МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Олексій Горський
“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО РІВНЯ
“МАГІСТР”

Тема: “Програмний засіб «Система контролю успішності навчання» з розширеними можливостями”

Виконавець: Манжула Ксенія Олександрівна

Керівник: к.т.н Горський Олексій Миколайович

Нормоконтролер: асс Кравченко Ольга Сергіївна

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олексій Горський
"___" _____ 2023р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студентки
Манжули Ксенії Олександрівни

1. Тема проекту: “Програмний засіб «Система контролю успішності навчання» з розширеними можливостями” затверджена наказом ректора від 29.09.2023 р. № 1994/ст.

2. Термін виконання проекту: з 02.10.2023р. до 31.12.2023р.

3. Вихідні данні до проекту: розробити програмний продукт за допомогою інтегрованого середовища розробки QtDesigner.

4. Зміст пояснювальної записки:
   1. Дослідження освітньої діяльності університету з точки зору ймовірнісної оцінки рівню опанування компетеностей студентом.
   2. Проектування архітектури програмного засобу.
   3. Структура програмного засобу.
   4. Програмний засіб.

5. Перелік обов'язкового графічного матеріалу:
   1. Логічна архітектура підсистем.
   2. Модель бази даних.
   3. Результат роботи програмного засобу.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Ознайомлення з постановкою задачі та вивчення літератури Написання 1 розділу, представлення керівнику	02.10.2023 – 09.10.2023	
2.	Написання 2 розділу, представлення керівнику	09.10.2023 – 15.10.2023	
3.	Написання 3 розділу, представлення керівнику	16.10.2023 – 22.10.2023	
4.	Написання 4 розділу, представлення керівнику	23.10.2023 – 29.10.2023	
5.	Загальне редагування та друк пояснювальної записки, графічного матеріалу. Проходження антиплагіату.	До 11.12.2023	
6.	Проходження нормо-контролю, перепліт пояснювальної записки. Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	До 11.12.2023	
7.	Передзахист.	11.12.2023 – 15.12.2023	
8.	Отримання відгуку керівника, рецензії.	До 08.11.2023	
9.	Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, ГМ, CD-R з електронними копіями ПЗ, ГМ, презентації, відгук керівника, рецензія, довідка про успішність, 1 папка, 1 конверт)	До 08.11.2023	
10.	Захист дипломної роботи перед ЕК.	До 25.12.2023	

Дата видачі завдання 02.10.2023 р.

Керівник: к.т.н Олексій ГОРСЬКИЙ

Завдання прийняв до виконання: Ксенія МАНЖУЛА

РЕФЕРАТ

Пояснювальна записка до дипломного проекту “Програмний засіб «Система контролю успішності навчання» з розширеними можливостями»: 147 с., 42 рис. , 20 табл., 35 інформаційних джерел.

КОМПЕТЕНТНОСТІ, ПОТОЧНА УСПІШНІСТЬ, КОНТРОЛЬНА УСПІШНІСТЬ, ВАРІАЦІЙНИЙ РЯД, ГЕНЕРАЛЬНА СЕРЕДНЯ ВАРІАЦІЙНОГО РЯДУ, НЕЧІТКА МНОЖИНА, НЕЧІТКА ГРУПА

Об’єкт розробки – застосунок, що зберігає дані успішності студентів, а також на основі цих даних, базових понять статистики та теорії нечітких множин формує ймовірний висновок про рівень опанування студентом обраних компетентностей.

Мета роботи – отримати ймовірні результати перевірки рівню компетентності студента.

Метод дослідження – розробка ймовірнісного методу оцінки стану компетенцій студента та відповідного ПЗ, що реалізує роботу даного методу.

Для реалізації поставленого завдання, що має нетривіальний характер, прийнято рішення використовувати базові поняття статистика та теорії нечітких множин.

Результат роботи – готове ПЗ для проведення ймовірнісної оцінки компетентностей (компетенцій) студента на основі даних успішності СКУН - може бути використане вищими навчальними закладами як для відповідно отримання ймовірнісної статистики компетентностей студентів, так і для адміністративного контролю над даними успішності та даними основних елементів навчального середовища.

ABSTRACT

Explanatory note to the thesis "The Learning Progress Monitoring System software tool with advanced features": 47 pp., 42 figures, 20 tables, 35 information sources.

COMPETENCIES, CURRENT PERFORMANCE, CONTROL PERFORMANCE, VARIATION SERIES, GENERAL AVERAGE OF THE VARIATION SERIES, FUZZY SET, FUZZY GROUP

The object of development is an application that stores student performance data and, based on this data, basic concepts of statistics and fuzzy set theory, forms a probable conclusion about the level of student mastery of selected competencies.

The purpose of the work is to obtain probable results of checking the level of student competence.

The research method is the development of a probabilistic method for assessing the state of student competencies and the corresponding software that implements this method.

To realize the task, which is non-trivial, it was decided to use the basic concepts of statistics and fuzzy set theory.

The result of the work - ready-made software for probabilistic assessment of student competencies based on the data of the SCUN - can be used by higher education institutions both for obtaining probabilistic statistics of student competencies and for administrative control over the data of academic performance and data of the main elements of the learning environment.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

СКУН – Система контролю успішності навчання (назва застосунку)

CRUD – CREATE/READ/UPDATE/DELETE операції

ВНЗ – вищі навчальні заклади

БД – база даних

ПУ – поточна успішність

КУ – контрольна успішність

ВРПУ – варіаційний ряд поточної успішності

ВРКУ – варіаційний ряд контрольної успішності

ГС – генеральна середня

ГСВРПУ – генеральна середня варіаційного ряду поточної успішності

ГСВРКУ – генеральна середня варіаційного ряду контрольної успішності

НГ – нечітка група

ВСТУП

На сьогоднішній день основним та найбільш важливим компонентом ринку праці є спеціалісти у різних галузях людської діяльності. При чому, під терміном «спеціаліст» у даній роботі мається на увазі той, хто має відповідну спеціальність та достатній рівень компетентності для виконання різноманітних завдань своєї спеціальності, які включають як типові завдання, так і нестандартні.

Очевидним є те, що у якості майбутніх спеціалістів для різних компаній виступають студенти. Стандартним методом, що використовується різними компаніями для того, аби визначити рівень компетентності студента та його відповідність вимогам самої компанії, є співбесіда. Однак, як показала практика останніх років, назавжди у працедавця є можливість провести співбесіду навіть у режимі онлайн. А таких випадках, коли співбесіда неможлива, працедавець хотів би мати хоча б ймовірнісні дані про можливий рівень компетентності можливого майбутнього спеціаліста. Джерелом таких даних можуть стати самі ВНЗ.

З іншого боку, такі дані є актуальними та бажаними і для ВНЗ, оскільки знання про рівень компетентностей студентів дає можливість робити висновки про актуальність існуючих освітніх програм та, відповідно, дає можливість відредагувати освітні програми під актуальні потреби ринку праці. За замовчуванням, у ВНЗ використовується система оцінювання студента (ECTS), яка не дає інформацію про компетентність.

Даний дипломний проект є логічним продовженням мого попереднього дипломного проекту на тему «Мобільний застосунок оцінки і візуалізації стану поточної успішності студента університету», тому його задачею так само являється ймовірнісний аналіз компетентності студента. Однак, оскільки даний проект є не тільки логічним продовженням, але і повною реконструкцією, то задачу даного проекту, актуальність якої доказана вище, можна сформувати наступним чином: розробка застосунку,

який би на основі існуючих даних успішності (як поточної, так і контрольної) проводив аналіз рівню обраних компетентностей студента, та представляв ймовірнісні результати даного аналізу у зручній формі (таблиця та електронний звіт у форматі .docx). Оскільки застосунок повинен проводити аналіз на основі даних успішності студента, то, відповідно, другорядною задачею є забезпечення можливості для оперування цими даними, а також ще однією другорядною задачею є забезпечення можливості для оперування даними адміністративних одиниць системи. Під адміністративними одиницями, у даній роботі маються на увазі елементи, що приймають участь у процесі оцінювання студентів та у процесі аналізу компетентностей.

Метою роботи є формування на основі існуючої освітньої системи (компетентнісного підходу зокрема) архітектури майбутнього застосунку для ймовірного аналізу компетентностей студента, відповідне проектування та його реалізація.

Об’єктом дослідження в роботі виступає процес трансформації даних успішності студента (поточна та контрольна успішність) у ймовірнісну оцінку обраної компетентності/компетентностей студента. Другорядними об’єктами дослідження виступають адміністративні процеси над успішністю студентів, адміністративні процеси над операційними одиницями майбутньої системи, а також процеси представлення даних операційних одиниць та результатів аналізу у вигляді .docx файлів.

Основними завданнями роботи є:

1. формування вербальних моделей операційних одиниць;
2. формування списку задач застосунку;
3. визначення вимог до СКУН (функціональні/нефункціональні/системні);
4. формування та обґрунтування архітектурних рішень;
5. формування переліку програмних компонентів СКУН із їх відповідним описом;
6. опис структури віддаленої бази даних;

7. реалізація раніше визначених програмних компонентів;
8. демонстрація результатів роботи готового застосунку;

Елементи **наукової новизни** роботи включають в себе не лише сформований автором даної роботи ймовірнісний метод оцінки стану компетенцій студента, але і отримані на основі сформованого автором методу, який включає використання базових елементів статистики та нечітких множин, ймовірнісні показники вибіркових компетентностей студентів, а також текстові еквіваленти даних показників та зручний вид представлення результатів аналізу (формат .docx). Під вибірковими у даній роботі мається на увазі те, що ВНЗ мають можливість самим формувати список компетентностей, та, відповідно, обирати для яких з них буде проводитися ймовірнісний аналіз.

Практична значимість роботи полягає в розробці унікального на сьогоднішній день програмного забезпечення, яке можна використовувати для отримання ймовірнісних даних про рівень опанування студентом обраних компетентностей, список яких визначає ВНЗ самостійно. До того ж, застосунок СКУН можливо використосувати у якості адміністративного інструменту для керування інформаційними одиницями у ВНЗ та автоматизації роботи із даними успішності із можливістю експорту даних у вигляді електронних таблиць. Даний застосунок може дати можливість ВНЗ гнучко налаштовувати свої навчальні програми від актуальні потреби ринку праці, та, одночасно з цим, працедавці можуть отримати ймовірнісну інформацію про рівень компетентності цікавлячого спеціаліста. Завдяки цьому ВНЗ будуть мати можливість формувати у студентів компетенції (а також ймовірнісно оцінювати стани), що мають аткуальність у конкретний період часу на ринку праці, що дасть змогу підвищити конкурентоспроможність самих студентів, та, з деякою ймовірністю, може підвищити рівень задоволення потреб ринку праці.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ОСВІТНЬОЇ ДІЯЛЬНОСТІ УНІВЕРСИТЕТУ З ТОЧКИ ЗОРУ ЙМОВІРНІСНОЇ ОЦІНКИ РІВНЮ ОПАНУВАННЯ КОМПЕТЕНТНОСТЕЙ СТУДЕНТОМ

1.1. Опис основних операційних одиниць системи

Аналіз структури декількох вітчизняних університетів та опис відповідних вербальних моделей було зроблено у попередній роботі [1]. На основі даних цього аналізу, можна зробити висновок про те, що у більшості випадків інформаційна структура різних ВНЗ складається із набору типових компонентів, тобто операційних одиниць, що взаємодіють одне з одним. Результатом вищезазначеної взаємодії є саме освітній процес.

У даній роботі, під інформаційною одиницею мається на увазі матеріальна чи нематеріальна сутність, яка може здійснювати певні операції над іншими сутностями, бути ціллю операцій інших сутностей, а також має певний набір інформаційних полів, що характеризують дану сутність.

Відповідно до згаданого вище аналізу, існує можливість стверджувати, що інформаційна структура практично будь-якого університету складається із наступних операційних одиниць:

1. Факультет;
2. Кафедра;
3. Викладач;
4. Група;
5. Студент;
6. Дисципліна;
7. Оцінка;

Факультет – це операційна одиниця, що відображає реальний факультет, який входить до складу університету. Один університет, як правило, включає декілька факультетів.

Кафедра – це операційна одиниця, що відображає реальну кафедру, яка входить до факультету. Один факультет, як правило, включає декілька кафедр.

Викладач – це операційна одиниця, що відображає реального викладача університету. Один викладач, як правило, офіційно може значитися лише за однією кафедрою.

Група – це інформаційна одиниця, що позначає реальну групу студентів. Як правило, група може значитися лише за однією кафедрою.

Студент – це операційна одиниця, що позначає реального студента університету. Як правило, один студент може значитися лише в одній групі.

Дисципліна – це операційна одиниця, що позначає реальну навчальну дисципліну, що викладається студентам. Як правило, одна дисципліна може числитися лише за однією кафедрою.

Оцінка – це операційна одиниця, що позначає реальну оцінку деякого типу, яку отримує деякий студент від деякого викладача за деяку дисципліну.

Однак, відповідно до поставленого завдання та до вимоги «Універсальності використання» (детально про вимоги описано у розділі 2), вищезазначений список операційних одиниць можна змінити, а саме доповнити та об’єднати деякі види одиниць в єдину групу. Таким чином, до списку додаються такі елементи як «Університет», «Компетентність / Компетенція», «Локальний адміністратор» та «Глобальний адміністратор». Одночасно з цим, одиниці «Глобальний адміністратор», «Локальний адміністратор» та «Викладач» поєднуються в єдину групу «Користувачі системи».

Університет – це операційна одиниця, яка позначає реальний університет. Дана одиниця введена через наявність вимоги «Універсальність використання».

Компетентність / компетенція – це операційна одиниця, яка позначає компетентність, для якої проводиться підрахунок. Кожний ВНЗ, як правило, сам визначає список компетентностей, який культивують студенти під час

навчання. Дана одиниця введена через загальний напрям застосунку (основна задача якого – робота з компетентність).

Глобальний адміністратор – це інформаційна одиниця, що позначає людину, що займається адміністративним контролем роботи всієї системи. У даному випадку, під адміністративним контролем мається на увазі адміністративне управління списком університетів, факультетів, кафедр та локальних адміністраторів. Дана сутність є унікальною – система має лише одного глобального адміністратора, та введена через наявність вимоги «Універсальність використання».

Локальний адміністратор – це інформаційна одиниця, що позначає людину, що займається адміністративним контролем тих аспектів системи, що пов’язані тільки із тим університетом, за яким закріплено локального адміністратора. У даному випадку, під адміністративним контролем мається на увазі адміністративне управління списком груп, студентів, викладачів, дисциплін, компетентностей та здійснення ймовірнісного аналізу компетентностей. Як вже було зазначено раніше, один локальний адміністратор повинен бути закріплений лише за одним університетом. Дана сутність введена через наявність вимоги «Універсальність використання».

Опис інформаційних полів вищезазначених операційних одиниць представлено у таблиці 1.1.

Таблиця 1.1.

Вербальний опис операційних одиниць

Операційна одиниця	**Інформаційні поля**
1	**2**
Університет	Назва
Факультет	Назва, університет
Кафедра	Назва, факультет
Група	Номер, кафедра, факультет, університет

Продовження таблиці 1.1.

1	2
Студент	ПІП, група, кафедра, факультет, університет
Дисципліна	Назва, список викладачів, кафедра, факультет, університет
Компетентність	Назва, список ключових дисциплін, кафедра, факультет, університет
Оцінка	Викладач, студент, тип оцінки, дисципліна, дата виставлення, коротке значення (3-5), довге значення (60-100, за ECTS)
Глобальний адміністратор	Логін, пароль, ПІП
Локальний адміністратор	Логін, пароль, ПІП, університет
Викладач	Логін, пароль, ПІП, кафедра, факультет, університет

1.2. Вербальна модель логічної структури СКУН

СКУН призначена для проведення аналізу рівню опанування студентом певного набору компетентностей та представленням на основі цього аналізу ймовірнісних даних компетентності студента у двох форматах: у числовому та у текстовому (текстовий висновок). Додатково, СКУН може слугувати у якості інструменту адміністративного управління (CRUD - операції) над даними основних для ВНЗ операційних одиниць. СКУН підтримує наступні види кінцевих користувачів: глобальний адміністратор, локальний адміністратор, викладач.

СКУН фактично є складеною структурою, що складається із наступних компонентів:

- Підсистема глобального адміністратора – відповідає за реалізацію функціоналу глобального адміністратора;

- Підсистема локального адміністратора – відповідає за реалізацію фунціоналу локального адміністратора;
- Підсистема викладача – відповідає за реалізацію функціоналу викладача (тобто формування та оновлення даних успішності).

Як можна побачити із переліку вище, як і у випадку попереднього проекту ([1]), компоненти системи розділені відповідно до видів кінцевих користувачів самої системи. Однак у даному випадку різниця полягає у самих групах. Попередній проект налічував наступні групи користувачів: «Адміністратор», «Викладач» та «Студент». СКУН, на відміну від своєї попередньої версії, має більш адміністративний характер та спрямована на використання лише адміністрацією ВНЗ (через відповідного локального адміністратора, який входить до адміністрації ВНЗ).

Кожна із систем, представлених вище (за виключенням підсистеми викладача), включає в себе набір дочірніх підсистем.

Підсистема глобального адміністратора включає в себе наступні дочірні підсистеми:

- Підсистема університетів – відповідає за здійснення транзакцій над операційними одиницями типу «Університет»;
- Підсистема факультетів – відповідає за здійснення транзакцій над операційними одиницями типу «Факультет»;
- Підсистема кафедр – відповідає за здійснення транзакцій над операційними одиницями типу «Кафедра»;
- Підсистема локальних адміністраторів – відповідає за здійснення транзакцій над операційними одиницями типу «Локальний адміністратор».

Підсистема локального адміністратора включає в себе наступні дочірні підсистеми:

- Підсистема груп – відповідає за здійснення транзакцій над операційними одиницями типу «Група»;

- Підсистема студентів – відповідає за здійснення транзакцій над операційними одиницями типу «Студент»
- Підсистема викладачів – відповідає за здійснення транзакцій над операційними одиницями типу «Викладач»;
- Підсистема дисциплін – відповідає за здійснення транзакцій над операційними одиницями типу «Дисципліна»;
- Підсистема компетентностей – відповідає за здійснення транзакцій над операційними одиницями типу «Компетентність/Компетенція»;
- Підсистема аналізу – відповідає за здійснення аналізу рівню опанування студентом компетентностей.

Як можна зрозуміти із вищезазначеного, структура компонентів СКУН має ієрархічний, деревоподібний характер. Графічно це відображено на рисунку 1.1.

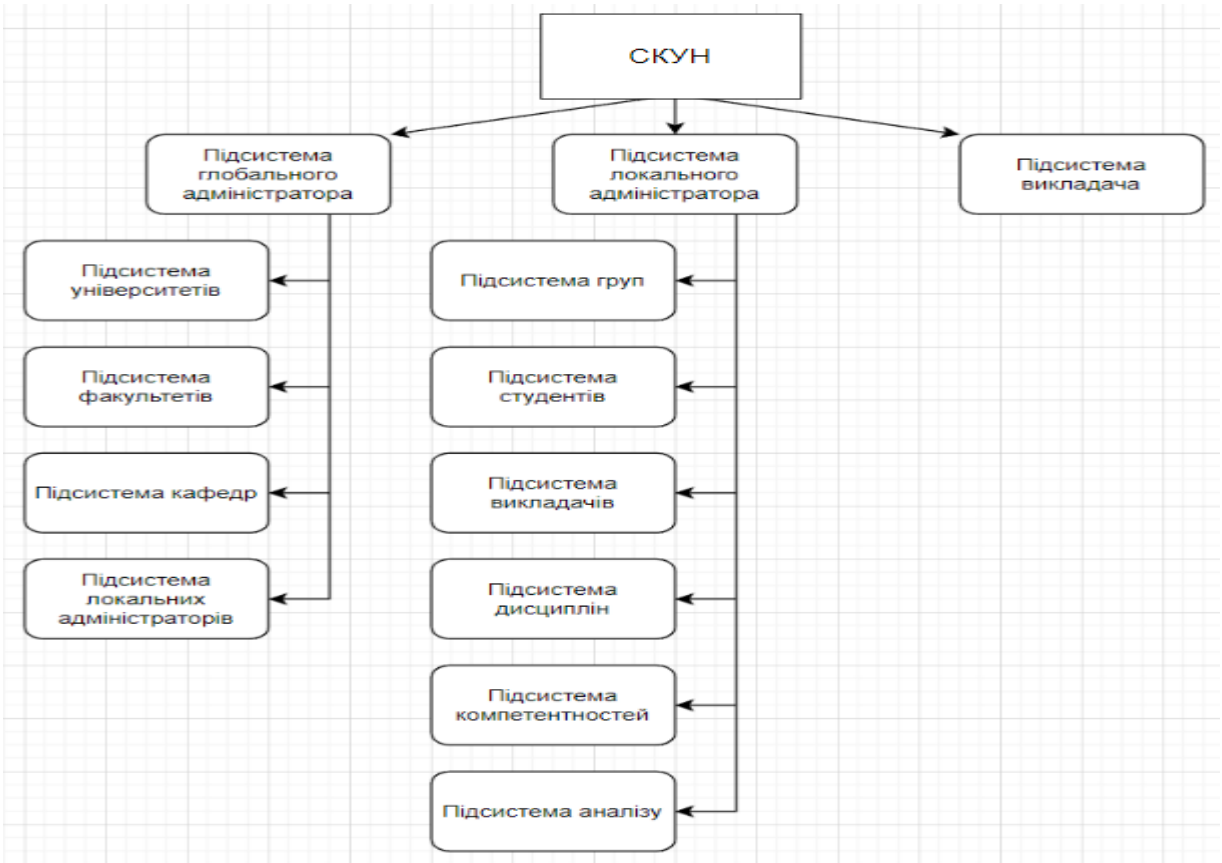


Рис. 1.1. Логічна структура СКУН (архітектура підсистем)

1.3. Аналіз програмного забезпечення із схожим функціоналом

Як вже було зазначено у вступі, застосунок СКУН, як і його попередня версія, є унікальним за своїм функціоналом. Аналіз існуючих на сьогоднішній день програмних систем ([2-8]), що призначені для оцінювання студентів та зберігання даних їх успішності показав, що хоча програмних засобів саме для оцінки та зберігання даних, тобто для адміністративної роботи над даними успішності, дуже багато, але ні один із досліджених засобів не має функціоналу роботи із компетентностями студентів.

Однак, другорядною задачею СКУН є саме адміністративний контроль над даними успішності студентів. Із цієї причини, на рисунку А.1. у додатку А представлено, сформовану на основі аналізу [2-13], порівняльну таблицю популярних на сьогоднішній день програмних систем для оцінювання студентів, що здатні оперувати даними успішності.

1.4. Онтологічні моделі предметної області

На основі сформованої раніше вербальної моделі логічної структури застосунку СКУН (див. пункт 1.2.), можна сформувати перелік онтологічних моделей (а саме онтологію задач системи, онтологію процесів системи та онтологію об’єктів системи), які дають змогу чітко визначити вимоги до самого застосунку.

Онтологія задач системи представлена нижче на рисунку 1.3. Як можна побачити на даному рисунку, кінцеві задачі діляться на два напрями: огранізааційний (планування, організація та контроль стану СКУН) та виконавчий (сбір/обробка/видалення даних успішності).

Онтологія процесів системи представлена нижче на рисунку 1.4. Як і у випадку із задачами, процеси можна розділити на групи, а саме на три групи: процеси організації учбових процесів та оновлення СКУН, процеси контролю над даними успішності (сбір/обробка/відображення/видалення даних поточної/контрольної успішності) та процеси ймовірнісного розрахунку компетенцій студента (формування списку цільових дисциплін,

розрахунок певних показників (ВРПУ, ВРКУ, ГСВРПУ, ГСВРКУ) та процеси аналізу отриманих показників (перевірка ГСВРПУ та ГСВРКУ на належність визначеним нечітким множинам).

Онтологія об’єктів системи представлена нижче на рисунку 1.5. Як можна побачити на даному рисунку, об’єктами онтології виступають описані раніше (див. пункт 1.1.) основні операційні одиниці, а також додатков операційні одиниці, які є елементами існуючої системи оцінювання (такі як різного виду документація, різні учбові події, тощо).

