Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеню вищої освіти – магістр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

Тема роботи:
«розробка інформаційної системи аналізу роботи студентів в учбовому процесі, в програмному середовищі python»

Виконав студент гр. КН-23м. _____ Сільман А.М

Керівник _____ Попов С.О.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: інформаційних технологій

Кафедра: автоматизації, комп'ютерних наук і технологій

Ступінь вищої освіти: *Магістр*

Спеціальність: 122 – Комп'ютерні науки

Затверджую

Зав. кафедри: к.т.н. Рубан С.А.

_____

« 29 » червня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи КН-23М Сільману Артему Миколайовичу

1. Тема кваліфікаційної роботи: «Розробка інформаційної системи аналізу роботи студентів в учбовому процесі, в програмному середовищі Python»

затверджено наказом по університету № *589с* від *04.06.2024 р.*

2. Термін здачі кваліфікаційної роботи: *20.12.2024 р.*

3. Склад кваліфікаційної роботи: Пояснювальна записка обсягом 60 с. презентація у Microsoft PowerPoint (12 слайдів) в електронному та друкованому вигляді

4. Консультанти кваліфікаційної роботи:

_____

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	Вступ	04.06.2024 – 18.06.2024
2	Розділ 1	19.06.2024 – 10.07.2024
3	Розділ 2	11.07.2024 – 15.08.2024
4	Розділ 3	16.08.2024 – 30.09.2024
5	Висновки	01.10.2024 – 10.10.2024
6	Оформлення кваліфікаційної роботи	11.10.2024 – 31.10.2024
7	Підготовка презентації та графічного матеріалу	01.11.2024 – 15.11.2024
8	Підготовка доповіді до захисту	11.12.2024

6. Дата видачі завдання: *29.06.2024р.*

Керівник _____ Попов С.О.

7. Запевнення: Я, Сільман Артем Миколайович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Здобувач Сільман Артем Миколайович

АНОТАЦІЯ

Сільман А. М. «Розробка інформаційної системи аналізу роботи студентів в учбовому процесі, в програмному середовищі Python».

Кваліфікаційна робота на здобуття ступеню вищої освіти магістр за освітньо-професійною програмою «Інформаційні системи та технології» зі спеціальності 122 – Інженерія програмного забезпечення. – Криворізький національний університет, Кривий Ріг, 2024.

Метою роботи є розроблення інформаційної системи, яка автоматизує процеси збору, аналізу та обробки даних про успішність та відвідуваність студентів, сприяючи підвищенню ефективності управління навчальним процесом.

Об’єктом дослідження є процеси аналізу успішності студентів у навчальному процесі.

У першому розділі розглянуто теоретичні основи аналізу роботи студентів та вимоги до сучасних інформаційних систем для їхньої підтримки.

У другому розділі проведено розробку інформаційної системи, що дозволяє автоматизувати збирання, обробку та аналіз даних про академічну успішність студентів, використовуючи мову програмування Python та бібліотеки для роботи з базами даних і візуалізацією даних.

У третьому розділі виконано тестування і валідацію системи, а також аналіз результатів її впровадження у навчальний процес.

У роботі розроблено програмний інструмент, що дозволяє навчальним закладам автоматизувати моніторинг успішності студентів, виявляти проблеми у навчальному процесі та приймати обґрунтовані управлінські рішення

Ключові слова:

ІНФОРМАЦІЙНА СИСТЕМА, АНАЛІЗ ДАНИХ, УСПІШНІСТЬ СТУДЕНТІВ, PYTHON, БАЗА ДАНИХ, АВТОМАТИЗАЦІЯ.

ABSTRACT

Silman A. M. "Development of an Information System for Analyzing Student Performance in the Educational Process, in the Python Programming Environment".

Master's qualification work to obtain the degree of higher education in the educational and professional program "Information Systems and Technologies" in the specialty 122 – Software Engineering. – Kryvyi Rih National University, Kryvyi Rih, 2024.

The purpose of the work is to develop an information system that automates the processes of collecting, analyzing, and processing data on students' performance and attendance, contributing to the improvement of the efficiency of educational process management.

The object of the research is the processes of analyzing students' academic performance in the educational process.

The first chapter reviews the theoretical foundations of student performance analysis and the requirements for modern information systems to support it.

The second chapter is devoted to the development of an information system that automates the collection, processing, and analysis of data on students' academic performance using the Python programming language and libraries for database management and data visualization.

The third chapter presents the testing and validation of the system, as well as an analysis of its implementation results in the educational process.

The study developed a software tool that enables educational institutions to automate the monitoring of students' performance, identify issues in the educational process, and make informed management decisions.

Keywords:

INFORMATION SYSTEM, DATA ANALYSIS, STUDENT PERFORMANCE, PYTHON, DATABASE, AUTOMATION.

ЗМІСТ

ВСТУП

В сучасних умовах розвитку освіти все більшого значення набувають технології, які сприяють підвищенню ефективності навчального процесу. Інформаційні системи, що автоматизують різні аспекти навчальної діяльності, дозволяють викладачам і адміністраторам оперативно отримувати дані про успішність студентів, виявляти слабкі сторони в їх підготовці, а також аналізувати тенденції в процесі навчання.

Одним із перспективних підходів до розробки таких систем є використання мов програмування високого рівня, зокрема Python, який завдяки своїй гнучкості та широкому набору бібліотек забезпечує ефективне створення програмних рішень різної складності.

Метою даної дипломної роботи є розробка інформаційної системи, яка дозволить аналізувати результати роботи студентів протягом навчального процесу, відстежувати прогрес і надавати рекомендації для покращення успішності. Розробка цієї системи передбачає використання Python як основного інструменту для обробки даних, побудови аналітичних моделей та забезпечення зручного інтерфейсу для користувачів.

Об'єкт роботи – інформаційна система аналізу роботи студентів.

*Завдання роботи***:**

– огляд і аналіз літературних джерел з теми роботи;

– розробка методичних основ роботи інформаційної систем, що розробляється;

– розробка структурної схеми інформаційної системи;

– розробка алгоритму роботи інформаційної системи;

– розробка програмного забезпечення інформаційної системи;

– висновки з результатів розробки інформаційної систем і рекомендацій з її викоритання.

В ході виконання роботи будуть розглянуті основні принципи побудови інформаційних систем для навчальних закладів, методи обробки та аналізу

даних про студентів, а також сучасні підходи до інтеграції таких систем у навчальний процес.

Ключовим завданням інформаційної системи є автоматизація збору та обробки даних про результати роботи студентів, що дозволить значно спростити рутинні процеси аналізу. Це забезпечить викладачам можливість більш оперативно реагувати на зміни в успішності студентів, а також надавати індивідуальні рекомендації з урахуванням особистих потреб кожного студента. Окрім цього, така система сприятиме покращенню організації навчального процесу та підвищенню його ефективності.

Особливістю розробки є інтеграція різних методів аналізу, таких як статистичні моделі, методи кластеризації та машинного навчання, що допоможе краще розуміти поведінку студентів та виявляти закономірності в їх успішності. Використання Python для реалізації цього функціоналу є оптимальним вибором завдяки наявності розвинених бібліотек для роботи з даними, таких як NumPy, Pandas, Matplotlib, а також Scikit-learn для побудови та тестування моделей машинного навчання.

Також в роботі буде приділено увагу розробці користувацького інтерфейсу, який забезпечить зручний доступ до основних функцій системи. Це передбачає можливість візуалізації даних, що дозволить викладачам швидко отримувати необхідну інформацію у зрозумілому форматі. Для цього можуть бути використані інструменти для побудови графіків і діаграм, такі як Matplotlib або Plotly, що забезпечать наочне подання результатів аналізу.

Таким чином, реалізація інформаційної системи аналізу роботи студентів сприятиме підвищенню якості навчання за рахунок своєчасного надання викладачам та адміністраторам необхідних даних, що допоможе в прийнятті обґрунтованих рішень щодо вдосконалення навчального процесу.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ НАВЧАННЯ СТУДЕНТІВ В ХОДІ НАВЧАЛЬНОГО ПРОЦЕСУ

1.1 Огляд сучасних систем аналізу роботи студентів

Сучасні інформаційні системи аналізу роботи студентів є важливим інструментом для підвищення якості навчального процесу. Такі системи дозволяють автоматизувати збір, обробку та аналіз даних щодо навчальної діяльності студентів, що суттєво полегшує процес оцінювання та контролю знань. Зокрема, ці системи надають можливість викладачам отримувати оперативну інформацію про прогрес студентів, виявляти слабкі місця в їх підготовці та своєчасно вживати заходів для підвищення ефективності навчання.

Одним із ключових аспектів сучасних систем аналізу є їх здатність інтегруватися з іншими освітніми платформами, такими як системи дистанційного навчання Moodle, Google Classroom або системи управління навчанням Learning Management Systems, LMS. Це дозволяє автоматизувати процеси збору та зберігання даних, що стосуються результатів виконання тестів, відвідуваності, виконаних завдань та інших видів діяльності студентів.

Серед популярних систем аналізу роботи студентів можна виділити такі платформи, як Classroom Analytics, яка надає візуалізацію прогресу студентів у вигляді діаграм і графіків, та Tableau, що дозволяє створювати індивідуальні звіти для аналізу результатів діяльності кожного студента. Системи подібного роду не лише допомагають викладачам в оцінюванні роботи студентів, але й дозволяють самим студентам стежити за своїм навчальним процесом та самостійно аналізувати власні досягнення.

Окрім готових платформ, все більшого поширення набувають індивідуально розроблені рішення, які адаптовані до специфіки кожного навчального закладу. Такі системи можуть містити унікальні функції, наприклад, відстеження мотивації студентів, аналіз продуктивності в режимі реального часу або синхронізацію з іншими навчальними ресурсами. Вони

дозволяють зробити навчальний процес більш прозорим, структурованим і персоналізованим.

Індивідуально розроблені інформаційні системи мають значні переваги над готовими рішеннями, оскільки враховують специфічні потреби конкретного навчального закладу та його освітньої програми. Такі рішення дозволяють гнучко налаштовувати систему під різні навчальні дисципліни, типи оцінювання та методи навчання, що значно підвищує їх ефективність у порівнянні з універсальними платформами. Наприклад, можна налаштувати систему для автоматизованого оцінювання лабораторних робіт, проведення тестування з адаптивними питаннями або інтеграції з існуючими платформами для дистанційного навчання, такими як Moodle чи Google Classroom.

Крім того, такі системи можуть містити розширені аналітичні інструменти, які допоможуть викладачам відслідковувати динаміку розвитку кожного студента та проводити комплексний аналіз їхніх академічних досягнень. Це дозволяє швидше і точніше виявляти труднощі, з якими стикається студент, та пропонувати індивідуальні шляхи їх подолання. Наприклад, система може автоматично генерувати рекомендації для покращення результатів, надавати доступ до додаткових навчальних матеріалів або пропонувати консультації з викладачем.

Ще однією перевагою індивідуальних рішень є можливість їх інтеграції з системами управління даними студентів (ERP-системами), що забезпечує безперервну синхронізацію між навчальним процесом і адміністративною діяльністю закладу. Це дозволяє уникнути дублювання даних та значно підвищує ефективність роботи як викладачів, так і адміністрації.

Загалом, сучасні системи аналізу роботи студентів орієнтовані на забезпечення об'єктивної та зручної системи оцінювання, покращення зворотного зв'язку між викладачем та студентом, а також сприяють підвищенню ефективності організації навчального процесу в цілому.

1.2 Основні методи підвищення ефективності навчального процесу

Підвищення ефективності учбового процесу є однією з ключових задач освітніх установ, оскільки від цього залежить якість підготовки фахівців. На сьогодні існує кілька методів та підходів, які дозволяють зробити навчання більш результативним, гнучким і адаптованим до потреб студентів та вимог ринку праці.

1. Використання інформаційних технологій.

Інтеграція інформаційних технологій в навчальний процес дозволяє значно підвищити його ефективність. Системи управління навчальним процесом Learning Management Systems, LMS, електронні підручники, інтерактивні платформи та засоби дистанційного навчання дозволяють автоматизувати значну частину навчальної діяльності. Це полегшує доступ до навчальних матеріалів, забезпечує можливість самостійного вивчення, а також сприяє кращому контролю за успішністю студентів. Зокрема, такі системи дозволяють відстежувати виконання завдань, здавати тести онлайн, отримувати зворотний зв'язок в режимі реального часу, що значно підвищує мотивацію до навчання.

2. Персоналізація навчального процесу.

Сучасні підходи в освіті орієнтовані на персоналізацію навчання, що дозволяє враховувати індивідуальні потреби, здібності та інтереси кожного студента. Використання адаптивних технологій навчання дозволяє будувати навчальні програми, які враховують прогрес студента та його сильні і слабкі сторони. Це може бути досягнуто за допомогою адаптивних тестувань, які автоматично змінюють рівень складності завдань на основі результатів попередніх спроб, або систем рекомендацій, які пропонують індивідуальний набір матеріалів для вивчення.

Такі адаптивні підходи стають особливо актуальними в умовах дистанційного навчання та використання онлайн-платформ. Завдяки штучному інтелекту та великим даним освітні системи можуть відстежувати поведінку студентів, аналізувати їхні успіхи та прогалини, а також надавати індивідуальні рекомендації. Це дозволяє підвищити мотивацію до навчання,

оскільки студент отримує саме ті завдання, які відповідають його рівню знань та інтересам, що сприяє глибшому засвоєнню матеріалу.

Окрім цього, сучасні підходи включають інтеграцію ігрових елементів (гейміфікація), які роблять процес навчання більш захоплюючим та сприяють розвитку навичок, таких як критичне мислення, командна робота та вирішення проблем. Наприклад, створення навчальних симуляцій та інтерактивних завдань, де студенти можуть застосовувати свої знання в реальних умовах, допомагає їм не тільки краще засвоїти матеріал, але й підготуватися до реальних професійних викликів.

3. Активне навчання.

Методи активного навчання, такі як проблемно-орієнтоване навчання, групові проекти, рольові ігри та кейс-методи, є потужними засобами для підвищення ефективності навчання. Вони сприяють активному залученню студентів у процес здобуття знань, розвитку критичного мислення та навичок командної роботи. Використання таких підходів дозволяє студентам глибше опановувати матеріал та краще застосовувати його на практиці.

Такі методи сприяють не лише кращому засвоєнню знань, але й розвитку так званих м'яких навичок soft skills, які стають усе більш важливими у сучасному світі. Завдяки груповим проектам студенти вчаться ефективно комунікувати, розподіляти обов'язки, вирішувати конфлікти та приймати спільні рішення. Рольові ігри дозволяють зануритися в конкретні професійні ситуації та відпрацьовувати навички прийняття рішень, реагування на виклики та взаємодії з іншими учасниками процесу.

Кейс-методи, коли студенти аналізують реальні або вигадані ситуації, дозволяють набути навичок аналізу проблем, пошуку рішень і розробки стратегій дій. Вони також допомагають навчитися враховувати різні точки зору, прогнозувати можливі наслідки і розвивати стратегічне мислення. Проблемно-орієнтоване навчання (Problem-Based Learning) спрямоване на вирішення конкретних завдань або проблем, стимулюючи студентів самостійно шукати необхідну інформацію, аналізувати її та пропонувати

рішення. Такий підхід сприяє розвитку дослідницьких навичок і підготовці до реальної професійної діяльності.

4. Формативне оцінювання.

Формативне оцінювання є важливим інструментом для моніторингу прогресу студентів протягом усього навчального процесу. Це метод, який передбачає регулярне проведення оцінювання та надання студентам зворотного зв'язку на всіх етапах навчання. Такий підхід дозволяє студентам постійно коригувати свою навчальну діяльність та працювати над покращенням результатів. Формативне оцінювання сприяє активному залученню студентів до процесу саморефлексії та самоконтролю, що стимулює досягнення високих результатів.

5. Впровадження змішаного навчання.

Змішане навчання blended learning поєднує в собі традиційні методи викладання та електронні ресурси, що дозволяє більш гнучко організовувати учбовий процес. Це дозволяє поєднувати переваги особистого спілкування між викладачем та студентом з можливостями самостійної роботи онлайн. Завдяки змішаному підходу можна оптимізувати навчальний час, краще організовувати домашню роботу та індивідуальні завдання.

6. Менторство та підтримка.

Наявність менторства, коли студентам надається індивідуальна підтримка та керівництво з боку досвідчених викладачів або старших студентів, також суттєво підвищує ефективність навчання. Ментори допомагають студентам вирішувати навчальні проблеми, надають рекомендації щодо організації навчальної діяльності, мотивації та професійного розвитку. Цей підхід стимулює особистісний ріст і сприяє формуванню професійних навичок.

Таким чином, сучасні методи підвищення ефективності навчального процесу ґрунтуються на інтеграції інформаційних технологій, активних методів навчання, персоналізації освітніх програм та формуванні культури саморозвитку. Використання цих методів дозволяє зробити навчальний

процес більш гнучким, адаптованим до потреб студентів та ефективним у досягненні освітніх цілей.

Крім того, важливим аспектом сучасної освіти є розвиток автономії студентів, що стимулює їх до самостійного пошуку інформації, критичного аналізу джерел і постійного вдосконалення своїх знань та навичок. В умовах стрімкого розвитку технологій і швидких змін у суспільстві здатність до саморегульованого навчання стає ключовою для успішної кар'єри.

Інтеграція інформаційних технологій у навчальний процес також забезпечує доступ до різноманітних навчальних ресурсів, які можна використовувати в будь-який час і з будь-якого місця. Онлайн-курси, вебінари, інтерактивні симуляції та інші цифрові інструменти дозволяють студентам поглиблювати свої знання у власному темпі, розширюючи можливості для самонавчання. Це сприяє розвитку навичок, таких як управління часом, постановка цілей і відповідальність за власні результати.

Персоналізація освітніх програм, що враховує індивідуальні потреби і цілі кожного студента, робить процес навчання більш цілеспрямованим та ефективним.

1.3 Проблеми традиційних підходів до оцінювання роботи студентів

Традиційні підходи до оцінювання роботи студентів, які ґрунтуються на класичних методах перевірки знань, таких як письмові екзамени, тести та усні відповіді, залишаються широко поширеними в освітніх закладах. Однак вони мають низку недоліків, які обмежують їх ефективність і не завжди відповідають вимогам сучасного навчального процесу.

1. Обмеженість у відображенні повної картини знань.

Традиційні методи оцінювання часто орієнтовані на перевірку лише певних аспектів знань студентів, зокрема запам'ятовування фактів та формул. Вони можуть не враховувати розвиток таких важливих навичок, як критичне мислення, вміння застосовувати знання на практиці, аналіз і синтез інформації. Наприклад, стандартний письмовий екзамен оцінює здатність студента відтворити завчену інформацію в умовах обмеженого часу, але не завжди

дозволяє виявити його справжні знання і вміння вирішувати комплексні проблеми.

2. Відсутність безперервного оцінювання.

Традиційні підходи зазвичай передбачають оцінювання роботи студента на основі кількох ключових завдань, таких як семестрові екзамени чи контрольні роботи. Це створює ситуацію, коли оцінка залежить від одноразового результату, що може бути випадковим або неповним відображенням знань. Такі методи не враховують прогрес студента протягом усього періоду навчання і не забезпечують безперервного моніторингу розвитку його навичок і знань.

3. Суб'єктивність оцінювання.

Традиційні методи оцінювання можуть бути суб'єктивними, особливо у випадках усних відповідей чи есе. Викладачі можуть мати різні стандарти оцінки, що призводить до неоднакових результатів для студентів з однаковим рівнем знань. Суб'єктивність оцінювання може вплинути на справедливість і прозорість процесу оцінки, що, у свою чергу, може демотивувати студентів.

4. Психологічний стрес і тиск.

Традиційні методи оцінювання, такі як письмові екзамени та контрольні роботи, часто створюють значний психологічний стрес для студентів. В умовах обмеженого часу студенти можуть відчувати тиск, що негативно впливає на їх здатність продемонструвати реальні знання. Такі підходи можуть призводити до надмірного акцентування на оцінках, замість фокусу на процесі навчання і глибокому розумінні матеріалу.

