Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання і програмного забезпечення

**КВАЛІФІКАЦІЙНА РОБОТА**

**на здобуття ступеня вищої освіти магістра**

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–24м
_____/ С.О. Антонов /

Керівник кваліфікаційної роботи _____/ Д. В. Швець /

Завідувач кафедри _____/ А. М. Стрюк /

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«____» _____ 2025 р.

**ЗАВДАННЯ**

**на кваліфікаційну роботу**

студенту групи ІПЗ–24м Антонову Сергію Олександровичу

1. Тема: «Розробка програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів» затверджена наказом по КНУ № 204с від «10» квітня 2025 р.

2. Термін подання студентом закінченої роботи: «30» листопада 2025 р.

3. Вихідні дані по роботі: розроблений програмний комплекс має забезпечити оцінювання успішності студентів та прогнозувати подальший розвиток їх навчальної траекторії.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): провести аналіз існуючих наукових досліджень за заданою тематикою, обґрунтувати функціонал розроблюваної системи, створити програмне забезпечення для прогнозування успішності здобувачів вищої освіти, здійснити тестування розробленого додатку.

5. Перелік ілюстративного матеріалу: функціональна схема, блок–схема алгоритму, зображення екранних форм додатку.

Календарний план:

| № | Найменування етапів кваліфікаційної роботи | Термін виконання |
|---|---|---|
| 1 | Аналіз літературних джерел та огляд інтернет-ресурсів з заданої тематики | 06.01.25 – 14.02.25 |
| 2 | Пошук існуючих методів вирішення проблеми | 15.02.25 – 10.03.25 |
| 3 | Формулювання актуальності роботи і аналіз предметної області | 11.03.25 – 27.03.25 |
| 4 | Оформлення матеріалів першого розділу роботи | 28.03.25 – 13.04.25 |
| 5 | Визначення об’єкту, предмету та мети дослідження | 14.04.25 – 02.05.25 |
| 6 | Оформлення матеріалів другого розділу роботи | 03.05.25 – 16.05.25 |
| 7 | Проектування програмного комплексу | 17.05.25 – 06.06.25 |
| 8 | Оформлення матеріалів третього розділу роботи | 07.06.25 – 25.06.25 |
| 9 | Розробка функціональної схеми та алгоритму програми | 26.06.25 – 14.07.25 |
| 10 | Розробка програмного забезпечення системи | 15.07.25 – 11.09.25 |
| 11 | Оформлення матеріалів четвертого розділу роботи | 12.09.25 – 04.10.25 |
| 12 | Формування прикладів використання ПЗ | 05.10.25 – 17.10.25 |
| 13 | Оформлення матеріалів п’ятого розділу роботи | 18.10.25 – 02.11.25 |
| 14 | Аналіз економічної ефективності розробки | 03.11.25 – 15.11.25 |
| 15 | Остаточне оформлення пояснювальної записки | 16.11.25 – 29.11.25 |

Дата видачі завдання: «05» січня 2025 р.

Студент: _____ / С. О. Антонов /

Керівник роботи: _____ / Д. В. Швець /

# РЕФЕРАТ

МАРКІВСЬКИЙ ЛАНЦЮГ, ТРАЕКТОРІЯ НАВЧАННЯ, ОЦІНЮВАННЯ, ПРОГНОЗУВАННЯ УСПІШНОСТІ, ПРОГРАМНИЙ КОМПЛЕКС.

Пояснювальна записка: 82 с., 14 рис., 7 табл., 2 дод., 27 джерел.

Метою кваліфікаційної роботи стало створення програмного комплексу, здатного здійснювати прогнозування успішності здобувачів вищої освіти із застосуванням апарату Марківських ланцюгів. Об'єктом проектування є програмний комплекс, функціонування якого базується на математичному моделюванні динаміки навчальних результатів.

У теоретичному розділі обґрунтовано актуальність розробки, розкрито стан сучасних підходів до прогнозування академічної успішності та проаналізовано програмні рішення, що вже представлені на ринку. Окреслено їхні переваги та наявні обмеження, що дало можливість визначити напрями вдосконалення та сформулювати завдання, необхідні для створення конкурентоспроможної системи.

Практична частина роботи присвячена створенню функціональної структури майбутнього продукту, розробленню алгоритмічної логіки та побудові інтерфейсу, орієнтованого на зручність і наочність використання. Було забезпечено повний цикл проєктування - від опрацювання архітектури до тестування і перевірки працездатності програмного забезпечення в різних умовах експлуатації.

Створений програмний комплекс демонструє потенціал для застосування у різних освітніх середовищах, оскільки дозволяє підвищити обґрунтованість прогнозів щодо динаміки навчальних досягнень здобувачів вищої освіти та може слугувати інструментом для підтримки відповідних управлінських рішень.

# ABSTRACT

MARKOV CHAIN, LEARNING TRAJECTORY, ASSESSMENT, SUCCESS FORECASTING, SOFTWARE COMPLEX.

Explanatory note: 82 p., 16 fig., 7 tabl., 2 app., 27 references.

The aim of the qualification work was to create a software complex capable of predicting the academic success of higher education applicants using Markov chains. The object of design is a software complex, the functioning of which is based on mathematical modelling of the dynamics of academic results.

The theoretical section substantiates the relevance of the development, reveals the state of modern approaches to predicting academic success, and analyses software solutions already available on the market. Their advantages and limitations are outlined, which made it possible to identify areas for improvement and formulate the tasks necessary to create a competitive system.

The practical part of the work is devoted to creating the functional structure of the future product, developing algorithmic logic, and building an interface focused on ease and clarity of use. A comprehensive design cycle was provided, encompassing architecture development, software testing, and verification of performance under various operating conditions.

The created software complex demonstrates potential for application in various educational environments, as it enables the enhancement of forecast validity regarding the dynamics of academic achievements of higher education seekers and can serve as a tool to support relevant management decisions.

# ЗМІСТ

# ВСТУП

Ця робота присвячена розробці програмного забезпечення для прогнозування академічного успіху студентів вищої освіти через використання сучасних математичних методів, зокрема методів марківського аналізу. Вона розглядає підходи до побудови системи, яка здатна виявляти та оцінювати ймовірні сценарії розвитку академічної траєкторії кожного студента, на основі послідовного аналізу його результатів навчальної діяльності.

Основна ідея дослідження полягає в математичному моделюванні навчальних траєкторій студентів як процесу переходів між різними станами. Такий підхід дозволяє не лише оцінювати фактичні досягнення студентів, як-от оцінки за курсові роботи, заліки та іспити, але й аналізувати їхній прогрес у часі та відхилення від типового плану проходження навчальної траекторії. Крім того, це дає змогу відображати складну структуру освітнього процесу, де існують численні фактори впливу на результати навчання, зокрема індивідуальні особливості студента, обрані курси, а також зовнішні чинники.

Для побудови такого програмного забезпечення передбачено використання не тільки класичних марківських ланцюгів, що описують імовірнісні переходи між дискретними станами системи, а й їх розширених варіантів. Наприклад, приховані марківські моделі дозволяють враховувати приховані фактори, які впливають на динаміку навчальної траєкторії студента. Це може бути рівень його мотивації, якість підготовки до занять або індивідуальні стратегії навчання. Також можливе використання сумішових марківських моделей, що дають змогу моделювати гетерогенність у вибірці студентів - наприклад, виділяти підгрупи з різними профілями успішності.

У результаті такого математичного моделювання можливо створити інтелектуальну систему підтримки прийняття рішень, яка допомагатиме адміністрації навчального закладу або викладачам здійснювати більш точне прогнозування майбутніх результатів навчання кожного студента. Така

система дозволить автоматично виявляти студентів з підвищеним ризиком відставання або потенційного відрахування, а також забезпечить можливість розробки індивідуальних планів підтримки для цих студентів. Наприклад, своєчасне надання додаткових консультацій або індивідуальних занять з урахуванням виявлених слабких місць у їхній підготовці.

Таким чином, ця робота має на меті створити комплексне програмне забезпечення, що інтегрує методи статистичного та математичного аналізу для формування науково обґрунтованих прогнозів успішності студентів. Запропонований підхід здатен суттєво підвищити якість управління освітнім процесом у вищих навчальних закладах, сприяючи не лише зменшенню кількості відрахувань, а й покращенню загальної академічної успішності та професійної підготовки випускників.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГНОЗУВАННЯ УСПІШНОСТІ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ НА ОСНОВІ МАРКІВСЬКИХ ЛАНЦЮГІВ

## 1.1 Актуальність роботи

За останні десятиліття стрімке зростання обсягів академічних даних сприяло активному розвитку нових підходів до аналізу успішності здобувачів вищої освіти. Нині навчальні заклади накопичують величезні масиви інформації: від результатів іспитів і курсових робіт до відомостей про відвідуваність, активність у навчальному процесі та індивідуальні траєкторії засвоєння матеріалу. Такі дані стають важливим джерелом для аналізу й прогнозування академічної успішності студентів.

