умови виконано, перелік вибраних дисциплін тимчасово зберігається у localStorage браузера під ключем selectedDisciplines, після чого студент перенаправляється до компонента StudentFormComponent за маршрутом /student-form.

2.4.2 Заповнення форми

На другому етапі студенту пропонується форма для введення або уточнення особистих академічних даних (рис. 2.12).

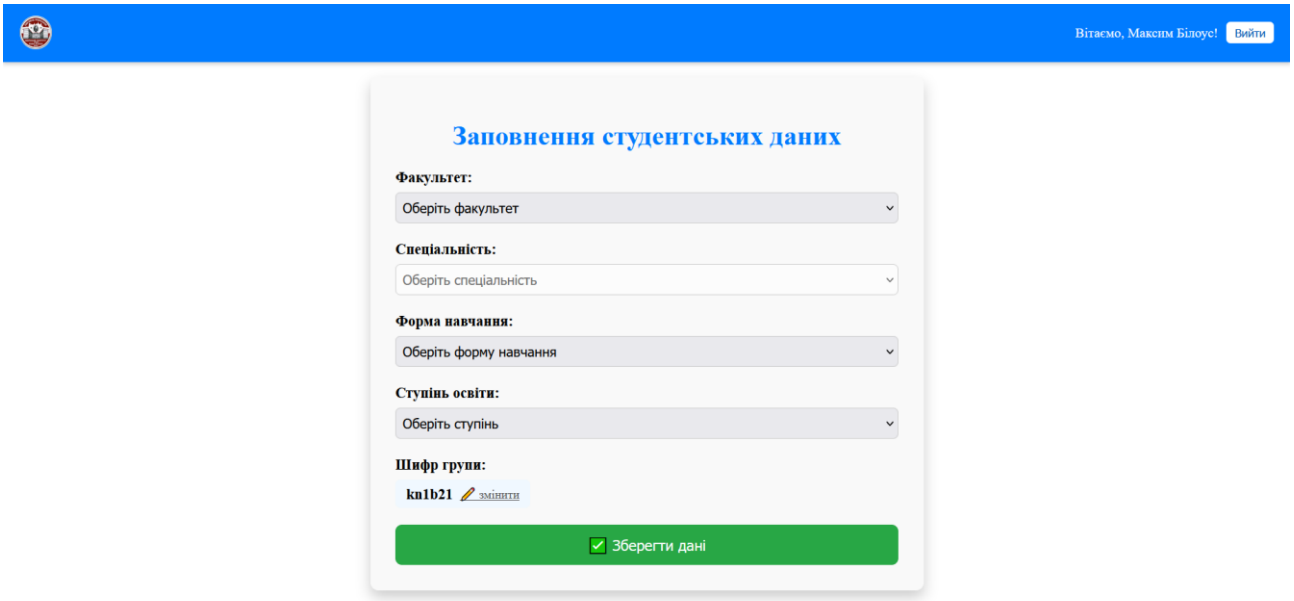


Рис. 2.12. Форма заповнення студентських даних.

Компонент StudentFormComponent (додаток Г) під час ініціалізації (ngOnInit) виконує декілька важливих дій. По-перше, із localStorage зчитується перелік дисциплін, вибраних на попередньому етапі. Якщо запис відсутній або пошкоджений, система ініціалізує порожній список. По-друге, за допомогою сервісу GoogleSheetsService завантажується таблиця FacultiesDepartments, яка містить факультети зі спеціальностями цих факультетів. На основі цієї таблиці формується структура faculties, де кожен факультет асоціюється з переліком спеціальностей, які використовуються для заповнення випадаючих списків у формі.

Форма містить поля для вибору або введення наступних параметрів: факультет, спеціальність, форма навчання, освітній ступінь та шифр

академічної групи. Перші чотири параметри обираються із заздалегідь визначених списків. Шифр групи, навпаки, може бути заповнений автоматично на основі електронної пошти студента (якщо з неї вдалося визначити групу), або студент може вручну змінити його — для цього передбачено перемикач useCustomGroup (рис. 2.13). Валідація форми включає перевірку на обов'язковість усіх полів, а також перевірку формату шифру групи (дозволяються лише латинські літери, цифри та дефіс).

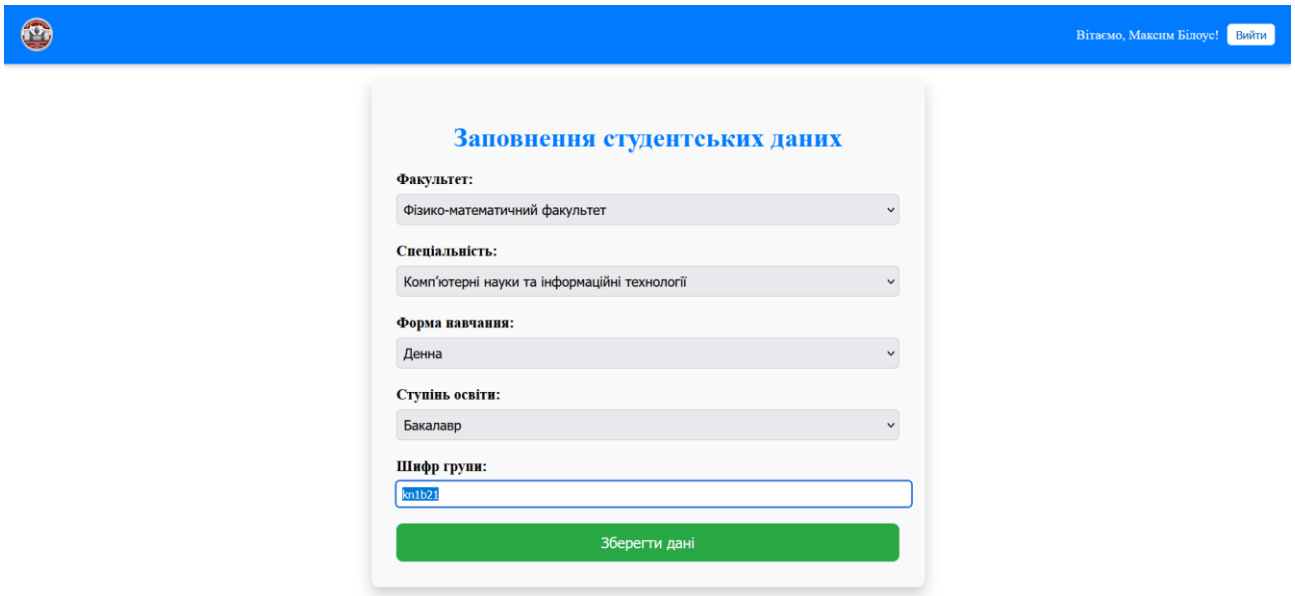


Рис. 2.13. Ручне введення шифру групи.

Після заповнення форми студент натискає на кнопку збереження даних, яка викликає метод submitForm(). У цьому методі знову перевіряється, чи користувач авторизований, чи всі поля форми заповнені, і чи шифр групи не містить заборонений символів. Якщо всі перевірки успішні, застосунок викликає метод updateOrAddStudent із сервісу GoogleSheetsService, який надсилає зібрані дані в аркуш Students таблиці Google Sheets через Google Apps Script Web App.

Після успішного збереження даних застосунок перенаправляє студента до окремої сторінки з повідомленням про завершення вибору (рис. 2.14). Ця сторінка реалізована у вигляді окремого компонента SuccessComponent (додаток Д). Вона містить два варіанти дій: студент може повернутись на сторінку вибору дисциплін (/discipline), якщо хоче змінити свій вибір, або

вийти з облікового запису та закрити вкладку браузера, завершивши сесію. Цей завершальний етап дозволяє студенту або оновити свій вибір, або залишити систему з упевненістю, що всі його дані збережені.

