Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ

завідувач кафедри

мережевих та інтернет технологій

_____ Юрій КРАВЧЕНКО

«___»_____2023 року

КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»

за спеціальністю 172 «Телекомунікації та радіотехніка»

освітньо-професійна програма «Мережеві та інтернет технології»

на тему:

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ НАРАХУВАННЯ СТИПЕНДІЇ СТУДЕНТІВ ВНЗ

Виконав: студент групи МІТ-41

Данило БАЙДА

Керівник: Доцент кафедри мережевих та інтернет технологій

д.т.н., доцент Андрій ДУДНІК

Київ 2023

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Юрій КРАВЧЕНКО

«_____»_____2023 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Байді Данилі Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи:
Розробка інформаційної системи обліку нарахування стипендії студентів ВНЗ

затверджена на засіданні кафедри МІТ «07» грудня 2022 р. протокол №5
2. Термін здачі закінченої роботи «31» травня 2023 р.
3. Вихідні дані до проекту
(роботи)

Сучасні технології створення додатків

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)

Вступ

1. Дослідження методів та алгоритмів побудови інформаційних систем. Постановка задачі

2. Визначення шляхів та методів вирішення задачі

3. Розробка додатку

5. Перелік графічного матеріалу 8-10 слайдів

Дата видачі завдання

Керівник роботи д.т.н., доцент Андрій ДУДНІК
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання
Д.А.Байда
(підпис)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	20.01.2023	
2	Розділ 1	15.02.2023	
3	Розділ 2	15.03.2023	
4	Розділ 3	20.04.2023	
5	Доповідь та слайди	25.05.2023	
6	Пояснювальна записка	31.05.2023	

Здобувач вищої освіти _____ Данило БАЙДА
(підпис)

Керівник _____ Андрій ДУДНІК
(підпис)

РЕФЕРАТ

Пояснювальна записка: 70 сторінок, 37 рисунків, 7 додатків, 9 джерел

Об'єкт дослідження: інформаційні системи та бази даних.

Мета роботи(проекту): створення інформаційної системи обліку нарахування стипендії студентам ВНЗ.

Методи дослідження: робота з Oracle DB та Oracle Application Express, методи аутентифікації та адміністрування.

Короткий зміст роботи: у зв'язку з масовою пандемією та війною велика кількість галузей перейшла у онлайн формат. Через брак зручних інструментів для синхронізованої роботи з певними видами даних та документів, буде доцільно розробити додаток, що дозволить наприклад відділенням ВНЗ більш ефективно зберігати та оброблювати інформацію.

По ходу роботи було розроблено додаток, із зручними інструментами роботи та обробки інформації.

Практичне значення роботи полягає у тому, що створений додаток допоможе наглядно продемонструвати приклад використання технологій Oracle для створення додатків з зручною навігацією, широким функціоналом, та інтерактивною взаємодією з користувачем. Її результати дозволять покращити якість та ефективність створених додатків на основі Oracle, а також забезпечити кращий користувацький досвід. Результати дослідження можуть бути використані при розробці та оптимізації різних додатків у майбутньому.

Результати здійснених у дипломному проекті досліджень можуть бути використані як приклад створення додатків із зручним інтерфейсом для використання у діяльності бухгалтерій та інших відділів ВНЗ.

Ключові слова: інформаційні системи, бази даних, СУБД, діаграма сутність-зв'язок.

ABSTRACT

Explanatory Note: 70 pages, 37 figures, 7 appendices, 9 references.

Research Object: Information systems and databases.

Objective of the work (project): Creation of an information system for accounting the awarding of scholarships to university students.

Research Methods: Work with Oracle DB and Oracle Application Express, authentication and administration methods.

Brief summary of the work: Due to the widespread pandemic and war, a large number of industries have transitioned to an online format. Due to the lack of convenient tools for synchronized work with certain types of data and documents, it would be advisable to develop an application that will allow university departments, for example, to store and process information more efficiently.

During the course of the work, an application was developed with convenient tools for working and processing information.

The practical significance of the work lies in the fact that the created application will help visually demonstrate the use of Oracle technologies for creating applications with user-friendly navigation, extensive functionality, and interactive user interaction. Its results will improve the quality and efficiency of applications based on Oracle and provide a better user experience. The research results can be used in the development and optimization of various applications in the future.

The results of the research carried out in the diploma project can serve as an example of creating applications with a user-friendly interface for use in the activities of accountants and other departments of universities.

Keywords: information systems, databases, DBMS, entity-relationship diagram.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

КП - курсовий проект

ПК - персональний комп’ютер

ТЗ – технічне завдання

ПЗ – пояснювальна записка

БД – база даних

СУБД – система управління базами даних

ІС – інформаційна система

ER – діаграма сутність-зв’язок

GUI – графічний інтерфейс користувача

Oracle APEX – Oracle Application Express

ВСТУП

У 21 столітті інформаційні технології полегшують життя людству багатьма способами: предмети побуту на дистанційному керуванні, системи "розумного будинку", соціальні мережі, програми для менеджменту тощо. Одним з таких прикладів є бази даних. Вони представляють собою не тільки організований набір даних, а ще й опис та засоби для їх обробки. Бази даних використовуються майже в кожній сфері праці людини, адже саме вони дозволяють швидко та зручно працювати з великою кількістю даних. Особливе значення вони мають для сучасних інформаційних систем. Можна з впевненістю казати, що на сьогоднішній момент бази даних є тим, на що спирається та навколо чого будується будь-яка інформаційна система. Існує декілька типів організації баз даних, але не можна заперечувати, що найуживанішою залишається реляційна модель організації. Суть реляційної бази даних проста – вона зберігає дані у вигляді таблиць. Саме цю модель організації використовують більшість систем управління базами даних. Їх головна функція – створення та управління набором взаємопов'язаних між собою даних. Завдяки зручному графічному інтерфейсу користувача, розподіленим базам даних, клієнт-серверним застосункам, паралельному пошук даних та їх інтелектуальному аналізу, вони суттєво полегшують та пришвидшують роботу та розробку баз даних, а, отже, й інформаційних систем у цілому.

У даному дипломному проекті метою розробки було створення інформаційної системи, для обліку нарахування стипендії студентам ВНЗ. Ця система повинна забезпечувати у багатокористувацькому режимі роботи узгодженість даних, їх цілісність, високу доступність та продуктивність. Система повинна включати в себе базу даних та інтерфейс для роботи з нею. Головними критеріями цього інтерфейсу є інтуїтивність та зручність використання.

База даних розроблялася в Oracle Application Express 23.1. Oracle APEX – це одна з найпопулярніших у світі платформа додатків, яка дає змогу створювати

масштабовані безпечні додатки з функціями світового рівня. Ці програми можна розгортати де завгодно – у хмарі чи локально. Використовуючи APEX, розробники можуть швидко розробляти та розгортати привабливі програми. APEX пропонує рішення для різноманітних випадків використання, від простої заміни електронної таблиці до критичної корпоративної системи, якою тисячі людей користуються щодня.

РОЗДІЛ 1. ОБ’ЄКТ ТА МЕТОДИ ДОСЛІДЖЕННЯ

1.1 Об'єкт та предмет дослідження

Об'єктом дослідження даного проекту є облік нарахування стипендії студентам ВНЗ. Для кожного студента інформаційна система повинна забезпечувати збереження та обробку повного набору персональних даних, а саме повне ім’я, номер залікової книжки, номер паспорта, дату зарахування та термін навчання у ВНЗ, середній бал студента за минулий семестр, групу, до якої він входить. Кожен семестр система фіксуватиме стипендію студента, її вид, його академічні оцінки в дисциплінах, вивчених ним упродовж року, інформацію про групи студентів та дисципліни, які група вивчає. Із вище зазначеного випливає, що система має зберігати інформацію про навчальні семестри.

Передбачуваними користувачами системи будуть як викладачі та адміністрація ВНЗ, так і деякі студенти. Інформаційна система повинна мати змогу відповідати на запити, направлені до неї від користувачів. Студентам в першу чергу система буде потрібна для перегляду своїх оцінок та інформації про види стипендій, викладачам – для внесення та моніторингу оцінок студента, а адміністрації – для керування та нарахування студентам стипендії відповідного виду.

У результаті аналізу предметної області можна визначити, що особливу увагу слід приділяти реалізації сутності стипендії, студента та семестру. Саме вони будуть нести в особі основну інформацію в цій базі даних, та саме до них найчастіше будуть звертатися користувачі, формуючи свої запити та вносячи в систему нову інформацію. Задля полегшення формування запитів, підвищення комфортності використання системи та спрощення організації бази даних, слід створити додаткові сутності, які втілять у собі групи студентів та академічні дисципліни, які вони вивчають впродовж навчання. Також під час розробки задля її спрощення було вирішено допустити навчання однієї групи студентів впродовж декількох семестрів одночасно. Звичайно, це додає певні елементи

абстракції в створювану інформаційну систему, але в той же самий час додає їй універсальності, яку в подальшому можна адаптувати до будь-яких необхідних користувачу умов.

Щодо зв’язків вище описаних сутностей. Існує три види зв’язків для баз даних, а саме: one to one (один до одного), many to one (багато до одного), many to many (багато до багатьох). В нашій інформаційній системі ми в основному будемо використовувати останній з представлених зв’язків. Тобто, багато студентів в багатьох семестрах відвідуючи багато дисциплін, можуть отримувати багато видів стипендій. Але є виключення, адже багато студентів можуть навчатися в одній й тій самій групі. В цьому випадку, ми використаємо зв’язок many to one.

Для подальшої роботи необхідно створити діаграму сутності-зв'язку. Для цього, було використану функціонал ресурсу Diagrams.net[1]. Це безкоштовне крос-платформне програмне забезпечення для створення графіків із відкритим кодом, розроблене на HTML5 і JavaScript. Його інтерфейс можна використовувати для створення діаграм, таких як блок-схеми, каркасні схеми, діаграми UML, організаційні діаграми та мережеві діаграми.

На рисунку 1.1 представлена ER-діаграма для досліджуваної предметної області.

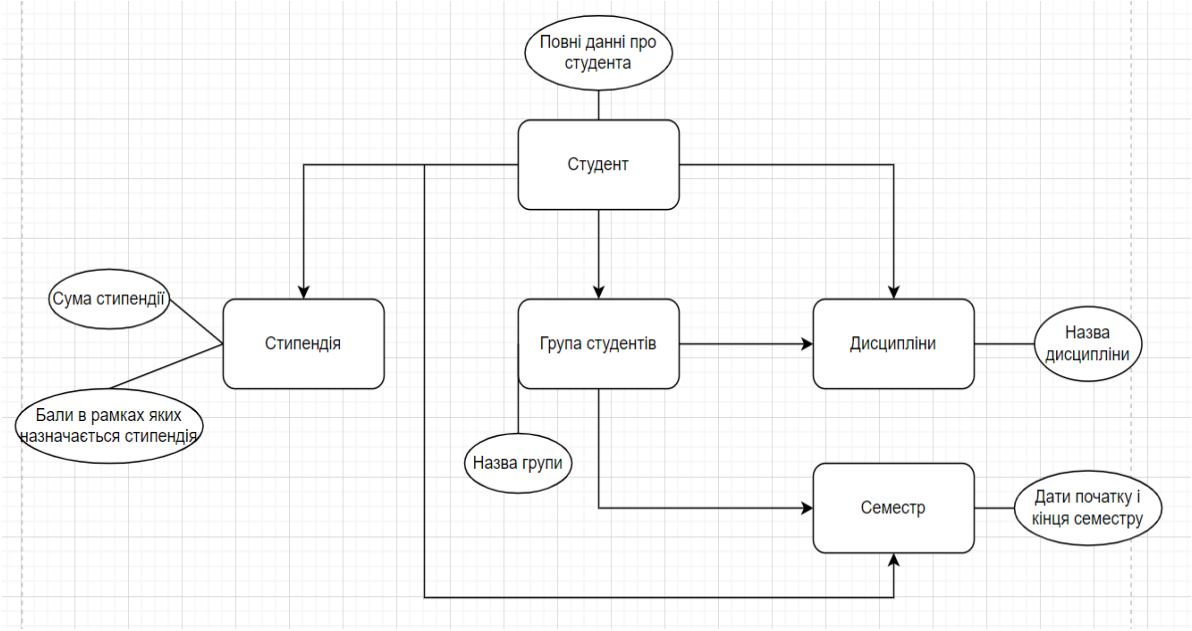

Рисунок 1.1 – ER-діаграма обліку нарахування стипендії студентам ВНЗ

Отже, інформаційна система обліку стипендії студентів ВНЗ повинна забезпечувати перегляд, внесення та зміну даних, виводити повну інформацію стосовно студентів, їх академічних успіхів та стипендій.

1.2 Аналіз методів і засобів проектування та реалізації інформаційних систем

База даних — це організований набір даних, які зберігаються та доступні в електронному вигляді. Вони призначені для маніпулювання великими обсягами інформації шляхом введення, зберігання, пошуку та керування цією інформацією[9]. Невеликі бази даних можна зберігати у файловій системі, тоді як великі бази даних розміщуються на комп’ютерних кластерах або хмарних сховищах. У цьому випадку інформаційна система, яка розробляється, сама по собі є досить локальною, до того ж для її розробки достатньо використовувати невелику базу, якої буде достатньо для перевірки працездатності всіх її функцій. Дизайн баз даних охоплює формальні методи та практичні міркування, включаючи моделювання даних, ефективне представлення та зберігання даних, мови запитів, безпеку та конфіденційність даних, а також питання розподілених обчислень, включаючи підтримку одночасного доступу та відмовостійкість.

Прикладами використання баз даних можуть бути:

– автоматизовані системи обліку;

– система менеджменту, яка зберігає у базах даних інформацію про робітників;

– словники або лінгвістичні бази даних;

– бази даних лікарень та інших медзакладів з інформацією про клієнтів;

– різноманітні реєстри та каталоги.

Реляційна база даних є однією з найбільш часто використовуваних систем зберігання даних. Вона має низку переваг і обмежень, які перешкоджають її використанню, але все ще є ефективною системою для зберігання інформації про зв’язки між різними об’єктами.

Мета реляційної моделі полягає в тому, щоб забезпечити декларативний метод для визначення даних і запитів: користувачі безпосередньо вказують, яку інформацію містить база даних і яку інформацію вони хочуть отримати від неї, а програмне забезпечення системи управління базою даних піклується про опис структур даних для зберігання даних та процедури пошуку для формування відповідей на запити.