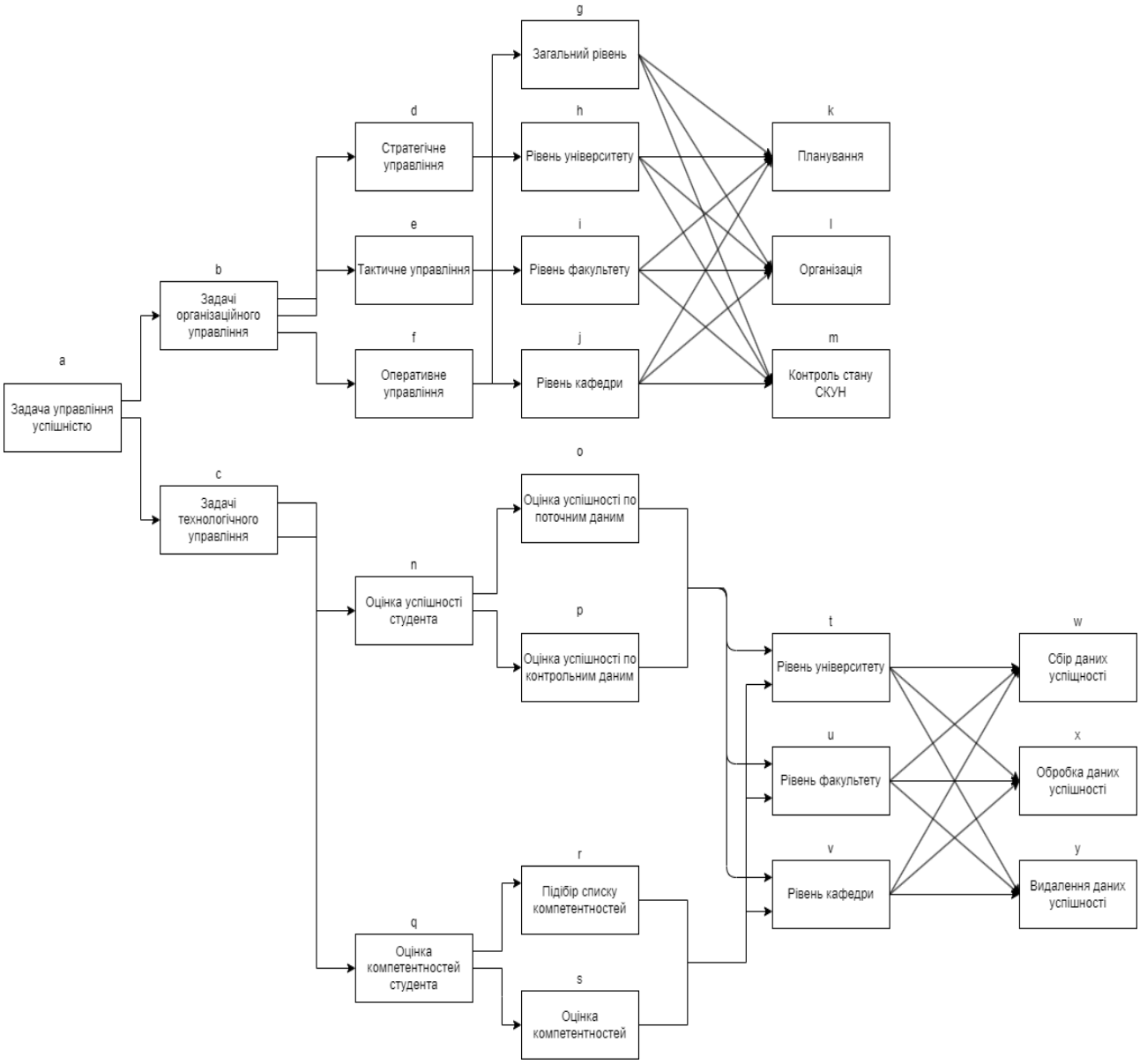


Рис. 1.3. Онтологія задач СКУН

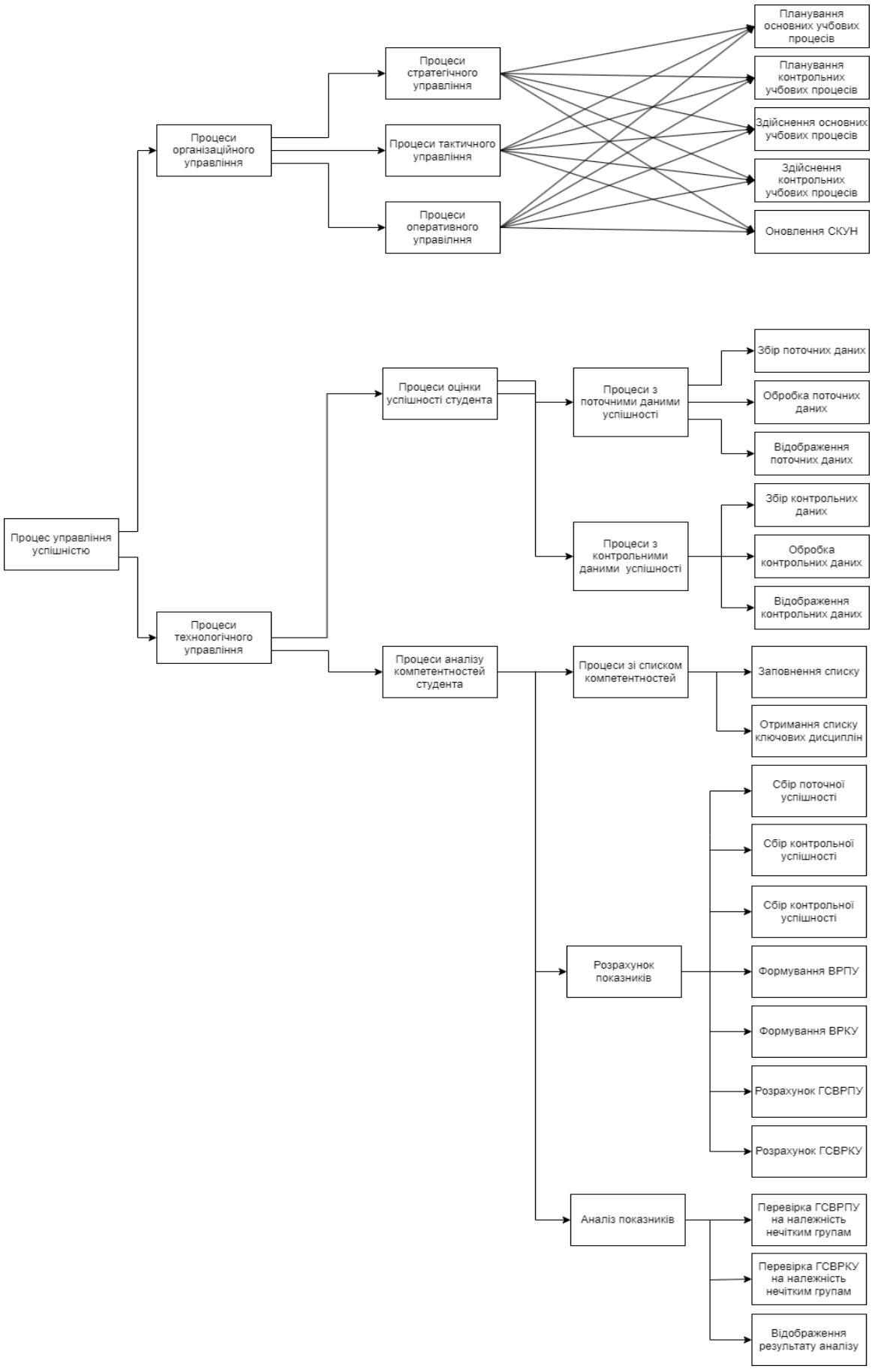


Рис. 1.4. Онтологія процесів СКУН

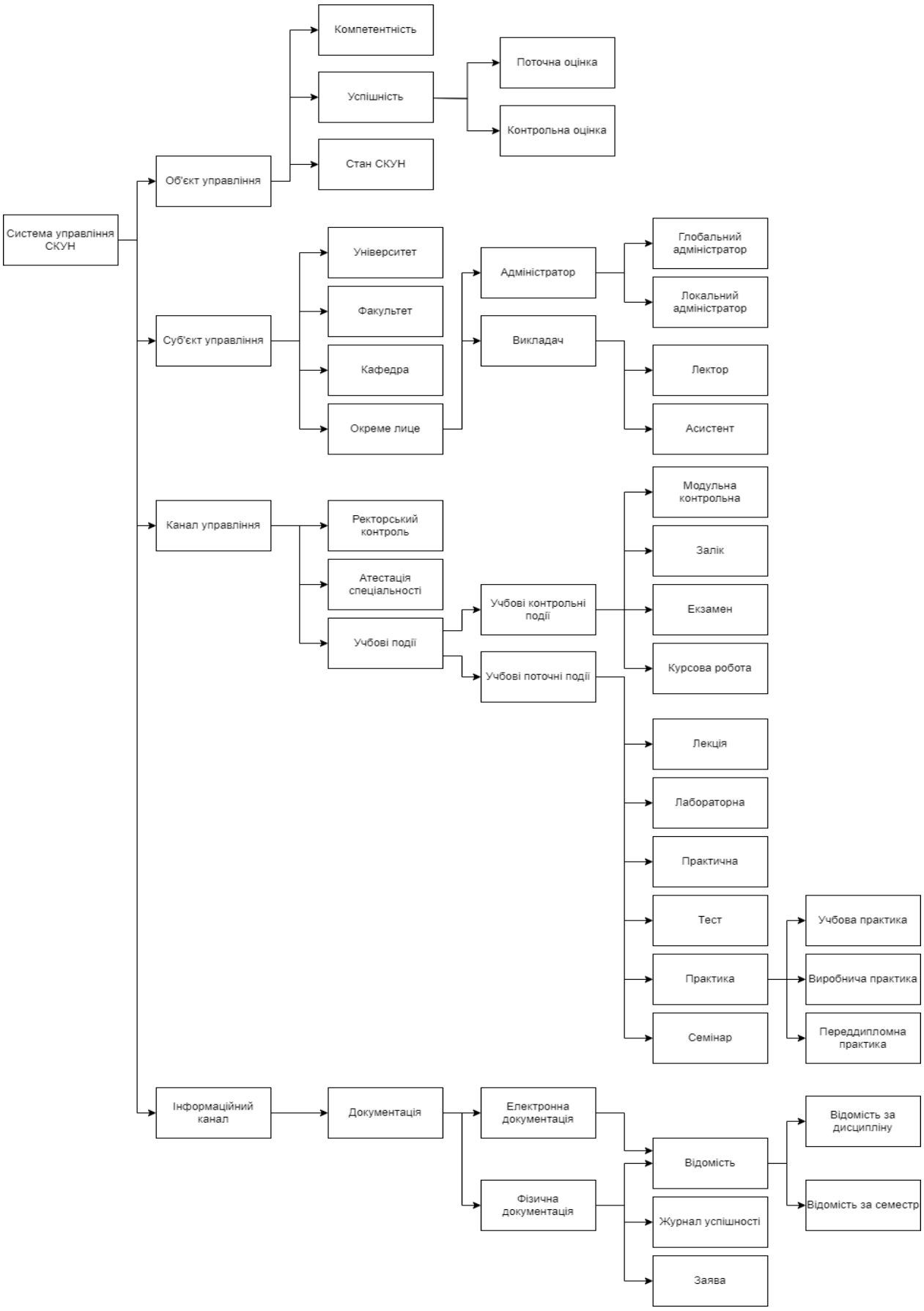


Рис. 1.5. Онтологія об’єктів СКУН

Висновки

У даному розділі було розглянуто освітню діяльність середовища, у якому планується використовувати застосунок СКУН, а саме освітню діяльність ВНЗ. Під час дослідження було визначено основні операційні одиниці, виділено логічні зв’язки між ними та визначено перелік характеристик, притаманних кожній операційній одиниці (із урахуванням виділених раніше зв’язків).

На основі цього дослідження було сформовано вербальну модель майбутньої інформаційної системи. СКУН має дворівневу ієрархічну структуру, що побудована відповідно до груп ролей майбутніх користувачів.

Для первинного визначення вимог, на основі сформованої раніше вербальної моделі, було використано онтологічний підхід до проектування, тобто побудовано три онтології (онтологія задач, онтологія процесів та онтологія об’єктів), що характеризують операційне середовище (роботу ВНЗ) з уточненням у бік роботи із компетентностями студентів.

З точки зору розробки та інтеграції компонентів, даний проект має середню складність реалізації, оскільки запланований для реалізації функціонал, у більшій мірі, стосується адміністративного управління даними успішності студентів та даними операційних одиниць.

З точки зору потенціалу даний проект має високу складність, адже аналіз компетентностей студентів є новим напрямком, і подібного ПЗ на даний момент не існує. Це доказується аналізом відомих програм-аналогів, що наведено у пункті 1.3.

Беручи до уваги вищезазначене, можна зробити припущення, що розробка даного проекту може зайняти приблизно 3-4 місяці. Термін може бути подовжено, якщо врахувати нові користувацькі вимоги.

РОЗДІЛ 2
ПРОЕКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ СКУН

2.1. Проектування логічної архітектури

На відміну від обраного для попередньої версії застосунку СКУН багаторівневого шаблону проектування, у якості архітектурного шаблону проектування для даного дипломного проекту обрано архітектурний патерн MVC.

Дане рішення було прийнято через зміну цільової платформи, мови програмування та фреймворку.

Однак зауважимо, що особливості обраної для використання віддаленої бази даних (Firebase) та фреймворку (PySide) дають змогу провести адаптацію існуючого архітектурного патерну. Таким чином, оскільки Firebase є No-SQL базою даних, а PySide підтримує можливість занесення даних у БД напряму без використання об’єктів-моделей, обраний архітектурний шаблон спрощується методом вилучення моделей із нього (або, іншими словами, моделі поєднуються із контролерами). Таким чином, застосунок СКУН повинен складатися лише з представлень (View) та контролерів (Controller). Детальніше про обрані архітектурні аспекти зазначено у пункті 3.1.

Раніше зазначалося (пункт 1.2.), що СКУН має ієрархічну структуру, основу якою складають три основні підсистеми, що відповідають ролям користувачів:

- Підсистема глобального адміністратора;
- Підсистема локального адміністратора;
- Підсистема викладача;

Також, раніше зазначалося, що дані підсистеми включають в себе набір дочірніх підсистем.

Відповідно, підсистема глобального адміністратора включає в себе наступні дочірні підсистеми:

- Підсистема університетів;

- Підсистема факультетів;
- Підсистема кафедр;
- Підсистема локальних адміністраторів;

Підсистема локального адміністратора включає в себе наступні дочірні підсистеми:

- Підсистема груп;
- Підсистема студентів;
- Підсистема викладачів;
- Підсистема дисциплін;
- Підсистема компетентностей;
- Підсистема аналізу.

Графічно, дана архітектура виглядає таким чином, як це представлено на рисунку 2.1.

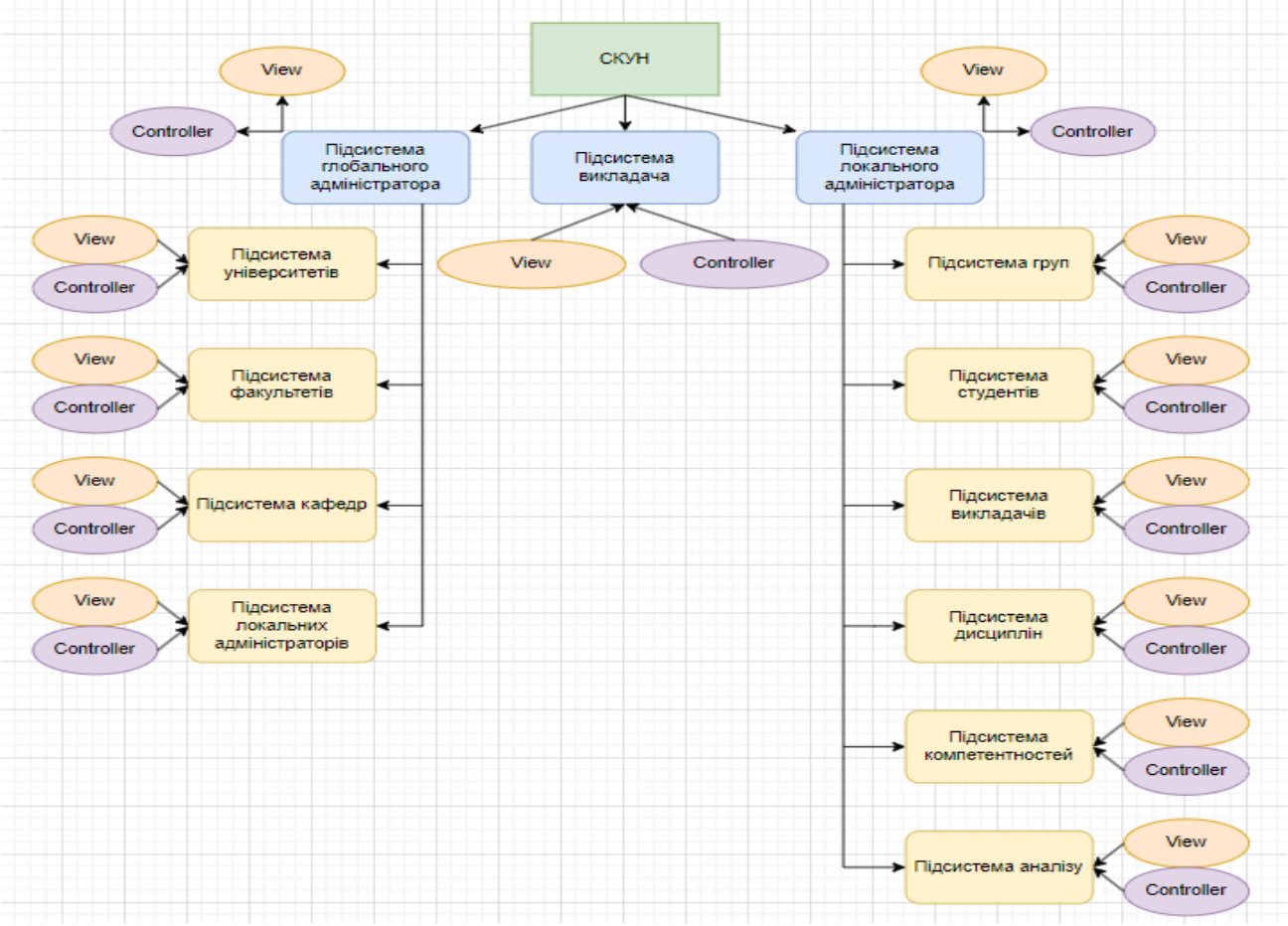


Рис. 2.1. Логічна архітектура СКУН

Як можна побачити на рисунку вище, логічна архітектура включає в себе не тільки перелік підсистем СКУН, але і, відповідно до обраного архітектурного патерну, представлення та контролери для кожної системи та підсистеми у застосунку.

Представлення (View) – це компонент, що відповідає за представлення даних (а також отримання даних) користувачеві [10-14]. Зазвичай, дані беруться із відповідної моделі (моделей), однак у даній роботі представлення отримують дані від контролерів.

Контролери (Controller) – це компонент, що відповідає за здійснення бізнес-логіки застосунку [10-14]. Він отримує сигнали від представлень та здійснює відповідну реакцію.

Також, на даній схемі можна побачити, що підсистеми не впливають на роботу одне одного, що дозволяє істотно зменшити залежності у системі.

Таким чином, базуючись на тому, що представлена логічна структура системи дотримується принципів єдиної відповідальності (Single Responsibility із SOLID) та мінімізації залежностей, можна зробити висновок, що логічна система є коректною та є придатною до програмної реалізації.

2.2. Функціональні вимоги

На основі сформованої раніше логічної архітектури (див. пункт 2.1.), опису основних операційних одиниць (див. пункт 1.1.) та описі вербальної моделі логічної структури (див. пункт 1.2.) для системи СКУН можна виділити перелік функціональних вимог, який представлений у таблиці 2.1.

Таблиця 2.1.

Перелік функціональних вимог до СКУН

№	Вимога
1	**2**
Фв1	У застосунку повинна бути передбачена авторизація
Фв2	У застосунку не повинна бути передбачена пряма реєстрація

	нових користувачів

Продовження таблиці 2.1.

1	**2**
Фв3	У застосунку повинно бути реалізоване інформаційне меню головного адміністратора
Фв4	В інформаційному меню головного адміністратора повинен бути реалізований перехід до інших підсистем: підсистеми університетів, підсистеми факультетів, підсистеми кафедр, підсистеми локальних адміністраторів (меню цих підсистем)
Фв5	В інформаційному меню головного адміністратора повинна бути реалізована статистика підконтрольних операційних одиниць
Фв6	У системі повинно бути реалізоване меню університетів
Фв7	У меню університетів повинна бути реалізована можливість створення нового університету
Фв8	У меню університетів повинна бути реалізована можливість оновлення обраного університету
Фв9	У меню університетів повинна бути реалізована можливість видалення обраного університету
Фв10	У меню університетів повинна бути реалізована можливість перегляду списку всіх університетів у вигляді таблиці
Фв11	У меню університетів повинна бути реалізована можливість експорту даних всіх університетів у відповідну таблицю (.xsls)
Фв12	У системі повинно бути реалізоване меню факультетів
Фв13	У меню факультетів повинна бути реалізована можливість створення нового факультету
Фв14	У меню факультетів повинна бути реалізована можливість оновлення обраного факультету

Продовження таблиці 2.1.

1	2
Фв15	У меню факультетів повинна бути реалізована можливість видалення обраного факультету
Фв16	У меню факультетів повинна бути реалізована можливість перегляду списку всіх факультетів у вигляді таблиці
Фв17	У меню факультетів повинна бути реалізована можливість експорту даних всіх факультетів у відп. табл. (.xsls)
Фв18	У системі повинно бути реалізоване меню кафедр
Фв19	У меню кафедр повинна бути реалізована можливість створення нової кафедри
Фв20	У меню кафедр повинна бути реалізована можливість оновлення обраної кафедри
Фв21	У меню кафедр повинна бути реалізована можливість видалення обраної кафедри
Фв22	У меню кафедр повинна бути реалізована можливість перегляду списку всіх кафедр у вигляді таблиці
Фв23	У меню кафедр повинна бути реалізована можливість експорту даних всіх кафедр у відповідну таблицю (.xsls)
Фв24	У системі повинно бути реалізоване меню локальних адміністраторів
Фв25	У меню лок. адміністраторів повинна бути реалізована можливість ств. нового локального адміністратора
Фв26	У меню локальних адміністраторів повинна бути реалізована можливість оновлення обраного локального адміністратора
Фв27	У меню локальних адміністраторів повинна бути реалізована можливість видалення обраного локального адміністратора

Продовження таблиці 2.1.

1	2
Фв28	У меню локальних адміністраторів повинна бути реалізована можливість перегляду списку всіх локальних адміністраторів у вигляді таблиці
Фв29	У меню локальних адміністраторів повинна бути реалізована можливість експорту даних всіх локальних адміністраторів у відповідну таблицю (.xsls)
Фв30	У застосунку повинно бути реалізоване інформаційне меню локального адміністратора
Фв31	В інформаційному меню локального адміністратора повинен бути реалізований перехід до інших підсистем: підсистеми груп, підсистеми студентів, підсистеми викладачів, підсистеми дисциплін, підсистеми компетенцій, підсистеми аналізу компетенцій (меню цих підсистем)
Фв32	В інформаційному меню локального адміністратора повинна бути реалізована статистика підконтрольних операційних одиниць
Фв33	У системі повинно бути реалізоване меню груп
Фв34	У меню груп повинна бути реалізована можливість створення нової навчальної групи
Фв35	У меню груп повинна бути реалізована можливість оновлення обраної навчальної групи
Фв36	У меню груп повинна бути реалізована можливість видалення обраної навчальної групи
Фв37	У меню груп повинна бути реалізована можл. перегляду списку всіх навчальних груп конкретного університету у вигляді таблиці

Продовження таблиці 2.1.

1	2
Фв38	У меню груп повинна бути реалізована можливість експорту даних всіх груп конкретного університету (той, за яким закріплено локального адміністратора) у відповідну таблицю (.xsls)
Фв38	У системі повинно бути реалізоване меню студентів
Фв39	У меню студентів повинна бути реалізована можливість створення нового студента
Фв40	У меню студентів повинна бути реалізована можливість оновлення обраного студента
Фв41	У меню студентів повинна бути реалізована можливість видалення обраного студента
Фв42	У меню студентів повинна бути реалізована можливість перегляду списку всіх студентів конкретного університету (той, за яким закріплено локального адміністратора) у вигляді таблиці
Фв43	У меню студентів повинна бути реалізована можливість експорту даних всіх студентів конкретного університету (той, за яким закріплено локального адміністратора) у відповідну таблицю (.xsls)
Фв44	У системі повинно бути реалізоване меню викладачів
Фв45	У меню викладачів повинна бути реалізована можливість створення нового викладача
Фв46	У меню викладачів повинна бути реалізована можливість оновлення обраного викладача
Фв47	У меню викладачів повинна бути реалізована можливість видалення обраного викладача

Продовження таблиці 2.1.

1	2
Фв48	У меню викладачів повинна бути реалізована можливість перегляду списку всіх викладачів конкретного університету (той, за яким закріплено локального адміністратора) у вигляді таблиці
Фв49	У меню викладачів повинна бути реалізована можливість експорту даних всіх викладачів конкретного університету (той, за яким закріплено локального адміністратора) у відповідну таблицю (.xsls)
Фв50	У системі повинно бути реалізоване меню дисциплін
Фв51	У меню дисциплін повинна бути реалізована можливість створення нової дисципліни
Фв52	У меню дисциплін повинна бути реалізована можливість оновлення обраної дисципліни
Фв53	У меню дисциплін повинна бути реалізована можливість видалення обраної дисципліни
Фв54	У меню дисциплін повинна бути реалізована можливість перегляду списку всіх дисциплін конкретного університету (той, за яким закріплено локального адміністратора) у вигляді таблиці
Фв55	У меню дисциплін повинна бути реалізована можливість експорту даних всіх дисциплін конкретного університету (той, за яким закріплено локального адміністратора) у відповідну таблицю (.xsls)
Фв56	У системі повинно бути реалізоване меню компетентностей
Фв57	У меню компетентностей повинна бути реалізована можливість створення нової компетентності

Продовження таблиці 2.1.

1	2
Фв58	У меню компетентностей повинна бути реалізована можливість оновлення обраної компетентності
Фв59	У меню компетентностей повинна бути реалізована можливість видалення обраної компетентності
Фв60	У меню компетентностей повинна бути реалізована можливість перегляду списку всіх компетентностей конкретного університету (той, за яким закріплено локального адміністратора) у вигляді таблиці
Фв61	У меню компетентностей повинна бути реалізована можливість експорту даних всіх компетентностей конкретного університету (той, за яким закріплено локального адміністратора) у відповідну таблицю (.xsls)
Фв62	У системі повинно бути реалізоване меню аналізу компетентностей
Фв63	У меню аналізу компетентностей повинна бути реалізована можливість проведення аналізу обраних компетентностей конкретного університету (той, за яким закріплено локального адміністратора) у обраного студента конкретного університету.
Фв64	У меню аналізу компетентностей повинна бути реалізована можливість перегляду результатів аналізу у вигляді таблиці
Фв65	У меню аналізу компетентностей повинна бути реалізована можливість експорту результатів аналізу у відповідну таблицю (.xsls)
Фв66	У застосунку повинно бути реалізоване меню викладача (не плутати із меню викладачів)

Закінчення таблиці 2.1.

1	2
Фв67	У меню викладача повинна бути реалізована можливість створення нової оцінки
Фв68	У меню викладача повинна бути реалізована можливість оновлення обраної оцінки
Фв69	У меню викладача повинна бути реалізована можливість видалення обраної оцінки
Фв70	У меню викладача повинна бути реалізована можливість перегляду списку всіх оцінок, які були виставлені саме цим викладачем у вигляді таблиці

Як можна побачити у таблиці вище, система СКУН має значну кількість функціональних вимог, що є наслідком її складної структури. Вимоги, для наочності, були розділені кольорами на групи, відповідно до основних підсистем СКУН.

2.3. Нефункціональні вимоги

Перелік нефункціональних вимог, сформованих на основі функціональних вимог (див. пункт 2.2.) представлено у таблиці 2.2.

Таблиця 2.2.

Перелік нефункціональних вимог до СКУН

№	Вимога
1	2
Нфв1	Цільова операційна система - Windows
Нфв2	Авторизація у системі повинна включати ввід користувачем логіну та паролю
Нфв3	Максимальний час виконання транзакцій повинен становити 5-7 секунд

Продовження таблиці 2.2.

1	2
Нфв4	Логіни у користувачів системи не повинні співпадати
Нфв5	Система передбачає наявність користувацького інтерфейсу
Нфв6	Користувацький інтерфейс у системі повинен бути оформлений у пастельних тонах
Нфв7	За одним університетом може бути закріплено лише одного локального адміністратора
Нфв8	Один локальний адміністратор може бути закріплений лише за одним університетом
Нфв9	Головний адміністратор не повинен мати доступу до функціоналу локального адміністратора
Нфв10	Головний адміністратор не повинен мати доступу до функціоналу викладача
Нфв11	Локальний адміністратор не повинен мати доступу до функціоналу головного адміністратора
Нфв12	Локальний адміністратор не повинен мати доступу до функціоналу викладача
Нфв13	Викладач не повинен мати доступу до функціоналу головного адміністратора
Нфв14	Викладач не повинен мати доступу до функціоналу локального викладача
Нфв15	Статистка підконтрольних операційних одиниць в інформаційному меню головного адміністратора повинна включати такі дані: дата/час, кількість університетів, кількість факультетів, кількість кафедр, кількість локальних адміністраторів

Продовження таблиці 2.2.

1	2
Нфв16	Статистка підконтрольних операційних одиниць в інформаційному меню локального адміністратора повинна включати такі дані: дата/час, кількість груп, кількість студентів, кількість викладачів, кількість дисциплін, кількість компетенцій (обмежується університетом, за яким закріплено локального адміністратора).
Нфв17	Усі меню підсистем головного адміністратора (меню університетів, меню факультетів, меню кафедр, меню локальних адміністраторів) повинні мати вбудовані поля для вводу даних (тобто, поля не повинні бути розташовані на окремих формах)
Нфв18	Усі меню підсистем локального адміністратора (меню груп, меню студентів, меню викладачів, меню дисциплін, меню компетентностей, меню аналізу компетентностей) повинні мати вбудовані поля для вводу даних (тобто, поля не повинні бути розташовані на окремих формах)
Нфв19	Перехід до різних меню повинен бути здійснений при натисканні на відповідні кнопки
Нфв20	Там, де це можливо, дані повинні заповнюватися автоматично (наприклад, у випадаючих списках)
Нфв21	Експорт даних повинен проходити у файл електронних таблиць SCA_DATA.xlsx, який за замовчуванням знаходиться у місці, де знаходиться виконавчий файл

Закінчення таблиці 2.2.

1	2
Нфв22	Якщо файлу SCA_DATA.xlsx нема у місці, де знаходиться виконавчий файл, то він, при виконанні експорту даних, повинен створюватися автоматично
Нфв23	Експорт даних проводиться у відповідні сторінки файлу SCA_DATA.xlsx
Нфв24	Якщо при експорті необхідна сторінка у файлі SCA_DATA.xlsx відсутня, то вона повинна бути створена автоматично. Назва сторінки повинна відповідати типу даних, що експортується
Нфв25	Експорт результатів аналізу повинен проходити у файл формату .docx. Даний файл повинен знаходитися у тій же директорії, в якій знаходиться виконавчий файл.
Нфв26	Файл для експорту результатів аналізу повинен мати назву, автоматично сформовану за наступним шаблоном: Прізвище Ім’я По-батькові (група) - Competence analyze result
Нфв27	Якщо файл для експорту результатів аналізу відсутній у директорії, то при здійсненні експорту результатів аналізу файл повинен створюватися автоматично
Нфв28	У якості сховища даних застосунок повинен використовувати віддалену БД Firebase

Нефункціональні вимоги у таблиці 2.2. розділені на чотири групи відповідно до функціоналу:

- вимоги, що стосуються авторизації та контролю доступу;
- вимоги, що стосуються статистики даних;
- вимоги, що стосуються оформлення маню та переходів;
- вимоги, що стосуються експорту даних.

2.4. Системні вимоги

Для коректної роботи системи СКУН, ПК повинен задовольняти наступним мінімальним системним вимогам:

- Операційна система: Windows 7;
- 86/64-bit CPU (Intel / AMD);
- 4 GB RAM;
- Процесор: 2,5 GHz.

Висновки

У даному розділі проводиться проектування логічної структури системи СКУН, яке базується на раніше описаних онтологічних моделях системи та на вербальній моделі системи у цілому. Під час проектування було обрано архітектурний патерн, на основі якого буде побудовано систему СКУН фізично, а також обґрунтовано адаптацію даного патерну під конкретні умови розробки системи СКУН із урахуванням аспектів використання зовнішніх ресурсів.

Результатом проектування є конкретна структура системи, яка побудована із врахуванням обраного архітектурного патерну.

На основі даної структури, представлено перелік функціональних вимог. Перелік, для наочності, візуально розділено на групи, відповідно до видів користувачів. Склад списку функціональних вимог дає змогу підтвердити раніше сформований висновок про складність системи СКУН з точки зору програмної реалізації.

У додаток до раніше сформованих функціональних вимог, представлено список нефункціональних вимог. Наостанок представлено мінімальні системні вимоги для коректної роботи застосунку. Дані вимоги дають змогу зробити висновок про те, що застосунок СКУН має високу ймовірність запуску на більшості ПК.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1. Архітектурні рішення

Для реалізації поставленого завдання, а точніше для розробки програмної системи СКУН та реалізації вищезазначених функціональних та нефункціональних вимог (див. пункти 2.1. – 2.4.), у якості мови програмування обрано Python.

Відповідно до статистики, зібраної у [17-20], в останні роки Python набув високого рівню популярності (1-3 місце у світі). Дана тенденція росту популярності пояснюється основними перевагами самої мови: простота для засвоєння навіть новачками, лаконічність, крос-платформенність, широка палітра безкоштовних бібліотек та велика кількість навчального матеріалу на різних мовах. У межах даного проекту використовується Python 3.10.

Як було зазначено раніше, однією із переваг Python є крос-платформенність. Однак, за замовчуванням, Python не надає можливості розробки застосунків із графічним користувацьким інтерфейсом. Тому, для задоволення потреби наявності користувацького інтерфейсу, прийнято рішення використовувати безкоштовну бібліотеку PySide6 наряду із вбудованим у дану бібліотеку конструктором QtDesigner.

PySide6 [21-23] – це безкоштовна бібліотека, яка надає можливість прив’язати мову Python до інструментарію Qt, який раніше за замовчуванням працював лише з мовою програмування C++.Дана бібліотека також є аналогом до бібліотека PyQt6, однак, на відміну від попередньої, є повністю безкоштовною.

QtDesigner [24-25] – це безкоштовний, вбудований у бібліотеку PySide6, конструктор користувацького інтерфейсу, заснований на технології Drag_And_Drop та вистежуванні сигналів. На виході даний конструктор представляє файл із розширенням .ui. Головною особливістю даного конструктора є те, що він дає можливість напряму писати HTML/CSS код для

налаштування зовнішнього вигляду та поведінки елементів користувацького інтерфейсу.

