**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВИЙ ПРОЕКТ**

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: **Моніторингова система контролю успішності учнів навчального закладу**

**Студент**

**групи** КП-82 Глушко Анастасія Валеріївна

(підпис)

**Викладач**

**к.т.н, доцент кафедри**

**СПіСКС Петрашенко А.В.**

(підпис)

Київ – 2021

**АНОТАЦІЯ**

Даний курсовий проект створений для моніторингу успішності учнів навчального закладу. Даний документ призначений для звітності процесу розробки програмного забезпечення зазначеної вище системи. У документі викладена актуальність та проблематика аналізу великого обсягу даних, аналіз використаного інструментарію (опис мови програмування, порівняння використаних бібліотек та бази даних), описана структура бази даних (з графічною частиною), опис програмного забезпечення (загальний, опис модулів та основних алгоритмів роботи), аналіз функціонування засобів масштабування, та опис результатів аналізу всієї системи (з графічною частиною та фрагментами коду).

Як результат даного проекту стала інтерактивна аналітика досліджуваних даних (можна переглянути у Додатку А у вигляді графічної частини).

ЗМІСТ

[ЗМІСТ 3](#_Toc72958023)

[ВСТУП 4](#_Toc72958024)

[АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ 5](#_Toc72958025)

[СТРУКТУРА БАЗИ ДАНИХ 8](#_Toc72958026)

[ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9](#_Toc72958027)

[ЗАГАЛЬНА СТРУКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9](#_Toc72958028)

[ОПИС МОДУЛІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 10](#_Toc72958029)

[ОПИС ОСНОВНИХ АЛГОРИТМІВ РОБОТИ 11](#_Toc72958030)

[АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ 12](#_Toc72958031)

[ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ 13](#_Toc72958032)

[ВИСНОВОК 14](#_Toc72958033)

[ЛІТЕРАТУРА 15](#_Toc72958034)

[ДОДАТКИ 16](#_Toc72958035)

[A. ГРАФІЧНІ МАТЕРІАЛИ 16](#_Toc72958036)

[Б. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ 20](#_Toc72958037)

ВСТУП

В сучасному світі освіта є досить актуальною темою. Так як наука стрімко розвивається як і сучасні технології, то для будь-якої розвиненої країни однією із найголовніших цілей є забезпечити хорошу освіту для своїх громадян. Саме оцінки є важливим фактором, що показує якість освіти та рівень знань в учнів. Таким чином, аналізуючи інформацію про оцінки учнів навчальних закладів, можна робити висновки про ефективність навчального процесу. Також аналіз показників успішності є ключовим моментом розробки стратегії вдосконалення роботи навчальних закладів. Оскільки кількість навчальних закладів дуже велика, відповідно і даних для обробки багато, а, отже, існує проблема збереження та аналізу цих даних.

Даний проект створюється в навчальних цілях в рамках дисципліни «Бази даних». Метою розробки даного курсового проекту виконання комплексного завдання щодо створення програмного забезпечення – моніторингової системи.

Даний курсовий проект призначений для для моніторингу успішності учнів навчального закладу, а також виявлення закономірностей в появі тих чи інших даних. Також система буде видавати з мінімальним користувацьким інтерфейсом сортовані дані, розділені на категорії дані у числовому та графічному виді.

АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ

Використана **мова програмування Python 3.9(1).** Для даної курсової роботи було обрано мову програмування Python, тому що вона забезпечує ефективне виконання завдань системної інтеграції.Системна інтеграція в інженерії — поєднання компонентів підсистем в єдину систему та забезпечення роботи окремих підсистем як єдиної системи. системна інтеграція є процесом об'єднання різних обчислювальних систем і програмних застосунків фізично або функціонально(2).

Python відрізняється двома особливостями, корисними для системної інтеграції і, які не підтримуються багатьма іншими мовами. Одна - типи даних, явно розроблені для зв'язування (interfacing); інша - качина типізація в поєднанні з малим, але гнучким набором стандартних інтерфейсів.

Серед основних переваг мови Python можна назвати такі:

* Чистий синтаксис (для виділення блоків слід використовувати відступи);
* переносність програм (що властиве більшості інтерпретованих мов);
* стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
* можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
* стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
* зручний для розв'язання математичних проблем (має засоби роботи з компклесними числами може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);

відкритий код (можливість редагувати його іншими користувачами).

За базу даних було взято **MongoDB**(3). Це найпопулярніша серед нереляційних СКБД, документо-орієнтована система керування базами даних із відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Вибір даної СКБД обгрунтовується тим що вона забезпечує можливість отримання неприведених до норм даних та подальшу роботу із ними, просто кодується і управляється, швидко працює на множині машин не обмежена експлуатаційними технологіями.