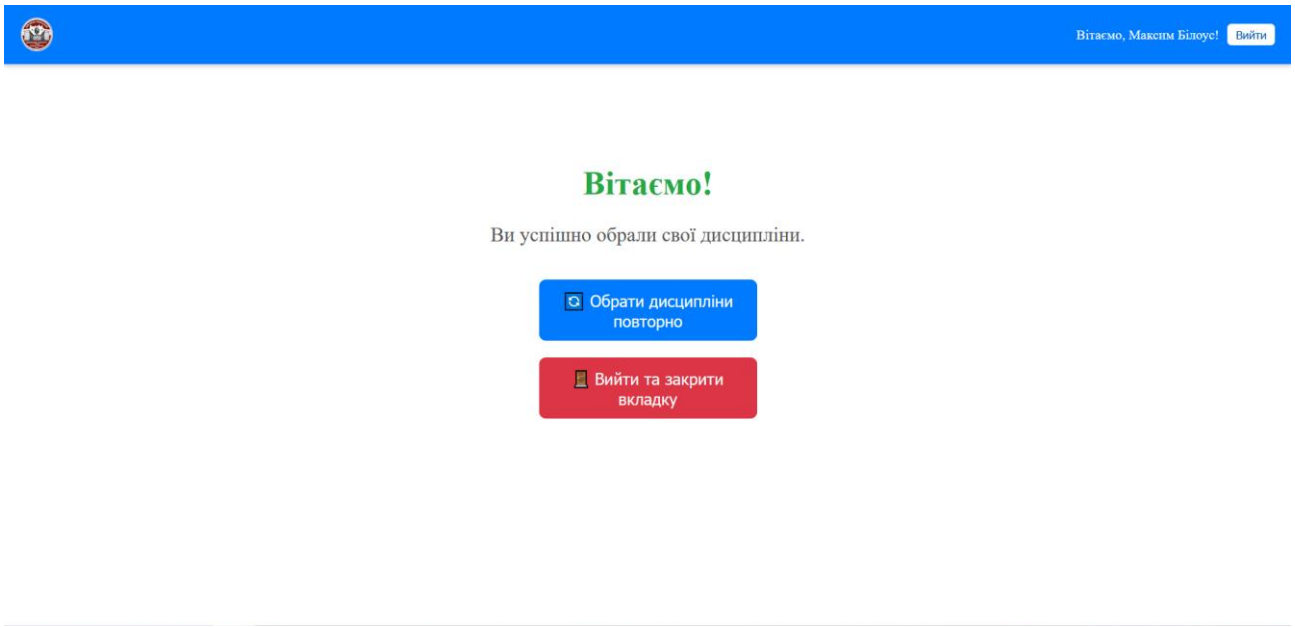


Рис. 2.14. Кінцева сторінка застосунку вибору дисциплін.

2.5 Збереження даних у таблицю Students

Збереження вибору студента відбувається після завершення заповнення форми в компоненті StudentFormComponent. Усі зібрані на попередніх етапах дані – ім’я та прізвище, електронна пошта, факультет, форма навчання, шифр групи, спеціальність, ступінь освіти та перелік вибраних дисциплін – передаються на серверну частину застосунку для зберігання у таблиці Students, розміщеній у Google Sheets.

Для цієї мети в застосунку реалізовано окремий метод updateOrAddStudent() у сервісі GoogleSheetsService. Цей метод формує HTTP POST-запит до розгорнутого Google Apps Script Web App, який діє як серверна точка прийому даних. Запит надсилається у форматі application/x-www-form-urlencoded, що дозволяє уникнути проблем із авторизацією CORS, характерних для взаємодії між клієнтським застосунком і сервісами Google.

У тілі запиту передаються всі необхідні параметри: action=updateOrAddStudent, name, email, group, faculty, specialty, formOfStudy, degree та selectedDisciplines. Дисципліни передаються у вигляді рядка, де назви

розділено крапками з комами. Після отримання запиту Apps Script обробляє його у функції doPost(e) й виконує пошук існуючого запису в таблиці Students на основі email як унікального ідентифікатора.

Якщо запис з таким email уже існує, його дані оновлюються відповідно до надісланих полів — це дозволяє студенту змінити свій вибір без створення дубліката. Якщо запису немає, скрипт створює новий рядок із усіма переданими даними. Таким чином, реалізовано єдину точку входу для обох операцій — оновлення та додавання, що спрощує логіку клієнтського коду та зменшує ймовірність помилок на сервері.

Структура таблиці Students підтримується у фіксованому форматі, де кожен стовпець відповідає одному з полів, що надсилаються з форми. Це дозволяє легко аналізувати або експортувати дані у майбутньому — наприклад, для створення звітів або подальшої фільтрації в адміністративній частині системи.

Після успішного завершення запиту застосунок очищає збережений у localStorage список вибраних дисциплін і перенаправляє студента на сторінку з підтвердженням. У разі помилки з боку Apps Script або мережі, студент отримує повідомлення з поясненням та може повторити спробу.

Таким чином, реалізована система зберігання забезпечує надійне та централізоване фіксування результатів вибору кожного студента, автоматично розрізняючи перше подання та оновлення, і повністю інтегрується з таблицею Google Sheets через безпечний та зручний механізм Web App.

Висновки до розділу 2

У цьому розділі було детально розглянуто процес розробки студентського функціоналу інформаційної системи автоматизованого вибору вибіркових освітніх компонентів. Було проведено повноцінну підготовку середовища розробки, зокрема встановлено та налаштовано Angular CLI, Node.js і Visual Studio Code, створено новий Angular-проект, а також підключено Firebase і Google Sheets як ключові взаємодії з користувачами та зберігання даних.

Система автентифікації реалізована на основі Firebase Authentication, що забезпечує доступ до функціоналу лише для студентів із корпоративною поштою університету. Авторизація відбувається через спеціальний сервіс AuthService, який спрощує обробку стану користувача в межах усього застосунку.

Завантаження переліку вибіркових дисциплін реалізовано через прямий запит до Google Sheets API за допомогою GoogleSheetsService. Система підтримує фільтрацію дисциплін за факультетами та кафедрами, що забезпечує гнучкий і зручний інтерфейс пошуку для студентів. Після вибору дисциплін студент переходить до окремої форми, де вводить або уточнює персональні академічні дані.

Збереження даних реалізовано через єдиний метод updateOrAddStudent(), який надсилає інформацію до серверного скрипта на Google Apps Script. На основі електронної пошти система визначає, чи потрібно оновити існуючий запис, чи створити новий. Усі поля (Ім’я та прізвище, електронна пошта, факультет, форма навчання, шифр групи, спеціальність, ступінь освіти та обрані дисциліни) записуються до таблиці Students у Google Sheets.

Після завершення процедури вибору студент потрапляє на сторінку підтвердження, де має змогу або повторно перейти до вибору дисциплін, або завершити сесію, вийшовши з акаунту.

Загалом реалізований функціонал забезпечує повний і зручний цикл вибору вибіркових компонентів зі сторони студента, гарантує коректне збереження даних і дозволяє інтегрувати його в загальну інформаційну систему університету.

РОЗДІЛ 3. АНАЛІЗ ДАНИХ ПРО ВИБІР ВИБІРКОВИХ ОСВІТНІХ КОМПОНЕНТІВ ЗА ДОПОМОГОЮ GOOGLE APPS SCRIPT

3.1 Підготовка до аналізу даних

Першим кроком аналізу є створення окремої копії даних у новому файлі електронної таблиці (рис. 3.1). Скрипт формує копію аркуша Students (який містить інформацію про вибір дисциплін студентами) у новому файлі Google Таблиць з фіксованою назвою «Дані вибору вибіркових навчальних дисциплін здобувачів вищої освіти». Важливо, що перед кожним таким копіюванням система перевіряє цільовий файл і видаляє у ньому попередній аркуш Students. Таким чином усуваються застарілі дані та попереджається дублювання аркушів у файлі. Лише після вилучення старого аркуша до файлу копіюється актуальний аркуш Students з основної таблиці. У результаті в зазначеному файлі міститься оновлена копія всіх необхідних даних про вибір вибіркових дисциплін.