У сучасних умовах зростаючої цифровізації освітнього процесу дані про результати навчання, оцінки за курсові роботи, вибір курсів і спеціалізацій, а також інші важливі параметри інтегруються в єдину систему, що дозволяє моделювати складні послідовності академічних подій. Одним з найбільш ефективних інструментів у цьому контексті виступають марківські ланцюги [1], які застосовуються для відображення часової динаміки академічних станів студентів. Використання марківських моделей базується на припущенні, що майбутній стан студента (наприклад, успішне завершення курсу, ризик відставання чи відрахування) визначається лише його поточним станом, без необхідності зберігати повну історію всіх попередніх кроків. Це дозволяє спростити розрахунки та зосередитися на ключових змінах у навчальній траєкторії.

Особливої актуальності використання марківських ланцюгів набуває в контексті зростаючих вимог до якості освіти, жорсткої конкуренції між університетами та необхідності підвищення успішності студентів. Сучасні освітні установи повинні не лише надавати якісні знання, а й забезпечувати ефективне управління навчальними процесами. У цьому плані прогнозування

на основі марківських моделей виступає потужним засобом підтримки студентів, оскільки дозволяє:

– оперативно виявляти студентів, які можуть потребувати додаткової підтримки;

– розробляти індивідуальні рекомендації щодо вибору навчальних курсів і напрямків;

– своєчасно призначати консультації або коригуючі заходи, щоб запобігти ризику відставання чи відрахування.

Такий підхід особливо важливий у сучасній академічній спільноті, де гнучкість і персоналізація навчального процесу виходять на перший план. Завдяки марківським моделям університети отримують можливість підвищити якість навчання, створити умови для успішного засвоєння матеріалу та всебічного розвитку кожного студента. Крім того, використання цих методів є частиною глобального тренду впровадження інтелектуальних технологій у сферу освіти, що відповідає сучасним викликам і вимогам суспільства.

Таким чином, розробка та впровадження програмного забезпечення для прогнозування академічного успіху із застосуванням марківських ланцюгів є не лише актуальним, а й необхідним кроком для підвищення ефективності освітнього процесу та досягнення високих стандартів підготовки майбутніх фахівців.

## 1.2 Дослідження особливості предметної галузі

Сфера прогнозування успішності здобувачів вищої освіти охоплює широкий спектр даних і факторів. У сучасних університетах та інших закладах вищої освіти збирається велика кількість інформації, пов’язаної з навчальною діяльністю студентів. Ці дані включають як кількісні, так і якісні характеристики: результати іспитів, курсових робіт, заліків, вибір освітніх

програм, участь у додаткових освітніх заходах, а також показники відвідуваності, виконання завдань та інші форми активності.

Особливістю цієї предметної галузі є наявність складної часової динаміки в поведінці студентів. Процес навчання не є статичним – він постійно розвивається у часі, і кожен студент має індивідуальну траєкторію. Це означає, що не лише поточні результати мають значення, а й порядок, у якому студент проходить курси та отримує оцінки. Таким чином, виникає необхідність моделювати саме послідовності подій, що відкриває перспективи для застосування стохастичних моделей, зокрема марківських ланцюгів.

Ще однією особливістю предметної галузі є те, що процес навчання можна розглядати як послідовність станів: від початкового стану «новий студент» через різні етапи академічного прогресу (наприклад, складання окремих курсів, отримання проміжних оцінок) до фінального стану, пов’язаного з випуском або відрахуванням [2]. Ці стани не завжди безпосередньо спостерігаються, адже деякі внутрішні фактори, такі як психологічна мотивація чи рівень розуміння матеріалу, не мають прямих цифрових відображень у даних. Усе це створює додаткову складність та потребує використання методів, які здатні працювати з прихованими процесами та неповними даними.

У практичному плані важливим завданням є також своєчасне виявлення студентів із високим ризиком академічної неуспішності або відрахування. Це необхідно не лише для забезпечення якісної освіти та запобігання академічному відставанню, а й для підвищення загальної ефективності навчального процесу та оптимального розподілу ресурсів (наприклад, призначення наставників чи додаткових консультацій).

У цій предметній галузі існує низка викликів, пов’язаних зі складністю та варіативністю даних. Дані про студентів часто мають різну структуру: від цифрових оцінок до текстових коментарів викладачів. Вони можуть бути неповними (через пропуски або відсутність даних про проміжні етапи) і потребують додаткової обробки. Крім того, у багатьох випадках

спостерігається необхідність адаптації моделей до змінних умов освітнього середовища, наприклад, змін навчальних планів чи впровадження нових курсів.

Таким чином, предметна галузь прогнозування академічної успішності студентів у вищій освіті є актуальною та водночас складною. Її ключові риси полягають в динамічному розвитку, різнорідності даних, появі прихованих факторів і потребі в точному моделюванні процесів, що дозволяє підвищити якість освіти та надати цінні інструменти для адміністративного управління та підтримки студентів. Тому використання математичних моделей, зокрема марківських ланцюгів та їхніх варіацій, є обґрунтованим напрямом дослідження в цій галузі.

### 1.3 Аналіз досліджень з обраної теми

На основі проведених досліджень можна вказати на декілька прикладів успішного використання марківських моделей для прогнозування академічної успішності. Так, у статті [3] було продемонстровано, що побудова моделі марківських процесів прийняття рішень дозволяє ефективно оптимізувати вибір навчальних курсів. Це, у свою чергу, веде до підвищення рівня своєчасного завершення навчання та значного підвищення загальної успішності студентів, що доводить практичну користь застосування таких моделей у плануванні освітніх траєкторій.

Дослідження [4] також стало важливим внеском у цю сферу, продемонструвавши, що традиційні марківські ланцюги можна успішно використовувати для аналізу переходів між різними академічними станами студентів. Такий підхід дає змогу будувати прогнози щодо ризиків відрахування та оцінювати потенціал для академічного зростання, що є головними аспектами для своєчасного втручання та підтримки студентів у процесі навчання.

Робота [5] та його колег продемонструвала ще одну перспективну можливість Марківських моделей — моделювання еволюції знань студентів у

медичній освіті. Завдяки використанню Марківських ланцюгів у цьому контексті вдалося досягти високої точності прогнозування результатів ліцензійних іспитів, що свідчить про універсальність підходу та його придатність навіть у вузькоспеціалізованих сферах навчання.

Ще одним напрямом, який підтверджує ефективність використання марківських моделей, є дослідження [6] у контексті масових відкритих онлайн-курсів. Вони продемонстрували, що послідовне моделювання активності студентів на основі марківських моделей допомагає прогнозувати майбутню успішність, а також своєчасно виявляти студентів, яким потрібна додаткова підтримка чи консультації. Це відкриває нові можливості для адаптивного навчання в умовах масової дистанційної освіти.

Крім класичних марківських ланцюгів можна використати приховані Марківські моделі та сумішові моделі.

Приховані Марківські моделі дозволяють моделювати ситуації, коли справжній академічний стан студента не є безпосередньо спостережуваним, а може бути оцінений на основі емісійних характеристик, таких як оцінки, відвідуваність, активність у навчальних платформах тощо. За допомогою спеціальних алгоритмів (Baum-Welch [7], Viterbi [8]) можливо визначити найбільш ймовірні послідовності прихованих станів, що дозволяє ретельно прогнозувати академічну траєкторію та своєчасно виявляти студентів з підвищеним ризиком.

Водночас дослідження [9] показало важливість застосування прихованих марківських моделей для розкриття латентних станів академічного розвитку студентів. Завдяки цьому підходу можна виявляти приховані фактори, що впливають на успішність студентів, а також здійснювати ранню діагностику ризику відрахування, що дозволяє своєчасно вживати заходів підтримки.

Сумішові моделі [10] дозволяють кластеризувати студентів за подібними траєкторіями розвитку, що відкриває можливості для персоналізації підтримки. Так, відомо, що студенти можуть належати до

різних підгруп - від «сталих», які стабільно досягають високих результатів, до «мандрівників», траєкторії яких характеризуються значними коливаннями успішності [11].

Отже, проведені дослідження у різних галузях освітньої діяльності підтверджують, що марківські моделі та їхні різновиди мають великий потенціал для покращення якості навчального процесу, підвищення успішності студентів та розробки ефективних систем підтримки в освіті.

## 1.4 Аналіз існуючого ПЗ для прогнозування успішності студентів та висновки по виконаному аналізу

Ринок програмних рішень, спрямованих на прогнозування успішності студентів, представлений переважно універсальними аналітичними платформами та хмарними системами освітньої аналітики. Найпоширеніші інструменти не орієнтовані виключно на освітню сферу, але активно використовуються закладами вищої освіти завдяки підтримці методів машинного навчання та прогнозного моделювання, а також завдяки можливостям візуалізації даних.

До таких інструментів, наприклад, належить IBM SPSS Modeler [12], що пропонує широкі можливості побудови моделей класифікації та регресії. Завдяки блочному підходу та розвиненому функціоналу статистичного аналізу платформа дозволяє створювати моделі ризику відрахування, прогнозувати середні бали та виконувати оцінювання навчальних тенденцій. Продукт орієнтований на корпоративне використання та має високий поріг входу, що ускладнює його впровадження в невеликих навчальних закладах.