**Бібліотеки**

* **matplotlib**— це бібліотека Python 2D, яка представляє числові дані у різноманітних форматах та інтерактивних середовищах на різних платформах. Також ця бібліотека - це математичне розширення NumPy. Він надає об'єктно-орієнтований API для вбудови ділянок у додатки, що використовують набір інструментів для загального графічного інтерфейсу, таких як Tkinter, wxPython, Qt або GTK+. Matplotlib може використовуватися в скриптах Python, оболонках Python та IPython, серверах веб-додатків та чотирьох графічних наборів інструментів для користувацького інтерфейсу. Бібліотека sciPy використовує matplotlib(4).
* **numpy** — математична бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. А також інструменти для інтеграції C/C++ і Fortran коду, корисну лінійну алгебру, перетворення Фур'є і можливості випадкових чисел. Крім очевидних наукових застосувань, NumPy також може бути використаний як ефективний багатовимірний контейнер загальних даних. Можна визначити довільні типи даних. Це дає змогу без проблем і швидкої інтеграції NumPy з різноманітними базами даних. NumPy має ліцензію за ліцензією BSD, що дозволяє повторно використовувати кілька обмежень (5).

СТРУКТУРА БАЗИ ДАНИХ

База даних оцінок була створена самостійно і є вміщує в собі необхідні для аналізу успішності дані. Вона має одну таблицю marks з усіма даними про оцінку. Складається з наступних полів:

* id – ідентифікатор запису, унікальне поле у базі даних, первинний ключ. Тип даних - ObjectId;
* subject – назва предмету. Тип даних - String;
* student\_name – ім’я учня. Тип даних - String;
* mark – безпосередньо бал. Тип даних - Integer;
* max\_mark – максимальний бал, який можна отримати. Тип даних - Integer;
* grade – номер класу. Тип даних - Integer;
* specialty – спеціальність класу. Тип даних - String;
* date – дата виставлення оцінки. Тип даних - Datatime;

Графічне представлення бази даних можна побачити на рис. 1 (додаток А).

ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

# ЗАГАЛЬНА СТРУКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення моніторингової системи відстеження та аналізу оцінок в навчальному закладі має консольний інтерфейс, що значно спрощує структуру програми.

Структура даного програмного забезпечення складається з 7 файлів, кожен з яких відповідає за важливу функцію даної системи.

* **analyzer.py –** містить декілька алгоритмів аналізу сутностей даної бази даних, таких як середня ціна, коефіцієнт кореляції і т. д.;
* **database.py –** описує клас, що підключається та взаємодіє з базою даних;
* **generator.py –** описує функції генерації даних для заповнення бази даних;
* **marks.py –** описує структуру, що зберігаються у базі даних та алгоритми роботи із структурою;
* **graphics\_builder.py –** описує алгоритми аналізу бази даних і побудови на основі отриманих даних графіків та їх виведення в інтерактивному режимі;
* **main.py –** дає консольний інтерфейс керування і редагування бази даних, а також виводу всієї інформації;

ОПИС МОДУЛІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У програмі можна виділити декілька основних логічних модулів, що мають певну самостійність і обмінюються один з одним даними. Модулям необов’язково відповідають конкретні класи чи ієрархії класів. У програмі є 4 основні модулі: модуль генерації даних, модуль керування, модуль роботи з графіками та модуль бази даних. Деякі з них містять вкладені модулі. Взаємодія між модулями відбувається за допомогою виклику методів.

* **Модуль бази даних** містить функції, що виконують запис та зчитку даних, зберігання, модифікацію даних, видача результатів пошуку. Всі модулі програми взаємодіють з модулем бази даних. Це така собі обгортка для інтерфейсу MongoDB, створена саме під задачі даної системи.
* **Модуль керування** взаємодіє з усіма іншими, зокрема з модулем бази даних і містить об’єкт його класу. Його функції полягають у координуванні роботи всієї програми та надання інтерфейсу до всіх модулів.
* **Модуль генерації даних** містить список предметів, спеціальностей та алгоритм створення правдоподібних даних в залежності від класу.
* **Модуль роботи з графіками** реалізовує вивід графіків аналітики в інтерактивному вигляді за допомогою бібліотек numpy та matplotlib.