5. Недостатня увага до індивідуальних особливостей студентів.

Традиційні підходи до оцінювання рідко враховують індивідуальні особливості студентів, такі як їхні когнітивні здібності, стиль навчання та особистісні риси. Вони орієнтовані на стандартизовані методи перевірки, які можуть бути ефективними для одних студентів і неефективними для інших. Це створює нерівні умови для студентів з різними стилями навчання і здібностями, що може призводити до несправедливого оцінювання.

6. Низька інтерактивність та відсутність зворотного зв'язку.

Традиційні методи оцінювання не завжди забезпечують оперативний зворотний зв'язок для студентів. Часто студенти отримують підсумкові оцінки без детального аналізу своїх помилок і рекомендацій щодо поліпшення результатів. Відсутність регулярного зворотного зв'язку знижує мотивацію до навчання і не дозволяє студентам своєчасно коригувати свої навчальні стратегії.

Для подолання цих проблем важливо впроваджувати сучасні методи оцінювання, які забезпечують регулярний, детальний і конструктивний зворотний зв'язок.

Одним із таких підходів є формувальне оцінювання, яке відрізняється тим, що воно не лише підсумовує досягнення студента, але й слугує інструментом для постійного вдосконалення.

Викладачі можуть надавати студентам рекомендації під час навчального процесу, вказувати на сильні та слабкі сторони, а також пропонувати стратегії для подальшого розвитку.

Іншою важливою складовою сучасних підходів до оцінювання є саморефлексія та взаємооцінювання.

Завдяки цьому студенти навчаються оцінювати власну роботу, аналізувати власні досягнення та помилки, а також отримують можливість обговорювати результати з однолітками.

Це допомагає не лише глибше розуміти навчальний матеріал, а й розвивати критичне мислення та відповідальність за свої результати.

Застосування цифрових технологій також може значно покращити процес оцінювання.

Адаптивні системи тестування, інтерактивні платформи та онлайн-інструменти дозволяють автоматично аналізувати помилки студентів та надавати індивідуальні рекомендації.

Це дозволяє студентам отримувати зворотний зв'язок у реальному часі, що підвищує ефективність навчання та мотивацію до самовдосконалення.

7. Відсутність підготовки до реальних умов.

Традиційні методи оцінювання часто не готують студентів до вирішення реальних життєвих чи професійних задач, оскільки вони сфокусовані на теоретичних аспектах навчання. Студенти можуть засвоїти знання, але не мати навичок їх застосування в реальних ситуаціях, що знижує їхню конкурентоспроможність на ринку праці. Практичні та інтерактивні методи оцінювання, такі як проекти, презентації та групові завдання, є більш ефективними для підготовки студентів до реальних викликів.

1.4 Переваги використання інформаційних систем для підвищення ефективності навчання

Використання інформаційних систем у навчальному процесі значно підвищує ефективність освіти за рахунок автоматизації багатьох аспектів викладання та оцінювання знань. Сучасні інформаційні системи дозволяють зручно та оперативно обробляти великі обсяги даних, забезпечують гнучкість та персоналізацію навчального процесу, а також сприяють активній взаємодії між студентами та викладачами. Ось ключові переваги використання таких систем у навчанні:

1. Автоматизація процесів та зменшення адміністративного навантаження

Однією з основних переваг інформаційних систем є автоматизація рутинних операцій, таких як ведення журналів відвідуваності, облік результатів тестування, контроль виконання завдань і збереження оцінок. Викладачі можуть використовувати ці системи для автоматичного збирання й обробки даних, що значно знижує їх адміністративне навантаження і дозволяє більше часу приділяти безпосередньо навчальному процесу. Наприклад, системи управління навчанням (Learning Management Systems, LMS) дозволяють автоматизувати розклад занять, розсилку матеріалів та контроль за виконанням завдань.

2. Покращення якості зворотного зв’язку.

Інформаційні системи забезпечують можливість швидкого й ефективного зворотного зв’язку між студентами і викладачами. Системи надають детальні

звіти про виконання завдань, результати тестувань та прогрес студентів. Викладачі можуть оперативно надавати коментарі щодо результатів, вказуючи на помилки і даючи рекомендації для їх виправлення. Це стимулює студентів працювати над своїми слабкими сторонами і сприяє безперервному покращенню їхньої успішності.

3. Персоналізація навчального процесу.

Інформаційні системи дозволяють адаптувати навчальний процес під потреби кожного студента. Наприклад, завдяки аналізу даних про успішність студента система може рекомендувати додаткові навчальні матеріали або завдання для повторення конкретних тем, що викликають труднощі. Це сприяє кращому засвоєнню матеріалу і дозволяє кожному студенту працювати у власному темпі. Крім того, студенти можуть використовувати різні навчальні ресурси (відеолекції, електронні підручники, інтерактивні вправи) залежно від своїх індивідуальних потреб.

4. Підвищення доступності навчальних матеріалів.

Інформаційні системи надають можливість студентам отримувати доступ до навчальних матеріалів у будь-який зручний для них час і з будь-якого місця. Завдяки онлайн-платформам, таким як Moodle чи Google Classroom, студенти можуть переглядати лекції, читати літературу, виконувати завдання та здавати їх онлайн. Це особливо важливо для дистанційного навчання та самостійної роботи студентів, адже дає їм змогу вчитися незалежно від географічного розташування або розкладу.

Крім доступу до навчальних матеріалів, інформаційні системи також забезпечують інтерактивність навчального процесу. Студенти можуть брати участь у дискусіях на форумах, задавати питання викладачам у режимі реального часу, проходити тести та миттєво отримувати результати. Це підвищує ефективність зворотного зв’язку та дозволяє викладачам швидко реагувати на проблеми, з якими можуть стикатися студенти.

Інформаційні системи також сприяють колаборації між студентами. За допомогою спільних документів, групових чатів та проєктних платформ

студенти можуть працювати разом над завданнями, не зважаючи на відстань чи індивідуальні графіки. Це дозволяє студентам розвивати навички командної роботи та комунікації, що є важливими у сучасному світі, де значна частина роботи виконується в дистанційних або міжнародних командах.

Ще однією перевагою інформаційних систем є можливість індивідуалізованого підходу до навчання. Платформи, як-от Moodle, можуть надавати різні траєкторії навчання залежно від рівня знань студента. Завдяки адаптивним алгоритмам система може автоматично підбирати завдання відповідної складності або пропонувати додаткові матеріали для глибшого опанування теми.

Окрім цього, інформаційні системи полегшують управління навчальним процесом для викладачів. Вони можуть організовувати курси, розміщувати матеріали, виставляти оцінки, створювати тести й автоматично оцінювати їх. Це значно зменшує адміністративне навантаження на викладачів, дозволяючи їм більше часу приділяти індивідуальній роботі зі студентами та вдосконаленню навчальних матеріалів.

5. Можливість аналізу великих обсягів даних.

Інформаційні системи дозволяють зібрати й аналізувати великі обсяги даних щодо навчальної діяльності студентів. Це дає можливість викладачам відстежувати прогрес кожного студента, виявляти тенденції та слабкі місця у навчальному процесі. На основі цих даних можна приймати більш обґрунтовані рішення щодо коригування методів викладання, вдосконалення програм навчання та оптимізації навчального процесу загалом. Крім того, такі дані можуть бути використані для оцінки ефективності навчальних програм і навчальних планів.

6. Інтерактивність та залучення студентів.

Сучасні інформаційні системи сприяють активному залученню студентів у навчальний процес завдяки використанню інтерактивних елементів, таких як вікторини, форуми, вебінари та спільна робота над проектами. Такі підходи роблять навчання більш динамічним і цікавим, стимулюють студентів до

активної участі в освітніх заходах. Крім того, інтерактивність допомагає покращити засвоєння матеріалу, оскільки студенти отримують можливість застосовувати знання на практиці.

7. Гнучкість і адаптивність.

Інформаційні системи забезпечують гнучкість в організації навчального процесу. Студенти можуть самостійно планувати свій навчальний час, отримувати доступ до необхідних матеріалів у будь-який момент і контролювати свій прогрес. Викладачі також мають змогу налаштовувати різні типи завдань, організовувати дистанційне навчання або впроваджувати елементи змішаного навчання, що робить процес більш адаптивним і зручним як для студентів, так і для викладачів.

Інформаційні системи в освіті не лише забезпечують доступ до матеріалів і можливість планування, але й сприяють створенню індивідуальних траєкторій навчання для кожного студента. Завдяки аналітичним інструментам, викладачі можуть відстежувати прогрес кожного студента, виявляти проблемні аспекти й адаптувати навчальні програми відповідно до потреб і рівня знань конкретної людини. Це дозволяє більш ефективно використовувати час, надаючи можливість студентам приділяти більше уваги тим темам, які потребують додаткового опрацювання.

Також інформаційні системи відкривають можливості для впровадження гейміфікації у навчальний процес. Використання елементів ігрового дизайну, таких як бали, рейтинги, нагороди, сприяє підвищенню мотивації студентів. Це допомагає створити конкурентне середовище, де студенти мають можливість змагатися або співпрацювати між собою для досягнення навчальних цілей. Гейміфіковані завдання роблять процес навчання більш цікавим і динамічним, що особливо важливо для підтримки інтересу в довгостроковій перспективі.

Дистанційне та змішане навчання, яке активно впроваджується за допомогою інформаційних систем, дозволяє студентам поєднувати різні форми навчання. Наприклад, студенти можуть проходити теоретичний

матеріал онлайн, а практичні заняття або обговорення проводити у форматі живих зустрічей. Це робить навчальний процес більш гнучким та адаптованим до різних обставин, таких як географічна віддаленість студентів або потреба у поєднанні навчання з роботою.

Завдяки автоматизації багатьох рутинних процесів, таких як перевірка завдань, збирання результатів тестування, складання розкладів тощо, викладачі можуть більше часу приділяти безпосередній взаємодії зі студентами та наданню індивідуальних рекомендацій. Інформаційні системи також полегшують процес комунікації між викладачами і студентами, забезпечуючи швидкий обмін інформацією через форуми, чати, відеоконференції або електронну пошту.

Окрім цього, сучасні системи управління навчанням Learning Management Systems, LMS дозволяють інтегрувати різноманітні формати навчальних матеріалів – від текстів і презентацій до відео, інтерактивних симуляцій та віртуальних лабораторій. Це сприяє створенню багатофункціональних середовищ для навчання, де студенти можуть взаємодіяти з різними типами контенту, що відповідають їхнім навчальним стилям. Деякі студенти краще засвоюють матеріал через відео або аудіо, тоді як інші надають перевагу читанню або практичним завданням. Інформаційні системи дозволяють задовольнити ці різноманітні потреби, що робить навчання більш ефективним для кожного окремого студента.

Ще однією перевагою інформаційних систем є їх здатність забезпечувати безперервний доступ до навчальних ресурсів. Навчальні матеріали, записи лекцій, результати тестів, аналітичні звіти про успішність – усе це доступно студентам у будь-який час. Це особливо важливо для студентів, які навчаються дистанційно або в умовах, коли фізична присутність у навчальному закладі обмежена.

Інформаційні системи також підтримують концепцію безперервного навчання протягом життя (lifelong learning), оскільки дозволяють студентам повертатися до матеріалів після завершення курсу, оновлювати свої знання

або проходити нові курси для підвищення кваліфікації. Це є важливою частиною сучасного суспільства, де швидкість змін у професійних сферах вимагає постійного оновлення знань та навичок.

Інформаційні системи відіграють важливу роль у забезпеченні прозорості процесу навчання. Студенти можуть бачити свої оцінки, отримувати коментарі до завдань та мати доступ до статистики власних досягнень. Це сприяє підвищенню відповідальності студентів за власні результати та формуванню культури самоконтролю.

Щодо викладачів, інформаційні системи дозволяють автоматизувати процес збору та аналізу даних про успішність студентів. Викладачі можуть легко відслідковувати динаміку прогресу групи або окремих студентів, визначати теми, які потребують додаткового опрацювання, а також коригувати навчальні стратегії залежно від результатів аналізу. Така система підтримки прийняття рішень дозволяє зробити процес навчання більш обґрунтованим та ефективним.

Важливо зазначити, що впровадження інформаційних систем у навчальний процес вимагає відповідної технічної підготовки викладачів. Для успішного використання всіх переваг таких систем викладачі повинні мати навички роботи з цифровими інструментами, знати можливості платформ та мати уявлення про те, як адаптувати свої методи викладання до нових технологічних реалій. Це може потребувати додаткового навчання та підтримки з боку освітніх установ.

Таким чином, інформаційні системи не лише спрощують організацію навчального процесу, але й роблять його більш адаптивним, індивідуалізованим і ефективним. Вони дозволяють студентам та викладачам гнучко керувати своїм часом, отримувати постійний доступ до навчальних ресурсів та ефективно взаємодіяти між собою. Інтеграція таких систем сприяє розвитку культури самонавчання, підвищенню мотивації та відповідальності за результати навчання, що є ключовими факторами успішної освіти в умовах сучасного світу.

1.5 Ідентифікація математичної моделі та синтез керування навчальним процесом

Ідентифікація математичної моделі процесу навчання студентів є важливим етапом для подальшого синтезу системи керування ефективністю їх роботи. Математична модель дозволяє кількісно оцінювати динаміку успішності студентів, враховувати різноманітні фактори впливу та прогнозувати результати їхньої роботи залежно від змін навчального процесу.

Ідентифікація моделі.

Процес ідентифікації моделі передбачає вибір адекватного математичного опису навчального процесу з урахуванням реальних даних про успішність студентів. Основними параметрами, що впливають на ефективність навчання, можуть бути:

- кількість витраченого часу на підготовку до занять;
- активність на лекціях та практичних заняттях;
- результати контрольних робіт і тестів;
- рівень зацікавленості студентів у навчальному процесі.

Для ідентифікації математичної моделі використовуються статистичні методи та методи машинного навчання. Наприклад, можна застосовувати методи регресійного аналізу або нейронні мережі, щоб знайти залежності між успішністю студентів та впливовими факторами. Це дозволяє створити модель, яка відображатиме динаміку змін успішності залежно від введених даних.

Синтез системи керування

Після побудови математичної моделі необхідно синтезувати систему керування, яка дозволить покращити ефективність навчання. Основними завданнями системи керування є:

- моніторинг та аналіз успішності студентів;
- прогнозування результатів на основі поточних даних;
- рекомендації щодо покращення методик викладання та навчання.

Застосування адаптивних алгоритмів керування дозволяє постійно коригувати навчальні методики залежно від результатів студентів. Наприклад,

на основі даних про зниження успішності можна збільшувати час на додаткові заняття або змінювати стратегію подачі матеріалу.

Одним із підходів до синтезу керування є використання алгоритмів оптимізації, які дозволяють знаходити найкращі параметри для підвищення результативності студентів. Такі методи можуть бути реалізовані на основі математичних моделей.

Використання моделі для автоматизації навчального процесу.

Створена модель може бути інтегрована в інформаційну систему для автоматизації навчального процесу. Це дозволить автоматично збирати дані, аналізувати результати та формувати рекомендації щодо індивідуального навчального плану для кожного студента. Інтеграція математичних моделей у систему керування дасть можливість підвищити точність оцінки успішності та швидкість реагування на зміни в навчальному процесі.

Таким чином, ідентифікація математичної моделі та синтез системи керування навчальним процесом дозволяють оптимізувати процеси навчання, підвищити їх ефективність і автоматизувати аналіз успішності студентів.

1.6 Огляд математичних моделей управління учбовим процесом

Математичне моделювання є важливим інструментом для дослідження, аналізу та оптимізації процесів управління. Моделі, побудовані на основі математичних принципів, дозволяють детально описати поведінку системи, виявити ключові взаємозв’язки між елементами та розробити алгоритми ефективного керування. В контексті освітнього процесу, де важливо контролювати навчальну діяльність, аналізувати успішність студентів та адаптувати процеси під індивідуальні потреби, застосування математичних моделей є доцільним для вдосконалення якості управління навчальними процесами.

Існує кілька підходів до математичного моделювання процесів управління, які включають використання диференціальних рівнянь, дискретних моделей, стохастичних процесів, а також методи теорії керування та оптимізації.

Розглянемо основні типи математичних моделей, що застосовуються для управління складними процесами:

Детерміновані моделі – дані моделі описують системи, поведінка яких повністю визначена початковими умовами та математичними рівняннями. Вони широко застосовуються для опису систем, де всі фактори можуть бути чітко виміряні й спрогнозовані. Детерміновані моделі є основою для багатьох систем автоматичного керування і забезпечують прогнозовану реакцію системи на зміни вхідних параметрів.

Стохастичні моделі – вони використовуються для опису систем, де певні процеси мають випадковий характер або невизначеність. Такі моделі враховують імовірнісні характеристики змінних і є корисними у випадках, коли не можна чітко передбачити результати дій через вплив непередбачуваних факторів. Стохастичні моделі знаходять застосування в освіті для моделювання поведінки студентів, прогнозування успішності та аналізу ризиків.

Дискретні моделі – вони використовуються для моделювання процесів, які відбуваються у визначені моменти часу. Такі моделі зручні для опису систем, де події відбуваються не безперервно, а у вигляді окремих кроків. У контексті управління навчальними процесами дискретні моделі дозволяють аналізувати етапи виконання студентами завдань, моніторинг проміжних результатів і прийняття рішень на основі даних про попередні етапи навчання.

Моделі на основі теорії керування – ці моделі розроблені для оптимізації процесів керування в системах із зворотним зв’язком. Вони дозволяють враховувати відхилення в процесі та коригувати управлінські дії для досягнення бажаного результату. У навчальних процесах такі моделі можуть використовуватися для адаптації навчальних матеріалів і завдань залежно від поточного рівня знань студента.

Оптимізаційні моделі – націлені на пошук найкращих рішень для управління системою при певних обмеженнях. У контексті освіти, це може

стосуватися розробки оптимальних стратегій навчання, розподілу ресурсів між студентами або вибору ефективних методик викладання.

1.7 Постановка задачі математичного моделювання процесу навчання

Математичне моделювання навчального процесу є важливим етапом у вдосконаленні систем управління освітою. В умовах сучасного світу, де обсяги інформації стрімко зростають, а методи навчання стають все більш індивідуалізованими, виникає необхідність у розробці нових підходів для оптимізації процесу навчання. Задача математичного моделювання полягає у створенні формальної математичної моделі, яка б відображала динаміку процесу навчання, взаємозв'язки між студентами, викладачами та навчальними матеріалами, а також могла б бути використана для аналізу, прогнозування й оптимізації навчального процесу.

Визначення основних елементів моделі.

Математична модель навчального процесу має базуватися на ключових елементах, що характеризують цей процес. До таких елементів належать:

Студент – суб'єкт навчання, який отримує знання, розвиває навички та компетенції. Кожен студент може характеризуватися індивідуальними параметрами, такими як початковий рівень знань, швидкість засвоєння матеріалу, мотивація, успішність, кількість зусиль, витрачених на навчання, тощо.

Викладач – суб'єкт, який керує процесом навчання, надає навчальні матеріали, проводить заняття та здійснює контроль за прогресом студентів. Його роль полягає в організації навчального процесу та наданні зворотного зв’язку студентам щодо їх успішності.

Навчальні матеріали та завдання – об'єкти, які використовуються для передачі знань студентам. Це можуть бути лекції, підручники, тести, лабораторні роботи тощо. Навчальні матеріали мають різні рівні складності, що може впливати на швидкість та якість їх засвоєння.

Час – один із важливих параметрів моделі, оскільки процес навчання розгортається в часі. Час може впливати як на ефективність засвоєння знань, так і на адаптивність навчального процесу.

Оцінки та зворотний зв'язок – кількісні показники, що відображають рівень засвоєння матеріалу студентами. Вони можуть бути використані для корекції навчального процесу, підвищення мотивації студентів або зміни навчальної стратегії.

Формалізація процесу навчання.