Більшість реляційних баз даних використовують мову визначення даних і запитів SQL, ці системи реалізують те, що можна розглядати як інженерне наближення до реляційної моделі.

Інші моделі включають ієрархічну модель і мережеву модель. Деякі системи, що використовують ці старі архітектури, все ще використовуються сьогодні в центрах обробки даних із великими потребами в даних або там, де існуючі системи настільки складні й абстрактні, що перехід на системи, що використовують реляційну модель, був би надто дорогим. Також варто зазначити існування новішої об’єктно-орієнтованої моделі даних.

Ключові особливості реляційних баз даних включають можливість зробити дві таблиці схожими на одну, об’єднати кілька таблиць разом за ключовими полями, створювати складні індекси, які добре працюють і якими

легко керувати, і підтримувати цілісність даних для максимальної точності даних.

Існує багато переваг, пов’язаних із використанням реляційної бази даних для керування потребами в даних. Наприклад, якщо є необхідність переглянути всі контакти у телефонній книзі (або інших типів), то все, що потрібно зробити, це ввести один запит у пошуковий рядок і миттєво побачити всі контакти, перелічені там. Це економить час від необхідності проходити вручну.

Переваги реляційної бази даних:

– простота моделі. На відміну від інших типів моделей баз даних, реляційна модель бази даних набагато простіша. Вона не вимагає жодних складних запитів, оскільки не має обробки чи структурування запитів, тому для обробки даних достатньо простих запитів SQL;

– простота використання. Користувачі можуть легко отримувати необхідну інформацію за лічені секунди, не потураючи складності бази даних. Для виконання складних запитів використовується спеціалізована мова запитів;

– точність. Ключовою особливістю реляційних баз даних є те, що вони чітко визначені та добре організовані, тому дані не дублюються. Реляційні бази даних мають точність завдяки своїй структурі без дублювання даних;

– цілісність даних. Реляційні бази даних також широко використовуються для забезпечення цілісності даних, оскільки вони забезпечують узгодженість у всіх таблицях;

– нормалізація. Оскільки дані стають дедалі складнішими, зростає потреба в ефективних способах їх зберігання. Нормалізація — це метод, який розбиває інформацію на керовані блоки, щоб зменшити розмір сховища. Дані можна розбити на різні рівні, будь-який рівень потребує підготовки перед переходом на інший рівень нормалізації даних. Нормалізація бази даних також гарантує, що реляційна база даних не має

аномалій чи відхилень у своїй структурі і нею можна точно маніпулювати. Це гарантує збереження цілісності під час використання даних із цієї бази даних для прийняття бізнес-рішень;

– одночасне використання. Кілька користувачів можуть отримати доступ до бази даних для отримання інформації одночасно, навіть якщо дані оновлюються;

– безпека. Дані захищені, оскільки система управління реляційною базою даних дозволяє лише авторизованим користувачам мати прямий доступ до даних. Жоден неавторизований користувач не може отримати доступ до інформації.

Хоча використання реляційних баз даних має більше переваг, воно також має деякі обмеження. Розглянемо обмеження та недоліки використання реляційної бази даних:

– проблема технічного обслуговування. Підтримка реляційної бази даних з часом ускладнюється через збільшення даних. Розробникам і програмістам доводиться витрачати багато часу на підтримку бази даних;

– вартість. Налаштування та обслуговування системи реляційної бази даних є дорогим. Початкова вартість лише програмного забезпечення може бути досить високою для невеликих підприємств, але це стає ще складніше, якщо врахувати необхідність найму професійного технічного спеціаліста, який також повинен мати досвід роботи з цим конкретним видом програми;

– фізичне зберігання. Реляційна база даних складається з рядків і стовпців, що вимагає багато фізичної пам’яті, оскільки кожна виконана операція залежить від окремого сховища. Вимоги до фізичної пам’яті можуть зростати разом зі збільшенням обсягу даних;

– відсутність масштабованості. Під час використання реляційної бази даних на кількох серверах її структура змінюється і стає важкою для обробки, особливо коли кількість даних велика. Через це дані не можна

масштабувати на різних серверах фізичного зберігання. Зрештою, це впливає на її продуктивність, тобто відсутність доступності даних і часу завантаження тощо. Оскільки база даних стає більшою або більш розподіленою з більшою кількістю серверів, це матиме негативні наслідки, такі як затримка та доступність, що впливає на загальну продуктивність;

– складність у структурі. Реляційні бази даних можуть зберігати дані лише в табличній формі, що ускладнює представлення складних зв'язків між об'єктами. Це проблема, оскільки багатьом програмам потрібна більше ніж одна таблиця для зберігання всіх необхідних даних, необхідних для логіки програми.

– зниження продуктивності з часом. Реляційна база даних може стати повільнішою не лише через її залежність від кількох таблиць. Коли в системі є велика кількість таблиць і даних, це спричиняє збільшення складності. Це може призвести до уповільнення часу відповіді на запити або навіть до повної їх відмови залежно від того, скільки людей увійшли на сервер у певний час.

У підсумку, реляційні бази даних традиційно використовуються для управління даними в організації. Основні переваги використання реляційних баз даних полягають у тому, що до них можна легко надсилати запити, вони дозволяють використовувати збережені процедури для маніпулювання даними та забезпечують послідовний дизайн бази даних. Вони також мають обмеження, коли мова йде про великі обсяги транзакцій або зберігання великих обсягів даних, може виникнути проблема швидкості[5]. З навчальною метою саме реляційна модель підходить найбільше для виконання даного проекту.

Для створення та управлінням базами даних використовують спеціальне програмне забезпечення – системи управління базами даних. При розробці інформаційної системи будемо використовувати СУБД Oracle. База даних

Oracle — це система управління реляційною базою даних від корпорації Oracle[2]. Розглянемо основні переваги даної СУБД:

– продуктивність. Oracle має процедури та принципи функціонування, які допомагають досягти високого рівня продуктивності бази даних. Існує можливість аналізу та зменшення часу виконання запитів і операцій за допомогою методів оптимізації продуктивності. Ця техніка допомагає швидше отримувати та змінювати дані;

– портативність. Базу даних Oracle можна перенести на будь-які інші платформи. Її можна використовувати для приблизно 20 мережевих протоколів, а також понад 100 апаратних платформ. Ця база даних спрощує написання програми;

– резервне копіювання та відновлення. Завжди краще зробити правильну резервну копію всього онлайн-резервного копіювання та відновлення Oracle. База даних Oracle спрощує швидке відновлення за допомогою функції Recovery Manager. Вона може відновити файли бази даних під час простою або збоїв. Її можна використовувати для онлайн-резервного копіювання, архівного резервного копіювання та постійного архівування. Також є можливість використовувати SQL* PLUS для відновлення, яке називається відновленням, керованим користувачем;

– PL/SQL. однією з найбільших переваг використання бази даних Oracle є підтримка розширення PL/SQL для процедурного програмування;

– кілька баз даних. База даних Oracle дозволяє керувати кількома екземплярами бази даних на одному сервері. Керування ресурсами бази даних і об’єднання екземплярів можуть працювати разом, щоб керувати службами в кількох екземплярах;

– технологія Flashback. Ця перевага надається в останній версії Oracle. Це дозволяє відновити ті дані, які були неправильно видалені або втрачені через людські помилки, як-от випадкове видалення цінних даних, видалення неправильних даних або скидання таблиці.

Як і у всього, в даної СУБД є певні недоліки, зокрема:

– складність. Oracle не рекомендується використовувати, якщо користувачі слабо технічно підготовлені та мають обмежені технічні навички, необхідні для роботи з базою даних Oracle. Також не рекомендується використовувати, якщо компанія шукає базу даних з обмеженою функціональністю та просту у використанні[6];

– висока вартість ліцензії. Ціна продуктів Oracle дуже висока в порівнянні з іншими базами даних. Тому користувачі частіше вибирають інші менш дорогі варіанти, такі як MS SQL Server, MySQL тощо;

– складність керування. Бази даних Oracle часто набагато складніші з точки зору керування певними видами діяльності.

Незважаючи на певні недоліки, дана СУБД все ще залишається досить комфортною у використанні. Для розробки було використано Oracle Application Express 23.1.

РОЗДІЛ 2. ВИЗНАЧЕННЯ ШЛЯХІВ ТА МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Визначення шляхів та методів розробки бази даних

Перед початком розробки, остаточно визначимо модель нашої майбутньої БД.

Реляційна модель – це підхід до керування даними за допомогою структури та мови, де всі дані представлені в термінах кортежів, згруповані у відношення. База даних, організована в термінах реляційної моделі, називається реляційною базою даних. Реляційна модель даних дозволяє розробнику бази даних створювати послідовне, логічне представлення інформації. Узгодженість досягається шляхом включення оголошених обмежень у проект бази даних, який зазвичай називають логічною схемою. Теорія включає процес нормалізації бази даних, за допомогою якого дизайн з певними бажаними властивостями може бути обраний із набору логічно-еквівалентних альтернатив. Плани доступу та інші деталі впровадження та роботи обробляються механізмом СУБД і не відображаються в логічній моделі. Це контрастує зі звичайною практикою для СУБД SQL, в яких налаштування продуктивності часто вимагає внесення змін до логічної моделі.

Основним реляційним будівельним блоком є домен або тип даних, який сьогодні зазвичай скорочується як тип. Кортеж – це невпорядкований набір значень атрибутів[8]. Атрибут у свою чергу – це невпорядкована пара імені атрибута та імені типу. Значення атрибута – це конкретне дійсне значення для типу атрибута. Це може бути або скалярне значення або більш складний тип.

Відношення складається із заголовка та тіла. Заголовок – це набір атрибутів. Тіло - це набір кортежів. Заголовок відношення також є заголовком кожного з його кортежів.

Після аналізу даної інформації, а також переваг та недоліків даної моделі в попередньому розділі, було вирішена, що саме вона найбільше підходить під задачу реалізації інформаційної системи обліку нарахування стипендії студентам ВНЗ.

У попередньому розділі було створено спрощену схему БД, а саме діаграму ER. Ця модель даних представляє собою концептуальну схему. Створення логічної схеми даних є одним з етапів проектування бази даних. При її побудові, в неї слід включити набір таблиць із даними та їх типами та первинними ключами, а також зв’язки між відношеннями, що представляють собою зовнішні ключі.

2.2 Визначення вимог до додатку

Додаток повинен відповідати таким вимогам:

- Навігація та інтерфейс: Додаток має мати зручний та зрозумілий інтерфейс, щоб користувачі могли легко орієнтуватися в ньому, взаємодіяти з експонатами та виконувати різноманітні дії.

- Інтерактивність: Додаток має надавати можливість взаємодії користувачів з даними шляхом різних інструментів, кнопок, перетягувань та інших взаємодіючих елементів.

- Продуктивність: Додаток має працювати без затримок та лагів, щоб надати користувачам приємний досвід використання.

- Ієрархія користувачів: Додаток має надавати можливість користувачам мати різні рівні доступу та взаємодії з додатком.

- Сумісність: Додаток має бути сумісним з різними операційними системами та пристроями, зокрема комп'ютерами, планшетами та смартфонами.

РОЗДІЛ 3. РОЗРОБКА БАЗИ ДАНИХ ТА ДОДАТКУ

3.1 Проектування схеми бази даних

Розпочати роботу над інформаційною системою потрібно з моделювання даних. Моделювання даних – це процес створення моделі даних для даних, які зберігатимуться в БД. Ця модель даних є концептуальним представленням об’єктів даних, зв’язків між різними об’єктами даних і правил. Моделювання даних допомагає у візуальному представленні і забезпечує виконання бізнес-правил і дотримання нормативних вимог. Моделі даних забезпечують узгодженість умов іменування, значень за замовчуванням, семантики, безпеки, одночасно забезпечуючи якість даних[7].

В СУБД модель даних визначається як абстрактна модель, яка організовує опис даних, семантику даних і обмеження узгодженості даних. Модель даних наголошує на тому, які дані потрібні та як вони мають бути організовані, а не на тому, які операції виконуватимуться з даними. Модель даних схожа на будівельний план архітектора, який допомагає будувати концептуальні моделі та встановлювати зв’язок між елементами даних.

Основною метою використання моделі даних є забезпечення точного представлення всіх об’єктів даних, необхідних для бази даних. Відсутність даних призведе до створення помилкових звітів і дасть невірні результати. Також модель даних допомагає проектувати базу даних на концептуальному, фізичному та логічному рівнях.

На даному етапі розробки нам необхідно створити логічну та фізичну модель даних нашої БД.

Логічна модель даних використовується для визначення структури елементів даних і встановлення зв’язків між ними. Логічна модель даних додає додаткову інформацію до елементів концептуальної моделі даних. Перевага використання логічної моделі даних полягає в тому, що вона забезпечує фундамент для створення основи для фізичної моделі. Однак структура

моделювання залишається загальною. Також потрібно перевірити та налаштувати деталі зв’язків.

Фізична ж модель даних описує специфічну для СУБД реалізацію моделі даних. Вона пропонує абстракцію бази даних і допомагає створити схему. Фізична модель даних допомагає візуалізувати структуру БД шляхом представлення ключів таблиць даних, обмежень, індексів, тригерів та інших функцій СУБД.

Задля побудови логічної та фізичної моделі даних, скористаємося безкоштовним онлайн застосунком DBDiagram[3]. Цей ресурс дозволив об’єднати логічну то фізичну модель в одну, в результаті було отримано схему, зображену на Рисунку 3.1. У додатку А знаходиться код, згенерований сервісом, за допомогою якого можна відтворити дану модель.