ОПИС ОСНОВНИХ АЛГОРИТМІВ РОБОТИ

* **Алгоритм побудови графіків** було реалізовано за допомогою потужного математичного пакету matplotlib. Найчастіше це перший пакет, пов'язаний з візуалізацією у Python. Спочатку задаємо назву графіку, потім встановлюємо панель управління, задаємо за які атрибути відповідають X та Y, підписуємо виведені дані та виводимо графік. Всю внутрішню реалізацію містять у собі функції математичного пакету matplotlib.
* **Алгоритм генерації випадкових даних для заповнення бази даних** було реалізовано за допомогою пакету names – для генерації випадкових імен та random – вибору випадкових даних із заданої множини даних.

АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ

Масштабування(6) - можливість збільшити продуктивність проекту за мінімальний час шляхом додавання ресурсів. Зазвичай масштабування мається на увазі не переписування коду, а або додавання серверів, або нарощування ресурсів існуючого. З цього типу виділяють вертикальну та горизонтальну масштабування. Вертикальне - це коли додають більше оперативної пам’яті, дисків і т.д. на вже існуючий сервер, а горизонтальне - це коли ставлять більше серверів в дата-центри, і сервери там взаємодіють.

В базі даних MongoDB, не виникає проблем з горизонтальним масштабуванням(7), що і необхідно було у даному курсовому проекті для створення засобів резервування та відновлення даних, призначених для оперативного пакетного збереження фрагментів усієї бази даних з можливістю її відновлення з урахуванням необхідності підключення додаткового комп’ютера як елемента горизонтального масштабування (тобто додаткового сервера).

ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ

У додатку А представлено 7 графіків демонстрації результатів аналізу моніторингової системи контролю успішності учнів навчального закладу.

На рис. 2 представлено графік залежності успішності від предмету. З аналізу бачимо, що середній бал з предметів становить близько 8-9.

На рис. 3 представлено графік залежності успішності від предмету для певного класу. З аналізу бачимо, що найнижчий середній бал у учнів 9го класу з предмету Німецька мова.

На рис. 4 представлено графік залежності успішності від предмету для певної спеціальності. З графіку бачимо, найнижчий показник успішності за гуманітарною спеціалізацією з предмету Етика.

На рис. 5 представлено графік залежності успішності від спеціальності. З аналізу бачимо, що найнижчий показник успішності з предмету Англійська мова у класах з економічною спеціалізацією.

На рис. 6 представлено графік залежності успішності від класу. З аналізу бачимо, що найнижчий показник успішності з предмету Українська мова спостерігається у 5 класі.

На рис. 7 представлена кореляція успішності та спеціальності. З аналізу бачимо, що коефіцієнт кореляції малий, отже, кореляція слабка, а це означає, що показник успішності з предмету Англійська мова не залежить від спеціальності класів.

На рис. 8 представлена регресія залежності успішності та вікової категорії учнів. З аналізу бачимо невелике пониження успішності з предмету Література з ростом вікової категорії учнів.

ВИСНОВОК

Була успішно опрацьована відповідна технічна література для успішного написання курсового проекту (перелік літератури наведено нижче). В ході розробки даного програмного забезпечення було розроблено засіб генерації даних, що генерує псевдовипадкові дані, за допомогою вищезазначених бібліотек були додані засоби валідації та фільтрації даних з достатньо великою швидкістю. Були додані засоби аналізу даних та графічне представлення результатів за допомогою існуючих математичних пакетів мови Python. Програмне забезпечення було успішно протестовано на декількох комп’ютерах, швидкість роботи бази даних достатньо велика. Також було реалізовано реплікації бази даних.

В ході виконання даного курсового проекту було досягнуто поставленої мети: було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з базами даних, а також були здобуті навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації.

ЛІТЕРАТУРА

1. Python – Вікіпедія [Електронний ресурс]:

<https://uk.wikipedia.org/wiki/Python>;

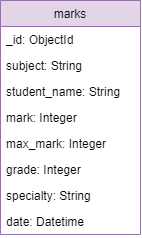
1. Почему Python так хорош в научных вычислениях [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://habr.com/post/349482/>;
2. MongoDB – Вікіпедія [Електронний ресурс]:

<https://uk.wikipedia.org/wiki/MongoDB>:

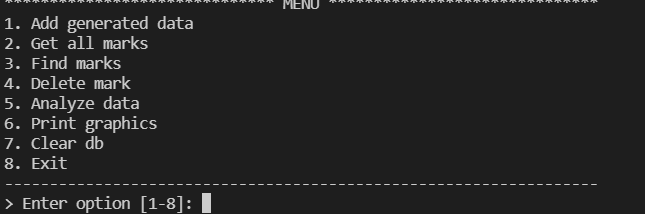
1. Matplotlib [Електронний ресурс]. – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Matplotlib>;
2. NumPy [Електронний ресурс]. – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/NumPy>;
3. Горизонтальное масштабировнаие: когда и как? [Електронний ресурс]. – Режим доступу до ресурсу: <https://habr.com/company/oleg-bunin/blog/319526>;
4. 5 причин використовувати Монго - [Електронний ресурс]: <http://echo.lviv.ua/dev/9693>