Відповідно до вимог (див. пункт 2.3.), у якості сховища даних система повинна використовувати Firebase. Firebase [26-30] – це не стільки БД, скільки сервіс, який надає можливість розробникам швидше та якісніше працювати над своїми проектами, не відволікаючись на розробку власної СУБД, оскільки практично всі функції у Firebase вже реалізовані – необхідно їх лише викликати. БД Firebase має тип No-SQL, що дає можливість зберігати дані у деревоподібній структурі. Наряду із функціоналом управління даними, Firebase надає можливість аналізу різноманітної статистики використання даних, обліку користувачів та власну вбудовану систему реєстрації/авторизації. Підтримується хостинг файлів JavaScript, HTML, CSS та інших. Через Cloud Functions реалізована динамічна підтримка Node.js. У межах даної роботи, на сервері Firebase буде створена віддалена БД, та буде використовуватися лише функціонал роботи конкретно з даними. Вбудована система авторизації/реєстрації використовуватися не буде.

Таким чином, для взаємодії із віддаленою базою на сервері Firebase, буде використовуватися безкоштовна бібліотека firebase_admin. Firebase_admin – це безкоштовна бібліотека, що надає розробникам розширений доступ до віддалених баз даних на сервері Firebase. Доступ надається на основі унікального для кожної БД секретного ключу. Дана бібліотека є повністю безкоштовною.

Також, вимоги (див. пункт 2.3.) передбачають можливість експорту даних операційних одиниць. Система СКУН повинна підтримувати два виду експорту: експорт до файлу типу .xlsx та .docx. Відповідно, для імпорту до першого формату прийнято рішення використовувати бібліотеку openpyxl, а до другого – бібліотеку python-docx.

Openpyxl [32-33] – це безкоштовна Python - бібліотека для роботи із файлами електронних таблиць з розширенням .xlsx. Дана бібліотека дозволяє

не тільки працювати із сторінками та таблицями, але і гнучко налаштовувати їх (розміри, колір, заливка, тощо).

Python-docx [34-35] – це безкоштовна Python - бібліотека для роботи із документами із розширенням .docx. Дана бібліотека дозволяє швидко створювати та гнучко редагувати електронні документи (додавання нових параграфів, форматування тексту, робот із вбудованими таблицями, зображеннями, іншими об’єктами).

У даній роботі представляється авторський алгоритм розрахунку рівню опанування компетентностей студентом. Даний метод базується на теорії нечітких множин та на базових поняттях статистики та реалізує наступний алгоритм:

- Отримуємо список дисциплін, що є першорядними та другорядними для аналізуємої компетентності;
- Для кожної із знайдених дисциплін формуємо список поточної успішності;
- Для кожної із знайдених дисциплін знаходимо список контрольної успішності;
- Визначаємо нечітке поняття;
- Знаходимо/розраховуємо ВРПУ;
- Знаходимо/розраховуємо ВРКУ;
- Знаходимо/розраховуємо ГСВРПУ;
- Знаходимо/розраховуємо ГСВРКУ;
- Визначаємо область визначення (числовий інтервал);
- Визначаємо лінгвістичні терми;
- Обираємо числові значення для функції приналежності;
- Перевіряємо генеральну середню варіаційного ряду поточної успішності на відповідність визначеним раніше нечітким термам;
- Перевіряємо генеральну середню варіаційного ряду контрольної успішності на відповідність визначеним раніше нечітким термам;

- Виводимо результат у вигляді висновку та/або у вигляді діаграми.

3.2. Опис основних директорій та файлів

Загальна фізична структура системи СКУН зображена на рисунку 3.1.

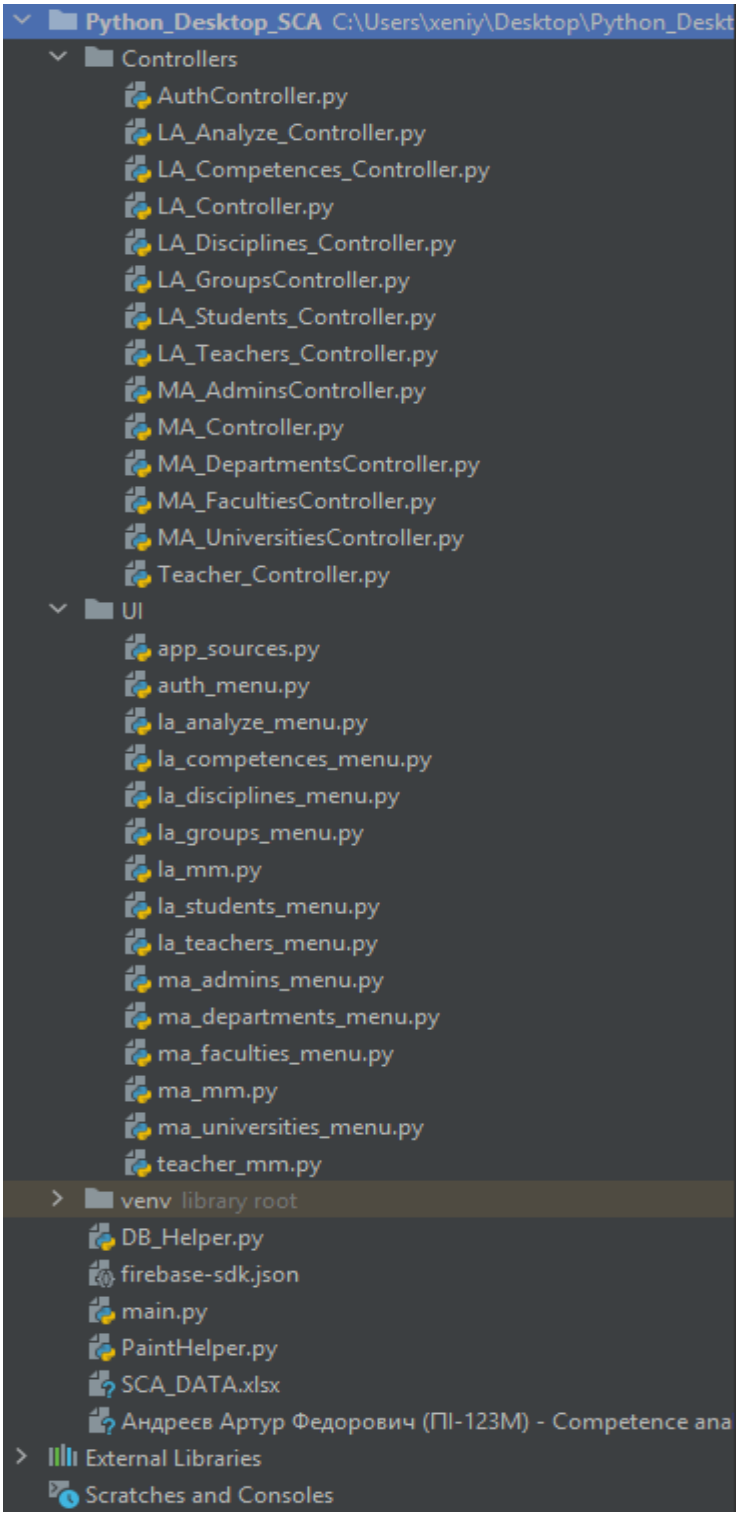


Рис. 3.1. Фізична архітектура СКУН

Як можна побачити на ньому, майже всі файли у застосунку розміщені у трьох основних директоріях:

- Controllers – директорія, в якій знаходяться всі файли - контролери, що відповідають за бізнес-логіку застосунку;
- UI – директорія, в якій знаходять вже сконвертовані файли користуваціого інтерфейсу (іншими словами, представлення (Views));
- Venv – системна директорія, яка слугує для віртуального середовища самого проекту.

Опис файлів у директорії Controllers представлено у таблиці 3.1.

Опис файлів у директорії UI представлено у таблиці 3.2.

Файли у директорії venv описані не будуть, оскільки вони сформовані автоматично при створенні проекту та при завантаженні зовнішніх бібліотек.

Таблиця 3.1.

Опис файлів у директорії Controllers

Назва файлу	**Опис файлу**
1	**2**
AuthController.py	Файл-контролер, який містить бізнес-логіку системи авторизації
LA_Analyze_Controller.py	Файл-контролер, який містить бізнес-логіку системи аналізу компетентностей студента
LA_Competences_Controller.py	Файл-контролер, який містить бізнес-логіку меню компетентностей
LA_Controller.py	Файл-контролер, який містить бізнес-логіку меню локального адміністратора

Закінчення таблиці 3.1.

1	2
LA_Disciplines_Controller.py	Файл-контролер, який містить бізнес-логіку меню дисциплін
LA_Groups_Controller.py	Файл-контролер, який містить бізнес-логіку меню груп
LA_Students_Controller.py	Файл-контролер, який містить бізнес-логіку меню студентів
LA_Teachers_Controller.py	Файл-контролер, який містить бізнес-логіку меню викладачів
MA_AdminsController.py	Файл-контролер, який містить бізнес-логіку меню локальних адміністраторів
MA_Controller.py	Файл-контролер, який містить бізнес-логіку меню головного адміністратора
MA_DepartmentsController.py	Файл-контролер, який містить бізнес-логіку меню кафедр
MA_FacultiesController.py	Файл-контролер, який містить бізнес-логіку меню факультетів
MA_UniversitiesController.py	Файл-контролер, який містить бізнес-логіку меню університетів
Teacher_Controller.py	Файл-контролер, який містить бізнес-логіку меню викладача

Таблиця 3.2.

Опис файлів у директорії UI

Назва файлу	Опис файлу
1	2
App_sources.py	Файл із додатковими ресурсами застосунку (іконки)

Продовження таблиці 3.2.

1	2
Auth_menu.py	Файл користувацького інтерфейсу (представлення) меню авторизації
La_analyze_menu.py	Файл користувацького інтерфейсу (представлення) меню аналізу компетентностей студента
La_competences_menu.py	Файл користувацького інтерфейсу (представлення) меню компетентностей/компетенцій
La_disciplines.py	Файл користувацького інтерфейсу (представлення) меню дисциплін
La_groups.py	Файл користувацького інтерфейсу (представлення) меню груп
La_mm.py	Файл користувацького інтерфейсу (представлення) меню локального адміністратора
La_students_menu.py	Файл користувацького інтерфейсу (представлення) меню студентів
La_teachers_menu.py	Файл користувацького інтерфейсу (представлення) меню викладачів
Ma_admins_menu.py	Файл користувацького інтерфейсу (представлення) меню локальних адміністраторів
Ma_departments.py	Файл користувацького інтерфейсу (представлення) меню кафедр
Ma_faculties.py	Файл користувацького інтерфейсу (представлення) меню факультетів

Закінчення таблиці 3.2.

1	2
Ma_mm.py	Файл користувацького інтерфейсу (представлення) меню головного адміністратора
Ma_universities.py	Файл користувацького інтерфейсу (представлення) меню університетів
Teacher_mm.py	Файл користувацького інтерфейсу (представлення) меню викладача

Опис файлів, що знаходяться поза межами вказаних вище директорій представлено у таблиці 3.3.

Таблиця 3.3.

Опис файлів поза директорій

Назва файлу	Опис файлу
DB_Helper.py	Файл, який слугує для встановлення з’єднання із віддаленою БД
Firebase-sdk.json	Файл, який слугує для забезпечення можливості встановлення з’єднання із віддаленою БД
Main.py	Ключовий (виконавчий) файл застосунку
PaintHelper.py	Файл, який слугує для здійснення оформлення таблиць при експорті даних

3.3. Опис програмних компонентів

Опис методів у файлі AuthController.py представлено у таблиці 3.4.

Опис методів у файлі LA_Analyze_Controller.py представлено у таблиці 3.5.

Опис методів у файлі LA_Competences_Controller.py представлено у таблиці 3.6.

Опис методів у файлі LA_Competences.py представлено у таблиці 3.7.

Опис методів у файлі LA_Disciplines_Controller.py представлено у таблиці 3.8.

Опис методів у файлі LA_Groups_Controller.py представлено у таблиці 3.9.

Опис методів у файлі LA_Students_Controller.py представлено у таблиці 3.10.

Опис методів у файлі LA_Teachers_Controller.py представлено у таблиці 3.11.

Опис методів у файлі MA_AdminsController.py представлено у таблиці 3.12.

Опис методів у файлі MA_Controller.py представлено у таблиці 3.13.

Опис методів у файлі MA_DepartmentsController.py представлено у таблиці 3.14.

Опис методів у файлі MA_FacultiesController.py представлено у таблиці 3.15.

Опис методів у файлі MA_UniversitiesController.py представлено у таблиці 3.16.

Опис методів у файлі Teacher_Controller.py представлено у таблиці 3.17.

Таблиця 3.4.

Опис методів у файлі AuthController.py

Метод	**Опис**	**Вхідні дані**	**Вихідні дані**
checkInBase	Перевірка наявності користувача у БД	str	Bool

auth	Авторизація	-	-
clearData	Очистка полів логіну та паролю	-	-

Таблиця 3.5.

Опис методів у файлі LA_Analyze_Controller.py

Метод	**Опис**	**Вхідні дані**	**Вихідні дані**
1	**2**	**3**	**4**
GetDisciplinesIdsArr	Отримання списку номерів дисциплін	str	list
GetEntityById	Отримання конкретної операційної сутності із конкретної таблиці	str, str	object
GetAllMarksByStudentAndDiscipline	Отримання всіх оцінок конкретного студента по конкретній дисципліні	str, object	list
SearchMarksByType	Пошук оцінок конкретного типу	list, list	list
GetCheckedPerformance	Отримання конкретної (поточна/контрольна) успішності	list, str, list	list
GetCountOfOccurrences	Отримання кількості входження оцінки у список	list, int	int

Продовження таблиці 3.5.

1	2	3	4
GetCheckedVariativeRow	Отримання варіаційного ряду конкретної (поточна / контрольна) успішності	list	Dict
GetCheckedGeneralAverage	Отримання генеральної середньої змінної	dict	float
MakeCalculation	Аналіз входження числового значення до нечіткої групи	float, float, float, float, float	float
MakeFuzzyAnalyze	Здійснення аналізу конкретної генеральної середньої змінної	float	dict
MakeCompetenceAnalyze	Запуск аналізу компетентностей	str, str	list
FormatData	Форматування дати	str	str
getActualDateAndTime	Отримання поточної дати та часу	-	str
fillGroupsSpinner	Заповнення випадаючого списку груп	-	-

Закінчення таблиці 3.5.

1	2	3	4
fillStudentsSpinner	Заповнення випадаючого списку студентів	-	-
fillCompetencesList	Заповнення списку компетентностей	-	-
clearData	Очистка вибору у випадаючих списках	-	-
StartCompetenceAnalyze	Ініціалізація аналізу компетентностей	-	-
Export	Експорт результатів аналізу компетентностей	-	-
getRecords	Отримання даних із таблиці результатів	-	list
getCheckedCompetencesNames	Отримання назв обраних компетентностей	-	str

Таблиця 3.6.

Опис методів у файлі LA_Competences_Controller.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
getEntityByName	Отримання сутності у конкретній таблиці по конкретному полю	str, str, str	object

Продовження таблиці 3.6.

1	2	3	4
setTime	Виставлення поточної дати/часу	-	-
fillDepartmentsSpinner	Заповнення випадаючого списку кафедр	-	-
fillFacultiesSpinner	Заповнення випадаючого списку факультетів	-	-
cellChecked	Отримання даних обраної компетенції через відслідковування натискання у таблиці	int, int	-
transformData	Отримання списку викладачів із строки	str	list
clearSelection	Очистка виділення у списку дисциплін	-	-
fillDisciplinesList	Заповнення списку дисциплін	-	-
loadCompetencesData	Завантаження даних компетентностей	-	-
createNewCompetence	Створення нової компетентності	-	-

Продовження таблиці 3.6.

1	2	3	4
updateCompetence	Оновл.обраної компетенції	-	-
deleteCompetence	Видалення обр. компетенції	-	-
export	Ініціалізація експорту даних компетентностей	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних компетентностей	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_ma_mm	Повернення до головного меню лок. адм.	-	-

Таблиця 3.7.

Опис методів у файлі LA_Competences.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
setData	Встановлення статистики опер. одиниць	-	-

Закінчення таблиці 3.7.

1	2	3	4
closeEvent	Відслідк. закриття форми	QCloseEvent	-
setTime	Виставл. поточної дати/часу	-	-
back_to_auth	Повернення до меню автор.	-	-
go_to_groups_menu	Перехід до меню груп	-	-
go_to_students_menu	Перехід до меню студентів	-	-
go_to_teachers_menu	Перехід до меню викладачів	-	-
go_to_disciplines_menu	Перехід до меню дисциплін	-	-
go_to_competences_menu	Перехід до меню компетенцій	-	-
go_to_analyze_menu	Перехід до меню аналізу	-	-

Таблиця 3.8.

Опис методів у файлі LA_Disciplines_Controller.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
getEntityByName	Отримання сутності у таблиці	str, str, str	object

Продовження таблиці 3.8.

1	2	3	4
setTime	Виставлення поточної дати/часу	-	-
fillDepartmentsSpinner	Заповнення вип. списку кафедр	-	-
fillFacultiesSpinner	Заповнення вип. списку факультетів	-	-
cellChecked	Отримання даних обраної дисципліни через відслідковування натискання у таблиці	int, int	-
transformData	Отримання списку викладачів із строки	str	list
clearSelection	Очистка виділення у списку дисциплін	-	-
fillTeachersList	Заповнення списку викладачів	-	-
loadDisciplinesData	Завантаження даних дисциплін	-	-
createNewDiscipline	Створення нової дисципліни	-	-
updateDiscipline	Оновлення обраної дисципліни	-	-

Закінчення таблиці 3.8.

1	2	3	4
deleteDiscipline	Видалення обраної дисципліни	-	-
export	Ініціалізація експорту даних дисциплін	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних дисциплін	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_la_mm	Повернення до головного меню локального адміністратора	-	-

Таблиця 3.9.

Опис методів у файлі LA_Groups_Controller.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
getEntityByName	Отримання сутності у таблиці по полю	str, str, str	object

Продовження таблиці 3.9.

1	2	3	4
setTime	Виставлення поточної дати/часу	-	-
fillDepartmentsSpinner	Заповнення випадаючого списку кафедр	-	-
fillFacultiesSpinner	Заповнення випадаючого списку факультетів	-	-
cellChecked	Отримання даних обраної групи через відслідковування натискання у таблиці	int, int	-
loadGroupsData	Завантаження даних груп	-	-
createNewGroup	Створення нової групи	-	-
updateGroup	Оновлення обраної групи	-	-
deleteGroup	Видалення обраної групи	-	-
export	Ініціалізація експорту даних груп	-	-

Закінчення таблиці 3.9.

1	2	3	4
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних груп	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_la_mm	Повернення до головного меню локального адміністратора	-	-

Таблиця 3.10.

Опис методів у файлі LA_Students_Controller.py

Метод	**Опис**	**Вхідні дані**	**Вихідні дані**
1	**2**	**3**	**4**
getEntityByName	Отримання сутності у таблиці по полю	str, str, str	object
setTime	Виставлення поточної дати/часу	-	-
fillGroupsSpinner	Заповнення випадаючого списку груп	-	-

Продовження таблиці 3.10.

1	2	3	4
cellChecked	Отримання даних обраної групи через відслідковування натискання у таблиці	int, int	-
loadStudentsData	Завантаження даних студентів	-	-
createNewStudent	Створення нового студента	-	-
updateStudent	Оновлення обраного студента	-	-
deleteStudent	Видалення обраного студента	-	-
export	Ініціалізація експорту даних студентів	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних студентів	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-

Закінчення таблиці 3.10.

1	2	3	4
back_to_la_mm	Повернення до головного меню локального адміністратора	-	-

Таблиця 3.11.

Опис методів у файлі LA_Teachers_Controller.py

Метод	Опис	Вхідні дані	Вихідні дані
1	**2**	**3**	**4**
getEntityByName	Отримання конкретної сутності у конкретній таблиці по конкретному полю	str, str, str	object
setTime	Виставлення поточної дати/часу	-	-
cellChecked	Отримання даних обраного викладача через відслідковування натискання у таблиці	int, int	-
fillDepartmentsSpinner	Заповнення випадаючого списку кафедр	-	-

Закінчення таблиці 3.11.

1	2	3	4
fillFacultiesSpinner	Заповнення вип. списку факультетів	-	-
loadTeachersData	Завантаження даних викладачів	-	-
createNewTeacher	Ств. нового викладача	-	-
updateTeacher	Оновл. обраного викладача	-	-
deleteTeacher	Видалення обраного викладача	-	-
export	Ініціалізація експорту даних викладачів	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних викладачів	Worksheet, Workbook, str	-
closeEvent	Відслідк. закриття форми	QCloseEvent	-
back_to_la_mm	Повернення до головного меню локального адміністратора	-	-

Таблиця 3.12.

Опис методів у файлі MA_AdminsController.py

Метод	**Опис**	**Вхідні дані**	**Вихідні дані**
1	**2**	**3**	**4**
setTime	Виставлення поточної дати/часу	-	-
fillUniversitiesSpinner	Заповнення випадаючого списку університетів	-	-
cellChecked	Отримання даних обраного викладача через відслідковування натискання у таблиці	int, int	-
loadAdminsData	Завантаження даних локальних адміністраторів	-	-
createNewLocalAdmin	Створення нового локального адміністратора	-	-
updateLocalAdmin	Оновлення обраного локального адміністратора	-	-

Закінчення таблиці 3.12.

1	2	3	4
deleteLocalAdmin	Видалення обр. локального адміністратора	-	-
export	Ініціалізація експорту даних лок.адміністраторів	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних лок.адміністраторів	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_ma_mm	Повернення до гол. меню головного адміністратора	-	-

Таблиця 3.13.

Опис методів у файлі MA_Controller.py

Метод	Опис	Вхідні дані	Вихідні дані
1	**2**	**3**	**4**
setTime	Виставлення поточної дати/часу	-	-

Закінчення таблиці 3.13.

1	2	3	4
back_to_auth	Повернення до меню авторизації	-	-
go_to_universities_menu	Перехід до меню університетів	-	-
go_to_faculties_menu	Перехід до меню факультетів	-	-
go_to_departments_menu	Перехід до меню кафедр	-	-
go_to_local_admins_menu	Перехід до меню факультетів	-	-
setData	Встановлення статистики підконтрольних операційних одиниць	-	-

Таблиця 3.14.

Опис методів у файлі MA_DepartmentsController.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
setTime	Виставлення поточної дати/часу	-	-
fillFacultiesSpinner	Заповнення випадаючого списку факультетів	-	-

Продовження таблиці 3.14.

1	2	3	4
cellChecked	Отримання даних обраної кафедри через відслідковування натискання у таблиці	int, int	-
loadDepartmentsData	Завантаження даних кафедр	-	-
createNewDepartment	Створення нової кафедри	-	-
updateDepartment	Оновлення обраної кафедри	-	-
deleteDepartment	Видалення обраної кафедри	-	-
export	Ініціалізація експорту даних кафедр	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних кафедр	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-

Закінчення таблиці 3.14.

1	2	3	4
back_to_ma_mm	Повернення до головного меню головного адміністратора	-	-

Таблиця 3.15.

Опис методів у файлі MA_FacultiesController.py

Метод	Опис	Вхідні дані	Вихідні дані
1	**2**	**3**	**4**
setTime	Виставлення поточної дати/часу	-	-
fillUniversitiesSpinner	Заповнення випадаючого списку університетів	-	-
cellChecked	Отримання даних обраного факультету через відслідковування натискання у таблиці	int, int	-
loadFacultiesData	Завантаження даних факультетів	-	-
createNewFaculty	Створення нового факультетів	-	-

Закінчення таблиці 3.15.

1	2	3	4
updateFaculty	Оновлення обр. факультету	-	-
deleteFaculty	Видалення обр. факультету	-	-
export	Ініціалізація експорту даних факультетів	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних факультетів	Worksheet, Workbook, str	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_ma_mm	Повернення до гол. меню головного адміністратора	-	-

Таблиця 3.16.

Опис методів у файлі MA_UniversitiesController.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
setTime	Виставлення поточної дати/часу	-	-

Продовження таблиці 3.16.

1	2	3	4
cellChecked	Отримання даних обраного університету через відслідковування натискання у таблиці	int, int	-
loadUniversitiesData	Завантаження даних університетів	-	-
createNewUniv	Створення нової кафедри	-	-
updateUniv	Оновлення обраної кафедри	-	-
deleteUniv	Видалення обраної кафедри	-	-
export	Ініціалізація експорту даних унівеситетів	-	-
fillHeaders	Заповнення та оформлення заголовків таблиці для експорту	Worksheet, Font, PatternFill, Alignment, Workbook	-
startExport	Експорт даних університетів	Worksheet, Workbook, str	-

Закінчення таблиці 3.16.

1	2	3	4
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_ma_mm	Повернення до головного меню головного адміністратора	-	-

Таблиця 3.17.

Опис методів у файлі Teacher_Controller.py

Метод	Опис	Вхідні дані	Вихідні дані
1	2	3	4
getEntityByName	Отримання конкретної сутності у конкретній таблиці по конкретному полю	str, str, str	object
getStudentByFIOAndGroup	Отримання студента із конкретним ФІО та конкретної групи	str, str, str, str	object
setTime	Виставлення поточної дати/часу	-	-

Продовження таблиці 3.17.

1	2	3	4
setShortFIO	Форматування ФІО студента	-	-
fillGroupsSpinner	Заповнення випадаючого списку груп	-	-
fillStudentsSpinner	Заповнення випадаючого списку студентів	-	-
fillDisciplinesSpinner	Заповнення випадаючого списку дисциплін	-	-
fillTypesSpinner	Заповнення випадаючого списку типів оцінок	-	-
cellChecked	Отримання даних обраної оцінки через відслідковування натискання у таблиці	int, int	-
loadMarksData	Завантаження даних оцінок	-	-
createNewMark	Створення нової оцінки	-	-
updateMark	Оновлення обраної оцінки	-	-

Закінчення таблиці 3.17.

1	2	3	4
deleteMark	Видалення обраної оцінки	-	-
closeEvent	Відслідковування закриття форми	QCloseEvent	-
back_to_auth	Повернення до меню авторизації	-	-

Код контролеру, що відповідає за роботу з локальними адміністраторами, представлено у лістінгу 1 (додаток Б).

Код контролеру, що відповідає за роботу з викладачами, представлено у лістінгу 2 (додаток Б).

Код контролеру, що відповідає за роботу з успішністю, представлено у лістінгу 3 (додаток Б).

Код контролеру, що відповідає за роботу з компетентностями, представлено у лістінгу 4 (додаток Б).

3.4. Структура бази даних

Кінцева структура віддаленої БД, що використовується системою СКУН для зберігання даних успішності та даних операційних одиниць, представлена на рисунку 3.2. Як можна побачити на даному рисунку, структура віддаленої БД відповідає сформованим та описним раніше (див. пункт 1.1.) структурі операційних одиниць. Важливо розуміти, що на рисунку 3.2. зображені не таблиці, а вузли дерева (оскільки база даних, що використовується, має тип NoSQL, яка працює на основі логічних дереа, а не таблиць). Також дана структура відповідає онтологічним моделям, що були сформовані раніше (див. пункт 1.4.).

Рис. 3.2. Структура віддаленої БД

Як зазначалося раніше, No-SQL бази даних мають іншу структуру, ніж бази даних, що базуються на SQL (MySQL, PostgreSQL, тощо). Записи у таких БД базуються на словниках. Відповідно, приклад запису у такій БД представлено на рисунку 3.3.

Рис. 3.3. Запис про компетентність у віддаленій БД

На рисунку вище можна побачити структуру операційної одиниці типу «Компетентність».

На рисунку 3.4. представлено структуру операційної одиниці типу «Кафедра».

На рисунку 3.5. представлено структуру операційної одиниці типу «Дисципліна».

На рисунку 3.6. представлено структуру операційної одиниці типу «Факультет».

На рисунку 3.7. представлено структуру операційної одиниці типу «Група».

На рисунку 3.8. представлено структуру операційної одиниці типу «Оцінка».

На рисунку 3.9. представлено структуру операційної одиниці типу «Студент».

На рисунку 3.10. представлено структуру операційної одиниці типу «Університет».

На рисунку 3.11. представлено структуру операційної одиниці типу «Користувач» (а конкретно структуру операційної одиниці «Головний адміністратор»).

На рисунку 3.12. представлено структуру операційної одиниці типу «Користувач» (а конкретно структуру операційної одиниці «Локальний адміністратор»).

На рисунку 3.13. представлено структуру операційної одиниці типу «Користувач» (а конкретно структуру операційної одиниці «Викладач»).

Усі поля для різних записів було реалізовано на основі сформованого раніше вербального опису системи (див. пункт 1.2.) та вербальному описі операційних одиниць (див. пункт 1.1.).

