Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Система автоматизації обліку з використанням засобів GAS

(тема)

Виконав:
здобувач 3 року навчання,
групи КІУКІу-22-2
Вадим КАЛІНІН
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Дмитро РОСІНСЬКИЙ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Калініну Вадиму В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система автоматизації обліку з використанням засобів GAS

затверджена наказом по університету від " 26 " травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи 1. Засоби хмарної екосистеми Google.
2. Модульна побудова програмної системи.
3. Розробка основного фунуціоналу системи.

4. Перелік питань, що потрібно опрацювати у роботі

1. Аналіз сучасних систем автоматизації обліку
2. Застосування GAS для автоматизації обліку в ЗВО
3. Розробка архітектури системи
4. Алгоритмічне забезпечення системи
5. Реалізація модулів автоматизації обліку
6. Верифікація та тестування системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз сучасних систем автоматизації обліку	10.06.25-13.06.25	
2	Застосування GAS для автоматизації обліку в	14.06.25-17.06.25	
	ЗВО		
3	Розробка алгоритмічного забезпечення	18.06.25-21.06.25	
4	Реалізація модулів автоматизації обліку	23.06.25-28.06.25	
5	Верифікація та тестування системи	30.06.25-02.07.25	
6	Оформлення матеріалів кваліфікаційної роботи	03.07.25-05.07.25	
7	Подання кваліфікаційної роботи керівникові та	07.07.25-09.07.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	10.07.25-11.07.25	
	вання		

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Дмитро РОСІНСЬКИЙ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 78 с., 10 рис., 3 табл., 1 дод., 23 джерела.

АВТОМАТИЗАЦІЯ ОБЛІКУ, ДОКУМЕНТООБІГ, ЗАКЛАД ВИЩОЇ ОСВІТИ, ІНФОРМАЦІЙНА СИСТЕМА, КОМУНІКАЦІЯ, ОБЛІКОВІ ЗАПИСИ, ТРИГЕРИ, ХМАРНІ ТЕХНОЛОГІЇ, GAS

Метою кваліфікаційної роботи є розробка системи автоматизації обліку для закладу вищої освіти із використанням Google Apps Script (GAS).

У роботі здійснено аналіз проблеми фрагментованості облікових процесів у ЗВО та обґрунтовано необхідність інтеграції цифрових інструментів на базі хмарних сервісів. Запропоновано архітектуру інформаційної системи, яка поєднує Google Forms, Sheets, Drive та інші сервіси Google Workspace. Реалізовано модулі збору, обробки, звітності та комунікації на основі GAS. Оцінено переваги та обмеження підходу, зокрема його доступність, гнучкість, адаптивність та масштабованість.

Робота підтверджує доцільність впровадження подібних систем у вітчизняних освітніх установах.

ABSTRACT

Bachelor’s thesis: 78 pages, 10 figures, 3 tables, 1 appendix, 23 sources.

ACCOUNTING AUTOMATION, ACCOUNTS, CLOUD TECHNOLOGIES, COMMUNICATION, DOCUMENT FLOW, GAS, HIGHER EDUCATION INSTITUTION, INFORMATION SYSTEM, TRIGGERS.

The aim of this qualification paper is to develop an accounting automation system for a higher education institution using Google Apps Script (GAS).

The work analyses the fragmentation problem in university record-keeping processes and justifies the integration of cloud-based tools. The proposed information system architecture integrates Google Forms, Sheets, Drive, and other Google Workspace services. GAS-based modules for data collection, processing, reporting, and communication were implemented. The advantages and limitations of the solution – including accessibility, flexibility, adaptability, and scalability – were evaluated.

The findings confirm the feasibility of implementing such systems in Ukrainian educational institutions.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЗВО – заклад вищої освіти

API – програмний інтерфейс застосунків (англ., Application Programming Interface)

CRM – система управління взаємовідносинами з клієнтами (англ., Customer Relationship Management)

CSV – формат текстових файлів із роздільниками (англ., Comma-Separated Values)

ERP – система планування ресурсів підприємства (англ., Enterprise Resource Planning)

GAS – мова сценаріїв Google Apps Script (англ., Google Apps Script)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

PDF – формат електронного документа (англ., Portable Document Format)

RBAC – керування доступом на основі ролей (англ., Role-Based Access Control)

RPA – роботизована автоматизація процесів (англ., Robotic Process Automation)

SMS – служба коротких повідомлень (англ., Short Message Service).

UI – користувацький інтерфейс (англ., User Interface)

VBA – Visual Basic for Applications – мова макросів у Microsoft Office

ВСТУП

Сучасні заклади вищої освіти стикаються з численними викликами у сфері управління навчальним процесом та обліку. Традиційні методи ведення документообігу, що базуються на паперових носіях або застарілих локальних системах, не відповідають потребам динамічного освітнього середовища XXI століття. Зростаюча кількість студентів, ускладнення навчальних програм, підвищення вимог до звітності та необхідність забезпечення прозорості освітнього процесу створюють потребу в комплексному підході до автоматизації обліку.

Основна проблема полягає в тому, що більшість українських університетів продовжують використовувати фрагментарні рішення для різних аспектів навчального процесу. Облік студентів ведеться в одній системі, управління оцінками – в іншій, комунікація зі студентами здійснюється через треті сервіси, а звітність формується вручну з різних джерел. Такий підхід призводить до дублювання даних, можливості виникнення помилок, значних витрат часу на рутинні операції та складнощів у забезпеченні актуальності інформації.

Особливо гостро ця проблема проявляється в умовах змішаного та дистанційного навчання, коли традиційні методи контролю та обліку стають неефективними. Викладачі вимушені витрачати значну частину робочого часу на адміністративні завдання замість зосередження на якості освітнього процесу. Студенти, у свою чергу, не мають оперативного доступу до інформації про свої академічні досягнення, що знижує їх мотивацію та ускладнює планування навчальної діяльності.

У сучасному цифровому середовищі автоматизація облікових процесів набуває дедалі більшого значення як для бізнесу, так і для освітніх та адміністративних установ. Облік даних, ресурсів або подій традиційно вимагав значних часових витрат, ручного введення та регулярного контролю,

що призводило до виникнення помилок, дублювання інформації та неефективного використання людських ресурсів. З розвитком хмарних технологій з’явилися нові інструменти, які дають змогу організувати облік у зручному, масштабованому та доступному форматі.

Одним із таких інструментів є Google Apps Script (GAS) – мова сценаріїв на базі JavaScript, що дозволяє автоматизувати роботу з екосистемою Google Workspace. Завдяки інтеграції з Google Sheets, Forms, Docs, Drive та іншими сервісами, GAS відкриває широкі можливості для створення повноцінних систем обліку без необхідності використання складних або платних платформ. Її застосування дозволяє реалізовувати гнучкі сценарії збору, обробки та візуалізації даних, що особливо актуально для малих організацій або внутрішніх корпоративних потреб.

Обраний підхід дозволяє продемонструвати практичну цінність інструментів GAS у задачах автоматизації, а також сформувати загальну методику побудови подібних рішень для інших сфер застосування.

1 АНАЛІЗ ПРОБЛЕМИ

1.1 Поняття та завдання автоматизації обліку

Автоматизація обліку є сучасною відповіддю на глобальну тенденцію цифрової трансформації організаційної діяльності. Вона охоплює перехід від традиційних, здебільшого паперових або частково комп’ютеризованих форм обліку до інтегрованих, інтелектуально орієнтованих інформаційних систем. У контексті закладів вищої освіти (ЗВО) автоматизація обліку означає впровадження спеціалізованих програмних засобів і сервісів, що забезпечують реєстрацію, обробку, систематизацію, зберігання й аналітичне представлення всіх видів облікової інформації – від студентських даних до фінансової документації.

Сутність цього процесу полягає у створенні технологічно єдиного середовища, в якому облік більше не є ізольованою функцією певного відділу (наприклад, бухгалтерії чи деканату), а інтегрується в усі ланки організаційної структури: академічну, адміністративну, господарську. Це дозволяє не лише спростити операційні процедури, але й забезпечити повну відтворюваність інформації, її логічну узгодженість та захист. Така система, на відміну від статичних електронних таблиць, функціонує у режимі реального часу, автоматично виконує перерахунки, надсилає повідомлення, формує звіти за заданими параметрами.

Автоматизовані облікові системи суттєво змінюють логіку управління у ЗВО. По-перше, вони усувають надлишкове дублювання інформації, типове для паперових реєстрів і ізольованих Excel-файлів. Наприклад, інформація про студента, введена один раз у загальну систему (через Google Forms або персональну картку), автоматично потрапляє до журналу відвідуваності, фінансової облікової бази (для стипендіального фонду), системи реєстрації на курси та модуля оцінювання. Це зменшує витрати часу персоналу, а також

мінімізує помилки через повторне введення.

По-друге, автоматизація забезпечує високу точність і достовірність даних. У дослідженні, проведеному в Запорізькій державній інженерній академії, впровадження системи обліку навчального навантаження на базі Excel із VBA-модулями дозволило зменшити частку помилок при формуванні графіків та розрахунку ставок викладачів із 12 % до 1,7 %, а середній час на складання розкладу було скорочено в 2,3 раза [1].

По-третє, автоматизація обліку відкриває можливості аналітики та прогнозування. Впровадження системи з інтегрованим модулем візуалізації дозволяє аналізувати динаміку відвідуваності, академічну успішність, розподіл стипендіального фонду чи навантаження по кафедрах – у вигляді графіків, діаграм, зведених таблиць. Ці дані, у свою чергу, стають основою для стратегічного планування, внутрішнього аудиту й акредитаційних процедур. Згідно з даними звіту Університету Дьюка (США), після переходу на Google Apps-інтегровану систему обліку, час, витрачений на підготовку щоквартального звіту про навчальне навантаження, зменшився із 12 годин до менш ніж 2, завдяки автоматичному агрегуванню та фільтрації даних [2].

По-четверте, автоматизовані системи сприяють прозорості та підзвітності. Адміністрація, викладачі, студенти отримують доступ до релевантної інформації відповідно до ролей. Наприклад, студент може бачити тільки власні оцінки й академічний рейтинг; викладач – відомості про групу; керівник – зведену статистику за курсом, семестром або кафедрою. Ця структурованість підвищує довіру до процесів, зменшує кількість звернень і запитів, покращує контроль за внутрішніми механізмами ЗВО.

Крім того, сучасна автоматизація обліку передбачає інтеграцію з іншими цифровими інструментами, зокрема хмарними сервісами (Google Sheets, Docs, Forms), системами управління курсами (Moodle, Canvas), CRM-платформами, бухгалтерськими програмами. Однією з найбільш універсальних і доступних технологій у цьому напрямі є Google Apps Script (GAS) – хмарна мова сценаріїв на основі JavaScript, яка дозволяє будувати

повноцінні мікросервіси в екосистемі Google Workspace без витрат на сервери або ліцензії. GAS широко застосовується для реалізації таких сценаріїв, як: автоматичне заповнення таблиць, формування PDF-звітів, регулярна відправка email-сповіщень, створення приватних інтерфейсів взаємодії з користувачем (наприклад, через вебдодатки).

Загалом, роль автоматизації обліку в сучасному ЗВО не зводиться лише до механічного спрощення операцій – вона формує нову якість управління, в якому дані перетворюються на ресурс прийняття обґрунтованих рішень, звіти стають доступними в один клік, а персонал звільняється від рутинних завдань для роботи над більш значущими стратегічними цілями закладу. Сучасні дослідження у цій сфері підкреслюють критичну важливість переходу від традиційних методів ведення обліку до автоматизованих систем управління інформаційними потоками.

Слободянюк І. І. та Розгонюк Н. В. у своєму дослідженні інтеграції облікових підсистем зазначають, що «створення єдиної інформаційної екосистеми ЗВО потребує системного підходу до автоматизації всіх облікових процесів, від студентського обліку до фінансового планування» [3]. Це твердження підкреслює необхідність розгляду автоматизації не як точкового впровадження окремих програмних рішень, а як комплексної трансформації інформаційного середовища навчального закладу.

Концептуальні основи автоматизації обліку в освітніх закладах базуються на теоретичних положеннях інформаційних систем управління та специфічних вимогах освітньої сфери. Традиційно облік у вищих навчальних закладах характеризується значною складністю через необхідність одночасного ведення декількох взаємопов'язаних облікових підсистем: студентського обліку, кадрового обліку науково-педагогічних працівників, обліку навчального навантаження, фінансового обліку та обліку матеріальних ресурсів.

Дослідження Ковтун О. І. та Бутенко Н. М. демонструють, що «автоматизація обліку навчального навантаження у ЗВО дозволяє підвищити

точність розрахунків на 85% та скоротити час на формування звітності з 40 до 6 годин на місяць» [1, с. 94]. Ці емпіричні дані свідчать про суттєвий позитивний вплив автоматизації на ефективність облікових процесів.

Теоретичний аналіз показує, що автоматизація обліку в контексті закладів вищої освіти має специфічні особливості, які відрізняють її від автоматизації в комерційних структурах. По-перше, це циклічність багатьох процесів, пов'язаних з навчальним роком та семестровою структурою. По-друге, необхідність інтеграції з державними реєстрами та системами електронного урядування. По-третє, специфічні вимоги до звітності перед органами управління освітою.

Сучасні підходи до автоматизації обліку у вищій школі базуються на принципах системного аналізу та процесного підходу. Холод М. В. у дослідженні інтелектуалізації системи нарахування стипендій підкреслює, що «ефективна автоматизація облікових процесів у ЗВО вимагає не лише технологічних рішень, але й реінжинірингу бізнес-процесів відповідно до можливостей сучасних інформаційних систем» [4, с. 84].

Фундаментальне значення автоматизації обліку полягає в трансформації ролі облікового персоналу від виконавців рутинних операцій до аналітиків та консультантів з прийняття управлінських рішень. Це особливо актуально для закладів вищої освіти, де адміністративний персонал часто поєднує облікові функції з іншими видами діяльності.

Міжнародний досвід, представлений у дослідженні Meijer M. та de Groot R., демонструє, що «впровадження рольової моделі доступу в академічних системах підвищує безпеку облікових даних та забезпечує відповідність принципам академічної доброчесності» [5, с. 107]. Це вказує на необхідність врахування специфічних етичних та правових аспектів при автоматизації обліку в освітніх закладах.

1.2 Основні принципи автоматизації обліку

Теоретичний аналіз сучасних підходів до автоматизації обліку дозволяє сформулювати систему принципів, які є фундаментальними для успішного впровадження автоматизованих рішень у закладах вищої освіти.