ДОДАТКИ

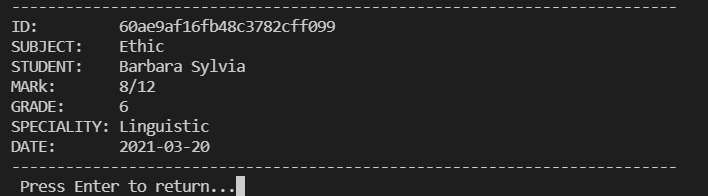
# ГРАФІЧНІ МАТЕРІАЛИ



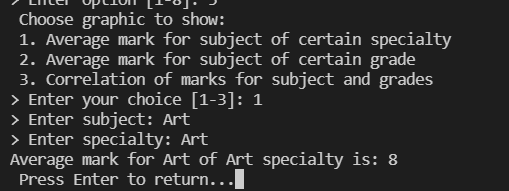
*Рис. 1 – ER-діаграма бази даних*



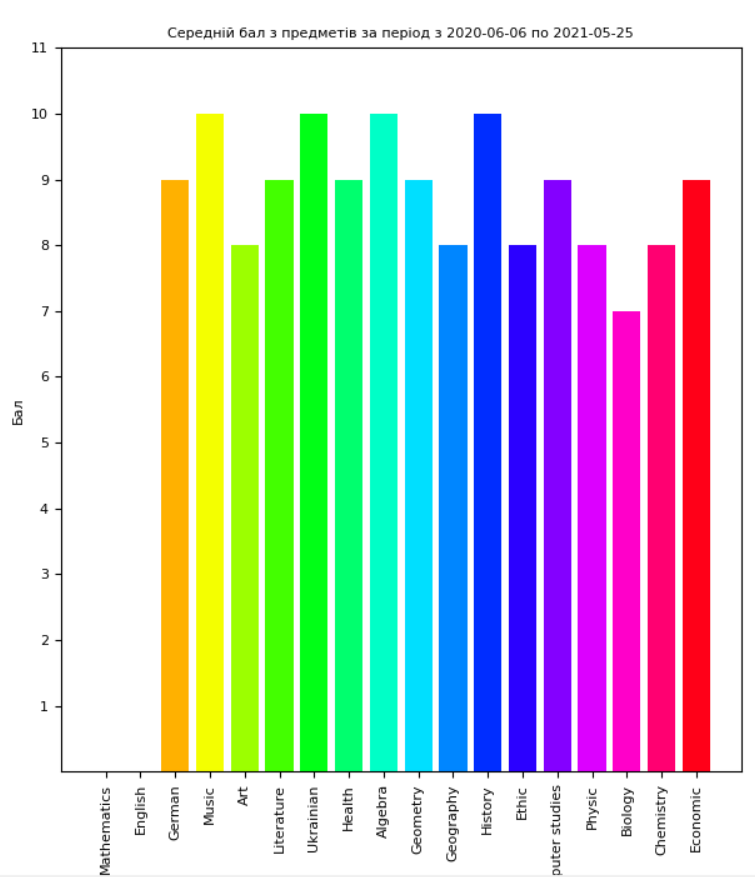
*Рис.2 – Головне меню*



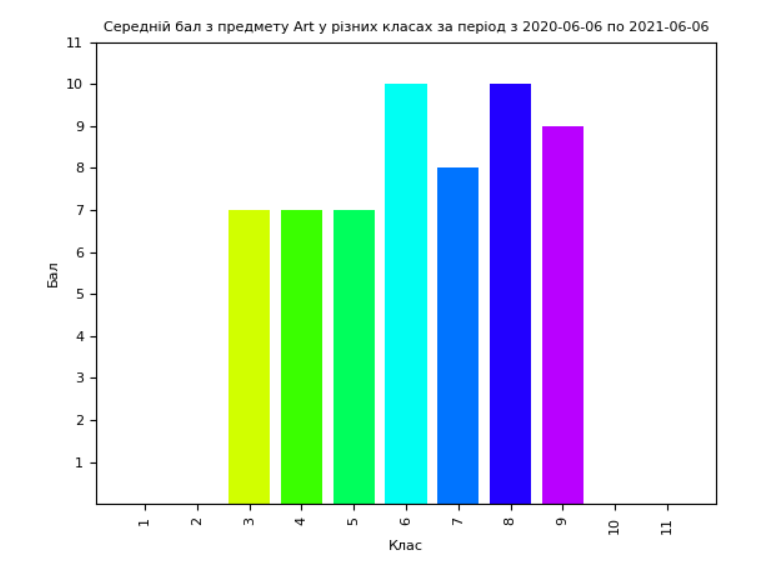
*Рис.3 – Вивід даних*