Процес навчання можна описати як систему, в якій відбувається безперервний обмін інформацією між студентами та викладачами через навчальні матеріали, завдання та оцінювання. Математична модель має формалізувати цей процес, використовуючи різні змінні та параметри, що описують взаємодію між основними елементами системи.

Одним із підходів до моделювання процесу навчання є розгляд його як *динамічної системи*, де знання студента змінюються з часом під впливом навчальних дій (лекцій, завдань, тестів тощо). Це можна виразити у вигляді диференціальних рівнянь або систем різницевих рівнянь для дискретного часу, що відображатимуть залежність між рівнем знань студента та часом:

$$\frac{dz(t)}{dt} = f(z(t), u(t), p(t)), \qquad (1.1)$$

де z(t),) – рівень знань студента на момент часу t;

u(t) – інтенсивність навчальних дій у момент часу t (кількість годин, витрачених на навчання, обсяг матеріалу тощо);

p(t)– інші параметри (наприклад, мотивація студента, складність завдань тощо);

f(z,u,p) – функція, яка описує динаміку процесу навчання, що враховує вплив навчальних дій та параметрів на рівень знань.

Визначення цільової функції.

В процесі математичного моделювання важливо визначити цільову функцію, яка відображає мету навчального процесу. Як правило, метою є максимізація рівня знань студентів при мінімальних затратах часу та ресурсів. Цільова функція може бути виражена наступним чином

$$\max \int_0^T z(t)dt, \qquad (1.2)$$

де z(t), – рівень знань студента на момент часу t,

T – тривалість навчального періоду.

При цьому можуть накладатися певні обмеження, наприклад, на кількість часу, яку студент може присвячувати навчанню, або на кількість доступних навчальних матеріалів. Це дозволяє побудувати модель, яка оптимізує навчальний процес відповідно до реальних умов.

Підходи до моделювання взаємодії елементів системи.

Окрім формалізації основних елементів, важливо врахувати взаємодію між ними. Наприклад, взаємодія студента з викладачем може бути описана через моделі зворотного зв'язку. Викладач, отримуючи інформацію про успішність студента, може коригувати навчальні завдання або адаптувати методику викладання для покращення результатів. Модель такої взаємодії може бути представлена як система рівнянь з керуванням, де викладач виступає регулятором процесу

$$u(t) = g(z(t), feedback) \qquad (1.3)$$

де g – функція, що описує вплив зворотного зв’язку на вибір викладачем стратегії навчання.

Стохастичний підхід до моделювання.

Окрім детермінованих моделей, важливо також враховувати випадкові фактори, що можуть впливати на процес навчання, такі як стрес, мотивація або непередбачувані зовнішні обставини. Для цього застосовується стохастичне моделювання, де процес навчання розглядається як випадковий процес

$$\frac{dz(t)}{dt} = f(z(t)), u(t)) + \delta W(t) \qquad (1.4)$$

де W(t) – випадковий процес (наприклад, білий шум);

σ\sigmaσ – коефіцієнт, що визначає інтенсивність випадкових впливів.

Таке моделювання дозволяє отримати більш реалістичні результати та врахувати невизначеність, яка завжди присутня в реальних умовах навчання.

Моделювання мотивації та продуктивності студентів.

Мотивація є ключовим фактором, що впливає на продуктивність навчання. В рамках математичної моделі можна описати мотивацію як змінну, що залежить від різних факторів, таких як складність завдань, отримані оцінки, взаємодія з викладачем, тощо. Мотивація студента може бути включена в модель у вигляді функції, що впливає на рівень засвоєння знань

$$z(t) = h(m(t), u(t), p(t))$$

(1.5)

де m(t) – рівень мотивації в момент часу t, а h – функція, що визначає залежність рівня знань від мотивації.

1.8 Створення математичної моделі процесу аналізу роботи студентів

Створення математичної моделі аналізу роботи студентів передбачає формалізацію різних аспектів навчального процесу, таких як оцінка знань, відстеження успішності та взаємодія студентів з навчальними матеріалами. Метою побудови такої моделі є оптимізація процесу навчання через автоматизований аналіз результатів та надання викладачам інструментів для моніторингу й корекції навчальних програм.

1. Моделювання основних параметрів навчального процесу

Процес навчання студентів можна описати через набір параметрів, що характеризують їх роботу:

- Zi(t) — рівень знань студента і у момент часу t;
- Ui(t)— інтенсивність навчання студента і (час, витрачений на навчання, активність на заняттях, виконання домашніх завдань);
- Ai(t) — оцінка студента і в результаті виконання завдань або тестування;

- Mi(t) — мотивація студента і в момент часу t, що впливає на його продуктивність.

У цій системі часу навчання надається вирішальне значення, оскільки параметри знань змінюються залежно від інтенсивності навчального процесу та рівня залученості студента.

2. Визначення функції знань

Функція знань студента Zi(t) відображає зміну рівня засвоєння матеріалу з часом. Вона залежить від кількох факторів:

- u(t) — інтенсивність навчання (кількість витраченого часу на навчання);
- p(t)— складність матеріалу;
- r(t) — рівень мотивації студента.

Опис знань студента можна представити через рівняння динаміки засвоєння знань

$$\frac{dZi(t)}{dt} = f(Ui(t), P(t), R(t)) \qquad (1.6)$$

де f – функція, що описує зміну рівня знань у залежності від вхідних параметрів (інтенсивності навчання, складності матеріалу та мотивації);

Ui(t) інтенсивність навчання студента і у момент часу t;

P(t) – складність матеріалу, що впливає на швидкість засвоєння знань;

R(t) – мотиваційні фактори, що коригують швидкість навчання.

3. Модель оцінювання роботи студентів.

Процес оцінювання студентів є невід'ємною частиною аналізу їхньої роботи. Оцінка Ai(t) визначається через функцію, що залежить від поточного рівня знань студента Zi(t), якості виконання завдань і активності на заняттях

$$\mathrm{Ai(t)} = \mathrm{g(Zi(t), Qi(t), ai(t),} \qquad (1.7)$$

де Qi(t)– якість виконання завдань студентом і;

αi(t) – участь студента в заняттях (активність, відвідуваність тощо).

Оцінка може бути побудована на базі різних підходів: за допомогою тестування, оцінювання виконаних робіт або шляхом спостереження за участю студента в навчальному процесі.

4. Оптимізація навчального процесу.

Один з ключових аспектів аналізу роботи студентів — це оптимізація навчального процесу для підвищення ефективності засвоєння матеріалу. Для цього модель повинна включати механізми зворотного зв'язку, що дозволяють викладачам коригувати навчальні завдання та інтенсивність навчання на основі отриманих результатів.

Процес оптимізації можна описати через систему рівнянь з керуванням:

$$\mathrm{Ui(t+1)} = \mathrm{Ui(t)} + \Delta\mathrm{U(Zi(t), Ai(t), ai(t)} \qquad (1.8)$$

де ΔU, — зміна інтенсивності навчання в залежності від рівня знань Zi(t), оцінок Ai(t) та активності αi(t).

Таким чином, система автоматично коригує інтенсивність навчання на основі результатів поточного аналізу.

5. Застосування машинного навчання для прогнозування результатів.

Для прогнозування подальших результатів роботи студентів може бути використане машинне навчання. Алгоритми, такі як лінійна регресія, дерева рішень або методи класифікації, дозволяють передбачити майбутні оцінки студентів на основі наявних даних про їхню поточну успішність, мотивацію та активність.

Основне рівняння для прогнозування оцінки Ai(t+1)на основі попередніх даних виглядає так

$$\mathrm{Ai(t+1)} = \mathrm{h(Zi(t), Ui(t), Mi(t))}, \qquad (1.9)$$

де h – функція, що навчається алгоритмом машинного навчання для прогнозування оцінок. Це рівняння дозволяє визначити, як зміни в рівні знань і мотивації впливатимуть на підсумкові результати студента.

6. Моделювання взаємодії студентів із навчальними матеріалами.

Навчальні матеріали є важливим чинником успішності студентів. Складність завдань P(t) і їхній відповідний вплив на засвоєння знань можуть бути описані через таку модель

$$\mathrm{Zi(t+1)} = \mathrm{Zi(t)} + \mathrm{f(Ui(t), P(t)} \qquad (1.10)$$

де f(Ui(t), P(t) – функція, що визначає, як складність матеріалів P(t) впливає на рівень знань студентів при певній інтенсивності навчання Ui(t).

7. Аналіз відхилень і побудова рекомендацій.

Одним із ключових завдань математичної моделі є виявлення відхилень у навчальній діяльності студентів, що дозволить викладачам або системі автоматично надавати індивідуальні рекомендації. Наприклад, у випадках, коли студент не досягає заданого рівня знань Zi(t) система може запропонувати додаткові навчальні матеріали або зміни в стратегії навчання.

Модель зворотного зв'язку може бути представлена у вигляді наступного рівняння

$$\Delta \mathrm{Ui(t)} = \mathrm{k(Ai(t) - A_{min})} \qquad (1.11)$$

де A_{min} − мінімальна допустима оцінка,

k - коефіцієнт корекції інтенсивності навчання.

8. Оцінка результатів та ефективність моделі.

Ефективність математичної моделі для аналізу роботи студентів оцінюється на основі результатів тестування та реальних показників успішності студентів. Для цього проводиться порівняння прогнозованих результатів з фактичними оцінками. Це дозволяє оцінити точність моделі і внести необхідні корективи для покращення алгоритмів прогнозування та оптимізації навчального процесу.

Ефективність моделі можна оцінювати за допомогою кількох критеріїв:

Точність прогнозування. Основним критерієм є порівняння фактичних результатів (оцінок студентів) з прогнозованими значеннями, отриманими за допомогою моделі. Для цього використовують різні метрики похибок, такі як середньоквадратична похибка (MSE), середня абсолютна похибка (MAE) та коефіцієнт детермінації (R^2).

Чим менша похибка, тим краща модель виконує свої функції прогнозування, а отже, ефективність математичної моделі є високою.

Адаптивність моделі. Важливим критерієм є здатність моделі адаптуватися до нових даних і змін у навчальному процесі. Ефективна модель повинна вміти

коригувати свої прогнози, коли змінюються умови навчання, наприклад, при введенні нових навчальних матеріалів, зміні складності завдань або зміні навчальних методик.

Швидкість обчислень. Для великих навчальних груп або великих обсягів даних важливо, щоб модель швидко обробляла вхідні дані й надавала результати у режимі реального часу. Висока швидкість обчислень дозволяє проводити аналіз ефективності навчання та корекцію завдань без затримок, що підвищує загальну якість навчального процесу.

Гнучкість і масштабованість. Модель повинна бути гнучкою й здатною обробляти різні типи даних про студентів — оцінки, активність, кількість годин *навчання*, виконання завдань. Масштабованість системи дозволяє використовувати модель для різної кількості студентів та курсів, зокрема у великих університетах або навчальних платформах.

Зворотний зв'язок та інтерактивність. Модель повинна надавати викладачам і студентам інформативний зворотний зв'язок. Викладачам — для аналізу поточних успіхів студентів і корекції навчального процесу, студентам — для відстеження власних досягнень і підвищення мотивації. Це може бути реалізовано через інтерактивні дашборди та автоматизовані рекомендації.

Інтеграція з іншими системами. Модель повинна бути інтегрована з іншими системами управління навчанням LMS або базами даних для автоматичного збирання даних про студентів, а також для оновлення навчальних *матеріалів* та завдань у режимі реального часу. Це дозволяє автоматизувати процеси аналізу й мінімізувати ручне введення інформації.

*Валідація та тестування моделі***.**

Для забезпечення високої точності та надійності математичної моделі важливими є *етапи* валідації та тестування. Модель має бути перевірена на реальних даних, що відображають успішність студентів у різних ситуаціях. Для цього використовують методи:

Крос-валідація – у цьому методі дані розділяються на кілька підмножин, і модель навчається на одній частині даних, а тестується на іншій. Це дозволяє

визначити, наскільки добре модель узагальнює результати і не переобучається на конкретних даних.

Тестування на різних навчальних курсах. Модель має бути перевірена на різних предметах і курсах для того, щоб оцінити її ефективність у різних контекстах. Наприклад, аналіз роботи студентів на технічних курсах може відрізнятися від аналізу на гуманітарних дисциплінах через різну складність матеріалу та структуру завдань.

Перевірка на різних рівнях складності. Модель має бути протестована на групах студентів із різним рівнем підготовки — початковим, середнім і високим. Це дозволяє оцінити її універсальність і здатність точно прогнозувати результати незалежно від рівня знань студентів на початку навчання.

Поліпшення моделі.

Після проведення тестування та валідації моделі можуть бути внесені необхідні корективи для підвищення її точності й продуктивності. Це може бути досягнуто шляхом:

Уточнення функцій і параметрів. Якщо під час тестування буде виявлено, що певні змінні або параметри (наприклад, мотивація або якість виконання завдань) суттєво впливають на результати, модель може бути уточнена шляхом введення додаткових змінних або коригування існуючих.

Використання більш точних методів машинного навчання. Замість простих алгоритмів, таких як лінійна регресія, можуть бути використані більш складні алгоритми, такі як градієнтний бустинг або нейронні мережі, які забезпечують більш точне прогнозування і кращу адаптацію до даних.

Аналіз аномалій і виняткових ситуацій. Модель може бути поліпшена через введення спеціальних механізмів для обробки аномальних ситуацій, таких як різкі зміни мотивації або раптове зниження результатів. Це допоможе вчасно реагувати на незвичайні обставини та коригувати навчальний процес у реальному часі.

1.9 Алгоритми керування процесом навчання на основі математичної моделі

Алгоритми керування процесом навчання є ключовими елементами для оптимізації роботи студентів і підвищення ефективності навчального процесу. Використовуючи математичну модель, розробляються алгоритми, які дозволяють автоматично регулювати інтенсивність навчання, адаптувати навчальні матеріали та надавати індивідуальні рекомендації для студентів. Основною метою цих алгоритмів є досягнення балансу між рівнем знань студентів та складністю навчальних завдань, щоб покращити засвоєння матеріалу та результативність навчання.

1. Основні етапи керування процесом навчання

Алгоритми керування можуть бути поділені на кілька основних етапів:

Збір даних про навчальний процес. На першому етапі відбувається збір інформації про рівень знань студентів, їхню активність у навчальному процесі, оцінки та інші показники, які впливають на навчання.

Аналіз зібраних даних. На цьому етапі математична модель аналізує отримані дані, визначає поточний рівень знань студентів і прогнозує їхній майбутній розвиток. Аналіз також включає виявлення відхилень у результатах, визначення сильних і слабких сторін студентів.

Корекція процесу навчання. На основі результатів аналізу відбувається корекція навчального процесу — зміна складності завдань, коригування інтенсивності навчання, надання додаткових матеріалів для покращення розуміння.

Оцінка результатів і адаптація. Після впровадження змін система оцінює їхній вплив на навчальний процес і за необхідності вносить подальші корективи.

2. Алгоритми адаптивного навчання

Алгоритми адаптивного навчання є важливою складовою керування процесом. Вони дозволяють автоматично підлаштовувати навчальні матеріали

та завдання під індивідуальні потреби студентів. Основними компонентами адаптивних алгоритмів є:

Адаптація рівня складності матеріалів. Складність завдань регулюється в залежності від рівня знань студента. Наприклад, якщо студент демонструє високі результати, система підвищує складність завдань, надаючи йому можливість заглибитися в матеріал. Якщо ж рівень знань нижчий, система спрощує завдання та надає додаткові пояснення або ресурси для засвоєння базових понять.

Індивідуальні рекомендації. Алгоритми можуть надавати студентам персоналізовані рекомендації щодо навчальних ресурсів, що допомагають заповнити прогалини у знаннях або закріпити матеріал. Ці рекомендації можуть включати додаткові вправи, лекції чи відео для перегляду.

Регулювання темпу навчання. Студенти з різним рівнем підготовки можуть потребувати різного часу для засвоєння матеріалу. Алгоритми адаптивного навчання враховують ці особливості та регулюють темп проходження матеріалу, дозволяючи кожному студенту вчитися у своєму ритмі.

3. Алгоритми зворотного зв'язку та корекції.

Алгоритми зворотного зв'язку забезпечують викладачам можливість оперативно реагувати на прогрес студентів. На основі даних моделі аналізу роботи студентів, система може надавати рекомендації викладачам про те, які аспекти потребують уваги. До таких алгоритмів відносяться:

Аналіз відхилень від очікуваних результатів. Якщо система виявляє, що студент відстає від очікуваних результатів, алгоритм може надати рекомендації щодо додаткових занять, повторення матеріалу або індивідуальних консультацій з викладачем.

Корекція інтенсивності навчання. Алгоритм може автоматично регулювати інтенсивність навчання студента, пропонуючи більш складні завдання або зменшуючи їх кількість у разі перевантаження студента. Це допомагає уникнути втрати мотивації та перевтоми.

4. Алгоритми прогнозування результатів

Прогнозування результатів є важливою частиною керування навчальним процесом. Алгоритми машинного навчання використовуються для прогнозування підсумкових результатів на основі даних про поточну активність студентів, їхню успішність та рівень мотивації. Основні етапи прогнозування:

Моделі лінійної регресії. Лінійна регресія дозволяє оцінити, як зміна інтенсивності навчання або активності студента впливатиме на його кінцеву оцінку. Формула прогнозування може виглядати так:

$$Ai(t+1) = \beta_0\_\beta_1 Zi(t) + \beta_2 Ui(t) \qquad (1.12)$$

де Ai(t+1) – прогнозована оцінка студента, Zi(t) – рівень знань студента на момент часу t, Ui(t),– інтенсивність навчання.

Нейронні мережі. Більш складні методи, такі як нейронні мережі, можуть використовуватися для виявлення прихованих закономірностей у даних та прогнозування результатів на основі великої кількості змінних. Нейронні мережі можуть враховувати не тільки успішність і активність студентів, але й їхні індивідуальні характеристики, такі як мотивація або стиль навчання.

5. Оптимізаційні алгоритми.

Оптимізаційні алгоритми допомагають знайти найкращі параметри для керування процесом навчання. Вони використовуються для того, щоб автоматично визначати оптимальні значення параметрів навчального процесу, такі як тривалість навчання, обсяг завдань або складність матеріалу. Основні методи оптимізації:

Генетичні алгоритми. Генетичні алгоритми дозволяють знайти оптимальні параметри навчання шляхом еволюційного процесу, в якому відбираються найбільш успішні варіанти навчальних стратегій.

Алгоритми градієнтного спуску. Використовуються для мінімізації функції помилки прогнозу оцінок або рівня знань студентів. Алгоритм шукає мінімум функції втрат, коригуючи параметри навчального процесу, такі як інтенсивність або обсяг навчальних матеріалів.

6. Алгоритми підтримки мотивації.

Алгоритми підтримки мотивації студентів спрямовані на забезпечення того, щоб навчальний процес був не тільки ефективним, але й мотивуючим. Для цього використовуються алгоритми, які аналізують мотиваційні фактори та надають рекомендації для підтримання зацікавленості студентів:

Мотиваційні підказки. Алгоритми можуть надсилати студентам підказки або нагадування, коли їх активність знижується, наприклад, нагадування про майбутні дедлайни або поради щодо управління часом.

Система заохочень. Алгоритми можуть впроваджувати систему заохочень, що стимулює студентів до активної роботи, наприклад, надання бонусів за своєчасне виконання завдань або активну участь у заняттях.

Алгоритми підтримки мотивації можуть включати не лише підказки й заохочення, а й більш складні механізми, що враховують індивідуальні потреби та психологічні особливості студентів. До таких механізмів належать:

1. Адаптивне навантаження

Алгоритм може автоматично регулювати кількість та складність завдань на основі поточної продуктивності студента. Якщо студент демонструє високу успішність та активно бере участь у навчальному процесі, система може поступово збільшувати складність завдань для стимулювання подальшого розвитку. Якщо ж студент показує зниження результатів або втрачає мотивацію, завдання можуть бути спрощені для підтримки рівня залученості й запобігання демотивації.

2. Гейміфікація.