Ще однією поширеною платформою є RapidMiner [13], який також застосовується для прогнозного аналізу даних успішності. Він забезпечує широкий набір алгоритмів машинного навчання, можливість роботи з великими наборами даних і підтримку інтеграції з різними джерелами зберігання. Проте вільна версія обмежена функціоналом, а повна комерційна ліцензія вимагає значних фінансових витрат.

Поширеним інструментом освітньої аналітики є Microsoft Power BI [14], який дає змогу створювати інтерактивні візуалізації та виконувати базовий прогнозний аналіз. Його використання потребує роботи в хмарі або налаштування окремого серверного середовища. Для багатьох навчальних закладів це створює певні бар'єри у впровадженні, а прогнозні можливості суттєво обмежені порівняно з професійними системами машинного навчання.

Ще одним варіантом є платформа Orange Data Mining [15], орієнтована на навчальні та дослідницькі проєкти. Вона забезпечує швидке створення моделей класифікації та кластеризації, підтримує інтерактивні візуалізації й має зручний інтерфейс. Попри це, система не пропонує готових рішень для освітніх процесів і застосовується переважно як інструмент для експериментів.

Попри значну різноманітність доступних рішень, більшість платформ покладається на хмарну інфраструктуру або передбачає складні процедури розгортання. Це створює ризики витоку персональних даних студентів, обмежує автономність закладу та підвищує вартість володіння програмним продуктом. Системи універсального призначення не враховують специфіку навчальних планів, локальних моделей оцінювання та внутрішніх методів аналізу успішності. Крім того, їх адаптація під конкретні навчальні процеси часто вимагає залучення окремих фахівців, що збільшує час і вартість впровадження.

Десктопний застосунок, створений спеціально для аналізу академічної успішності, усуває ці недоліки. Він не залежить від зовнішніх серверів та хмарних сервісів, забезпечує повний контроль над даними всередині локальної інфраструктури та гарантує конфіденційність інформації про студентів. Локальна обробка дає змогу використовувати будь-які кастомні алгоритми прогнозування, зокрема моделі на основі марковських ланцюгів, без необхідності адаптації до стандартів хмарних середовищ.

Десктопний формат дозволяє працювати незалежно від стабільності мережевого з'єднання, що дозволяє використовувати її в умовах обмеженого

доступу до Інтернету та відключень електропостачання. Окрім того, використання локальної архітектури робить систему доступною навіть на комп’ютерах із невисокими технічними характеристиками, що підвищує її універсальність у навчальному середовищі.

Узагальнюючи результати аналізу, можна стверджувати, що наявні аналоги забезпечують широкий спектр можливостей для прогнозної аналітики, але не враховують специфічних потреб навчальних закладів у персоналізації, безпеці даних та автономності роботи. Десктопне програмне забезпечення зможе поєднати вузьку спеціалізацію з гнучкістю налаштування та створить інструмент, що повністю адаптується під потреби закладу освіти, без залежності від сторонніх сервісів чи інфраструктури.

# 2 НАУКОВИЙ АПАРАТ ДОСЛІДЖЕННЯ З ПРОГНОЗУВАННЯ УСПІШНОСТІ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ

## 2.1 Формулювання основних напрямків досліджень при розробці програмного комплексу для прогнозування успішності здобувачів вищої освіти

Розглянемо мету, об’єкт і предмет дослідження, поставленні завдання та методи, що будуть використовуватися в ході роботи.

***Мета дослідження*** полягає в розробці програмного комплексу, що дозволяє прогнозувати академічну успішність студентів на основі використання марківських ланцюгів, а також виявляти студентів із підвищеним ризиком відставання або відрахування.

***Об’єкт дослідження***: процеси прогнозування академічної успішності здобувачів вищої освіти.

***Предмет дослідження***: методи та алгоритми оцінки навчальної діяльності студентів з використанням марківських ланцюгів.

***Поставленні завдання дослідження:***

а) проаналізувати наукові основи теорії марківських ланцюгів;

б) дослідити сучасні підходи до прогнозування академічної успішності студентів;

в) виявити вимоги до функціональних і нефункціональних характеристик програмного комплексу;

г) розробити архітектуру та алгоритми програмного комплексу для прогнозування успішності студентів.

***Методи дослідження:***

а) теоретичний аналіз літературних джерел, присвячених тематиці марківських процесів та їх застосування;

б) методи математичного моделювання (марківські ланцюги, приховані марківські моделі);

в) методи аналізу даних та статистичної обробки для оцінювання ймовірностей переходів;

г) прототипування програмного забезпечення.

## 2.2 Викладення основних наукових положень стосовно використання Марковських ланцюгів

Розглянемо деякі основні теоретичні відомості стосовно марківських ланцюгів.

Марківські ланцюги належать до фундаментального класу стохастичних процесів [16], які широко застосовуються для моделювання та аналізу систем, що змінюються у часі під впливом випадкових факторів. Основною особливістю цих процесів є так звана «властивість Маркова» або «відсутність пам’яті», що означає: імовірність переходу системи до наступного стану залежить лише від її поточного стану і не потребує знання попередньої історії розвитку [17]. Такий підхід дозволяє значно спростити математичний опис складних явищ, зводячи багатоступеневі залежності до простіших співвідношень.

Розглянемо основні поняття, пов’язані з теоретичними основами марковських ланцюгів.

***Стан*** являє собою дискретну характеристику або опис системи в певний момент часу. У загальному випадку під станом розуміють будь-який елемент скінченного або зліченного простору станів *S*. Система у кожний момент часу знаходиться лише в одному зі станів, і цей стан може змінюватися при переході на наступний крок.

***Простір станів*** – це множина, яка включає всі можливі стани, у яких може перебувати система. Простір може бути як скінченним (кількість станів обмежена), так і нескінченним (наприклад, натуральні числа або інші зліченні множини).

Дана множина описується наступним виразом

$$S = \{s_1, s_2, \dots, s_n\} \tag{2.1}$$

***Перехідні ймовірності*** – це ймовірності $p_{ij}$ переходу системи зі стану $i$ у стан $j$ за один дискретний часовий крок. Всі ці ймовірності організовуються у вигляді матриці перехідних ймовірностей:

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{32} & \cdots & p_{nn} \end{pmatrix} \tag{2.2}$$

Матриця $P$ має властивість стохастичності: сума ймовірностей у кожному рядку дорівнює 1, тобто

$$\sum_{j=1}^{n} p_{ij} = 1 \tag{2.3}$$

***Розподіл станів*** – це вектор $\pi(t)$, який описує ймовірності перебування системи у кожному з можливих станів у момент часу $t$. Він визначається формулою

$$\pi(t) = \left[\pi_{1(t)}, \pi_{2(t)}, \dots, \pi_{n(t)}\right] \tag{2.4}$$

Початковий розподіл станів при цьому задається вектором $\pi(0)$.

Що стосується динамічного розвитку системи, то еволюція системи у часі описується за допомогою ітераційного співвідношення:

$$\pi(t+1) = \pi(t) \cdot P \tag{2.5}$$

Іншими словами, розподіл станів у момент часу *t+1* отримується як добуток поточного розподілу та матриці перехідних ймовірностей. Для передбачення станів системи на довільний часовий горизонт використовується піднесення матриці *P* до степеня *t*:

$$\pi(t) = \pi(0) \cdot P^t \tag{2.6}$$

Марківські ланцюги мають низку важливих властивостей, серед яких: стаціонарний розподіл, ергодичність, періодичність і неперіодичність.

Якщо існує вектор $\pi*$, що задовольняє умову

$$\pi^* = \pi^* \cdot P \tag{2.7}$$

то за достатньо великого часу *t* розподіл станів *π(t)* збігається до стаціонарного розподілу незалежно від початкового стану.

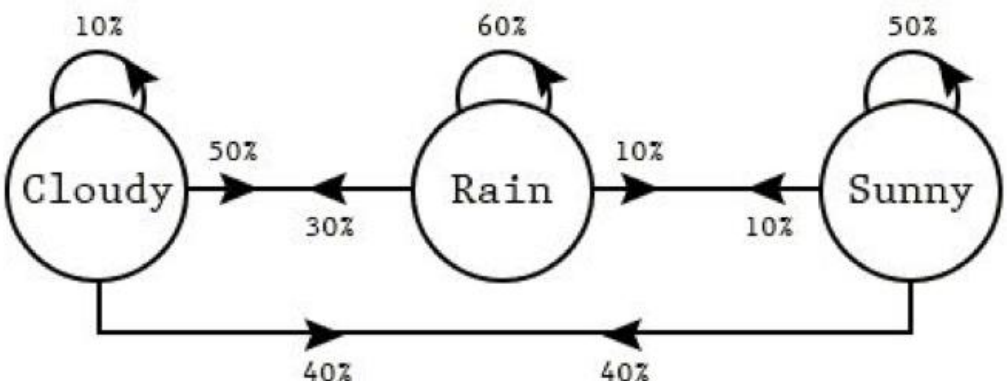



Рисунок 2.1 – Приклад марківського ланцюга з трьома станами

Ергодичністю, своєю чергою, називають властивість ланцюга, за якої стаціонарний розподіл існує та є єдиним, а процес «забуває» початковий стан.