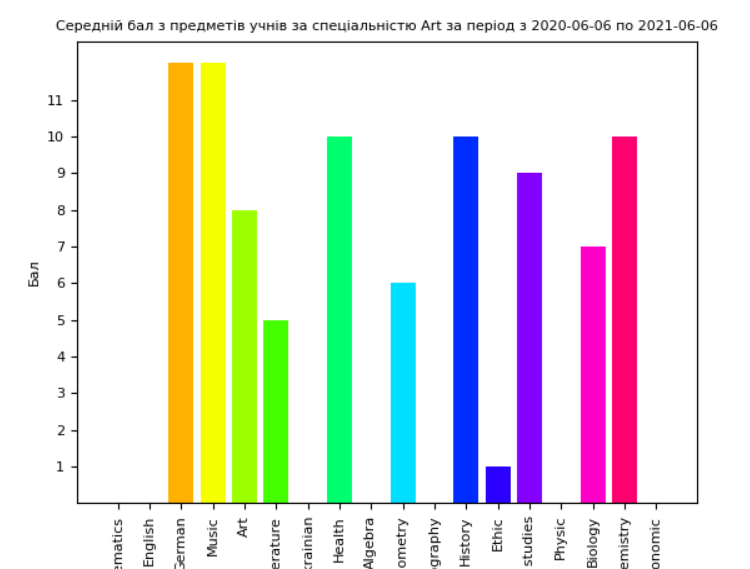
*Рис.4 – Аналіз середніх оцінок з предмета за спеціальністю*



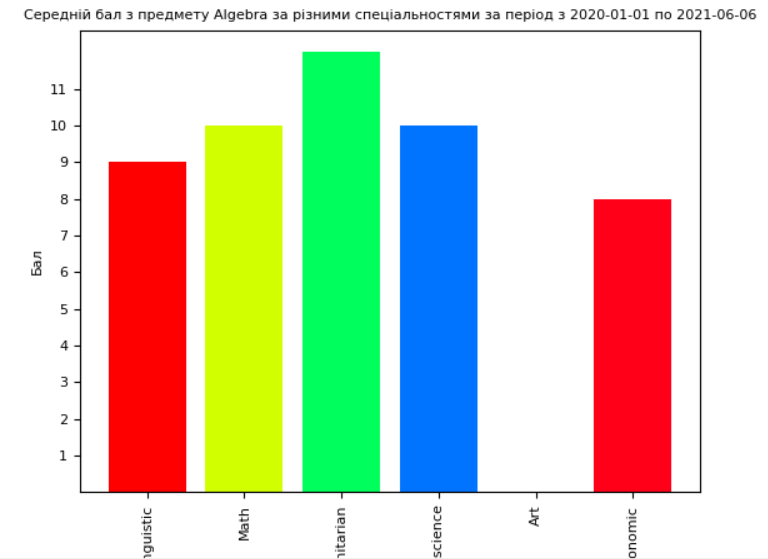
*Рис. 5 – Графік залежності успішності від предмету*



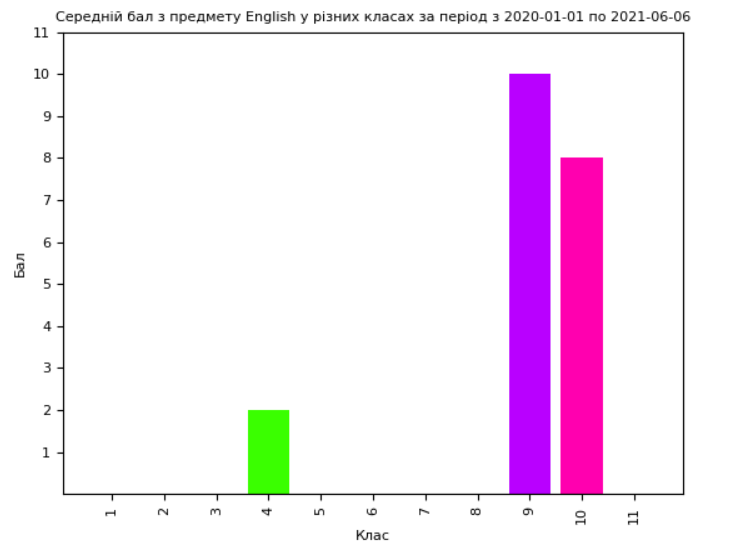
*Рис. 6 – Графік успішності предмету для певного класу*