Гейміфікація — це потужний інструмент для підвищення мотивації. Алгоритми можуть впроваджувати елементи гри у навчальний процес, такі як бали, рівні, нагороди за досягнення. Студенти можуть змагатися один з одним або досягати певних "рівнів успішності", що робить процес навчання більш цікавим і залучає їх до активної участі.

3. Індивідуальні навчальні траєкторії.

Алгоритми можуть створювати індивідуальні навчальні плани для кожного студента, враховуючи їхні сильні та слабкі сторони. Такий підхід не тільки

підвищує ефективність навчання, але й дозволяє студентам відчувати себе більш впевненими та мотивованими, оскільки навчання стає для них особисто значущим.

4. Соціальна взаємодія.

Мотивація студентів може бути значно підвищена через соціальні фактори, такі як підтримка з боку однокурсників чи викладачів. Алгоритми можуть стимулювати взаємодію між студентами, наприклад, через спільні проекти або обговорення у навчальних групах. Це створює атмосферу співпраці та підтримки, що сприяє активнішому залученню до навчання.

5. Візуалізація прогресу.

Алгоритми можуть забезпечувати візуалізацію прогресу студента у вигляді графіків, діаграм або дашбордів, які демонструють досягнення і допомагають стежити за прогресом. Така візуалізація робить результати навчання більш очевидними та мотивує студентів продовжувати навчання.

6. Психологічна підтримка

Алгоритми можуть включати елементи психологічної підтримки, наприклад, надсилати студентам мотивуючі повідомлення або поради щодо управління стресом під час підготовки до іспитів. Така підтримка може бути особливо важливою для студентів, які відчувають емоційне виснаження або тиск через навчальні навантаження.

7. Аналіз мотиваційних тригерів

Алгоритми можуть аналізувати поведінкові та емоційні дані студентів для виявлення мотиваційних тригерів. Наприклад, система може виявити, що студенту потрібна додаткова мотивація в певний час дня або під час виконання складних завдань. На основі цих даних алгоритм може пропонувати індивідуальні мотиваційні заходи, такі як короткі перерви, відпочинок або зміна типу завдань для збереження концентрації та інтересу.

1.10 Оцінка ефективності синтезованого керування

Оцінка ефективності синтезованого керування процесом навчання є важливим етапом для перевірки якості розробленої системи та її впливу на навчальний процес. Цей етап дозволяє виявити, наскільки алгоритми керування на основі математичної моделі досягають поставлених цілей, а також чи відповідають вони очікуванням щодо підвищення успішності студентів та ефективності навчального процесу.

1. Критерії оцінки ефективності

Ефективність синтезованого керування можна оцінювати за кількома основними критеріями:

Підвищення успішності студентів. Основним показником ефективності є рівень покращення успішності студентів. Це може бути оцінено шляхом порівняння середнього балу студентів до та після впровадження системи. Аналіз змін успішності дозволяє зробити висновки про якість адаптації навчального процесу та відповідність рівня складності навчальних матеріалів знанням студентів.

Зниження кількості відстаючих студентів. Важливим критерієм є зменшення кількості студентів, які демонструють низьку успішність або відстають у навчанні. Якщо система керування ефективно адаптує завдання під індивідуальні потреби, кількість таких студентів має суттєво знизитися.

Зменшення кількості повторних спроб здачі завдань. Система, що коригує навчальний процес у режимі реального часу, повинна сприяти зменшенню випадків, коли студенти не можуть виконати завдання з першої спроби. Це свідчить про оптимальність рівня складності та ефективність адаптації навчального матеріалу.

Покращення мотивації студентів. Оцінка ефективності також може включати аналіз мотивації студентів. Якщо система керування забезпечує підтримку мотивації, кількість студентів, які активно залучені до навчального процесу, має зрости, а також зменшитися кількість тих, хто втрачає інтерес до навчання.

Зменшення витрат часу на навчання. Оптимізація навчального процесу має сприяти більш ефективному використанню часу як студентами, так і викладачами. Якщо система допомагає студентам швидше засвоювати матеріал і виконувати завдання, це свідчить про її ефективність.

2. Методологія оцінки

Для оцінки ефективності керування можна застосовувати такі методи:

Статистичний аналіз. Дані про успішність студентів, їхню активність у навчальному процесі та інші показники можуть бути зібрані й проаналізовані з використанням статистичних методів. Це дозволяє отримати об'єктивну картину впливу системи керування на навчальний процес. Показники, як-от середні бали, відсоток успішно виконаних завдань, темпи зростання знань студентів, можуть бути основою для оцінки.

Аналіз відхилень. Порівняння очікуваних результатів із фактичними може дати уявлення про ефективність моделі та алгоритмів керування. Якщо прогнозовані результати збігаються з реальними або навіть перевищують їх, це свідчить про те, що модель ефективно виконує свої функції.

Експериментальний підхід. Для більш точного визначення ефективності системи можна провести експерименти, де частина студентів буде навчатися із застосуванням алгоритмів керування, а інша — за традиційною методикою. Порівняння результатів дозволить визначити, наскільки ефективною є синтезована система в реальних умовах.

3. Визначення ключових показників ефективності (KPI)

Ключові показники ефективності (KPI) допомагають кількісно оцінити ефективність системи керування. До основних KPI, які можуть бути використані для оцінки, належать:

Рівень успішності студентів: середній бал та відсоток виконаних завдань.

Рівень залученості: кількість активних учасників навчального процесу, відвідуваність лекцій, участь у дискусіях.

Час виконання завдань: середній час, необхідний студентам для виконання завдань.

Кількість повторних спроб: кількість спроб виконання завдань з першого разу порівняно з повторними спробами.

Кількість відстаючих студентів: відсоток студентів, які не досягають мінімального порогу успішності.

4. Аналіз недоліків та шляхів їх усунення

Оцінка ефективності також передбачає аналіз недоліків системи та визначення можливостей для її вдосконалення. Серед можливих проблем, які можуть бути виявлені під час оцінки:

Неправильна адаптація до індивідуальних потреб студентів. Якщо система занадто уніфікована або не враховує всі особливості кожного студента, це може призвести до недосягнення цілей навчання.

Надмірна складність або спрощеність навчальних завдань. Якщо алгоритми неправильно оцінюють рівень підготовки студентів, це може призвести до демотивації та погіршення успішності.

Низька мотивація викладачів. Якщо викладачі не інтегрують алгоритми в процес, ефективність системи може бути знижена.

Виявлені недоліки дозволяють вносити необхідні зміни в алгоритми керування, покращувати процес адаптації та робити систему більш ефективною.

Окрім визначення основних показників ефективності та аналізу недоліків, подальший розвиток та вдосконалення алгоритмів синтезованого керування навчальним процесом можуть базуватися на таких важливих аспектах:

Зворотній зв'язок від користувачів. Одним з ключових елементів успішного функціонування системи керування є отримання зворотного зв'язку від користувачів — як від студентів, так і від викладачів. Відгуки можуть стосуватися складності завдань, зручності користування системою, її здатності мотивувати та підтримувати зацікавленість. На основі таких відгуків можна адаптувати інтерфейс та функціонал системи, покращуючи користувацький досвід та підвищуючи ефективність керування навчальним процесом.

*Моніторинг та коригування***.** Для того, щоб система залишалася ефективною протягом усього навчального періоду, важливо проводити регулярний моніторинг її роботи. Це включає аналіз того, як часто студенти користуються системою, які показники успішності вони демонструють та наскільки точно система виконує функції адаптації та керування навчальним процесом. У разі виявлення збоїв або зниження ефективності можуть бути впроваджені коригуючі заходи для оптимізації роботи алгоритмів.

Інтеграція з іншими системами. Для підвищення ефективності синтезованого керування навчальним процесом можлива інтеграція з іншими інформаційними системами, які використовуються в навчальних закладах. Наприклад, інтеграція з системами управління контентом (LMS), інструментами для проведення онлайн-іспитів, базами даних оцінок та іншими ресурсами може значно покращити якість керування та зробити його більш комплексним і ефективним.

Автоматичне вдосконалення алгоритмів. Алгоритми керування процесом навчання можуть використовувати методи машинного навчання для автоматичного вдосконалення з часом. На основі даних про продуктивність студентів система може навчатися та краще адаптувати свої стратегії керування до індивідуальних потреб. Такий підхід дозволяє не тільки автоматично вносити корективи, але й розвивати систему у відповідь на зміни в навчальному процесі, забезпечуючи довгострокову ефективність.

Довгострокові показники ефективності. Для отримання більш повної картини ефективності системи необхідно аналізувати її вплив у довгостроковій перспективі. Наприклад, можна дослідити, як впровадження синтезованих алгоритмів керування впливає на загальний рівень знань студентів, їхню здатність застосовувати отримані знання на практиці, а також їхню кар’єрну успішність після закінчення навчання.

Такі довгострокові дослідження можуть надати цінні дані для подальшого вдосконалення системи та підтвердити її корисність не тільки в межах навчального закладу, а й на етапах професійної підготовки випускників.

Висновок: У процесі аналізу підходів до підвищення ефективності навчання студентів в ході навчального процесу було встановлено, що ключовими факторами, які впливають на результативність навчання, є адаптація навчальних методик до індивідуальних потреб студентів, впровадження інтерактивних і цифрових технологій, а також створення сприятливого середовища для розвитку критичного мислення та самостійної роботи.

Розглянуті підходи, такі як використання проектного навчання, гейміфікації, персоналізації освітніх траєкторій та активне застосування сучасних інформаційних технологій, дозволяють значно підвищити рівень зацікавленості студентів, їхню мотивацію та результативність засвоєння матеріалу.

Важливим аспектом є інтеграція зворотного зв’язку у навчальний процес, що забезпечує можливість своєчасного коригування методик і форм навчання. Крім того, значна увага повинна приділятися розвитку м'яких навичок (soft skills) у студентів, що є необхідною умовою їхньої подальшої професійної реалізації.

Таким чином, для досягнення високої ефективності навчального процесу необхідне поєднання традиційних і сучасних підходів, із акцентом на активну участь студентів та гнучкість освітніх програм.

РОЗДІЛ 2
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕНЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ РОБОТИ СТУДЕНТІВ В УЧБОВОМУ ПРОЦЕСІ

2.1 Структурна схема системи аналізу роботи студентів

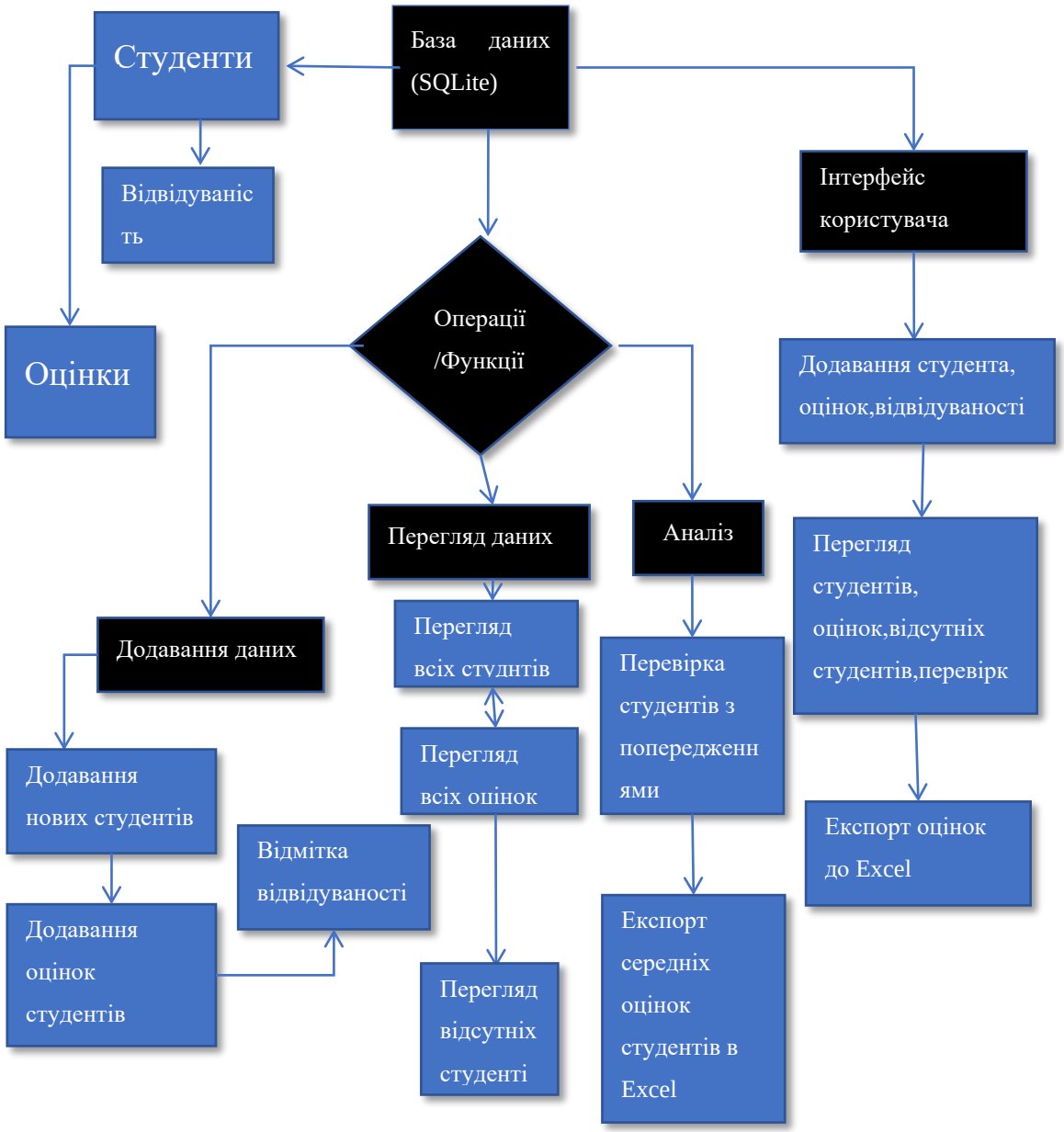

Рисунок 2.1 – Структурна схема системи аналізу роботи студентів та даної програми

На зображенні представлена структурна схема системи аналізу роботи студентів, яка ілюструє ключові компоненти системи, їх функції та взаємозв'язки між ними. Ця схема демонструє логічний порядок дій, починаючи від отримання та зберігання даних про студентів до їх подальшого аналізу та взаємодії з користувачем через інтерфейс.

Основні компоненти схеми:

1. *База даних (БД)* – це центральний елемент системи, що відповідає за зберігання всієї інформації про студентів. Вона містить усі дані, пов'язані з їх навчальною діяльністю, включаючи оцінки, відвідуваність, загальні академічні показники тощо. База даних слугує фундаментом для всіх операцій, які виконуються в системі.

*Від бази даних відходять три основні напрямки***:**

2. *Оцінка* – цей блок відповідає за управління оцінками студентів. Він дозволяє перевіряти успішність кожного студента, аналізуючи їхні оцінки, які зберігаються в базі даних. У цьому контексті система також може здійснювати порівняння результатів студентів, відображати тенденції в їх навчанні та допомагати викладачам у визначенні прогалин у знаннях студентів.

3. *Відвідуваність* – цей компонент забезпечує відстеження присутності студентів на заняттях. Інформація про відвідуваність також зберігається в базі даних і використовується для аналізу активності студентів, їх участі в навчальному процесі та впливу на загальний рівень успішності.

4. *Інтерфейс користувача* – це частина, через яку користувачі взаємодіють із системою. Інтерфейс дозволяє додавати нові записи, редагувати існуючі дані, здійснювати пошук інформації та формувати звіти. Він є важливим інструментом для викладачів і адміністраторів, які працюють з системою.

5. *Операції (функції)* – цей блок представляє основні дії, які можуть виконуватися з даними. Він об'єднує всі функції системи, які стосуються роботи з інформацією про студентів. До основних функцій відносяться:

6. *Додавання даних*: введення інформації про студентів, їхні оцінки, відвідуваність та інші важливі показники. Це початковий етап, на якому інформація надходить до системи.

7. *Видалення даних*: можливість видаляти або коригувати помилкові чи застарілі записи, які більше не потрібні.

8. *Перетворення даних*: операції, що стосуються перетворення інформації, наприклад, зміна форматів даних для подальшого аналізу або передачі.

9. *Перевірка коректності*: контроль за правильністю введених даних, наприклад, перевірка наявності всіх необхідних оцінок чи інформації про відвідування.

10. *Аналіз*: цей процес передбачає глибше дослідження отриманих даних, їх оцінку, порівняння та узагальнення результатів. На основі цього аналізу можуть створюватися звіти про успішність студентів, їх відвідуваність, активність та інші параметри, важливі для моніторингу навчального процесу.

11. *Інтерфейс користувача* – це важлива частина системи, яка забезпечує зв'язок між користувачем та системою. Він дозволяє працювати з даними за допомогою зручного інструменту, який надає можливість:

12. *Додавання нових записів*: користувачі можуть додавати інформацію про нових студентів, їхні оцінки, результати іспитів, рівень відвідуваності та інші показники.

13. *Управління даними*: редагування вже існуючих даних, їх оновлення або видалення непотрібної інформації.

14. *Перетворення даних*: зміна даних для їх подальшого використання або передачі до інших модулів системи.

15. Передача даних в базу: система інтегрує нові або оновлені дані до бази даних, де вони зберігаються для подальшого аналізу.

Загальна структура системи:

Схема показує, як інформація про студентів надходить до системи (через інтерфейс користувача або введення даних вручну), як ці дані зберігаються, обробляються та аналізуються. Результати аналізу можуть бути представлені

в різних формах, таких як звіти або графіки, які допомагають користувачам оцінити успішність студентів або їх активність у навчальному процесі.

2.2 Алгоритм роботи системи аналізу роботи студентів

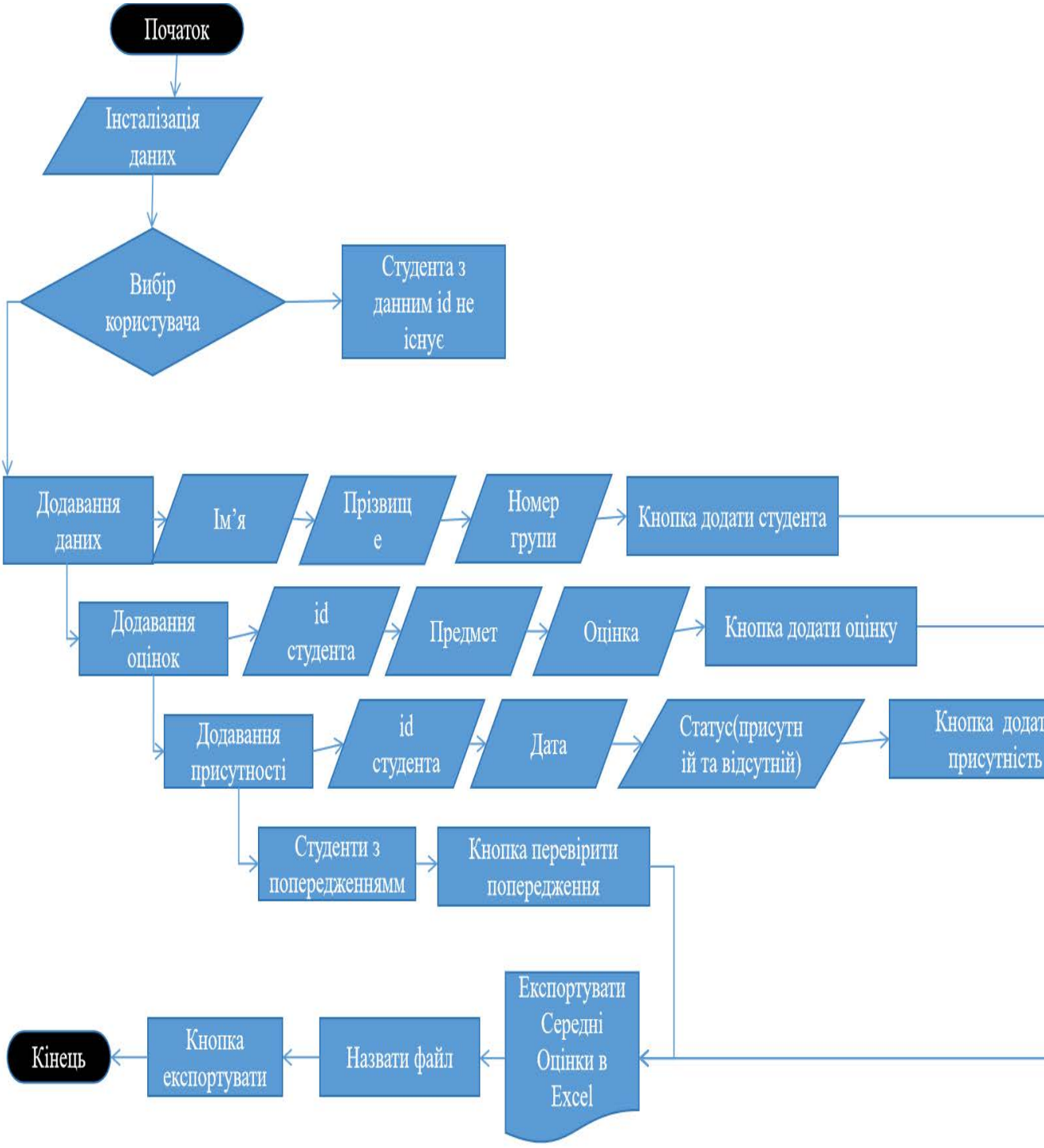

Рисунок 2.2 – Блок схема алгоритму роботи системи аналізу роботи студентів

1. Початок

- Це початковий етап роботи програми.