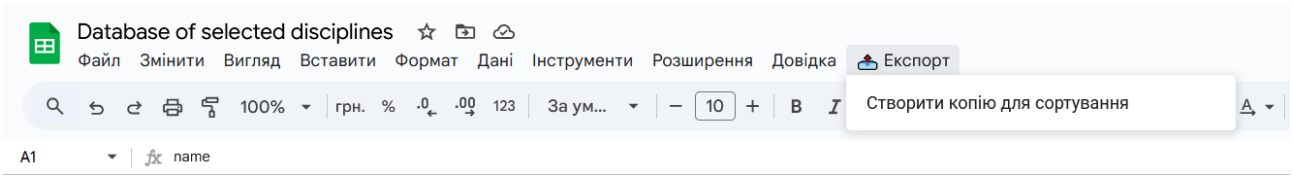


Рис. 3.1. Нова вкладка меню для експорту даних в окремий файл.

Створена копія слугує робочим середовищем для подальшого сортування та аналізу. В окремому файлі можна виконувати різноманітні операції впорядкування даних, не побоюючись порушити цілісність або формат основної (оригінальної) таблиці, що використовується для збору та зберігання інформації. Іншими словами, копія аркуша Students є ізольованим простором, де дані можна вільно реорганізовувати (сортувати, розподіляти по додаткових аркушах тощо) з метою підготовки їх до аналізу, не впливаючи на вихідні записи. Такий підхід підвищує надійність обробки: усі аналітичні перетворення здійснюються над копією, тоді як оригінальна таблиця залишається незмінною і захищеною від випадкових правок під час аналізу.

3.2 Сортування студентів за факультетами

Сортування даних про вибір дисциплін за факультетами здійснюється роздільно для різних рівнів навчання. Для цього було розроблено дві функції Google Apps Script: sortFacultyBachelor() та sortFacultyMasters() (додаток Е). Функція sortFacultyBachelor() опрацьовує записи, що стосуються студентів освітнього рівня бакалавра (включно з молодшими спеціалістами), тоді як sortFacultyMasters() відповідає за обробку даних для студентів магістерського рівня. Розподіл на дві окремі функції зумовлений необхідністю врахувати відмінності у контингенті студентів та наборі вибіркових дисциплін для бакалаврату й магістратури, а також для зручності подальшого представлення результатів (щоб не змішувати в одному звіті дані різних освітніх рівнів).

Кожна з функцій сортування за факультетами виконує багаторівневе групування даних за принципом «факультет → група → дисципліна → студенти». На першому етапі функція перебирає всі записи копії аркуша Students, відфільтровуючи їх за відповідним освітнім рівнем (бакалаврським або магістерським). Далі вибрані дані впорядковуються за факультетами: всі студенти певного факультету об’єднуються в одну категорію. Для кожного виявленого факультету створюється окремий аркуш у файлі аналізу, назва якого відповідає назві факультету та містить уточнення рівня освіти (наприклад, у назві аркуша додається позначення «(бакалаври)» або «(магістри)», щоб відрізнити таблиці для різних рівнів).

На щойно створених аркушах, присвячених окремим факультетам, дані організовано за навчальними групами. Зокрема, кожна академічна група на відповідному факультеті представлена у вигляді окремої колонки. У заголовку колонки зазначається шифр або назва навчальної групи, що дає змогу чітко ідентифікувати цю частину даних. Далі всередині колонки послідовно перераховуються вибіркові дисципліни, обрані студентами даної групи, та відповідні списки студентів. Типово це реалізовано таким чином: спочатку в колонці вказується назва вибіркової дисципліни, під нею наведено перелік студентів цієї групи, які обрали зазначену дисципліну. Потім у тій самій

колонці подається наступна дисципліна, і з нового рядка – список студентів, що її обрали, і так далі для всіх дисциплін, вибраних студентами даної групи. Таким чином, у межах одного аркуша, присвяченого факультету, можна легко побачити, які вибіркові курси обрали студенти кожної академічної групи цього факультету, і перелік цих студентів. Структура даних при такому сортуванні чітко відображає ієрархію: спершу факультет (як окремий аркуш), у ньому – групи (як окремі колонки), а в межах кожної групи – перелік вибраних дисциплін та студенти, що їх обрали (рис. 3.2).

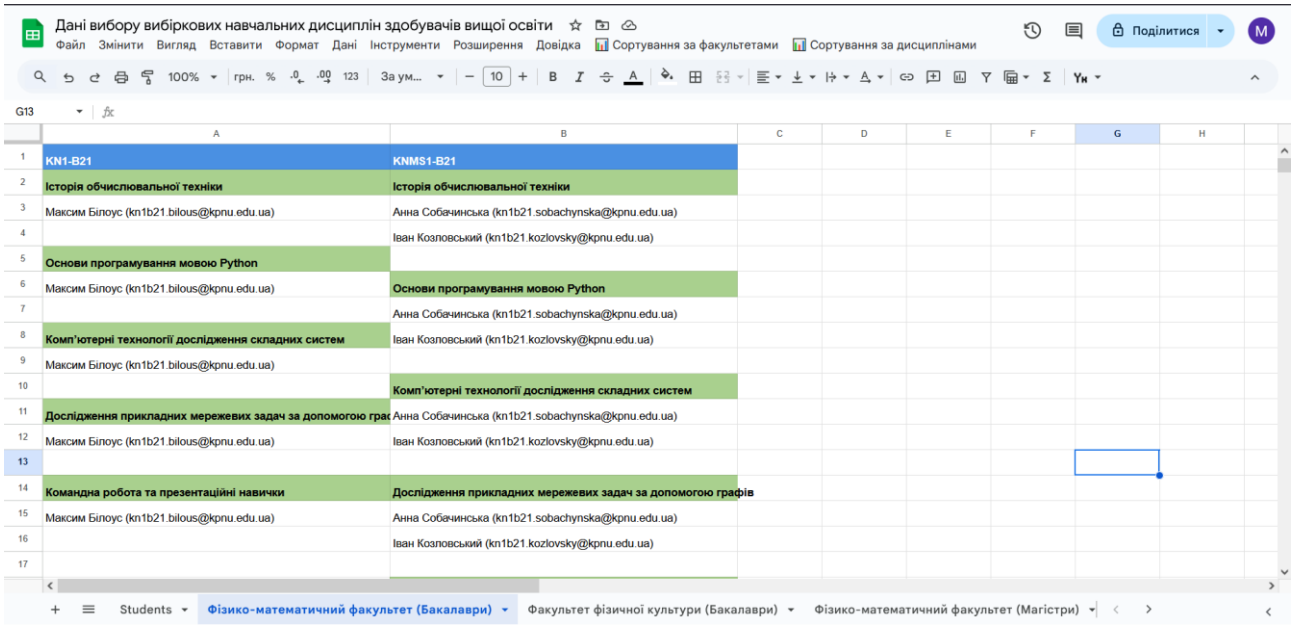


Рис. 3.2. Дані, відсортовані за факультетами.

3.3 Сортування студентів за дисциплінами

Аналогічно до попереднього випадку, для сортування записів за вибірковими навчальними дисциплінами реалізовано дві окремі функції: sortDisciplinesBachelor() та sortDisciplinesMasters(). Перша з них опрацьовує дані щодо вибору дисциплін студентами бакалаврату (та молодшими спеціалістами), а друга – для контингенту магістрів. Розділення за рівнями забезпечує коректне групування, адже перелік доступних вибіркових предметів і кількість студентів можуть різнитися між бакалаврами та магістрами.

Функції сортування за дисциплінами формують підсумкову структуру даних за принципом «дисципліна → кількість студентів → групи → студенти».

При виконанні, кожна з функцій збирає повний перелік унікальних вибіркових дисциплін, що були обрані відповідною категорією студентів, після чого створює окремий аркуш для відображення результатів сортування. У цьому аркуші кожна дисципліна представлена як окрема колонка. Назва вибіркової навчальної дисципліни зазначається у верхній частині колонки (у заголовку). Поряд із назвою (або в наступному рядку під нею) може вказуватися загальна кількість студентів, які обрали цю дисципліну – такий підрахунок відразу дає уявлення про популярність курсу серед студентів відповідного рівня.