Принцип системності передбачає розгляд автоматизації як комплексного процесу, що охоплює всі рівні організаційної ієрархії та всі види облікової діяльності. Слободянюк І. І. та Розгонюк Н. В. обґрунтовують цей принцип наступним чином: «Інтеграція облікових підсистем в єдину інформаційну екосистему ЗВО забезпечує синергетичний ефект, коли загальна ефективність системи перевищує суму ефективностей окремих компонентів» [3, с. 45]. Практична реалізація цього принципу передбачає створення єдиної інформаційної архітектури, яка забезпечує взаємодію між різними підсистемами обліку. Наприклад, у разі оновлення інформації про студента в модулі академічної мобільності, ці зміни повинні автоматично відображатися у пов’язаних підсистемах: навчального навантаження, реєстрації дисциплін, фінансового обліку. Як показала практика Національного університету «Львівська політехніка», впровадження такої інтегрованої архітектури дозволило зменшити кількість дублювань даних на 65 % та суттєво скоротити час на внутрішнє погодження документів [3].

Іншим принципом, без якого не може існувати ефективна система обліку, є достовірність і точність даних. Облікова система повинна забезпечувати точну відповідність вхідних, проміжних і вихідних даних, а також забезпечувати логічний контроль на всіх етапах обробки. Наприклад, при нарахуванні стипендії система має автоматично порівнювати академічну успішність студента, його соціальні пільги, а також поточну квоту на стипендіальний фонд. Приклад такої реалізації було описано в дослідженні фахівців з Харківського національного економічного університету, де через Google Apps Script було реалізовано модуль динамічного порівняння даних про середній бал, податкові обмеження та порядок виплат. Це забезпечило

зниження випадків помилкових нарахувань до статистично нульового рівня [4].

Принцип адаптивності особливо важливий для освітніх закладів через динамічність нормативно-правового середовища та постійні зміни у вимогах до звітності. Цей принцип передбачає здатність системи автоматизації швидко адаптуватися до нових вимог без кардинальної реструктуризації. Дослідження Duke University показує, що «гнучкість інформаційних систем обліку є критичним фактором для забезпечення відповідності мінливим вимогам регулятивних органів» [2, с. 12]. У 2023 році Одеський національний університет імені І. І. Мечникова впровадив GAS рішення для автоматичного створення навчальних планів, які оновлюються при зміні вхідних даних у шаблоні Excel. Це дозволило уникнути щорічної ручної корекції понад 300 навчальних маршрутів та модулів [6].

Принцип безпеки інформації набуває особливого значення в контексті обліку персональних даних студентів та працівників. Сучасні підходи до забезпечення інформаційної безпеки в автоматизованих системах обліку базуються на багаторівневій системі захисту, що включає технічні, організаційні та правові заходи. Meijer M. та de Groot R. підкреслюють, що «реалізація принципу безпеки в академічних системах вимагає балансу між доступністю інформації для авторизованих користувачів та захистом від несанкціонованого доступу» [5, с. 110]. У межах проєкту цифровізації внутрішньої документації в Університеті Гронінгена (Нідерланди) впроваджено модель RBAC (Role-Based Access Control) у системі GAS, що передбачає поділ користувачів на адміністраторів, інспекторів, викладачів та студентів з обмеженим доступом до різних блоків даних [5].

Принцип економічної доцільності передбачає оптимізацію співвідношення між витратами на впровадження та експлуатацію системи автоматизації та економічним ефектом від її використання. Особливістю застосування цього принципу в освітніх закладах є необхідність врахування не лише прямих економічних ефектів, але й соціальних та освітніх наслідків

автоматизації.

Принцип масштабованості визначає здатність системи автоматизації розширюватися відповідно до зростання обсягів інформації та кількості користувачів. Для закладів вищої освіти це особливо актуально через циклічні коливання навантаження (періоди вступної кампанії, сесій, звітності) та потенційне розширення функціоналу системи.

Принцип стандартизації забезпечує сумісність різних компонентів системи автоматизації та можливість інтеграції з зовнішніми системами. Черниш О. А. у дослідженні автоматизації формування навчальних планів зазначає, що «дотримання міжнародних стандартів обміну даними є критично важливим для забезпечення інтероперабельності освітніх інформаційних систем» [6, с. 76].

Принцип зручності використання (usability) передбачає створення інтуїтивно зрозумілого інтерфейсу, який мінімізує потребу в спеціальній підготовці користувачів. Цей принцип особливо важливий для закладів освіти, де користувачами системи є не лише IT-спеціалісти, але й викладачі, адміністративний персонал та студенти з різним рівнем комп'ютерної грамотності.

Принцип прозорості та підзвітності відображає специфічні вимоги до облікових систем у сфері освіти, пов'язані з необхідністю забезпечення публічного контролю за використанням бюджетних коштів та дотриманням академічних стандартів. Реалізація цього принципу передбачає створення механізмів автоматичного логування всіх операцій та формування аудиторських слідів.

Також важливим є принцип відкритості та інтегративності, що забезпечує можливість взаємодії з іншими інформаційними платформами – Moodle, Zoom, ЄДЕБО, Google Workspace, Microsoft 365. На практиці це реалізується через API або сценарії автоматичної синхронізації даних. Наприклад, інтеграція GAS із Google Calendar у КПІ дозволила реалізувати механізм автоматичного створення календаря консультацій і екзаменів на

основі заповнених форм, що суттєво зменшило кількість конфліктів у розкладі [7].

Сукупність зазначених принципів формує теоретичну основу для проєктування та впровадження ефективних систем автоматизації обліку в закладах вищої освіти. Дотримання цих принципів забезпечує не лише технічну ефективність рішень, але й їх відповідність специфічним потребам освітньої сфери.

1.3 Аналіз сучасних систем автоматизації обліку

Сучасний ринок систем автоматизації обліку для закладів вищої освіти характеризується значною диверсифікацією рішень, які можна класифікувати за декількома критеріями:

- типом розгортання (локальні, хмарні, гібридні);
- функціональним охопленням (спеціалізовані, комплексні), походженням (зарубіжні, вітчизняні);
- моделлю ліцензування (комерційні, відкриті).

Аналіз вітчизняного ринку показує, що більшість українських закладів вищої освіти використовують власні розробки або адаптовані версії комерційних систем. Так, Національний технічний університет «Харківський політехнічний інститут» протягом 2021 року активно проводив роботу з удосконалення функціональних модулів автоматизованої системи управління навчальним процесом університету, що свідчить про тенденцію до створення індивідуальних рішень, адаптованих під специфічні потреби конкретного закладу [8].

Комерційні системи автоматизації обліку представлені як глобальними рішеннями міжнародних корпорацій, так і локальними розробками вітчизняних компаній. До першої категорії належать системи типу SAP Student Lifecycle Management, Oracle Student Information System, Microsoft Dynamics 365 for Education, які пропонують широкий функціонал та високий

рівень інтеграції, проте вимагають значних інвестицій у впровадження та супровід.

Вітчизняні рішення представлені системами типу АСУ «ВНЗ», «Universys WS», та іншими локальними розробками. Порівняльний аналіз [9] автоматизованих систем управління навчанням показує, що система Moodle, призначена для створення викладачами якісних навчальних курсів, має переваги у сфері дистанційного навчання, проте потребує доповнення для повноцінного вирішення завдань обліку.

Особливу нішу займають хмарні рішення, які набувають все більшої популярності через економічну ефективність та швидкість впровадження. Хмарна система jSolutions забезпечує створення розкладу занять, базу даних учнів, студентів, викладачів, класів, груп та широкий функціонал електронних даних, демонструючи потенціал хмарних технологій для автоматизації облікових процесів.

Сегмент відкритих систем представлений переважно освітніми платформами типу Moodle, які при відповідній адаптації можуть вирішувати завдання автоматизації обліку. Проте, як показує практика, такі системи часто потребують значних доробок для забезпечення повноцінного функціоналу обліку.

Тенденції розвитку ринку свідчать про зростання попиту на інтегровані рішення, які забезпечують автоматизацію не лише окремих облікових процесів, а всього життєвого циклу управління освітнім закладом. Це підтверджується дослідженнями, які показують, що використання систем автоматизації значно впливає на конкурентні переваги освітнього закладу на ринку освітніх послуг.

Згідно з кейсом Blue Prism, впровадження RPA (Robotic Process Automation) у фінансовому відділі Університету Калгарі дозволило скоротити ручні години на 70 %, що еквівалентно приблизно 125 000 годин на рік [10]. Хоча рішення не ґрунтується на GAS, для закладів з подібними завданнями зв’язка Google Workspace + GAS може стати доступнішим

альтернативним інструментом з низьким порогом входження.

У серії матеріалів “Google Apps Script Case Studies: Real-World Process Automation” [11] описуються кейси автоматизації злиття даних, обробки форм, створення шаблонів документів і розсилки результатів. Наприклад, сценарії автоматичної перевірки форм, оновлення записів в Google Sheets, налаштування календарів та конвертації у PDF демонструють практичну ефективність GAS без необхідності бізнес-супроводу або серверного адмінстрування.

Таблиця 1.1 – Порівняння ERP та GAS-рішень

Критерій	ERP-системи	Google Workspace + GAS
Вартість	Висока ліцензія, складне впровадження	Мінімальні витрати, без потреби в серверах
Масштаб-ність	Повністю покривають усі підсистеми ЗВО	Орієнтовані на конкретні процеси
Гнучкість	Зміни дорого обходяться та потребують часу	Легке оновлення сценаріїв скриптів
IT персонал	Потрібні IT фахівці, адміністратори	Достатньо базових навичок програмування (JavaScript)
Інтеграція	Вбудована ERP архітектура	API-зашити для інтеграції з LMS, CRM, ЄДЕБО

Далі розглянуті практичні кейси використання Google Apps Script для автоматизації обліку у ЗВО та споріднених освітніх установах.

Controlled Digital Lending у бібліотеці NYU Shanghai [12]. Під час пандемії COVID-19 бібліотека NYU Shanghai стикнулася з проблемою: багато друкованих видань стали недоступні через закриття приміщень. Для забезпечення доступу до резервних матеріалів бібліотекарі розробили рішення на основі Google Apps Script. Вони створили Google Spreadsheet із тригерами GAS, які дозволяли працівникам логувати видачу електронних

копій на обмежений період (від кількох годин до одного дня) відповідно до кількості фізичних примірників, що знаходяться в бібліотеці.

Під час видачі скрипт перевіряв «owned to loaned ratio», обмежував одночасні позички, надсилав email-сповіщення користувачам про закінчення терміну і оновлював лог доступу. Технічне рішення реалізували за декілька тижнів без залучення додаткових серверних ресурсів або дорогого ПЗ. Завдяки цьому бібліотека продовжила обслуговування користувачів навіть у кризових умовах, зберігаючи конфіденційність та контролюючи дотримання авторського права.

Система інвентаризації їдальні в CSUEB (США) [13]. Команда Стенфордського університету CSUEB реалізувала автоматизовану систему обліку запасів у їдальні (Food Pantry), використовуючи GAS та Google Sheets. Попередня система (ручні записи) приносила помилки, проблеми з консольдацією даних і відсутність інформованості про поточний стан запасів.

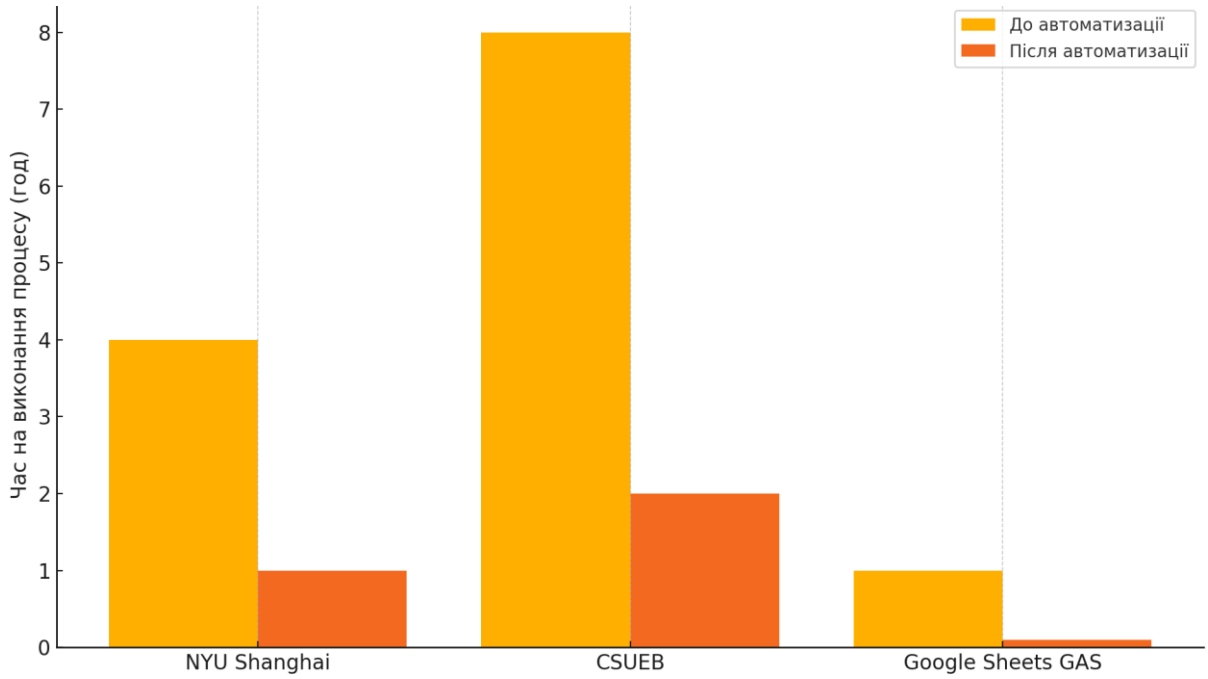

Рисунок 1.1 – Ефективність автоматизації у кейсах GAS

Сценарії GAS автоматично обробляли три операції: надходження, видачу і перегляд поточного запасу. При зміні даних створювався журнал

транзакцій, оновлювався баланс і надсилався email-сповіщення у разі низького запасу. Дані відображалися в дашборді на Looker Studio, що оновлюється кожні 15 хвилин. Результати: обробка запасів скоротилася з 6–8 годин до менше ніж 2 години на день, точність обліку підвищилася, планування стало оперативнішим. Автори оцінюють, що співробітники отримали можливість більше часу приділяти підтримці студентів, а не рутинним процесам.