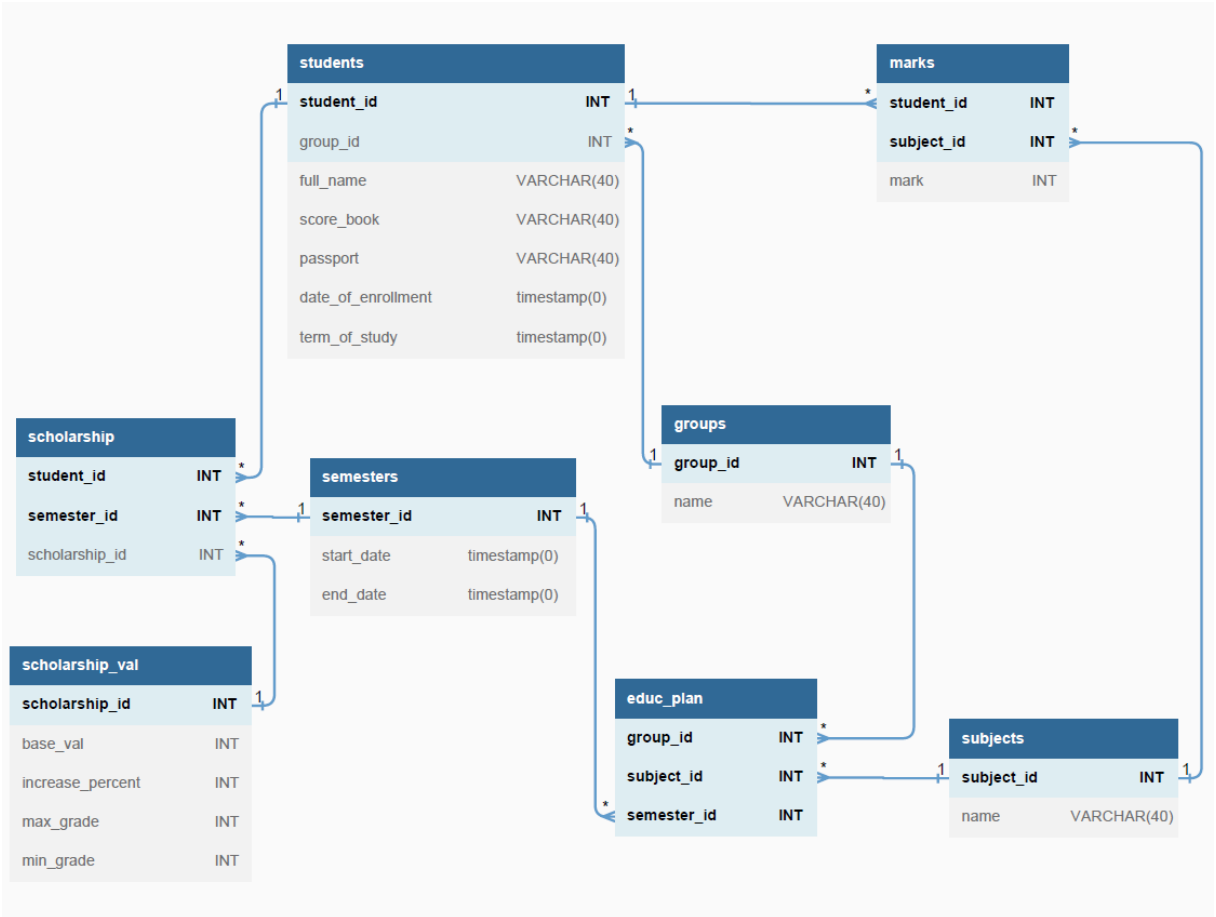

Рисунок 3.1 – Логічна та фізична моделі бази даних

Сутності, описані в попередніх розділах при створенні ER-діаграми, тепер отримали свою реалізацію:

– таблиця з даними про студентів – STUDENTS,

– таблиця з даними про семестри – SEMESTERS,

– таблиця з даними про групи студентів – GROUPS,

– таблиця з даними про академічні дисципліни – SUBJECTS,

– таблиця з даними про стипендію та її види – SCHOLARSHIP_VAL.

Для спрощення та пришвидшення роботи БД було створено три допоміжні таблиці, які зв’язують між собою визначені сутності:

– SCHOLARSHIP зв'язує студентів та семестр, в якому вони отримують стипендію,

– MARKS зв'язує студента та дисципліну, за яку він отримує оцінку,

– EDUC_PLAN зв'язує групу студентів з дисципліною, яку вони вивчають в цьому семестрі.

3.2 Створення об’єктів бази даних

Після розробки схем, настав час створювати таблиці в СУБД та заповнювати їх даними. Дані було згенеровано за допомогою безкоштовного онлайн ресурсу Mockaroo[4]. Код для створення таблиць представлено в додатку Б, де також прописане створення індексів. Було вирішено створити індекси по полям імен для таблиці студентів, груп, семестрів, та дисциплін, оскільки система часто буде звертатися до БД шукаючи співпадіння по цим полям, а також по назвам груп студентів, оскільки в них знаходяться унікальні значення, і по ним може проходити вибірка. Код для внесення згенерованих даних представлено в додатку В. На рисунках 3.2–3.17 представлено структуру створених таблиць та внесені в них дані.

SCHOLARSHIP_VAL

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
SCHOLARSHIP_ID	NUMBER(6,0)	N		1
BASE_VAL	NUMBER(6,0)	Y		
INCREASE_PERCENT	NUMBER(6,0)	Y		
MAX_GRADE	NUMBER(6,0)	Y		
MIN_GRADE	NUMBER(6,0)	Y		

Рисунок 3.2 – Структура таблиці SCHOLARSHIP_VAL

SCHOLARSHIP_VAL

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Samp Queri

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

SCHOLARSHIP_ID	BASE_VAL	INCREASE_PERCENT.	MAX_GRADE	MIN_GRADE
1	82	40	91	35
2	60	30	62	55
3	22	10	31	10
4	75	40	84	9
5	3	70	92	72
6	37	70	47	2
7	99	50	81	34
8	12	60	95	79
9	68	40	89	11
10	3	20	20	1
11	86	50	82	2
12	1	10	75	70
13	48	20	100	90
14	67	20	90	80
15	0	0	0	0

Рисунок 3.3 – Дані таблиці SCHOLARSHIP_VAL

SCHOLARSHIP

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
STUDENT_ID	NUMBER(6,0)	N		1
SEMESTER_ID	NUMBER(6,0)	N		2
SCHOLARSHIP_ID	NUMBER(6,0)	Y		

Рисунок 3.4 – Структура таблиці SCHOLARSHIP

SCHOLARSHIP

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Samp Queri

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

STUDENT_ID	SEMESTER_ID	SCHOLARSHIP_ID
1	1	3
2	3	2
3	3	1
4	7	6
5	4	5
6	4	15
7	6	10
8	7	9
9	7	8
10	7	7
11	2	15
12	5	11
13	1	13
14	2	12
15	6	6

Рисунок 3.5 – Дані таблиці SCHOLARSHIP

STUDENTS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
STUDENT_ID	NUMBER(6,0)	N		1
GROUP_ID	NUMBER(6,0)	Y		
FULL_NAME	VARCHAR2(40 BYTE)	Y		
SCORE_BOOK	VARCHAR2(40 BYTE)	Y		
PASSPORT	VARCHAR2(40 BYTE)	Y		
DATE_OF_ENROLLMENT	DATE	Y		
TERM_OF_STUDY	DATE	Y		

Рисунок 3.6 – Структура таблиці STUDENTS

STUDENTS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Sample Queries

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

STUDENT_ID...	GROUP_ID	FULL_NAME	SCORE_BOOK	PASSPORT	DATE_OF_ENROLLMENT	TERM_OF_STUDY
1	1	Carlin Hancorn	0880642	924198320	08/18/2020	10/21/2022
2	1	Norrie Ovill	4754281	999630186	10/15/2019	11/23/2022
3	1	Joya Pratte	6653751	807608304	12/31/2019	10/18/2022
4	1	Katya Bearcroft	7943376	301478518	03/01/2020	01/16/2023
5	1	Rana Emer	4232767	727037755	06/22/2020	01/18/2023
6	1	Hyatt Nials	6592536	094002878	06/29/2020	11/17/2022
7	1	Tabbitha Chander	9779864	079903279	06/17/2020	01/02/2023
8	1	Janel Pedwell	5720734	766781059	02/10/2020	11/30/2022
9	1	Daron Woodro...	0864848	122593587	08/07/2020	09/22/2022
10	1	Jess Nolton	0945045	214386344	02/10/2020	05/25/2023
11	2	Arnaldo Knellen	9510000	472311311	01/09/2020	10/25/2022
12	3	Tiffy Millsap	7686848	916382197	11/26/2019	12/08/2022
13	4	Charley Tudgay	6717249	427386952	12/14/2019	04/12/2023
14	5	Igor Felgate	7068624	309376031	07/17/2020	07/21/2022
15	5	Ray Seeking	1080408	667237784	11/20/2019	04/23/2023
16	4	Test student	0890089	8909890	09/01/2022	06/30/2026

Рисунок 3.7 – Дані таблиці STUDENTS

SEMESTERS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
SEMESTER_ID	NUMBER(6,0)	N		1
START_DATE	DATE	Y		
END_DATE	DATE	Y		

Рисунок 3.8 – Структура таблиці SEMESTERS

SEMESTERS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Sample Queries

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

SEMESTER_ID	START_DATE	END_DATE
1	09/01/2019	12/20/2019
2	01/15/2020	06/20/2020
3	09/01/2020	12/20/2020
4	01/15/2021	06/20/2021
5	09/01/2021	12/20/2021
6	01/15/2022	06/20/2022
7	09/01/2022	12/20/2022

Рисунок 3.9 – Дані таблиці SEMESTERS

GROUPS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
GROUP_ID	NUMBER(6,0)	N		1
NAME	VARCHAR2(40 BYTE)	Y		

Рисунок 3.10 – Структура таблиці GROUPS

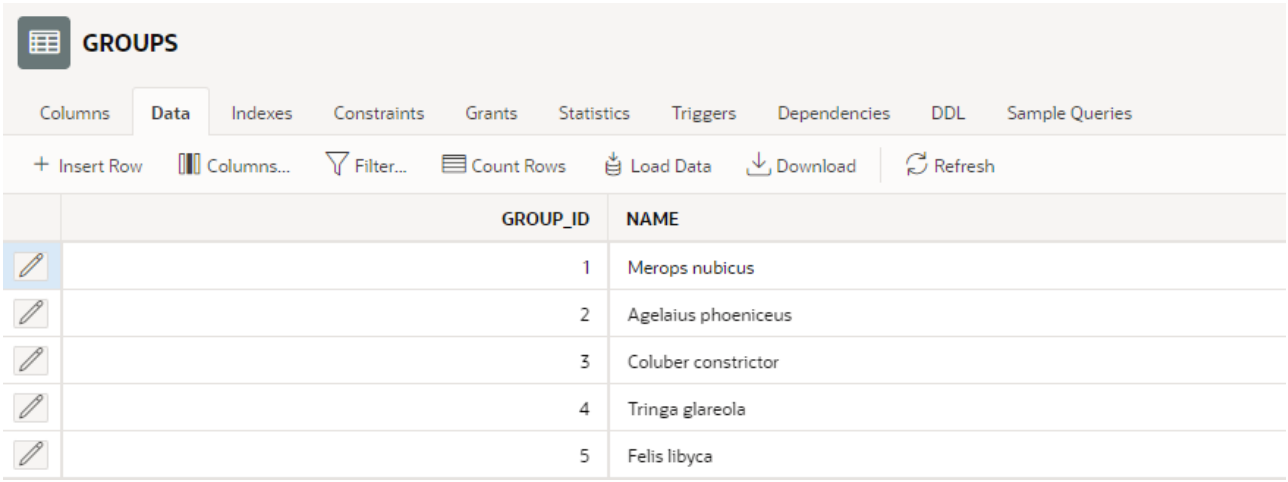

Рисунок 3.11 – Дані таблиці GROUPS

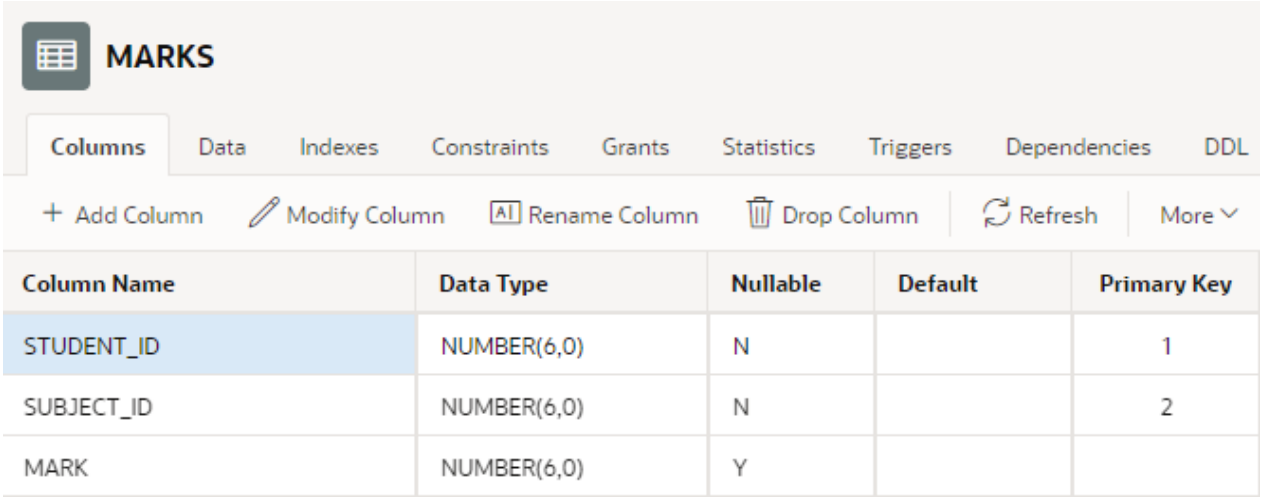

Рисунок 3.12 – Структура таблиці MARKS

MARKS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Sample Queries

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

STUDENT_ID	SUBJECT_ID	MARK
1	15	41
2	1	61
3	4	96
4	3	2
5	6	60
6	5	27
7	8	88
8	9	32
9	11	80
10	7	63
11	10	52
12	12	53
13	14	60
14	13	95
15	2	1
16	1	100

Рисунок 3.13 – Дані таблиці MARKS

EDUC_PLAN

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
GROUP_ID	NUMBER(6,0)	N		1
SUBJECT_ID	NUMBER(6,0)	N		3
SEMESTER_ID	NUMBER(6,0)	N		2

Рисунок 3.14 – Структура таблиці EDUC_PLAN

EDUC_PLAN

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Sample Queries

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

GROUP_ID	SUBJECT_ID	SEMESTER_ID
1	3	1
1	8	2
1	12	3
1	7	4
1	13	4
1	1	6
1	2	7
1	9	7
1	11	7
1	14	7
2	8	2
3	4	5
4	10	1
5	15	2
5	5	6

Рисунок 3.15 – Дані таблиці EDUC_PLAN

SUBJECTS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL

+ Add Column | Modify Column | Rename Column | Drop Column | Refresh | More

Column Name	Data Type	Nullable	Default	Primary Key
SUBJECT_ID	NUMBER(6,0)	N		1
NAME	VARCHAR2(40 BYTE)	Y		

Рисунок 3.16 – Структура таблиці SUBJECTS

SUBJECTS

Columns | Data | Indexes | Constraints | Grants | Statistics | Triggers | Dependencies | DDL | Sample Queries

+ Insert Row | Columns... | Filter... | Count Rows | Load Data | Download | Refresh

SUBJECT_ID	NAME
1	Ukraine
2	Poland
3	Italy
4	Greece
5	Portugal
6	Philippines
7	South Africa
8	Argentina
9	Canada
10	Vietnam
11	Spain
12	Japan
13	Taiwan
14	Montenegro
15	Egypt

Рисунок 3.17 – Дані таблиці SUBJECTS

Заповнивши таблиці, можемо почати створювати функціонал бази даних. Для реалізації редагування та створення користувачами контенту в БД, для таблиць було створено тригери, для генерації ID (рисунок 3.18), код яких знаходиться в Додатку Г.

