НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КУРСОВА РОБОТА

з дисципліни «Інженерія програмного забезпечення»

на тему: «Аналіз успішності студентів учбового закладу»

Студента 2 курсу групи ІО-61

напряму підготовки: 123 - Комп’ютерна інженерія

номер залікової 6103

Валько Антон Вікторович

Керівник старший викладач, к.т.н., с.н.с.

Антонюк Андрій Іванович

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2018 рік

**ЗМІСТ**

ЗМІСТ….………………………………………………………………………….......2

ВСТУП……………………………………………………………………………......3

1. ОГЛЯД MVC

1.1 Загальна характеристика

1.2 Структура MVC

1.3 Шаблони програмування, що використовуються в MVC

1.4 Завдання MVC

1.5 Python

1.6 Використання концепції MVC в даній роботі

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ……………………..………x

* 1. Прецеденти
  2. Ескізи графічного інтерфейсу
  3. Таблиця відповідності елементів бібліотеці Tkinter

3. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Діаграма класів

3.2 Опис моделі

3.3 Опис View

3.4 Опис класів, які реалізують логіку та працездатність програми

4. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАПЕЗПЕЧЕННЯ

ВИСНОВКИ

ЛІТЕРАТУРА

ДОДАТКИ

ДОДАТОК 1: ПРОГРАМНИЙ КОД ПРОЕКТУ

**Вступ**

Об’єктом розробки даної курсової роботи є програма з функціональністю аналізу успішності студентів учбового закладу.

Метою курсової роботи є закріплення теоретичних знань і практичних навичок з проектування, моделювання, розробки та тестування програмного забезпечення з графічним інтерфейсом.

Результат курсової роботи – готовий програмний додаток з графічним інтерфейсом, написаний на мові програмування Python.

Графічний інтерфейс користувача – інтерфейс між комп’ютером та його користувачем, реалізований за допомогою бібліотеки Tkinter.

1. **ОГЛЯД MVC**
   1. *Загальна характеристика*

Model – view – controller (MVC, «модель – представлення – поведінка») – схема використання кількох шаблонів проектування, за допомогою яких модель даних програми, інтерфейс користувача і взаємодія з користувачем розділені на три окремих компонента таким чином, щоб модифікація одного з компонентів впливала на інші мінімально. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

* 1. *Структура MVC*

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

* Модель (Model) є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.
* Вигляд (View) може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.
* Контролер (Controller) одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних. Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

* 1. *Шаблони програмування, що використовуються в MVC*

Для реалізації схеми MVC використовується досить велика кількість шаблонів проектування (залежно від складності архітектурного рішення), основні з яких «спостерігач», «стратегія» та «компонувальник».

Найбільш типова реалізація відокремлює представлення від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій (підписка/сповіщення). При кожній зміні внутрішніх даних в моделі вона оповіщає все залежні від неї представлення, і вони оновлюються. Для цього використовується шаблон «спостерігач» При обробці реакції користувача представлення вибирає, залежно від потрібної реакції, потрібний контролер, який забезпечить той чи інший зв’язок з моделлю. Для цього використовується шаблон «стратегія», або замість цього може бути модифікація з використанням шаблону «команда». Крім того, можуть використовуватися й інші шаблони проектування, наприклад, «фабричний метод», який дозволить задати за замовчуванням тип контролера для відповідного представлення.

* 1. *Завдання MVC*

Основна мета застосування цієї концепції полягає в відокремленні бізнес-логіки (моделі) від її візуалізації (представлення, виду). За рахунок такого поділу підвищується можливість повторного використання. Найбільш корисне застосування даної концепції в таких випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах та/або з різних точок зору. Зокрема, виконуються наступні завдання:

* До однієї моделі можна приєднати кілька представлень, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і діаграми.
* Не торкаючись реалізації представлень, можна змінити реакції на дії користувача (натискання на кнопку, введення даних), для цього досить використовувати інший контролер.
* Ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можна досягти того, що програміст, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть знати про те, який вигляд буде використовуватися.
  1. *Python*

Дану курсову роботу буде виконано, використовуючи мову програмування Python та бібліотеку Tkinter. Бібліотека Tkinter зручна та має велику кількість можливостей. Вона зможе надати нам змогу управляти графічними виглядом елементів.

* 1. *Використання концепції MVC в даній роботі*

В даній роботі буде використовуватися модифікована версія шаблону MVC, яка об’єднує представлення і контролер в один логічний об’єкт. MVC розділить систему на дві умовні частини, а саме: модель даних (база даних або файл), вигляд даних (графічний інтерфейс користувача), об’єднаний з керуванням (логіка програми). Це буде зроблено з метою відокремлення моделі даних від інтерфейсу користувача, щоб зміни в будь-якій з цих частин системи мінімально впливали на інші частину нашої системи.