- Програма запускається, користувач бачить початковий екран або консоль, що вказує, що програма готова до роботи.

2. Ініціалізація бази даних

- На цьому етапі програма перевіряє та налаштовує середовище для зберігання даних:

- Якщо база даних вже існує (наприклад, файл із попередніми даними), програма завантажує ці дані в оперативну пам’ять.

- Якщо база даних відсутня, програма створює її:

- Генерує структури, таблиці, для зберігання даних про студентів, оцінки, присутність тощо.

- Переконується, що всі необхідні файли, бібліотеки або сервери доступні для роботи.

- Перевіряє можливість зчитування/запису даних у базу (виправлення помилок доступу).

- Це забезпечує готовність програми до подальших операцій.

3. Вибір дії користувача

- На цьому етапі користувач отримує меню з доступними діями:

- Додавання студента.

- Додавання оцінки.

- Додавання статусу присутності.

- Перегляд попереджень.

- Експорт даних до файлу.

- Перегляд таблиць.

- Завершення роботи програми.

- Користувач обирає одну з дій.

4. Гілка вибору дій

- *Якщо дія не обрана*:

- Якщо користувач не зробив жодного вибору або обрав некоректну дію, програма повертається до пункту "Вибір дії користувача" з повідомленням про необхідність зробити вибір.

- *Якщо обрана дія пов'язана зі студентами (наприклад, додавання оцінки чи присутності)*:

- Програма перевіряє, чи існує студент із вказаним ID у базі даних.

- *Якщо студента з таким ID не існує*:

- Програма видає повідомлення про помилку (наприклад, "Студента з цим ID не знайдено") та повертає користувача до вибору дій.

- *Якщо студент існує*:

- Програма переходить до виконання обраної дії.

5. Виконання обраної дії

- Додавання студента:

- Користувач вводить дані про студента (ім'я, прізвище, ID тощо).

- Програма перевіряє коректність введених даних (наприклад, унікальність ID, правильність формату).

- Дані зберігаються в базі.

- *Додавання оцінки*:

- Користувач обирає студента (вказує ID).

- Вводиться оцінка.

- Програма перевіряє коректність оцінки (наприклад, діапазон значень) і додає її до бази.

- *Додавання присутност***і**:

- Користувач обирає студента.

- Вводиться статус присутності (наприклад, "присутній", "відсутній").

- Програма оновлює базу даних відповідно до введених даних.

- *Перегляд попереджень*:

- Програма аналізує дані у базі, наприклад:

- Виявляє студентів із низькими середніми балами.

- Студентів із великою кількістю пропусків.

- Результати виводяться на екран.

- *Експорт даних*:

- Програма генерує файл (наприклад, у форматі Excel), який містить середні оцінки студентів, їхню активність або інші вибрані дані.

- Цей файл зберігається у вказаному місці.

- *Перегляд таблиць*:

- Програма виводить таблицю з даними на екран за запитом користувача.

6. Конвертація даних у файл (Експорт)

- Якщо користувач обирає експорт даних, програма:

- Зчитує всі необхідні дані з бази.

- Форматує ці дані у структуровану таблицю (наприклад, Excel).

- Зберігає файл на диску.

- Повідомляє користувача про успішне збереження.

7. Кінець

- На цьому етапі користувач завершує роботу програми.

- Програма може запропонувати зберегти незбережені дані, вийти чи перезапуститися.

- Виводиться повідомлення про завершення роботи.

Система аналізу роботи студентів розроблена для ефективного управління інформацією про студентів, їхні оцінки та присутність. Нижче наведено детальний опис алгоритму роботи системи, базуючись на представленому коді.

1. Ініціалізація та налаштування середовища

```
# Імпорт необхідних бібліотек
import pandas as pd
import sqlite3
from google.colab import files
import ipywidgets as widgets
from IPython.display import display, clear_output
```

Рисунок 2.2. - Імпорт бібліотек

Першим кроком у роботі системи є імпорт необхідних бібліотек та встановлення середовища:

Імпорт бібліотек:

- pandas для обробки даних.
- sqlite3 для роботи з базою даних SQLite.
- google.colab.files для завантаження файлів у Google Colab.
- ipywidgets для створення інтерактивного графічного інтерфейсу.
- IPython.display для відображення віджетів та очищення виводу.

2. Створення та налаштування бази даних

```
# Функція для створення таблиць
def create_tables():
    conn = sqlite3.connect('students.db')
    cursor = conn.cursor()
```

Рисунок 2.3. - Функція для створення таблиць

Система використовує базу даних SQLite для зберігання інформації про студентів, їхні оцінки та присутність. Функція create_tables() відповідає за створення необхідних таблиць у базі даних:

Таблиця students:

```
# Створення таблиці для студентів
cursor.execute('''
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    surname TEXT NOT NULL,
    group_number TEXT NOT NULL
)
''')
```

Рисунок 2.4. - Створення таблиці для студентів

Поля:

- id: Унікальний ідентифікатор студента (автоматично збільшується).
- name: Ім'я студента.
- surname: Прізвище студента.
- group_number: Номер групи, до якої належить студент.

Таблиця grades:

```
# Створення таблиці для оцінок
cursor.execute('''
CREATE TABLE IF NOT EXISTS grades (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER,
    subject TEXT NOT NULL,
    grade INTEGER NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students(id)
)
''')
```

Рисунок 2.5. - Створення таблиці для оцінок

Поля:

- id: Унікальний ідентифікатор оцінки.

- student_id: Ідентифікатор студента (посилання на таблицю students).

- subject: Назва предмету.

- grade: Оцінка студента за предмет.

Таблиця attendance:

```
# Створення таблиці для присутності
cursor.execute('''
CREATE TABLE IF NOT EXISTS attendance (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER,
    date TEXT NOT NULL,
    status TEXT NOT NULL,  -- "present" або "absent"
    FOREIGN KEY (student_id) REFERENCES students(id)
)
''')

conn.commit()
conn.close()
```

Рисунок 2.6. - Створення таблиці для присутності

Поля:

- id: Унікальний ідентифікатор запису присутності.

- student_id: Ідентифікатор студента (посилання на таблицю students).

- date: Дата проведення заняття.

- status: Статус присутності ("присутній" або "відсутній").

Функція викликається одразу після її визначення для створення таблиць, якщо вони ще не існують:

create_tables()

```
# Виклик функції для створення таблиць
create_tables()
```

Рисунок 2.7. - Виклик функції для створення таблиць

3. Функції для Управління Даними

Система містить кілька функцій для додавання та отримання даних з бази даних:

Додавання Студента (add_student):

```
# Функції для роботи з базою даних
def add_student(name, surname, group_number):
    conn = sqlite3.connect('students.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO students (name, surname, group_number) VALUES (?, ?, ?)",
                   (name, surname, group_number))
    conn.commit()
    conn.close()
```

Рисунок 2.8. - Функції для роботи з базою даних та add_student

1. Приймає ім'я, прізвище та номер групи студента.

2. Вставляє новий запис у таблицю students.

Додавання Оцінки (add_grade):

```
def add_grade(student_id, subject, grade):
    conn = sqlite3.connect('students.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO grades (student_id, subject, grade) VALUES (?, ?, ?)",
                   (student_id, subject, grade))
    conn.commit()
    conn.close()
```

Рисунок 2.9. - Функція add_grade

1. Приймає ідентифікатор студента, предмет та оцінку.

2. Вставляє новий запис у таблицю grades.

Додавання Присутності ***(****add_attendance****):***

```
def add_attendance(student_id, date, status):
    conn = sqlite3.connect('students.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO attendance (student_id, date, status) VALUES (?, ?, ?)",
                   (student_id, date, status))
    conn.commit()
    conn.close()
```

Рисунок 2.9. - Функція add_attendance

1. Приймає ідентифікатор студента, дату та статус присутності.
2. Вставляє новий запис у таблицю attendance.

Отримання всієї інформації про Студентів (get_students):

```
def get_students():
    conn = sqlite3.connect('students.db')
    df = pd.read_sql_query("SELECT * FROM students", conn)
    conn.close()
    return df
```

Рис.2.10. Функція get_students

1. Повертає DataFrame з усіма записами таблиці students.

Отримання всієї інформації про оцінки (get_grades):

```
def get_grades():
    conn = sqlite3.connect('students.db')
    df = pd.read_sql_query("SELECT * FROM grades", conn)
    conn.close()
    return df
```

Рисунок 2.11. - Функція get_grades

1. Повертає DataFrame з усіма записами таблиці grades.

Отримання Відсутніх Студентів (get_absent_students):

```
def get_absent_students():
    conn = sqlite3.connect('students.db')
    df = pd.read_sql_query("SELECT * FROM attendance WHERE status = 'absent'", conn)
    conn.close()
    return df
```

Рисунок 2.12. - Функція get_absent_students

1. Повертає DataFrame з усіма записами таблиці attendance, де статус "відсутній".

Перевірка Попереджень (check_warnings):

```
def check_warnings():
    conn = sqlite3.connect('students.db')
    query = """
    SELECT students.id, students.name, students.surname, COUNT(attendance.status) as absences
    FROM attendance
    JOIN students ON attendance.student_id = students.id
    WHERE attendance.status = 'absent'
    GROUP BY students.id
    HAVING absences >= 3  -- наприклад, 3 пропуски для попередження
    """
    df = pd.read_sql_query(query, conn)
    conn.close()
    return df
```

Рисунок 2.13. - Функція check_warnings

1. Аналізує дані про присутність та визначає студентів, які мають 3 або більше пропусків.

2. Повертає DataFrame з інформацією про таких студентів.

Збереження середніх оцінок у Excel (save_average_grades_to_excel):

```
def check_warnings():
    conn = sqlite3.connect('students.db')
    query = """
    SELECT students.id, students.name, students.surname, COUNT(attendance.status) as absences
    FROM attendance
    JOIN students ON attendance.student_id = students.id
    WHERE attendance.status = 'absent'
    GROUP BY students.id
    HAVING absences >= 3  -- наприклад, 3 пропуски для попередження
    """
    df = pd.read_sql_query(query, conn)
    conn.close()
    return df
```

Рисунок 2.14. - Функція save_average_grades_to_excel

1. Об'єднує дані з таблиць grades та students.

2. Обчислює середню оцінку кожного студента за всіма предметами.

3. Зберігає результат у вказаний Excel-файл та повертає шлях до файлу.

4. Інтерактивний графічний інтерфейс

Для покращення користувацького досвіду система використовує бібліотеку ipywidgets для створення інтерактивного графічного інтерфейсу. Інтерфейс організований у вигляді вкладок, кожна з яких відповідає певній функціональності:

Додати Студента

```
# 1. Додати Студента
add_student_title = widgets.HTML("<h2>Додати Студента</h2>")
student_name = widgets.Text(description="Ім'я:")
student_surname = widgets.Text(description="Прізвище:")
student_group = widgets.Text(description="Номер Групи:")
add_student_button = widgets.Button(description="Додати Студента", button_style='success')

def on_add_student_clicked(b):
    name = student_name.value.strip()
    surname = student_surname.value.strip()
    group = student_group.value.strip()

    if name and surname and group:
        add_student(name, surname, group)
        output_add_student.clear_output()
        with output_add_student:
            print(f"Студента {name} {surname} успішно додано.")
        # Очищення полів вводу
        student_name.value = ''
        student_surname.value = ''
        student_group.value = ''
    else:
        with output_add_student:
            print("Будь ласка, заповніть усі поля.")

add_student_button.on_click(on_add_student_clicked)
output_add_student = widgets.Output()

add_student_box = widgets.VBox([
    add_student_title,
    student_name,
    student_surname,
    student_group,
    add_student_button,
    output_add_student
])
```

Рисунок 2.15. - Функція додати студента

Компоненти:

- Заголовок.

- Текстові поля для введення імені, прізвища та номера групи.

- Кнопка для додавання студента.

- Область виводу для повідомлень.

Алгоритм дії:

1. Користувач вводить ім'я, прізвище та номер групи.

2. Натискає кнопку "Додати Студента".

3. Система перевіряє, чи всі поля заповнені.

4. Якщо так, додає запис у таблицю students та відображає повідомлення про успіх.

5. Якщо ні, відображає повідомлення про необхідність заповнення всіх полів.

Додати оцінку

```
# 2. Додати Оцінку
add_grade_title = widgets.HTML("<h2>Додати Оцінку</h2>")
grade_student_id = widgets.IntText(description="ID Студента:")
grade_subject = widgets.Text(description="Предмет:")
grade_value = widgets.IntSlider(description="Оцінка:", min=1, max=100, step=1)
add_grade_button = widgets.Button(description="Додати Оцінку", button_style='info')

def on_add_grade_clicked(b):
    student_id = grade_student_id.value
    subject = grade_subject.value.strip()
    grade = grade_value.value

    # Перевірка, чи існує студент
    students_df = get_students()
    if student_id in students_df['id'].values:
        add_grade(student_id, subject, grade)
        output_add_grade.clear_output()
        with output_add_grade:
            print(f"Оцінку {grade} з предмету {subject} додано студенту з ID {student_id}.")
        # Очищення полів вводу
        grade_student_id.value = 0
        grade_subject.value = ''
        grade_value.value = 1
    else:
        with output_add_grade:
            print(f"Студента з ID {student_id} не існує.")

add_grade_button.on_click(on_add_grade_clicked)
output_add_grade = widgets.Output()

add_grade_box = widgets.VBox([
    add_grade_title,
    grade_student_id,
    grade_subject,
    grade_value,
    add_grade_button,
    output_add_grade
])
```

Рисунок 2.16 - Функція додати оцінку

Компоненти:

- Заголовок.

- Поле для введення ID студента.

- Текстове поле для введення предмету.

- Слайдер для вибору оцінки.

- Кнопка для додавання оцінки.

- Область виводу для повідомлень.

Алгоритм дії:

Користувач вводить ID студента, предмет та вибирає оцінку

На першому етапі користувач системи, яким може бути викладач або адміністратор навчального закладу, вводить у відповідні поля у графічному інтерфейсі системи необхідну інформацію. Зокрема, він зазначає:

- *ID студента* — унікальний ідентифікатор, що призначений кожному студенту під час реєстрації. Це значення дозволяє точно визначити, до якого студента слід додати оцінку.

- *Предмет* — вибирається зі списку доступних предметів або вводиться вручну. Це дозволяє системі знати, за який предмет додаватиметься оцінка.

- *Оцінка* — користувач вибирає або вводить оцінку, яку отримав студент. Оцінки можуть бути у формі числа (зазвичай у діапазоні від 1 до 100), що відповідає шкалі оцінювання.

Натискає кнопку "Додати Оцінку"

Після введення всіх необхідних даних користувач натискає на кнопку "Додати Оцінку", яка активує наступний етап процесу. Ця дія ініціює обробку інформації системою та запуск алгоритму для перевірки правильності введених даних. Система отримує введені значення і починає виконувати перевірки.

- *Система перевіряє, чи існує студент з вказаним ID*

На цьому етапі система перевіряє наявність студента з вказаним унікальним ідентифікатором (ID) у своїй базі даних. Ця перевірка є необхідною, щоб уникнути можливих помилок під час введення даних. Наприклад, викладач міг помилково ввести неправильний ID або студент з таким ідентифікатором міг бути не зареєстрований у системі.

Процес перевірки відбувається автоматично, і система здійснює пошук студента у своїй базі даних, де зберігається вся інформація про зареєстрованих студентів. Якщо система знаходить студента з вказаним ID, вона переходить до наступного етапу.

- *Якщо студент існує, додає оцінку у таблицю grades та відображає повідомлення про успіх*

У разі успішної ідентифікації студента система додає введену оцінку до відповідної таблиці бази даних, яка містить інформацію про всі оцінки студентів. Таблиця grades містить такі поля:

- *ID студента* — посилання на студента, для якого додається оцінка.

- *Предмет* — вказує, за який предмет виставляється оцінка.

- *Оцінка* — числове значення оцінки, яку отримав студент.

- *Дата* — дата, коли оцінка була додана.

Після успішного додавання оцінки в базу даних система генерує повідомлення про успішне завершення операції, яке відображається користувачу. Це повідомлення підтверджує, що оцінка була успішно додана для зазначеного студента і збережена у базі даних.

- *Якщо студент не існує, система відображає повідомлення про відсутність студента з таким ID*

У випадку, коли система не знаходить студента з вказаним ID, вона генерує повідомлення про помилку. Це повідомлення інформує користувача, що у базі даних немає студента з таким ідентифікатором, і тому оцінка не може бути додана.

Це повідомлення допомагає уникнути помилок у введенні даних та дає користувачу можливість виправити ситуацію, перевіривши правильність введених даних або зареєструвавши нового студента у випадку, якщо він ще не доданий у систему

- *Додати присутність*

```
# 3. Додати Присутність
add_attendance_title = widgets.HTML("<h2>Додати Присутність</h2>")
attendance_student_id = widgets.IntText(description="ID Студента:")
attendance_date = widgets.Text(description="Дата (РРРР-ММ-ДД):")
attendance_status = widgets.Dropdown(options=['присутній', 'відсутній'], description="Статус:")
add_attendance_button = widgets.Button(description="Додати Присутність", button_style='warning')

def on_add_attendance_clicked(b):
    student_id = attendance_student_id.value
    date = attendance_date.value.strip()
    status = attendance_status.value

    # Базова перевірка
    if not date:
        with output_add_attendance:
            print("Будь ласка, введіть дійсну дату.")
        return

    # Перевірка, чи існує студент
    students_df = get_students()
    if student_id in students_df['id'].values:
        add_attendance(student_id, date, status)
        output_add_attendance.clear_output()
        with output_add_attendance:
            print(f"Присутність для студента з ID {student_id} на {date} встановлено як {status}.")
        # Очищення полів вводу
        attendance_student_id.value = 0
        attendance_date.value = ''
        attendance_status.value = 'присутній'
    else:
        with output_add_attendance:
            print(f"Студента з ID {student_id} не існує.")

add_attendance_button.on_click(on_add_attendance_clicked)
output_add_attendance = widgets.Output()

add_attendance_box = widgets.VBox([
    add_attendance_title,
    attendance_student_id,
    attendance_date,
    attendance_status,
    add_attendance_button,
    output_add_attendance
])
```

Рисунок 2.17 - Функція додати присутність

Компоненти:

- Заголовок.

- Поле для введення ID студента.

- Текстове поле для введення дати.

- Випадаючий список для вибору статусу присутності.