Деякі ланцюги мають циклічну (періодичну) поведінку, тоді як інші – неперіодичну, що впливає на збіжність до стаціонарного розподілу.

Також варто розглянути приховані марківські моделі [18] як окремий клас математичних моделей, які поєднують ідеї ймовірнісних переходів між станами та ймовірності спостережень, що виникають у цих станах. Головна відмінність від класичних марківських ланцюгів полягає в тому, що у прихованих марківських моделях реальні, так звані приховані стани, безпосередньо не спостерігаються. Натомість можна бачити лише певні спостережувані дані або сигнали, які опосередковано «відображають» ці приховані стани.

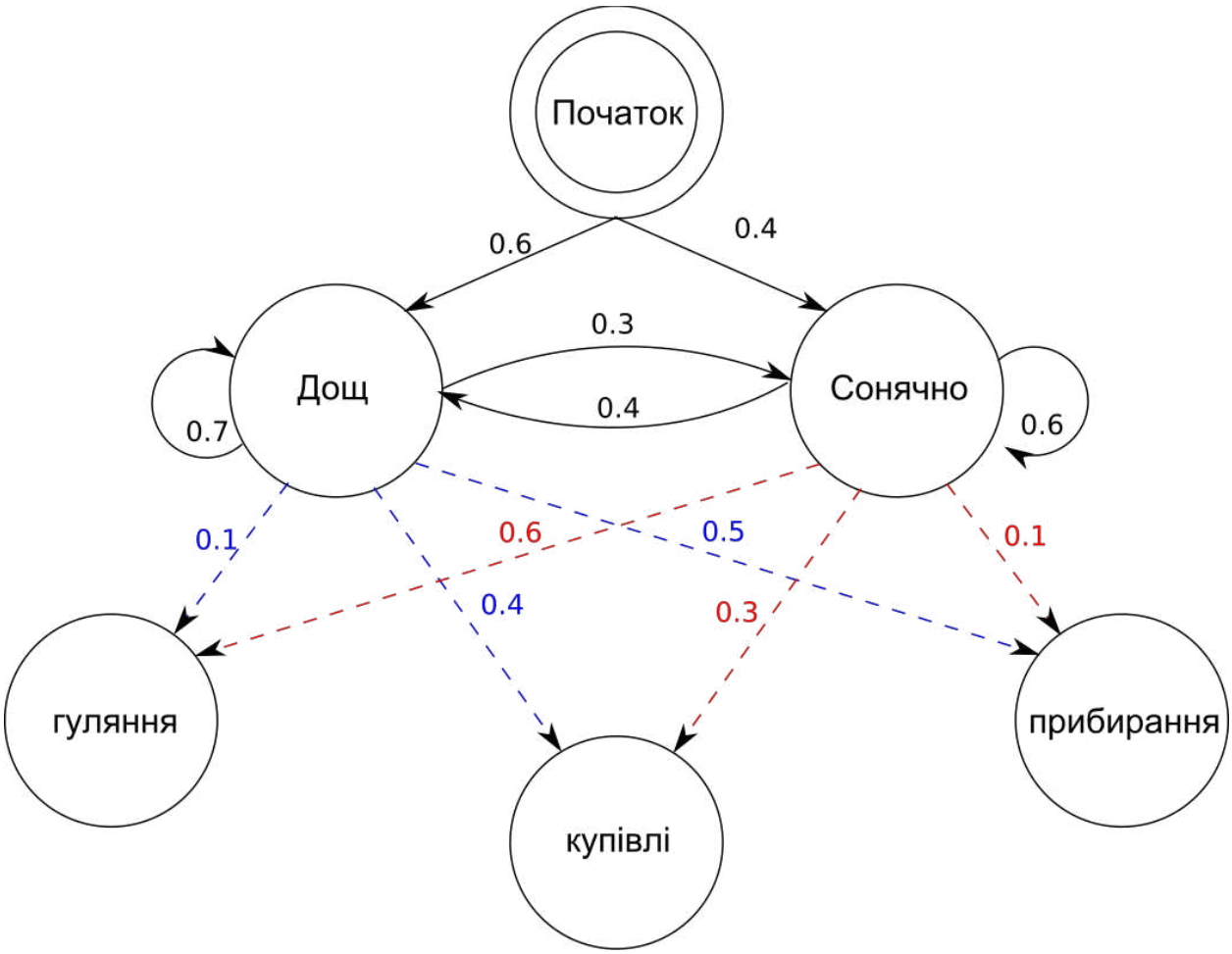



Рисунок 2.2 – Приклад прихованого марківського ланцюга [19]

Інтуїтивно приховані марківські моделі можна уявити як систему, що має внутрішні «режими роботи» або «стратегічні сценарії», які не видно зовні. Ми отримуємо лише спостережувані наслідки, але самі приховані режими

залишаються невідомими. Проте, знаючи статистичні закономірності переходів між режимами та способи їх прояву у вигляді спостережень, ми можемо робити обґрунтовані висновки про структуру цих прихованих процесів.

Основні складові прихованої марківської моделі формують єдину цілісну структуру, де все взаємопов'язано. У центрі моделі стоять приховані стани, які визначають «режими» або «положення» системи у кожний момент часу. Ці стани неможливо спостерігати напряму, але вони визначають її поведінку та впливають на розвиток подій. Приклади таких прихованих станів у різних галузях – це фази ринку у фінансах, окремі етапи біологічного процесу або внутрішні режими роботи технічної системи.

Далі у моделі виділяються спостереження – це дані, які ми реально фіксуємо або реєструємо. Вони є зовнішніми проявами діяльності системи та, фактично, наслідками активності прихованих станів. Наприклад, у випадку аналізу мовлення це можуть бути конкретні звуки або слова, у фінансових дослідженнях – коливання цін акцій, а в біологічних системах – показники виміряних біомаркерів.

Залежність спостережень від прихованих станів означає, що кожному прихованому стану відповідає характерний набір імовірностей, з якими можуть з’являтися певні спостереження. Це створює своєрідний «малюнок» або «профіль» для кожного стану, завдяки якому можна прогнозувати, які спостереження ймовірніші за інших.

При цьому динаміка системи полягає у поступових переходах між прихованими станами. На відміну від простих спостережуваних даних, самі ці приховані стани можуть змінюватися з часом.

Особливість такої динаміки полягає в тому, що ймовірність переходу в новий стан залежить лише від поточного стану, а не від усієї історії змін. Це властивість була успадкована від класичних марківських ланцюгів і забезпечує зручність для моделювання складних процесів.

Науковий апарат марківських ланцюгів застосовується в багатьох галузях – від біології та фізики до економіки, фінансів і теорії управління. Їх універсальність і здатність відображати стохастичну динаміку систем дозволяють моделювати різноманітні процеси: черги обслуговування, фінансові ризики, еволюцію популяцій, а також поведінку складних систем у різних сферах.

Таким чином, марківські ланцюги пропонують вагомий математичний інструмент для дослідження й прогнозування послідовних явищ, де кожен наступний стан визначається лише поточним імовірнісним сценарієм розвитку.

# 3 ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ПРОГНОЗУВАННЯ УСПІШНОСТІ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ

## 3.1 Функціональні та нефункціональні вимоги до програмного комплексу для прогнозування успішності здобувачів вищої освіти

Проєктування програмного комплексу для прогнозування успішності здобувачів вищої освіти передбачає визначення функціональних та нефункціональних вимог, що забезпечать його ефективність при використанні, надійність та ергономічність.

Функціональні вимоги описують основні можливості системи, які необхідно реалізувати для досягнення поставлених цілей. Основними функціональними вимогами до програмного комплексу є:

а) ***Збір та обробка даних***, що забезпечить імпорт даних про академічну діяльність студентів, включаючи оцінки, інформацію про відвідуваність, виконання завдань та інші релевантні параметри;

б) ***Формування послідовностей подій***, що полягає в реалізації механізмів представлення навчальних траєкторій студентів у вигляді послідовностей станів для подальшого аналізу;

в) ***Побудова моделей*** - система має підтримувати створення та налаштування моделей марківських ланцюгів (у тому числі прихованих моделей) для прогнозування результатів;

г) Комплекс також повинен виконувати ***прогнозування майбутнього стану успішності*** кожного студента, зокрема визначення його категорії, яка демонструє ймовірність успішного завершення курсу студентом чи наявності ризику його відрахування;

д) ***Візуалізація результатів***, яка полягає в реалізації зрозумілого інтерфейсу для відображення прогнозів.

Нефункціональні вимоги визначають характеристики, які впливають на якість роботи системи та взаємодію з користувачем, та полягають у надійності для стабільної роботи без збоїв; масштабованості в плані обсягу даних; реалізації захисту персональних даних студентів; високої продуктивності та наявності інтуїтивно зрозумілого інтерфейсу.

Отже, визначення функціональних та нефункціональних вимог дозволяє окреслити архітектуру та загальні принципи проєктування програмного комплексу. Це створює основу для його подальшої розробки, тестування та впровадження та гарантує відповідність кінцевого продукту реальним потребам користувачів і освітнього середовища.