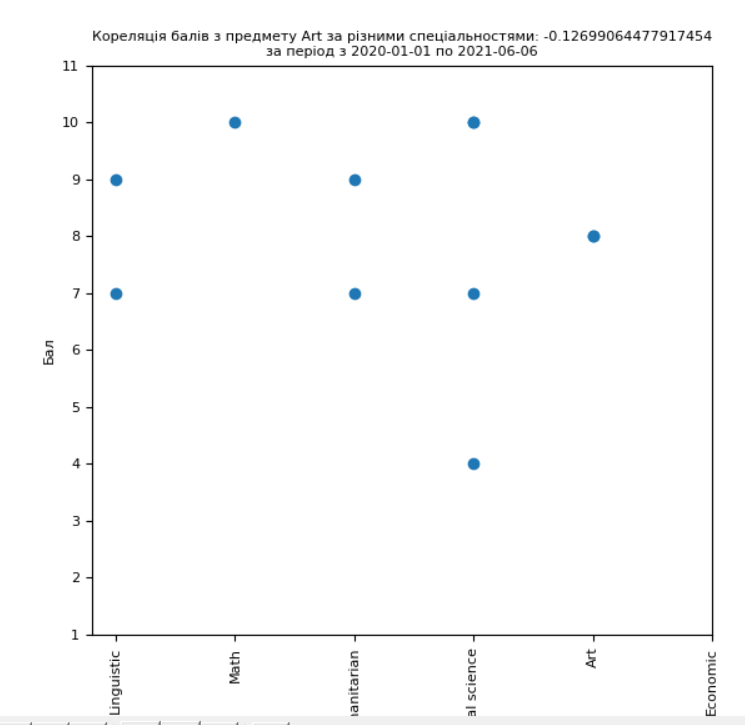
*Рис. 7 – Графік залежності успішності від предмету для певної спеціальності*



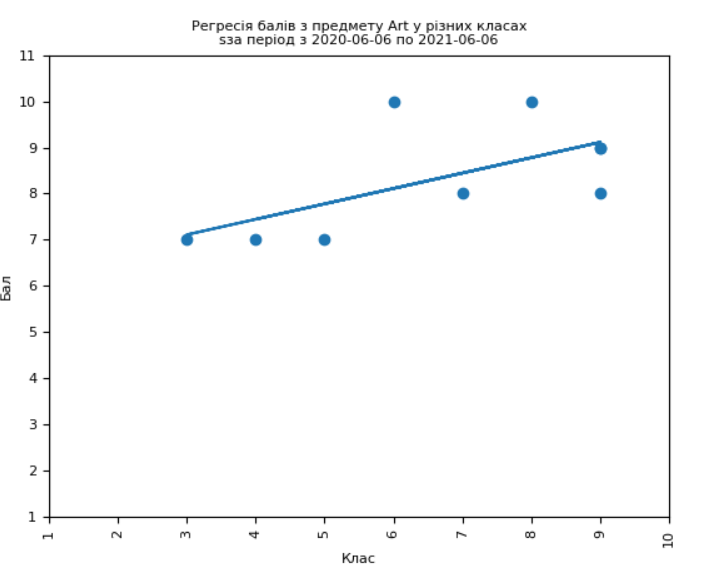
*Рис. 8 – Графік залежності успішності від спеціальності*