Нижче у межах кожної колонки подано детальну інформацію, розподілену за академічними групами. Це означає, що для кожної групи, студенти якої обрали дану дисципліну, в колонці виділяється підблок: спершу зазначається назва чи шифр групи, а під ним – перелік студентів цієї групи, що вибрали відповідний курс. Далі наводиться наступна група (якщо є інші групи, де студенти обрали ту саму дисципліну) та список її студентів, і так продовжується, доки не буде перелічено всі групи, представлені серед виборців даного курсу. Таким чином, кожна колонка дисципліни містить повну інформацію про всіх студентів, які її обрали, згруповану за навчальними групами. Завдяки цьому можна відразу побачити як загальну кількість зацікавлених у певному курсі студентів, так і те, як ці студенти розподіляються між різними групами. Формат представлення у вигляді «дисципліна – кількість – групи – студенти» робить аналіз виборів прозорим: можна легко визначити найпопулярніші курси та зрозуміти, де сконцентровані студенти, що зробили той чи інший вибір (рис. 3.3).

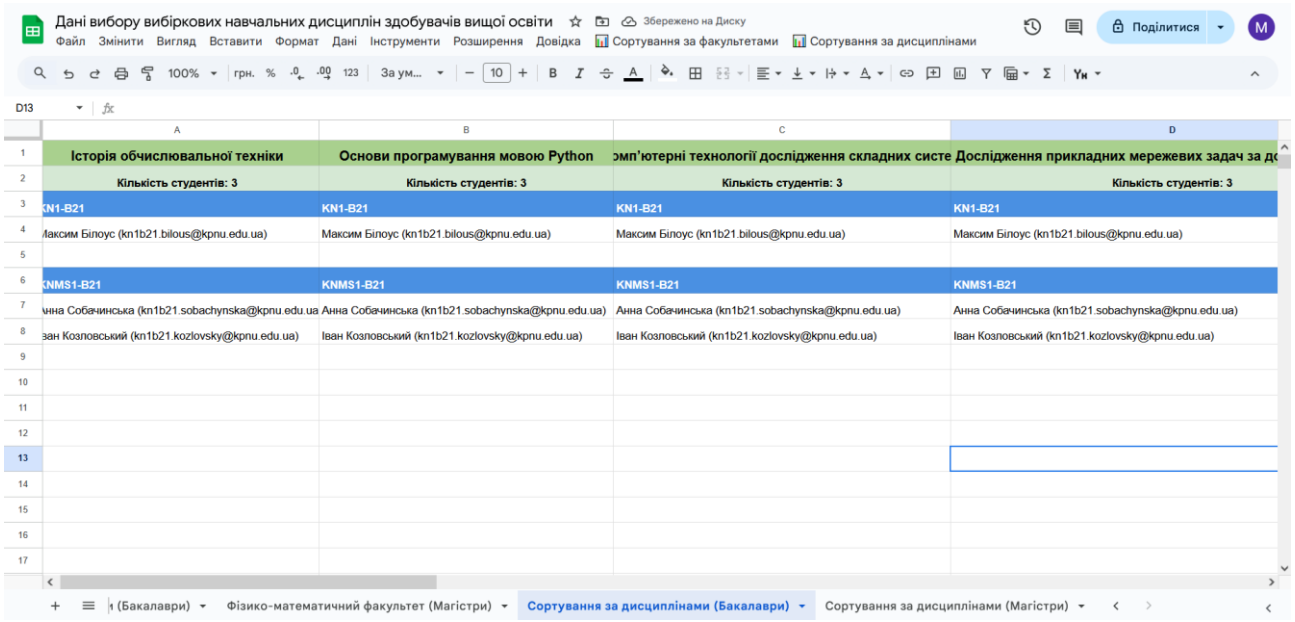


Рис. 3.3. Дані, відсортовані за дисциплінами.

3.4 Реалізація меню для запуску сортування

Для інтерактивного запуску описаних вище процедур сортування в середовищі електронної таблиці було створено спеціальне користувацьке меню. Це меню реалізовано у функції onOpen(), яка автоматично виконується при відкритті файлу Google Таблиць. Під час завантаження копії таблиці скрипт додає до інтерфейсу дві нові вкладки меню: Сортування за факультетами та Сортування за дисциплінами. Кожен із цих пунктів меню містить підменю з двох опцій, що відповідають окремим освітнім рівням. Зокрема, у меню Сортування за факультетами користувачу доступні два підпункти: Бакалаври/Молодші спеціалісти та Магістри. Аналогічно, у меню Сортування за дисциплінами передбачено вибір між сортуванням для бакалаврів (молодших спеціалістів) і для магістрів.

Вибір будь-якого з цих підпунктів негайно викликає виконання відповідної функції сортування. Наприклад, якщо користувач обирає команду «Бакалаври/Молодші спеціалісти» у меню сортування за факультетами, запускається функція sortFacultyBachelor(), яка генерує аркуші з розподілом студентів-бакалаврів за факультетами. Якщо ж обрано пункт «Магістри» у тому самому меню, виконується функція sortFacultyMasters(). Відповідно, пункти підменю у розділі сортування за дисциплінами викликають функції

sortDisciplinesBachelor() (для аналізу вибору дисциплін бакалаврів/молодших спеціалістів) та sortDisciplinesMasters() (для магістрів).

Наявність користувацького меню з відповідними командами робить систему більш зручною та зрозумілою для кінцевого користувача. Адміністратори або інші особи, що працюють із даними, можуть запускати складні алгоритми сортування одним кліком без необхідності відкривати редактор коду чи розуміти внутрішню логіку скриптів. Це підвищує ергономічність рішення та знижує імовірність помилок, оскільки всі дії виконуються через перевірені меню-скрипти (рис. 3.4).

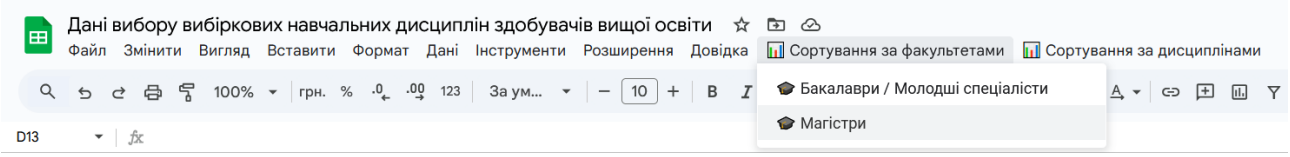


Рис. 3.4. Нові вкладки меню для запуску сортування.

Висновки до розділу 3

У цьому розділі представлено методику автоматизованого опрацювання даних про вибір вибіркових навчальних дисциплін здобувачами вищої освіти за допомогою Google Apps Script. Розроблені скрипти дозволили сформувати окрему робочу копію таблиці з даними та здійснити багаторівневе сортування цих даних за факультетами і навчальними дисциплінами. Отримані в результаті аркуші-звіти чітко відображають розподіл вибору дисциплін серед студентів різних факультетів та освітніх рівнів, що істотно спрощує подальший аналіз і виявлення закономірностей. Інтеграція функцій сортування у вигляді зручного меню в інтерфейсі Google Sheets забезпечила легкий доступ користувачів до засобів аналізу без потреби прямого втручання в код. Таким чином, використання Google Apps Script у даному випадку дало змогу автоматизувати структуризацію значного обсягу освітніх даних та підвищило ефективність їх аналізу.

ВИСНОВКИ

Вибіркові освітні компоненти є важливою частиною освітнього процесу у закладах вищої освіти. Однак традиційні методи вибору дисциплін часто є неефективними, оскільки потребують значних витрат часу та ресурсів. У даній роботі було розглянуто проблему автоматизації цього процесу та запропоновано відповідну модель інформаційної системи.

У першому розділі було проведено аналіз предметної області, охарактеризовано чинні нормативи щодо вибору вибіркових компонентів, описано досвід упровадження подібних систем в інших закладах вищої освіти, а також виявлено проблеми, які виникають при спробах автоматизації цього процесу. Особливу увагу було приділено перевагам, які надає автоматизована система — зокрема, зменшенню навантаження на адміністративний персонал, усуненню дублювання даних та підвищенню прозорості вибору дисциплін студентами.