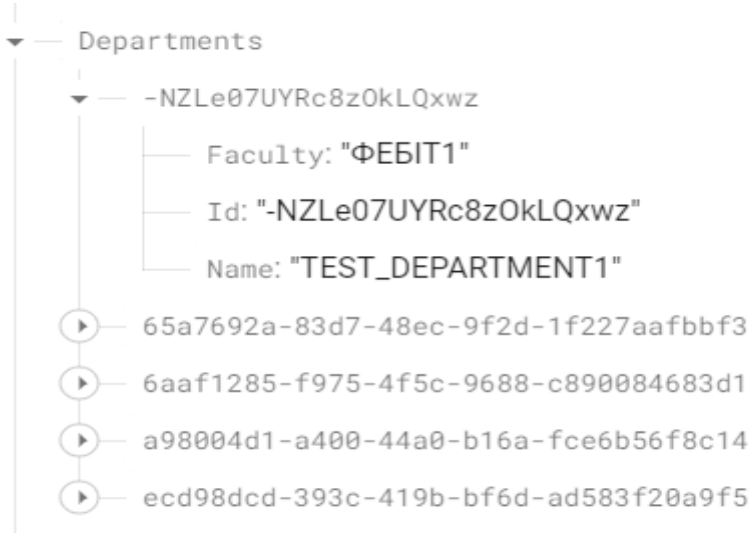


Рис. 3.4. Запис про кафедру у віддаленій БД

Рис. 3.5. Запис про дисципліну у віддаленій БД

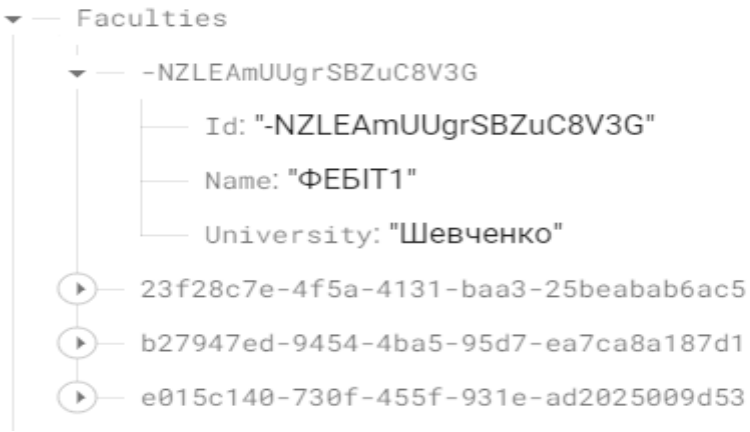


Рис. 3.6. Запис про факультет у віддаленій БД

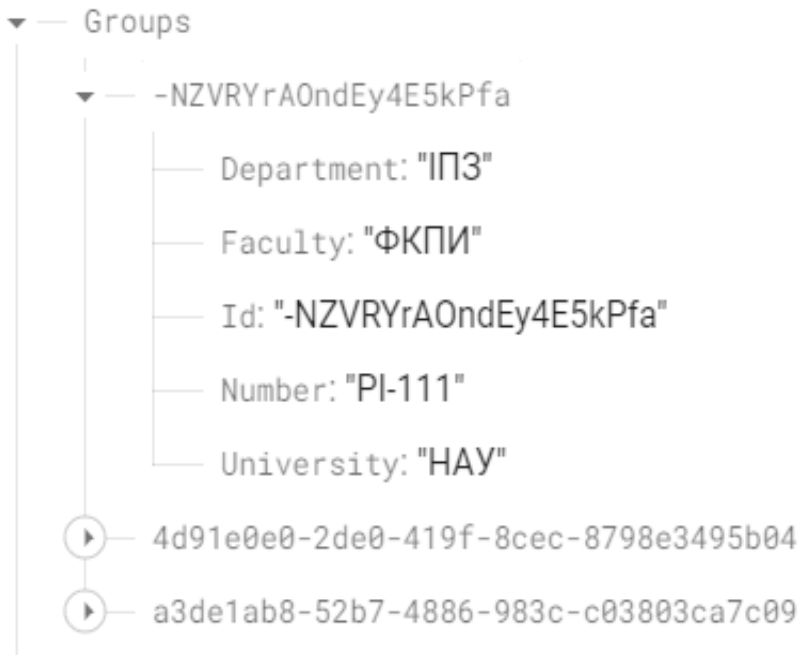


Рис. 3.7. Запис про групу у віддаленій БД

Рис. 3.8. Запис про оцінку (успішність) у віддаленій БД

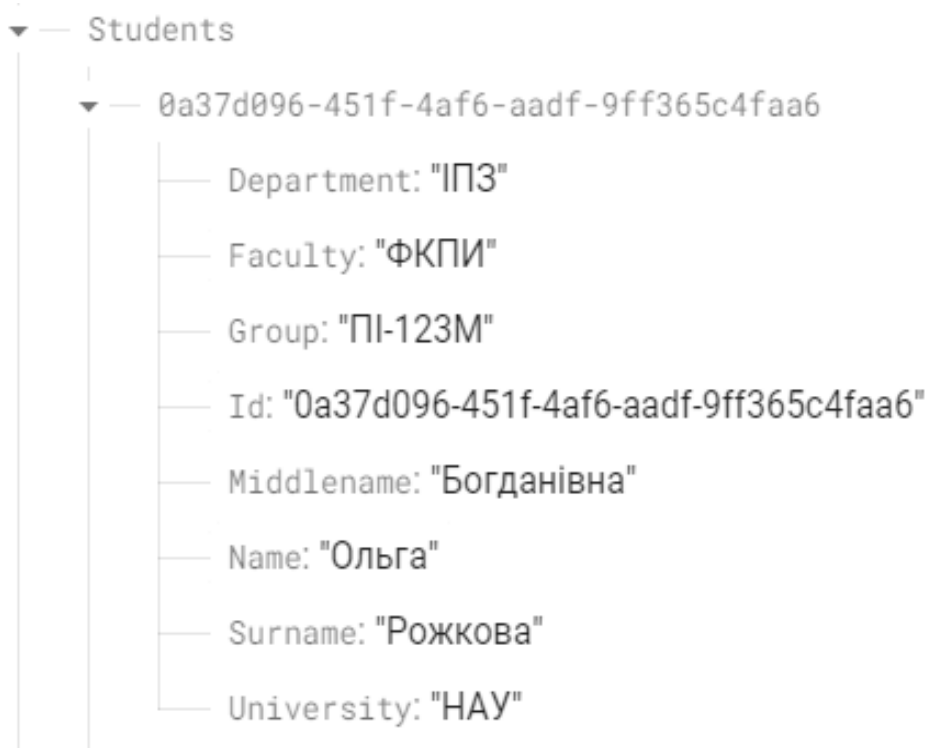


Рис. 3.9. Запис про студента у віддаленій БД

Рис. 3.10. Запис про університет у віддаленій БД

Рис. 3.11. Запис про головного адміністратора у віддаленій БД

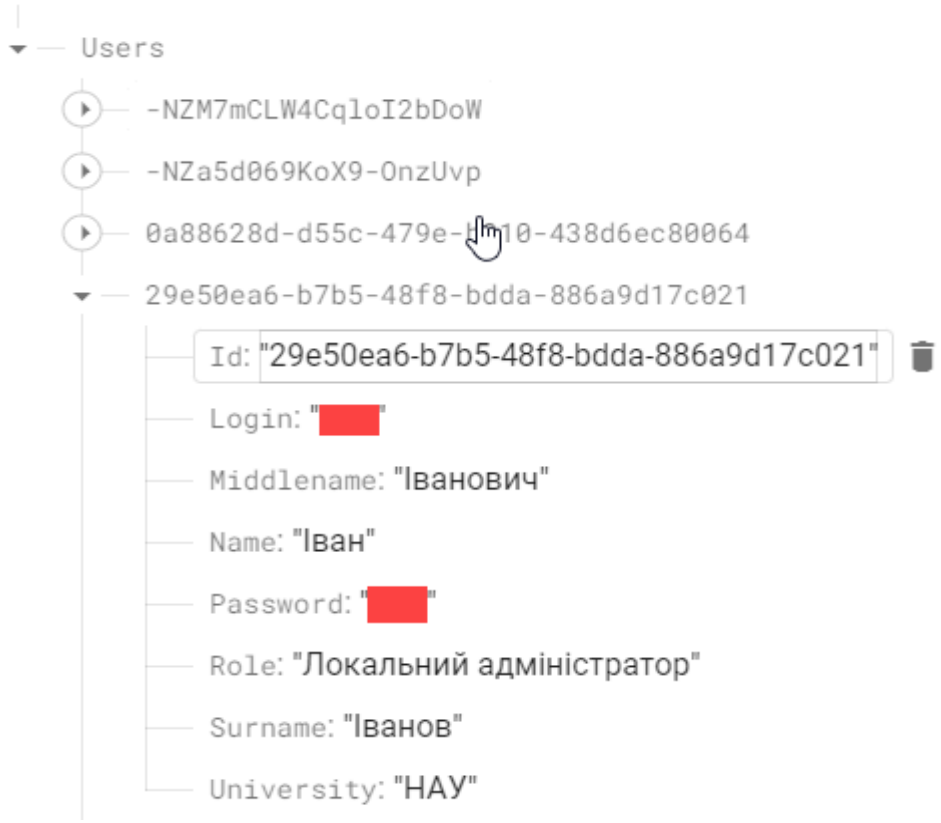


Рис. 3.12. Запис про локального адміністратора у віддаленій БД

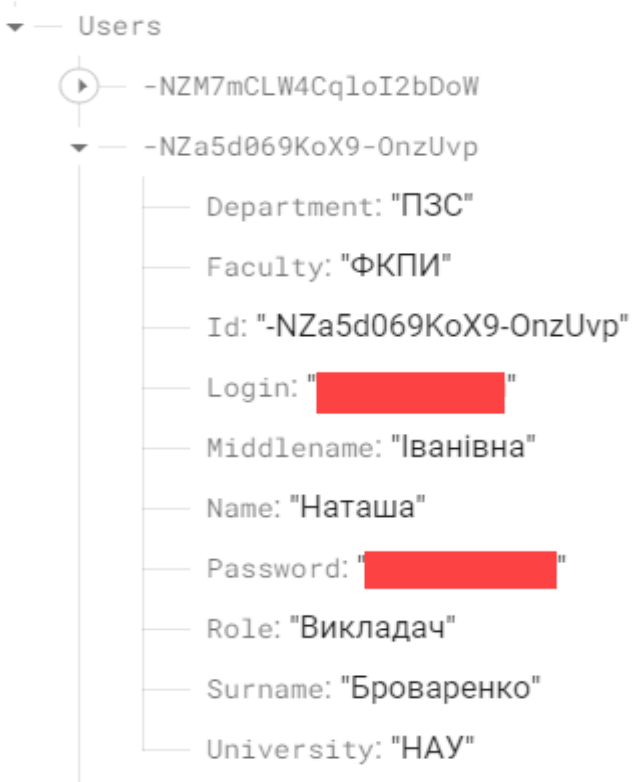


Рис. 3.13. Запис про викладача у віддаленій БД

Висновки

У даному розділі наводиться опис фізичної структури системи СКУН, що була розроблена на основі аналізу логічної архітектури, що була представлена раніше (див. пункт 2.1. – 2.4.).

В першу чергу, було наведено та аргументовано перелік архітектурних рішень, що реалізуються під час розробки проекту. Кожне архітектурне рішення було коротко описане.

Далі було проведено аналіз основних директорій застосунку. Список директорій було виділено на основі обраного раніше архітектурного патерну. Одночасно з цим було наведено опис файлів у цих директоріях.

Надалі було надано опис всіх методів бізнес-логіки застосунку (методи у контролерах). Для кожного методу було наведено короткий опис а також перелік вхідних та вихідних даних.

Наостанок було наведено актуальну на момент закінчення проекту структуру віддаленої БД, із описом реалізованих операційних одиниць. Для кожної операційної одиниці було представлено у вигляді відповідних рисунків приклад реалізації у віддаленій NoSQL БД Firebase. На рисунках видно , що кожна операційна одиниця має перелік полів, опис яких представлено раніше (див. пункт 1.1.).

Таким чином, відповідно до вербального опису системи (див. пункт 1.1 – 1.2.), сформмованим онтологічним моделям (див. пункт 1.4.), визначеним раніше функціональним (див. пункт 2.2.) та нефункціональним вимогам (див. пункт 2.3.), а також спроектованій раніше логічній архітектурі (див. пункт 2.1.), на основі продемонстрованих у даному розділі результатів, можна зробити висновок, що розроблена фізична структура застосунку СКУН відповідає раніше сформованим описам, та логічній архітектурі.

Доказ відповідності системи СКУН виділеним раніше функціональним/нефункціональним вимогам буде представлено у наступному розділі.

РОЗДІЛ 4.
ДЕМОНСТРАЦІЯ РОБОТИ ЗАСТОСУНКУ

6.1. Демонстрація авторизації

Меню авторизації застосунку представлено на рисунку 4.1.

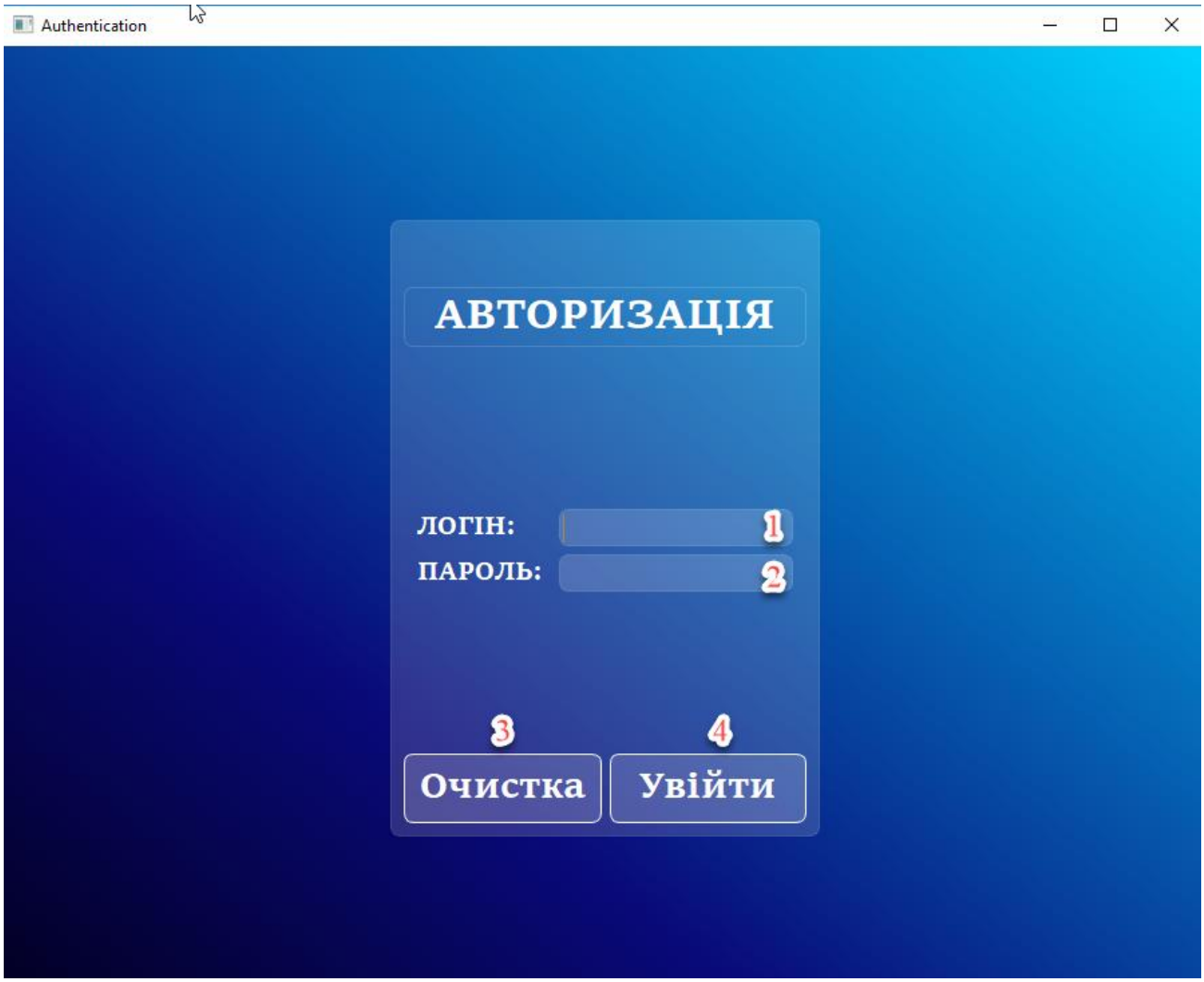


Рис. 4.1. Меню авторизації

Меню авторизації у системі налічує наступні інтерактивні елементи, із якими користувач може взаємодіяти:

- поле для вводу логіну (1);
- поле для вводу паролю (2);
- кнопка очистки полів (3);

- кнопка входу (4);

Користувач вводить свій логін та пароль у відповідні поля (1 та 2) та натискає на кнопку входу (3), після чого відбувається двох-фазова перевірка: спершу перевіряється чи є користувач із таким логіном у системі, а потім – чи співпадають паролі. Якщо обидва етапи перевірки пройшли успішно – система визначає роль користувача та відбувається перехід до відповідного меню (головного адміністратора, локального адміністратора або викладача). При чому поточне меню приховується. Якщо хоча б одна із фаз перевірки закінчилася невдачею – не відбувається нічого, однак і переходу не буде.

6.2. Демонстрація функціоналу головного адміністратора

Інформаційне меню головного адміністратора представлене на рисунку 4.2.

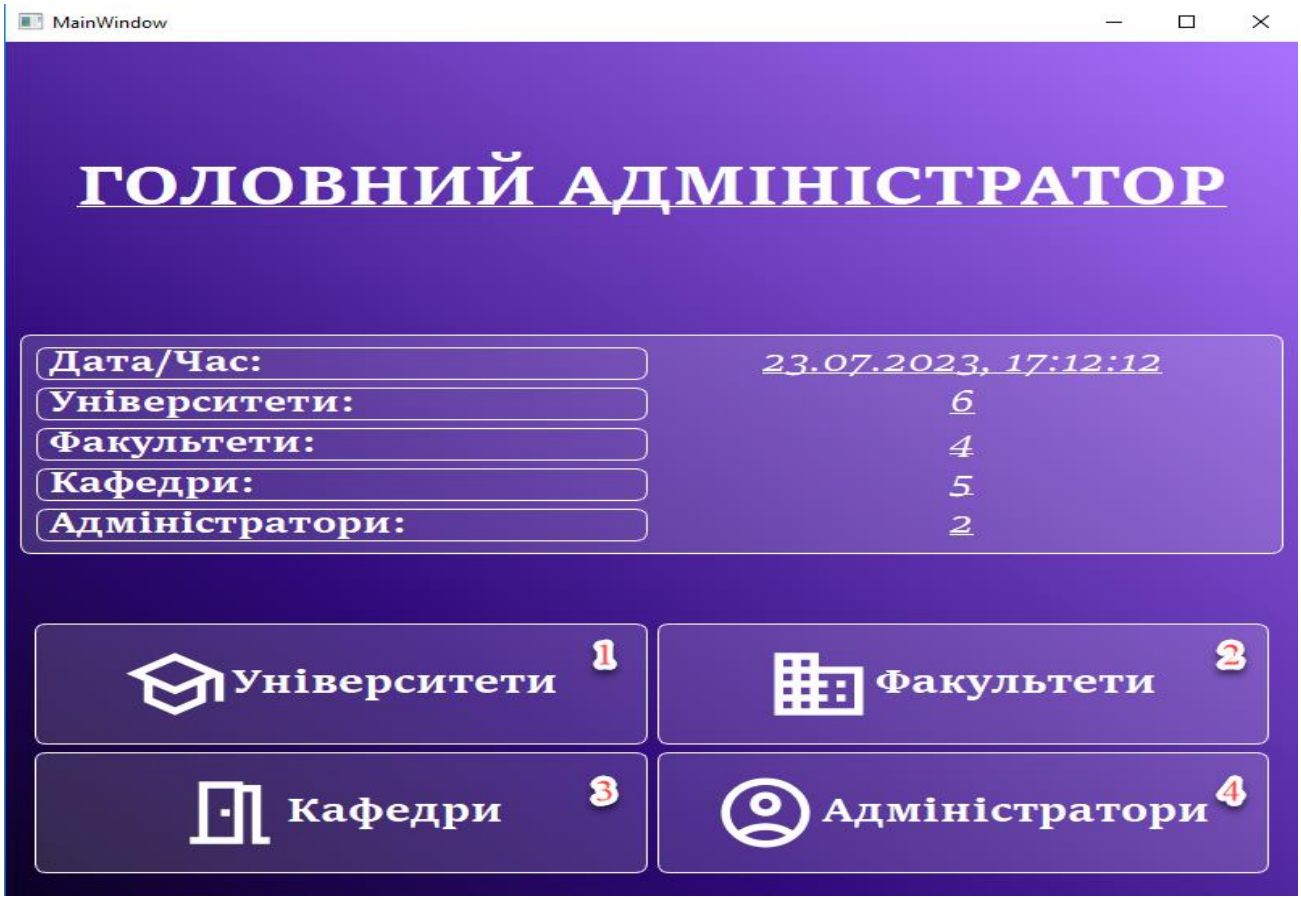


Рис. 4.2. Інформаційне меню головного адміністратора

Інформаційне меню головного адміністратора налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- кнопка меню університетів (1);
- кнопка меню факультетів (2);
- кнопка меню кафедри (3);
- кнопка меню локальних адміністраторів (4).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

При натисканні на кнопку меню університетів (1) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню факультетів (2) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню кафедр (3) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню локальних адміністраторів (4) відбувається перехід до відповідного меню. При чому поточне меню приховується.

Над кнопками (1-4) знаходиться інформаційна панель, на якій відображається актуальна дата та час та актуальна кількість університетів / факультетів / кафедр / локальних адміністраторів у системі. Взаємодіяти (вводити свої значення, копіювати дані) із цією панеллю користувач не може – можливий лише перегляд. Дані до цієї панелі завантажуюиться автоматично. Дата та час оновлюється у режимі реального часу.

Меню університетів представлено на рисунку 4.3.

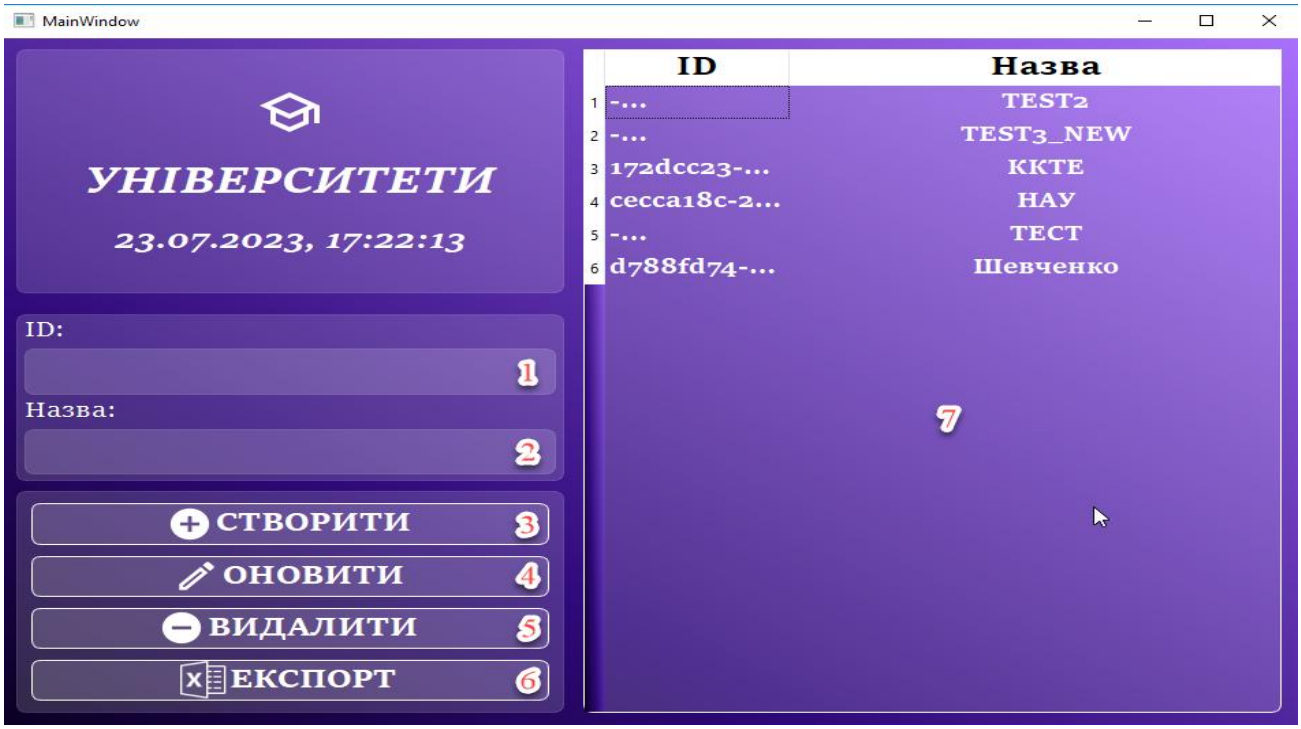


Рис. 4.2. Меню університетів

Меню університетів налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу назви (2);
- кнопка створення університету (3);
- кнопка оновлення університету (4);
- кнопка видалення університету (5);
- кнопка експорту даних університетів (6);
- таблиця для університетів (7).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

При натисканні у таблиці (6) на будь-яку клітину із першого стовпця (стовпець «ID») у поле (1) та поле (2) відбувається автоматичне занесення даних відповідного університету (один університет = одна стрічка у таблиці). При натисканні у таблиці (6) на будь-яку іншу клітину відбувається очистка полів (1) та (2).

Стовпці у таблиці (6) можна розтягувати.

При натисканні на кнопку (3) відбувається створення нового університету та занесення його у відповідну БД. Назва нового університету береться із поля (2). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (4) відбувається оновлення обраного університету. Нова назва університету береться із поля (2). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (5) відбувається видалення обраного університету. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (6) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Universities». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Universities» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.3.

	A	B	C	D	E
1	ID	НАЗВА			
2	-NZBXyB1GSbtHgQ2w09V	ТЕСТ			
3	-NZGDuWzQE3bGmvURMVr	TEST2			
4	-NZKmNNI6EuxZYPvP88s	TEST3_NEW			
5	172dcc23-de5a-4d0e-8a86-51d77f0899af	ККТЕ			
6	cecca18c-2b0b-4766-b847-74f46c2fc4c4	НАУ			
7	d788fd74-4ed0-42f8-bf61-01271a3b6642	Шевченко			
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence: ...

Рис. 4.3. Результат експорту даних університетів

Меню факультетів представлено на рисунку 4.4.

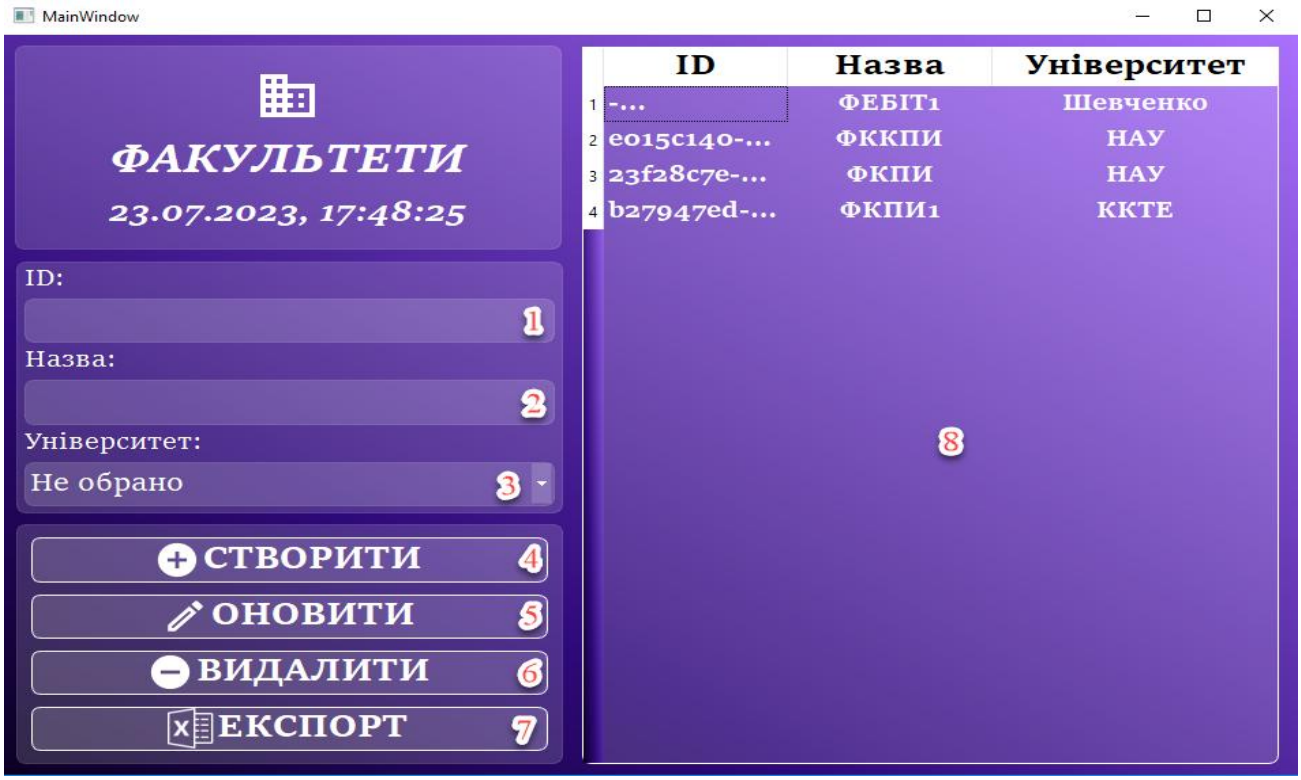


Рис. 4.4. Меню факультетів

Меню факультетів налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу назви (2);
- випадаючий список університетів (3);
- кнопка створення факультету (4);
- кнопка оновлення факультету (5);
- кнопка видалення факультету (6);
- кнопка експорту даних факультетів (7);
- таблиця для факультетів (8).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір

(стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (3) заповнюється при запуску форми автоматично.

При натисканні у таблиці (8) на будь-яку клітину із першого стовпця (стовпець «ID») у поле (1) та поле (2) відбувається автоматичне занесення даних відповідного факультету (один факультет = одна стрічка у таблиці), а у випадаючому списку автоматично обирається університет. При натисканні у таблиці (8) на будь-яку іншу клітину відбувається очистка полів (1) та (2).