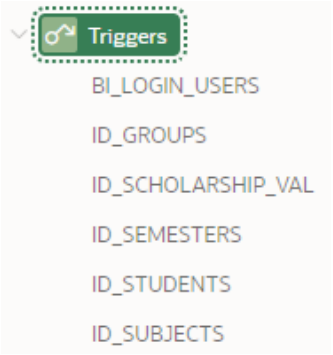

Рисунок 3.18 – Тригери, для генерації ID

Також, тут можна побачити тригер BI_LOGIN_USERS, з аналогічним функціоналом, створеним для таблиці з зареєстрованих користувачів.

3.3 Проектування та реалізація додатку

Після створення таблиць БД та заповнення їх даними, можемо приступати до створення для інформаційної системи програмного додатку. Для цього, як зазначалося в попередніх розділах, будемо використовувати Oracle Application Express.

Oracle Application Express — це швидкий інструмент розробки веб-додатків для бази даних Oracle. Використовуючи лише веб-браузер, користувач може розробляти професійні програми, які є швидкими та безпечними. Завдяки вбудованим функціям, таким як теми інтерфейсу користувача, елементи керування навігацією, обробники форм і гнучкі звіти, Oracle APEX прискорює процес розробки додатків. З точки зору кінцевого користувача, для розгорнутих програм потрібен лише браузер і доступ до бази даних Oracle, з використанням якої працює APEX. Ще одним плюсом цього середовища є його вбудованість в основну інфраструктуру Oracle. Таким чином, весь наш цикл розробки проходив в єдиній інфраструктурі, що значно пришвидшило роботу над системою.

На рисунку 3.19 показано попереднє налаштування створення додатку.

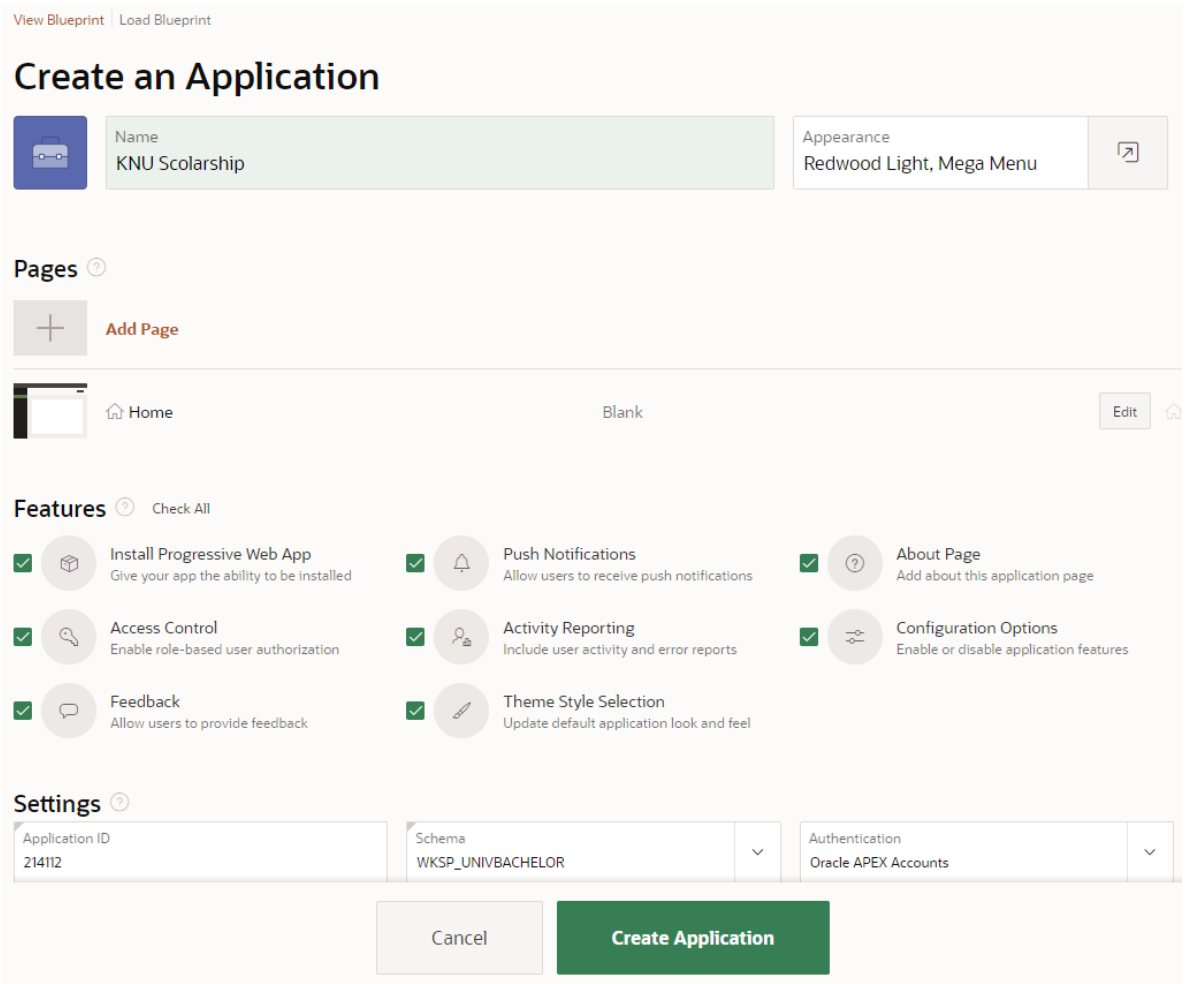

Рисунок 3.19 – Налаштування при створенні додатку

Після створення першої версії додатку, було розпочато роботу над створенням системи реєстрації та розділу користувачів по типу прав доступу.

3.3 Створення процесу реєстрації

При створенні додатку, Oracle автоматично створив свою систему аутентифікації, записавши всіх користувачів, які мають доступ до онлайн-workspace де розробляється інформаційна система. На етапі тестування цієї системи було б достатньо, але так як однією з вимог до нашого додатку є можливість реєстрації нових користувачів(з яких тільки деякі в перспективі матимуть доступ до Oracle workspace), то розробимо нову систему аутентифікації. Для цього, в першу чергу потрібно створити таблицю, в якій будуть зберігатися дані користувачів. На рисунку 3.20, зображено її структуру.

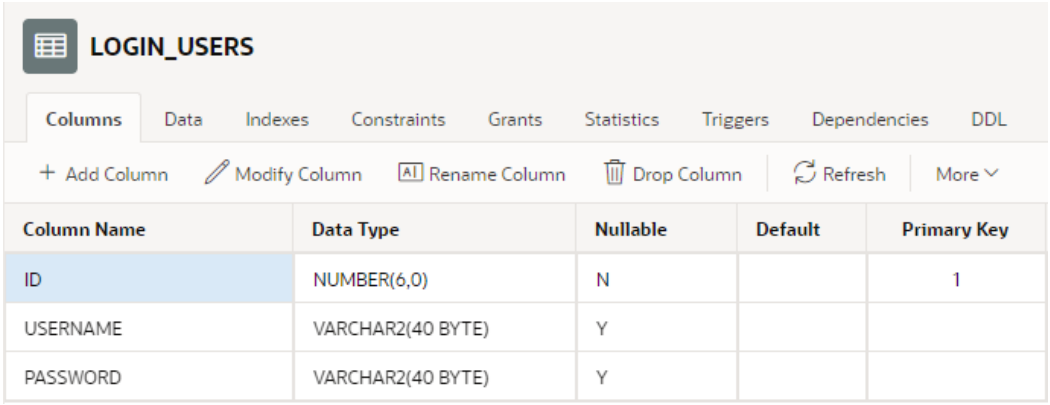

Рисунок 3.20 – Структура таблиці LOGIN_USERS

Також, була створена функція для входу у додаток використовуючі дані зареєстрованого користувача. Код створення таблиці та функції знаходиться у Додатку Д.

Далі, налаштуємо новий процес аутентифікації. Для цього, зайдемо в Oracle App Builder (рисунок 3.21), оберемо наш додаток, зайдемо в Shared Components (рисунок 3.22), та оберемо в блоці Security пункт Authentication Schemes. Відкриється вікно із списком доступних схем аутентифікації, де знаходилась тільки автоматично згенерована схема Oracle APEX Accounts.

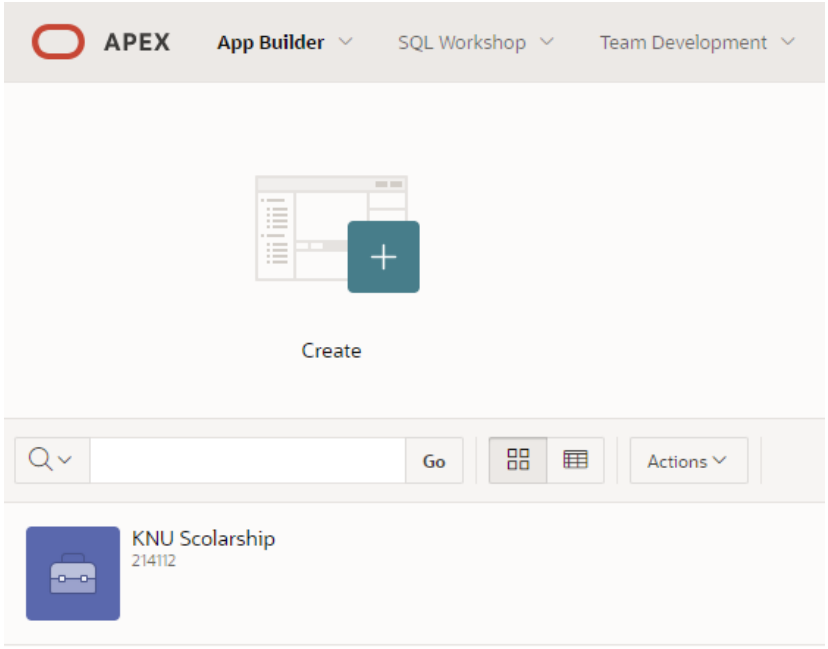

Рисунок 3.21 – Сторінка Oracle App Builder

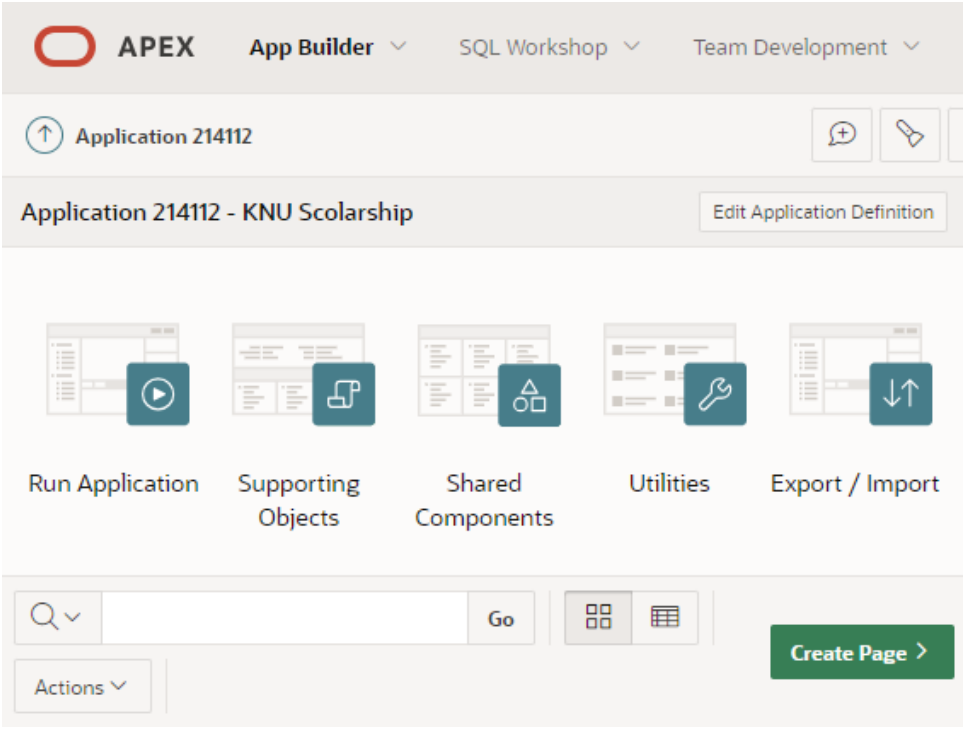

Рисунок 3.22 – Сторінка додатку KNU Scolarship

Створимо нашу схему аутентифікації. Для цього, натиснемо кнопку Create, назвемо нашу схему custom_authentication, і в пункті Authentication Function Name пропишемо вже створену нами функцію (рисунок 3.23).