- Кнопка для додавання запису присутності.

- Область виводу для повідомлень.

Алгоритм дії:

1. Користувач вводить ID студента, дату та вибирає статус присутності.

2. Натискає кнопку "Додати Присутність".

3. Система перевіряє, чи введена дата.

4. Перевіряє, чи існує студент з вказаним ID.

5. Якщо все вірно, додає запис у таблицю attendance та відображає повідомлення про успіх.

6. Якщо ні, відображає відповідне повідомлення про помилку.

Переглянути попередження

```
# 4. Переглянути Попередження
view_warnings_title = widgets.HTML("<h2>Студенти з Попередженнями</h2>")
view_warnings_button = widgets.Button(description="Перевірити Попередження", button_style='danger')

def on_view_warnings_clicked(b):
    warnings_df = check_warnings()
    output_warnings.clear_output()
    with output_warnings:
        if warnings_df.empty:
            print("Студентів з попередженнями немає.")
        else:
            display(warnings_df)

view_warnings_button.on_click(on_view_warnings_clicked)
output_warnings = widgets.Output()

view_warnings_box = widgets.VBox([
    view_warnings_title,
    view_warnings_button,
    output_warnings
])
```

Рисунок 2.18 - Функція перегляду попереджень

Компоненти:

- Заголовок.

- Кнопка для перевірки попереджень.

- Область виводу для таблиці з попередженнями.

Алгоритм дії:

1. Користувач натискає кнопку "Перевірити Попередження".

2. Система виконує аналіз присутності та визначає студентів з 3 або більше пропусками.

3. Відображає таблицю зі списком таких студентів або повідомлення про відсутність попереджень.

Експортувати середні оцінки в Excel

```
# 5. Експортувати Середні Оцінки в Excel
export_excel_title = widgets.HTML("<h2>Експортувати Середні Оцінки в Excel</h2>")
excel_file_name = widgets.Text(value='average_grades.xlsx', description="Назва файлу:")
export_excel_button = widgets.Button(description="Експортувати", button_style='primary')

def on_export_excel_clicked(b):
    file_name = excel_file_name.value.strip()
    if not file_name.endswith('.xlsx'):
        file_name += '.xlsx'
    try:
        file_path = save_average_grades_to_excel(file_name)
        output_export_excel.clear_output()
        with output_export_excel:
            print(f"Файл '{file_path}' успішно створено. Завантаження...")
            files.download(file_path)
    except Exception as e:
        with output_export_excel:
            print(f"Помилка: {e}")

export_excel_button.on_click(on_export_excel_clicked)
output_export_excel = widgets.Output()

export_excel_box = widgets.VBox([
    export_excel_title,
    excel_file_name,
    export_excel_button,
    output_export_excel
])
```

Рисунок 2.19 - Функція перегляду Студентів

Компоненти:

- Заголовок.

- Текстове поле для введення назви файлу.

- Кнопка для експорту.

- Область виводу для повідомлень.

Алгоритм дії:

1. Користувач вводить назву файлу (наприклад, average_grades.xlsx).

2. Натискає кнопку "Експортувати".

3. Система об'єднує дані про оцінки та студентів, обчислює середні оцінки.

4. Зберігає результат у вказаний Excel-файл.

5. Відображає повідомлення про успішне створення файлу та ініціює його завантаження.

Переглянути усіх Студентів

```
# 6. Переглянути Усіх Студентів
view_students_title = widgets.HTML("<h2>Всі Студенти</h2>")
view_students_button = widgets.Button(description="Оновити", button_style='success')

def on_view_students_clicked(b):
    students_df = get_students()
    output_view_students.clear_output()
    with output_view_students:
        if students_df.empty:
            print("Студентів не знайдено.")
        else:
            display(students_df)

view_students_button.on_click(on_view_students_clicked)
output_view_students = widgets.Output()

view_students_box = widgets.VBox([
    view_students_title,
    view_students_button,
    output_view_students
])
```

Рисунок 2.20 - Функція перегляду Студентів

Компоненти:

- Заголовок.

- Кнопка для оновлення (відображення всіх студентів).

- Область виводу для таблиці з інформацією про студентів.

Алгоритм Дії:

1. Користувач натискає кнопку "Оновити".

2. Система отримує всі записи з таблиці students.

3. Відображає таблицю з інформацією про всіх студентів або повідомлення про відсутність записів.

Переглянути усі оцінки

```
# 7. Переглянути усі оцінки
view_grades_title = widgets.HTML("<h2>Всі Оцінки</h2>")
view_grades_button = widgets.Button(description="Оновити", button_style='success')

def on_view_grades_clicked(b):
    grades_df = get_grades()
    output_view_grades.clear_output()
    with output_view_grades:
        if grades_df.empty:
            print("Оцінок не знайдено.")
        else:
            display(grades_df)

view_grades_button.on_click(on_view_grades_clicked)
output_view_grades = widgets.Output()

view_grades_box = widgets.VBox([
    view_grades_title,
    view_grades_button,
    output_view_grades
])
```

Рисунок 2.21 - Функція переглянути усі оцінки

Компоненти:

- Заголовок.

- Кнопка для оновлення (відображення всіх оцінок).

- Область виводу для таблиці з інформацією про оцінки.

Алгоритм дії:

1. Користувач натискає кнопку "Оновити".

2. Система отримує всі записи з таблиці grades.

3. Відображає таблицю з інформацією про всі оцінки або повідомлення про відсутність записів.

Переглянути відсутніх Студентів

```
# 8. Переглянути Відсутніх Студентів
view_absentees_title = widgets.HTML("<h2>Відсутні Студенти</h2>")
view_absentees_button = widgets.Button(description="Оновити", button_style='info')

def on_view_absentees_clicked(b):
    absentees_df = get_absent_students()
    output_view_absentees.clear_output()
    with output_view_absentees:
        if absentees_df.empty:
            print("Відсутніх студентів не знайдено.")
        else:
            display(absentees_df)

view_absentees_button.on_click(on_view_absentees_clicked)
output_view_absentees = widgets.Output()

view_absentees_box = widgets.VBox([
    view_absentees_title,
    view_absentees_button,
    output_view_absentees
])
```

Рисунок 2.22 - Функція перегляду відсутніх Студентів

Компоненти:

- Заголовок.

- Кнопка для оновлення (відображення відсутніх студентів).

- Область виводу для таблиці з інформацією про відсутніх студентів.

Алгоритм дії:

1. Користувач натискає кнопку "Оновити".

2. Система отримує всі записи з таблиці attendance, де статус "відсутній".

3. Відображає таблицю з інформацією про відсутніх студентів або повідомлення про відсутність таких записів.

5. Організація вкладок та відображення інтерфейсу

Всі окремі функціональні блоки організовані у вигляді вкладок за допомогою ipywidgets.Tab. Кожна вкладка містить відповідні віджети для взаємодії з користувачем:

```
tabs = widgets.Tab(children=[
    add_student_box,
    add_grade_box,
    add_attendance_box,
    view_warnings_box,
    export_excel_box,
    view_students_box,
    view_grades_box,
    view_absentees_box
])

tab_titles = [
    'Додати Студента',
    'Додати Оцінку',
    'Додати Присутність',
    'Переглянути Попередження',
    'Експортувати в Excel',
    'Всі Студенти',
    'Всі Оцінки',
    'Відсутні Студенти'
]

for i, title in enumerate(tab_titles):
    tabs.set_title(i, title)
```

```
# Організація всіх розділів у вкладки
tabs = widgets.Tab(children=[
    add_student_box,
    add_grade_box,
    add_attendance_box,
    view_warnings_box,
    export_excel_box,
    view_students_box,
    view_grades_box,
    view_absentees_box
])

tab_titles = [
    'Додати студента',
    'Додати оцінку',
    'Додати присутність',
    'Переглянути попередження',
    'Експортувати в Excel',
    'Всі студенти',
    'Всі оцінки',
    'Відсутні студенти'
]

for i, title in enumerate(tab_titles):
    tabs.set_title(i, title)
```

Рисунок 2.23 - Всі розділи у вкладках

Відображення вкладок

display(tabs)

```
# Відображення вкладок
display(tabs)
```

Рисунок 2.24 - Відображення вкладок

Алгоритм роботи вкладок:

Створення вкладок:

1. Кожен функціональний блок (наприклад, додавання студента, додавання оцінки тощо) розміщується у окремому VBox, який потім додається до списку children вкладок.

Назви вкладок:

- Встановлюються зрозумілі назви для кожної вкладки відповідно до її функціональності.

Відображення:

- Всі вкладки відображаються одночасно, дозволяючи користувачу легко перемикатися між різними функціями системи.

6. Основні кроки роботи системи

Алгоритм роботи системи можна поділити на наступні основні етапи:

Ініціалізація системи:

- Імпорт бібліотек та встановлення необхідних модулів.
- Створення бази даних та необхідних таблиць, якщо вони ще не існують.

Введення даних:

- Користувач вводить інформацію про студента, оцінки або присутність через відповідні форми у вкладках.

Перевірка та валідація:

- Система перевіряє коректність введених даних (наприклад, існування студента за ID, заповнення всіх полів).

Збереження даних:

- Коректні дані зберігаються у відповідних таблицях бази даних.

Аналіз даних:

- Система аналізує дані про присутність для визначення студентів з попередженнями.
- Обчислює середні оцінки студентів за всіма предметами.

Відображення Даних:

- Користувач може переглядати всі записи про студентів, оцінки та присутність через відповідні вкладки.
- Відображення попереджень та відсутніх студентів здійснюється у спеціальних вкладках.

Експорт даних:

1. Середні оцінки можуть бути експортовані у форматі Excel та завантажені на локальний пристрій користувача.

Завершення роботи:

1. Користувач може завершити взаємодію з системою, закривши або перезапустивши нотатник.

7. Взаємодія користувача з системою

Користувач взаємодіє з системою через графічний інтерфейс, який складається з кількох вкладок, кожна з яких відповідає певній функціональності:

- *Додати Студента:* Введення та збереження нових студентів.

- *Додати Оцінку*: Запис оцінок для існуючих студентів.

- *Додати Присутність*: Введення інформації про присутність студентів на з-няттях.

- *Переглянути Попередженн***я**: Отримання списку студентів з частими пропусками.

- *Експортувати в Excel*: Генерація та завантаження Excel-файлу зі середніми оцінками.

- *Всі Студенти*: Перегляд списку всіх студентів.

- *Всі Оцінки*: Перегляд всіх записаних оцінок.

- *Відсутні Студенти*: Перегляд списку всіх відсутніх студентів.

Кожна вкладка містить відповідні віджети для введення даних, кнопки для виконання дій та області виводу для відображення результатів або повідомлень.

8. Обробка винятків та повідомлень

Система передбачає базову обробку помилок та надання зворотного зв'язку користувачу:

Валідація Вводу:

- Перевірка заповнення обов'язкових полів.

- Перевірка існування студентів за їхніми ID перед додаванням оцінок або присутності.

Повідомлення Про Статус Дій:

- Повідомлення про успішне додавання записів.

- Інформація про помилки або відсутність необхідних даних.

Обробка Винятків:

- Перехоплення та відображення повідомлень про помилки під час експорту даних у Excel.

9. Завантаження та Експорт Даних

Система дозволяє експортувати обчислені середні оцінки студентів у форматі Excel для подальшого аналізу або збереження:

Генерація Файлу:

- Система об'єднує дані з таблиць grades та students.
- Обчислює середні оцінки для кожного студента.
- Зберігає результат у вказаний файл формату .xlsx.

Завантаження Файлу:

- Після успішного створення файлу, система автоматично ініціює його завантаження на локальний пристрій користувача за допомогою files.download().

Для блок-схеми алгоритму роботи системи аналізу роботи студентів можна виділити основні етапи, представлені в коді. Ось етапи алгоритму, які можуть бути включені в блок-схему:

1. *Початок*: Ініціалізація програми та імпорт необхідних бібліотек.
2. *Створення таблиць в базі даних*: Виконується функція create_tables(), яка створює таблиці для студентів, оцінок і присутності.
3. *Функціональні дії користувача:*

1. *Додати студента*: Перевіряє введені дані, зберігає їх у таблиці students.
2. *Додати оцінку*: Перевіряє наявність студента, потім зберігає оцінку в таблиці grades.
3. *Додати присутність*: Перевіряє наявність студента, потім записує його присутність чи відсутність у таблиці attendance.
4. *Перевірка попереджень:* Функція check_warnings() перевіряє студентів з певною кількістю пропусків (наприклад, 3 або більше) та відображає список студентів з попередженнями.

5. *Експорт середніх оцінок в Excel:* Функція save_average_grades_to_excel() обчислює середню оцінку кожного студента та зберігає її в Excel-файлі.

6. *Перегляд усіх студентів, оцінок, відсутніх студентів*: Відображає відповідні дані на основі вибору користувача.

7. *Кінець*: Завершення роботи алгоритму.

Дана блок-схема зображає логіку програми для роботи з базою даних студентів. Вона включає наступні основні етапи:

Початок процесу:

- Імпорт необхідних бібліотек для реалізації програми.

- Створення таблиць у базі даних для зберігання даних про студентів, оцінки та присутність.

Дії користувача:

- Користувач має кілька варіантів дій:

- *Додати студента*: введення даних про нового студента та збереження у базі даних.

- *Додати оцінку*: додавання оцінки студенту після перевірки його наявності в базі.

- *Додати присутність*: запис присутності студента на заняттях.

- *Перевірка попереджень*: аналіз пропусків студентів і перевірка, чи потрібно створити попередження.

Функції аналізу та експорту:

- Експорт даних про оцінки студентів у формат Excel для подальшої обробки чи аналізу.

- Перегляд загальної інформації про студентів, включаючи оцінки та відвідуваність.

Кінець: Завершення виконання програми після виконання необхідних дій.

Основні особливості:

- *Циклічність*: Програма чекає на дії користувача після кожного кроку.

- *Умовні переходи*: Виконуються залежно від обраних дій користувача.

- *Автоматизація перевірки*: Доступність попереджень і перевірки даних інтегрована у процес.

2.3 Функціональні модулі системи

Інформаційна система для аналізу роботи студентів є комплексною програмною платформою, яка включає кілька функціональних модулів, кожен із яких відповідає за різні аспекти роботи зі студентськими даними. Модулі забезпечують можливість управління студентами, їх оцінками, відвідуваністю занять, а також автоматизують процеси генерації звітів та попереджень. Усі модулі взаємодіють між собою через спільну базу даних, що дозволяє зручно зберігати та обробляти інформацію. Нижче описані основні функціональні модулі системи

1. Модуль керування студентами

Рисунок 2.26. - Схема модуля керування студентами

Цей модуль забезпечує основну функцію системи — додавання та управління студентами, які навчаються в навчальному закладі. Він дозволяє користувачам вводити дані про кожного студента, включаючи їх ім’я, прізвище та номер групи, до бази даних. Основна перевага цього модуля — можливість легко додавати нових студентів і оновлювати інформацію про існуючих. Модуль реалізований через зручний інтерфейс, що використовує графічні віджети для введення даних.

Основний функціонал модуля:

- Додавання нових студентів до бази даних.

- Можливість редагування вже наявної інформації про студентів.

- Перегляд списку всіх зареєстрованих студентів із можливістю пошуку та фільтрації.

Завдяки цьому модулю адміністратори або викладачі можуть швидко та безпомилково реєструвати нових студентів, а також контролювати поточний склад навчальних груп.

2. Модуль введення оцінок

Рисунок 2.27. - Схема модуля введення оцінок

Оцінки є ключовим показником успішності студентів, тому цей модуль забезпечує введення та управління даними про успішність студентів з різних предметів. Користувачі можуть вводити оцінки, вказуючи предмет, студентський номер (ID) та саму оцінку. Модуль дозволяє вводити оцінки у діапазоні від 1 до 100, що відповідає загальноприйнятій системі оцінювання.

Основний функціонал модуля:

- Введення оцінок для кожного студента за певний предмет.

- Автоматичне зберігання оцінок у базі даних та можливість їх подальшого аналізу.

- Перегляд усіх оцінок із можливістю слідкувати за студентами або предметами.

Модуль також інтегрується з функцією автоматичного обчислення середніх оцінок для кожного студента, що полегшує аналіз успішності для викладачів.

3. Модуль керування присутністю

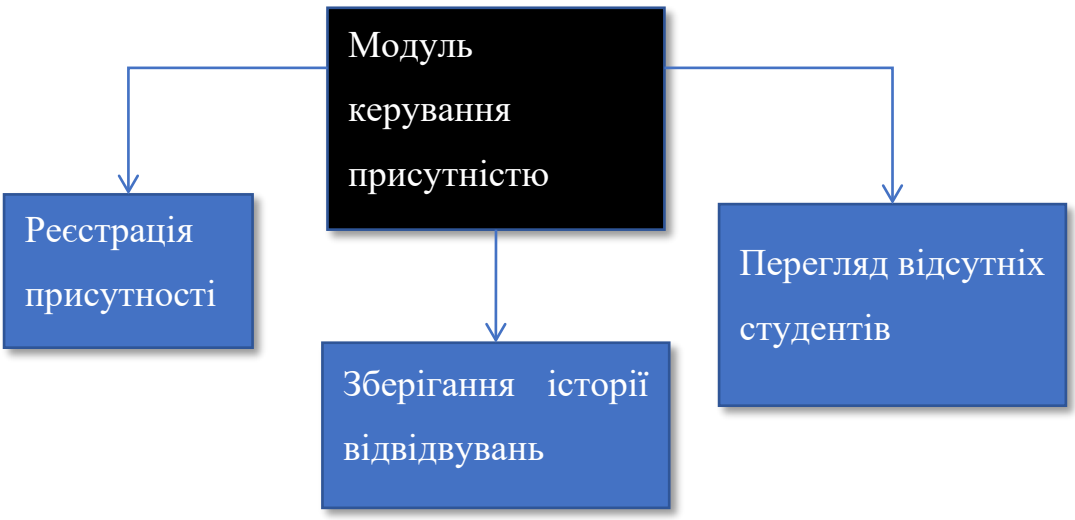

Рисунок 2.28. - Схема модуля керування присутністю

Контроль відвідуваності студентів є важливою складовою навчального процесу. Цей модуль дозволяє фіксувати присутність або відсутність студентів на заняттях. Адміністратори чи викладачі можуть вводити дані про те, чи був студент присутній або відсутній на конкретний день.

Основний функціонал модуля:

- Реєстрація присутності або відсутності студентів на заняттях.

- Зберігання історії відвідувань із зазначенням дати та статусу ("присутній" або "відсутній").

- Можливість перегляду відсутніх студентів за конкретний період часу.

Цей модуль інтегрується з іншими компонентами системи, що дозволяє формувати звіти про відвідуваність і виявляти студентів, які мають проблеми з відвідуванням занять.

4. Модуль попереджень

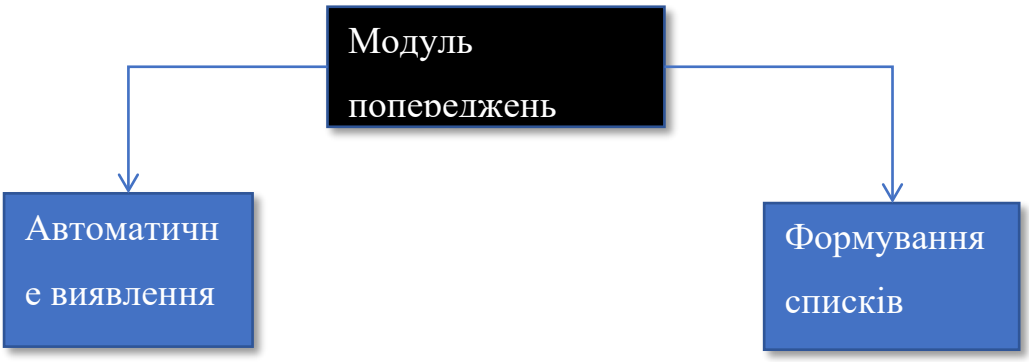

Рисунок 2.29. - Схема модуля попереджень

Для того, щоб швидко реагувати на низьку відвідуваність або проблеми з успішністю, система має модуль попереджень. Цей модуль автоматично перевіряє кількість відсутностей у студентів і визначає тих, хто перевищив допустиму кількість пропусків. Модуль попереджень допомагає викладачам або адміністраторам навчального закладу ідентифікувати студентів, які можуть мати труднощі, та вжити необхідних заходів для покращення їхньої успішності.