Таблиця 1.2 – Порівняльний аналіз кейсів

Показник	NYU Shanghai CDL	CSUEB Food Pantry	Google Sheets Balance Script
Тип процесу	Цифрові позички	Інвентаризація харчових запасів	Облік складських залишків
Складність реалізації	SER+Timer triggers + email + quota checks	Forms, GAS dashboard, email alerts	onFormSubmit-тригер та базові команди
Результат	Безперебійний сервіс бібліотеки	Суттєве скорочення часу (2–8 годин → <2)	Автоматичний нарахунок залишку
Інфраструктура	GAS + Sheets	GAS + Sheets + Looker Studio + Forms	GAS + Sheets

Автоматизований облік залишків у Google Sheets via GAS [14]. GAS був використаний для нарахування залишків на складі на основі даних із Google Forms. При кожній новій форми (операція «Прихід» або «Видаток») скрипт запускався, діставав операцію та кількість, обчислював баланс і заносив результат у відповідну комірку таблиці.

Це рішення дало можливість уникнути ручних обчислень, швидко і точно обробляти нові дані одразу після їх введення. Сценарій легко

розширювався для складнішої логіки (RSS, обмежень, журналювання).

1.4 Переваги та недоліки сучасних рішень

Аналіз сучасних систем автоматизації обліку виявляє значну поляризацію їхніх характеристик, що створює як можливості, так і обмеження для закладів вищої освіти при виборі оптимального рішення.

Комерційні системи автоматизації демонструють високий рівень функціональної повноти та технічної надійності. Їхні основні переваги включають комплексність рішень, професійну технічну підтримку, регулярні оновлення та відповідність міжнародним стандартам. Здебільшого такі системи забезпечують інтеграцію всіх аспектів управління освітнім закладом, від прийому абітурієнтів до випуску фахівців. Проте, як свідчить практика впровадження, вони характеризуються високою вартістю ліцензій та значними витратами на впровадження, що може становити від 100 до 500 тисяч доларів США для середнього університету.

Критичним недоліком комерційних рішень є їхня орієнтація на стандартизовані бізнес-процеси, що не завжди відповідає специфіці вітчизняної освітньої системи. Необхідність адаптації таких систем під українське законодавство та освітні стандарти часто призводить до додаткових витрат та технічних ускладнень.

Вітчизняні системи автоматизації характеризуються кращою адаптованістю до особливостей національної освітньої системи та нормативно-правового середовища. Типова автоматизована система управління закладом вищої освіти (ЗВО) складається із автономних модулів: структура ЗВО, вебсайт ЗВО, навчальна частина, кафедра, навчальний розклад, абітурієнт, документообіг, деканат, кафедра, вчені ради ЗВО, що демонструє комплексний підхід до вирішення завдань автоматизації.

Переваги вітчизняних рішень включають нижчу вартість впровадження, можливість безпосереднього спілкування з розробниками

українською мовою, швидкість адаптації до змін у законодавстві. Проте такі системи часто характеризуються обмеженими можливостями масштабування, нестабільністю розробників та потенційними проблемами з довгостроковою підтримкою.

Хмарні рішення представляють собою компроміс між функціональністю та економічною ефективністю. Їхні переваги включають низькі початкові витрати на впровадження, автоматичні оновлення, масштабованість та доступність з будь-якого місця. Проте хмарні системи піднімають питання безпеки даних, залежності від інтернет-з'єднання та потенційних проблем з відповідністю вимогам законодавства щодо зберігання персональних даних.

Дослідження показують, що найбільш критичними недоліками існуючих рішень є недостатня інтеграція між різними модулями системи, складність користувацьких інтерфейсів та обмежені можливості кастомізації. Це підтверджується практичним досвідом впровадження, коли освітні заклади вимушені використовувати декілька різних систем для вирішення комплексних завдань автоматизації.

Особливо гостро проблема інтеграції проявляється при необхідності забезпечення взаємодії з державними інформаційними системами, такими як ЄДЕБО, ЄДРПОУ, системи електронного документообігу. Більшість існуючих рішень не забезпечують seamless інтеграцію з такими системами, що призводить до дублювання інформації та додаткових трудовитрат.

Аналіз користувацького досвіду виявляє, що значна частина систем автоматизації характеризується складними та неінтуїтивними інтерфейсами, що вимагає тривалого навчання персоналу та знижує ефективність використання системи. Це особливо критично для освітніх закладів, де користувачами системи є не лише IT-спеціалісти, але й викладачі, адміністративний персонал та студенти.

Технічні обмеження існуючих рішень часто включають недостатню продуктивність при обробці великих обсягів даних, обмежені можливості

створення нестандартних звітів та аналітики, а також проблеми з мобільною адаптацією інтерфейсів.

Перспективним напрямом розвитку є впровадження технологій штучного інтелекту та машинного навчання для автоматизації аналітичних процесів та прогнозування. Проте більшість існуючих систем не підтримують такі можливості, що обмежує їхній потенціал для вирішення сучасних завдань управління освітнім закладом.

Проведений аналіз дозволяє констатувати, що сучасний ринок систем автоматизації обліку для закладів вищої освіти характеризується фрагментованістю рішень та відсутністю універсального продукту, який би повністю задовольняв потреби різних типів освітніх закладів. Це створює передумови для розробки альтернативних підходів до автоматизації, зокрема, на базі гнучких хмарних платформ типу Google Apps Script.

2 ЗАСТОСУВАННЯ GOOGLE APPS SCRIPT (GAS) ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ В ЗВО

2.1 Специфіка ЗВО як предметної області

2.1.1 Документообіг та звітність

Документообіг у закладі вищої освіти включає створення, обробку, зберігання та передачу великої кількості різноманітних документів, які супроводжують навчальний процес. Ця сфера має особливу важливість для автоматизації, оскільки традиційні паперові форми документообігу є надзвичайно трудомісткими та схильними до помилок.

Первинні документи включають заявки студентів на зарахування, переведення, надання академічних відпусток, екзаменаційні відомості, протоколи засідань кафедр та факультетських рад. Ці документи формуються на найнижчому рівні управління та є основою для всіх подальших операцій системи. Автоматизація їх створення та обробки може значно скоротити час виконання рутинних операцій.

Звітні документи формуються на основі накопичених даних про навчальний процес та включають списки студентів, зведення про успішність, статистичні звіти для адміністрації та зовнішніх організацій. Частина цих звітів має стандартизований формат, визначений нормативними документами міністерства освіти, інша частина створюється для внутрішніх потреб університету. Система повинна забезпечувати автоматичне формування всіх необхідних звітів з актуальними даними.

Архівні документи включають особисті справи студентів, екзаменаційні відомості минулих років, накази про зарахування та відрахування, дипломи та додатки до дипломів. Ці документи повинні зберігатися протягом тривалого часу та бути доступними для довідок.

Система автоматизації повинна забезпечувати надійне зберігання архівних документів та швидкий пошук необхідної інформації.

Нормативні документи визначають правила організації навчального процесу, системи оцінювання, порядок проведення екзаменів та інші аспекти функціонування університету. Хоча ці документи не створюються системою автоматизації, вона повинна враховувати їх вимоги у своєму функціонуванні та забезпечувати відповідність всіх процесів чинному законодавству.

2.1.2 Особливості інформаційних потоків

Інформаційні потоки в закладі вищої освіти характеризуються складністю, багатоспрямованістю та сезонністю. Розуміння цих потоків є критично важливим для проектування ефективної системи автоматизації, яка повинна забезпечувати своєчасну обробку та передачу інформації між різними рівнями управління.

Вертикальні потоки інформації забезпечують передачу даних між різними рівнями управління університетом. Знизу вгору передається інформація про поточні результати навчання, відвідуваність студентів, виконання навчальних планів та інші операційні дані. Зверху вниз передаються управлінські рішення, зміни в навчальних планах, накази та розпорядження адміністрації. Система автоматизації повинна забезпечувати швидку та точну передачу цієї інформації.

Горизонтальні потоки включають обмін інформацією між підрозділами одного рівня, наприклад, між кафедрами в рамках одного факультету або між факультетами університету. Цей тип потоків особливо важливий для координації міжкафедрального викладання дисциплін та організації міждисциплінарних проєктів.

Сезонність інформаційних потоків визначається особливостями навчального процесу, коли пікові навантаження припадають на початок навчального року, періоди екзаменаційних сесій та терміни подачі звітності.

Система повинна бути спроектована з урахуванням цих коливань навантаження та забезпечувати стабільну роботу в періоди пікової активності.

Зовнішні інформаційні потоки включають взаємодію з міністерством освіти, іншими навчальними закладами, роботодавцями та громадськими організаціями. Ці потоки часто мають стандартизований формат та вимагають дотримання специфічних протоколів передачі даних. Система автоматизації повинна підтримувати необхідні формати експорту та імпорту даних для забезпечення ефективної взаємодії з зовнішніми системами.

2.2 Загальна характеристика GAS

Google Apps Script являє собою хмарну платформу для розробки застосунків, яка базується на мові програмування JavaScript і забезпечує глибоку інтеграцію з екосистемою Google Workspace [15]. Ця технологія була створена компанією Google з метою надання користувачам можливості розширення функціональності стандартних Google-сервісів через створення власних автоматизованих рішень.

Для закладів вищої освіти Google Apps Script виступає потужним інструментом автоматизації рутинних процесів, обробки даних та створення веб-застосунків без необхідності розгортання власної серверної інфраструктури [16]. Особливо цінним є те, що GAS дозволяє університетам створювати складні системи обліку студентів, управління навчальним процесом та автоматизації адміністративних завдань, використовуючи лише веб-браузер та базові знання JavaScript.

Платформа забезпечує безсерверну архітектуру, що означає відсутність необхідності встановлення або налаштування серверів з боку освітнього закладу. Усі скрипти виконуються на серверах Google, що гарантує високу доступність, масштабованість та надійність системи. Це особливо важливо для університетів, де відсутність власного потужного IT-відділу може стати

перешкодою для впровадження сучасних інформаційних систем.

Використання Google Apps Script у системі обліку ЗВО пояснює функціональна схема на рисунку 2.1. Вона демонструє, як дані з форм і таблиць обробляються скриптами GAS, проходять валідацію, перетворюються на звіти, надсилаються адресатам, плануються у календарі та зберігаються в архіві. У центрі архітектури показано GAS як координатора усіх облікових дій.

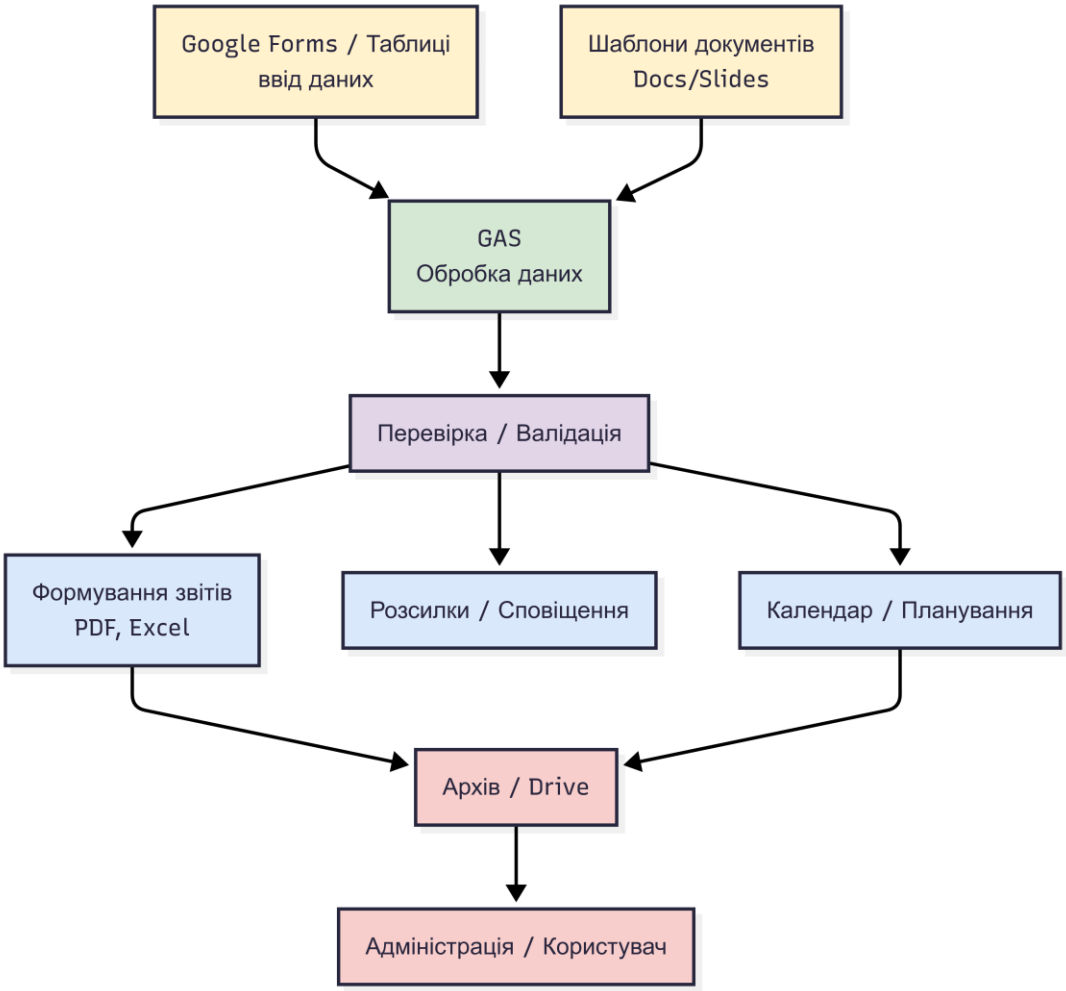

Рисунок 2.1 – Функціональна схема застосування GAS у системі обліку ЗВО

2.2.1 Архітектура та технічні особливості

Google Apps Script виконується у спеціалізованому середовищі на основі рушія JavaScript V8, який також використовується у браузері Google

Chrome [15] (рисунок 2.2). Це забезпечує високу продуктивність виконання скриптів та сумісність з сучасними стандартами ECMAScript. Для університетів це означає можливість створення складних систем обліку без значних інвестицій в апаратне забезпечення та системне програмне забезпечення.

Середовище виконання GAS має певні технічні обмеження, які важливо враховувати при проєктуванні систем для закладів вищої освіти. Максимальний час виконання одного скрипта складає шість хвилин, що достатньо для більшості завдань з обробки студентських даних, але може потребувати оптимізації для особливо складних операцій з великими обсягами інформації. Існують також обмеження на кількість API-викликів, які складають 100,000 викликів на добу, що при правильному проектуванні системи цілком достатньо для обслуговування потреб середнього університету.

GAS реалізує багаторівневу модель безпеки, яка особливо важлива для освітніх закладів, де обробляються персональні дані студентів та конфіденційна інформація про навчальний процес. Система авторизації базується на протоколі OAuth 2.0, що забезпечує контроль доступу до сервісів Google без необхідності передачі паролів або інших конфіденційних даних.