1. **Проектування програмного додатку**
   1. *Прецеденти*

Можливості користувача додатку можна побачити на Рис. 2.1.

Користувач може додавати студентів до бази даних, видаляти студентів з

бази даних, оновлювати відображення, переглядати відображення.

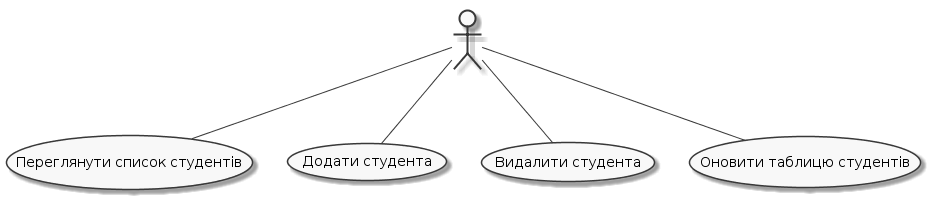


Рис. 2.1. Можливості користувача

Користувач може побачити успішність групи, вибравши курс, спеціальність

та групу та натиснувши на кнопку «Вибрати» (Рис. 2.2).

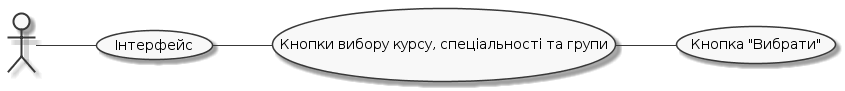


Рис. 2.2. Операція входу до вікна перегляду таблиці успішності студентів

Користувач може побачити успішність групи у вигляді діаграми, натиснувши

на кнопку «Статистика» (Рис. 2.3).

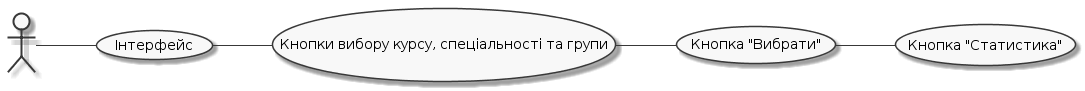


Рис. 2.3. Операція перегляду успішності групи у вигляді діаграми

Користувач може додавати та видаляти студентів з групи, увійшовши в меню

редагування, натиснувши кнопку «Редагувати» (Рис. 2.4).

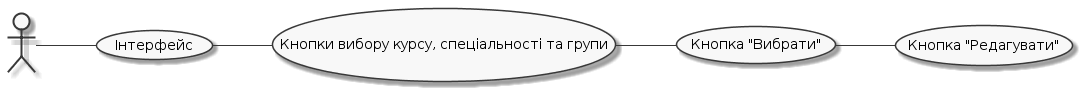


Рис. 2.4. Операція входу до меню редагування

Користувач може додати вказаного студента до групи, натиснувши кнопку

«Додати» у вікні редагування та вписавши потрібну інформацію (Рис. 2.5).

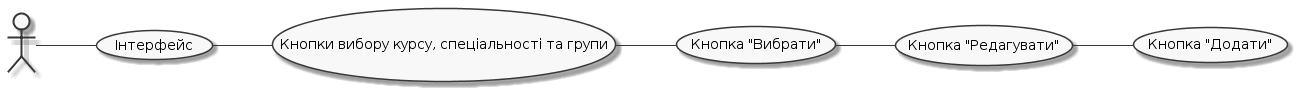


Рис. 2.5. Операція додавання студента до групи

Користувач може видалити вказаного студента з групи, натиснувши кнопку

«Видалити» у вікні редагування (Рис. 2.6).

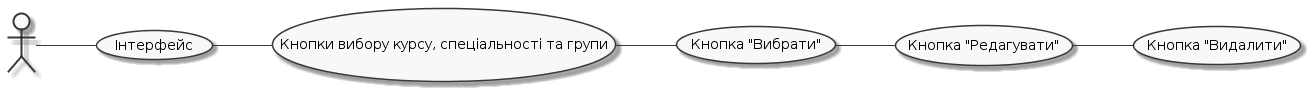


Рис. 2.6. Операція видалення студента з групи

Користувач може оновити список студентів після внесених змін, натиснувши

кнопку «Оновити» (Рис. 2.7).



Рис. 2.7. Операція оновлення відображення таблиці студентів

* 1. *Ескізи графічного інтерфейсу*

На рисунках 2.8, 2.9, 2.10 та 2.11 можна побачити ескізи всіх можливих вікон

програми.