У другому розділі представлено процес розробки клієнтської частини системи, орієнтованої на студентів. Застосунок реалізовано з використанням фреймворку Angular та інтегровано з Google Workspace. Було реалізовано функціонал авторизації через Gmail-домени університету з обмеженням доступу для сторонніх користувачів. Студенти мають змогу фільтрувати перелік дисциплін за факультетами та кафедрами, обирати бажані курси у зручному інтерфейсі та підтверджувати свій вибір, після чого заповнювати персональну форму. Усі зібрані дані передаються до Google Sheets через Web App, що забезпечує зручне централізоване зберігання. Було розроблено логіку повторного запису даних: якщо студент уже існує в таблиці, інформація оновлюється, що виключає дублікати та дозволяє повторно змінювати вибір у рамках однієї сесії.

У третьому розділі описано засоби аналізу інформації про вибір дисциплін, реалізовані у вигляді Google Apps Script-скриптів, що працюють із копією основної таблиці. Було реалізовано механізм експорту таблиці з

даними до окремого файлу, який служить робочим середовищем для сортування. Система дозволяє здійснювати сортування як за факультетами (з групуванням студентів за академічними групами та дисциплінами), так і за дисциплінами (з підрахунком кількості студентів і розподілом за групами). Розподіл даних відбувається окремо для студентів бакалаврату та магістратури. Для зручності адміністраторів у копії таблиці реалізовано користувацьке меню з пунктами, що запускають відповідні функції сортування в один клік.

Запропонована система дозволяє значно спростити процес вибору вибіркових дисциплін, зробити його більш гнучким та прозорим. Аналітична частина дипломної роботи вже була використана навчальним відділом Кам’янець-Подільського національного університету імені Івана Огієнка для підготовки звітів щодо вибору студентами вибіркових дисциплін. Тому вона може бути використана у навчальних закладах для покращення управління вибірковими компонентами та подальшого аналізу студентських уподобань.

Перспективи розвитку даної системи включають розширення її функціоналу за рахунок впровадження механізмів рекомендацій на основі попередніх виборів студентів, інтеграцію з університетськими інформаційними системами та використання методів штучного інтелекту для персоналізації вибору дисциплін.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ahmed A.A.A., Ganapathy A. Creation of automated content with embedded artificial intelligence: a study on learning management system for educational entrepreneurship. Academy of Entrepreneurship Journal, 2021. 10 с.
2. Clow M. Angular 5 Projects: Learn to Build Single Page Web Applications Using 70+ Projects. Apress, 2018. 487 с.
3. Maphosa, M., Doorsamy, W., & Paul, B. A review of recommender systems for choosing elective courses. International Journal of Advanced Computer Science and Applications, 2020. 9 с. URL: https://thesai.org/Publications/ViewPaper?Volume=11&Issue=9&Code=IJACSA&SerialNo=33 (Дата звернення: 18.12.2024)
4. Semerikov, S. O. 8th Workshop on Cloud Technologies in Education: Report. CTE WorkshopProceedings, 2021. 69 с.
5. What do students want most from their college mobile app? URL: https://www.ellucian.com/blog/college-mobile-apps-what-students-value (Дата звернення: 16.01.2025)
6. Wilken J. Angular in Action / Jeremy Wilken. – Shelter Island, New York: Manning Publications, 2018. 320 с.
7. Білоус М. Модель автоматизованої системи вибору та аналізу вибіркових освітніх компонентів. Збірник матеріалів наукової конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам'янець-Подільського національного університету імені Івана Огієнка у 2024-2025 н.р., 9-10 квітня 2025 року [Електронний ресурс]. Кам'янець-Подільський : Кам'янець-Подільський національний університет імені Івана Огієнка, фізико-математичний факультет, 2025. С. 10-13. URL: http://elar.kpnu.edu.ua/xmlui/handle/123456789/8094 (Дата звернення: 20.05.2025)

8. Вибір загальноуніверситетських дисциплін. Львівський національний університет імені Івана Франка. URL: https://lnu.edu.ua/vybir-zahalnouniversytetskykh-dystsyplin (Дата звернення: 23.11.2024)
9. Закон України про вищу освіту. URL: https://zakon.rada.gov.ua/laws/show/1556-18#Text (Дата звернення: 05.12.2024)
10. Качурівський, В.О. & Качурівська, Г.М. Моделювання інформаційно-документальної системи презентації освітніх компонент освітньої програми. Центральноукраїнський науковий вісник. Технічні науки, 2022. 9 с. URL: https://mapiea.kntu.kr.ua/pdf/6(37)_II/4.pdf (Дата звернення: 12.01.2025)
11. Сайт продукту Angular. URL: https://angular.dev/ (Дата звернення: 13.12.2024)
12. Сайт продукту Firebase. URL: https://firebase.google.com/ (Дата звернення: 09.01.2025)
13. Сайт продукту Google Sheets. URL: https://docs.google.com/spreadsheets/u/0/ https://firebase.google.com/ (Дата звернення: 29.11.2024)
14. Семигіна, Т.В. & Новак, А.С. Інформаційні системи для вибіркових дисциплін у закладах вищої освіти. Sherman Oaks, California : GS Publishing Services, 2023. 8 с. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4585293 https://firebase.google.com/ (Дата звернення: 03.12.2024)

ДОДАТКИ

Додаток А. Реалізація AuthService

```
import { Injectable } from '@angular/core';
import { Auth, signInWithPopup, GoogleAuthProvider, signOut, onAuthStateChanged }
from '@angular/fire/auth';
import { BehaviorSubject } from 'rxjs';
import { MessageService } from './message.service';

@Injectable({
 providedIn: 'root',
})
export class AuthService {
 private provider = new GoogleAuthProvider();

 private userSubject = new BehaviorSubject<{ name: string; email: string } |
null>(null);

 constructor(private auth: Auth, private messageService: MessageService) {
  onAuthStateChanged(this.auth, (user) => {
   if (user) {
    const email = user.email || '';
    const name = user.displayName || '';
    if (email.endsWith('@kpnu.edu.ua')) {
     this.userSubject.next({ name, email });
    } else {
     this.userSubject.next(null);
    }
   } else {
    this.userSubject.next(null);
   }
  });
 }

 loginWithGoogle(): Promise<void> {
  return signInWithPopup(this.auth, this.provider)
   .then((result) => {
    const email = result.user.email;
    const name = result.user.displayName;
    if (email && email.endsWith('@kpnu.edu.ua')) {
     this.messageService.add(`Ласкаво просимо, ${name}!`);
     setTimeout(() => window.location.reload(), 500);
    } else {
     throw new Error('Вхід дозволено лише студентам університету.');
    }
   })
   .catch((error) => {
    console.error('Помилка авторизації:', error);
    this.messageService.add(error.message || 'Авторизація не вдалася.');
```

```
    });
  }

  logout(): Promise<void> {
   return signOut(this.auth)
    .then(() => {
     this.messageService.add('Ви успішно вийшли із системи.');
     this.userSubject.next(null);
     setTimeout(() => window.location.reload(), 500);
    })
    .catch((error) => {
     console.error('Помилка при виході:', error);
     this.messageService.add('Помилка при виході.');
    });
  }

  getCurrentUser(): { name: string | null; email: string | null; group: string |
null } | null {
   const user = this.auth.currentUser;
   if (user) {
    const email = user.email || '';
    const name = user.displayName || '';
    const groupMatch = email.match(/([a-zA-Z]+\d+[a-zA-Z]+\d+)/);
    const group = groupMatch ? groupMatch[0] : '';
    return { name, email, group };
   }
   return null;
  }

  getCurrentUserObservable() {
   return this.userSubject.asObservable();
  }

  isLoggedIn(): boolean {
   return this.userSubject.value !== null;
  }

  getAuthToken(): Promise<string | null> {
   const user = this.auth.currentUser;
   return user ? user.getIdToken() : Promise.resolve(null);
  }
}
```