Концепція областей видимості (Scopes) дозволяє точно визначити, до яких сервісів та з якими правами має доступ кожен скрипт. Це означає, що система обліку студентів може мати доступ лише до конкретних таблиць Google Sheets та визначених папок на Google Drive, не маючи можливості доступу до особистих файлів користувачів або інших конфіденційних ресурсів.

Важливою особливістю є те, що скрипти виконуються від імені користувача, який їх запускає, та мають доступ тільки до тих ресурсів, до яких має доступ цей користувач. Це створює природний механізм контролю доступу, коли викладачі можуть працювати лише зі своїми дисциплінами, а

адміністратори - з відповідними їм розділами системи.

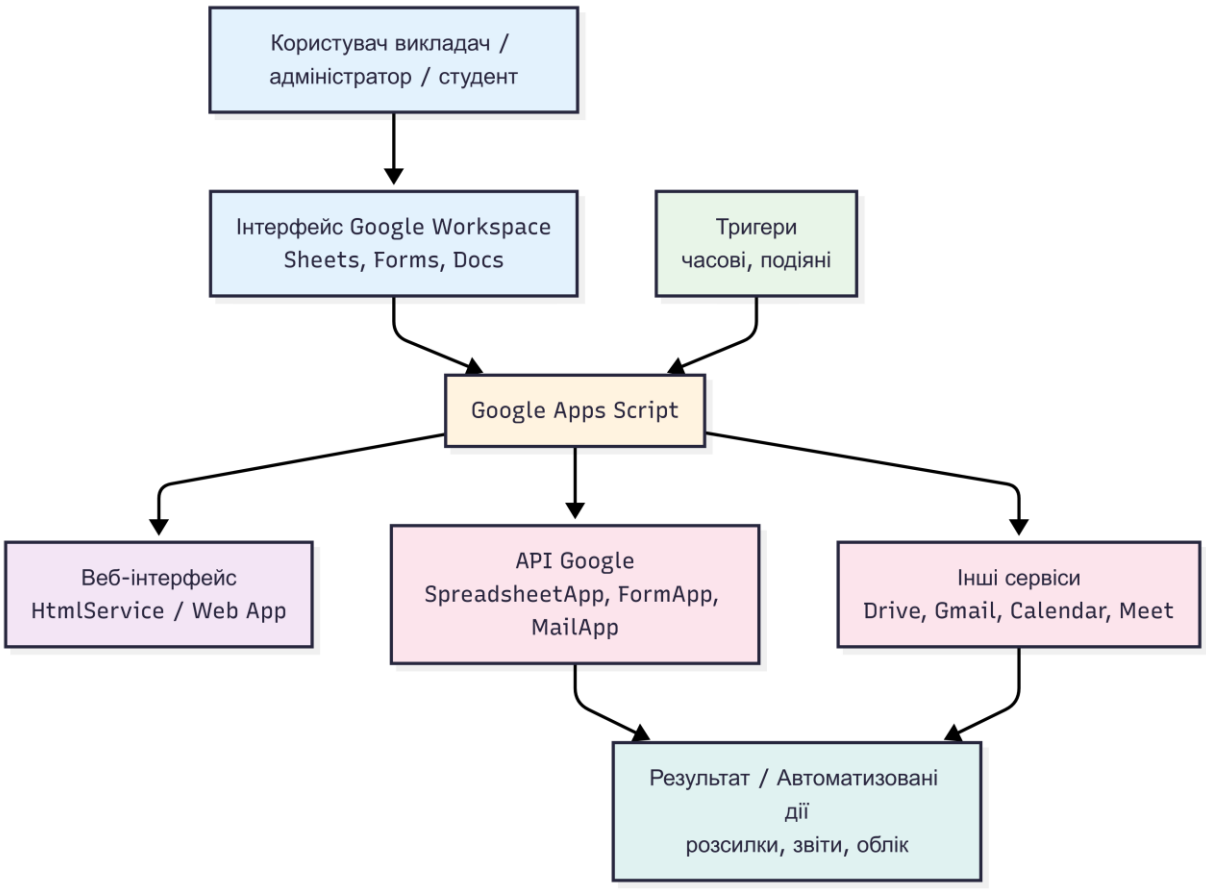

Рисунок 2.2 – Архітектура та принцип роботи Google Apps Script у ЗВО

2.2.2 Основні компоненти та сервіси

Google Sheets через GAS перетворюється з простого табличного редактора на потужну базу даних для університетських систем обліку. Ця інтеграція дозволяє створювати складні системи управління студентськими даними, автоматизації розрахунків оцінок та генерації звітів [16].

Основною перевагою використання Google Sheets як основи для системи обліку є її доступність та зрозумілість для кінцевих користувачів. Викладачі та адміністратори можуть працювати з даними у звичному табличному інтерфейсі, в той час як GAS забезпечує автоматизацію складних операцій у фоновому режимі.

Система дозволяє зберігати великі обсяги академічних даних з

автоматичним резервним копіюванням, що критично важливо для освітніх закладів. Можливість одночасної роботи декількох відділів університету з однією базою даних забезпечує узгодженість інформації та виключає дублювання даних.

Версійність та історія змін, які автоматично ведуться Google Sheets, дозволяють відстежувати всі модифікації студентських записів, що важливо для забезпечення прозорості навчального процесу та можливості аудиту системи.

Автоматизація комунікацій в університетському середовищі через Gmail стає особливо актуальною в умовах зростаючих вимог до оперативності інформування студентів та викладачів. GAS дозволяє створювати системи автоматичного сповіщення про зміни в розкладі, результати іспитів, нагадування про важливі події та персоналізовані повідомлення.

Система може автоматично відправляти повідомлення студентам про отримані оцінки, причому кожен студент отримує лише інформацію, що стосується його особисто. Це забезпечує конфіденційність даних та відповідає вимогам захисту персональної інформації.

Можливість створення шаблонів повідомлень дозволяє стандартизувати комунікацію університету зі студентами, забезпечуючи єдиний стиль та формат офіційних повідомлень. Система також може автоматично враховувати мову навчання студента та відправляти повідомлення відповідною мовою.

Управління документообігом та файлами через Google Drive стає невід'ємною частиною сучасного університетського середовища. GAS дозволяє автоматизувати створення структури папок для нових семестрів, розподіл документів між факультетами та кафедрами, а також забезпечити контрольований доступ до різних категорій документів.

Система може автоматично створювати папки для кожної навчальної групи, дисципліни або викладача, забезпечуючи правильну організацію

документообігу. Автоматичне призначення прав доступу гарантує, що студенти бачать лише ті матеріали, які їм призначені, а викладачі мають доступ до всіх необхідних їм ресурсів.

Можливість автоматичного архівування старих документів та створення резервних копій важливих файлів забезпечує збереження навчальних матеріалів та офіційних документів університету на довгий термін.

2.2.3 Тригери та автоматизація

GAS підтримує різні типи тригерів, які дозволяють автоматизувати виконання завдань без втручання користувача [15]. Прості тригери активуються при певних діях користувача, таких як редагування документа або відкриття таблиці. Це дозволяє створювати системи, які автоматично перевіряють коректність введених даних, розраховують додаткові показники або оновлюють пов'язані записи.

Встановлювані тригери працюють за часовими інтервалами або при настанні певних подій. Для університетів це означає можливість створення систем, які автоматично генерують звіти наприкінці семестру, відправляють нагадування про здачу завдань або оновлюють статистичні дані.

Особливо корисними для закладів вищої освіти є тригери форм, які активуються при надходженні нових відповідей через Google Forms. Це дозволяє автоматично обробляти заявки студентів, реєстрацію на курси або результати анкетування, негайно інтегруючи їх у загальну систему обліку.

Автоматизація рутинних процесів в університетському середовищі через GAS дозволяє значно підвищити ефективність роботи адміністративного персоналу та викладачів. Система може автоматично розраховувати середні бали студентів, визначати рейтинги, формувати списки студентів для отримання стипендій або відраховування.

Можливість автоматичного створення звітів різних форматів

забезпечує своєчасне надання інформації керівництву університету, міністерству освіти та іншим зацікавленим сторонам. Система може генерувати як стандартні звіти, так і спеціалізовані аналітичні документи відповідно до потреб конкретного закладу.

Інтеграція з календарними системами дозволяє автоматично планувати події, нагадувати про важливі дати та координувати роботу різних підрозділів університету. Це особливо важливо в умовах складного навчального процесу з множинними дисциплінами, групами та викладачами.

2.2.4 Веб-застосунки та користувацький інтерфейс

GAS дозволяє створювати повноцінні веб-застосунки з використанням HTML, CSS та JavaScript, які можуть функціонувати як самостійні системи обліку для університетів. Ці застосунки можуть мати складний користувацький інтерфейс з множинними формами, таблицями та графіками, забезпечуючи зручну роботу з даними для всіх категорій користувачів [15].

Веб-застосунки на основі GAS можуть бути розгорнуті як внутрішні системи університету або як публічні сервіси для студентів. Вони автоматично адаптуються до різних пристроїв та браузерів, забезпечуючи доступність системи з будь-якого місця та в будь-який час.

Можливість інтеграції з іншими веб-сервісами та API дозволяє створювати комплексні рішення, які можуть взаємодіяти з існуючими системами університету, зовнішніми базами даних або навіть з системами інших освітніх закладів.

Сучасні веб-застосунки на основі GAS можуть бути розроблені з урахуванням принципів адаптивного дизайну, що забезпечує комфортну роботу як на настільних комп'ютерах, так і на мобільних пристроях. Це особливо важливо для студентів, які часто використовують смартфони та планшети для доступу до навчальних ресурсів.

Система може підтримувати множинні мови інтерфейсу, що актуально

для університетів з міжнародними програмами або для регіонів з кількома офіційними мовами. Автоматичне визначення мови користувача або можливість її ручного вибору забезпечує зручність використання системи для всіх категорій користувачів.

Врахування потреб користувачів з особливими потребами через підтримку спеціальних технологій доступності робить систему більш інклюзивною та відповідає сучасним вимогам до освітніх технологій.

2.2.5 Переваги та обмеження використання GAS

Основною перевагою використання Google Apps Script для закладів вищої освіти є значне зниження витрат на створення та підтримку інформаційних систем. Відсутність необхідності в потужному серверному обладнанні, системних адміністраторах та ліцензіях на дороге програмне забезпечення робить GAS доступним навіть для невеликих університетів з обмеженим бюджетом [16].

Швидкість розробки та впровадження систем на основі GAS значно перевищує традиційні підходи до створення корпоративних додатків. Це дозволяє університетам швидко адаптуватися до змінних вимог навчального процесу та регулятивних вимог.

Автоматичне резервне копіювання та висока надійність інфраструктури Google забезпечують безпеку даних університету без додаткових зусиль з боку IT-служби. Це особливо важливо для збереження студентських записів та інших критично важливих даних.

Незважаючи на численні переваги, використання GAS має певні обмеження, які необхідно враховувати при проектуванні систем для університетів. Залежність від інтернет-з'єднання може стати проблемою в регіонах з нестабільним доступом до мережі або під час технічних проблем у Google.

Обмеження на продуктивність та кількість операцій можуть

потребувати оптимізації для великих university з десятками тисяч студентів. Це може вимагати розділення системи на кілька компонентів або використання додаткових технологій для обробки великих обсягів даних.

Питання контролю над даними та їх місцезнаходженням може бути критичним для деяких освітніх закладів, особливо тих, які підпорядковуються суворим регулятивним вимогам щодо зберігання персональних даних студентів.

2.2.6 Перспективи розвитку

Розвиток Google Apps Script тісно пов'язаний з еволюцією всієї екосистеми Google Workspace та появою нових технологій в освітній сфері [14, 15]. Інтеграція з системами штучного інтелекту та машинного навчання відкриває нові можливості для аналізу успішності студентів, прогнозування трендів в освіті та персоналізації навчального процесу.

Розширення можливостей мобільних додатків на базі GAS може привести до створення комплексних мобільних рішень для студентів, які будуть інтегровані з основними системами університету. Це включає мобільні застосунки для перегляду розкладу, отримання оцінок, комунікації з викладачами та доступу до навчальних матеріалів.

Інтеграція з IoT-пристроями може дозволити автоматизувати процеси, пов'язані з відвідуванням занять, використанням навчальних аудиторій та лабораторного обладнання, що підвищить ефективність використання університетських ресурсів.

Google Apps Script може стати основою для створення нового покоління освітніх систем, які будуть характеризуватися високою гнучкістю, швидкістю адаптації та низькими витратами на впровадження. Це особливо важливо в умовах динамічних змін в освітній сфері та необхідності швидкого реагування на нові виклики.

Розвиток хмарних технологій та підвищення їх доступності може

призвести до масового переходу освітніх закладів на рішення на основі GAS, що створить нові стандарти для університетських інформаційних систем. Це може сприяти стандартизації процесів в освіті та полегшити обмін даними між різними навчальними закладами.

Можливість створення спільних освітніх платформ, які об'єднують кілька університетів або навіть країн, може революціонізувати підходи до організації навчального процесу та створити нові форми міжнародного співробітництва в освіті.

2.3 Визначення вимог до створюваної системи

2.3.1 Цілі та завдання системи

Метою розробки системи автоматизації обліку на базі Google Apps Script є створення інтегрованого рішення, яке забезпечить ефективне управління всіма аспектами навчального процесу в закладі вищої освіти. Система повинна об'єднати функції обліку студентів, управління оцінками, комунікації, звітності та документообігу в єдиному цифровому середовищі, доступному з будь-якого пристрою та місця.

Першочерговим завданням є створення централізованої бази даних студентів, яка міститиме всю необхідну інформацію про навчальний процес. Ця база повинна забезпечувати зберігання персональних даних студентів, інформації про їх академічні досягнення, відвідуваність, участь у додаткових заходах та інші релевантні дані. Важливим аспектом є забезпечення цілісності та консистентності даних, що досягається через використання єдиних стандартів введення та валідації інформації.

Другим ключовим завданням є автоматизація процесів оцінювання та контролю успішності студентів. Система повинна забезпечувати автоматичний розрахунок поточних та підсумкових оцінок, формування рейтингових списків, визначення студентів, які претендують на отримання

стипендій або знаходяться під ризиком відрахування. Це вимагає реалізації складних алгоритмів обробки даних, які враховують специфіку різних форм контролю та системи оцінювання конкретного навчального закладу.

Третім важливим завданням є забезпечення ефективної комунікації між усіма учасниками освітнього процесу. Система повинна надавати можливості для автоматичного інформування студентів про зміни в розкладі, результати контрольних заходів, важливі оголошення та інші релевантні події. Викладачі повинні мати змогу швидко і зручно спілкуватися зі студентами, а адміністрація – ефективно координувати роботу різних підрозділів університету.