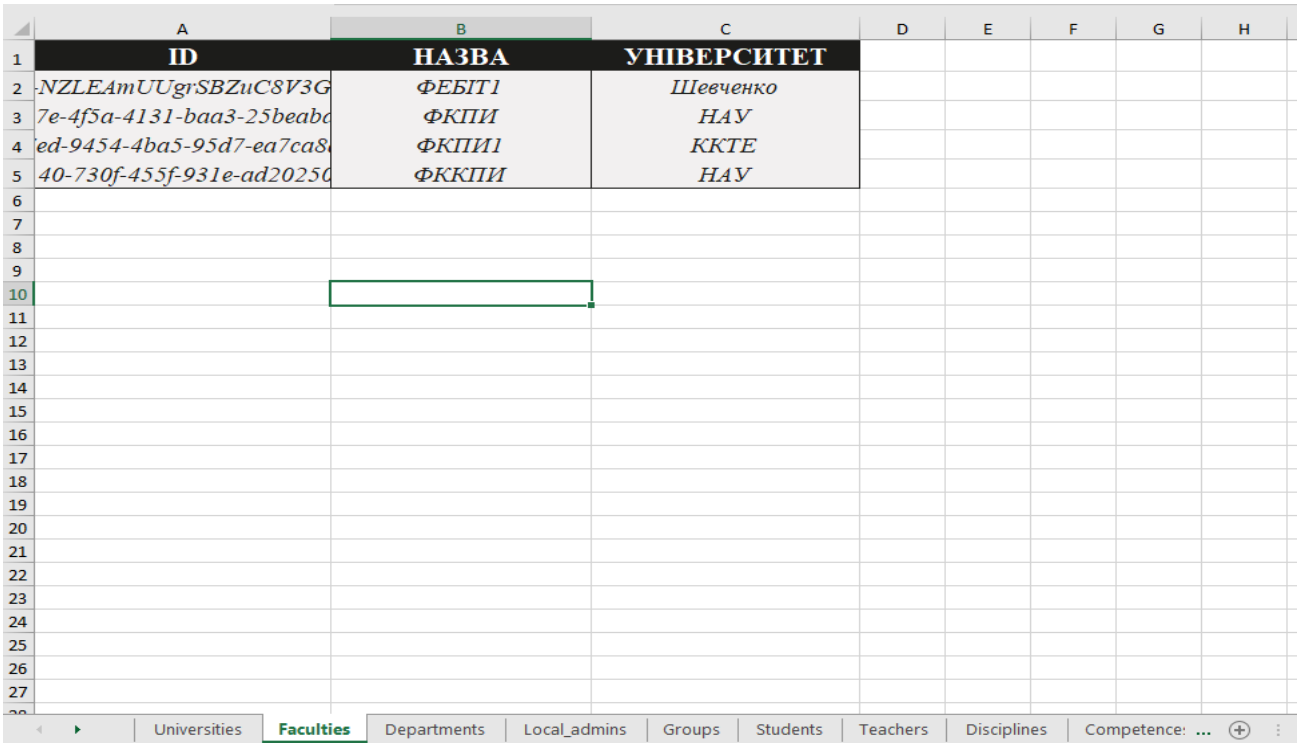
Стовпці у таблиці (8) можна розтягувати.

При натисканні на кнопку (4) відбувається створення нового факультету та занесення його у відповідну БД. Назва нового факультету береться із поля (2), а значення університету – із випадаючого списку (3). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (5) відбувається оновлення обраного факультету. Нова назва факультету береться із поля (2), а новий університет – із випадаючого списку (3). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (6) відбувається видалення обраного факультету. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (7) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Faculties». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Faculties» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.5.



	A	B	C
1	ID	НАЗВА	УНІВЕРСИТЕТ
2	-NZLEAmUUgrSBZuC8V3G	ФЕБІТ1	Шевченко
3	7e-4f5a-4131-baa3-25beabc	ФКПИ	НАУ
4	ed-9454-4ba5-95d7-ea7ca8	ФКПИ1	ККТЕ
5	40-730f-455f-931e-ad20250	ФККПИ	НАУ

Рис. 4.5. Результат експорту даних факультетів

Меню кафедр представлено на рисунку 4.6.

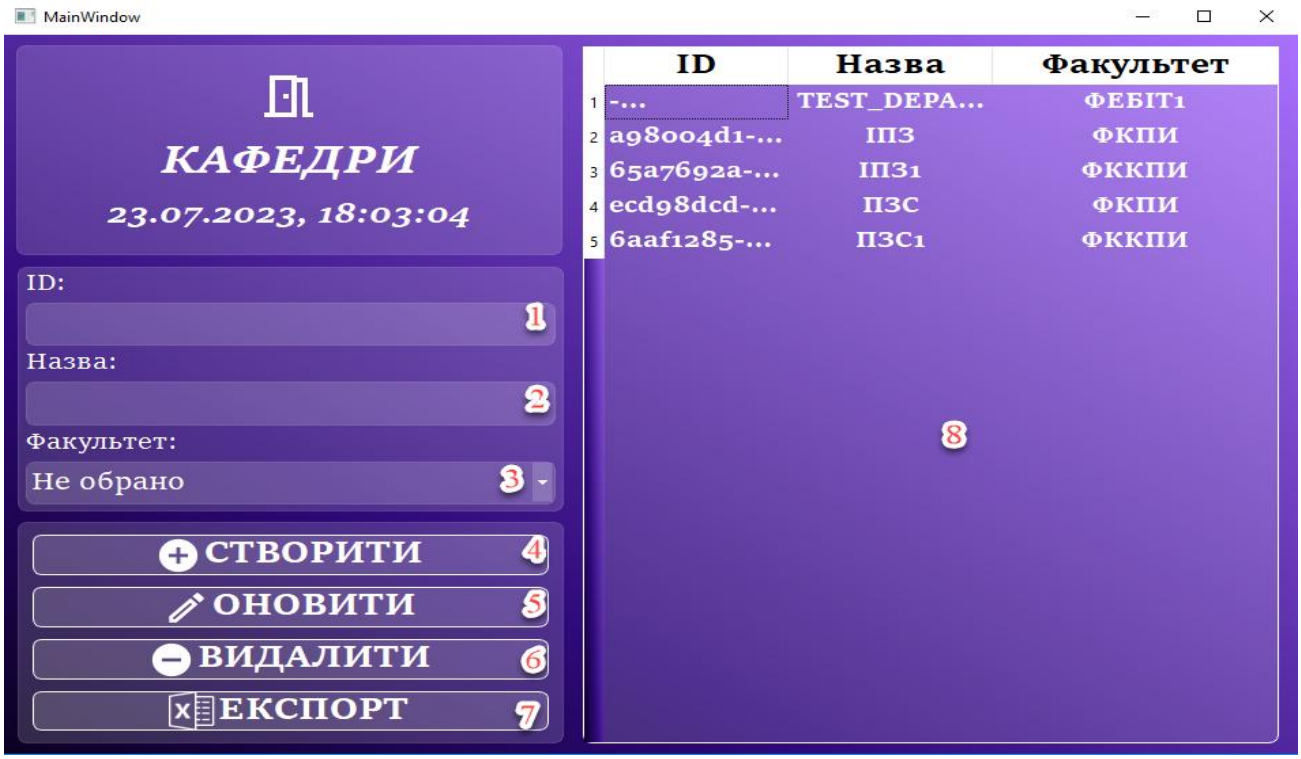


Рис. 4.6. Меню кафедр

Меню кафедр налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу назви (2);
- випадаючий список факультетів (3);
- кнопка створення кафедр (4);
- кнопка оновлення кафедр (5);
- кнопка видалення кафедр (6);
- кнопка експорту даних кафедр (7);
- таблиця для кафедр (8).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (3) заповнюється при запуску форми автоматично.

При натисканні у таблиці (8) на будь-яку клітину із першого стовпця (стовпець «ID») у поле (1) та поле (2) відбувається автоматичне занесення даних відповідної кафедри (одна кафедра = одна стрічка у таблиці), а у випадаючому списку автоматично обирається факультет. При натисканні у таблиці (8) на будь-яку іншу клітину відбувається очистка полів (1) та (2).

Стовпці у таблиці (8) можна розтягувати.

При натисканні на кнопку (4) відбувається створення нової кафедри та занесення його у відповідну БД. Назва нової кафедри береться із поля (2), а значення факультету – із випадаючого списку (3). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (5) відбувається оновлення обраної кафедри. Нова назва кафедри береться із поля (2), а новий факультет – із випадаючого списку (3). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (6) відбувається видалення обраної кафедри.

Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (7) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Departments». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Departments» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.7.

	A	B	C	D	E
1	ID	НАЗВА	ФАКУЛЬТЕТ		
2	-NZLe07UYRc8zOkLQxwz	TEST_DEPARTMENT1	ФЕБІТ1		
3	17692a-83d7-48ec-9f2d-1f227aafb	ІП31	ФККПИ		
4	f1285-f975-4f5c-9688-c890084683	ПЗС1	ФККПИ		
5	004d1-a400-44a0-b16a-fce6b56f8	ІП3	ФКПИ		
6	98dcd-393c-419b-bf6d-ad583f20a	ПЗС	ФКПИ		
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence: ...

Рис. 4.7. Результат експорту даних кафедр

Меню локальних адміністраторів представлено на рисунку 4.8.

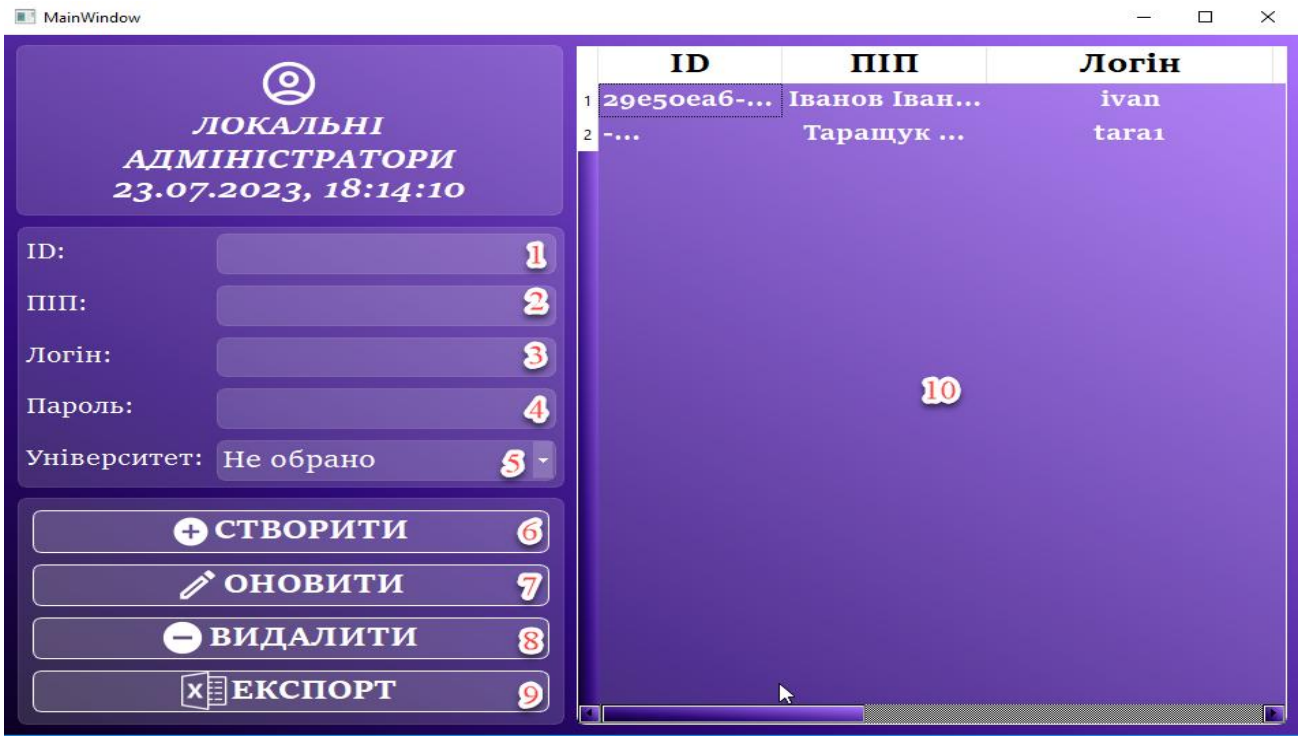


Рис. 4.8. Меню локальних адміністраторів

Меню локальних адміністраторів налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу ПІП адміністратора (2);
- поле для вводу логіну (3);
- поле для вводу паролю (4);
- випадаючий список університетів (5);
- кнопка створення адміністраторів (6);
- кнопка оновлення адміністраторів (7);
- кнопка видалення адміністраторів (8);
- кнопка експорту даних адміністраторів (9);
- таблиця для локальних адміністраторів (10).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише

перегляд та копіювання значення.

Випадаючий список (5) заповнюється при запуску форми автоматично.

При натисканні у таблиці (10) на будь-яку клітину із першого стовпця (стовпець «ID») у поля (1-4) відбувається автоматичне занесення даних відповідного локального адміністратора (один адміністратор = одна стрічка у таблиці), а у випадаючому списку (5) автоматично обирається університет. При натисканні у таблиці (10) на будь-яку іншу клітину відбувається очистка полів (1-4).

Стовпці у таблиці (10) можна розтягувати.

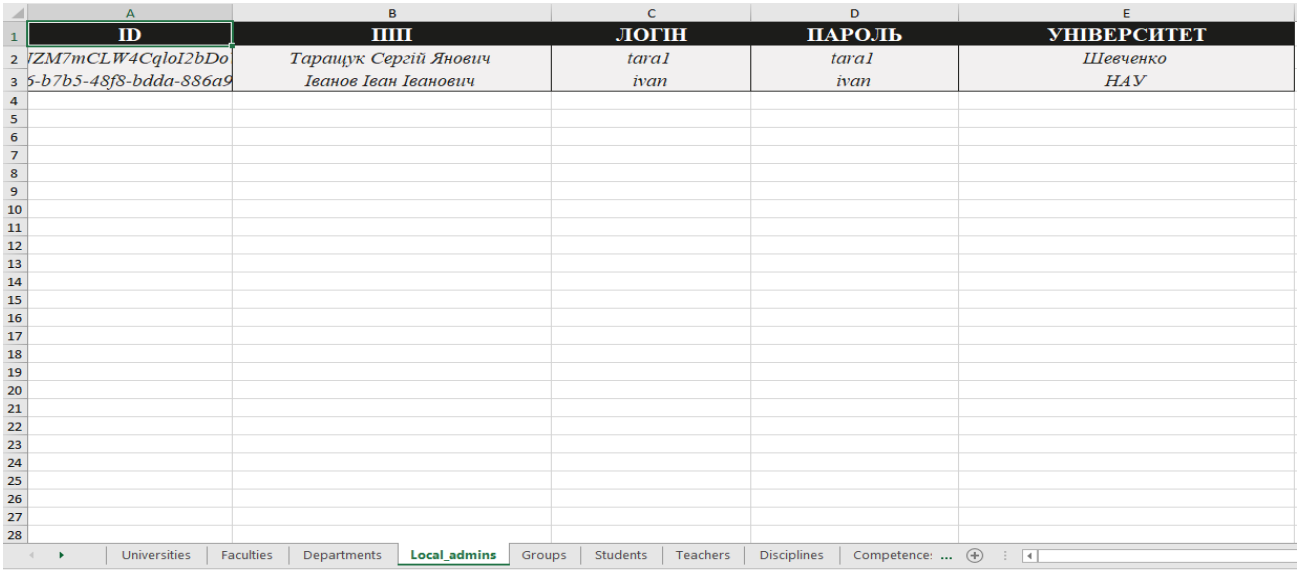
Таблицю (10) можна прокручувати.

При натисканні на кнопку (6) відбувається створення нового локального адміністратора та занесення його у відповідну БД. Дані нового локального адміністратора беруться із полів (2-4) та випадаючого списку 5. При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (7) відбувається оновлення обраного локального адміністратора. Нові дані беруться із полів (2-4) та випадаючого списку 5. Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (8) відбувається видалення обраного локального адміністратора. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (8) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Local_admins». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Local_admins» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.9.



ID	ПІП	ЛОГІН	ПАРОЛЬ	УНІВЕРСИТЕТ
ZM7mCLW4CqloI2bDo	Таращук Сергій Янович	tara1	tara1	Шевченко
5-b7b5-48f8-bdda-886a9	Іванов Іван Іванович	ivan	ivan	НАУ

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence: ...

Рис. 4.9. Результат експорту даних локальних адміністраторів

6.3. Демонстрація функціоналу локального адміністратору

Інформаційне меню локального адміністратора представлено на рисунку 4.10.

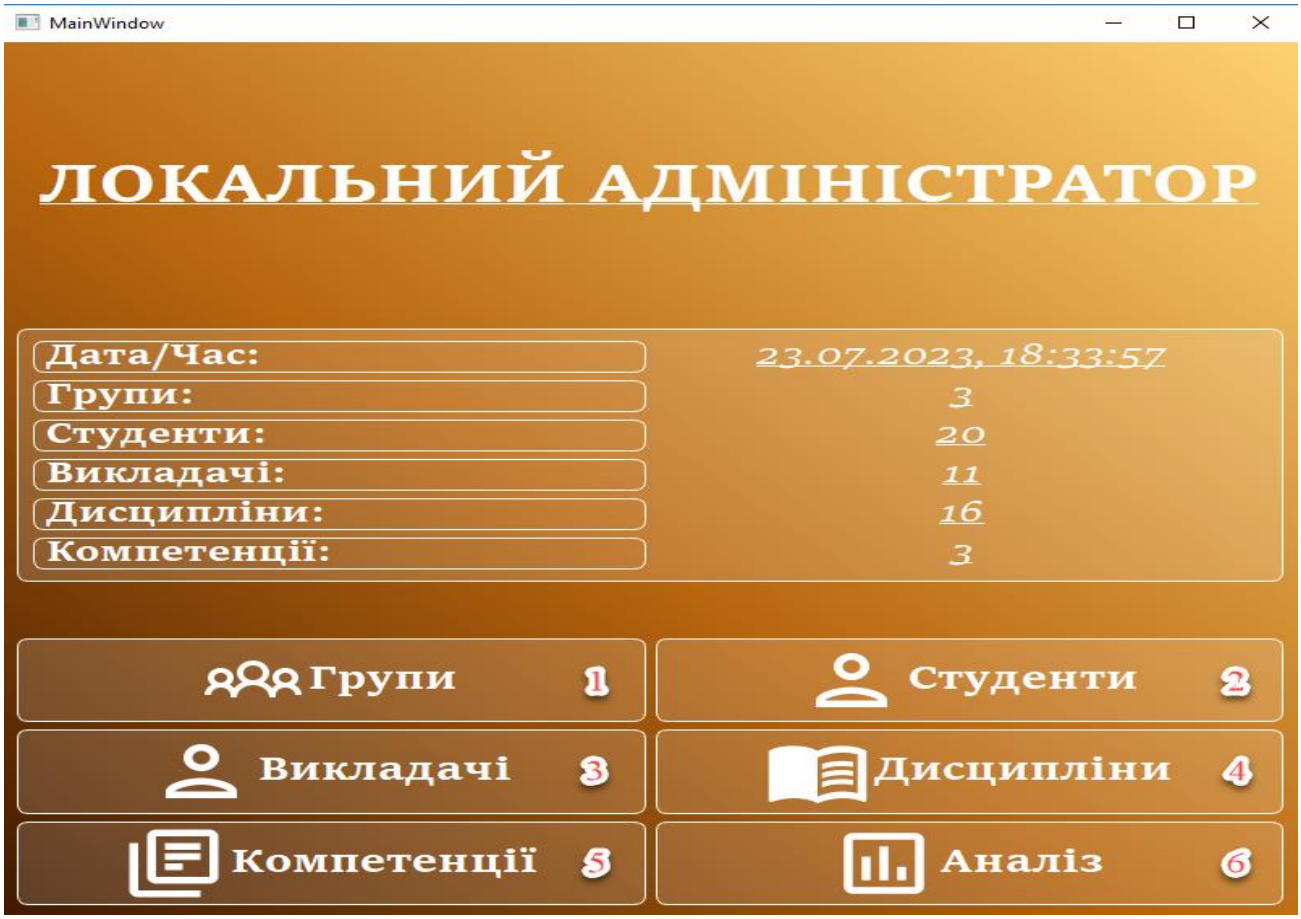


Рис. 4.10. Інформаційне меню локального адміністратора

Інформаційне меню локального адміністратора налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- кнопка меню груп (1);
- кнопка меню студентів (2);
- кнопка меню викладачів (3);
- кнопка меню дисциплін (4);
- кнопка меню компетенцій (5);
- кнопка меню аналізу компетенцій (6).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

При натисканні на кнопку меню груп (1) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню студентів (2) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню викладачів (3) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню дисциплін (4) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню компетенцій (5) відбувається перехід до відповідного меню. При чому поточне меню приховується.

При натисканні на кнопку меню аналізу компетенцій (6) відбувається перехід до відповідного меню. При чому поточне меню приховується.

Над кнопками (1-6) знаходиться інформаційна панель, на якій відображається актуальна дата та час та актуальна кількість груп / студентів / викладачів / дисциплін / компетентностей у системі. Взаємодіяти із цією панеллю користувач не може – можливий лише перегляд.

Меню груп представлено на рисунку 4.11.

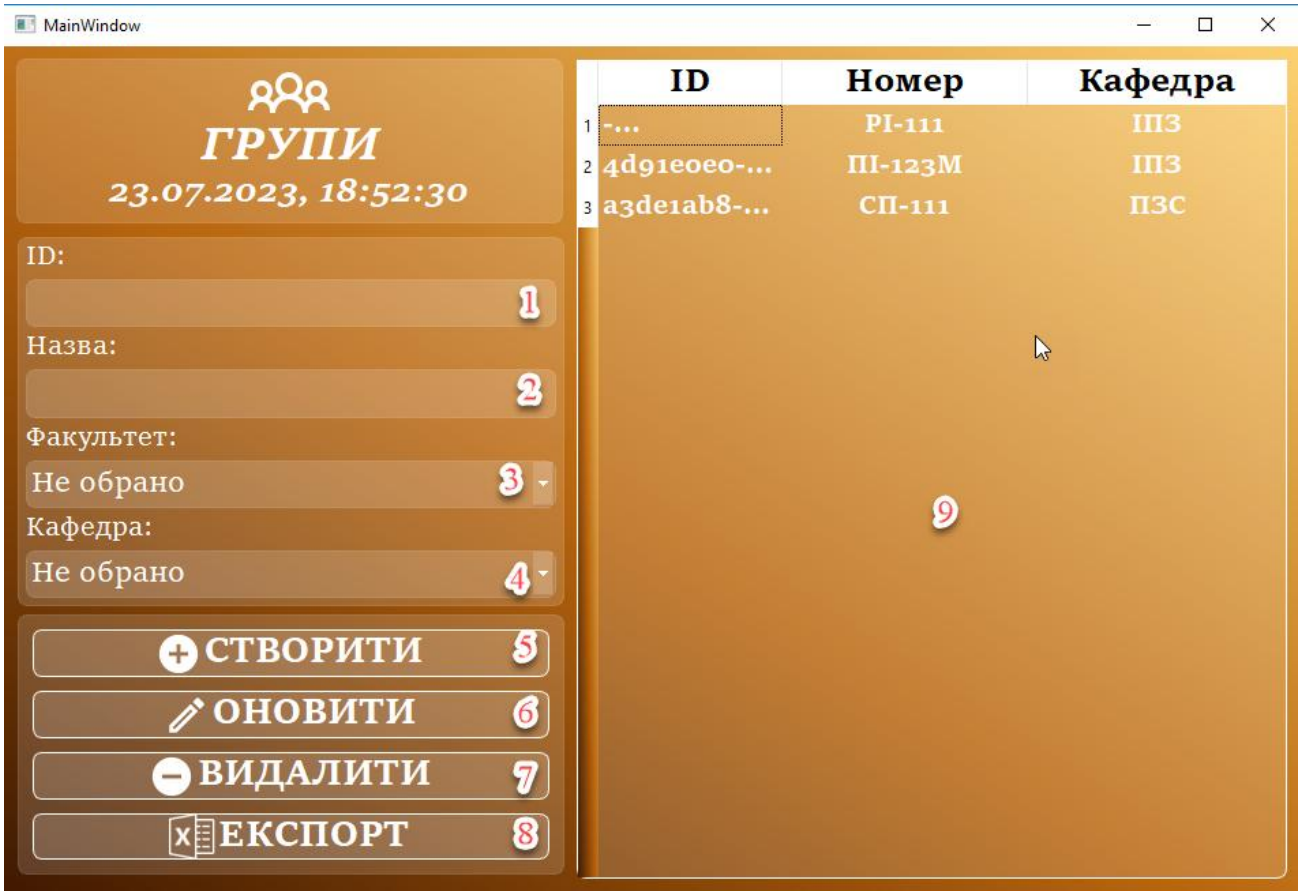


Рис. 4.11. Меню груп

Меню груп налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу назви (номеру) групи (2);
- випадаючий список факультетів (3);
- випадаючий список кафедр (4);
- кнопка створення груп (5);
- кнопка оновлення груп (6);
- кнопка видалення груп (7);
- кнопка експорту даних груп (8);
- таблиця для груп (9).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір

(стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (3) заповнюється при запуску форми автоматично.

Випадаючий список (4) заповнюється при запуску форми автоматично.

При натисканні у таблиці (9) на будь-яку клітину із першого стовпця (стовпець «ID») у поля (1-2) відбувається автоматичне занесення даних відповідної навчальної групи (одна група = одна стрічка у таблиці), а у випадаючих списках (3-4) автоматично обираються факультет та кафедра. При натисканні у таблиці (9) на будь-яку іншу клітину відбувається очистка полів (1-2).

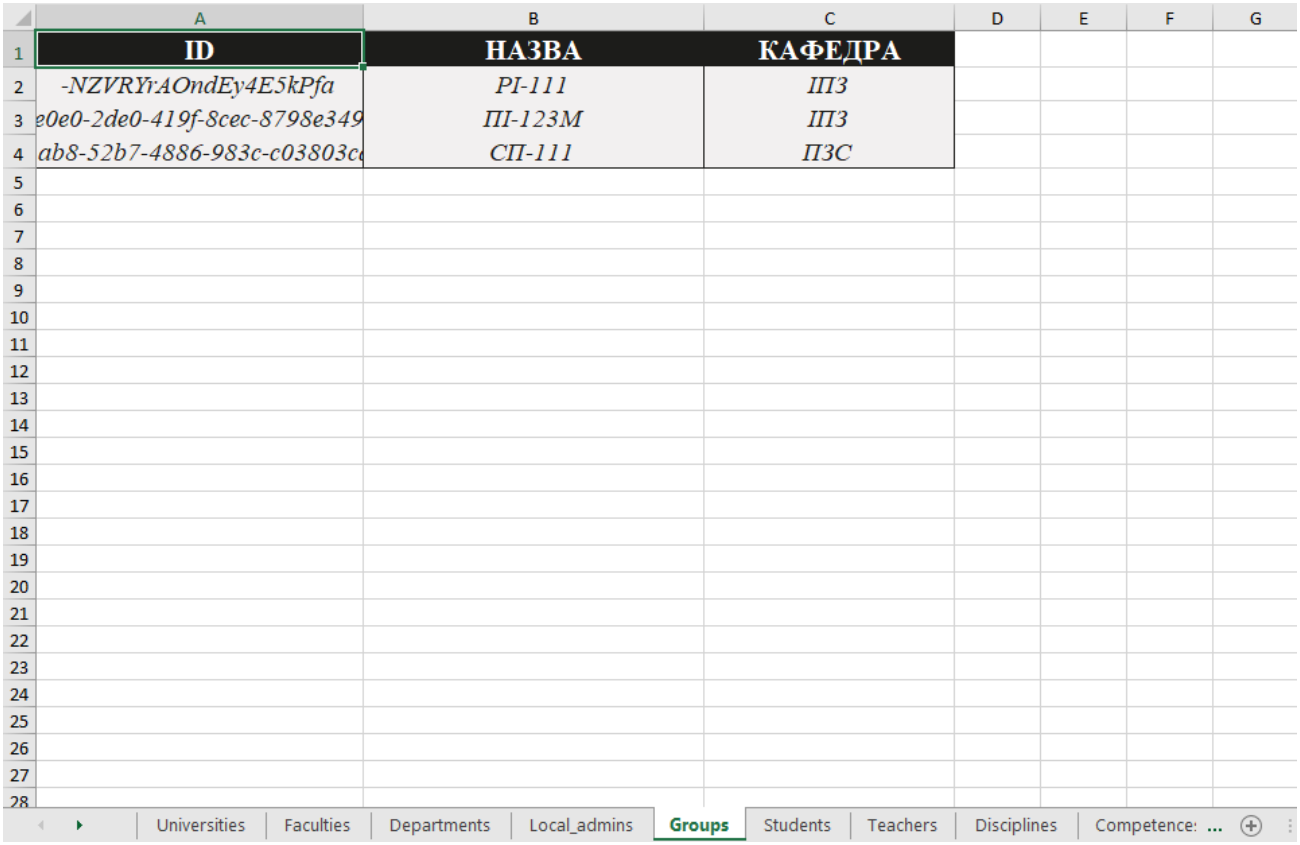
Стовпці у таблиці (9) можна розтягувати.

При натисканні на кнопку (5) відбувається створення нової групи та занесення її у відповідну БД. Дані нової групи беруться із полів (1-2) та випадаючих списків (3-4). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (6) відбувається оновлення обраної групи. Нові дані беруться із полів (1-2) та випадаючих списків (3-4). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (7) відбувається видалення обраної групи. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (8) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Groups». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Groups» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.12.



ID	НАЗВА	КАФЕДРА
-NZVRYrAOndEy4E5kPfa	РІ-111	ІПЗ
e0e0-2de0-419f-8cec-8798e349	ПІ-123М	ІПЗ
ab8-52b7-4886-983c-c03803c	СП-111	ПЗС

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence: ...