### 3.2 Формулювання правил класифікації студентів за критеріями на основі інформації про зміни їх успішності у попередніх семестрах

З метою прогнозування майбутнього стану успішності студентів та оцінки ймовірності успішного завершення курсу ними, доцільно розробити механізми класифікації, які на основі оцінок за попередні семестри дадуть змогу призначити кожному здобувачеві знань певну категорію. Зазначена категорія характеризуватиме навчальну траєкторію студента та дозволить чітко оцінювати, чи потребує він допомоги для її стабілізації, чи здатен і в подальшому демонструвати необхідний рівень успішності.

Для класифікації навчальних траєкторій здобувачів знань було запропоновано чотири категорії: стабільний, нестабільний, ризиковий, критичний. Розглянемо їх докладніше.

*Стабільний.* Категорія відображає студента з вирівняною та передбачуваною навчальною траєкторією. Її отримують ті, чиї оцінки демонструють мінімальний розкид навколо середнього значення, а стандартне відхилення обчислених семестрових середніх не перевищує декількох балів. Важливою умовою є також переважання стабільних станів у марковській моделі: для кожного рівня, до якого потрапляв студент, ймовірність залишитися в тому самому стані повинна бути достатньо високою. Поєднання

малої варіативності оцінок і високої «самоповертальності» у станах свідчить про надійність результатів студента у навчальному процесі.


Рисунок 3.1 – Категорії студентів відповідно до моделі

*Нестабільний*. Якщо у даних не спостерігається вираженої стабільності, але й не виявлено ознак ризиковості або критичного погіршення, студент автоматично потрапляє до групи нестабільних. Для таких випадків характерні незначні коливання між семестрами, відсутність стійкої тенденції, слабка змінність коефіцієнтів моделі або суперечливі переходи між станами. Відповідно, вона не є негативною категорією, а є радше сигналом про те, що траєкторія студента недостатньо структурована, щоб віднести її до стійких чи ризикових. Ця категорія визначається як базова й використовується у разі, коли інші критерії не спрацьовують.

*Ризиковий*. Категорія призначена для студентів, у яких проявляються тенденції до зниження успішності. Ознаками є переважання переходів на нижчі стани над переходами на вищі, причому різниця між кількістю «падінь» та «підйомів» має бути помітною. Ще одним визначальним параметром є

негативний тренд, виявлений за допомогою лінійної регресії. Поєднання цих факторів створює підстави для класифікації студента як такого, що перебуває у зоні ризику та потребує посиленої уваги або підтримки.

*Критичний.* Найвищий рівень занепокоєння, який позначає студентів із глибокими та послідовними негативними змінами. Визначальною ознакою є тривала серія погіршень: алгоритм фіксує підряд щонайменше три переходи до нижчих станів, що свідчить про швидку деградацію навчальної динаміки. Додатково враховується усереднена ймовірність переходу до нижчих станів у марковській моделі - якщо модель передбачає подальше погіршення з високою ймовірністю, студент потрапляє до критичної категорії. Такий випадок свідчить про суттєві ризики і потребує втручання або зовнішньої допомоги.

Таким чином, система класифікації студентів ґрунтується на поєднанні марковського моделювання, статистичного аналізу та оцінки динамічних характеристик навчальної успішності. Кожна категорія відображає певний тип академічного профілю, що дозволяє точніше інтерпретувати індивідуальні траєкторії навчання.

Підсумовуючи описані критерії, можна констатувати, що студенти, які демонструють невелику варіативність оцінок та високі ймовірності повернення до одного й того ж стану марківської моделі, класифікуються як стабільні - їхня успішність є передбачуваною та послідовною. Натомість група нестабільних охоплює широкий спектр випадків, у яких відсутня як чітка позитивна, так і негативна тенденція, що вказує на нерівномірний характер освітньої динаміки. Ризикова категорія формується тоді, коли поведінка студента демонструє тенденцію до погіршення. Критичний рівень відображає найбільш небезпечні ситуації, де спостерігається тривала серія спадів і висока ймовірність подальшого падіння згідно з марковською моделлю.

Сукупно класифікаційна схема дозволяє виявити закономірності в динаміці навчальних досягнень, визначити стабільність чи нестійкість академічної поведінки, відокремити студентів з потенційними ризиками та вчасно ідентифікувати критичні випадки.

На основі отриманих результатів адміністрація навчального закладу може робити висновки щодо студентів, які потребують уваги для зниження ризиків можливих відрахувань через академічну неуспішність. Ці висновки можуть бути використані для адаптації навчальних стратегій та цільового педагогічного супроводу.

### 3.2 Вимоги до інструментальних засобів розробки та обладнання для експлуатації програмного комплексу для прогнозування успішності здобувачів вищої освіти

У таблиці 3.1 описані апаратні вимоги, які висуваються для ефективної роботи програмного комплексу для прогнозування успішності здобувачів вищої освіти.

Таблиця 3.1 – Вимоги до апаратного забезпечення

| Характеристика | Вимоги |
|---|---|
| Процесор (CPU) | 2,0 GHz та більше |
| Оперативна пам'ять (RAM) | 1,5 ГБ та більше |
| Місце на жорсткому диску (HDD/SSD) | 100 МБ |
| Графічний дісплей | Роздільна здатність 1280*1024 та вище |
| Маніпулятори | Клавіатура та миша |
| Інтерфейси для інсталяції | USB, CD(DVD)-ROM або доступ до Інтернет |

У таблиці 3.2, своєю чергою описані програмні вимоги, які перелічують інструментарій розробки програмного комплексу для прогнозування успішності здобувачів вищої освіти, та перелік операційних систем для її експлуатації.

Таблиця 3.2 – Вимоги до інструментального забезпечення

| **Характеристика** | **Вимоги** |
|---|---|
| Мова програмування | C# 6.0 |
| Бібліотека для розробки | .NET Framework 4.7.2 |
| Бібліотека для візуалізації Марківських ланцюгів | Microsoft Automatic Graph Layout (MSAGL) [20] |
| GUI | WinForms [21] |
| СУБД | SQLite |
| Операційні системи | Windows 8 - Windows 11 |

Врахування зазначених вимог має забезпечити успішну розробку та експлуатацію програмного комплексу для прогнозування успішності здобувачів вищої освіти.

# 4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОГНОЗУВАННЯ УСПІШНОСТІ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ НА ОСНОВІ МАРКІВСЬКИХ ЛАНЦЮГІВ

## 4.1 Функціональна схема програмного комплексу для прогнозування успішності здобувачів вищої освіти

На рисунку 4.1 наведено функціональну схему програмного програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі марківських ланцюгів.

Подана схема відображає архітектуру системи. Архітектура поділена на три логічні модулі: візуальний інтерфейс, марківську модель та базу даних. Кожен модуль виконує власну групу функцій, але водночас тісно взаємодіє з іншими, утворюючи цілісну систему обробки, аналізу та інтерпретації даних.

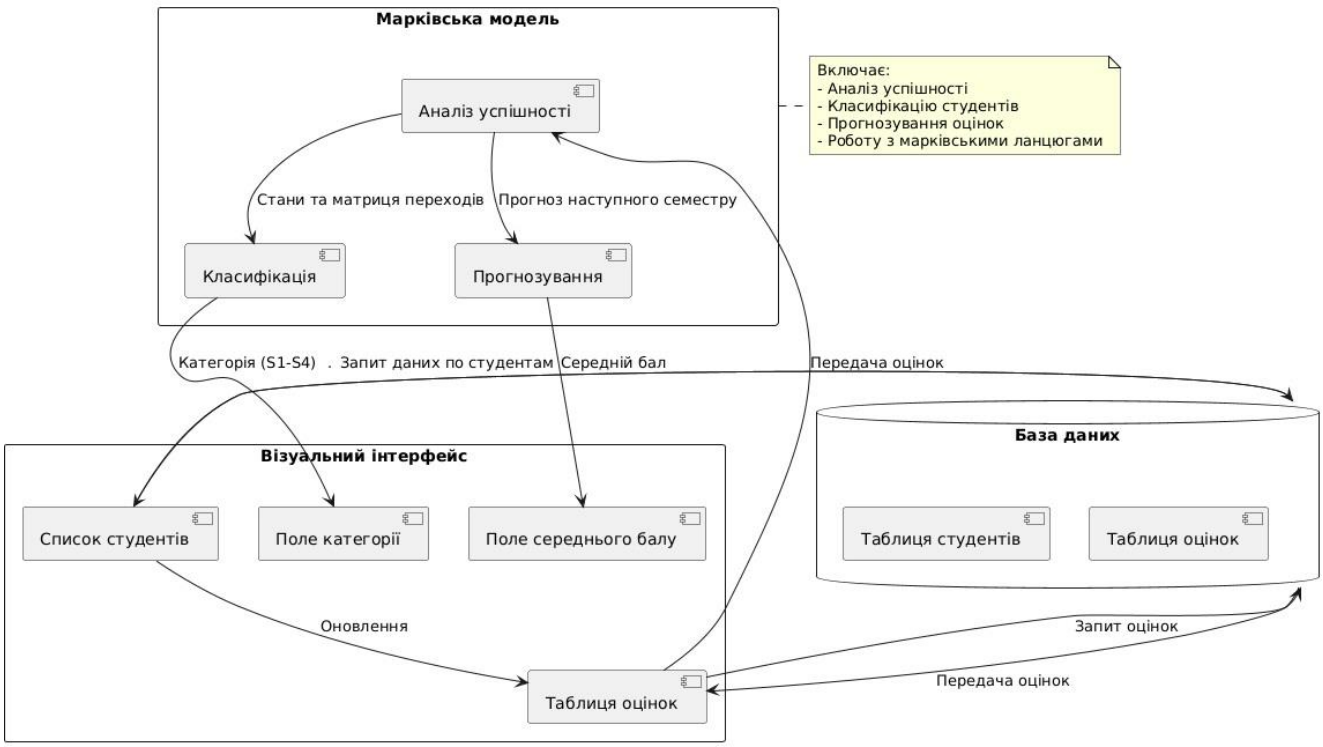



Рисунок 4.1 – Функціональна схема програмного програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі марківських ланцюгів

Візуальний інтерфейс є точкою входу та основним середовищем взаємодії користувача із системою. Він складається з кількох ключових елементів:

а) ***Список студентів***, що дозволяє обирати користувачу конкретного студента для проведення аналізу. Після вибору генерується запит до бази даних, з якої отримується інформація про відповідного студента.