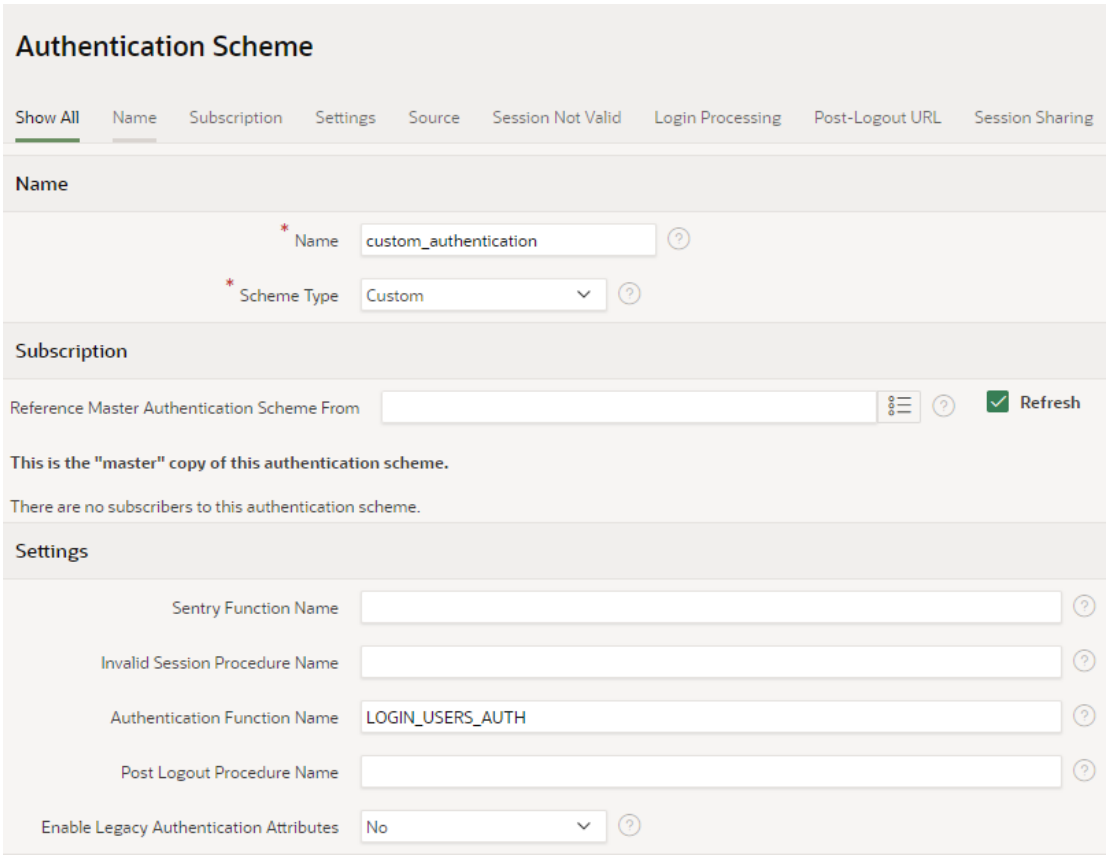

Рисунок 3.23 – Схема аутентифікації custom_authentication

В результаті, було створену нову схему аутентифікації, яку будемо використовувати в подальшому, а також побачили, де при необхідності можемо перемкнути цю схему на згенеровану Oracle.

3.4 Огляд та налаштування додатку

При створенні додатку, Oracle автоматично використав тему, та налаштування, які ми зазначили в первинних налаштуваннях.

Першим що користувач побачить у додатку буде звісно форма входу в нього, яку ми можемо побачити на рисунку 3.24.

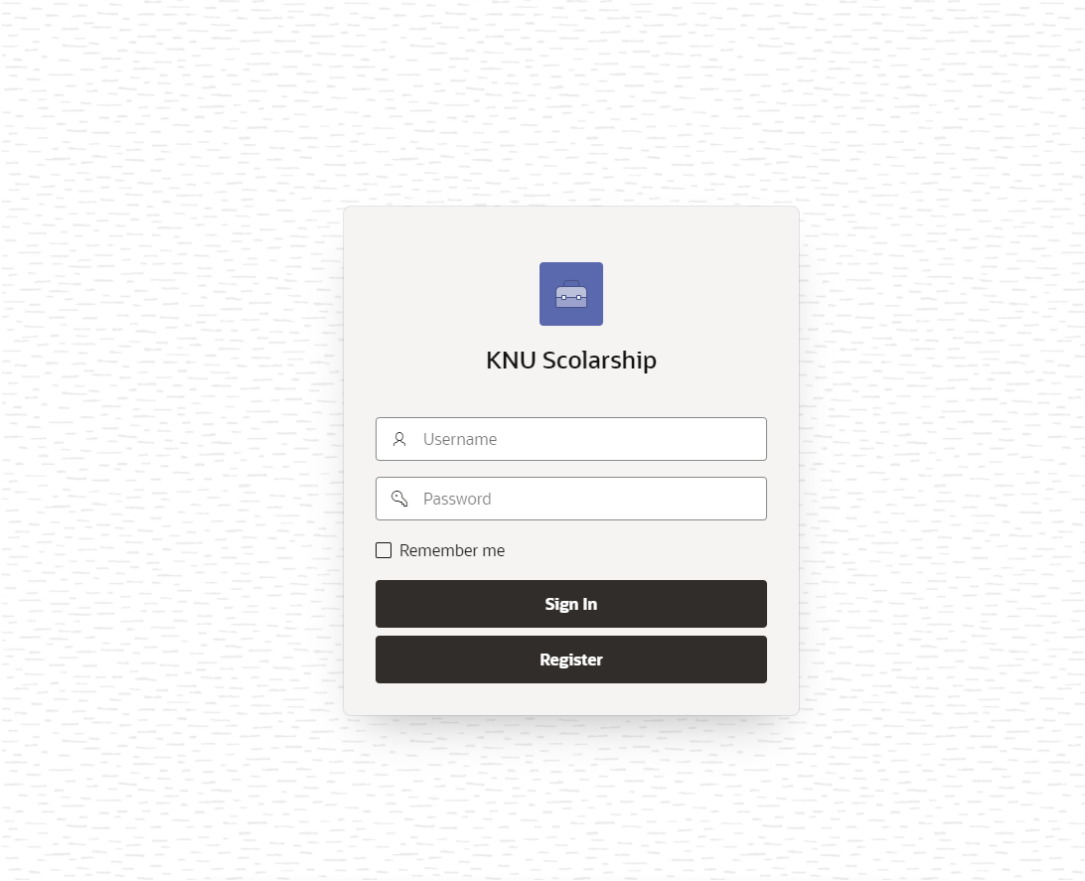

Рисунок 3.24 – Форма входу в додаток KNU Scolarship

Користувач, якщо він вже зареєстрований, може ввести свої дані для входу в додаток, або, якщо він є незареєстрований, ввести свої дані у форму, показану на рисунку 3.25, яка з’явиться після натискання кнопки Register.

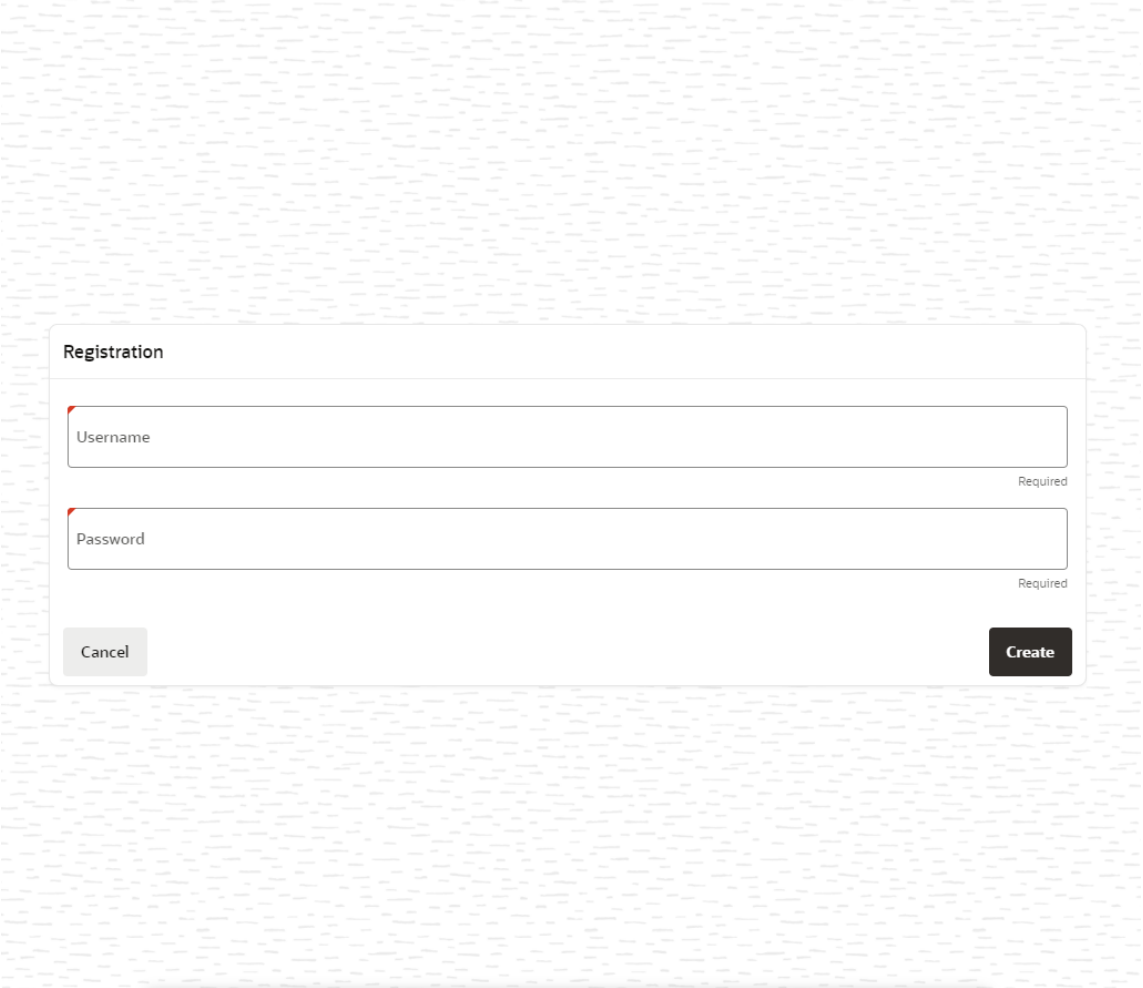

Рисунок 3.25 – Форма реєстрації в додатку KNU Scolarship

Після внесення даних, користувача поверне на форму входу (рисунок 3.24), де він може внести свої дані. Але, якщо адміністратори додатку, через адміністративну панель не надали користувачу жодного рівня доступу, він не зможе зайти у додаток. Звичайно, для великих веб-платформ, такий спосіб надання прав не є ефективним, але наш додаток не створювався для використання широким загалом, тому такий варіант реєстрації є оптимальним.

Зайдемо в додаток використовуючі дані користувача із правами Reader та Contributor. Такі користувачі мають доступ до всіх основних таблиць, а також право редагувати їх зміст. Перше, що побачить користувач буде головною сторінкою (рисунок 3.26). Тут знаходяться картки з посиланням на всі доступні користувачу таблиці додатку. Також, в шапці сайту знаходяться елементи з функціоналом, який дозволяє завантажити додаток, написати відгук, перейти на

сторінку About KNU Scholarship, а також дані користувача, які дозволяють включити повідомлення від додатку, а також вийти зі свого аккаунту.

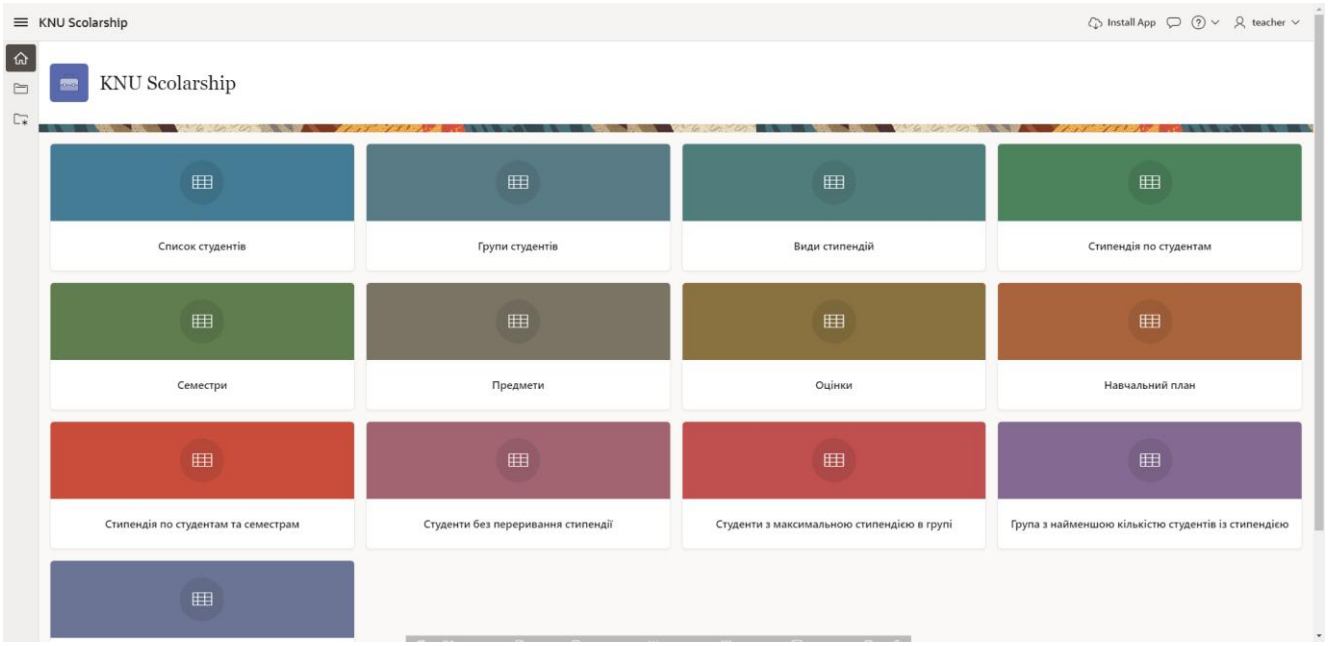

Рисунок 3.26 – Головна сторінка KNU Scolarship

Зліва знаходиться навігаційне меню, яке можна розкрити (рисунок 3.27).

Рисунок 3.27 – Навігаційне меню KNU Scolarship

Кнопка Home є посиланням на головну сторінку, Базові таблиці є папкою, де зібрані всі таблиці, створені нами в попередніх розділах, крім таблиці з зареєстрованими користувачами. При кліку на папку, вона розкриється, показуючи список посилань на таблиці (рисунок 3.28). Папка Кастомні таблиці працює аналогічно.