2.3.2 Функціональні вимоги

Функціональні вимоги до системи автоматизації обліку визначають конкретні можливості та операції, які повинна забезпечувати система для задоволення потреб користувачів. Ці вимоги формуються на основі аналізу поточних процесів у закладах вищої освіти та кращих практик використання інформаційних технологій в освітній сфері.

Система повинна забезпечувати повний цикл управління студентською інформацією, починаючи від реєстрації нових студентів і закінчуючи формуванням випускних документів. Модуль управління студентами має включати функції створення та редагування особистих карток студентів, відстеження їх академічного прогресу, управління переведеннями між групами та факультетами, обробку заяв на академічні відпустки та інші зміни статусу.

Підсистема управління оцінками повинна підтримувати різні форми контролю знань, включаючи поточні оцінки, результати контрольних робіт, іспитів та заліків. Система має автоматично розраховувати середні бали, накопичувальні оцінки та інші похідні показники відповідно до діючих нормативних документів. Важливою функцією є можливість налаштування

різних систем оцінювання для різних дисциплін або факультетів.

Модуль звітності повинен забезпечувати генерацію широкого спектра звітних документів, від простих списків студентів до складних аналітичних звітів про успішність навчання. Система має підтримувати як стандартні форми звітності, передбачені нормативними документами, так і довільні звіти для внутрішніх потреб університету. Усі звіти повинні генеруватися автоматично з актуальними даними та мати можливість експорту в різні формати.

Функція управління комунікацією включає можливості для масових розсилок повідомлень студентам, автоматичного сповіщення про важливі події, створення оголошень та координації роботи викладачів. Система повинна підтримувати різні канали комунікації, включаючи електронну пошту, SMS-повідомлення та веб-інтерфейс.

2.3.3 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, які не стосуються конкретних функцій, але є критично важливими для успішного функціонування системи в реальних умовах університетського середовища. Ці вимоги охоплюють аспекти продуктивності, безпеки, надійності, масштабованості та зручності використання.

Вимоги до продуктивності системи мають враховувати специфіку роботи закладу вищої освіти, де пікові навантаження можуть виникати під час реєстрації на курси, періодів екзаменаційних сесій або формування семестрових звітів. Система повинна забезпечувати прийнятний час відгуку навіть при одночасній роботі сотень користувачів. Час завантаження основних сторінок не повинен перевищувати п’яти секунд, а час виконання типових операцій, таких як пошук студента або внесення оцінки, має бути менше двох секунд.

Безпека системи є критично важливою з огляду на те, що вона оперує

персональними даними студентів та конфіденційною інформацією про навчальний процес. Система повинна забезпечувати багаторівневу автентифікацію користувачів, шифрування даних під час передачі та зберігання, аудит всіх операцій з даними та відповідність вимогам законодавства про захист персональних даних. Особливу увагу слід приділити розмежуванню доступу різних категорій користувачів до конфіденційної інформації.

Надійність системи передбачає забезпечення безперервної роботи протягом навчального року з мінімальним часом простоїв. Система повинна мати механізми автоматичного резервного копіювання даних, відновлення після збоїв та захисту від втрати інформації. Важливим аспектом є забезпечення цілісності даних при виконанні складних операцій, таких як переведення студентів або масове оновлення оцінок.

Масштабованість системи повинна забезпечувати можливість роботи з різними за розміром навчальними закладами, від невеликих інститутів з кількома сотнями студентів до великих університетів з десятками тисяч студентів. Архітектура системи має передбачати можливість нарощування функціональності та продуктивності без кардинального перепроектування.

Зручність використання є ключовим фактором успішного впровадження системи в університетському середовищі, де користувачі мають різний рівень комп'ютерної грамотності. Інтерфейс системи повинен бути інтуїтивно зрозумілим, підтримувати україномовну локалізацію, бути адаптованим для роботи на різних пристроях та мати детальну систему допомоги і навчальних матеріалів.

3 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ

3.1 Концептуальна схема системи

У процесі побудови системи автоматизованого обліку, що реалізується із використанням інструментів GAS, першочерговим етапом є проєктування концептуальної архітектури. Така схема повинна забезпечувати прозору обробку даних, розмежування доступу, підтримку гнучких форматів зберігання та надійність обміну інформацією між користувачами та сервісами.

Основними ролями в системі є:

- адміністратор системи – відповідає за ініціалізацію скриптів, моніторинг доступу, оновлення шаблонів документів і контроль роботи тригерів;

- користувачі (викладачі, студенти, працівники деканату) – взаємодіють із Google Forms для вводу даних, отримують звіти, довідки та повідомлення через Gmail або Drive;

- сервіси Google Workspace – виступають посередниками між введенням, обробкою, зберіганням і виведенням даних;

- скрипти GAS – реалізують логіку обробки інформації, що включає валідацію, логування, структурування, створення документів, надсилання повідомлень і зберігання у хмарі.

Концептуально система побудована за принципом подійно-орієнтованої архітектури (event-driven), у якій ключовим каталізатором є дії користувача або запуск тригера. Наприклад, після надсилання форми студентом, скрипт onFormSubmit обробляє введені дані, вносить інформацію до Google Таблиці, генерує індивідуальну довідку, зберігає її на Drive і надсилає студенту email із PDF-документом.

Для забезпечення масштабованості передбачається розподілення

системи на функціональні модулі:

- модуль збору даних (Forms + Triggers);
- модуль обробки (GAS logic + Sheet API);
- модуль виводу (Docs API, DriveApp, MailApp);
- модуль аудиту і контролю (логування, таблиці доступу, часові мітки).

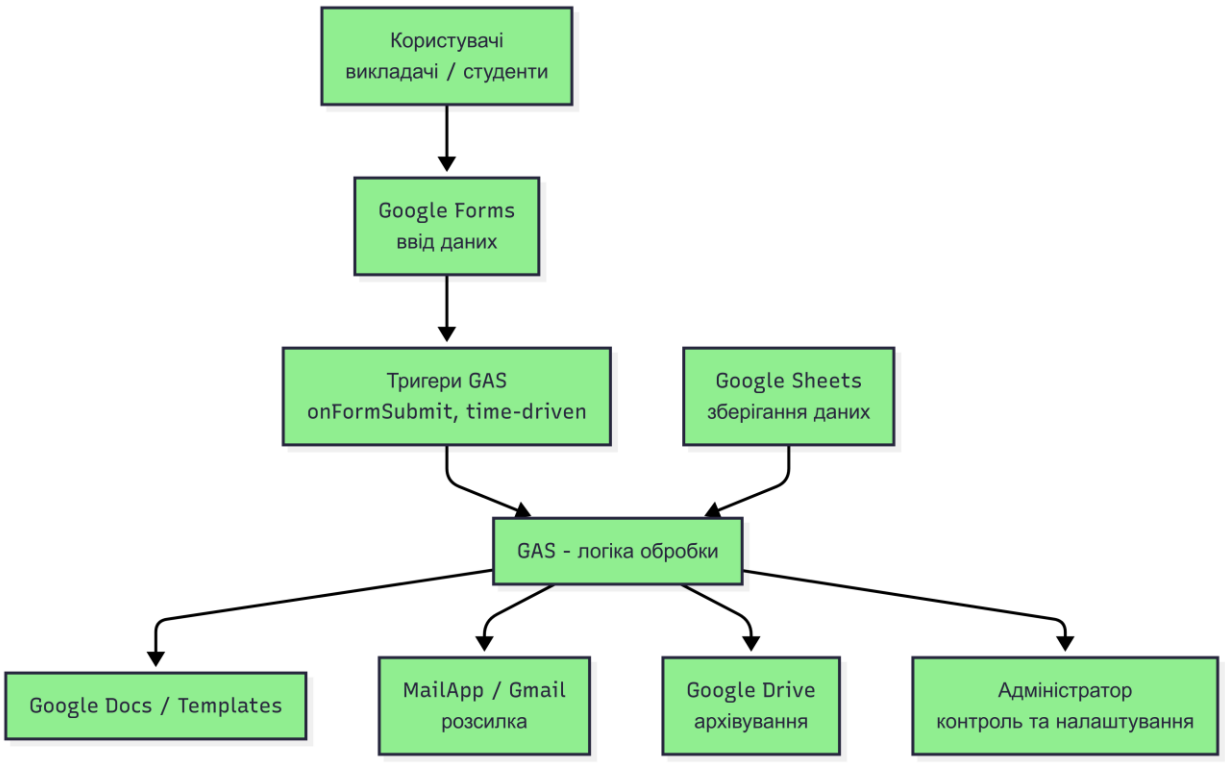

Рисунок 3.1 – Концептуальна схема архітектури системи автоматизованого обліку із застосуванням GAS

Таке структурування дозволяє гнучко адаптувати систему до потреб конкретного ЗВО – від обліку успішності до обробки розкладу чи інвентаризації.

Концептуальна схема архітектури системи автоматизованого обліку із застосуванням Google Apps Script наведена на рисунку 3.1. Вона ілюструє повний цикл від введення даних користувачами через Google Forms до обробки скриптами, збереження в таблицях, формування звітів, розсилки й архівування.

3.2 Вибір та обґрунтування програмних засобів та інструментів GAS

Розробка автоматизованої облікової системи вимагає ретельного вибору програмного середовища, здатного забезпечити:

- стабільну обробку даних;
- масштабованість;
- інтеграцію з поширеними сервісами;
- простоту використання для кінцевого користувача.

У цьому контексті Google Apps Script (GAS) і пов’язана з ним інфраструктура Google Workspace надають оптимальні можливості для освітнього закладу. На рисунку 3.2 наведена узагальнена схема, яка ілюструє зв’язки між усіма основними компонентами системи автоматизованого обліку на базі Google Apps Script. Вона показує, як дані збираються, обробляються, зберігаються, розсилаються та архівуються з урахуванням ролі адміністратора й користувачів.

Google Apps Script – хмарна платформа, яка дозволяє розширювати і автоматизувати функціонал сервісів Google, таких як Sheets, Forms, Docs, Calendar, Gmail, Drive тощо. Вона базується на JavaScript і надає доступ до спеціалізованих API (SpreadsheetApp, DocumentApp, FormApp, MailApp тощо), які дозволяють створювати складні облікові системи без потреби у власному серверному обладнанні.

Google Sheets. Використовується як основне сховище табличних даних: списків студентів, результатів, запитів тощо. Sheets підтримують роботу з макросами, імпорт-експорт CSV, та взаємодіють із GAS на рівні клітинок, діапазонів та структур.

Google Forms. Найбільш зручний механізм збору первинних облікових даних (реєстрація, заявки, збирання показників). Автоматично пов’язується з Google Таблицею, яка отримує нові рядки з кожним надходженням.

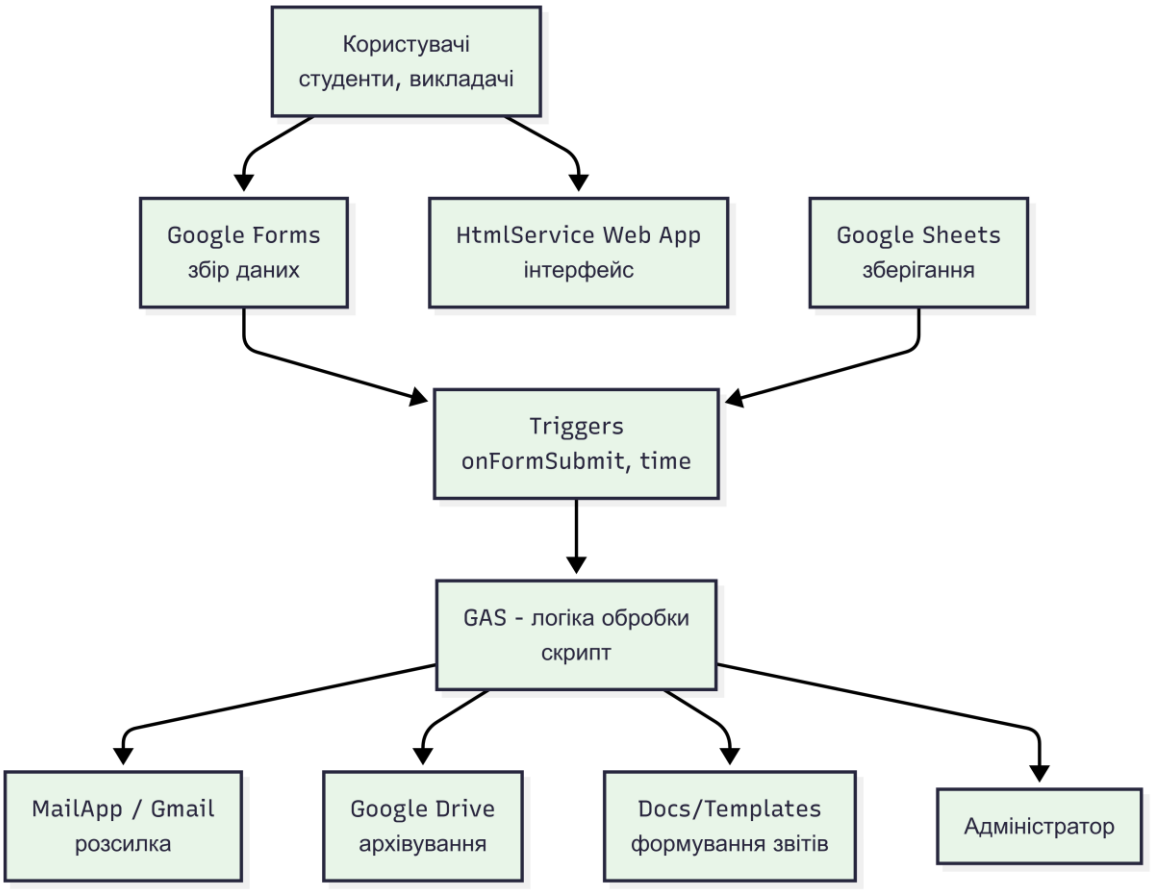

Рисунок 3.2 – Архітектура інструментів GAS у системі автоматизації обліку

Google Drive. Виступає сховищем для сформованих документів: PDF-довідок, звітів, таблиць. Через API DriveApp можна створювати нові файли, копії шаблонів, структурувати їх за папками.

Google Docs. Служить шаблонною базою для генерації персоналізованих документів, наприклад, довідок, наказів, актів. Можна автоматично заповнювати шаблони змінними через DocumentApp.

Gmail / MailApp. Використовується для надсилання результатів користувачам: повідомлень, прикріплень, розсилки результатів, підтверджень подачі форм. GAS забезпечує розсилку з вкладеннями та шаблонами.