Додаток Б. Реалізація GoogleSheetsService

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, of } from 'rxjs';
import { AuthService } from '../auth.service';
import { switchMap, catchError, map } from 'rxjs/operators';

@Injectable({
 providedIn: 'root',
})
export class GoogleSheetsService {
 private CLIENT_ID = '793196214477-
mep23ffei7csq780rv45dutr3aid7lo5.apps.googleusercontent.com';
 private apiKey = 'AIzaSyD-ZSP21vfAZYaiAEomx8m5A746GykGTWA';
 private DISCOVERY_DOCS =
['https://sheets.googleapis.com/$discovery/rest?version=v4'];
 private SCOPES = 'https://www.googleapis.com/auth/spreadsheets';
 private sheetId = '1z_OI42WirG69Voopl_jQ5ISkuXkN6rLZz7dL_v-dqQA';

 private webAppUrl =
'https://script.google.com/macros/s/AKfycbyojlwqN4cbCCzT4aHFn7k182UDnroklHCKT77-
39AU17l8xozp-8Ca24zGVJgs8W6rHw/exec';

 constructor(private http: HttpClient, private authService: AuthService) {}

 getDisciplines(): Observable<any> {
  const url =
`https://sheets.googleapis.com/v4/spreadsheets/${this.sheetId}/values/Discipline?
key=${this.apiKey}`;
  return this.http.get(url);
 }

 private getStudentRow(studentEmail: string): Observable<number> {
  const url =
`https://sheets.googleapis.com/v4/spreadsheets/${this.sheetId}/values/Students!B:
B?key=${this.apiKey}`;

  return this.http.get<{ values: string[][] }>(url).pipe(
   map((data) => {
    if (!data.values) return -1;

    const emails = data.values.map((row, index) => ({
     email: row[0],
     index: index + 1
    }));

    const student = emails.find((e) => e.email === studentEmail);
    return student ? student.index : -1;
   }),
   catchError((error) => {
```

```
    console.error('Помилка отримання рядка студента:', error);
    return of(-1);
   })
  );
 }

 updateStudentDisciplines(studentEmail: string, selectedDisciplines: string[]):
Observable<any> {
  return this.getStudentRow(studentEmail).pipe(
   switchMap((rowIndex) => {
    const disciplinesString = selectedDisciplines.join('; ');

    if (rowIndex === -1) {
     return this.authService.getAuthToken().then((token) => {
      if (!token) throw new Error('Не вдалося отримати токен авторизації');

      const headers = new HttpHeaders({ Authorization: `Bearer ${token}` });
      const url =
`https://sheets.googleapis.com/v4/spreadsheets/${this.sheetId}/values/Students!B:
B:append?valueInputOption=RAW`;

      const updateData = {
       values: [[studentEmail, disciplinesString]]
      };

      return this.http.post(url, updateData, { headers });
     });
    } else {
     const updateData = { values: [[disciplinesString]] };

     return this.authService.getAuthToken().then((token) => {
      if (!token) throw new Error('Не вдалося отримати токен авторизації');

      const headers = new HttpHeaders({ Authorization: `Bearer ${token}` });
      const url =
`https://sheets.googleapis.com/v4/spreadsheets/${this.sheetId}/values/Students!D$
{rowIndex}:D${rowIndex}?valueInputOption=RAW`;

      return this.http.put(url, updateData, { headers });
     });
    }
   }),
   catchError((error) => {
    console.error('Помилка оновлення дисциплін:', error);
    throw error;
   })
  );
 }

 // Виклик Web App скрипта
 callGoogleScript(data: any): Observable<any> {
```

```
  const body = new URLSearchParams();
  for (const key in data) {
   if (data.hasOwnProperty(key)) {
    const value = Array.isArray(data[key]) ? data[key].join('; ') : data[key];
    body.set(key, value);
   }
  }

  const headers = new HttpHeaders({
   'Content-Type': 'application/x-www-form-urlencoded',
  });

  return this.http.post(this.webAppUrl, body.toString(), {
   headers,
   responseType: 'text' as 'json'
  });
 }

 // Метод для оновлення або додавання студента
 updateOrAddStudent(student: {
  name: string;
  email: string;
  group: string;
  faculty: string;
  specialty: string;
  formOfStudy: string;
  degree: string;
  selectedDisciplines: string[];
 }): Observable<any> {
  const body = new URLSearchParams();
  body.set('action', 'updateOrAddStudent');
  body.set('name', student.name);
  body.set('email', student.email);
  body.set('group', student.group);
  body.set('faculty', student.faculty);
  body.set('specialty', student.specialty);
  body.set('formOfStudy', student.formOfStudy);
  body.set('degree', student.degree);
  body.set('selectedDisciplines', student.selectedDisciplines.join('; '));

  const headers = new HttpHeaders({ 'Content-Type': 'application/x-www-form-
urlencoded' });

  return this.http.post(this.webAppUrl, body.toString(), {
   headers,
   responseType: 'text' as 'json'
  });
 }

 getFacultiesDepartments(): Observable<any> {
```

```
  const url =
`https://sheets.googleapis.com/v4/spreadsheets/${this.sheetId}/values/FacultiesDe
partments?key=${this.apiKey}`;
  return this.http.get<any>(url);
 }
}
```

Додаток В. Реалізація DisciplineComponent

```
import { Component, OnInit } from '@angular/core';
import { GoogleSheetsService } from '../../services/google-sheets.service';
import { AuthService } from '../../auth.service';
import { MessageService } from '../../message.service';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router';
import { FormsModule } from '@angular/forms';

@Component({
 selector: 'app-discipline',
 standalone: true,
 templateUrl: './discipline.component.html',
 styleUrls: ['./discipline.component.scss'],
 imports: [CommonModule, FormsModule]
})
export class DisciplineComponent implements OnInit {
 disciplines: any[] = [];
 filteredDisciplines: any[] = [];
 departments: string[] = [];
 facultyDepartmentsMap: { [faculty: string]: Set<string> } = {};
 selectedFacultyForFilter: string = '';
 selectedTag: string = '';
 showDropdown: boolean = false;
 selectedDisciplines: string[] = [];
 maxSelection: number = 5;
 isLoading: boolean = false;
 isModalOpen: boolean = false;
 isSelectionInitialized: boolean = false;
 isCustomCountModalOpen: boolean = false;
 customMaxSelection: number | null = null;
 showScrollTop: boolean = false;

 constructor(
  private googleSheetsService: GoogleSheetsService,
  private authService: AuthService,
  private messageService: MessageService,
  private router: Router
 ) {}

 ngOnInit(): void {
  this.loadDisciplines();
  window.addEventListener('scroll', this.onWindowScroll, true);
 }

 ngOnDestroy(): void {
  window.removeEventListener('scroll', this.onWindowScroll, true);
 }

 onWindowScroll = (): void => {
```

```
 this.showScrollTop = window.pageYOffset > 400;
};

scrollToTop(): void {
 window.scrollTo({ top: 0, behavior: 'smooth' });
}

loadDisciplines(): void {
 this.googleSheetsService.getDisciplines().subscribe(data => {
  const rows = data.values.slice(1);

  this.disciplines = rows.map((row: string[]) => ({
   name: row[0],
   faculty: row[1],
   department: row[2],
   syllabus: row[3]
  }));

  this.filteredDisciplines = this.disciplines;
  // Побудова мапи факультет -> кафедри
  this.facultyDepartmentsMap = {};
  this.disciplines.forEach(d => {
   if (d.faculty && d.department) {
    if (!this.facultyDepartmentsMap[d.faculty]) {
     this.facultyDepartmentsMap[d.faculty] = new Set();
    }
    this.facultyDepartmentsMap[d.faculty].add(d.department.trim());
   }
  });

  // Ініціально зібрати всі кафедри
  this.departments = Array.from(
   new Set(this.disciplines.map(d => d.department?.trim()).filter(dep => !!dep))
  );
 });
}

getFaculties(): string[] {
 return Array.from(
  new Set(this.disciplines.map(d => d.faculty).filter(f => !!f))
 );
}

filterByFaculty(faculty: string): void {
 this.selectedFacultyForFilter = faculty;
 this.selectedTag = '';

 if (!faculty) {
  // Якщо обрано "Усі факультети"
  this.filteredDisciplines = this.disciplines;
  this.departments = Array.from(
```