б) ***Таблиця оцінок***, що відображає детальну матрицю оцінок студента по семестрах і предметах. Цей компонент автоматично оновлюється після вибору студента.

в) ***Поле категорії***, яке демонструє визначену категорію академічної стабільності (S1–S4).

г) ***Поле середнього балу***, що відображає прогнозоване значення середнього балу на наступний семестр.

Інтерфейс забезпечує швидкий доступ до структурованих даних, а також оновлення візуалізації після кожної зміни у виборі та при обробці інформації.

Марківська модель, своєю чергою, є налітичним ядром системи та реалізує основні алгоритми математичного аналізу успішності студентів. Вона фактично виконує роль інтелектуального центру, що обробляє дані й формує висновки. Марківська модель забезпечує науково обґрунтований підхід до оцінювання динаміки та прогнозування, що дозволяє виявляти приховані закономірності та тенденції в навчальних результатах студентів.

Її функціональні підсистеми забезпечують наступні можливості:

а) ***Аналіз успішності*** - на основі отриманих оцінок створює послідовності станів, класифікує середні бали по інтервалах, будує матрицю переходів між станами та обчислює статистичні показники динаміки.

б) ***Класифікація***, яка визначає категорію студента (S1–S4) на основі встановлених критеріїв, що включають стабільність успішності, коливання між інтервалами, наявність позитивного або негативного тренду та характер переходів між станами.

в) ***Прогнозування***, що використовує властивості марківських ланцюгів для визначення ймовірнісного розподілу станів у наступному семестрі та обчислює очікуваний середній бал.

База даних виступає джерелом інформації для системи. Вона містить дві основні таблиці:

а) ***Таблиця студентів***, що зберігає персональні дані, групу та ідентифікаційні ключі для кожного студента.

б) ***Таблиця оцінок***, яка містить структуровані дані про оцінки студентів за семестрами та предметами.

База даних забезпечує швидкий доступ до інформації про студентів, цілісність та узгодженість даних. Вона є важливим компонентом, що дозволяє системі працювати з великими обсягами інформації.

Завдяки такому підходу оцінювання студентів відбувається комплексно, з урахуванням як рівня знань, так і стабільності навчальної траєкторії, що забезпечує природніший розподіл між категоріями.

## 4.2 Алгоритм функціонування програмного комплексу для прогнозування успішності здобувачів вищої освіти

На рисунку 4.2 показаний алгоритм роботи програмного комплексу для прогнозування успішності здобувачів вищої освіти.

Програма розпочинає роботу з ініціалізації головної форми, у межах якої користувач отримує доступ до всіх функцій системи. Під час старту встановлюється з’єднання з базою даних, що забезпечує завантаження актуального списку студентів та дозволяє перейти до подальших обчислень і запитів.

Після вибору користувачем студента ідентифікатор останнього передається у модуль взаємодії з базою даних. Система формує цільовий SQL-запит, отримує весь масив оцінок, виконує їх структуризацію і впорядкування за семестрами. Дані перетворюються у внутрішній формат, оптимізований для аналітичного ядра програми.

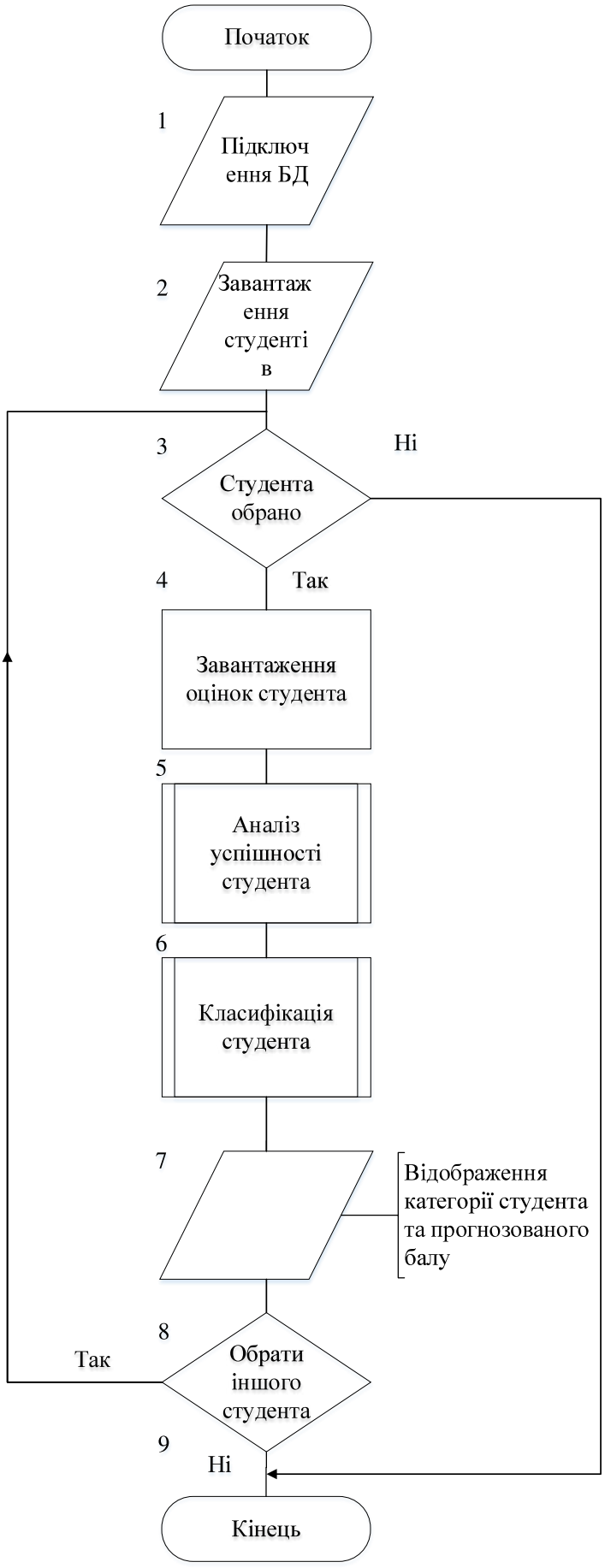



Рисунок 4.2 – Алгоритм роботи програмного засобу

Далі обчислюються середні значення балів студента за всі попередні навчальні періоди. Оцінки нормалізуються у фіксованому діапазоні, після чого кожен семестр переводиться у відповідний інтервал станів марківського ланцюга. Це дозволяє уніфікувати дані та підготувати їх до формування ймовірнісної моделі.

На основі дискретизованих значень будується матриця переходів, що відображає ймовірності зміни стану між суміжними семестрами. Матриця використовується для аналізу тенденцій, визначення стабільності й узгодженості академічної траєкторії, а також для прогнозування середнього балу на наступний семестр. Управляючий модуль проводить класифікацію студента за наперед визначеними критеріями, враховуючи поведінку в матриці переходів та інтенсивність змін оцінок.

Усі результати проведених обчислень передаються у візуальний інтерфейс. Таблиця оцінок оновлюється відповідно до отриманих даних, а категорія успішності й прогнозований середній бал виводяться у відповідних полях форми.

Програма підтримує розширену взаємодію з даними. Користувач може переглядати графіки змін середнього балу та аналізувати академічні результати студента у динаміці.

Для більшої наочності на рисунках 4.3 та 4.4 запропоновано більш детальну блок-схему алгоритму класифікації студента, яка детально описує блоки 4-7 алгоритму, що був наведений вище на рисунку 4.2. Перейдемо до її опису.

Процес класифікації починається з отримання повного набору оцінок студента за всі семестри. Для кожного навчального періоду обчислюється середній бал, після чого ці значення переводяться у дискретні стани відповідно до встановлених інтервалів успішності.

Така дискретизація формує послідовність станів, яка використовується для моделювання динаміки навчання.