*Рис. 9 – Графік залежності успішності від класу*



*Рис. 10 – Кореляція успішності та спеціальності*



*Рис. 11 – Регресія залежності успішності та вікової категорії учнів*

# Б. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

|  |
| --- |
| generator.py |
| from marks import Marks  from database import Database  import names  import datetime  import random  import string  class Generator:  start\_subjects = ['Mathematics', 'English', 'German', 'Music', 'Art', 'Literature',  'Ukrainian', 'Health']  mid\_subjects = ['English', 'Algebra', 'Geometry', 'Geography', 'History',  'German', 'Music', 'Art', 'Ethic', 'Literature',  'Computer studies', 'Ukrainian', 'Health']  top\_subjects = ['English', 'Algebra', 'Geometry', 'Physic', 'Geography', 'History',  'German', 'Biology', 'Chemistry', 'Literature', 'Computer studies', 'Ukrainian']  specialty = ['English', 'Math', 'Biology', 'Chemistry', 'Art']  @staticmethod  def get\_all\_subjects():  subjects\_list = Generator.start\_subjects  for subject in Generator.mid\_subjects:  if subjects\_list.count(subject) == 0:  subjects\_list.append(subject)  for subject in Generator.top\_subjects:  if subjects\_list.count(subject) == 0:  subjects\_list.append(subject)  return subjects\_list  @staticmethod  def generate\_doc():  student\_name = names.get\_full\_name()  subject = ''  grade = random.randint(1, 11)  if grade <= 4:  subject = random.choice(Generator.start\_subjects)  elif grade <= 9:  subject = random.choice(Generator.mid\_subjects)  else:  subject = random.choice(Generator.top\_subjects)  max\_mark = 12  mark = 0  if random.randint(0, 3) < 3:  mark = random.randint(7, 10)  elif random.randint(0, 2) > 0:  mark = random.randint(11, 12)  else:  mark = random.randint(1, 6)  specialty = random.choice(Generator.specialty)  d1 = datetime.datetime(2020, 1, 1, 0, 0, 0, 0)  d2 = datetime.datetime.now()  delta = d2 - d1  int\_delta = (delta.days \* 24 \* 60 \* 60) + delta.seconds  random\_second = random.randrange(int\_delta)  date = d1 + datetime.timedelta(seconds=random\_second)  marks = Marks(subject, student\_name, mark, max\_mark, grade, specialty, date)  return marks  @staticmethod  def generate\_marks(count):  marks\_list = list()  for i in range(count):  mark = Generator.generate\_doc()  marks\_list.append(mark)  return marks\_list |

|  |
| --- |
| database.py |
| from pymongo import MongoClient  from bson.objectid import ObjectId  import datetime  class Database:  def \_\_init\_\_(self):  self.client = MongoClient('localhost', 27017)  self.db = self.client['marks\_db']  self.collection = self.db['marks']  def insert(self, marks):  self.collection.insert\_one({'subject': marks.subject, 'student\_name': marks.student\_name, 'mark': marks.mark,  'max\_mark': marks.max\_mark, 'grade': marks.grade, 'specialty': marks.specialty,  'date': marks.date})  def insert\_list(self, marks\_list):  for mark in marks\_list:  self.insert(mark)  def get\_all(self):  return self.collection.find()  def get\_by\_grade(self, grade):  return self.collection.find({'grade': grade})  def get\_by\_dates(self, date1, date2):  return self.collection.find({'date': {'$gte': datetime.datetime.strptime(date1, '%Y-%m-%d %H:%M:%S'),  '$lt': datetime.datetime.strptime(date2, '%Y-%m-%d %H:%M:%S')}})  def get\_by\_subject(self, subject):  return self.collection.find({'subject': subject})  def get\_by\_id(self, \_id):  return self.collection.find({'\_id': ObjectId(\_id)})  def delete(self, \_id):  return self.collection.delete\_one({'\_id': ObjectId(\_id)})  def delete\_all(self):  return self.collection.delete\_many({})  def get\_by\_subject\_and\_dates(self, subject, date1, date2):  return self.collection.find({'subject': subject,  'date': {'$gte': datetime.datetime.strptime(date1, '%Y-%m-%d %H:%M:%S'),  '$lt': datetime.datetime.strptime(date2, '%Y-%m-%d %H:%M:%S')}})  def get\_by\_grade\_and\_dates(self, grade, date1, date2):  return self.collection.find({'grade': grade,  'date': {'$gte': datetime.datetime.strptime(date1, '%Y-%m-%d %H:%M:%S'),  '$lt': datetime.datetime.strptime(date2, '%Y-%m-%d %H:%M:%S')}})  def get\_by\_specialty\_and\_dates(self, specialty, date1, date2):  return self.collection.find({'specialty': specialty,  'date': {'$gte': datetime.datetime.strptime(date1, '%Y-%m-%d %H:%M:%S'),  '$lt': datetime.datetime.strptime(date2, '%Y-%m-%d %H:%M:%S')}})  def get\_by\_subject\_and\_specialty(self, subject, specialty):  return self.collection.find({'subject': subject,  'specialty': specialty})  def get\_by\_subject\_and\_grade(self, subject, grade):  return self.collection.find({'subject': subject,  'grade': grade})  def get\_unique\_subjects(self):  mark\_list = self.collection.find()  subjects\_list = list()  for mark in mark\_list:  if subjects\_list.count(mark.get('subject')) == 0:  subjects\_list.append(mark.get('subject'))  return subjects\_list |