```
    new Set(this.disciplines.map(d => d.department?.trim()).filter(dep => !!dep))
   );
   return;
  }

  // Якщо обрано конкретний факультет
  this.departments = Array.from(this.facultyDepartmentsMap[faculty] || []);
  this.filteredDisciplines = this.disciplines.filter(d => d.faculty === faculty);
 }


 filterByTag(department: string): void {
  this.selectedTag = department;

  if (!department) {
   // Якщо обрано "Усі кафедри"
   this.filteredDisciplines = this.selectedFacultyForFilter
    ? this.disciplines.filter(d => d.faculty === this.selectedFacultyForFilter)
    : this.disciplines;
   return;
  }

  // Якщо обрана конкретна кафедра
  this.filteredDisciplines = this.disciplines.filter(d =>
   d.department?.trim() === department &&
   (!this.selectedFacultyForFilter || d.faculty ===
this.selectedFacultyForFilter)
  );
 }

 resetFilter(): void {
  this.selectedFacultyForFilter = '';
  this.selectedTag = '';
  this.departments = Array.from(
   new Set(this.disciplines.map(d => d.department?.trim()).filter(dep => !!dep))
  );
  this.filteredDisciplines = this.disciplines;
 }

 toggleDropdown(): void {
  this.showDropdown = !this.showDropdown;
 }

 toggleDiscipline(disciplineName: string): void {
  if (!this.isSelectionInitialized) {
   this.isModalOpen = true;
   return;
  }

  const index = this.selectedDisciplines.indexOf(disciplineName);
```

```
  if (index > -1) {
   this.selectedDisciplines.splice(index, 1);
  } else {
   if (this.selectedDisciplines.length < this.maxSelection) {
    this.selectedDisciplines.push(disciplineName);
   } else {
    this.messageService.add(`Максимальна кількість вибору — ${this.maxSelection}
дисциплін.`);
   }
  }
 }
 confirmSelection(): void {
  const student = this.authService.getCurrentUser();
  if (!student || !student.email || !student.name || !student.group) {
   this.messageService.add('Будь ласка, увійдіть, щоб підтвердити вибір.');
   return;
  }

  if (this.selectedDisciplines.length === 0) {
   this.messageService.add('Виберіть хоча б одну дисципліну.');
   return;
  }

  // Зберігаємо вибір у localStorage
  localStorage.setItem('selectedDisciplines',
JSON.stringify(this.selectedDisciplines));

  this.messageService.add('Ваш вибір збережено. Заповніть додаткову
інформацію.');
  this.router.navigate(['/student-form']);
 }

 startSelection(): void {
  this.isModalOpen = false;
  this.isSelectionInitialized = true;
 }

 promptChangeMax(): void {
  this.isModalOpen = false;
  this.isCustomCountModalOpen = true;
 }

 confirmCustomSelection(): void {
 if (this.customMaxSelection && this.customMaxSelection > 0 &&
this.customMaxSelection <= 10) {
  this.maxSelection = this.customMaxSelection;
  this.isCustomCountModalOpen = false;
  this.isSelectionInitialized = true;
  this.messageService.add(`Встановлено вибір: ${this.maxSelection} дисциплін.`);
 } else {
  this.messageService.add('Введіть число від 1 до 10.');
```

```
 }
}
}
```

Додаток Г. Реалізація StudentFormComponent

```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { Router } from '@angular/router';
import { GoogleSheetsService } from '../../services/google-sheets.service';
import { AuthService } from '../../auth.service';
import { MessageService } from '../../message.service';

@Component({
 selector: 'app-student-form',
 standalone: true,
 imports: [CommonModule, FormsModule],
 templateUrl: './student-form.component.html',
 styleUrls: ['./student-form.component.scss']
})
export class StudentFormComponent implements OnInit {
 faculties: { [faculty: string]: string[] } = {};
 facultyList: string[] = [];
 specialties: string[] = [];

 selectedFaculty: string = '';
 selectedSpecialty: string = '';
 formOfStudy: string = '';
 degree: string = '';
 isLoading: boolean = false;
 groupFromEmail: string = '';
 customGroup: string = '';
 useCustomGroup: boolean = false;
 groupTouched: boolean = false;
 containsInvalidChars: boolean = false;

 enableCustomGroupEdit(): void {
  this.useCustomGroup = true;
 }

 selectedDisciplines: string[] = [];

 constructor(
  private router: Router,
  private googleSheetsService: GoogleSheetsService,
  private authService: AuthService,
  private messageService: MessageService
 ) {}

 ngOnInit(): void {
  this.loadFacultiesAndSpecialties();

  // ← Завантаження дисциплін з localStorage
  const saved = localStorage.getItem('selectedDisciplines');
```

```
  if (saved) {
   try {
    this.selectedDisciplines = JSON.parse(saved);
   } catch (e) {
    console.warn('! Неможливо прочитати selectedDisciplines з localStorage');
    this.selectedDisciplines = [];
   }
  }
  this.loadFacultiesAndSpecialties();
  const student = this.authService.getCurrentUser();
  if (student?.group) {
   this.groupFromEmail = student.group;
   this.customGroup = student.group; // значення за замовчуванням
  }
 }

 loadFacultiesAndSpecialties(): void {
  this.googleSheetsService.getFacultiesDepartments().subscribe((data) => {
   const rows = data.values || [];
   rows.forEach((row: string[], index: number) => {
    if (index === 0) {
     this.facultyList = row;
     row.forEach(faculty => {
      this.faculties[faculty] = [];
     });
    } else {
     row.forEach((specialty, colIndex) => {
      if (specialty && this.facultyList[colIndex]) {
       this.faculties[this.facultyList[colIndex]].push(specialty);
      }
     });
    }
   });
  });
 }

 onFacultyChange(): void {
  this.specialties = this.faculties[this.selectedFaculty] || [];
  this.selectedSpecialty = '';
 }

 submitForm(): void {
  const student = this.authService.getCurrentUser();
  if (!student || !student.email || !student.name || !student.group) {
   this.messageService.add('Будь ласка, увійдіть у систему.');
   return;
  }

  if (!this.selectedFaculty || !this.selectedSpecialty || !this.formOfStudy ||
!this.degree) {
   this.messageService.add('Будь ласка, заповніть усі поля.');
```

```
    return;
   }

   if (this.useCustomGroup && !this.customGroup.trim()) {
    this.groupTouched = true;
    this.messageService.add('Будь ласка, введіть шифр групи.');
    return;
   }
   const invalidCharsRegex = /[^A-Za-z0-9-]/;
   if (invalidCharsRegex.test(this.customGroup)) {
    this.messageService.add('Шифр групи має містити лише латинські літери, цифри
та дефіс.');
    return;
   }
   this.isLoading = true;

   this.googleSheetsService.updateOrAddStudent({
    name: student.name,
    email: student.email,
    group: this.customGroup,
    faculty: this.selectedFaculty,
    specialty: this.selectedSpecialty,
    formOfStudy: this.formOfStudy,
    degree: this.degree,
    selectedDisciplines: this.selectedDisciplines
   }).subscribe({
    next: () => {
     // Очищаємо localStorage
     localStorage.removeItem('selectedDisciplines');
     this.isLoading = false;
     this.messageService.add('Дані успішно збережено!');
     this.router.navigate(['/success']);
    },
    error: (err) => {
     console.error('❌ Помилка при збереженні:', err);
     this.messageService.add('Помилка при збереженні даних.');
     this.isLoading = false;
    }
   });
  }
  onGroupInputChange(): void {
   const invalidCharsRegex = /[^A-Za-z0-9-]/;

   if (invalidCharsRegex.test(this.customGroup)) {
    // Показуємо попередження тільки 1 раз за фокус
    if (!this.containsInvalidChars) {
     this.messageService.add('Шифр групи має містити лише латинські літери, цифри
та дефіс.');
    }

    this.containsInvalidChars = true;
```

```
  } else {
   this.containsInvalidChars = false;
  }
}
}
```

Додаток Д. Реалізація SuccessComponent

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router';
import { AuthService } from '../../auth.service';

@Component({
 selector: 'app-success',
 standalone: true,
 imports: [CommonModule],
 templateUrl: './success.component.html',
 styleUrls: ['./success.component.scss']
})
export class SuccessComponent {
 constructor(private router: Router, private authService: AuthService) {}

 // Повернення назад на вибір дисциплін
 chooseAgain(): void {
  this.router.navigate(['/discipline']);
 }

 // Вихід із аккаунту і закриття вкладки
 logoutAndClose(): void {
  this.authService.logout().then(() => {
   window.close();
  });
 }
}
```