Triggers (часові, подієві). Забезпечують повну автоматизацію без втручання користувача. Наприклад, onFormSubmit реагує на нову заявку, time-driven – на планове формування щотижневого звіту. Налаштовуються через інтерфейс Apps Script Editor.

HtmlService (за потреби). Дозволяє створювати кастомізовані

вебінтерфейси, які замінюють Forms. Наприклад, електронний кабінет користувача з формами, переглядом результатів, кнопками генерації звітів.

Таблиця 3.1 – Порівняльна характеристика

Засіб / сервіс	Призначення	Чому обрано
Google Sheets	Сховище даних	Простий, інтегрується з GAS
Google Forms	Збір даних	Доступність, адаптивність
Google Drive	Архівація звітів	Безпека, зручна організація
Google Docs	Генерація документів	Шаблонізація, підтримка PDF
GAS	Автоматизація логіки	Безсерверне виконання
Gmail / MailApp	Розсилка результатів	Вбудований канал комунікації
Тригери (Triggers)	Подійне та планове виконання	Повна автоматизація
HtmlService (опційно)	Розширення UI	Створення кастомізованих інтерфейсів

Технічна реалізація (приклад виклику даних з форми):

```
function onFormSubmit(e) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Заявки");
  var data = e.values;
  var name = data[1];
  var type = data[2];
  var date = new Date();
  sheet.appendRow([name, type, date]);
}
```

Використання Google Apps Script разом із сервісами Google Workspace дозволяє реалізувати повноцінну систему автоматизації обліку у ЗВО без додаткових витрат на ПЗ чи інфраструктуру. Інструменти забезпечують інтегровану роботу, масштабованість і адаптивність, а головне – дають змогу реагувати на реальні потреби освітнього процесу.

3.3 Модель даних системи

Одним із ключових етапів розробки будь-якої облікової системи є побудова моделі даних – логічної структури, яка відображає способи зберігання, обробки, зв’язку та валідації інформації. У випадку системи автоматизації на базі GAS модель ґрунтується на табличній основі, яку забезпечують Google Sheets, у поєднанні з формами введення на базі Google Forms [19].

На відміну від реляційних баз даних, які потребують спеціалізованого ПЗ (MySQL, PostgreSQL), у даному підході облікова база будується як структуровані аркуші, пов’язані логічними ключами, а логіка зв’язку реалізується за допомогою скриптів GAS.

Отже, проєктування моделі даних у Google Sheets з використанням GAS дає змогу створити надійну, розширювану та зручну систему обліку без необхідності у складних СУБД. Логічна структура забезпечує підтримку цілісності даних, а взаємозв’язки таблиць дозволяють гнучко формувати звіти, створювати шаблони документів та автоматизувати їх розсилку.

3.3.1 Структура даних у Google Sheets

Для реалізації обліку у ЗВО в системі проєктуються щонайменше три основні аркуші:

1) students_data – реєстр студентів:
   - student_id (унікальний ідентифікатор);
   - full_name;
   - group;
   - email;
   - status (контракт/бюджет);
   - enrollment_date;

2) requests_log – облік звернень:

- request_id;
- student_id (зовнішній ключ на students_data);
- request_type (довідка, зміна графіка тощо);
- timestamp;
- status (оброблено/очікує);

3) document_registry – журнал сформованих документів:

- doc_id;
- request_id;
- doc_type;
- file_link (посилання на файл у Google Drive);
- created_at.

Усі дані в цих таблицях взаємозв’язані за унікальними ключами (наприклад, student_id та request_id), що дає змогу використовувати GAS для формування звітів, перевірки на дублікати, об’єднання записів, побудови персоналізованих шаблонів документів.

3.3.2 Google Forms як засіб введення даних

Google Forms створює окремий аркуш логування, до якого GAS автоматично підключається через тригер onFormSubmit. Структура такого аркуша зазвичай включає:

- дата подання;
- ПІБ студента (або email);
- обраний тип довідки / звернення;
- додаткові параметри (наприклад, факультет, навчальний рік);
- підтвердження згоди.

Дані з форм автоматично дублюються в аркуш requests_log з присвоєнням request_id та подальшою обробкою скриптом.

3.3.3 Логічна схема бази даних

На основі таблиць та форм реалізується логічна модель у вигляді ER-діаграми (Entity-Relationship), де:

- Студенти пов’язані один-до-багатьох із Зверненнями;
- Звернення пов’язані один-до-одного з Документами;
- зовнішні ключі реалізуються через перевірку GAS (наприклад, перевірка відповідності student_id до введеного email);
- ідентифікатори створюються динамічно через скрипт (Utilities.getUuid() або формульне автозаповнення).

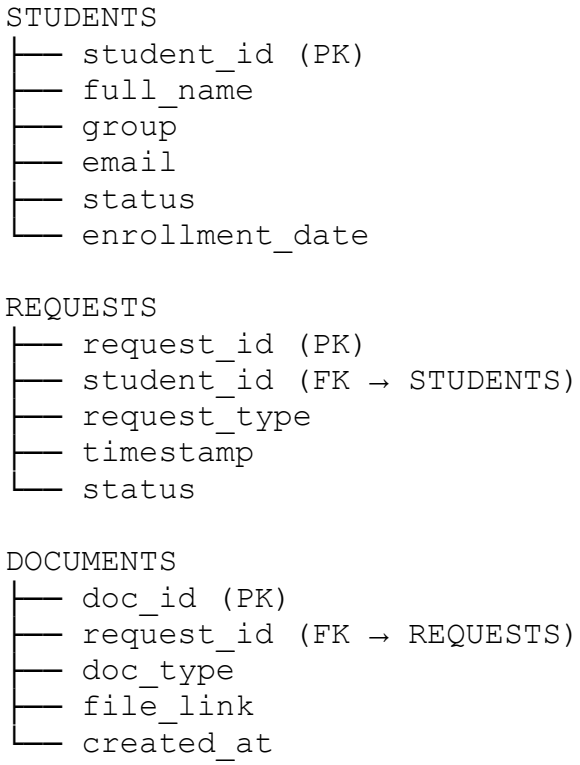

Рисунок 3.3 – Логічна структура даних

3.4 Алгоритмічне забезпечення системи

Реалізація системи автоматизованого обліку вимагає чіткого визначення процедур, що регламентують обробку вхідних даних, формування документів, зберігання результатів і взаємодію з користувачем.

Ці процедури фіксуються у вигляді алгоритмів, які реалізуються мовою JavaScript у середовищі Google Apps Script. Застосування Google Apps Script для реалізації алгоритмів системи автоматизації обліку у ЗВО дозволяє:

- забезпечити повну обробку даних без участі адміністратора;
- стандартизувати документообіг;
- зменшити час обслуговування заявок;
- знизити ризик помилок;
- реалізувати адаптивну логіку відповідно до типу звернення або статусу користувача.

Користувач (студент або співробітник) заповнює Google Form, яка записує дані у відповідний аркуш Google Sheets. У момент подання форми спрацьовує подієвий тригер onFormSubmit, який викликає скрипт для обробки інформації.

Алгоритм обробки заявки складається з таких етапів.

Етап 1. Отримати значення з форми.

Етап 2. Перевірити коректність (наявність, формат email, відсутність дубліката).

Етап 3. Присвоїти унікальний ідентифікатор request_id.

Етап 4. Записати дані у лог (таблицю requests_log).

Етап 5. Згенерувати шаблон довідки (Google Docs).

Етап 6. Зберегти файл у Drive.

Етап 7. Надіслати посилання на email заявника.

Приклад коду:

```
function onFormSubmit(e) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("requests_l
og");
  var data = e.values;
  var studentEmail = data[1];
  var requestType = data[2];
  var timestamp = new Date();
  var uuid = Utilities.getUuid();

  sheet.appendRow([uuid, studentEmail, requestType, timestamp,
```

```
"очікує"]);
  generateDocument(uuid, studentEmail, requestType);
}
```

Бізнес-логіка – це набір умов і дій, які формалізують поведінку системи відповідно до її призначення. У контексті облікової системи вона містить:

- умови: наприклад, обробляти заявку лише у робочий час або надсилати повторне повідомлення через 3 дні;
- гілки виконання: якщо студент контрактник – одна процедура, якщо бюджетник – інша;
- правила генерації файлів: шаблон довідки підбирається залежно від типу заявки.

Сценарій автоматичного формування документа поданий лістингом 3.1.

Лістинг 3.1 – Сценарій автоматичного формування документа

```
function generateDocument(uuid, email, type) {
  var templateId = selectTemplate(type);
  var copy =
DriveApp.getFileById(templateId).makeCopy("Довідка_" + uuid);
  var doc = DocumentApp.openById(copy.getId());
  var body = doc.getBody();

  body.replaceText('{{EMAIL}}', email);
  body.replaceText('{{DATE}}', new Date().toLocaleDateString());
  body.replaceText('{{TYPE}}', type);
  doc.saveAndClose();

  var url = copy.getUrl();
  MailApp.sendEmail(email, "Ваш документ готовий", "Посилання: "
+ url);
}
```

Логіка вибору шаблону:

```
function selectTemplate(type) {
  var templates = {
    "довідка про навчання": "1abc...DocID1",
    "виписка оцінок": "1def...DocID2"
  };
  return templates[type];
}
```

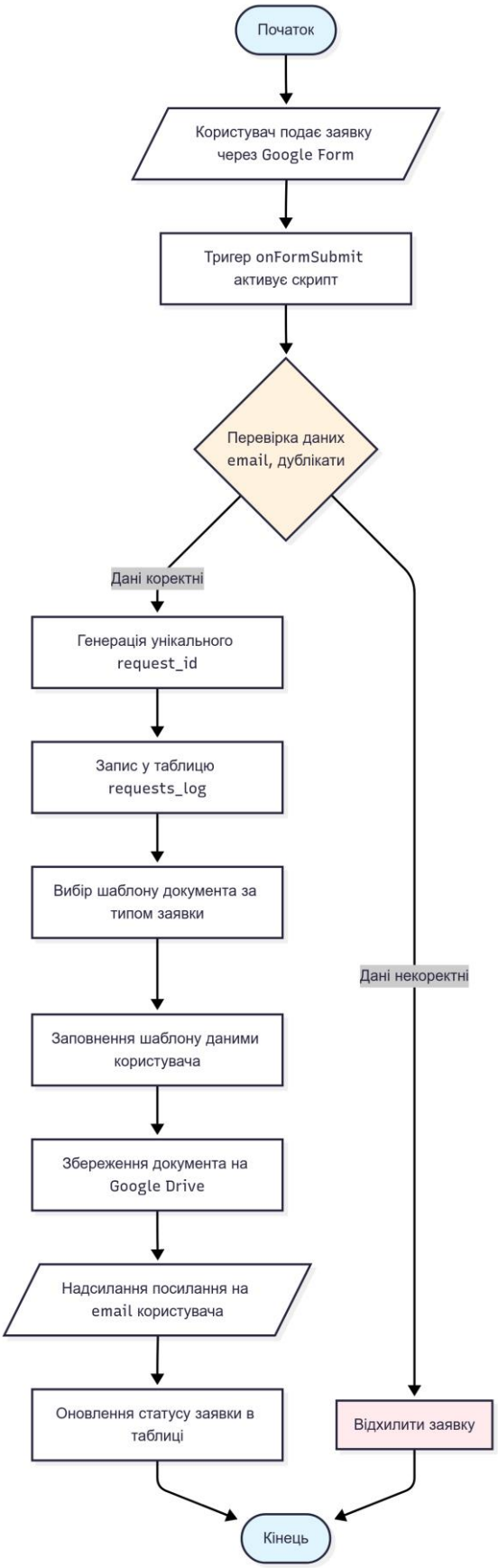

Рисунок 3.4 – Схема алгоритма обробки заявки

Після обробки заявка отримує новий статус (оброблено, відхилено), і ця інформація оновлюється в Google Sheets. Студент отримує повідомлення про результат із поясненням.

Алгоритмічне забезпечення побудовано за принципом модульності:

- onFormSubmit – обробка введення;
- generateDocument() – генерація довідки;
- sendNotification() – відправка повідомлень;
- updateStatus() – зміна статусу в журналі.

Це забезпечує гнучкість, повторне використання й полегшує супровід.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОБЛІКУ ЗАСОБАМИ GAS

4.1 Організація середовища розробки та налаштування GAS

Розробка системи автоматизованого обліку із використанням Google Apps Script передбачає попереднє налаштування середовища програмування, забезпечення доступу до сервісів Google Workspace, створення структури проєкту та ініціалізацію тригерів. Цей процес є вирішальним для забезпечення коректної роботи скриптів, інтеграції з таблицями, формами та документами, а також налагодження розгортання застосунку у вигляді вебінтерфейсу (опціонально).

Організація середовища розробки в GAS є швидкою, безкоштовною і не вимагає встановлення додаткового ПЗ. Інтеграція з Google Workspace забезпечує багатофункціональність – від збору й обробки даних до розсилки звітів і створення вебінтерфейсів. Цей підхід є оптимальним для впровадження автоматизації у ЗВО з мінімальними витратами ресурсів.

4.1.1 Створення та налаштування робочого середовища

Першим кроком є відкриття середовища Apps Script Editor, яке можна ініціювати безпосередньо з Google Таблиці або через script.google.com. При створенні нового проєкту визначається джерело, до якого буде прив’язано скрипт (наприклад, Google Sheets), що буде базою даних у системі [20].

Ключові дії під час створення:

1) вибір типу проєкту: standalone (окремий) або container-bound (вбудований у документ/таблицю);

2) задання імені файлу Code.gs або створення кількох модулів;

3) додавання сервісів через “Resources → Services” (наприклад, Google Drive API, Gmail API, якщо потрібна розширена функціональність);

4) увімкнення App Script API у Google Cloud Console для зовнішніх викликів.

GAS забезпечує інтерактивну розробку з можливістю автоматичного збереження, відлагодження (Logger.log()), доступу до журналу виконання (Executions), запуску функцій вручну, а також створення таймерів (time-driven triggers) через інтерфейс.

4.1.2 Інтеграція GAS із додатками Google Workspace

Google Apps Script має глибоку інтеграцію з додатками Google Workspace, кожен з яких представлений окремим інтерфейсом прикладного програмування (API):

- SpreadsheetApp – для взаємодії з таблицями: зчитування та запис у клітинки, маніпуляції з аркушами, обчислення, сортування, валідація тощо;
- FormApp – для створення і редагування Google Forms, динамічної зміни питань, обробки відповідей;
- DocumentApp – для автоматичного формування документів Google Docs, включно з шаблонізацією;
- MailApp – для надсилання email-повідомлень, у тому числі з вкладеннями;
- DriveApp – для збереження, пошуку і організації файлів на Google Drive;
- CalendarApp, HtmlService, Session тощо – для розширення функціоналу за потреби.