|  |
| --- |
| graphic\_builder.py |
| import numpy as np  import matplotlib.pyplot as plt  from analyzer import Analyzer  from generator import Generator  class GraphicsBuilder:  PLOT\_LABEL\_FONT\_SIZE = 8  @staticmethod  def get\_colors(n):  colors = []  cm = plt.cm.get\_cmap('hsv', n)  for i in np.arange(n):  colors.append(cm(i))  return colors  @staticmethod  def dict\_sort(\_keys, \_values):  keys = []  values = []  my\_dict = dict.fromkeys(\_keys, \_values)  my\_dict = sorted(my\_dict.items(), key=lambda x: x[1])  for k, v in my\_dict:  keys.append(k)  values.append(v)  return keys, values  def create\_graph(self, all\_marks, keys, key\_name, title):  values = []  for key in keys:  marks = [mark for mark in all\_marks if mark.get(key\_name) == key]  values.append(Analyzer.average\_mark(marks))  top\_keys = len(keys)  plt.title(title, fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.bar(np.arange(top\_keys), values, color=self.get\_colors(top\_keys))  plt.xticks(np.arange(top\_keys), keys, rotation=90, fontsize=8)  plt.yticks(fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.ylabel('Бал', fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.show()  def average\_mark\_graph(self, db, date1, date2):  GraphicsBuilder.create\_graph(self, list(db.get\_by\_dates(date1, date2)),  Generator.get\_all\_subjects(),  'subject',  'Середній бал з предметів за період з ' + str(date1) + ' по ' + str(date2))  def average\_mark\_by\_grade\_graph(self, db, date1, date2, grade):  GraphicsBuilder.create\_graph(self, list(db.get\_by\_grade\_and\_dates(grade, date1, date2)),  Generator.get\_all\_subjects(),  'subject',  'Середній бал з предметів учнів ' + str(grade) + ' класу')  def average\_mark\_by\_specialty\_graph(self, db, date1, date2, specialty):  GraphicsBuilder.create\_graph(self, list(db.get\_by\_specialty\_and\_dates(specialty, date1, date2)),  Generator.get\_all\_subjects(),  'subject',  'Середній бал з предметів учнів за спеціальністю ' + specialty)  def average\_mark\_by\_subject\_order\_by\_specialty\_graph(self, db, date1, date2, subject):  GraphicsBuilder.create\_graph(self, list(db.get\_by\_subject\_and\_dates(subject, date1, date2)),  Generator.specialty,  'specialty',  'Середній бал з предмету ' + subject + ' за різними спеціальностями')  def average\_mark\_by\_subject\_order\_by\_grade\_graph(self, db, date1, date2, subject):  GraphicsBuilder.create\_graph(self, list(db.get\_by\_subject\_and\_dates(subject, date1, date2)),  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],  'grade',  'Середній бал з предмету ' + subject + ' у різних класах')  def correlation\_mark\_for\_subject\_and\_specialty\_graph(self, db, date1, date2, subject):  all\_marks = list(db.get\_by\_subject\_and\_dates(subject, date1, date2))  specialities = Generator.specialty  keys = []  values = []  for mark in all\_marks:  keys.append(specialities.index(mark.get('specialty')))  values.append(mark.get('mark'))  plt.title('Кореляція балів з предмету ' + subject + ' за різними спеціальностями: ' + Analyzer.correlation\_coef(keys, values),  fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.scatter(keys, values)  x = np.array(keys)  y = np.array(values)  k, b = np.polyfit(x, y, 1)  plt.plot(x, k \* x + b)  plt.xticks(np.arange(len(specialities)), specialities, rotation=90, fontsize=8)  plt.yticks(fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.ylabel('Бал', fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.show()  def regression\_mark\_for\_subject\_and\_grade\_graph(self, db, date1, date2, subject):  all\_marks = list(db.get\_by\_subject\_and\_dates(subject, date1, date2))  keys = []  values = []  for mark in all\_marks:  keys.append(mark.get('grade'))  values.append(mark.get('mark'))  plt.title('Регресія балів з предмету ' + subject + ' у різних класах', fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.scatter(keys, values)  x = np.array(keys)  y = np.array(values)  k, b = np.polyfit(x, y, 1) # 2020-01-01 00:00:00  plt.plot(x, k \* x + b)  plt.xticks(rotation=90, fontsize=8)  plt.yticks(fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.ylabel('Бал', fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.xlabel('Клас', fontsize=self.PLOT\_LABEL\_FONT\_SIZE)  plt.show() |