Основний функціонал модуля:

- Автоматичне виявлення студентів, які мають більше трьох відсутностей.

- Формування списків студентів для видачі попереджень або вжиття інших заходів.

- Можливість перегляду історії відсутностей для кожного студента.

Цей модуль дозволяє адміністрації навчального закладу оперативно реагувати на можливі проблеми з навчальним процесом та підтримувати дисципліну.

5. Модуль експорту в Excel

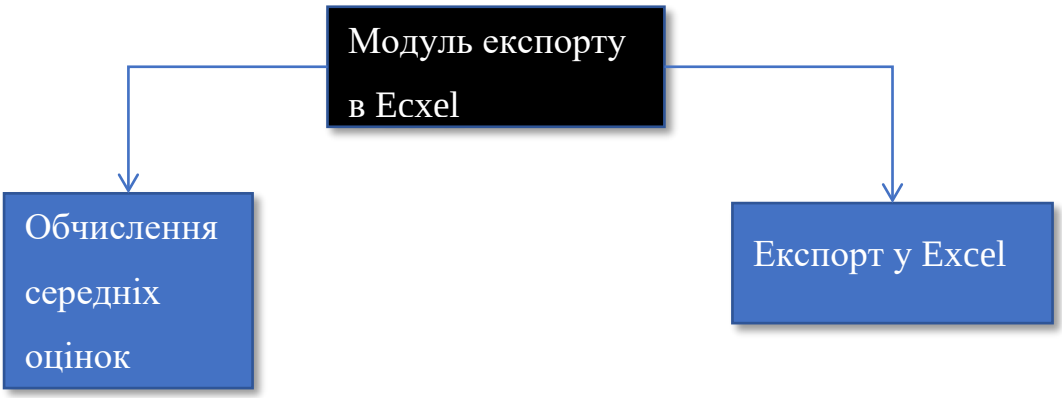

Рисунок 2.30. - Схема модуля експорту в Excel

Модуль експорту є важливим компонентом системи, який дозволяє користувачам зберігати зібрані дані у вигляді звітів у форматі Excel. Викладачі можуть створювати звіти про середні оцінки студентів та іншу аналітичну інформацію, зберігаючи їх у файл, який може бути використаний для подальшого аналізу або надання іншим особам.

Основний функціонал модуля:

- Обчислення середніх оцінок для кожного студента за всіма предметами.

- Можливість експорту даних у формат Excel із подальшим завантаженням файлу на комп'ютер.

- Зручний інтерфейс для введення назви файлу та вибору параметрів експорту.

Цей модуль полегшує роботу викладачів та адміністрації, дозволяючи автоматизувати процес створення звітів про успішність студентів.

6. Модуль перегляду даних

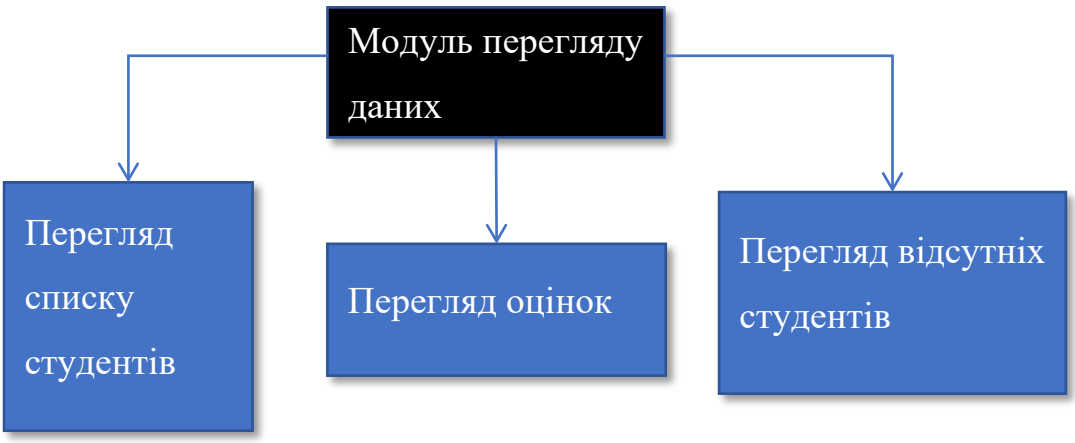

Рисунок 2.31. - Схема модуля перегляду даних

Останнім, але не менш важливим, є модуль перегляду даних. Він забезпечує доступ до всієї інформації, збереженої в системі, з можливістю швидкого пошуку та фільтрації. Користувачі можуть переглядати списки студентів, оцінок, дані про відвідуваність, а також отримувати звіти в реальному часі.

Основний функціонал модуля:

- Перегляд списку всіх зареєстрованих студентів.

- Перегляд усіх оцінок студентів із можливістю сортування за предметами або студентами.

- Перегляд відсутніх студентів із деталізацією за датами.

Завдяки цьому модулю, викладачі та адміністрація можуть оперативно отримувати потрібну інформацію для прийняття рішень.

Висновок: У рамках розробки програмного забезпечення для інформаційної системи аналізу роботи студентів в учбовому процесі було створено функціональну базу даних, яка дозволяє здійснювати управління інформацією про студентів, їхні оцінки та присутність.

Завдяки інтеграції бібліотек sqlite3, pandas та інтерактивних віджетів ipywidgets, розроблено зручний інтерфейс для додавання нових студентів, оцінок, а також реєстрації їхньої присутності на заняттях. Система забезпечує такі можливості:

- відстеження відсутніх студентів та формування списків студентів, які потребують попередження через велику кількість пропусків;

- аналіз середніх оцінок студентів із подальшим експортом даних у формат Excel;

- інтерактивне відображення списків студентів, оцінок та відсутніх осіб.

Розроблена система дозволяє автоматизувати рутинні завдання, пов’язані з обліком студентів, зменшує кількість помилок, які можуть виникати при ручному введенні даних, та спрощує аналіз отриманих результатів. Це сприятиме підвищенню ефективності роботи навчального закладу та дозволить приділяти більше уваги індивідуальному підходу до навчання кожного студента.

РОЗДІЛ 3 ЗАПУСК ПРОГРАМИ

3.1 Результат тестування

Додати Студента:

- Ця вкладка надає можливість додати інформацію про нового студента.

- Поля для заповнення:

- "Ім'я": введення імені студента.

- "Прізвище": введення прізвища студента.

- "Номер Групи": введення номера групи студента.

- Після заповнення полів і натискання кнопки "Додати Студента" програма додасть студента в базу даних і відобразить повідомлення про успішне додавання, як показано: "Студента Артем Сільман успішно додано."

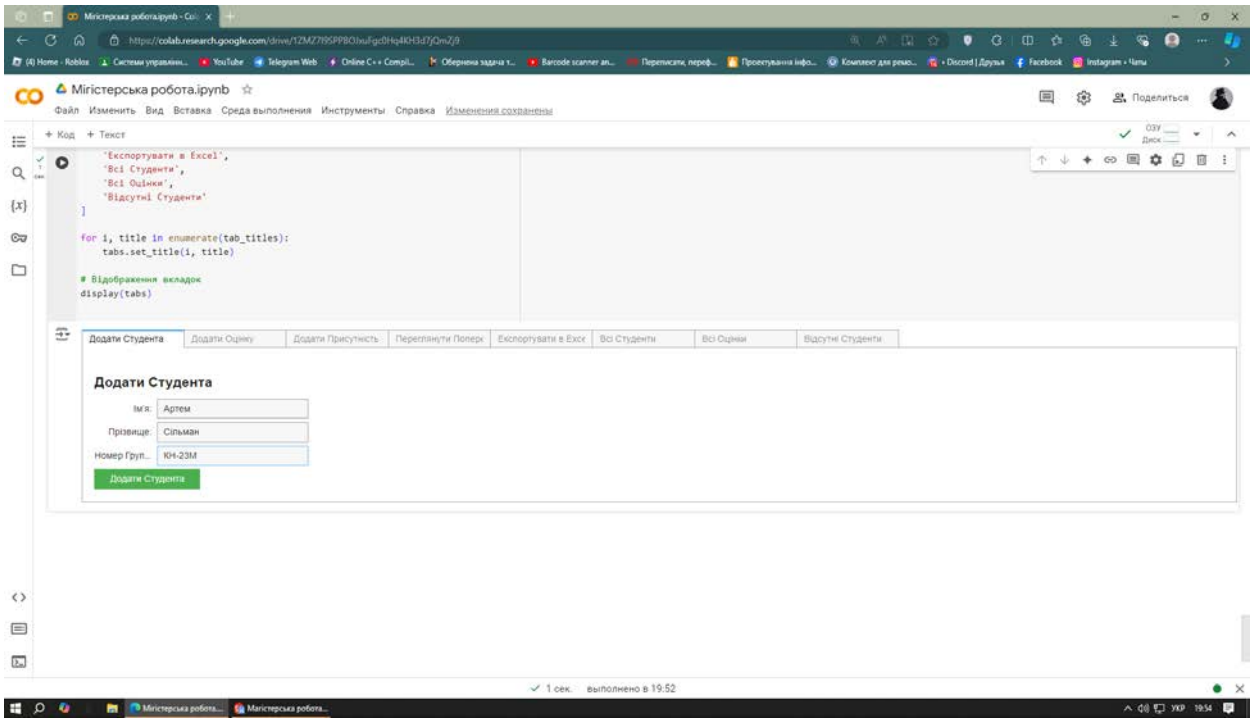

Рисунок 3.1 - Функція додати студента

Додати Оцінку:

- Вкладка для додавання оцінок студентам.

- Дає змогу ввести оцінку для конкретного студента за конкретний предмет.

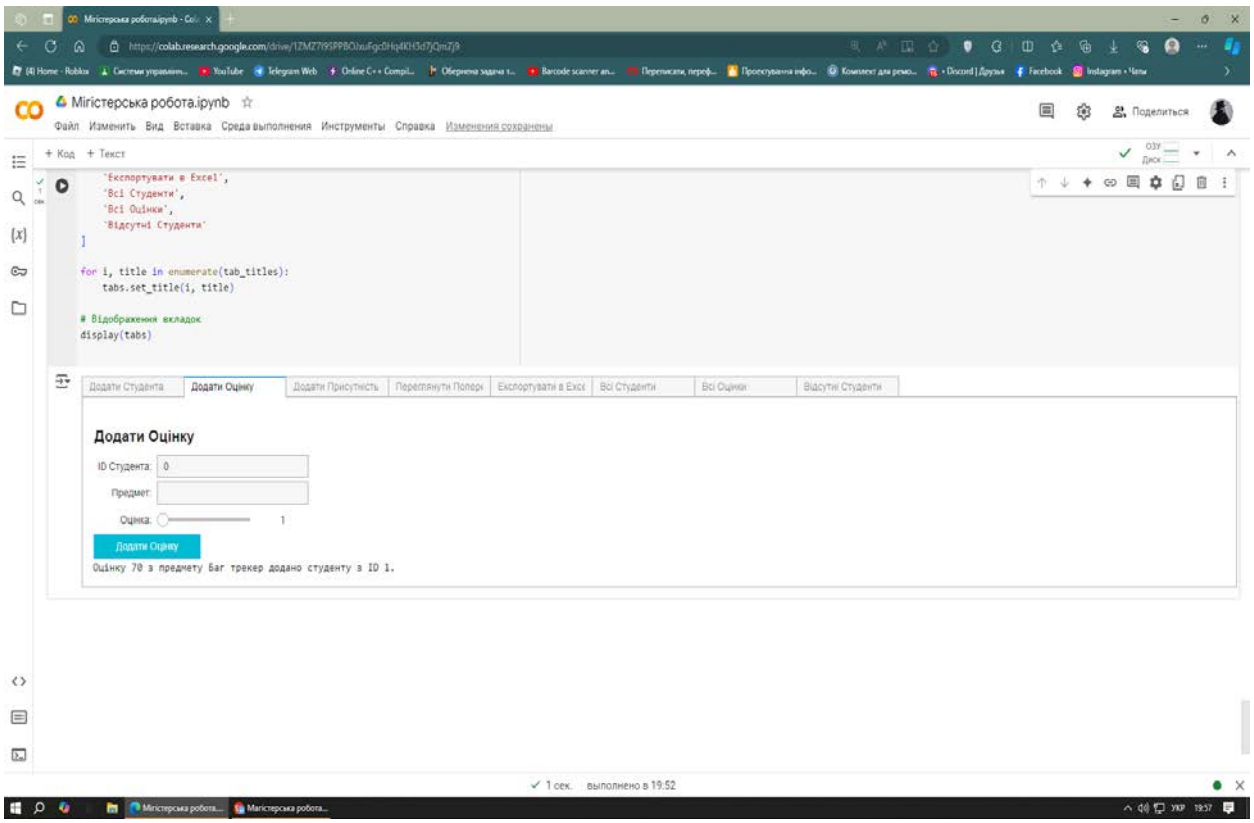

Рисунок 3.2 - Функція додати оцінку

Додати Присутність:

- Вкладка для позначення присутності або відсутності студента на заняттях.

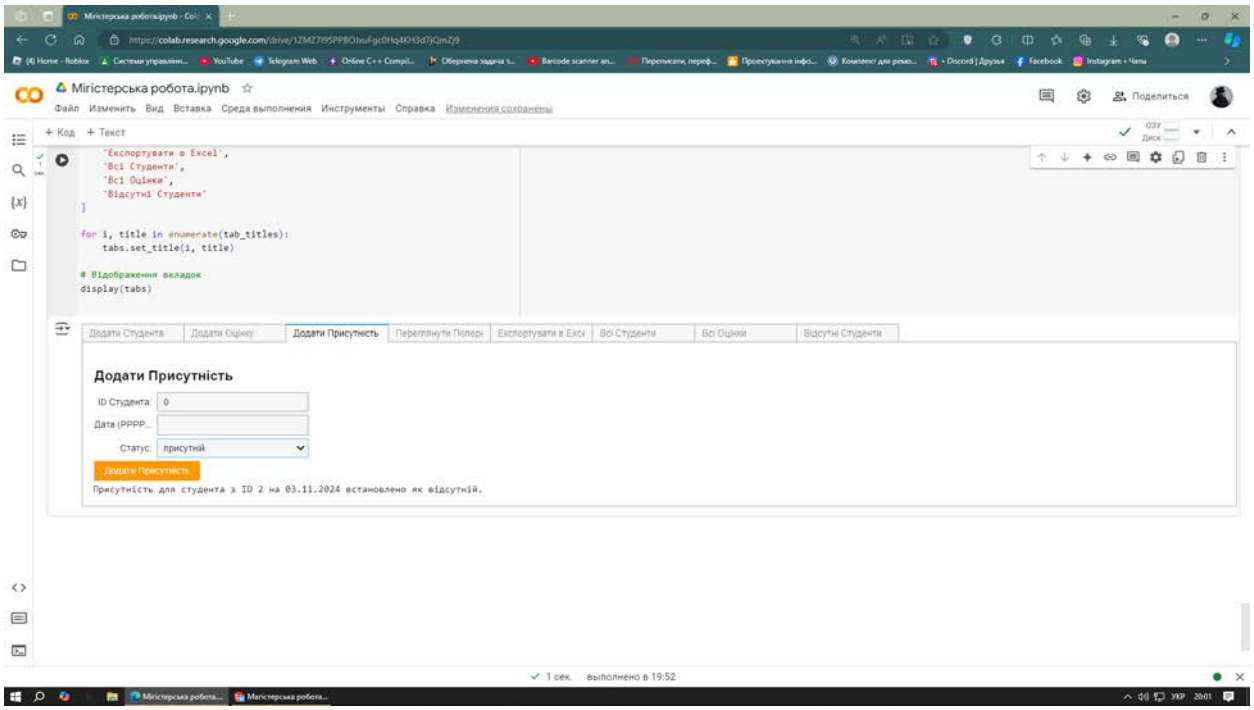

Рисунок 3.3 - Функція додати присутність

Переглянути Попередження:

- Ця вкладка призначена для перегляду студентів, які мають проблеми з відвідуваністю або низькі оцінки.

- Програма генерує попередження, якщо студент має більше допустимої кількості пропусків.

- Дозволяє обрати студента та вказати дату відвідування.

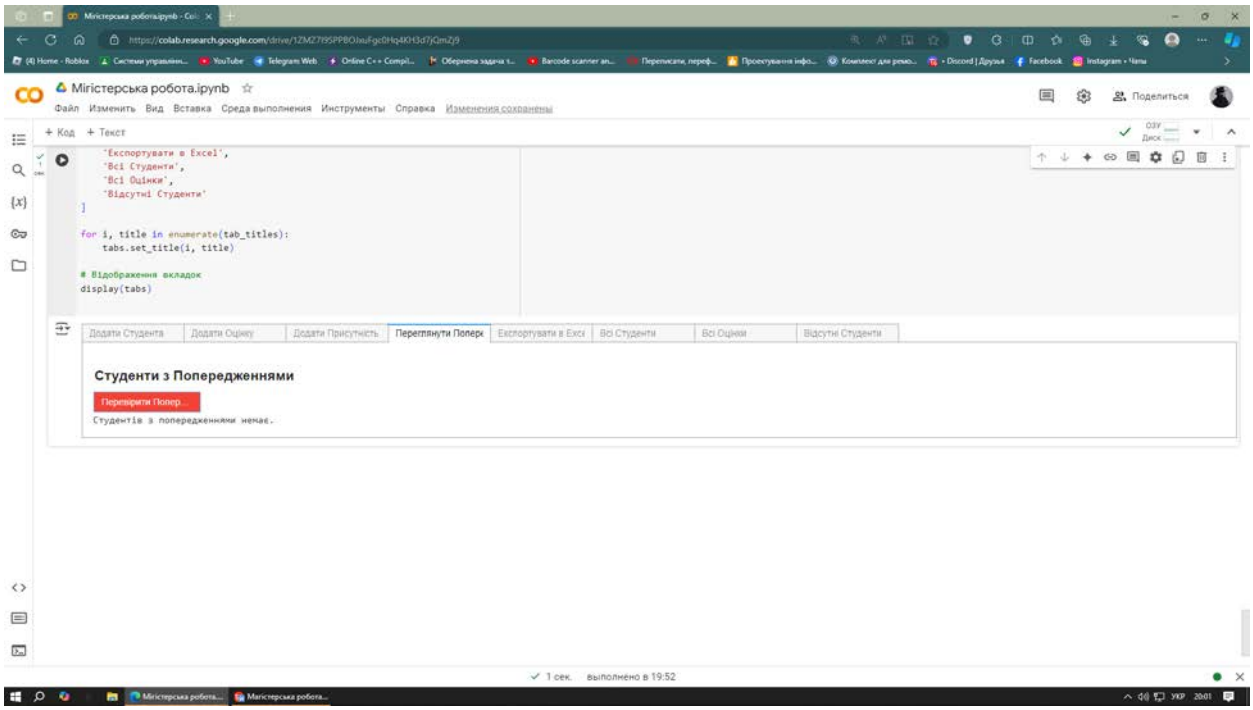

Рисунок 3.4 - Функція студенти з перередженням

Експортувати в Excel:

- Функція експорту даних до формату Excel.

- Користувач може зберігати звіти або дані про успішність студентів для подальшого аналізу або надання іншим користувачам.