Рис. 4.12. Результат експорту даних навчальних груп

Меню студентів представлено на рисунку 4.13.

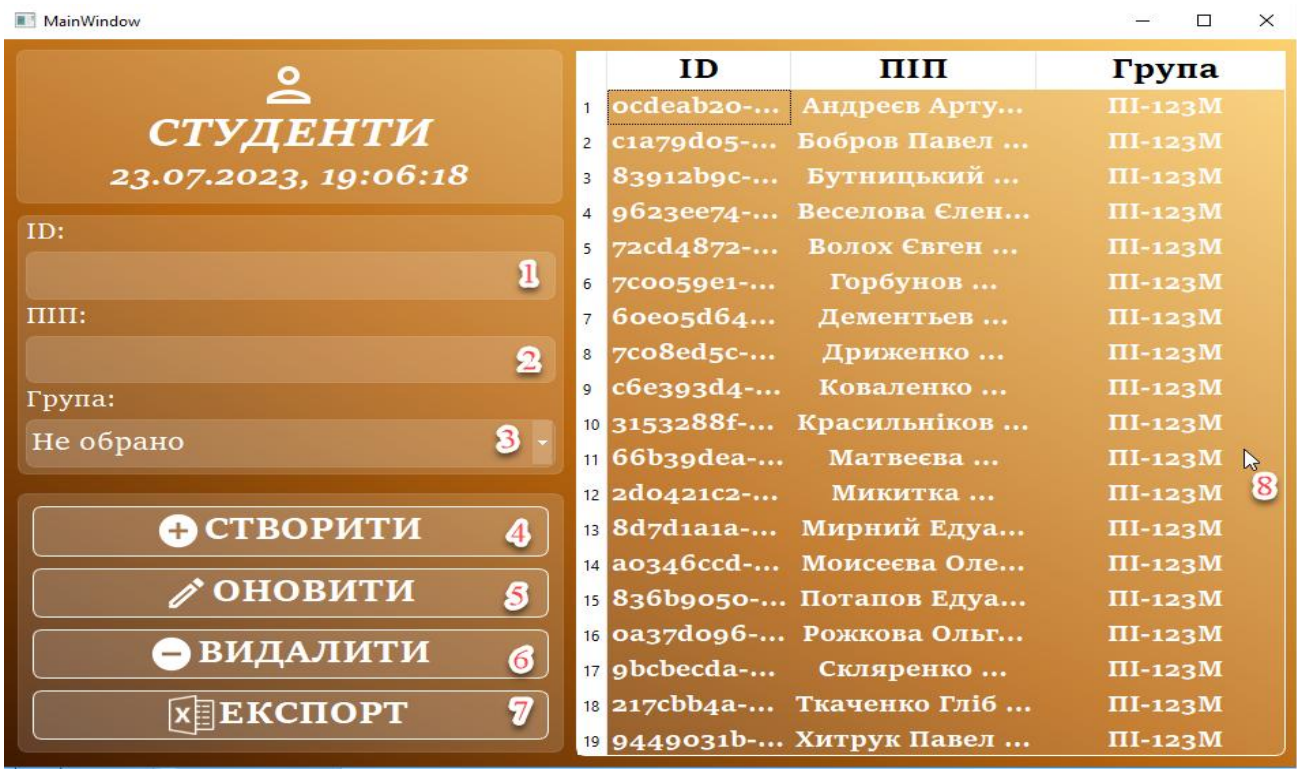


Рис. 4.13. Меню студентів

Меню студентів налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу ПІП студента (2);
- випадаючий список груп (3);
- кнопка створення студенту (4);
- кнопка оновлення студенту (5);
- кнопка видалення студенту (6);
- кнопка експорту даних студентів (7);
- таблиця для студентів (8).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (3) заповнюється при запуску форми автоматично.

При натисканні у таблиці (8) на будь-яку клітину із першого стовпця (стовпець «ID») у поле (1) та поле (2) відбувається автоматичне занесення даних відповідного студенту (один студент = одна стрічка у таблиці), а у випадаючому списку автоматично обирається група. При натисканні у таблиці (8) на будь-яку іншу клітину відбувається очистка полів (1) та (2).

Стовпці у таблиці (8) можна розтягувати.

При натисканні на кнопку (4) відбувається створення нового студенту та занесення його у відповідну БД. ПІП нового студенту береться із поля (2), а значення групи – із випадаючого списку (3). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (5) відбувається оновлення обраного студенту. Нове ПІП студенту береться із поля (2), а нова група – із випадаючого списку (3). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (6) відбувається видалення обраного студенту. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (7) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Students». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Students» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.14.

ID	ПІП	ГРУПА
096-451f-4af6-aadf-9ff365	Рожкова Ольга Богданівна	ПІ-123М
0-d330-4929-ab1a-94d3b	Андреєв Артур Федорович	ПІ-123М
4a-f465-462d-a57a-42f05e	Ткаченко Гліб Борисович	ПІ-123М
2-5e0a-4a4e-a221-ba728	Микитка Честислав Олегович	ПІ-123М
8f-bda1-42c8-ab25-f87f03	Красильніков Богдан Васильович	ПІ-123М
22-4cd3-4c0f-97fa-2e29a5	Шашков Даніїл Данилович	ПІ-123М
54-162d-4e4f-9e78-58d214	Дементьев Юрій Романович	ПІ-123М
a-d8bc-4a9f-adae-8d4d70	Матвеєва Кристина Григорівна	ПІ-123М
72-b989-4562-b4b4-7bebe	Волох Євген Устимович	ПІ-123М
1-6069-4e3f-87b0-60996b	Горбунов Богдан Валериєвич	ПІ-123М
5c-d908-4aab-b3fe-e64f26	Дриженко Євгеній Семенович	ПІ-123М
50-a56b-4426-bf2d-729c2	Потапов Едуард Васильович	ПІ-123М
c-600d-46d2-b31d-ae94e	Бутницький Зоремир Русланович	ПІ-123М
1a-12c8-41fc-803f-d31af9	Мирний Едуард Романович	ПІ-123М
b-2ae7-4e8a-abde-8257e	Хитрук Павел Борисович	ПІ-123М
74-899f-4f7c-b421-858892	Веселова Єлена Григорівна	ПІ-123М
a-d6a8-4aa9-8dd1-d4482	Скляренко Єкатерина Олексіївна	ПІ-123М
d-a37c-4b1d-a414-46984	Моисеєва Олеся Петрівна	ПІ-123М
5-d148-4675-8a53-a55e2	Бобров Павел Валериєвич	ПІ-123М
4-0913-45e5-8f1a-9b4264	Коваленко Еміль Янович	ПІ-123М

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence:

Рис. 4.14. Результат експорту даних студентів

Меню викладачів представлено на рисунку 4.15.

Рис. 4.15. Меню викладачів

Меню викладачів налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу ПІП викладача (2);
- поле для вводу логіну (3);
- поле для вводу паролю (4);
- випадаючий список факультетів (5);
- випадаючий список кафедр (6);
- кнопка створення адміністраторів (7);
- кнопка оновлення адміністраторів (8);
- кнопка видалення адміністраторів (9);
- кнопка експорту даних адміністраторів (10);
- таблиця для локальних адміністраторів (11).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише

перегляд та копіювання значення.

Випадаючий список (5) заповнюється при запуску форми автоматично.

Випадаючий список (6) заповнюється при запуску форми автоматично.

При натисканні у таблиці (11) на будь-яку клітину із першого стовпця (стовпець «ID») у поля (1-4) відбувається автоматичне занесення даних відповідного викладача (один викладач = одна стрічка у таблиці), а у випадаючих списках (5-6) автоматично обирається факультет та кафедра. При натисканні у таблиці (11) на будь-яку іншу клітину відбувається очистка полів (1-4).

Стовпці у таблиці (11) можна розтягувати.

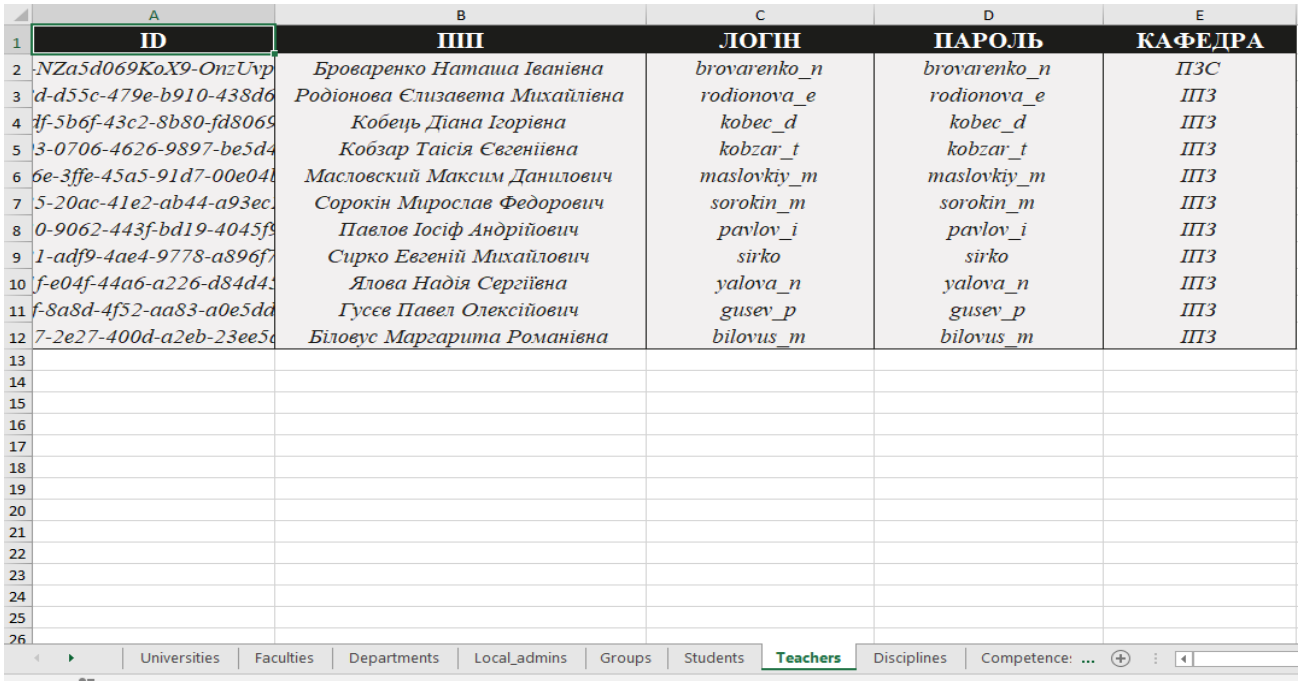
Таблицю (11) можна прокручувати.

При натисканні на кнопку (7) відбувається створення нового викладача та занесення його у відповідну БД. Дані нового викладача беруться із полів (2-4) та випадаючих списків (5-6). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (8) відбувається оновлення викладача. Нові дані беруться із полів (2-4) та випадаючих списків (5-6). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (9) відбувається видалення обраного викладача. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (10) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Teachers». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Teachers» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.16.



ID	ПІП	ЛОГІН	ПАРОЛЬ	КАФЕДРА
-NZa5d069KoX9-OnzUvp	Бровaренко Наташа Іванівна	brovarenko_n	brovarenko_n	ПЗС
d-d55c-479e-b910-438d6	Родіонова Єлизавета Михайлівна	rodionova_e	rodionova_e	ІПЗ
df-5b6f-43c2-8b80-fd8069	Кобець Діана Ігорівна	kobec_d	kobec_d	ІПЗ
3-0706-4626-9897-be5d4	Кобзар Таісія Євгеніївна	kobzar_t	kobzar_t	ІПЗ
6e-3ffe-45a5-91d7-00e04	Масловский Максим Данилович	maslovkiy_m	maslovkiy_m	ІПЗ
5-20ac-41e2-ab44-a93ec	Сорокін Мирослав Федорович	sorokin_m	sorokin_m	ІПЗ
0-9062-443f-bd19-4045f9	Павлов Іосіф Андрійович	pavlov_i	pavlov_i	ІПЗ
1-adf9-4ae4-9778-a896f7	Сирко Евгеній Михайлович	sirko	sirko	ІПЗ
f-e04f-44a6-a226-d84d4	Ялова Надія Сергіївна	yalova_n	yalova_n	ІПЗ
f-8a8d-4f52-aa83-a0e5dd	Гусєв Павел Олексійович	gusev_p	gusev_p	ІПЗ
7-2e27-400d-a2eb-23ee5	Біловус Маргарита Романівна	bilovus_m	bilovus_m	ІПЗ

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence: ...

Рис. 4.16. Результат експорту даних викладачів

Меню дисциплін представлено на рисунку 4.17.

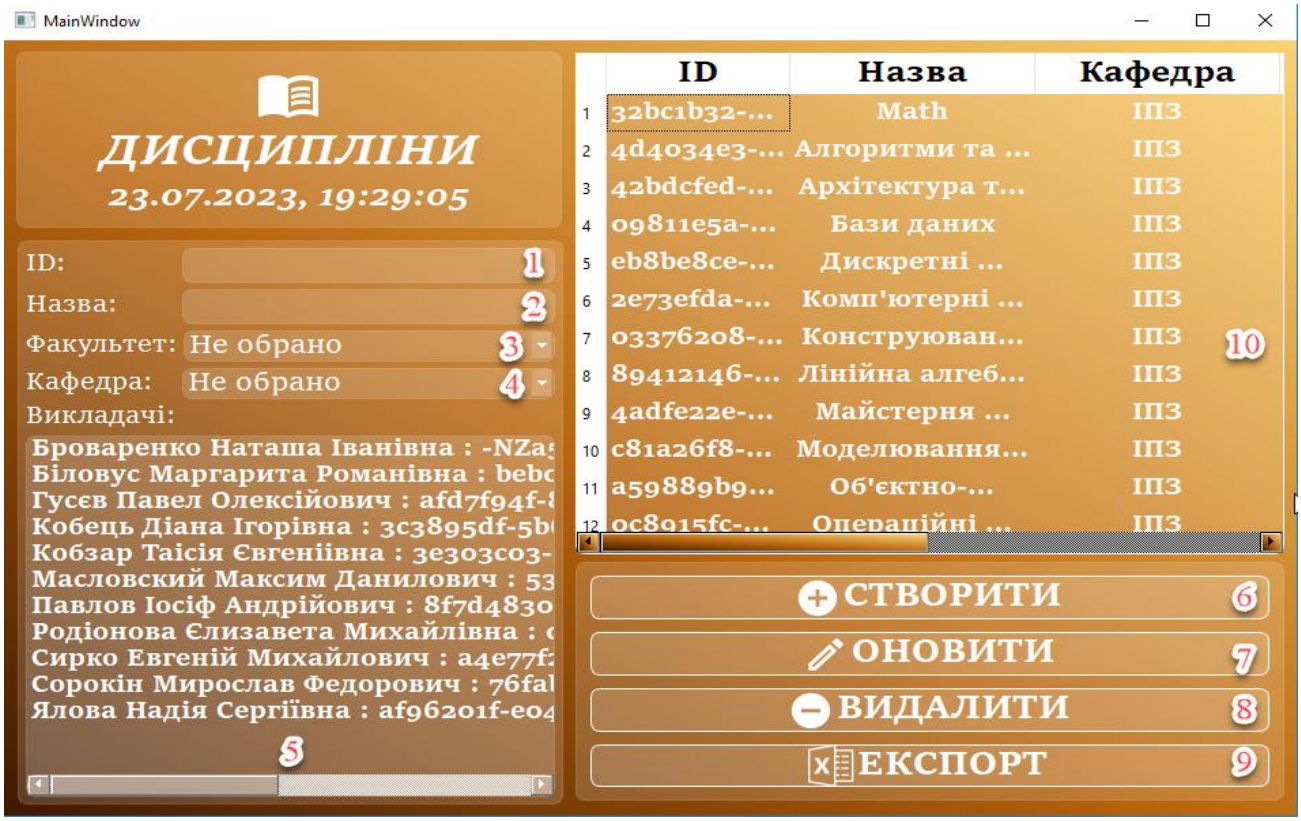


Рис. 4.17. Меню дисциплін

Меню дисциплін налічує наступні інтерактивні елементи, із якими

може взаємодіяти користувач:

- поле для id (1);
- поле для вводу назви дисципліни (2);
- випадаючий список факультетів (3);
- випадаючий список кафедр (4);
- список викладачів (5);
- кнопка створення дисципліни (6);
- кнопка оновлення дисципліни (7);
- кнопка видалення дисципліни (8);
- кнопка експорту даних дисциплін (9);
- таблиця для дисциплін (10).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (3) заповнюється при запуску форми автоматично.

Випадаючий список (4) заповнюється при запуску форми автоматично.

Список (5) заповнюється при запуску форми автоматично.

При натисканні у таблиці (10) на будь-яку клітину із першого стовпця (стовпець «ID») у поля (1-2) відбувається автоматичне занесення даних відповідної дисципліни (одна дисципліна = одна стрічка у таблиці), а у випадаючих списках (3-4), та списку (5) автоматично обираються факультет, кафедра та викладач/викладачі. При натисканні у таблиці (10) на будь-яку іншу клітину відбувається очистка полів (1-2).

Стовпці у таблиці (10) можна розтягувати.

Таблицю (10) можна прокручувати.

При натисканні на кнопку (6) відбувається створення нової дисципліни та занесення її у відповідну БД. Дані нової дисципліни беруться із полів (1-2), випадаючих списків (3-4) та списку (5). При чому, id генерується

автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (7) відбувається оновлення обраної дисципліни. Нові дані беруться із полів (1-2), випадаючих списків (3-4) та списку (5). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (8) відбувається видалення обраної дисципліни. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (9) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Disciplines». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Disciplines» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.18.

	A	B	C
1	ID	НАЗВА	КАФЕДРА
2	6208-a0e2-473a-bf6a-a314e8a	Конструювання та документування ПЗ	ІПЗ
3	14ce-9f0d-4a29-8464-dc9842eb	Програмування мобільних пристроїв	ІПЗ
4	e5a-6adc-47c4-abc5-7b855d90	Бази даних	ІПЗ
5	15fc-257c-4463-a1ac-8e12a498	Операційні системи	ІПЗ
6	efda-3dd5-4d8b-b3c2-f4db3ce0	Комп'ютерні технології аналізу даних	ІПЗ
7	b32-bd3e-4068-8677-6548055	Math	ІПЗ
8	cfed-7010-4749-827b-9fb83fb0	Архітектура та проектування ПЗ	ІПЗ
9	22e-71d1-471e-ad11-0ae45c00	Майстерня розробки ПЗ	ІПЗ
10	34e3-cd1d-424c-abef-013aed3e	Алгоритми та структури даних	ІПЗ
11	6e9b-6639-4b76-a606-2eb04f8	Фахова іноземна мова	ІПЗ
12	146-bc87-4c29-a520-4b2a4bd	Лінійна алгебра та аналітична геометрія	ІПЗ
13	89b9-5549-43af-8ceb-daeb8390	Об'єктно-орієнтоване програмування	ІПЗ
14	26f8-e385-47f5-b873-610aa01	Моделювання та аналіз ПЗ	ІПЗ
15	181d-d52a-4ceb-8651-0d691fd	Якість програмного забезп та тестування	ІПЗ
16	8ce-d453-44da-aee8-1ed32600	Дискретні структури	ІПЗ
17	c25-269f-4e1d-9004-d66a3120	Програмування для Інтернет	ІПЗ
18			
19			
20			
21			
22			
23			
24			

Universities | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competence: ...

Рис. 4.18. Результат експорту даних дисциплін

Меню компетенцій представлено на рисунку 4.19.

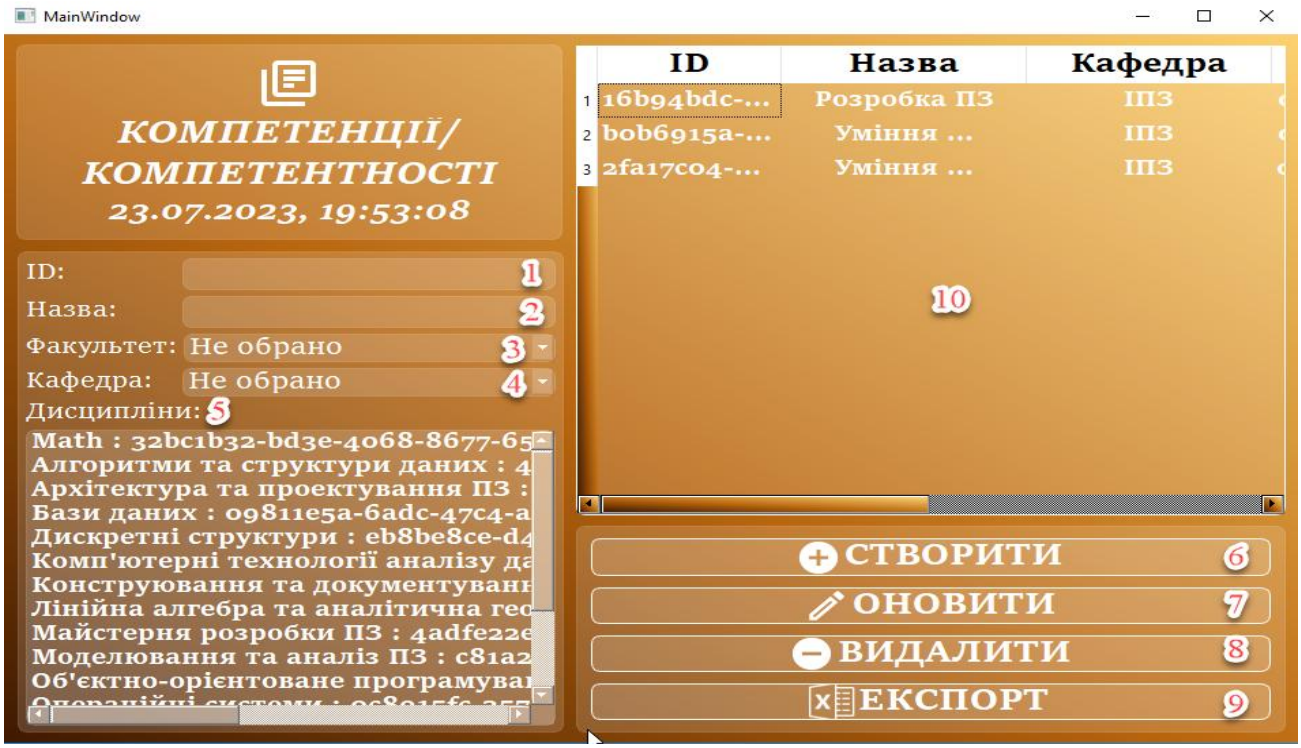


Рис. 4.19. Меню компетенцій

Меню компетенцій налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- поле для вводу назви компетенції (2);
- випадаючий список факультетів (3);
- випадаючий список кафедр (4);
- список дисциплін (5);
- кнопка створення компетенції (6);
- кнопка оновлення компетенції (7);
- кнопка видалення компетенції (8);
- кнопка експорту даних компетенції (9);
- таблиця для дисциплін (10).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (3) заповнюється при запуску форми автоматично.

Випадаючий список (4) заповнюється при запуску форми автоматично.

Список (5) заповнюється при запуску форми автоматично.

При натисканні у таблиці (10) на будь-яку клітину із першого стовпця (стовпець «ID») у поля (1-2) відбувається автоматичне занесення даних відповідної компетенції (одна компетенція = одна стрічка у таблиці), а у випадаючих списках (3-4), та списку (5) автоматично обираються факультет, кафедра та дисципліна/дисципліни. При натисканні у таблиці (10) на будь-яку іншу клітину відбувається очистка полів (1-2).

Стовпці у таблиці (10) можна розтягувати.

Таблицю (10) можна прокручувати.

При натисканні на кнопку (6) відбувається створення нової компетенції та занесення її у відповідну БД. Дані нової компетенції беруться із полів (1-2), випадаючих списків (3-4) та списку (5). При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (7) відбувається оновлення обраної компетенції. Нові дані беруться із полів (1-2), випадаючих списків (3-4) та списку (5). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (8) відбувається видалення обраної компетенції. Для видалення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (9) відбувається експорт даних із віддаленої БД до файлу SCA_DATA.xlsx до сторінки «Competences». Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Сторінка «Competences» також автоматично створюється, якщо її раніше у документі SCA_DATA.xlsx не було. Скріншот автоматично сформованої таблиці при експорті представлено на рисунку 4.20.

	A	B	C
1	ID	НАЗВА	КАФЕДРА
2	16b94bdc-54d5-475d-8f97-6f82d29c4e65	Розробка ПЗ	ІПЗ
3	2fa17c04-0ab3-4ad7-b81a-cb736707b2f7	Уміння працювати с БД1	ІПЗ
4	b0b6915a-e4e9-42f3-ba7c-7da0fa117712	Уміння працювати з ВЕБ технологіями	ІПЗ

... | Faculties | Departments | Local_admins | Groups | Students | Teachers | Disciplines | Competences

Рис. 4.20. Результат експорту даних компетенцій

Меню аналізу компетенцій студента представлено на рисунку 4.21.

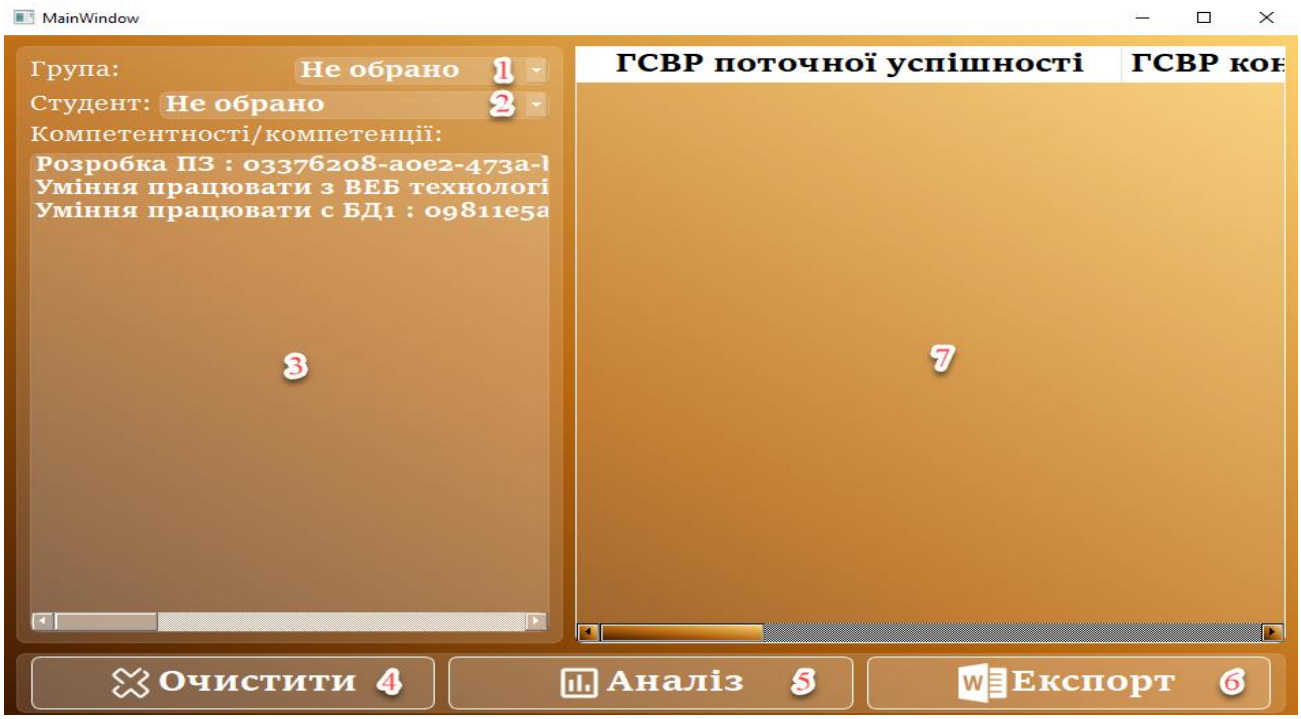


Рис. 4.21. Меню аналізу компетенцій студента

Меню аналізу компетенцій налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- випадаючий список груп (1);
- випадаючий список студентів (2);
- список компетенцій (3);
- кнопка очистки (4);
- кнопка аналізу компетенцій (5);
- кнопка експорту результатів аналізу (6);
- таблиця для результатів аналізу (7).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Випадаючий список (1) заповнюється при запуску форми автоматично.

Випадаючий список (2) заповнюється при запуску форми автоматично.

Список (3) заповнюється при запуску форми автоматично.

Стовпці у таблиці (10) можна розтягувати.

Таблицю (10) можна прокручувати.

При натисканні на кнопку (4) відбувається очистка обраних у випадаючих списках (1-2) та списку (3) значень.

При натисканні на кнопку (5) відбувається аналіз обраних у списку (3) компетентностей студента, обраного у випадаючому списку (2), та завантаження результатів у таблицю (7).

При натисканні на кнопку (6) відбувається експорт даних із віддаленої БД до NAME (GROUP) - Competence analyze result.docx, NAME - це ПІП студента (береться із випадаючого списку (2)), а GROUP – це його група (береться із випадаючого списку (1)). Документ автоматично створюється у директорії із виконавчим файлом (у випадку, якщо раніше даного документу там не було). Звіт про результати аналізу, сформований автоматично при експорті, представлено у додатку 7.

6.4. Демонстрація функціоналу викладача

Меню викладача представлено на рисунку 4.22.

Рис. 4.22. Меню викладача

Меню викладача налічує наступні інтерактивні елементи, із якими може взаємодіяти користувач:

- поле для id (1);
- випадаючий список груп (2);
- випадаючий список студентів (3);
- випадаючий список дисциплін (4);
- випадаючий список типів (5);
- поле для короткого значення оцінки (3-5) (6);
- поле для довгого значення оцінки (60-100) (7);
- кнопка створення компетенції (8);
- кнопка оновлення компетенції (9);
- кнопка видалення компетенції (10);

- таблиця для дисциплін (11).