Додаток Е. Скрипт для сортування даних за факультетами

```
function sortFacultyBachelor() {
 const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();
 const inputSheet = spreadsheet.getSheetByName("Students");
 const data = inputSheet.getDataRange().getValues();

 const headers = data[0];
 const degreeIndex = headers.indexOf("degree");
 const facultyIndex = headers.indexOf("faculty");
 const groupIndex = headers.indexOf("group");
 const nameIndex = headers.indexOf("name");
 const emailIndex = headers.indexOf("email");
 const disciplinesIndex = headers.indexOf("selectedDisciplines");

 const grouped = {};

 for (let i = 1; i < data.length; i++) {
  const degreeRaw = (data[i][degreeIndex] || '').toString().toLowerCase().trim();

  // Для скрипта, який виводить дані бакалаврів та молодших спеціалістів
  const validDegrees = ['бакалавр', 'молодший спеціаліст'];
  if (!validDegrees.includes(degreeRaw)) continue;

  // Для скрипта, який виводить дані магістрів
  if (degreeRaw !== 'магістр') continue;

  const faculty = data[i][facultyIndex] || 'Невідомий факультет';
  const rawGroup = (data[i][groupIndex] || '').toString();
  const formattedGroup = formatGroupCode(rawGroup.toUpperCase().replace(/[^A-Z0-
9Z]/g, ''));
  const name = data[i][nameIndex] || 'Невідоме ім\'я';
  const email = data[i][emailIndex] || 'Немає пошти';
  const disciplines = (data[i][disciplinesIndex] || '').split(';').map(d =>
d.trim());

  const studentInfo = `${name} (${email})`;

  if (!grouped[faculty]) grouped[faculty] = {};
  if (!grouped[faculty][formattedGroup]) grouped[faculty][formattedGroup] = {};

  disciplines.forEach(discipline => {
   if (!grouped[faculty][formattedGroup][discipline]) {
    grouped[faculty][formattedGroup][discipline] = [];
   }
   grouped[faculty][formattedGroup][discipline].push(studentInfo);
  });
 }


 // Створення аркушів
 for (const faculty in grouped) {
  const sheetName = `${faculty} (Бакалаври)`;
```

```
  let sheet = spreadsheet.getSheetByName(sheetName);
  if (!sheet) sheet = spreadsheet.insertSheet(sheetName);
  else sheet.clear();

  let col = 1;
  for (const group in grouped[faculty]) {
   let row = 1;
   sheet.getRange(row++, col).setValue(group).setFontWeight("bold")
    .setBackground("#4A90E2").setFontColor("white");

   for (const discipline in grouped[faculty][group]) {
    sheet.getRange(row++, col).setValue(discipline).setFontWeight("bold")
     .setBackground("#A9D08E");

    grouped[faculty][group][discipline].forEach(info => {
     sheet.getRange(row++, col).setValue(info);
    });

    row++;
   }

   col++;
  }

  sheet.autoResizeColumns(1, col - 1);
  const rowCount = sheet.getMaxRows();
  for (let r = 1; r <= rowCount; r++) {
   sheet.setRowHeight(r, 30);
  }
 }
}
function formatGroupCode(group) {
 const cleaned = group.toUpperCase().replace(/[^A-Z0-9]/g, '');
 const match = cleaned.match(/^([A-Z0-9]+)([A-Z][0-9]{2})(Z?)$/);
 return match ? `${match[1]}-${match[2]}${match[3] || ''}` : group;
}
```

Додаток Ж. Скрипт для сортування даних за дисциплінами

```
function sortDisciplinesBachelor() {
 const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();
 const inputSheet = spreadsheet.getSheetByName("Students");
 const data = inputSheet.getDataRange().getValues();

 const headers = data[0];
 const degreeIndex = headers.indexOf("degree");
 const groupIndex = headers.indexOf("group");
 const nameIndex = headers.indexOf("name");
 const emailIndex = headers.indexOf("email");
 const disciplinesIndex = headers.indexOf("selectedDisciplines");

 const groupedData = {}; // { discipline: { group: [students], studentsSet: Set() } }

 for (let i = 1; i < data.length; i++) {
  const row = data[i];

  const degree = (row[degreeIndex] || '').toString().toLowerCase().trim();

  // Для скрипта, який виводить дані бакалаврів та молодших спеціалістів
  const validDegrees = ['бакалавр', 'молодший спеціаліст'];
  if (!validDegrees.includes(degree)) continue;

  // Для скрипта, який виводить дані магістрів
  if (degreeRaw !== 'магістр') continue;

  const rawGroup = (row[groupIndex] || '').toString();
  const formattedGroup = formatGroupCode(rawGroup.toUpperCase().replace(/[^A-Z0-
9Z]/g, ''));
  const name = (row[nameIndex] || '').trim();
  const email = (row[emailIndex] || '').trim();
  const disciplines = (row[disciplinesIndex] || '').split(';').map(d =>
d.trim()).filter(d => d);

  const studentInfo = `${name} (${email})`;

  disciplines.forEach(discipline => {
   if (!groupedData[discipline]) {
    groupedData[discipline] = { studentsSet: new Set(), groups: {} };
   }

   if (!groupedData[discipline].groups[formattedGroup]) {
    groupedData[discipline].groups[formattedGroup] = [];
   }

   groupedData[discipline].groups[formattedGroup].push(studentInfo);
   groupedData[discipline].studentsSet.add(email);
  });
 }

 // Створення окремого аркуша
```

```
 const outputSheetName = "Сортування за дисциплінами (Бакалаври)";
 let outputSheet = spreadsheet.getSheetByName(outputSheetName);
 if (!outputSheet) outputSheet = spreadsheet.insertSheet(outputSheetName);
 else outputSheet.clear();

 let column = 1;

 for (let discipline in groupedData) {
  let row = 1;

  // Назва дисципліни
  outputSheet.getRange(row++, column).setValue(discipline).setFontWeight("bold")
   .setFontSize(12).setBackground("#A9D08E").setHorizontalAlignment("center");

  // Кількість студентів
  const studentCount = groupedData[discipline].studentsSet.size;
  outputSheet.getRange(row++, column).setValue(`Кількість студентів:
${studentCount}`)
   .setFontWeight("bold").setBackground("#D5E8D4").setHorizontalAlignment("center");

  // Дані по групах
  for (let group in groupedData[discipline].groups) {
   outputSheet.getRange(row++, column).setValue(group)
    .setFontWeight("bold").setBackground("#4A90E2").setFontColor("white");

   groupedData[discipline].groups[group].forEach(student => {
    outputSheet.getRange(row++, column).setValue(student);
   });

   row++;
  }

  column++;
 }

 outputSheet.autoResizeColumns(1, column - 1);
 const rowCount = outputSheet.getMaxRows();
 for (let r = 1; r <= rowCount; r++) {
  outputSheet.setRowHeight(r, 30);
 }
}

function formatGroupCode(group) {
 const cleaned = group.toUpperCase().replace(/[^A-Z0-9]/g, '');
 const match = cleaned.match(/^([A-Z0-9]+)([A-Z][0-9]{2})(Z?)$/);
 return match ? `${match[1]}-${match[2]}${match[3] || ''}` : group;
}
```