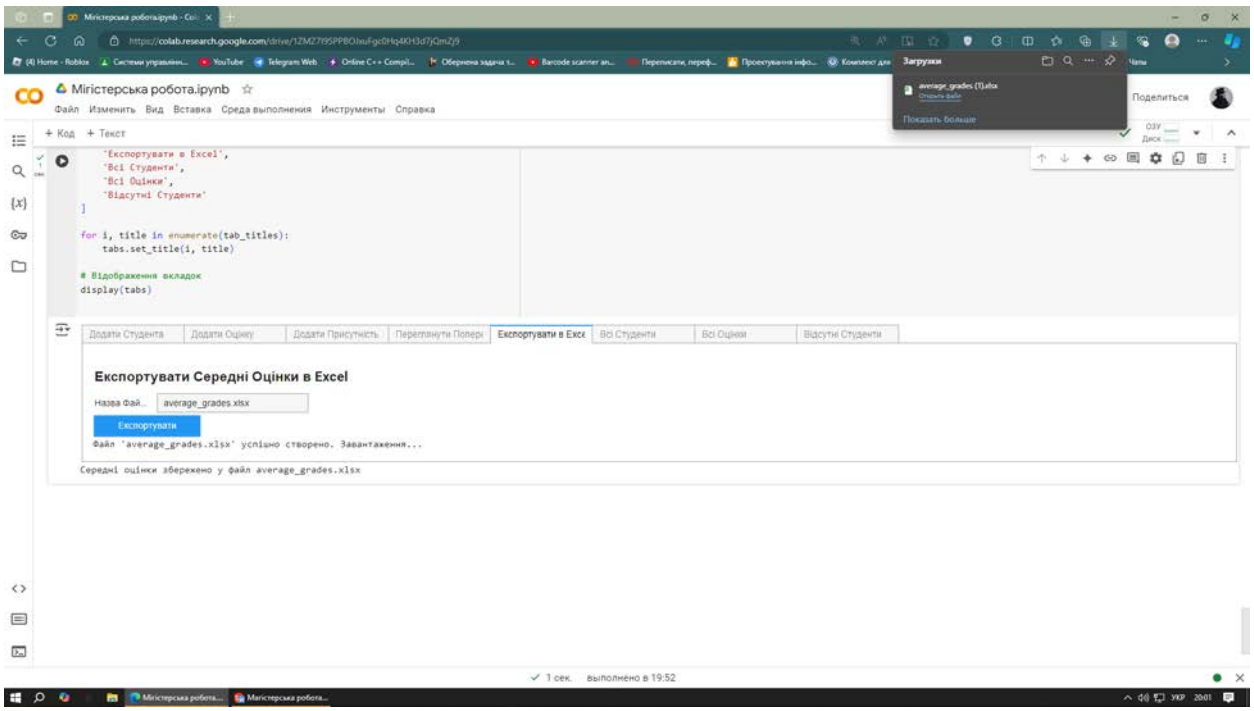

Рисунок 3.5 - Функція екпортувати в Excel

Всі Студенти:

- Вкладка, яка відображає список усіх студентів, зареєстрованих у системі.

- Забезпечує швидкий доступ до інформації про всіх студентів із можливістю фільтрації та сортування.

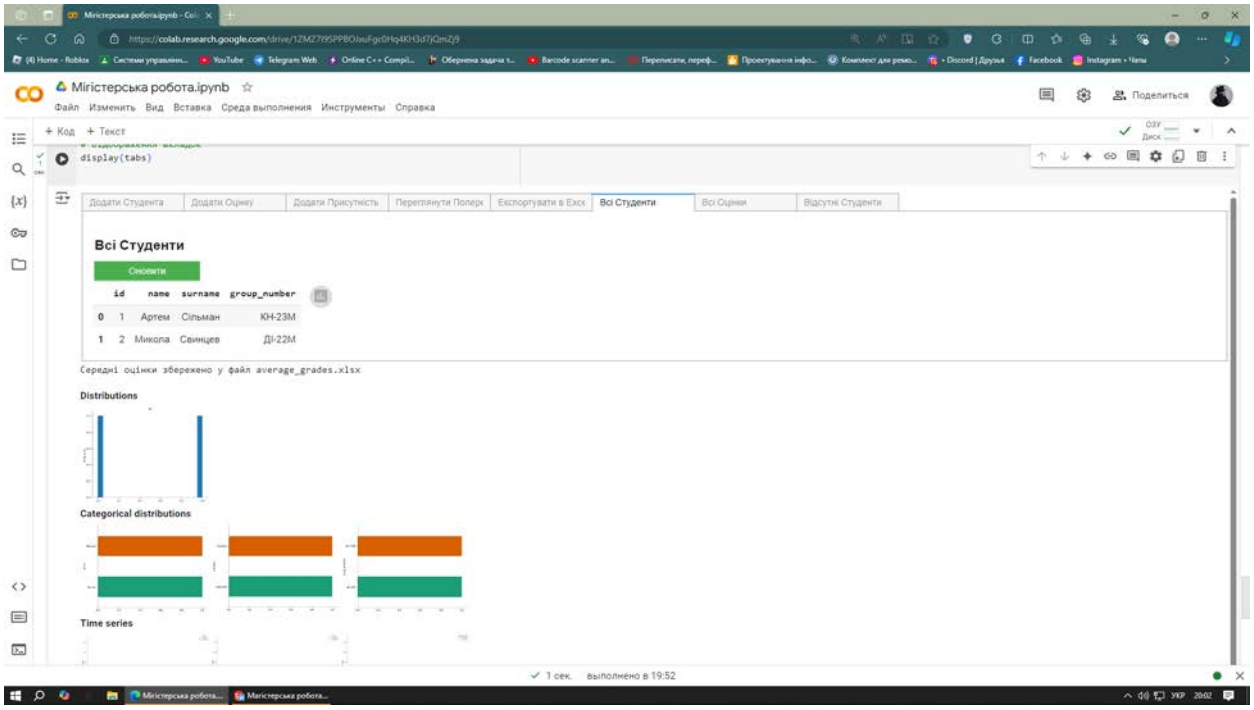

Рисунок 3.6 - Функція всі студенти

Всі Оцінки:

- Відображає усі оцінки, отримані студентами.

- Можливість перегляду оцінок з фільтруванням за предметами або конкретними студентами.

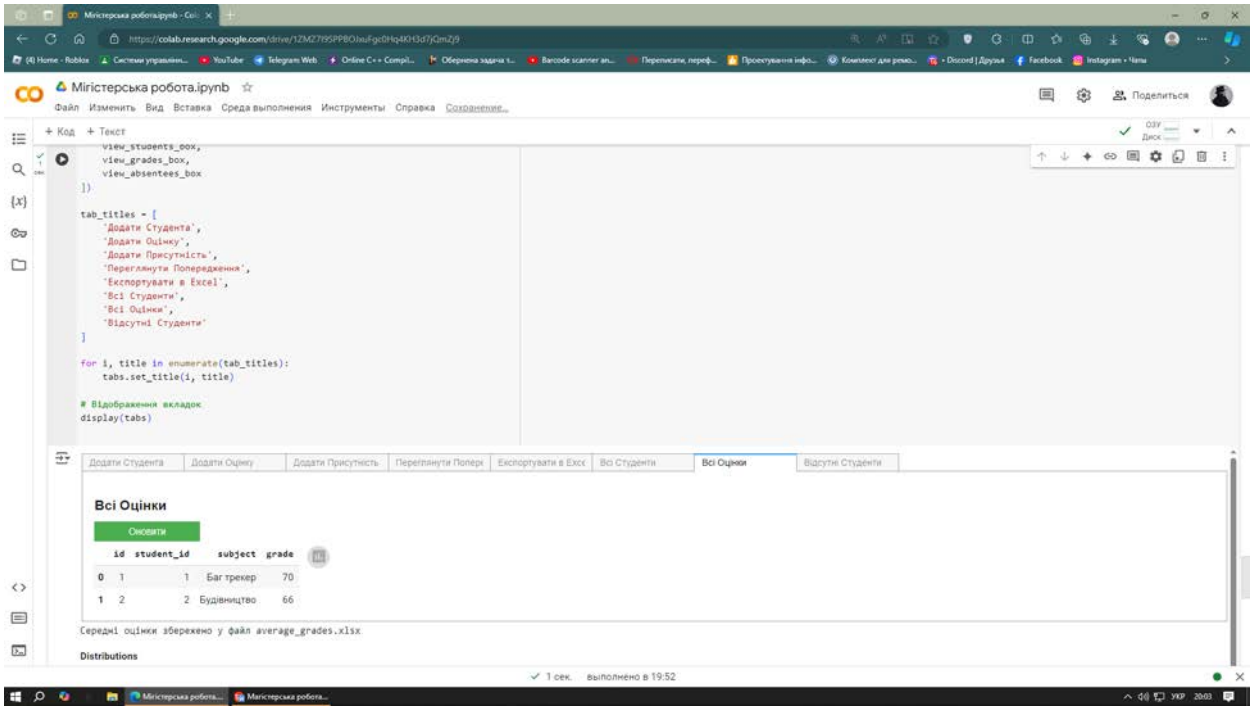

Рисунок 3.7 - Функція всі оцінки

Відсутні Студенти:

- Вкладка для перегляду студентів, які були відсутні на заняттях.

- Дозволяє швидко знайти інформацію про відсутніх студентів і переглянути їхню історію відвідувань.

Цей інтерфейс забезпечує зручний доступ до основних функцій для додавання, управління та перегляду даних про студентів, їхню успішність та відвідуваність.

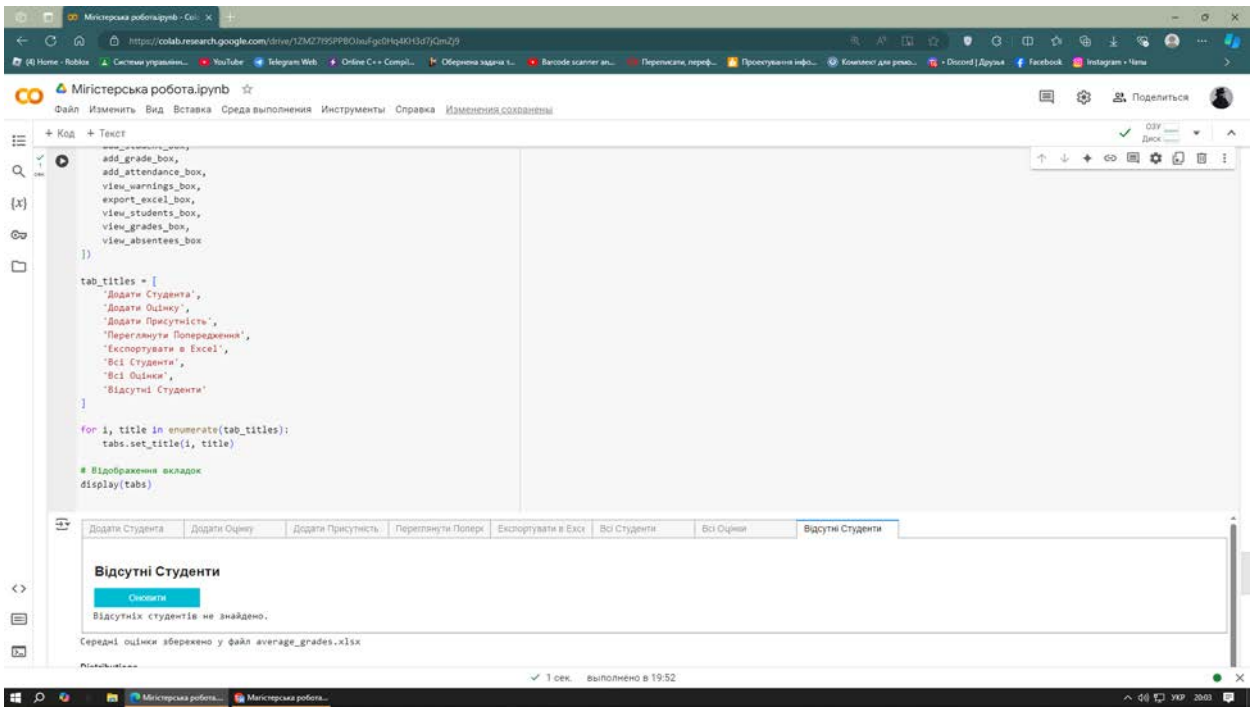

Рисунок 3.8 - Функція відсутні студенти

ВИСНОВКИ

Маємо розроблену інформаційну систему, яка автоматизує процеси збору, аналізу та обробки даних про успішність та відвідуваність студентів, сприяючи підвищенню ефективності управління навчальним процесом.

Були проведені:

1.Огляд аналіз літератури та існуючих систем

2.Теоретична основи для розробки іформаційної системи

3.Розробити структурну схему іформаційної системи

4.Розробити алгоритм роботи іформаційної системи

5.Розробити код програмне забезпечення для модулів іформаційної системи

6.Розробити інтерфейс системи

7.Розробка інструкції для користувачів

Важливим аспектом є інтеграція зворотного зв’язку у навчальний процес, що забезпечує можливість своєчасного коригування методик і форм навчання. Крім того, значна увага повинна приділятися розвитку м'яких навичок (soft skills) у студентів, що є необхідною умовою їхньої подальшої професійної реалізації.

Таким чином, для досягнення високої ефективності навчального процесу необхідне поєднання традиційних і сучасних підходів, із акцентом на активну участь студентів та гнучкість освітніх програм.

У рамках розробки програмного забезпечення для інформаційної системи аналізу роботи студентів в учбовому процесі було створено функціональну базу даних, яка дозволяє здійснювати управління інформацією про студентів, їхні оцінки та присутність.

Завдяки інтеграції бібліотек sqlite3, pandas та інтерактивних віджетів ipywidgets, розроблено зручний інтерфейс для додавання нових студентів, оцінок, а також реєстрації їхньої присутності на заняттях. Система забезпечує такі можливості:

- відстеження відсутніх студентів та формування списків студентів, які потребують попередження через велику кількість пропусків;

- аналіз середніх оцінок студентів із подальшим експортом даних у формат Excel;

- інтерактивне відображення списків студентів, оцінок та відсутніх осіб.

Розроблена система дозволяє автоматизувати рутинні завдання, пов’язані з обліком студентів, зменшує кількість помилок, які можуть виникати при ручному введенні даних, та спрощує аналіз отриманих результатів. Це сприятиме підвищенню ефективності роботи навчального закладу та дозволить приділяти більше уваги індивідуальному підходу до навчання кожного студента.

СПИСОК ЛІТЕАТУРИ

1. Лутц М. Вивчаємо Python. Основи програмування : навч. Посіб. Київ : Діалектика, 2019. 704 с.

2. Кузнєцова К. В., Попов В. В. Програмування на Python для початківців : навч.-метод. посіб. Львів : Літопис, 2020. 156 с.

3. Кауфман Р. Python для дітей. Веселий вступ до програмування : навч. посіб. Харків : Фабула, 2018. 280 с.

4. Саммерфілд М. Програмування на Python 3. Довідник і керівництво : монографія. Київ : BHV, 2017. 608 с.

5. Рахуба А. А., Ворона В. І. Основи розробки інформаційних систем : навч. посіб. Київ : Академія, 2016. 336 с.

6. Скуратівський В. В. Бази даних та інформаційні системи : підручник. Київ : Наука і освіта, 2019. 432 с.

7. Седов А. І. Інформаційні системи і бази даних : навч. посіб. Київ : Видавництво Логос, 2021. 512 с.

8. Гладков Ю. М. Аналіз даних з використанням Python : навч. посіб. Київ : ЦУЛ, 2022. 178 с.

9. Гусаков І. В. Методи обробки даних в навчальних інформаційних системах : монографія. Харків : НТУ ХПІ, 2015. 265 с.

10. Швейцер С. Ю. Програмування для наукових обчислень з Python : навч. посіб. Одеса : ОНУ, 2020. 195 с.

11. Головань М. П. Інформаційні системи та технології : підручник. Львів : Видавництво Львівської політехніки, 2018. 450 с.

12. Балабанов І. Т. Проектування баз даних : навч. посіб. Київ : ВД "Академія", 2017. 220 с.

13. Касатов Д. В. Основи програмування мовою Python : навч. посіб. Київ : Основа, 2021. 312 с.

14. Олійник Ю. В. Обробка даних з використанням Python : навч. посіб. Харків : ХНУРЕ, 2022. 154 с.

15. Петренко І. О., Лісова О. В. Інформаційні технології у навчанні : навч. посіб. Київ : Кондор, 2020. 295 с.

16. Андрієнко І. І., Шевчук П. В. Основи комп'ютерного моделювання : навч. посіб. Київ : ВЦ КНУ, 2019. 350 с.

17. Петриченко О. С. Основи програмування та алгоритми : підручник. Львів : Сполом, 2018. 298 с.

18. Коваль В. О. Розробка інформаційних систем : навч.-метод. посіб. Одеса : Видавництво ОДУ, 2017. 210 с.

19. Шевченко Л. В. Основи баз даних : навч. посіб. Київ : Дакор, 2019. 370 с.

20. Гавриленко С. М. Алгоритми та структури даних : навч. посіб. Харків : ПП "Електрокнига", 2020. 380 с.

21. Винник О. І., Поляков О. В. Статистичний аналіз даних у Python : навч. посіб. Київ : Наукова думка, 2021. 240 с.

22. Антонова І. С. Розробка програмного забезпечення для аналізу даних : навч. посіб. Одеса : ОНПУ, 2021. 165 с.

23. Ковальов М. І. Технології обробки великих даних : підручник. Київ : ВЦ КНТУ, 2018. 430 с.

24. Шаповалов О. А. Основи програмування на Python : навч.-метод. посіб. Львів : Вид-во НУ "Львівська політехніка", 2020. 250 с.

25. Малюта А. П., Данилов Р. О. Інформаційні системи в управлінні : навч. посіб. Київ : Центр учбової літератури, 2019. 340 с.

26. Смирнов В. І., Ткаченко С. С. Основи інформатики та обчислювальної техніки : навч. посіб. Київ : Академвидав, 2021. 220 с.

27. Береза А. О., Ніколаєнко Т. В. Програмування та обробка даних у Python : навч. посіб. Харків : Вид-во ХНУ ім. В.Н. Каразіна, 2019. 175 с.

28. Юрченко П. М. Програмні засоби аналізу даних : монографія. Дніпро : Видавництво ДДУ, 2018. 310 с.

29. Тимошенко О. І. Основи сучасних інформаційних технологій : навч. посіб. Одеса : Вид-во ОДЕКУ, 2021. 280 с.

30. Левченко А. П. Моделювання та аналіз даних у Python : навч. посіб. Київ : Центр навчальної літератури, 2022. 202 с.

31. Грущенко О. М., Лобач В. І. Інформатика для економістів : навч. посіб. Київ : КНЕУ, 2020. 330 с.

32. Ткачук І. В. Основи програмування : підручник. Київ : Центр учбової літератури, 2018. 450 с.

33. Савчук О. В. Технології розробки програмного забезпечення : навч. посіб. Харків : ХНУРЕ, 2021. 300 с.

34. Глушко С. П. Системний аналіз і управління : навч. посіб. Київ : Вид-во Київського нац. ун-ту, 2019. 225 с.

35. Коломієць Т. А., Семененко О. В. Python для аналітики : навч.-метод. посіб. Львів : Літопис, 2020. 198 с.

36. Шевченко С. О. Розробка баз даних для навчальних систем : монографія. Харків : ХНУ, 2019. 270 с.

37. Мирошник О. І. Системи підтримки прийняття рішень : підручник. Київ : Либідь, 2018. 365 с.

38. Василенко О. В. Аналіз даних та машинне навчання з Python : навч. посіб. Дніпро : ДНУ, 2022. 290 с.

39. Лисенко І. І., Бойко В. І. Комп'ютерні системи та мережі : навч. посіб. Одеса : Видавництво ОНПУ, 2020. 390 с.

40. Колесник М. Г. Програмування об'єктів та алгоритмів : навч. посіб. Київ : Центр учбової літератури, 2017. 245 с.

41. Моркун Н. В., Завсегдашня І. В. методичнів казівки до виконання кваліфікаційної роботи за другим (магістерським) освітнім рівнем для здобувачів спеціальності 122 комп’ютерні науки усіх форм навчання. Кривий Ріг, КНУ, 2020. 23 с.

42. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

43. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

44. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)

45. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).