При наведінні курсором миші на кнопки вони підсвічуються. Коли курсор миші покидає межі кнопки – вони повертають свій початковий колір (стан).

Поле (1) не може бути відредаговано вручну – можливий лише перегляд та копіювання значення.

Випадаючий список (2) заповнюється при запуску форми автоматично.

Випадаючий список (3) заповнюється при запуску форми автоматично.

Випадаючий список (4) заповнюється при запуску форми автоматично.

Випадаючий список (5) заповнюється при запуску форми автоматично.

При натисканні у таблиці (11) на будь-яку клітину із першого стовпця (стовпець «ID») у поля (1, 6, 7) відбувається автоматичне занесення даних відповідної оцінки (одна оцінка = одна стрічка у таблиці), а у випадаючих списках (3-5) автоматично обираються група, студент, дисципліна та оцінка. При натисканні у таблиці (11) на будь-яку іншу клітину відбувається очистка полів (1, 6, 7).

Стовпці у таблиці (11) можна розтягувати.

Таблицю (11) можна прокручувати.

При натисканні на кнопку (8) відбувається створення нової оцінки та занесення її у відповідну БД. Дані нової оцінки беруться із полів (1, 6, 7), випадаючих списків (3-5) . При чому, id генерується автоматично (не важливо, що при цьому знаходиться у полі (1)).

При натисканні на кнопку (9) відбувається оновлення обраної оцінки. Нові дані беруться із полів (1, 6, 7), випадаючих списків (3-5). Для оновлення обов’язково у полі (1) повинно бути указано id.

При натисканні на кнопку (10) відбувається видалення обраної оцінки. Для видалення обов’язково у полі (1) повинно бути указано id.

Висновки

У даному розділі проводиться демонстрація розробленого програмного засобу, який реалізує сформовані та досліджені у розділі 2 вимоги (див. пункти 2.2. – 2.3.).

Було продемонстровано роботу застосунку для визначених у розділі 1 ролей користувачів, а саме для головного адміністратора, локального адміністратора та викладача.

Демонстрація включає в себе скріншоти роботи кожного меню застосунку, представлення списку інтерактивних елементів у кожному меню та відповідним описом. Додатково, було продемонстровано як виглядають автоматично сформовані при експорті операційних одиниць документи.

Як можна побачити на рисунках (див. пункти 6.1 – 6.4.), що були представлені у даному розділі, застосунок СКУН має простий та інтуїтивно зрозумілий інтерфейс, для оформлення якого використовуються кольори, що не напружують очі, та «скляний формат» кнопок. Всі інші елементи користувацього інтерфесу (поля для вводу, випадаючі списки, тощо) так само мають «скляний формат», що знімає напругу з очей.

Також, алгоритми, що реалізовані у СКУН, спрямовані на максимальну автоматизацію роботи із даними – дані у поля (там, де це можливо) завантажуються автоматично, деякі дані при оперуванні операційними одиницями також формуються автоматично і ймовірнісний аналіз компетенцій так само практчно повністю автоматизований. У додатку до цього, реалізовано можливість автоматичного експорту даних практично усіх операційних одиниць у зручний формат електронних таблиць, та експорту даних результатів ймовірнісного аналізу у формат електронного документу (все зберігаєтьься у директорії, де знаходиться виконачий файл).

Таким чином, продемонстровані у даному розділі результати дають змогу зробити висновок про те, що розроблена система повністю відповідає визначеним раніше функціональним та нефункціональним вимогам.

ЗАГАЛЬНІ ВИСНОВКИ

Даний дипломний проект є логічним продовженням та повною переробкою мого попереднього дипломного проекту на тему «Мобільний застосунок оцінки і візуалізації стану поточної успішності студента університету (СКУН)».

Перед виконанням даної роботи було проведено повторний аналіз предметної області, якою як для попередньої, так і для нової версії СКУН була та є навчальна середа університету, однак у даному проекті предметна область аналізувалася не з точки зору успішності, а з точки зору компетенцій. Після повторного аналізу було проведено аналіз найбільш відомих зарубіжних програмних засобів по оцінці студентів - як і у випадку пепередньої версії, аналогів до нової версії СКУН виявлено не було, оскільки існуюче програмне забезпечення по оцінці не надає функціонал для роботи із компетентностями, що є базою для СКУН.

Наступним кроком виконання роботи було виявлення, дослідження, систематизація та представлення вимог до системи СКУН. Як результат, було представлено розширені (у порівнянні із аналогічними списками вимог до СКУН) списки функціональних, нефункціональних та системних вимог.

На основі вищезазначених вимог була сформована та описана фізична структура проекту, а саме були систематизовані та описані головні директорії, файли та методи, що реалізують бізнес-логіку застосунку. Аналіз даної фізичної структури дав змогу зробити висновок, що всі визначені раніше вимоги (у тому числі і архітектурні) були виконані.

Для розробленої системи було представлено детальну демонстрацію роботи. Продемонстроване в останньому розділі дає змогу підтвердити сформований раніше висновок про те, що всі поставлені перед фактичною розробкою вимоги були повністю виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Манжула К. О. ДИПЛОМНИЙ ПРОЕКТ бакалавра на тему "Мобільний застосунок оцінки і візуалізації стану поточної успішності студента університету". Київ, 2022. 99 с.
2. 10 best assessment apps. *Mimio Classroom Technology Blog*. [Electronic resource] / Mode of access : https://blog.mimio.com/10-best-assessment-apps (date of access: 17.07.2023).
3. 11 best exam and assessment platforms of 2021 - qorrect. *Qorrect*. [Electronic resource] / Mode of access : https://blog.qorrectassess.com/best-exam-and-assessment-platforms/ (date of access: 17.07.2023).
4. 6 apps that can help with student assessment. *Getting Smart*. [Electronic resource] / Mode of access : https://www.gettingsmart.com/2016/06/10/6-apps-that-can-help-with-student-assessment/ (date of access: 17.07.2023).
5. 8 best assessment tools for educators. *Mentimeter*. [Electronic resource] / Mode of access : https://www.mentimeter.com/blog/interactive-classrooms/best-assessment-tools (date of access: 17.07.2023).
6. Assessment apps for teachers. *Educational App Store*. [Electronic resource] / Mode of access: https://www.educationalappstore.com/app/category/assessment-apps (date of access: 17.07.2023).
7. Best assessment software. *G2 - Business software reviews*. [Electronic resource] / Mode of access : https://www.g2.com/categories/assessment (date of access: 17.07.2023).
8. GoReact reviews. *GetApp*. [Electronic resource] / Mode of access : https://www.getapp.com/hr-employee-management software/a/goreact/reviews/ (date of access: 17.07.2023).

9. Kahoot what is it: features, advantages, disadvantages, and faqs. *techprevue*. [Electronic resource] / Mode of access : https://www.techprevue.com/kahoot/ (date of access: 17.07.2023).

10. Lumio by SMART Reviews. *GetApp*. [Electronic resource] / Mode of access : https://www.getapp.com/education-childcare-software/a/lumio-by-smart/reviews/ (date of access: 17.07.2023).

11. Mentimeter: make fun and interactive presentations. *The Business Blocks*. [Electronic resource] / Mode of access : https://thebusinessblocks.com/gather/mentimeter/ (date of access: 17.07.2023).

12. Quizizz - overview. *TrustRadius*. [Electronic resource] / Mode of access : https://www.trustradius.com/products/quizizz/reviews?qs=pros-and-cons#overview (date of access: 17.07.2023).

13. Restifo D. Best free formative assessment tools and apps. *TechLearningMagazine*. [Electronic resource] / Mode of access : https://www.techlearning.com/how-to/formative-assessment-tools-and-apps (date of access: 17.07.2023).

14. MVC design pattern - geeksforgeeks. *GeeksforGeeks*. [Electronic resource] / Mode of access : https://www.geeksforgeeks.org/mvc-design-pattern/ (date of access: 19.07.2023).

15. Design patterns - MVC pattern. *Online Courses and eBooks Library | Tutorialspoint*. [Electronic resource] / Mode of access : https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm (date of access: 19.07.2023).

16. Hernandez R. D. The model view controller pattern – MVC architecture and frameworks explained. *freeCodeCamp.org*. [Electronic resource] / Mode of access : https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/ (date of access: 19.07.2023).

17. 20 most popular programming languages to learn in 2023. *Phaxis*. [Electronic resource] / Mode of access : https://phaxis.com/2023/05/03/20-most-popular-programming-languages-to-learn-in-2023/ (date of access: 22.07.2023).

18. Discover the top 20 programming languages worth learning. *Staffing Partner*. [Electronic resource] / Mode of access : https://staffingpartner.net/blog/most-wanted-programming-languages/ (date of access: 22.07.2023).

19. Top 10 programming languages to learn in 2023 - geeksforgeeks. *GeeksforGeeks*. [Electronic resource] / Mode of access : https://www.geeksforgeeks.org/top-10-programming-languages-to-learn/ (date of access: 22.07.2023).

20. Veeraraghavan S. Top 20 best programming languages to learn in 2023 | simplilearn. *Simplilearn.com*. [Electronic resource] / Mode of access : https://www.simplilearn.com/best-programming-languages-start-learning-today-article (date of access: 22.07.2023).

21. Fitzpatrick M. Create GUI applications with python and qt5: the hands-on guide to making apps with python. Independently Published, 2020.

22. PySide6. *PyPI*. [Electronic resource] / Mode of access : https://pypi.org/project/PySide6/ (date of access: 22.07.2023).

23. Fitzpatrick M. PySide6 tutorial 2023, create python guis with qt. *Python GUIs*. [Electronic resource] / Mode of access : https://www.pythonguis.com/pyside6-tutorial/ (date of access: 22.07.2023).

24. Embedded software development tools & cross platform IDE | qt creator. *Qt | Tools for Each Stage of Software Development Lifecycle*. [Electronic resource] / Mode of access : https://www.qt.io/product/development-tools (date of access: 22.07.2023).

25. Tutorials | qt creator manual. *Qt Documentation | Home*. [Electronic resource] / Mode of access : https://doc.qt.io/qtcreator/creator-tutorials.html (date of access: 22.07.2023).

26. ІЩо таке firebase?. *Avada Media*. [Electronic resource] / Mode of access : https://avada-media.ua/ua/services/firebase/ (дата звернення: 22.07.2023).

27. Firebase advantages and disadvantages. *Back4App Blog*. [Electronic resource] / Mode of access : https://blog.back4app.com/firebase-advantages-and-disadvantages/ (date of access: 22.07.2023).

28. Top 10 advantages of firebase. *Back4App Blog*. [Electronic resource] / Mode of access : https://blog.back4app.com/advantages-of-firebase/ (date of access: 22.07.2023).

29. Advantages and disadvantages of firebase. *LinkedIn*. [Electronic resource] / Mode of access : https://www.linkedin.com/pulse/advantages-disadvantages-firebase-nav-adalyn (date of access: 22.07.2023).

30. 5 benefits of firebase database must know in 2022 | flutter agency. *Flutter Agency - Mobile App Designing, Development & Consulting*. [Electronic resource] / Mode of access : https://flutteragency.com/5-benefits-firebase-database-system/ (date of access: 22.07.2023).

31. Bringing firebase admin to python. *The Firebase Blog*. [Electronic resource] / Mode of access : https://firebase.blog/posts/2017/04/bringing-firebase-admin-to-python/ (date of access: 22.07.2023).

32. Openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files – openpyxl 3.1.2 documentation. *openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files – openpyxl 3.1.2 documentation*. [Electronic resource] / Mode of access : https://openpyxl.readthedocs.io/en/stable/ (date of access: 22.07.2023).

33. Практичний підручник з python openpyxl із прикладами - інший. *Огляди, Ігри, Розваги, Липень 2023*. [Electronic resource] / Mode of access : https://uk.myservername.com/hands-python-openpyxl-tutorial-with-examples (дата звернення: 22.07.2023).

34. Quickstart – python-docx 0.8.11 documentation. *python-docx – python-docx 0.8.11 documentation*. [Electronic resource] / Mode of access : https://python-docx.readthedocs.io/en/latest/user/quickstart.html (date of access: 22.07.2023).

35. How to use python-docx. *vegibit*. [Electronic resource] / Mode of access : https://vegibit.com/how-to-use-python-docx/ (date of access: 22.07.2023).

ДОДАТОК А

Порівняльна характеристика програм – аналогів

	Цільова аудиторія	*Функції*	*Переваги*	*Недоліки*	Вартість
Lumio	Викладачі, спеціалісти по учбовим технологіям	Створення інтерактивних уроків, та контрольних, зберігання даних успішності, імпорт даних у .pdf, .pptx.	Простота використання, наявність готових шаблонів, інтеграція з Google, інтеграція з YouTube, інтеграція з Desmos, великий набір типів завдань та активностей, перегляд активності студентів у режимі реального часу, ігровий інтерфейс.	Відсутність української локалізації, відсутність інтеграції з Google Classroom.	Є безкоштовна версія, 59$
Kahoot!	Викладачі, студенти	Створення інтерактивних уроків, та контрольних, зберігання даних успішності, імпорт даних у .pdf, .pptx., автоматичне формування звітів та надання доступу до них іншим викладачам, можливість ділитися створеними тестами з іншими викладачами	Простота використання, наявність готових шаблонів, інтеграція з Google, інтеграція з YouTube, інтеграція з Google classroom, великий набір типів завдань та активностей, ігровий інтерфейс.	Відсутність української локалізації, відстеження рівня прогресу студента є складним процесом, необхідне постійне wi-fi з'єднання, доступ до гаджетів також може бути проблемою.	Є безкоштовна версія, від 3€
GoReact	Викладачі	Управління оцінюванням, сертифікація та ліцензування, імпорт/експорт даних, індивідуальне оцінювання, управління навчанням, багатокористувацька співпраця, управління ефективністю, показники ефективності, прокторинг, звітність та статистика, звітність/аналітика, скоринг, тестування навичок, оцінка навичок, створення тестів/вікторин, інтеграція зі	Дуже простий у використанні як для викладачів, так і для студентів, студентам подобається зворотній зв'язок, який вони отримують від професора, дуже інтуїтивно зрозумілий користувацький інтерфейс	Відсутність української локалізації, кольорова гама може бути занадто різкою, періодичні проблеми із звуком, необхідне постійне wi-fi з'єднання, відсутність безкоштовної версії	59$
Quizizz	Викладачі	Створення інтерактивних уроків, та контрольних, зберігання даних успішності, автоматичне формування звітів та надання доступу до них іншим викладачам, можливість ділитися створеними тестами з іншими викладачами	Простота використання, можливість кастомізації інтерфейсу, доступний на всіх платформах, наявність бібліотеки шаблонів	Відсутність української локалізації, у деяких випадках інтерфейс складний, замало безкоштовних шаблонів	Є безкоштовна версія, 19$
Mentimeter	Викладачі	Створення інтерактивних уроків, та контрольних, зберігання даних успішності, автоматичне формування звітів та надання доступу до них іншим викладачам, можливість ділитися створеними тестами з іншими викладачами	Простота використання, можливість кастомізації інтерфейсу, доступний на всіх платформах, наявність бібліотеки шаблонів, експорт результатів у різних форматах, є можливість офлайн-роботи, вбудований механізм аналітики	Сильна обмеженість безкоштовної версії, відсутність української локалізації	Є безкоштовна версія, 11.99$

Рис. А.1. Порівняльна характеристика програм – аналогів

Текст (лістінги) основних блоків програми

```
import datetime
import os

import PySide6
from PySide6 import QtWidgets, QtCore
from PySide6.QtCore import QTimer
from PySide6.QtGui import Qt
from PySide6.QtWidgets import QMainWindow, QAbstractScrollArea
from openpyxl import load_workbook, Workbook
from openpyxl.styles import Font, PatternFill, Alignment

import DB_Helper
from PaintHelper import paintCells
from UI.ma_admins_menu import MA_LocalAdmins_Menu_UI
import Controllers.MA_Controller


class MainAdminLocalAdminsMenuWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.ma_mm_window = None

        self.table_headers = ["ID", "ПІП", "Логін", "Пароль",
"Університет"]

        self.ui = MA_LocalAdmins_Menu_UI()
        self.ui.setupUi(self)

        self.timer = QTimer(self)
        self.timer.timeout.connect(self.setTime)
        self.timer.start(1000)


self.ui.AdminsTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlways
On)
        self.ui.AdminsTable.setColumnCount(5)
```

```
        self.ui.AdminsTable.setHorizontalHeaderLabels(self.table_headers)
        self.ui.AdminsTable.setColumnWidth(1, 150)
        self.ui.AdminsTable.setColumnWidth(2, 210)
        self.ui.AdminsTable.setColumnWidth(3, 210)
        self.ui.AdminsTable.setColumnWidth(4, 210)
        self.ui.AdminsTable.horizontalHeader().setStyleSheet(
            "QHeaderView { "
            "font-size: 16pt;"
            "font: 700 18pt \"Sitka Small\";}"
        )
        self.ui.admin_id_tf.setReadOnly(True)

        self.ui.AdminsTable.cellClicked.connect(self.cellChecked)
        self.ui.CreateAdminBtn.clicked.connect(self.createNewLocalAdmin)
        self.ui.UpdateAdminBtn.clicked.connect(self.updateLocalAdmin)
        self.ui.DeleteAdminBtn.clicked.connect(self.deleteLocalAdmin)
        self.ui.ExportAdminsBtn.clicked.connect(self.export)

        self.fillUniversitiesSpinner()
        self.loadAdminsData()

    def setTime(self):
        now = datetime.datetime.now()
        formatDate = now.strftime("%d.%m.%Y, %H:%M:%S")
        self.ui.DateAndTimeLabel.setText(formatDate)

    def fillUniversitiesSpinner(self):
        res = DB_Helper.getRef().child("Universities").get()
        self.ui.university_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for fac in temp:
            self.ui.university_id_spinner.addItem(fac["Name"])

    def cellChecked(self, row, column):
        if column == 0:
```

```
            item = self.ui.AdminsTable.currentItem()
            self.ui.admin_id_tf.setText(item.text())
            self.ui.admin_fio_tf.setText(self.ui.AdminsTable.item(row,
column + 1).text())
            self.ui.admin_login_tf.setText(self.ui.AdminsTable.item(row,
column + 2).text())
            self.ui.admin_pass_tf.setText(self.ui.AdminsTable.item(row,
column + 3).text())

            universities = [self.ui.university_id_spinner.itemText(i)
                            for i in
range(self.ui.university_id_spinner.count())]


self.ui.university_id_spinner.setCurrentIndex(universities.index

(self.ui.AdminsTable.item(row, column + 4).text()))

        else:
            self.ui.admin_id_tf.clear()
            self.ui.admin_fio_tf.clear()
            self.ui.admin_login_tf.clear()
            self.ui.admin_pass_tf.clear()
            self.ui.university_id_spinner.setCurrentIndex(0)


    def loadAdminsData(self):
        self.ui.AdminsTable.setRowCount(0)
        res = DB_Helper.getRef().child("Users").get()
        temp = []

        for elem in res.values():
            if elem['Role'] == 'Локальний адміністратор':
                temp.append(elem)

        self.ui.AdminsTable.setRowCount(len(temp))
        row = 0
        temp.sort(key=lambda dictionary: dictionary['Surname'])

        for user in temp:
```

```
            self.ui.AdminsTable.setItem(row, 0,
QtWidgets.QTableWidgetItem(user["Id"]))

            fio = user["Surname"] + " " + user["Name"] + " " +
user["Middlename"]
            fio_item = QtWidgets.QTableWidgetItem(fio)
            fio_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.AdminsTable.setItem(row, 1, fio_item)

            login = user["Login"]
            login_item = QtWidgets.QTableWidgetItem(login)
            login_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.AdminsTable.setItem(row, 2, login_item)

            password = user["Password"]
            password_item = QtWidgets.QTableWidgetItem(password)
            password_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.AdminsTable.setItem(row, 3, password_item)

            university = user["University"]
            university_item = QtWidgets.QTableWidgetItem(university)
            university_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.AdminsTable.setItem(row, 4, university_item)

            row += 1

    def createNewLocalAdmin(self):
        ref = DB_Helper.getRef().child("Users")
        temp = self.ui.admin_fio_tf.text().split(' ')
        name, surname, middlename = temp[1], temp[0], temp[2]

        new_local_admin = ref.push(
            {
                'Name': name,
                'Surname': surname,
                'Middlename': middlename,
                'Login': self.ui.admin_login_tf.text(),
                'Password': self.ui.admin_pass_tf.text(),
                'Role': 'Локальний адміністратор',
                'University': self.ui.university_id_spinner.currentText()
            }
```

```
        )

        key = new_local_admin.key
        item_ref = ref.child(key)

        item_ref.update({'Id': key})
        self.loadAdminsData()

    def updateLocalAdmin(self):
        key = self.ui.admin_id_tf.text()
        ref = DB_Helper.getRef().child("Users").child(key)

        temp = self.ui.admin_fio_tf.text().split(' ')
        name, surname, middlename = temp[1], temp[0], temp[2]

        ref.update({'Name': name})
        ref.update({'Surname': surname})
        ref.update({'Middlename': middlename})
        ref.update({'Login': self.ui.admin_login_tf.text()})
        ref.update({'Password': self.ui.admin_pass_tf.text()})
        ref.update({'University':
self.ui.university_id_spinner.currentText()})
        self.loadAdminsData()

    def deleteLocalAdmin(self):
        key = self.ui.admin_id_tf.text()
        ref = DB_Helper.getRef().child("Users").child(key)
        ref.delete()
        self.loadAdminsData()

    def export(self):
        files = [f for f in os.listdir('.') if os.path.isfile(f)]
        headers_font = Font(name='Times New Roman', size=16, bold=True,
color='ffffff')
        headers_fill = PatternFill(fill_type='solid', fgColor='1c1c1a')
        headers_alignment = Alignment(horizontal='center')

        if "SCA_DATA.xlsx" in files:
            wb = load_workbook('./SCA_DATA.xlsx')

            if "Local_admins" in wb.sheetnames:
```

```
                ws_admins = wb["Local_admins"]
            else:
                ws_admins = wb.create_sheet("Local_admins")

            self.fillHeaders(ws_admins, headers_font, headers_fill,
headers_alignment, wb)

        else:
            wb = Workbook()
            wb.remove(wb['Sheet'])
            ws_departments = wb.create_sheet("Local_admins")
            self.fillHeaders(ws_departments, headers_font, headers_fill,
headers_alignment, wb)

    def fillHeaders(self, ws_departments, headers_font, headers_fill,
headers_alignment, wb):
        ws_departments['A1'] = "ID"
        ws_departments['B1'] = 'ПІП'
        ws_departments['C1'] = 'ЛОГІН'
        ws_departments['D1'] = 'ПАРОЛЬ'
        ws_departments['E1'] = 'УНІВЕРСИТЕТ'

        ws_departments['A1'].font = headers_font
        ws_departments['A1'].fill = headers_fill
        ws_departments['A1'].alignment = headers_alignment
        ws_departments['B1'].font = headers_font
        ws_departments['B1'].fill = headers_fill
        ws_departments['B1'].alignment = headers_alignment
        ws_departments['C1'].font = headers_font
        ws_departments['C1'].fill = headers_fill
        ws_departments['C1'].alignment = headers_alignment
        ws_departments['D1'].font = headers_font
        ws_departments['D1'].fill = headers_fill
        ws_departments['D1'].alignment = headers_alignment
        ws_departments['E1'].font = headers_font
        ws_departments['E1'].fill = headers_fill
        ws_departments['E1'].alignment = headers_alignment
        self.startExport(wb, ws_departments, 'Departments')

    def startExport(self, wb, ws, table):
        res = DB_Helper.getRef().child("Users").get()
```

```
        temp = []
        temp1 = []

        for elem in res.values():
            if elem['Role'] == 'Локальний адміністратор':
                temp.append(elem)

        if ws.max_row == 1:

            for item in temp:
                id = item['Id']
                pip = item["Surname"] + " " + item["Name"] + " " +
item["Middlename"]
                login = item["Login"]
                password = item["Password"]
                university = item['University']
                temp1.append([id, pip, login, password, university])

            for row in temp1:
                ws.append(row)

            paintCells(ws)
            wb.save('SCA_DATA.xlsx')

        else:
            amount = ws.max_row - 1
            ws.delete_rows(2, amount)
            wb.save('SCA_DATA.xlsx')

            for item in temp:
                id = item['Id']
                pip = item["Surname"] + " " + item["Name"] + " " +
item["Middlename"]
                login = item["Login"]
                password = item["Password"]
                university = item['University']
                temp1.append([id, pip, login, password, university])

            for row in temp1:
                ws.append(row)
            paintCells(ws)
```

```
            wb.save('SCA_DATA.xlsx')

    def closeEvent(self, event: PySide6.QtGui.QCloseEvent) -> None:
        self.back_to_ma_mm()

    def back_to_ma_mm(self):
        self.ma_mm_window =
Controllers.MA_Controller.MainAdminMainMenuWindow()
        self.ma_mm_window.show()
        self.close()
```