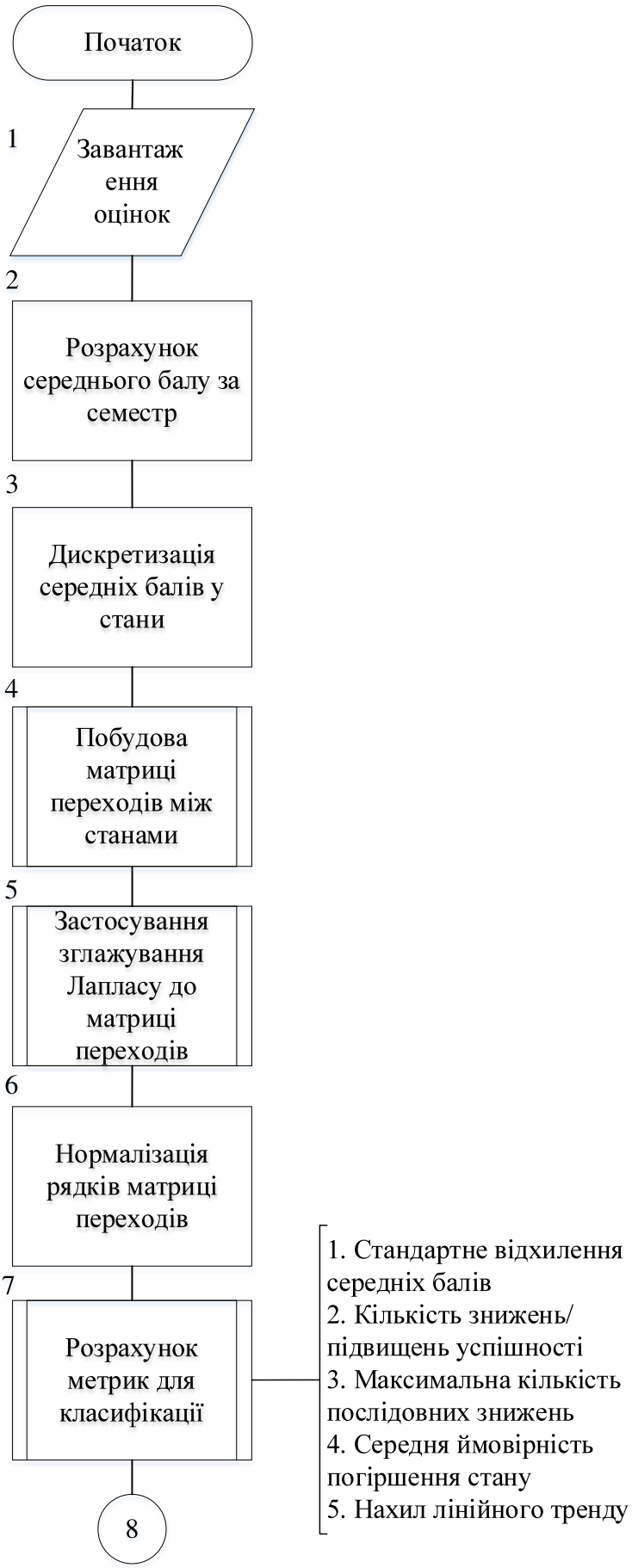



Рисунок 4.3 – Алгоритм класифікації студента за категоріями

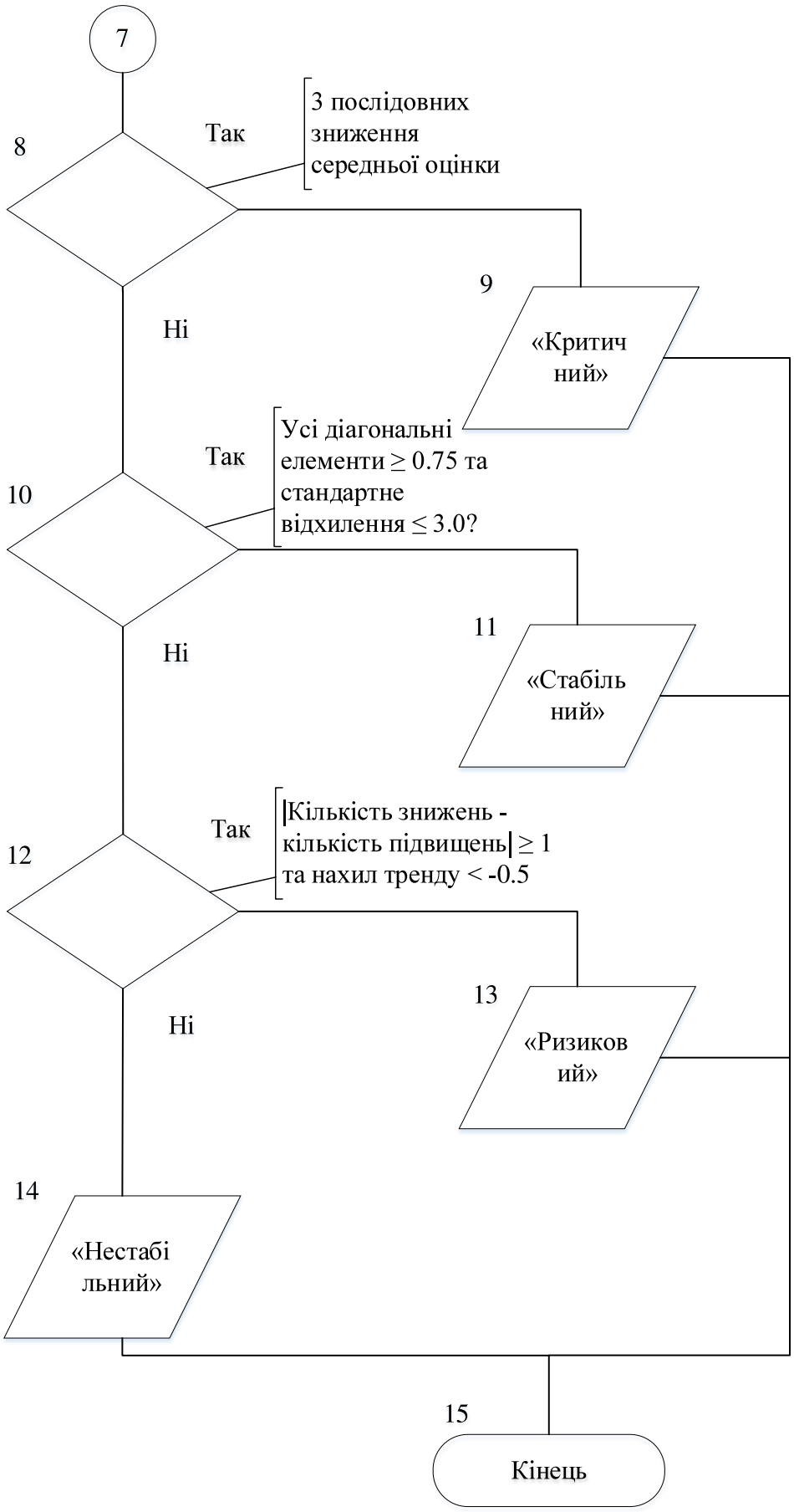



Рисунок 4.4 – Алгоритм класифікації студента за категоріями (продовження)

На основі цієї послідовності формується матриця переходів, що відображає частоти переходів між станами від одного семестру до наступного. Щоб уникнути нульових імовірностей та забезпечити коректність моделі, до матриці застосовується згладжування Лапласа [22, 23]. Після цього кожен рядок нормалізується, щоб імовірнісні значення коректно відображали структуру переходів і сумувалися до одиниці.

Після формування марківської моделі обчислюються основні показники, які відіграють роль у класифікації студента. Визначається стандартне відхилення середніх балів як показник стабільності. Підраховується кількість переходів, що ведуть до підвищення або зниження успішності, визначається максимальна довжина послідовних знижень, розраховується середня ймовірність погіршення стану на основі матриці переходів. Також оцінюється нахил тренду середніх балів, що відображає загальний напрямок змін успішності.

Після цього виконується послідовна перевірка критеріїв, що відповідають різним категоріям. Спершу оцінюється наявність критичного стану: перевіряється, чи є три й більше послідовних зниження і чи домінує ймовірність погіршення. Якщо ці умови виконуються, студент класифікується як S4 (критичний). Якщо ні - перевіряється відповідність категорії S1 (стабільний), яка передбачає низьку мінливість оцінок. У разі невідповідності здійснюється аналіз критеріїв для S3 (ризиковий), де визначальними є загальна тенденція до зниження та негативний нахил тренду. Якщо і ці умови не виконуються, студент автоматично відноситься до категорії S2, що позначає нестабільну, але не критичну динаміку.

Процес завершується в момент визначення відповідної категорії. Результат класифікації повертається в систему для подальшого використання у модулі відображення та в інших частинах програми, де ця інформація потрібна для аналізу та прийняття освітніх рішень.

### 4.3 Проектування бази даних програмного комплексу для прогнозування успішності здобувачів вищої освіти

У програмному забезпеченні для прогнозування академічної успішності успішності здобувачів вищої освіти застосовується реляційна база даних, створена на основі СУБД SQLite [24]. Вона використовується для зберігання основних інформаційних сутностей: студентів, їхніх навчальних груп і отриманих оцінок за різні семестри. База даних забезпечує централізоване зберігання даних, необхідних для виконання аналізу, побудови матриць переходів і класифікації студентів за марківськими моделями.