Рис. 2.8 Ескіз вікна вибору групи

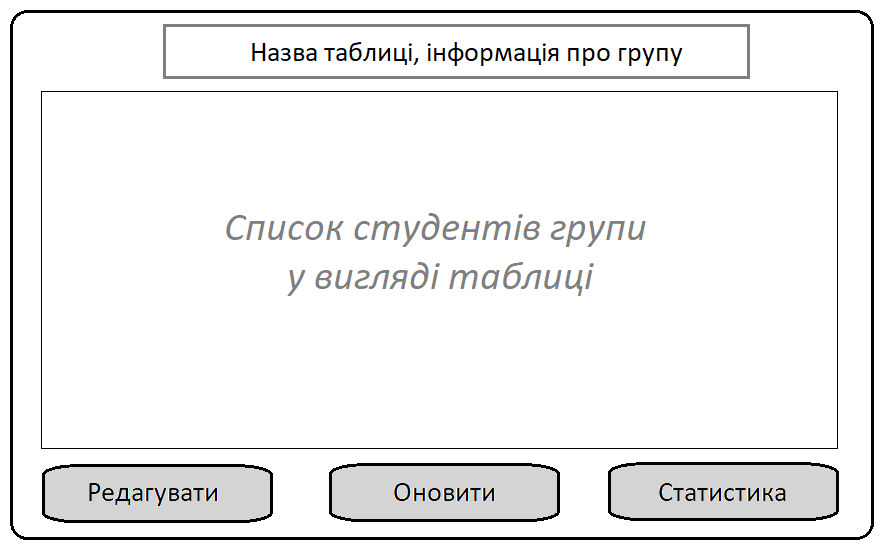


Рис. 2.9 Ескіз вікна відображення списку успішності студентів групи

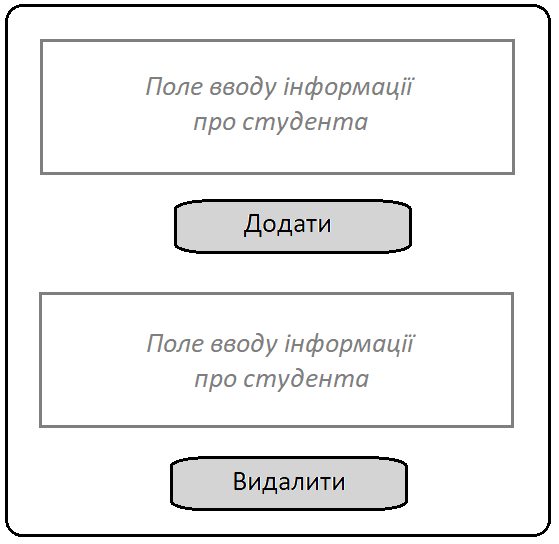


Рис. 2.10 Ескіз вікна редагування

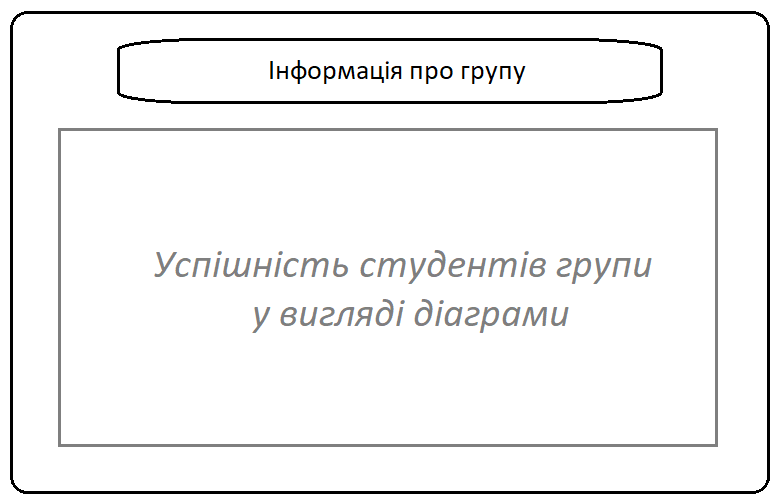


Рис. 2.11 Ескіз вікна відображення успішності студентів групи в вигляді діаграми

На рисунку 2.12 зображено діаграму граничних класів.

Рис. 2.12 Діаграма граничних класів

*2.3 Таблиця відповідності елементів бібліотеці Tkinter*

1. **РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ**
   1. *Діаграма класів*

На діаграма зображено діаграму класів програми. На ній можна побачити

класи, що реалізують інтерфейс користувача (View1, View2, View3, View4) та

клас моделі (Model), що забезпечує коректну роботу з базою даних.

Рис. 3.1 Діаграма класів

* 1. *Опис моделі*
  2. *Опис View*

1. **ТЕСТУВАННЯ ПРОГРАМНОГО ЗАПЕЗПЕЧЕННЯ**

**ВИСНОВКИ**

**ЛІТЕРАТУРА**

**ДОДАТКИ**

*ДОДАТОК 1: ПРОГРАМНИЙ КОД ПРОЕКТУ*