Приклад створення файлу на основі шаблону:

```
function generateDocFromTemplate(studentName, type) {
  var templateId = '1a2b3c4DocID'; // шаблон Google Docs
  var copy =
DriveApp.getFileById(templateId).makeCopy('Довідка_' +
studentName);
  var doc = DocumentApp.openById(copy.getId());
  var body = doc.getBody();
```

```
  body.replaceText('{{NAME}}', studentName);
  body.replaceText('{{TYPE}}', type);
  doc.saveAndClose();
}
```

Приклад запису результату до Google Sheets:

```
function logSubmission(studentEmail, type) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("requests_l
og");
  sheet.appendRow([new Date(), studentEmail, type, "очікує"]);
}
```

Інтеграція базується на авторизованому доступі до сервісів: скрипт виконується від імені користувача, який має відповідні права на об'єкти (таблиці, шаблони, диски).

4.1.3 Безпека та доступ

Коли скрипт виконується вперше, Google запитує дозволи на доступ до сервісів. Рівень доступу залежить від викликаних API: читання/запис до Google Drive, надсилання email тощо. Надійність середовища гарантується вбудованою інфраструктурою Google та ізоляцією скриптів у межах домену (якщо використовується акаунт edu-закладу).

4.2 Реалізація модулів автоматизації обліку

Автоматизація облікових процесів у ЗВО реалізується через модульну структуру скриптів, кожен з яких відповідає за окрему ділянку: збір і зберігання даних, формування документів, розсилку результатів, побудову зведених таблиць і аналітики. Модульність дозволяє масштабувати систему, спрощувати супровід і змінювати логіку без порушення загальної роботи [21]. Отже, система модулів у GAS дозволяє забезпечити повну

автоматизацію процесів у межах цифрової екосистеми Google Workspace: від збору й збереження інформації до формування документів і побудови аналітичних візуалізацій [18]. Це не лише скорочує час на обробку запитів, а й підвищує прозорість та надійність облікових операцій.

4.2.1 Автоматизація збору та зберігання даних

Всі користувацькі дані надходять у систему через Google Forms. Заповнені форми автоматично надсилають дані до відповідної Google Таблиці (requests_log), після чого активується обробка скриптом GAS. Основні етапи перелічені нижче.

Етап 1. Тригер onFormSubmit обробляє новий запис.

Етап 2. Скрипт присвоює унікальний request_id і записує заявку.

Етап 3. У таблицю додаються додаткові поля (наприклад, status, doc_link).

Етап 4. За потреби ініціюється перевірка на дублікати або фільтрація.

Фрагмент коду додавання заявки до журналу:

```
function onFormSubmit(e) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("requests_l
og");
  var values = e.values;
  var uuid = Utilities.getUuid();
  sheet.appendRow([uuid, values[1], values[2], new Date(),
"очікує"]);
}
```

4.2.2 Автоматичне формування звітів та аналітики

Формування звітів – це ключова функція в обліку: система повинна не лише реєструвати дані, а й оперативно створювати персоналізовані документи (довідки, підтвердження, накази) і аналітичні зведення.

Приклад генерації довідки на основі шаблону:

```
function generateCertificate(requestId, name, type) {
  var templateId = '1a2b3c...'; // ID шаблону
  var copy =
DriveApp.getFileById(templateId).makeCopy("Довідка_" + name);
  var doc = DocumentApp.openById(copy.getId());
  var body = doc.getBody();
  body.replaceText('{{NAME}}', name);
  body.replaceText('{{TYPE}}', type);
  doc.saveAndClose();
  return copy.getUrl();
}
```

Після створення документа його посилання записується до requests_log, а користувач отримує email з PDF-додатком.

4.2.3 Формування аналітики

Google Sheets дозволяє будувати зведені таблиці (Pivot Table) на основі зібраних даних. GAS може автоматично формувати такі таблиці або оновлювати діаграми для динамічної візуалізації:

```
function updateChartData() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("аналітика"
);
  var dataRange = sheet.getRange("A1:C100");
  var chart = sheet.newChart()
    .setChartType(Charts.ChartType.COLUMN)
    .addRange(dataRange)
    .setPosition(5, 5, 0, 0)
    .build();
  sheet.insertChart(chart);
}
```

4.2.4 Архітектура модулів

Модулі реалізуються у вигляді окремих функцій або класів (через JavaScript-об'єкти), наприклад:

- dataLogger – модуль запису у requests_log;

- reportGenerator – створення Google Docs із шаблонів;
- mailDispatcher – надсилання повідомлень користувачам;
- analyticsEngine – побудова зведених таблиць, діаграм;
- statusUpdater – оновлення поля status в таблиці.

Це дозволяє гнучко налаштовувати процеси без дублювання коду. схема модулів автоматизації обліку. На рисунку 4.1 проілюстровано взаємозв'язки між компонентами системи GAS і показано, як дані проходять крізь модулі – від збору з Google Forms, через обробку, зберігання, формування звітів, аналітику, до розсилки результатів.

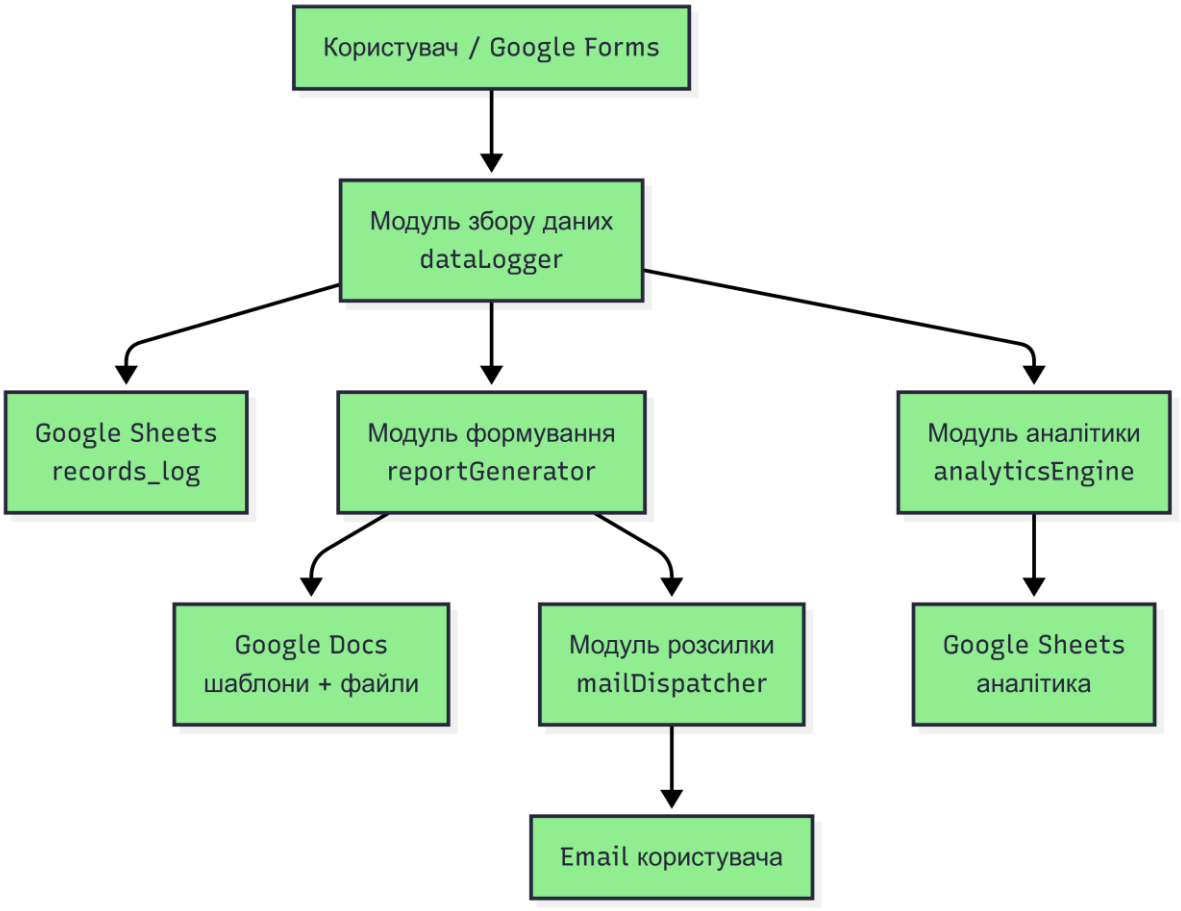

Рисунок 4.1 – Функціональна схема модулів автоматизації обліку в GAS

4.3 Реалізація користувацького інтерфейсу системи

Інтерфейс користувача відіграє ключову роль у сприйнятті та зручності

взаємодії з системою [22]. У межах проєкту реалізовано два типи (рівні) інтерфейсів:

- базовий функціональний UI, що реалізується за допомогою Google Forms і Google Sheets;
- розширений вебінтерфейс, побудований з використанням HtmlService у Google Apps Script.

Обидва підходи інтегруються із загальними модулями логіки та забезпечують доступ як для кінцевих користувачів (студентів, викладачів), так і для адміністратора системи. Отже, це дозволяє задовольнити потреби як кінцевих користувачів (простота), так і адміністраторів (гнучкість і керованість). Структура інтерфейсів адаптована до організаційної моделі ЗВО.

4.3.1 Розробка інтерфейсів у Google Forms та Sheets

Google Forms – найпростіший спосіб взаємодії користувача із системою: вона дозволяє заповнювати заявки, формувати звернення, ініціювати запити на довідки. Форма зв'язана із Google-таблицею, що виступає точкою входу для обробки даних.

Приклад полів Google Form:

- ПІБ студента;
- email;
- тип довідки (вибір зі списку);
- коментар (опціонально);
- підтвердження згоди на обробку даних.

Після відправлення, відповіді потрапляють до таблиці requests_log, де активується скрипт onFormSubmit.

Google Sheets використовується як адміністративний інтерфейс. Адміністратор бачить:

- список усіх заявок;

- статуси обробки;
- посилання на сформовані документи;
- можливість вручну редагувати поля або додати коментар.

Додатково в Google Sheets реалізуються візуальні індикатори, наприклад умовне форматування для швидкого розпізнавання необроблених заявок.

4.3.2 Розробка інтерфейсів на базі GAS Web Apps

Для більш гнучкого й естетичного представлення було реалізовано інтерфейс на базі Google Apps Script Web App, створений через HtmlService.

Основні можливості:

- вхід за email (використання Session.getActiveUser().getEmail());
- динамічна форма з вибором типу звернення;
- перегляд історії звернень;
- кнопки “Надіслати”, “Скасувати”, “Переглянути файл”;
- повідомлення про статус заявки (в обробці, завершено, відхилено);
- сторінка адміністратора (фільтрація заявок, вручну змінити статус, додати документ).

Приклад структури наведений нижче.

Файл Code.gs:

```
function doGet() {
  return HtmlService.createHtmlOutputFromFile('index');
}

function submitRequest(data) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("requests_l
og");
  sheet.appendRow([Utilities.getUuid(), data.name, data.type,
new Date(), "очікує"]);
  return "Заявка прийнята";
}
```

Файл index.html пояснюється лістингом 4.1.

Лістинг 4.1 – Файл index.html

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Подання заявки</h2>
    <form id="form">
      ПІБ: <input type="text" name="name"><br>
      Тип довідки:
      <select name="type">
        <option value="навчання">Довідка про навчання</option>
        <option value="оцінки">Виписка оцінок</option>
      </select><br>
      <input type="submit" value="Надіслати">
    </form>
    <script>
      const form = document.getElementById("form");
      form.onsubmit = async (e) => {
        e.preventDefault();
        const data = Object.fromEntries(new
FormData(form).entries());
        const res = await
google.script.run.withSuccessHandler(alert).submitRequest(data);
      };
    </script>
  </body>
</html>
```

Переваги Web App:

- динамічність (можна додати таби, діаграми, діалоги);
- підтримка CSS і JavaScript;
- можливість реалізувати адмінпанель;
- інтеграція в корпоративний домен Google.

На рисунку 4.2 наведено схему інтерфейсної взаємодії. Схема демонструє, як користувачі (студенти й адміністратори) взаємодіють із формами, вебінтерфейсами, таблицями й модулями GAS, а також як відбувається створення документів і надсилання результатів.

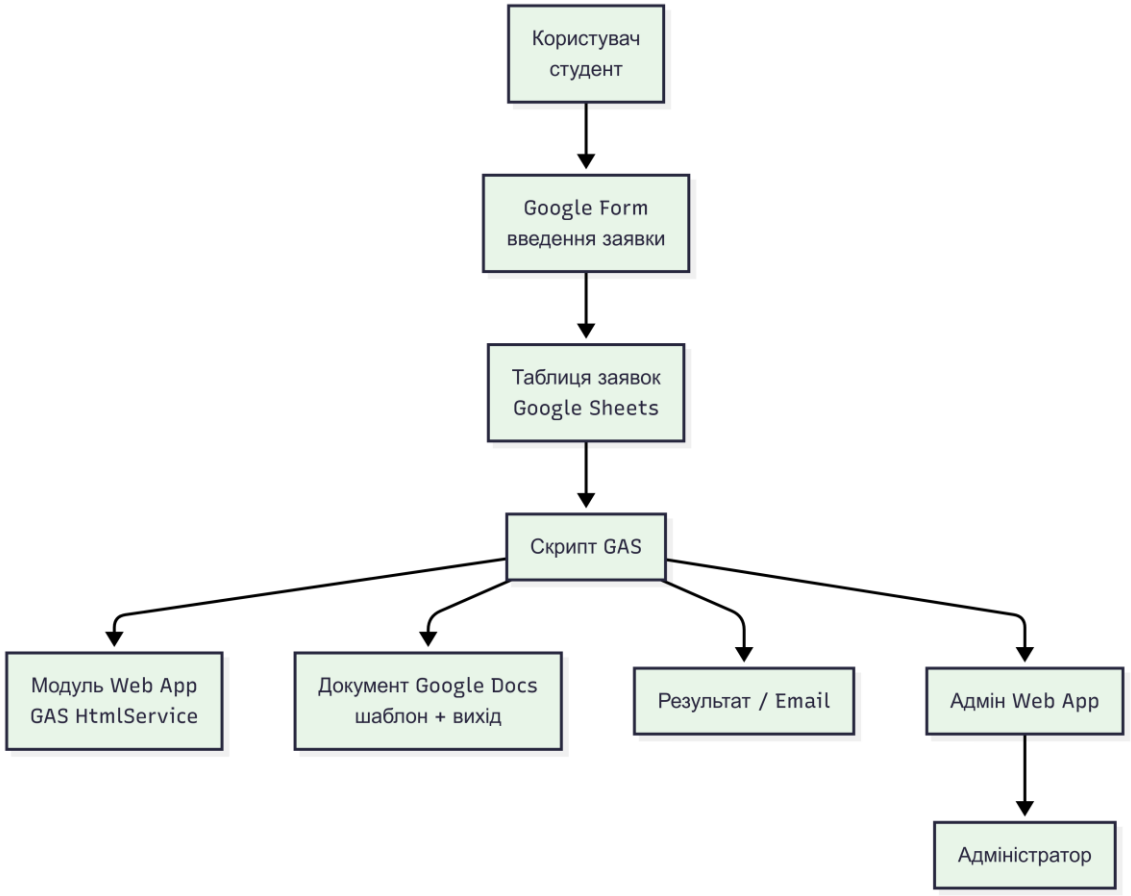

Рисунок 4.2 – Схема інтерфейсної взаємодії

4.4 Верифікація та тестування системи

Верифікація та тестування є ключовими етапами впровадження автоматизованої облікової системи, оскільки вони гарантують її функціональну надійність, коректність логіки, відповідність вимогам та стабільність взаємодії з Google Workspace API. У системах на базі GAS ці процеси поєднують ручне, напівавтоматичне та інтерактивне тестування через скрипти, вебінтерфейс і таблиці.

Тким чином, верифікація системи автоматизованого обліку в Google Workspace включає модульне тестування, перевірку логіки генерації документів, збереження даних та надсилання повідомлень. Завдяки прозорості GAS-інтерфейсу, розробник має змогу оперативно виявляти помилки, вносити виправлення і забезпечувати стабільність функціонування всього комплексу.

4.4.1 Розробка тестових сценаріїв

Тестові сценарії формуються відповідно до модулів і бізнес-логіки системи. Кожен сценарій описує:

- вихідні умови (дані у формі, таблиці, наявність шаблонів);
- очікувану поведінку системи;
- результат, що підлягає перевірці.

Приклади сценаріїв наведені нижче.

1. Збір даних.

1.1. Користувач заповнює Google Form з валідними даними.

1.2. Очікується: запис у requests_log, присвоєння request_id, статус «очікує».

2. Обробка заявки.

2.1. В таблиці requests_log є новий запис.

2.2. Запускається функція generateDocument().

2.3. Очікується: створення Google Docs, посилання в таблиці.

3. Надсилання результату.

3.1. Виявлено новий документ у Drive.

3.2. Запускається MailApp.sendEmail().

3.3. Очікується: лист з посиланням на документ.

4. Аналітика.

4.1. Є щонайменше 10 заявок.

4.2. Запускається analyticsEngine.

4.3. Очікується: оновлення діаграми в аналітика.

Реалізація тестів у коді (ручне виконання):