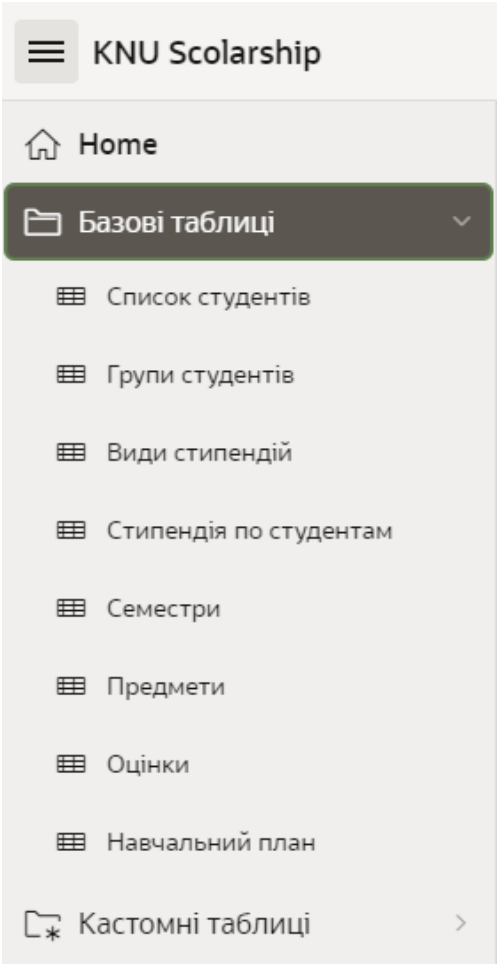

Рисунок 3.28 – Функціонал папок у навігаційному меню

При тестуванні додатку на різних пристроях було помічено, що розкриття папок лише при нажаті на стрілочку на цьому елементі може бути не зручним при використанні додатку на маленьких екранах. Для вирішення цієї проблеми було створено функцію, код якої можна знайти в Додатку Е, яка відкривала папку при нажаті на будь яку частину елемента папки. Для цього вона була записана на Global Page нашого додатку (рисунок 3.29).

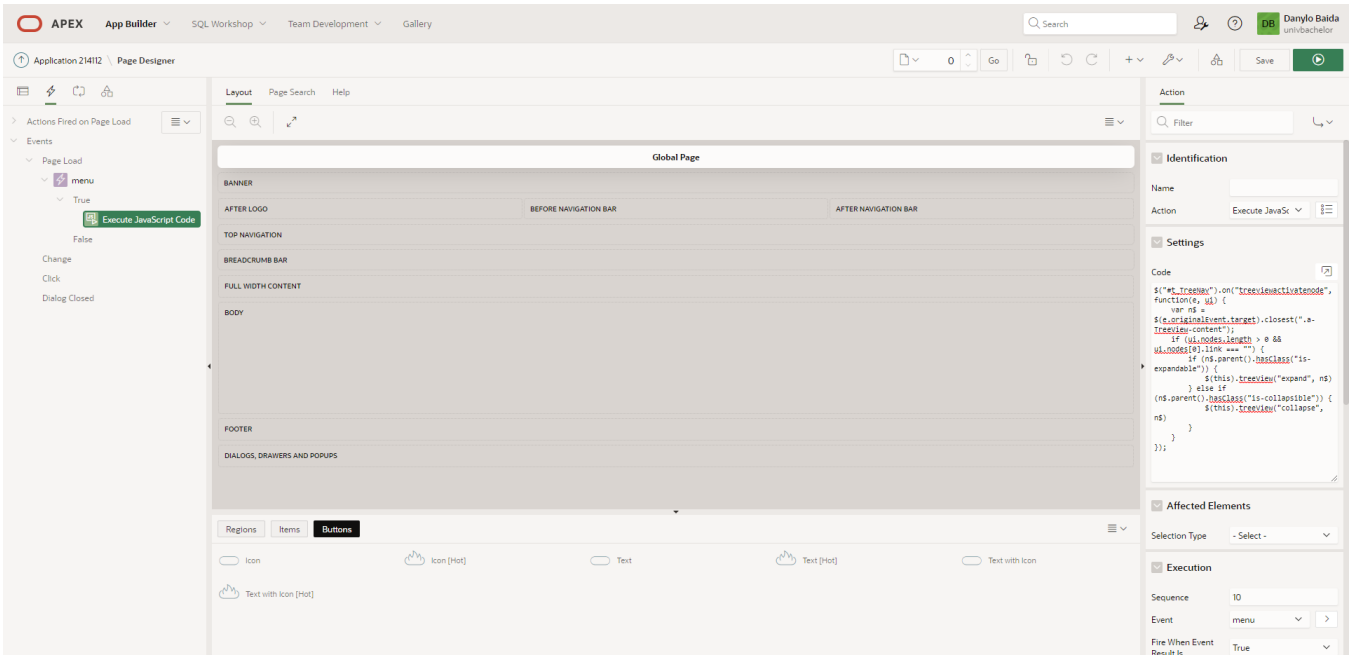

Рисунок 3.29 – Global Page у редакторі сторінок додатку KNU Scholarship

Таблиці в папці Кастомні таблиці були створені за для демонстрації створення нових таблиць з вже існуючих, через написання SQL скрипта адміністраторами додатку, а також для створення більш спеціалізованих, але часто вживаних таблиць. Код запитів знаходиться в додатку Ґ. Для реалізації одного з запитів було створено допоміжну функцію max_scholarship_by_group, код якої знаходиться в тому ж додатку.

При переході на будь яку з таблиць, користувач побачить приблизно таку сторінку, як представлено на рисунку 3.30. На ній показано сторінку із таблицею STUDENTS.

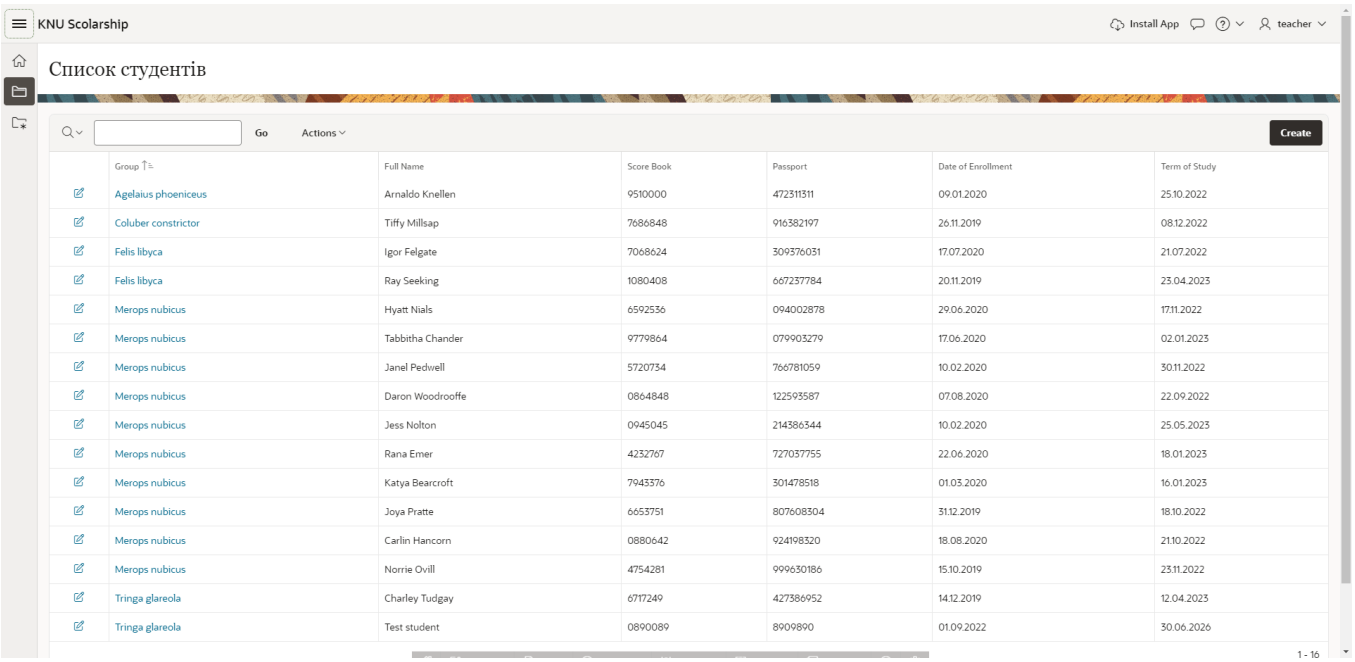

Рисунок 3.30 – Сторінка із таблицею STUDENTS

Ця сторінка дозволяє користувачу робити безліч дій із таблицею. Створення нових студентів, редагування даних вже наявних, пошук необхідного елементу, сортування, групування за ознаками, завантаження на пристрій користувача в декількох форматах. Таким чином, додаток забезпечує швидку та зручну роботу з даними. Користувач з правами Reader має всі ті самі можливості, крім створення та редагування даних в таблицях.

Коли ми у додаток заходить користувач з правами адміністратора, у навігаційному меню з’являються певні відмінності. З’являються два нових елементи, панель адміністратора, та таблиця користувачів (рисунок 3.31).

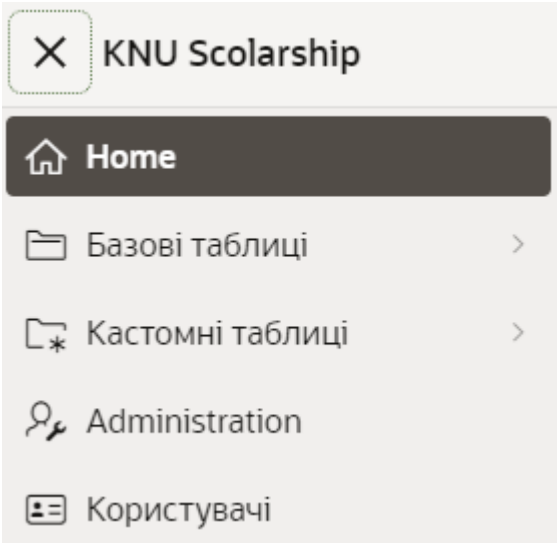

Рисунок 3.31 – Навігаційне меню KNU Scolarship з правами адміністратора

В таблиці користувачів, адміністратори можуть побачити всіх користувачів, зареєстрованих у додатку, незалежно від того чи надали їм певні рівні доступу та права, чи ні.

Панель адміністратора можна побачити на рисунку 3.32. В ній знаходяться безліч функцій, необхідних адміністраторам для керуванням додатком. Саме в пункті Access Control адміністратори прописують права, які надають користувачам. Також, тут знаходяться інструменти, які дозволяють графічно показати певні параметри та статистику додатку.

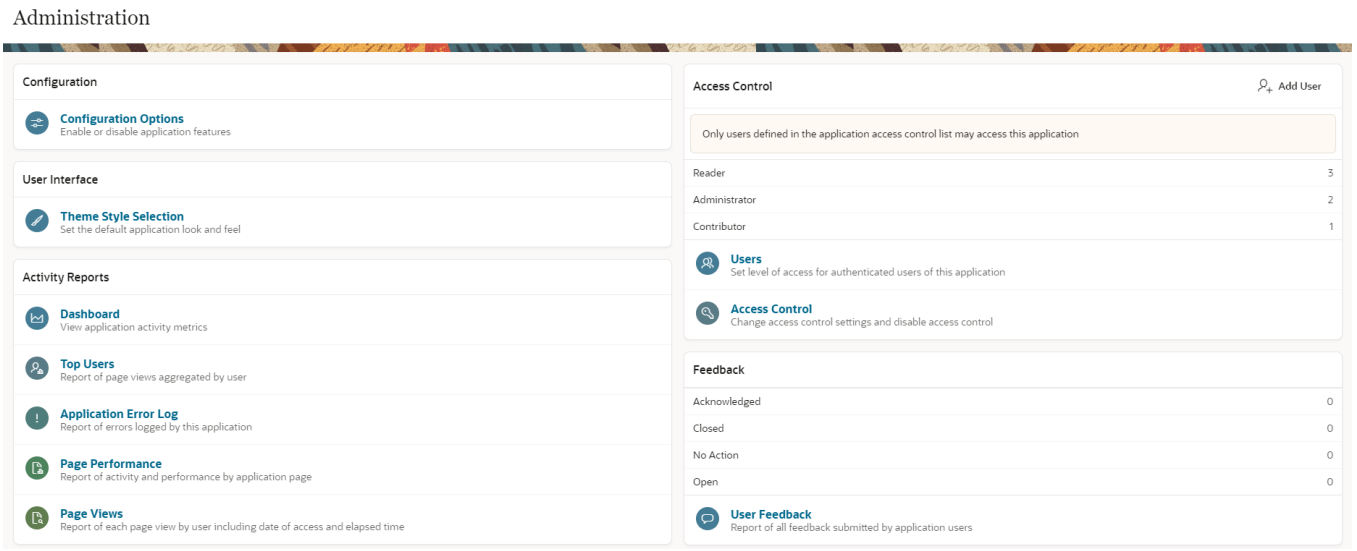

Рисунок 3.32 – Панель адміністратора

3.5 Аналіз функціонування інформаційної системи

Для аналізу функціонування інформаційної системи обліку нарахувань виплат стипендій студентам ВНЗ та побудованого програмного додатку, в Oracle APEX існує спеціальна сторінка із заголовком Dashboard (рисунок 3.33). Її також можна знайти в панелі адміністратора (рисунок 3.32). На цій сторінці ми можемо дізнатися повну статистику про будь-який елемент інформаційної системи: саму інформаційну систему в цілому, користувачів цієї системи, їх активність, зміни, внесені в систему, її продуктивність, програмний додаток та сторінки цього додатку, базу даних (рисунок 3.34-3.35). Інформацію з панелі адміністратора можна побачити на рисунку 3.36.