Лістінг 1 – Код контролеру для роботи з локальними адміністраторами

```
import datetime
import os

import PySide6
from PySide6 import QtWidgets, QtCore
from PySide6.QtCore import QTimer
from PySide6.QtGui import Qt
from PySide6.QtWidgets import QMainWindow
from openpyxl import load_workbook, Workbook
from openpyxl.styles import Font, PatternFill, Alignment

import DB_Helper
from PaintHelper import paintCells
from UI.la_teachers_menu import LA_Teachers_Menu_UI
import Controllers.LA_Controller


def getEntityByName(search_value, table, search_field):
    ref = DB_Helper.getRef().child(table).get()

    for elem in ref.values():
        if elem[search_field] == search_value:
            return elem
    return None


class LocalAdminTeachersMenuWindow(QMainWindow):
    def __init__(self, university):
```

```
        super().__init__()
        self.la_mm_window = None
        self.university = university
        self.table_headers = ["ID", "ПІП", "Логін", "Пароль", "Кафедра"]

        self.ui = LA_Teachers_Menu_UI()
        self.ui.setupUi(self)

        self.timer = QTimer(self)
        self.timer.timeout.connect(self.setTime)
        self.timer.start(1000)


self.ui.TeachersTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwa
ysOn)
        self.ui.TeachersTable.setColumnCount(5)

self.ui.TeachersTable.setHorizontalHeaderLabels(self.table_headers)
        self.ui.TeachersTable.setColumnWidth(1, 180)
        self.ui.TeachersTable.setColumnWidth(2, 150)
        self.ui.TeachersTable.setColumnWidth(3, 150)
        self.ui.TeachersTable.setColumnWidth(4, 190)
        self.ui.TeachersTable.horizontalHeader().setStyleSheet(
            "QHeaderView { "
            "font-size: 16pt;"
            "font: 700 18pt \"Sitka Small\";}"
        )
        self.ui.teacher_id_tf.setReadOnly(True)

        self.ui.TeachersTable.cellClicked.connect(self.cellChecked)
        self.ui.CreateTeacherBtn.clicked.connect(self.createNewTeacher)
        self.ui.UpdateTeacherBtn.clicked.connect(self.updateTeacher)
        self.ui.DeleteTeacherBtn.clicked.connect(self.deleteTeacher)
        self.ui.ExportTeacherBtn.clicked.connect(self.export)

        self.fillFacultiesSpinner()
        self.fillDepartmentsSpinner()
        self.loadTeachersData()

    def setTime(self):
        now = datetime.datetime.now()
```

```
        formatDate = now.strftime("%d.%m.%Y, %H:%M:%S")
        self.ui.DateAndTimeLabel.setText(formatDate)

    def cellChecked(self, row, column):

        if column == 0:
            item = self.ui.TeachersTable.currentItem()
            self.ui.teacher_id_tf.setText(item.text())
            self.ui.teacher_FIO_tf.setText(self.ui.TeachersTable.item(row,
column + 1).text())

self.ui.teacher_login_tf.setText(self.ui.TeachersTable.item(row, column +
2).text())

self.ui.teacher_password_tf.setText(self.ui.TeachersTable.item(row, column
+ 3).text())

            departments = [self.ui.department_id_spinner.itemText(i)
                           for i in
range(self.ui.department_id_spinner.count())]


self.ui.department_id_spinner.setCurrentIndex(departments.index

(self.ui.TeachersTable.item(row, column + 4).text()))

            department_name = self.ui.department_id_spinner.currentText()
            department = getEntityByName(department_name, "Departments",
"Name")
            faculty_name = department['Faculty']
            faculties = [self.ui.faculty_id_spinner.itemText(i)
                         for i in
range(self.ui.faculty_id_spinner.count())]

self.ui.faculty_id_spinner.setCurrentIndex(faculties.index(faculty_name))

        else:
            self.ui.teacher_id_tf.clear()
            self.ui.teacher_FIO_tf.clear()
            self.ui.teacher_login_tf.clear()
            self.ui.teacher_password_tf.clear()
```

```
        self.ui.department_id_spinner.setCurrentIndex(0)

    def fillDepartmentsSpinner(self):
        res = DB_Helper.getRef().child("Departments").get()

        faculties = [self.ui.faculty_id_spinner.itemText(i)
                     for i in range(self.ui.faculty_id_spinner.count())]

        self.ui.department_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            if item['Faculty'] in faculties:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for department in temp:
            self.ui.department_id_spinner.addItem(department["Name"])

    def fillFacultiesSpinner(self):
        res = DB_Helper.getRef().child("Faculties").get()
        self.ui.faculty_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            if item['University'] == self.university:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for faculty in temp:
            self.ui.faculty_id_spinner.addItem(faculty["Name"])

    def loadTeachersData(self):
        self.ui.TeachersTable.setRowCount(0)
        res = DB_Helper.getRef().child("Users").get()
        self.ui.TeachersTable.setRowCount(len(res))
        temp = []

        row = 0
```

```
        for item in res.values():
            if 'University' in item.keys():
                if item['University'] == self.university and item['Role']
== 'Викладач':
                    temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Surname'])

        for teacher in temp:
            self.ui.TeachersTable.setItem(row, 0,
QtWidgets.QTableWidgetItem(teacher["Id"]))

            fio = teacher['Surname'] + ' ' + teacher['Name'] + ' ' +
teacher['Middlename']
            fio_item = QtWidgets.QTableWidgetItem(fio)
            fio_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.TeachersTable.setItem(row, 1, fio_item)

            login = teacher['Login']
            login_item = QtWidgets.QTableWidgetItem(login)
            login_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.TeachersTable.setItem(row, 2, login_item)

            password = teacher['Password']
            password_item = QtWidgets.QTableWidgetItem(password)
            password_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.TeachersTable.setItem(row, 3, password_item)

            department_item =
QtWidgets.QTableWidgetItem(teacher["Department"])
            department_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.TeachersTable.setItem(row, 4, department_item)
            row += 1

    def createNewTeacher(self):
        ref = DB_Helper.getRef().child("Users")
        temp = self.ui.teacher_FIO_tf.text().split(' ')

        name = temp[1]
        surname = temp[0]
```

```
        middlename = temp[2]

        department_name = self.ui.department_id_spinner.currentText()
        department = getEntityByName(department_name, "Departments",
"Name")
        faculty_name = department['Faculty']
        faculty = getEntityByName(faculty_name, "Faculties", "Name")
        university_name = faculty['University']

        new_teacher = ref.push(
            {
                'Name': name,
                'Surname': surname,
                'Middlename': middlename,
                'Login': self.ui.teacher_login_tf.text(),
                'Password': self.ui.teacher_login_tf.text(),
                'Role': 'Викладач',
                'Department': department_name,
                'Faculty': faculty_name,
                'University': university_name
            }
        )

        key = new_teacher.key
        item_ref = ref.child(key)

        item_ref.update({'Id': key})
        self.loadTeachersData()

    def updateTeacher(self):
        key = self.ui.teacher_id_tf.text()
        temp = self.ui.teacher_FIO_tf.text().split(' ')

        department_name = self.ui.department_id_spinner.currentText()
        department = getEntityByName(department_name, "Departments",
"Name")
        faculty_name = department['Faculty']
        faculty = getEntityByName(faculty_name, "Faculties", "Name")
        university_name = faculty['University']

        name = temp[1]
```

```
        surname = temp[0]
        middlename = temp[2]

        ref = DB_Helper.getRef().child("Users").child(key)
        ref.update({'Name': name})
        ref.update({'Surname': surname})
        ref.update({'Middlename': middlename})
        ref.update({'Login': self.ui.teacher_login_tf.text()})
        ref.update({'Password': self.ui.teacher_password_tf.text()})
        ref.update({'Department': department_name})
        ref.update({'Faculty': faculty_name})
        ref.update({'University': university_name})

        self.loadTeachersData()

    def deleteTeacher(self):
        key = self.ui.teacher_id_tf.text()
        ref = DB_Helper.getRef().child("Users").child(key)
        ref.delete()
        self.loadTeachersData()

    def export(self):
        files = [f for f in os.listdir('.') if os.path.isfile(f)]
        headers_font = Font(name='Times New Roman', size=16, bold=True,
color='ffffff')
        headers_fill = PatternFill(fill_type='solid', fgColor='1c1c1a')
        headers_alignment = Alignment(horizontal='center')

        if "SCA_DATA.xlsx" in files:
            wb = load_workbook('./SCA_DATA.xlsx')

            if "Teachers" in wb.sheetnames:
                ws_teachers = wb["Teachers"]
            else:
                ws_teachers = wb.create_sheet("Teachers")

            self.fillHeaders(ws_teachers, headers_font, headers_fill,
headers_alignment, wb)

        else:
            wb = Workbook()
```

```
            wb.remove(wb['Sheet'])
            ws_teachers = wb.create_sheet("Teachers")
            self.fillHeaders(ws_teachers, headers_font, headers_fill,
headers_alignment, wb)

    def fillHeaders(self, ws_teachers, headers_font, headers_fill,
headers_alignment, wb):
        ws_teachers['A1'] = "ID"
        ws_teachers['B1'] = 'ПІП'
        ws_teachers['C1'] = 'ЛОГІН'
        ws_teachers['D1'] = 'ПАРОЛЬ'
        ws_teachers['E1'] = 'КАФЕДРА'
        ws_teachers['A1'].font = headers_font
        ws_teachers['A1'].fill = headers_fill
        ws_teachers['A1'].alignment = headers_alignment
        ws_teachers['B1'].font = headers_font
        ws_teachers['B1'].fill = headers_fill
        ws_teachers['B1'].alignment = headers_alignment
        ws_teachers['C1'].font = headers_font
        ws_teachers['C1'].fill = headers_fill
        ws_teachers['C1'].alignment = headers_alignment
        ws_teachers['D1'].font = headers_font
        ws_teachers['D1'].fill = headers_fill
        ws_teachers['D1'].alignment = headers_alignment
        ws_teachers['E1'].font = headers_font
        ws_teachers['E1'].fill = headers_fill
        ws_teachers['E1'].alignment = headers_alignment
        self.startExport(wb, ws_teachers, 'Users')

    def startExport(self, wb, ws, table):
        res = DB_Helper.getRef().child(table).get()
        temp = []

        if ws.max_row == 1:
            for item in res.values():
                if item['Role'] == 'Викладач':
                    id = item['Id']
                    pip = item["Surname"] + " " + item["Name"] + " " +
item["Middlename"]
                    login = item["Login"]
                    password = item["Password"]
```

```
                    department = item['Department']
                    temp.append([id, pip, login, password, department])

            for row in temp:
                ws.append(row)

            paintCells(ws)
            wb.save('SCA_DATA.xlsx')

        else:
            amount = ws.max_row - 1
            ws.delete_rows(2, amount)
            wb.save('SCA_DATA.xlsx')

            for item in res.values():
                if item['Role'] == 'Викладач':
                    id = item['Id']
                    pip = item["Surname"] + " " + item["Name"] + " " +
item["Middlename"]
                    login = item["Login"]
                    password = item["Password"]
                    department = item['Department']
                    temp.append([id, pip, login, password, department])

            for row in temp:
                ws.append(row)
            paintCells(ws)
            wb.save('SCA_DATA.xlsx')

    def closeEvent(self, event: PySide6.QtGui.QCloseEvent) -> None:
        self.back_to_la_mm()

    def back_to_la_mm(self):
        self.la_mm_window =
Controllers.LA_Controller.LocalAdminMainMenuWindow(self.university)
        self.la_mm_window.show()
        self.close()
```

Лістінг 2 – Код контролеру для роботи із викладачами

```
import datetime
```

```
import PySide6
from PySide6 import QtCore, QtWidgets
from PySide6.QtCore import QTimer
from PySide6.QtGui import Qt
from PySide6.QtWidgets import QMainWindow

import Controllers.AuthController

import DB_Helper
import UI.teacher_mm


def getEntityByName(search_value, table, search_field):
    ref = DB_Helper.getRef().child(table).get()

    for elem in ref.values():
        if elem[search_field] == search_value:
            return elem
    return None


def getStudentByFIOAndGroup(surname, name, middlename, group):
    ref = DB_Helper.getRef().child('Students').get()

    for elem in ref.values():
        if elem['Group'] == group and elem['Surname'] == surname and
elem['Name'] == name\
                and elem['Middlename'] == middlename:
            return elem
    return None


class TeacherMainMenuWindow(QMainWindow):
    def __init__(self, university, fio, id):
        super().__init__()
        self.university = university
        self.id = id
        self.fio = fio
        self.auth_menu_window = None
        self.ui = UI.teacher_mm.Teacher_Menu_UI()
```

```
        self.ui.setupUi(self)

        self.table_headers = ["Id", "Студент", "Дисципліна", "Тип", "КЗ",
"ДЗ", "Дата/Час"]

        self.timer = QTimer(self)
        self.timer.timeout.connect(self.setTime)
        self.timer.start(1000)

        self.setShortFIO()

        self.ui.mark_id_tf.setReadOnly(True)


self.ui.MarksTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysO
n)
        self.ui.MarksTable.setColumnCount(7)
        self.ui.MarksTable.setHorizontalHeaderLabels(self.table_headers)
        self.ui.MarksTable.setColumnWidth(0, 300)
        self.ui.MarksTable.setColumnWidth(1, 300)
        self.ui.MarksTable.setColumnWidth(2, 300)
        self.ui.MarksTable.setColumnWidth(3, 300)
        self.ui.MarksTable.setColumnWidth(4, 300)
        self.ui.MarksTable.setColumnWidth(5, 300)
        self.ui.MarksTable.setColumnWidth(6, 300)
        self.ui.MarksTable.horizontalHeader().setStyleSheet(
            "QHeaderView { "
            "font-size: 16pt;"
            "font: 700 18pt \"Sitka Small\";}"
        )

        self.ui.group_id_spinner.setStyleSheet(
            "color: white;"
            "font: 700 16pt \"Sitka Small\";"
        )

        self.ui.student_id_spinner.setStyleSheet(
            "color: white;"
            "font: 700 16pt \"Sitka Small\";"
        )
```

```
        self.ui.discipline_id_spinner.setStyleSheet(
            "color: white;"
            "font: 700 16pt \"Sitka Small\";"
        )

        self.ui.type_spinner.setStyleSheet(
            "color: white;"
            "font: 700 16pt \"Sitka Small\";"
        )

        self.fillGroupsSpinner()
        self.fillStudentsSpinner()
        self.fillDisciplinesSpinner()
        self.fillTypesSpinner()

        self.loadMarksData()

        self.ui.MarksTable.cellClicked.connect(self.cellChecked)
        self.ui.CreateMarkBtn.clicked.connect(self.createNewMark)
        self.ui.UpdateMarkBtn.clicked.connect(self.updateMark)
        self.ui.DeleteMarkBtn.clicked.connect(self.deleteMark)
        #self.ui.ExportStudentsBtn.clicked.connect(self.export)



        '''
        self.ui.GroupsMenuBtn.clicked.connect(self.go_to_groups_menu)
        self.ui.StudentsMenuBtn.clicked.connect(self.go_to_students_menu)
        self.ui.TeachersMenuBtn.clicked.connect(self.go_to_teachers_menu)

self.ui.DisciplinesMenuBtn.clicked.connect(self.go_to_disciplines_menu)

self.ui.CompetencesMenuBtn.clicked.connect(self.go_to_competences_menu)
        self.ui.AnalyzeMenuBtn.clicked.connect(self.go_to_analyze_menu)'''

    def setTime(self):
        now = datetime.datetime.now()
        formatDate = now.strftime("%d.%m.%Y, %H:%M:%S")
        self.ui.DateAndTimeLabel.setText(formatDate)

    def setShortFIO(self):
```

```
        temp = self.fio.split(' ')
        shorted = temp[0] + ' ' + temp[1][0] + '. ' + temp[2][0] + '.'
        self.ui.TeacherNameLabel.setText(shorted)

    def fillGroupsSpinner(self):
        res = DB_Helper.getRef().child("Groups").get()
        self.ui.group_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            if item['University'] == self.university:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Number'])

        for faculty in temp:
            self.ui.group_id_spinner.addItem(faculty["Number"])

    def fillStudentsSpinner(self):
        res = DB_Helper.getRef().child("Students").get()

        groups = [self.ui.group_id_spinner.itemText(i)
                  for i in range(self.ui.group_id_spinner.count())]

        self.ui.student_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            if item['Group'] in groups:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Surname'])

        for student in temp:
            fio = student["Surname"] + " " + student["Name"] + " " +
student["Middlename"]
            # item = fio + " : " + student["Id"]
            self.ui.student_id_spinner.addItem(fio)

    def fillDisciplinesSpinner(self):
        res = DB_Helper.getRef().child("Disciplines").get()
```

```
        temp = []

        for item in res.values():
            if item['University'] == self.university and self.id in
item['TeachersIds']:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for discipline in temp:
            self.ui.discipline_id_spinner.addItem(discipline["Name"])

    def fillTypesSpinner(self):
        types = ["Лекція", "Лабораторна робота", "Практична робота",
"Домашнє завдання",
                 "Курсова робота", "Залік", "Екзамен", "Модульна
контрольна робота"]

        for item in types:
            self.ui.type_spinner.addItem(item)

    def loadMarksData(self):
        self.ui.MarksTable.setRowCount(0)
        res = DB_Helper.getRef().child("Marks").get()
        self.ui.MarksTable.setRowCount(len(res))
        temp = []

        row = 0

        for item in res.values():
            if item['University'] == self.university and
item['Teacher_Id'] == self.id:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Date'])

        # "Id", "Студент", "Дисципліна", "Тип", "КЗ", "ДЗ", "Дата/Час"
        for mark in temp:
            self.ui.MarksTable.setItem(row, 0,
QtWidgets.QTableWidgetItem(mark["Id"]))
```

```
            # fio = student['Surname'] + ' ' + student['Name'] + ' ' +
student['Middlename']

            student_item = QtWidgets.QTableWidgetItem(mark['Student_Id'])
            student_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.MarksTable.setItem(row, 1, student_item)

            discipline_item =
QtWidgets.QTableWidgetItem(mark['Discipline'])
            discipline_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.MarksTable.setItem(row, 2, discipline_item)

            type_item = QtWidgets.QTableWidgetItem(mark['Type'])
            type_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.MarksTable.setItem(row, 3, type_item)

            short_value_item =
QtWidgets.QTableWidgetItem(str(mark['Short_value']))

short_value_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.MarksTable.setItem(row, 4, short_value_item)

            long_value_item =
QtWidgets.QTableWidgetItem(str(mark['Long_value']))
            long_value_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.MarksTable.setItem(row, 5, long_value_item)

            date_item = QtWidgets.QTableWidgetItem(str(mark['Date']))
            date_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.MarksTable.setItem(row, 6, date_item)

            row += 1

    def cellChecked(self, row, column):

        if column == 0:
            item = self.ui.MarksTable.currentItem()
            self.ui.mark_id_tf.setText(item.text())

            groups = [self.ui.group_id_spinner.itemText(i)
                      for i in range(self.ui.group_id_spinner.count())]
```

```
            students = [self.ui.student_id_spinner.itemText(i)
                        for i in
range(self.ui.student_id_spinner.count())]

            disciplines = [self.ui.discipline_id_spinner.itemText(i)
                           for i in
range(self.ui.discipline_id_spinner.count())]

            types = [self.ui.type_spinner.itemText(i)
                     for i in range(self.ui.type_spinner.count())]

            student = getEntityByName(self.ui.MarksTable.item(row, column
+ 1).text(), 'Students', 'Id')
            student_fio = student['Surname'] + ' ' + student['Name'] + ' '
+ student['Middlename']


self.ui.student_id_spinner.setCurrentIndex(students.index(student_fio))

self.ui.group_id_spinner.setCurrentIndex(groups.index(student['Group']))

self.ui.discipline_id_spinner.setCurrentIndex(disciplines.index

(self.ui.MarksTable.item(row, column + 2).text()))


self.ui.type_spinner.setCurrentIndex(types.index(self.ui.MarksTable.item(r
ow, column + 3).text()))

            self.ui.sv_tf.setText(self.ui.MarksTable.item(row, column +
4).text())
            self.ui.lv_tf.setText(self.ui.MarksTable.item(row, column +
5).text())

        else:
            self.ui.mark_id_tf.clear()
            self.ui.group_id_spinner.setCurrentIndex(0)
            self.ui.student_id_spinner.setCurrentIndex(0)
            self.ui.discipline_id_spinner.setCurrentIndex(0)
            self.ui.type_spinner.setCurrentIndex(0)
```

```
            self.ui.sv_tf.clear()
            self.ui.lv_tf.clear()

    def createNewMark(self):
        ref = DB_Helper.getRef().child("Marks")

        student_fio = self.ui.student_id_spinner.currentText().split(' ')

        surname = student_fio[0]
        name = student_fio[1]
        middlename = student_fio[2]

        formatDate = datetime.date.today().strftime("%d.%m.%Y")

        group = self.ui.group_id_spinner.currentText()

        student = getStudentByFIOAndGroup(surname, name, middlename,
group)

        new_mark = ref.push(
            {
                'Student_Id': student['Id'],
                'Teacher_Id': self.id,
                'Date': formatDate,
                'University': self.university,
                'Type': self.ui.type_spinner.currentText(),
                'Discipline': self.ui.discipline_id_spinner.currentText(),
                'Short_value': int(self.ui.sv_tf.text()),
                'Long_value': int(self.ui.lv_tf.text()),
            }
        )

        key = new_mark.key
        item_ref = ref.child(key)

        item_ref.update({'Id': key})
        self.loadMarksData()

    def updateMark(self):
        key = self.ui.mark_id_tf.text()
```

```
        student_fio = self.ui.student_id_spinner.currentText().split(' ')

        surname = student_fio[0]
        name = student_fio[1]
        middlename = student_fio[2]

        formatDate = datetime.date.today().strftime("%d.%m.%Y")

        group = self.ui.group_id_spinner.currentText()

        student = getStudentByFIOAndGroup(surname, name, middlename,
group)

        ref = DB_Helper.getRef().child("Marks").child(key)
        ref.update({'Student_Id': student['Id']})
        ref.update({'Teacher_Id': self.id})
        ref.update({'Date': formatDate})
        ref.update({'University': self.university})
        ref.update({'Type': self.ui.type_spinner.currentText()})
        ref.update({'Discipline':
self.ui.discipline_id_spinner.currentText()})
        ref.update({'Short_value': int(self.ui.sv_tf.text())})
        ref.update({'Long_value': int(self.ui.lv_tf.text())})
        self.loadMarksData()

    def deleteMark(self):
        key = self.ui.mark_id_tf.text()
        ref = DB_Helper.getRef().child("Marks").child(key)
        ref.delete()
        self.loadMarksData()

    def closeEvent(self, event: PySide6.QtGui.QCloseEvent) -> None:
        self.back_to_auth()

    def back_to_auth(self):
        self.auth_menu_window =
Controllers.AuthController.AuthMenuWindow()
        self.auth_menu_window.show()
        self.close()
```

Лістінг 3 – Код контролеру для роботи із успішністю

```
import datetime
import os

import PySide6
from PySide6 import QtCore, QtWidgets
from PySide6.QtCore import QTimer
from PySide6.QtGui import Qt
from PySide6.QtWidgets import QMainWindow
from openpyxl import load_workbook, Workbook
from openpyxl.styles import Font, PatternFill, Alignment

import Controllers.LA_Controller
import DB_Helper
from PaintHelper import paintCells
from UI.la_competences_menu import LA_Competences_Menu_UI

def getEntityByName(search_value, table, search_field):
    ref = DB_Helper.getRef().child(table).get()

    for elem in ref.values():
        if elem[search_field] == search_value:
            return elem
    return None

class LocalAdminCompetencesMenuWindow(QMainWindow):
    def __init__(self, university):
        super().__init__()
        self.la_mm_window = None
        self.university = university
        self.table_headers = ["ID", "Назва", "Кафедра", "Дисципліни"]

        self.ui = LA_Competences_Menu_UI()
        self.ui.setupUi(self)

        self.timer = QTimer(self)
        self.timer.timeout.connect(self.setTime)
        self.timer.start(1000)
```

```
self.ui.CompetencesTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarA
lwaysOn)
        self.ui.CompetencesTable.setColumnCount(4)

self.ui.CompetencesTable.setHorizontalHeaderLabels(self.table_headers)
        self.ui.CompetencesTable.setColumnWidth(1, 180)
        self.ui.CompetencesTable.setColumnWidth(2, 180)
        self.ui.CompetencesTable.setColumnWidth(3, 240)
        self.ui.CompetencesTable.horizontalHeader().setStyleSheet(
            "QHeaderView { "
            "font-size: 16pt;"
            "font: 700 18pt \"Sitka Small\";}"
        )

        self.ui.comp_disciplines_ids_list.setStyleSheet(
            "color: white;"
            "font: 700 14pt \"Sitka Small\";"
        )

        self.ui.comp_id_tf.setReadOnly(True)

        self.ui.CompetencesTable.cellClicked.connect(self.cellChecked)
        self.ui.CreateCompBtn.clicked.connect(self.createNewCompetence)
        self.ui.UpdateCompBtn.clicked.connect(self.updateCompetence)
        self.ui.DeleteCompBtn.clicked.connect(self.deleteCompetence)
        self.ui.ExportCompBtn.clicked.connect(self.export)

        self.fillDisciplinesList()
        self.fillFacultiesSpinner()
        self.fillDepartmentsSpinner()
        self.loadCompetencesData()

    def setTime(self):
        now = datetime.datetime.now()
        formatDate = now.strftime("%d.%m.%Y, %H:%M:%S")
        self.ui.DateAndTimeLabel.setText(formatDate)

    def fillDepartmentsSpinner(self):
        res = DB_Helper.getRef().child("Departments").get()
```

```
        faculties = [self.ui.faculty_id_spinner.itemText(i)
                     for i in range(self.ui.faculty_id_spinner.count())]

        self.ui.department_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            if item['Faculty'] in faculties:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for department in temp:
            self.ui.department_id_spinner.addItem(department["Name"])

    def fillFacultiesSpinner(self):
        res = DB_Helper.getRef().child("Faculties").get()
        self.ui.faculty_id_spinner.addItem('Не обрано')
        temp = []

        for item in res.values():
            if item['University'] == self.university:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for faculty in temp:
            self.ui.faculty_id_spinner.addItem(faculty["Name"])

    def cellChecked(self, row, column):

        if column == 0:
            item = self.ui.CompetencesTable.currentItem()
            self.ui.comp_id_tf.setText(item.text())

self.ui.comp_name_tf.setText(self.ui.CompetencesTable.item(row, column +
1).text())

            departments = [self.ui.department_id_spinner.itemText(i)
                           for i in
range(self.ui.department_id_spinner.count())]
```

```
self.ui.department_id_spinner.setCurrentIndex(departments.index

(self.ui.CompetencesTable.item(row, column + 2).text()))

            department_name = self.ui.department_id_spinner.currentText()
            department = getEntityByName(department_name, "Departments",
"Name")
            faculty_name = department['Faculty']
            faculties = [self.ui.faculty_id_spinner.itemText(i)
                         for i in
range(self.ui.faculty_id_spinner.count())]

self.ui.faculty_id_spinner.setCurrentIndex(faculties.index(faculty_name))

            teachers_ids = self.ui.CompetencesTable.item(row, column +
3).text()
            item_teachers = self.transformData(teachers_ids)

            teachers = [self.ui.comp_disciplines_ids_list.item(i)
                        for i in
range(self.ui.comp_disciplines_ids_list.count())]

            self.ui.comp_disciplines_ids_list.clearSelection()
            for item in teachers:
                id = item.text().split(' : ')[1]

                if id in item_teachers:
                    item.setSelected(True)
                else:
                    item.setSelected(False)

        else:
            self.ui.comp_id_tf.clear()
            self.ui.comp_name_tf.clear()
            self.ui.department_id_spinner.setCurrentIndex(0)
            self.ui.faculty_id_spinner.setCurrentIndex(0)
            self.ui.comp_disciplines_ids_list.clearSelection()

    def transformData(self, teachers_ids):
```

```
        temp = teachers_ids.split(',')
        res = [elem.lstrip() for elem in temp]
        return res

    def clearSelection(self):
        self.ui.comp_disciplines_ids_list.clearSelection()

    def fillDisciplinesList(self):
        res = DB_Helper.getRef().child("Disciplines").get()
        disciplines = []

        for elem in res.values():
            if elem['University'] == self.university:
                disciplines.append(elem)

        disciplines.sort(key=lambda dictionary: dictionary['Name'])

        for disc in disciplines:
            name = disc['Name']
            temp = name + " : " + disc['Id']
            self.ui.comp_disciplines_ids_list.addItem(temp)

    def loadCompetencesData(self):
        self.ui.CompetencesTable.setRowCount(0)
        res = DB_Helper.getRef().child("Competences").get()
        self.ui.CompetencesTable.setRowCount(len(res))
        temp = []

        row = 0

        for item in res.values():
            if item['University'] == self.university:
                temp.append(item)

        temp.sort(key=lambda dictionary: dictionary['Name'])

        for disc in temp:
            self.ui.CompetencesTable.setItem(row, 0,
QtWidgets.QTableWidgetItem(disc["Id"]))

            name_item = QtWidgets.QTableWidgetItem(disc['Name'])
```

```
            name_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.CompetencesTable.setItem(row, 1, name_item)

            department_item =
QtWidgets.QTableWidgetItem(disc["Department"])
            department_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.CompetencesTable.setItem(row, 2, department_item)

            disciplines_item =
QtWidgets.QTableWidgetItem(disc["Goal_Disciplines_Ids"])

disciplines_item.setTextAlignment(Qt.AlignmentFlag.AlignCenter)
            self.ui.CompetencesTable.setItem(row, 3, disciplines_item)

            row += 1

    def createNewCompetence(self):
        ref = DB_Helper.getRef().child("Competences")

        department_name = self.ui.department_id_spinner.currentText()
        department = getEntityByName(department_name, "Departments",
"Name")
        faculty_name = department['Faculty']
        faculty = getEntityByName(faculty_name, "Faculties", "Name")
        university_name = faculty['University']

        selected_disciplines =
self.ui.comp_disciplines_ids_list.selectedItems()
        selected_ids = [elem.text().split(' : ')[1] for elem in
selected_disciplines]
        ids = ', '.join(selected_ids)

        new_competence = ref.push(
            {
                'Name': self.ui.comp_name_tf.text(),
                'Department': department_name,
                'Faculty': faculty_name,
                'University': university_name,
                'Goal_Disciplines_Ids': ids
            }
        )
```

```
        key = new_competence.key
        item_ref = ref.child(key)

        item_ref.update({'Id': key})
        self.loadCompetencesData()

    def updateCompetence(self):
        key = self.ui.comp_id_tf.text()

        department_name = self.ui.department_id_spinner.currentText()
        department = getEntityByName(department_name, "Departments",
"Name")
        faculty_name = department['Faculty']
        faculty = getEntityByName(faculty_name, "Faculties", "Name")
        university_name = faculty['University']

        selected_disciplines =
self.ui.comp_disciplines_ids_list.selectedItems()
        selected_ids = [elem.text().split(' : ')[1] for elem in
selected_disciplines]
        ids = ', '.join(selected_ids)

        ref = DB_Helper.getRef().child("Competences").child(key)
        ref.update({'Name': self.ui.comp_name_tf.text()})
        ref.update({'Department':
self.ui.department_id_spinner.currentText()})
        ref.update({'Faculty': faculty_name})
        ref.update({'University': university_name})
        ref.update({'Goal_Disciplines_Ids': ids})
        self.loadCompetencesData()

    def deleteCompetence(self):
        key = self.ui.comp_id_tf.text()
        ref = DB_Helper.getRef().child("Competences").child(key)
        ref.delete()
        self.loadCompetencesData()

    def export(self):
        files = [f for f in os.listdir('.') if os.path.isfile(f)]
```

```
        headers_font = Font(name='Times New Roman', size=16, bold=True,
color='ffffff')
        headers_fill = PatternFill(fill_type='solid', fgColor='1c1c1a')
        headers_alignment = Alignment(horizontal='center')

        if "SCA_DATA.xlsx" in files:
            wb = load_workbook('./SCA_DATA.xlsx')

            if "Competences" in wb.sheetnames:
                ws_competences = wb["Competences"]
            else:
                ws_competences = wb.create_sheet("Competences")

            self.fillHeaders(ws_competences, headers_font, headers_fill,
headers_alignment, wb)

        else:
            wb = Workbook()
            wb.remove(wb['Sheet'])
            ws_competences = wb.create_sheet("Competences")
            self.fillHeaders(ws_competences, headers_font, headers_fill,
headers_alignment, wb)

    def fillHeaders(self, ws_competences, headers_font, headers_fill,
headers_alignment, wb):
        ws_competences['A1'] = "ID"
        ws_competences['B1'] = 'НАЗВА'
        ws_competences['C1'] = 'КАФЕДРА'
        ws_competences['D1'] = 'ДИСЦИПЛІНИ'
        ws_competences['A1'].font = headers_font
        ws_competences['A1'].fill = headers_fill
        ws_competences['A1'].alignment = headers_alignment
        ws_competences['B1'].font = headers_font
        ws_competences['B1'].fill = headers_fill
        ws_competences['B1'].alignment = headers_alignment
        ws_competences['C1'].font = headers_font
        ws_competences['C1'].fill = headers_fill
        ws_competences['C1'].alignment = headers_alignment
        ws_competences['D1'].font = headers_font
        ws_competences['D1'].fill = headers_fill
        ws_competences['D1'].alignment = headers_alignment
```

```
    self.startExport(wb, ws_competences, 'Competences')

def startExport(self, wb, ws, table):
    res = DB_Helper.getRef().child(table).get()
    temp = []

    if ws.max_row == 1:
        for item in res.values():
            id = item['Id']
            name = item['Name']
            department = item['Department']
            disciplines = item['Goal_Disciplines_Ids']
            temp.append([id, name, department, disciplines])

        for row in temp:
            ws.append(row)

        paintCells(ws)
        wb.save('SCA_DATA.xlsx')

    else:
        amount = ws.max_row - 1
        ws.delete_rows(2, amount)
        wb.save('SCA_DATA.xlsx')

        for item in res.values():
            id = item['Id']
            name = item['Name']
            department = item['Department']
            disciplines = item['Goal_Disciplines_Ids']
            temp.append([id, name, department, disciplines])

        for row in temp:
            ws.append(row)
        paintCells(ws)
        wb.save('SCA_DATA.xlsx')

def closeEvent(self, event: PySide6.QtGui.QCloseEvent) -> None:
    self.back_to_ma_mm()
```

Лістінг 4 – Код контролеру для роботи із компетентностями