```
function testSubmission() {
  var fakeData = {
    name: "Тестовий Студент",
    type: "довідка про навчання"
  };
```

```
  var result = submitRequest(fakeData);
  Logger.log("Результат: " + result);
}

function testGeneration() {
  var url = generateCertificate("abc-123", "Тестовий Студент",
"довідка");
  Logger.log("Посилання на документ: " + url);
}
```

GAS також дозволяє використовувати логування (Logger.log) та розділ «Executions», де можна переглянути історію виконання з результатами, помилками та часом виконання.

4.4.2 Аналіз та усунення помилок у роботі системи

У процесі інтеграції та початкового тестування було виявлено низку типових проблем.

1. Некоректні дані у формах (порожні поля, неправильні email). Рішення: додано перевірку (if (email.includes("@"))) і автоматичне виведення помилки.

2. Дублювання заявок. Рішення: реалізовано перевірку останніх записів за ключем email + тип.

3. Недоступність шаблону. Причина: відсутність доступу до файлу шаблону. Рішення: перевірка DriveApp.getFileById(templateId) із повідомленням про помилку.

4. Неправильна робота Web App (відсутній виклик функції з HTML). Рішення: додано обробку google.script.run.withFailureHandler(onError).

5. Повільне виконання скрипта при великій кількості заявок. Рішення: оптимізація доступу до діапазонів (читання масивів, обробка in-memory).

Код для перевірки дубліката:

```
function isDuplicate(email, type) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("requests_l
```

```
og");
  var data = sheet.getDataRange().getValues();
  for (var i = 1; i < data.length; i++) {
    if (data[i][1] === email && data[i][2] === type) return
true;
  }
  return false;
}
```

Схема верифікації та тестування системи (рисунок 4.3) демонструє, як дані проходять через форму, обробку, генерацію документів і аналітику, при цьому тестові функції перевіряють логіку, а журнал виконує роль інструменту для налагодження.

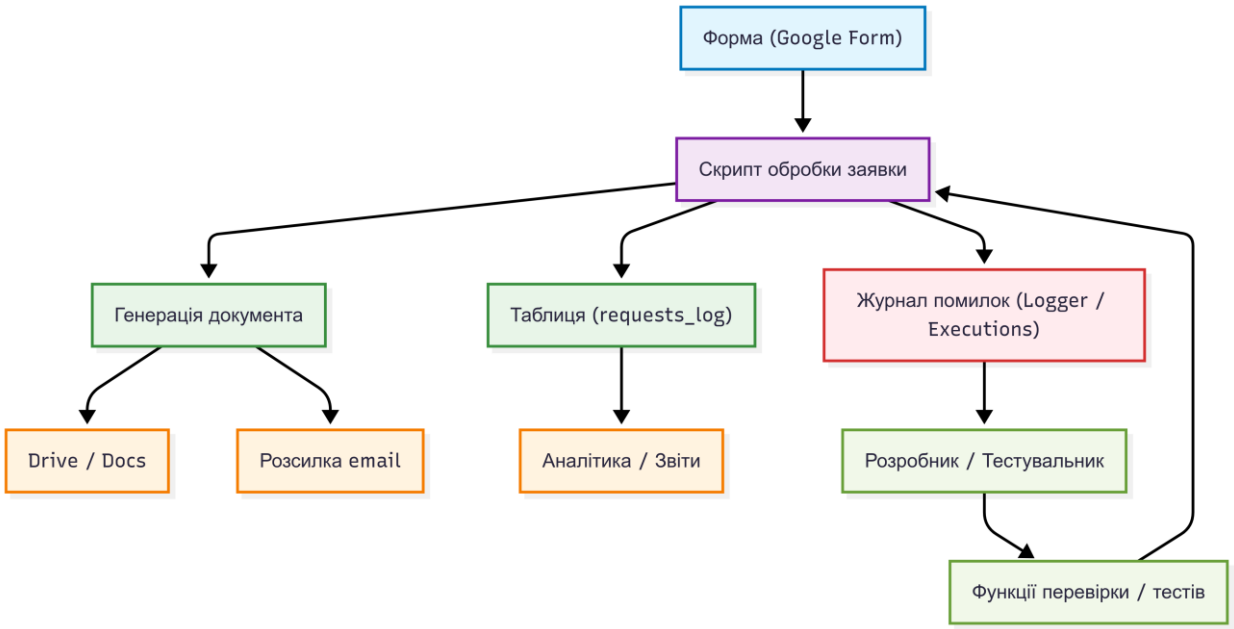

Рисунок 4.3 – Схема верифікації та тестування системи обліку

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено, спроєктовано та реалізовано систему автоматизації обліку з використанням інструментів Google Apps Script (GAS), орієнтовану на потреби закладу вищої освіти. Робота охоплює повний цикл розробки від аналізу предметної області до впровадження та верифікації робочого рішення.

На основі проведеного аналізу встановлено, що традиційні підходи до обліку інформації в ЗВО супроводжуються високим рівнем рутинних операцій, значними часовими витратами та ризиками помилок. Автоматизація цих процесів є необхідною умовою для ефективного функціонування адміністративних підрозділів та підвищення якості сервісу для студентів.

У межах роботи було реалізовано:

- інтеграцію Google Forms, Sheets, Docs, Drive та Gmail у єдину облікову систему, яка підтримує збір заявок, їх обробку, генерацію персоналізованих документів та автоматичну розсилку результатів;

- архітектуру на основі модулів, яка охоплює збирання, перевірку, логування, документування, розсилку та аналітику – з гнучкою можливістю масштабування;

- користувацький інтерфейс, реалізований у вигляді як стандартних Google Forms, так і спеціалізованого Web App на основі GAS HtmlService, що забезпечує інтуїтивну взаємодію користувачів і адміністраторів із системою;

- сценарії тестування і верифікації, що дозволяють забезпечити надійність та стабільність функціонування системи при обробці значної кількості звернень.

Основні переваги розробленої системи:

- мінімальні витрати на розгортання, завдяки використанню безкоштовного хмарного середовища Google Workspace;

- підтримка паралельної роботи багатьох користувачів без конфліктів;
- повна прозорість усіх етапів обробки завдяки логуванню та архівації;
- можливість легкої адаптації під інші форми документів або категорії користувачів.

Отже, розроблена система продемонструвала свою ефективність у задачах автоматизації облікових процесів у ЗВО. Вона може бути впроваджена як в рамках одного факультету, так і в масштабах університету. Використання GAS як платформи дозволяє швидко реагувати на зміну вимог, розширювати функціонал і підтримувати високий рівень надійності та доступності.

Перспективними напрямами подальшого розвитку є:

- інтеграція з API студентських електронних систем (наприклад, ЄДЕБО);
- реалізація багатомовного інтерфейсу;
- додавання авторизації через Google OAuth та обмеження доступу за ролями;
- створення панелі моніторингу з живими діаграмами в реальному часі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ковтун О. І., Бутенко Н. М. Автоматизація обліку навчального навантаження у ЗВО: можливості Excel і VBA // Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки. – 2019. – Т. 30 (69), № 2, ч. 1. – С. 92–97.

2. Duke University. Institutional Research and Planning. Annual Faculty Workload Report 2022. – Durham: Duke Press, 2022. – 34 p.

3. Слободянюк І. І., Розгонюк Н. В. Інтеграція облікових підсистем в єдину інформаційну екосистему ЗВО // Інформаційні технології і засоби навчання. – 2020. – № 3 (77). – С. 41–50.

4. Холод М. В. Інтелектуалізація системи нарахування стипендій студентам у ЗВО // Економіка та держава. – 2021. – № 5. – С. 83–87.

5. Meijer M., de Groot R. Role-based access models in academic systems: case of University of Groningen // European Journal of Higher Education IT. – 2020. – Vol. 12, Issue 2. – P. 103–115.

6. Черниш О. А. Автоматизація формування навчальних планів на основі GAS // Вісник ОНУ. Серія: Інформатика. – 2023. – № 1(28). – С. 74–80.

7. Дмитрієв О. В. Інтеграція GAS з Google Calendar у навчальному процесі // KPI Science News. – 2022. – № 4. – С. 60–64. – Режим доступу: https://kpi.ua/science-news/2022-4-60 (дата звернення: 11.06.2025).

8. Автоматизована система управління навчальним процесом НТУ «ХПІ» [Електронний ресурс]. – Режим доступу: https://www.kpi.kharkov.ua/ukr/it/asu-np/ (дата звернення: 13.06.2025).

9. Порівняльний аналіз автоматизованих систем управління навчанням MOODLE І «UNIVERSYS WS» [Електронний ресурс]. – Режим доступу: https://repository.sspu.edu.ua/items/23e597b0-697e-428a-a50c-3f254be444e5 (дата звернення: 13.06.2025).

10. Blue Prism. University of Calgary Reduces Manual Effort for Key

Finance Processes by 70 % With IA // Blue Prism Case Study. – 2023. – Режим доступу: (з опису кейсу на сайті Blue Prism)

11. Kashyap D. Google Apps Script Case Studies: Real World Process Automation Success Stories // Medium, 09.10.2023. – Режим доступу: https://dilipkashyap15.medium.com/... (дата звернення: 14.06.2025) Medium.

12. Xu Q., Lin L., Wu X. Implementing controlled digital lending with Google Drive and Apps Script: A case study at the NYU Shanghai Library // International Journal of Librarianship. – 2021. – Vol. 6, Issue 1. – С. 37–54. – Режим доступу: https://journal.calaijol.org/index.php/ijol/article/view/193 (дата звернення: 14.06.2025).

13. Bhagare R. Capstone Project: Smart Inventory System using Google Sheets and GAS in CSUEB Food Pantry. – LinkedIn, 2023. – Режим доступу: https://www.linkedin.com/pulse/capstone-project-smart-inventory-system-using-google-sheets-bhagare-lmqsf (дата звернення: 16.06.2025).

14. Zakharchenko Y. Automating Google Sheets with Apps Script: A Case Study. – LinkedIn, 2024. – Режим доступу: https://www.linkedin.com/pulse/automating-google-sheets-apps-script-case-study-yevhen-zakharchenko-usz8f (дата звернення: 16.06.2025).

15. Google Developers. Apps Script Overview. – Режим доступу: https://developers.google.com/apps-script/overview (дата звернення: 17.06.2025).

16. Sayers H. Automating Workflows in Google Workspace Using GAS // EdTech Magazine. – 2022. – Vol. 19, № 3. – Режим доступу: https://edtechmagazine.com/k12/article/2022/06/google-apps-script-education-workflows (дата звернення: 18.06.2025).

17. Шаповал Ю. А. Автоматизація адміністративних процесів у вищій освіті засобами GAS // Освіта і суспільство. – 2022. – № 1. – С. 31–38.

18. Sharma V. Automate Google Forms Processing with Apps Script. – Medium, 2023. – Режим доступу: https://medium.com/google-cloud/automate-google-forms-processing-with-apps-script-2f00d9df7e2a (дата звернення: 18.06.2025).

19. Богданов А. А. Автоматизовані системи збору та обробки даних на платформі Google Sheets // Інформаційні технології в освіті. – 2023. – № 41. – С. 51–60.

20. Smith A. Automating Workflows with Google Apps Script and Workspace APIs. – Medium, 2023. – Режим доступу: https://medium.com/google-cloud/google-apps-script-automation-2023 (дата звернення: 19.06.2025).

21. Ковальчук В. С. Розробка модулів генерації звітів у GAS // Проблеми цифровізації в освіті. – 2023. – № 2. – С. 44–52.

22. Литвиненко І. М. Розробка користувацьких інтерфейсів в Apps Script для облікових систем // Автоматизовані системи. – 2022. – № 3. – С. 52–58.

23. Соколова Н. Підходи до верифікації скриптів в Google Workspace // Системи обробки інформації. – 2023. – № 2. – С. 59–65.