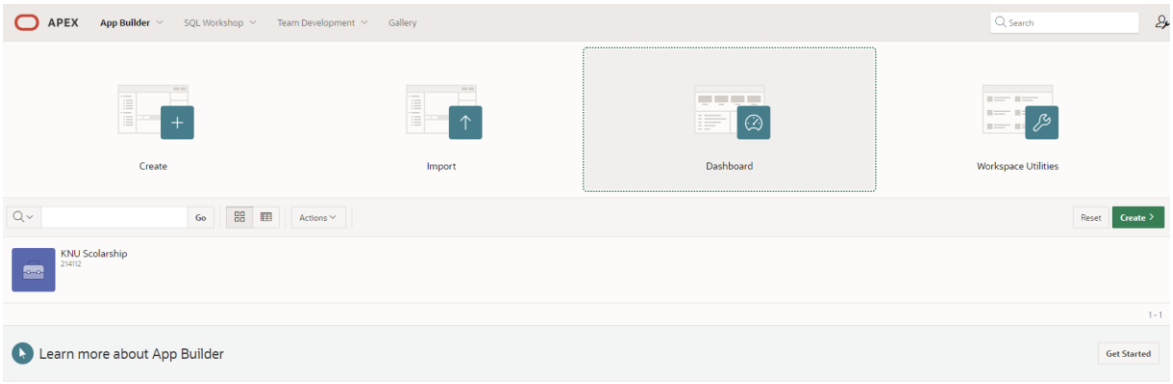

Рисунок 3.33 – Dashboard на сторінці створення додатків

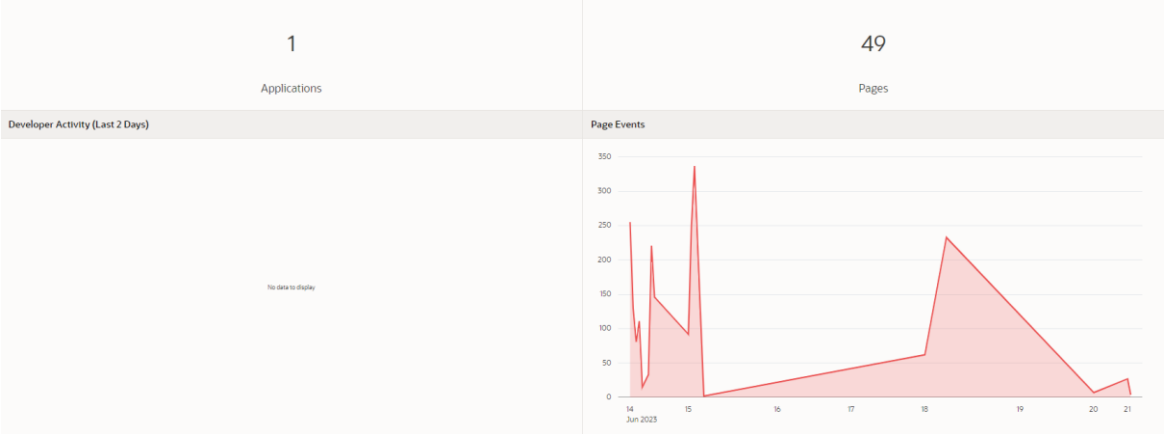

Рисунок 3.34 – Інформація на сторінці Dashboard

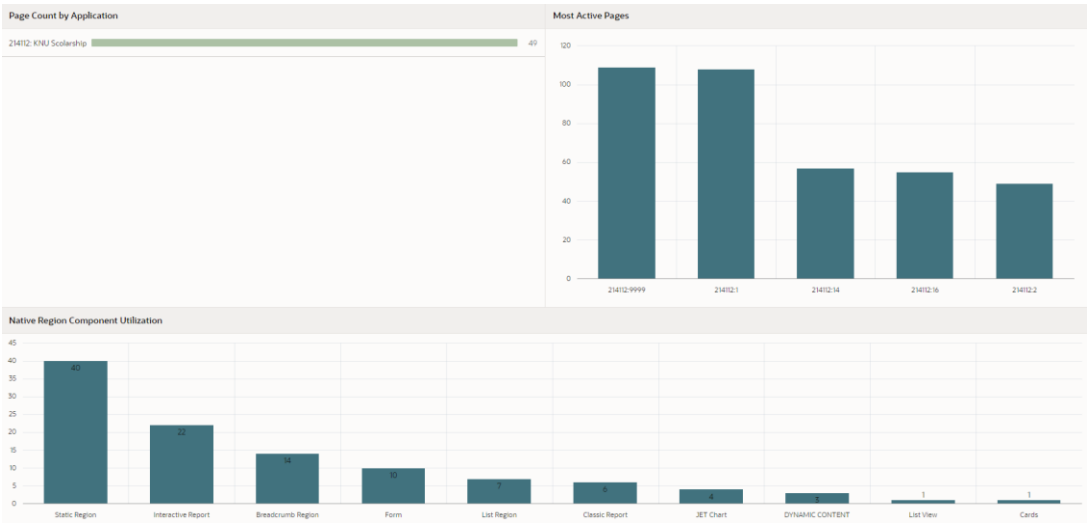

Рисунок 3.35 – Інформація на сторінці Dashboard

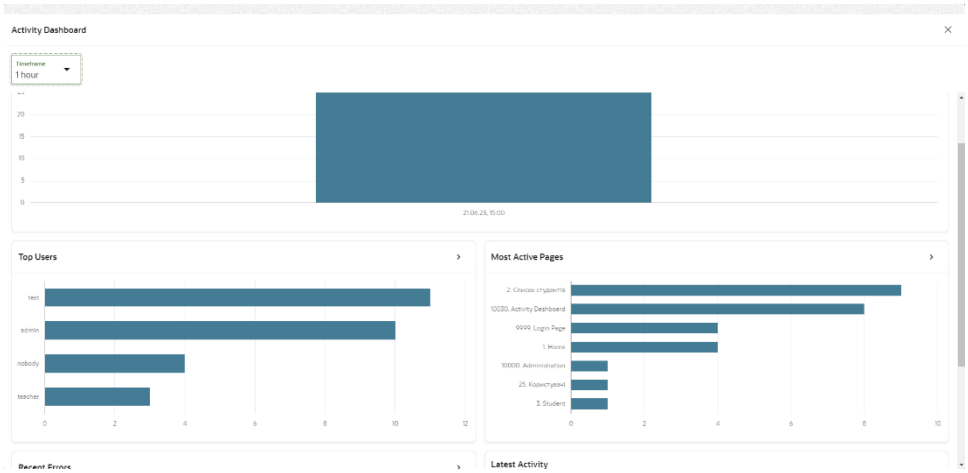

Рисунок 3.36 – Dashboard на панелі адміністратора

Проаналізувавши дану інформацію, не було виявлено жодних проблем в існуючій системі. Всі елементи працюють і не потребують змін.

ВИСНОВОК

1. В ході даного дипломного проекту було розроблено інформаційну систему обліку нарахувань стипендій студентам ВНЗ. У процесі розробки проекту було пройдено всі етапи класичної розробки інформаційних систем. Проаналізувавши завдання, було обрано оптимальні методи та шляхи розробки. Після цього, було створено спочатку ER-діаграму БД, а після неї – логічну та фізичну схему. Згідно до цієї схеми, було створено SQL код, який створив обрані таблиці, описав їх структуру та зв’язок. Для підготовки до наступного етапу був написаний код для заповнення створених таблиць згенерованими даними. Закінчивши цей етап, було створено запити, на які, згідно умов, система мала відповідати. Кінцевим етапом цього проекту було створення програмного додатку та інтерфейсу користувача.

2. Дослідження цільової аудиторії допомогло визначитися з кінцевим виглядом та наповненням додатку. Під час дослідження було виявлено ключові потреби та проблеми користувачів, що дозволило вигадати додаткові функції для покращення їх задоволеності від продукту.

3. Було проведено детальний аналіз всіх інструментів, доступних у Oracle.

4. Успішно розроблено прототип додатку, який є повністю функціональним. Він враховує потреби користувачів і може бути розширений та поліпшений у майбутньому. Прототип підтримує масштабованість, здатність працювати з різними обсягами даних та на різних пристроях. Крім того, було розроблено готову версію додатку, яка готова для впровадження кінцевим користувачам. Це означає, що додаток готовий для початку тестування і демонстрації його потенціалу перед потенційними користувачами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ресурс Diagrams.net, безкоштовне крос-платформне програмне забезпечення для створення графіків: https://app.diagrams.net/.
2. Технічна документація Oracle: https://docs.oracle.com/en/.
3. Онлайн застосунок DBDiagram, безкоштовний ресурс створення моделей даних: https://dbdiagram.io/.
4. Онлайн застосунок Mockaroo, безкоштовний генератор тестових даних: https://www.mockaroo.com/.
5. Безкоштовний ресурс для студентів, з навчальними матеріалами про дані та бази даних: https://databasetown.com/relational-database-benefits-and-limitations/.
6. Безкоштовний ресурс для студентів, з навчальними матеріалами по різним мовам програмування та базам даних: https://www.javatpoint.com/what-is-oracle.
7. Безкоштовний ресурс для студентів, з навчальними матеріалами по різним IT напрямам: https://www.guru99.com/data-modelling-conceptual-logical.html.
8. Безкоштовна онлайн енциклопедія: https://en.wikipedia.org/wiki/Relational_model#Database.
9. Онлайн енциклопедія: https://www.encyclopedia.com/science-and-technology/computers-and-electrical-engineering/computers-and-computing/databases.

ДОДАТКИ

Додаток А

Скрипт створення логічної та фізичної схеми БД за допомогою онлайн застосунку DBDiagram.

```
Table "scholarship_val" {
  "scholarship_id" INT [pk]
  "base_val" INT
  "increase_percent" INT
  "max_grade" INT
  "min_grade" INT
}

Table "scholarship" {
  "student_id" INT
  "semester_id" INT
  "scholarship_id" INT

Indexes {
  (student_id, semester_id) [pk]
}
}

Table "students" {
  "student_id" INT [pk]
  "group_id" INT
  "full_name" VARCHAR(40)
  "score_book" VARCHAR(40)
  "passport" VARCHAR(40)
  "date_of_enrollment" timestamp(0)
```

```
  "term_of_study" timestamp(0)
}

Table "semesters" {
  "semester_id" INT [pk]
  "start_date" timestamp(0)
  "end_date" timestamp(0)
}

Table "groups" {
  "group_id" INT [pk]
  "name" VARCHAR(40)
}

Table "marks" {
  "student_id" INT
  "subject_id" INT
  "mark" INT

Indexes {
  (student_id, subject_id) [pk]
}
}

Table "educ_plan" {
  "group_id" INT
  "subject_id" INT
  "semester_id" INT
```

```
Indexes {
  (group_id, subject_id, semester_id) [pk]
}
}

Table "subjects" {
  "subject_id" INT [pk]
  "name" VARCHAR(40)
}

Ref:"groups"."group_id" < "students"."group_id"
Ref:"students"."student_id" < "marks"."student_id"
Ref:"subjects"."subject_id" < "marks"."subject_id"
Ref:"students"."student_id" < "scholarship"."student_id"
Ref:"semesters"."semester_id" < "scholarship"."semester_id"
Ref:"scholarship_val"."scholarship_id" < "scholarship"."scholarship_id"
Ref:"groups"."group_id" < "educ_plan"."group_id"
Ref:"subjects"."subject_id" < "educ_plan"."subject_id"
Ref:"semesters"."semester_id" < "educ_plan"."semester_id"
```

Додаток Б

Скрипт для створення таблиць представлено в СУБД Oracle.

```
CREATE TABLE scholarship_val (
  SCHOLARSHIP_ID NUMBER(6,0),
  BASE_VAL NUMBER(6,0),
  INCREASE_PERCENT NUMBER(6,0),
  MAX_GRADE NUMBER(6,0),
  MIN_GRADE NUMBER(6,0),
  AFFECTIVE_FROM NUMBER(6,0),
  CONSTRAINT max_grade_MIN_MAX CHECK (MAX_GRADE
BETWEEN 0 AND 100) ENABLE,
  CONSTRAINT min_grade_MIN_MAX CHECK (MIN_GRADE BETWEEN
0 AND 100) ENABLE,
  CONSTRAINT scholarship_id_PK PRIMARY KEY (SCHOLARSHIP_ID)
ENABLE
);

CREATE TABLE scholarship (
  STUDENT_ID NUMBER(6,0),
  SEMESTER_ID NUMBER(6,0),
  SCHOLARSHIP_ID NUMBER(6,0),
  CONSTRAINT stud_sem_PK PRIMARY KEY (STUDENT_ID,
SEMESTER_ID) ENABLE
);

CREATE TABLE students (
  STUDENT_ID NUMBER(6,0),
  GROUP_ID NUMBER(6,0),
  FULL_NAME VARCHAR2(40),
```

```
  SCORE_BOOK VARCHAR(40),
  PASSPORT VARCHAR(40),
  DATE_OF_ENROLLMENT DATE,
  TERM_OF_STUDY DATE,
  CONSTRAINT student_id_PK PRIMARY KEY (student_id) ENABLE
);

CREATE TABLE semesters (
  SEMESTER_ID NUMBER(6,0),
  START_DATE DATE,
  END_DATE DATE,
  CONSTRAINT end_date_grater_then_start CHECK (END_DATE >=
START_DATE) ENABLE,
  CONSTRAINT semester_id PRIMARY KEY (SEMESTER_ID) ENABLE
);

CREATE TABLE groups (
  GROUP_ID NUMBER(6,0),
  NAME VARCHAR2(40),
  CONSTRAINT group_id_PK PRIMARY KEY (group_id) ENABLE
);

CREATE TABLE marks (
  STUDENT_ID NUMBER(6,0),
  SUBJECT_ID NUMBER(6,0),
  MARK NUMBER(6,0),
  CONSTRAINT mark_MIN_MAX CHECK (MARK BETWEEN 0 AND
100) ENABLE,
```

```
  CONSTRAINT stud_subj_PK PRIMARY KEY (STUDENT_ID,
SUBJECT_ID) USING INDEX ENABLE
);

CREATE TABLE educ_plan (
  GROUP_ID NUMBER(6,0),
  SUBJECT_ID NUMBER(6,0),
  SEMESTER_ID NUMBER(6,0),
  CONSTRAINT educ_PK PRIMARY KEY (GROUP_ID, SEMESTER_ID,
SUBJECT_ID) ENABLE
);

CREATE TABLE subjects (
  SUBJECT_ID NUMBER(6,0),
  NAME VARCHAR2(40),
  CONSTRAINT subject_id_PK PRIMARY KEY (SUBJECT_id) ENABLE
);

ALTER TABLE students ADD FOREIGN KEY (GROUP_ID)
REFERENCES groups (group_id) ENABLE;
ALTER TABLE marks ADD FOREIGN KEY (STUDENT_ID)
REFERENCES students (student_id) ENABLE;
ALTER TABLE marks ADD FOREIGN KEY (SUBJECT_ID)
REFERENCES subjects (subject_id) ENABLE;
ALTER TABLE scholarship ADD FOREIGN KEY (STUDENT_ID)
REFERENCES students (student_id) ENABLE;
ALTER TABLE scholarship ADD FOREIGN KEY (SEMESTER_ID)
REFERENCES semesters (semester_id) ENABLE;
```

```
ALTER TABLE scholarship ADD FOREIGN KEY (SCHOLARSHIP_ID)
REFERENCES scholarship_val (scholarship_id) ENABLE;
ALTER TABLE educ_plan ADD FOREIGN KEY (group_id) REFERENCES
groups (group_id) ENABLE;
ALTER TABLE educ_plan ADD FOREIGN KEY (subject_id)
REFERENCES subjects (subject_id) ENABLE;
ALTER TABLE educ_plan ADD FOREIGN KEY (semester_id)
REFERENCES semesters (semester_id) ENABLE;

CREATE INDEX student_name_index ON students (FULL_NAME);
CREATE INDEX student_group_index ON students (GROUP_ID);
CREATE INDEX group_name_index ON groups (NAME);
CREATE INDEX subject_name_index ON subjects (NAME);
CREATE INDEX semester_start_date_index ON semesters (START_DATE);
CREATE INDEX semesters_date_index ON semesters (START_DATE,
END_DATE);
```

Додаток В

Код для внесення даних, згенерованих за допомогою безкоштовного онлайн ресурсу Mockaroo.

```
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 1, 3);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 2, 8);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 3, 12);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 7, 14);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 4, 7);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 4, 13);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 6, 1);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 7, 2);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 7, 9);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (1, 7, 11);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (2, 2, 8);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (3, 5, 4);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (4, 1, 10);
```

```
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (5, 2, 15);
insert into EDUC_PLAN (GROUP_ID, SEMESTER_ID, SUBJECT_ID)
values (5, 6, 5);

insert into GROUPS (GROUP_ID, NAME) values (1, 'Merops nubicus');
insert into GROUPS (GROUP_ID, NAME) values (2, 'Agelaius phoeniceus');
insert into GROUPS (GROUP_ID, NAME) values (3, 'Coluber constrictor');
insert into GROUPS (GROUP_ID, NAME) values (4, 'Tringa glareola');
insert into GROUPS (GROUP_ID, NAME) values (5, 'Felis libyca');

insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (1, 15,
41);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (2, 1, 61);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (3, 4, 96);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (4, 3, 2);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (5, 6, 60);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (6, 5, 27);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (7, 8, 88);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (8, 9, 32);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (9, 11,
80);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (10, 7,
63);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (11, 10,
52);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (12, 12,
53);
```

```
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (13, 14, 60);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (14, 13, 95);
insert into MARKS (STUDENT_ID, SUBJECT_ID, MARK) values (15, 2, 1);

insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (1, 1, 3);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (2, 3, 2);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (3, 3, 1);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (4, 7, 6);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (5, 4, 5);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (6, 4, 15);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (7, 6, 10);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (8, 7, 9);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (9, 7, 8);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (10, 7, 7);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID, SCHOLARSHIP_ID) values (11, 2, 15);
```

```
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID,
SCHOLARSHIP_ID) values (12, 5, 11);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID,
SCHOLARSHIP_ID) values (13, 1, 13);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID,
SCHOLARSHIP_ID) values (14, 2, 12);
insert into SCHOLARSHIP (STUDENT_ID, SEMESTER_ID,
SCHOLARSHIP_ID) values (15, 6, 6);

insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (1, 82, 40, 91, 35, 2);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (2, 60, 30, 62, 55, 91);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (3, 22, 10, 31, 10, 83);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (4, 75, 40, 84, 9, 41);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (5, 3, 70, 92, 72, 14);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (6, 37, 70, 47, 2, 38);
```

```
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (7, 99, 50, 81, 34, 45);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (8, 12, 60, 95, 79, 93);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (9, 68, 40, 89, 11, 4);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (10, 3, 20, 20, 1, 41);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (11, 86, 50, 82, 2, 8);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (12, 1, 10, 75, 70, 10);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (13, 48, 20, 100, 90, 91);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (14, 67, 20, 90, 80, 3);
insert into SCHOLARSHIP_VAL (SCHOLARSHIP_ID, BASE_VAL,
INCREASE_PERCENT, MAX_GRADE, MIN_GRADE,
AFFECTIVE_FROM) values (15, 0, 0, 0, 0, 0);
```

```
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (1, '09/01/2019', '12/20/2019');
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (2, '01/15/2020', '06/20/2020');
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (3, '09/01/2020', '12/20/2020');
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (4, '01/15/2021', '06/20/2021');
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (5, '09/01/2021', '12/20/2021');
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (6, '01/15/2022', '06/20/2022');
insert into SEMESTERS (SEMESTER_ID, START_DATE, END_DATE)
values (7, '09/01/2022', '12/20/2022');

insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (1, 1, 'Carlin Hancorn', '0880642', '924198320',
'08/18/2020', '10/21/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (2, 1, 'Norrie Ovill', '4754281', '999630186',
'10/15/2019', '11/23/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (3, 1, 'Joya Pratte', '6653751', '807608304',
'12/31/2019', '10/18/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
```

```
TERM_OF_STUDY) values (4, 1, 'Katya Bearcroft', '7943376', '301478518',
'03/01/2020', '01/16/2023');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (5, 1, 'Rana Emer', '4232767', '727037755',
'06/22/2020', '01/18/2023');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (6, 1, 'Hyatt Nials', '6592536', '094002878',
'06/29/2020', '11/17/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (7, 1, 'Tabbitha Chander', '9779864', '079903279',
'06/17/2020', '01/02/2023');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (8, 1, 'Janel Pedwell', '5720734', '766781059',
'02/10/2020', '11/30/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (9, 1, 'Daron Woodrooffe', '0864848',
'122593587', '08/07/2020', '09/22/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (10, 1, 'Jess Nolton', '0945045', '214386344',
'02/10/2020', '05/25/2023');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
```

```
TERM_OF_STUDY) values (11, 2, 'Arnaldo Knellen', '9510582', '472311311',
'01/09/2020', '10/25/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (12, 3, 'Tiffy Millsap', '7686848', '916382197',
'11/26/2019', '12/08/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (13, 4, 'Charley Tudgay', '6717249', '427386952',
'12/14/2019', '04/12/2023');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (14, 5, 'Igor Felgate', '7068624', '309376031',
'07/17/2020', '07/21/2022');
insert into STUDENTS (STUDENT_ID, GROUP_ID, FULL_NAME,
SCORE_BOOK, PASSPORT, DATE_OF_ENROLLMENT,
TERM_OF_STUDY) values (15, 5, 'Ray Seeking', '1080408', '667237784',
'11/20/2019', '04/23/2023');
insert into SUBJECTS (SUBJECT_ID, NAME) values (1, 'Ukraine');
insert into SUBJECTS (SUBJECT_ID, NAME) values (2, 'Poland');
insert into SUBJECTS (SUBJECT_ID, NAME) values (3, 'Italy');
insert into SUBJECTS (SUBJECT_ID, NAME) values (4, 'Greece');
insert into SUBJECTS (SUBJECT_ID, NAME) values (5, 'Portugal');
insert into SUBJECTS (SUBJECT_ID, NAME) values (6, 'Philippines');
insert into SUBJECTS (SUBJECT_ID, NAME) values (7, 'South Africa');
insert into SUBJECTS (SUBJECT_ID, NAME) values (8, 'Argentina');
insert into SUBJECTS (SUBJECT_ID, NAME) values (9, 'Canada');
insert into SUBJECTS (SUBJECT_ID, NAME) values (10, 'Vietnam');
insert into SUBJECTS (SUBJECT_ID, NAME) values (11, 'Spain');
```

```
insert into SUBJECTS (SUBJECT_ID, NAME) values (12, 'Japan');
insert into SUBJECTS (SUBJECT_ID, NAME) values (13, 'Taiwan');
insert into SUBJECTS (SUBJECT_ID, NAME) values (14, 'Montenegro');
insert into SUBJECTS (SUBJECT_ID, NAME) values (15, 'Egypt');
```

Додаток Г

```
create or replace TRIGGER  "BI_LOGIN_USERS"
  before insert on "LOGIN_USERS"
  for each row
begin
  if :NEW."ID" is null then
    select "LOGIN_USERS_SEQ".nextval into :NEW."ID" from sys.dual;
  end if;
end;

create or replace TRIGGER  "ID_GROUPS"
  before insert on "GROUPS"
  for each row
begin
  if :NEW."GROUP_ID" is null then
    select "GROUPS_SEQ".nextval into :NEW."GROUP_ID" from sys.dual;
  end if;
end;

create or replace TRIGGER  "ID_SCHOLARSHIP_VAL"
  before insert on "SCHOLARSHIP_VAL"
  for each row
begin
  if :NEW."SCHOLARSHIP_ID" is null then
    select "SCHOLARSHIP_VAL_SEQ".nextval into
:NEW."SCHOLARSHIP_ID" from sys.dual;
  end if;
end;
```

```
create or replace TRIGGER  "ID_SEMESTERS"
  before insert on "SEMESTERS"
  for each row
begin
  if :NEW."SEMESTER_ID" is null then
    select "SEMESTERS_SEQ".nextval into :NEW."SEMESTER_ID" from
sys.dual;
  end if;
end;

create or replace TRIGGER  "ID_STUDENTS"
  before insert on "STUDENTS"
  for each row
begin
  if :NEW."STUDENT_ID" is null then
    select "STUDENTS_SEQ".nextval into :NEW."STUDENT_ID" from
sys.dual;
  end if;
end;

create or replace TRIGGER  "ID_SUBJECTS"
  before insert on "SUBJECTS"
  for each row
begin
  if :NEW."SUBJECT_ID" is null then
    select "SUBJECTS_SEQ".nextval into :NEW."SUBJECT_ID" from
sys.dual;
  end if;
end;
```

Додаток Ґ

Код запитів, що реалізують створення кастомних таблиць.

Таблиця Стипендія по студентам та семестрам:

```
SELECT s.STUDENT_ID, s.FULL_NAME, scv.BASE_VAL, sem.start_date,
sem.end_date FROM STUDENTS s
join SCHOLARSHIP sc on (s.student_id = sc.student_id)
join SCHOLARSHIP_VAL scv on (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
join SEMESTERS sem on (sem.semester_id = sc.semester_id)
WHERE (sem.start_date < SYSDATE) AND (sem.end_date > SYSDATE);
```

Таблиця Студенти без переривання стипендії:

```
SELECT s.STUDENT_ID, s.FULL_NAME, scv.BASE_VAL FROM
STUDENTS s
join SCHOLARSHIP sc on (s.student_id = sc.student_id)
join SCHOLARSHIP_VAL scv on (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
WHERE (scv.BASE_VAL != 0);
```

Таблиця Студенти з максимальною стипендією в групі:

Функція max_scholarship_by_group:

```
CREATE OR REPLACE FUNCTION max_scholarship_by_group(grp_id IN
NUMBER)
RETURN NUMBER
IS
scholarship NUMBER := 0;
BEGIN
SELECT MAX(scv.BASE_VAL) into scholarship FROM STUDENTS s
JOIN GROUPS g ON (s.GROUP_ID=g.GROUP_ID)
```

```
JOIN SCHOLARSHIP sc ON (sc.STUDENT_ID = s.STUDENT_ID)
JOIN SCHOLARSHIP_VAL scv ON (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
WHERE (g.GROUP_ID = grp_id)
GROUP BY g.GROUP_ID;
RETURN scholarship;
END max_scholarship_by_group;
```

Запит:

```
SELECT g.GROUP_ID, g.NAME, s.FULL_NAME, scv.BASE_VAL FROM
STUDENTS s
JOIN GROUPS g ON s.GROUP_ID=g.GROUP_ID
JOIN SCHOLARSHIP sc ON (sc.STUDENT_ID = s.STUDENT_ID)
JOIN SCHOLARSHIP_VAL scv ON (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
WHERE scv.BASE_VAL = max_scholarship_by_group(s.GROUP_ID);
```

Таблиця Група з найменшою кількістю студентів із стипендією:

```
SELECT g.GROUP_ID, g.NAME, COUNT(s.student_id)
CNT_OF_STUDENTS FROM STUDENTS s
join GROUPS g on (s.GROUP_ID = g.GROUP_ID)
join SCHOLARSHIP sc on (s.student_id = sc.student_id)
join SCHOLARSHIP_VAL scv on (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
join SEMESTERS sem on (sem.semester_id = sc.semester_id)
WHERE scv.BASE_VAL != 0
HAVING COUNT(s.student_id) = (SELECT MIN(COUNT(s.student_id))
FROM STUDENTS s
join GROUPS g on (s.GROUP_ID = g.GROUP_ID)
```

```
join SCHOLARSHIP sc on (s.student_id = sc.student_id)
join SCHOLARSHIP_VAL scv on (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
join SEMESTERS sem on (sem.semester_id = sc.semester_id)
WHERE scv.BASE_VAL != 0
GROUP BY g.GROUP_ID)
GROUP BY g.GROUP_ID, g.NAME;
```

Таблиця Середній бал за кожен семестр навчання в кожній групі:

```
SELECT g.GROUP_ID, g.NAME, AVG(m.MARK) AS AVG_MARK,
sm.START_DATE, sm.END_DATE
FROM STUDENTS s
JOIN GROUPS g ON (s.GROUP_ID=g.GROUP_ID)
JOIN SCHOLARSHIP sc ON (sc.STUDENT_ID = s.STUDENT_ID)
JOIN SCHOLARSHIP_VAL scv ON (scv.SCHOLARSHIP_ID =
sc.SCHOLARSHIP_ID)
JOIN SEMESTERS sm ON (sm.SEMESTER_ID = sc.SEMESTER_ID)
JOIN MARKS m ON (m.STUDENT_ID = s.STUDENT_ID)
GROUP BY g.GROUP_ID, g.NAME, sm.START_DATE, sm.END_DATE
ORDER BY g.GROUP_ID;
```

Додаток Д

Створення таблиці:

```
CREATE TABLE  LOGIN_USERS (
   ID NUMBER(6,0),
   USERNAME VARCHAR2(40),
   PASSWORD VARCHAR2(40),
   CONSTRAINT "LOGIN_USERS_PK" PRIMARY KEY ("ID")
   USING INDEX ENABLE
);
```

Функція для входу у додаток використовуючі дані зареєстрованого користувача:

```
create or replace FUNCTION  LOGIN_USERS_AUTH
(p_username in varchar2, p_password in varchar2)
return boolean
as
   user_check varchar2(1);
begin
   select 'x'
   into user_check
   from LOGIN_USERS
   where upper(USERNAME) = upper(p_username) and PASSWORD=
p_password;
```

```
    apex_util.set_authentication_result(0);
     return true;
exception when no_data_found then
    apex_util.set_authentication_result(4);
    return false;
end LOGIN_USERS_AUTH;
```

Додаток Е

```
$("#t_TreeNav").on("treeviewactivatenode", function(e, ui) {
    var n$ = $(e.originalEvent.target).closest(".a-TreeView-content");
    if (ui.nodes.length > 0 && ui.nodes[0].link === "") {
        if (n$.parent().hasClass("is-expandable")) {
            $(this).treeView("expand", n$)
        } else if (n$.parent().hasClass("is-collapsible")) {
            $(this).treeView("collapse", n$)
        }
    }
});
```