Використання SQLite у програмі дозволяє забезпечити простоту інтеграції, надійність зберігання даних і високу ефективність при обробці інформації.

Структуру БД наведено на рисунку 4.5.

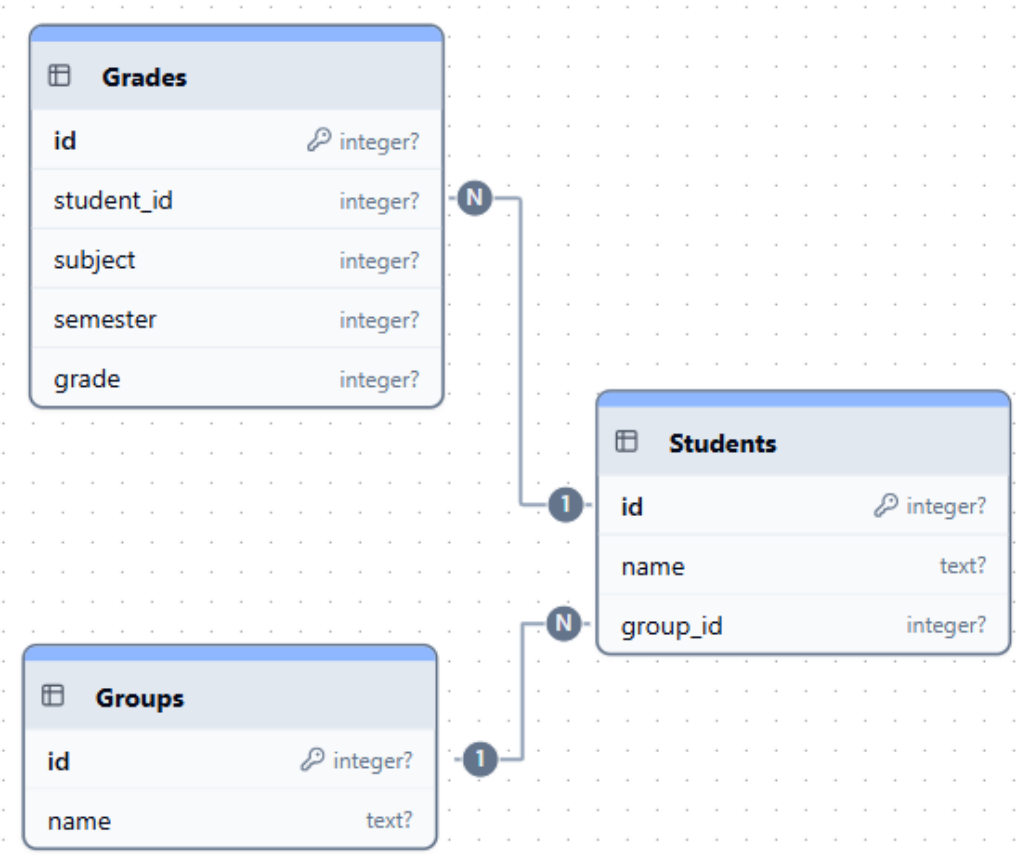



Рисунок 4.5 – Структура взаємозв’язку таблиць бази даних програмного комплексу для прогнозування успішності здобувачів вищої освіти

Таким чином, три таблиці утворюють цілісну структуру для зберігання навчальних даних: Groups містить назви академічних груп і слугує основою для логічного впорядкування студентів; Students прив’язує кожного студента до відповідної групи й забезпечує структуровану ієрархію; Grades накопичує інформацію про оцінки за предметами та семестрами, формуючи історію успішності, яка використовується для розрахунку середніх балів і подальшого аналізу в марківських моделях.

Зовнішні ключі між таблицями забезпечують логічну узгодженість: кожна оцінка прив’язана до конкретного студента, а кожен студент — до певної групи. Це дозволяє уникнути «висячих» записів і гарантує структурованість інформаційної моделі.

# 5 ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ПРОГНОЗУВАННЯ УСПІШНОСТІ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ НА ОСНОВІ МАРКІВСЬКИХ ЛАНЦЮГІВ

## 5.1 Приклади використання програмного комплексу для прогнозування успішності здобувачів вищої освіти

Зовнішній вигляд розробленого програмного комплексу наведено на рисунку 5.1.

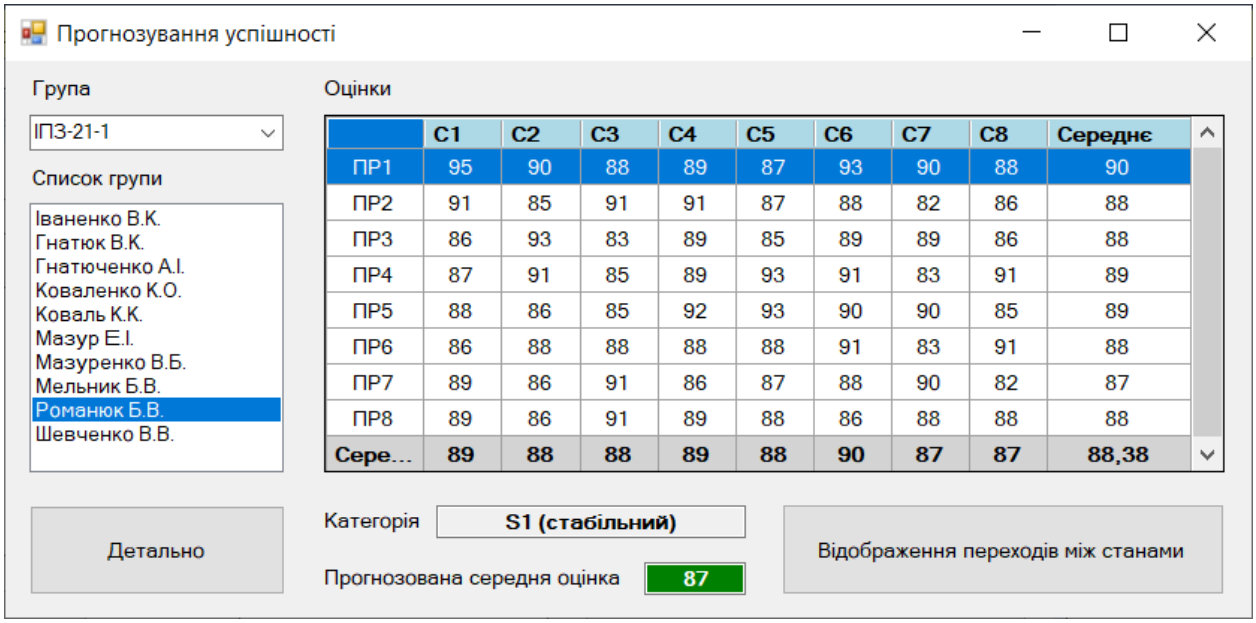



Рисунок 5.1 – Зовнішній вид головного вікна програмного комплексу для прогнозування успішності здобувачів вищої освіти (категорія «Стабільний»)

Інтерфейс програми забезпечує комфортну роботу з даними про академічну успішність. Головне вікно має наступну структуру: ліворуч розміщено навігаційну панель для швидкого переходу між групами та доступу до списку студентів, а з правої сторони - інтерактивна таблиця з оцінками, де дисципліни співвіднесені з відповідними семестрами.

Після вибору студента в переліку можна спостерігати динаміку зміни середнього балу та категорію його успішності.

У випадку виявлення порожніх полів або конфліктів у даних, програма автоматично повідомляє про це користувача через систему сповіщень.

Натискання кнопки «Детально» наводить дані по студенту у текстовому вигляді (рисунок 5.2).

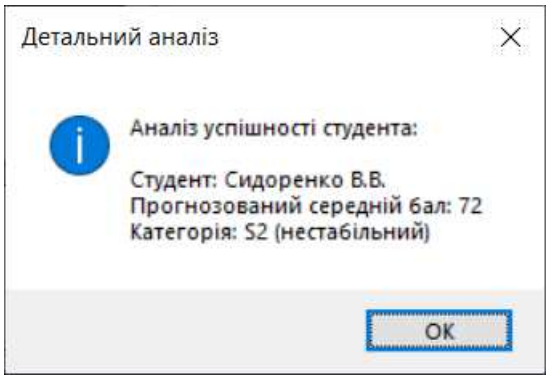



Рисунок 5.2 – Дані про студента

Як зазначалося вище, програма розподіляє студентів за чотирма категоріями, які описані в таблиці 5.1.

Таблиця 5.1 – Категорії, до яких віднесені студенти в результаті класифікації

| Шифр | Назва | Сенс |
|---|---|---|
| S1 | Стабільний | Розкид середніх балів за попередні семестри знаходиться у вузькому діапазоні |
| S2 | Нестабільний | Немає чітко вираженої динаміки змін |
| S3 | Ризиковий | Кількість негативних змін станів академічної успішності переважає кількість позитивних змін |
| S4 | Критичний | Протягом принаймні трьох семестрів спостерігається погіршення станів академічної успішності студента |

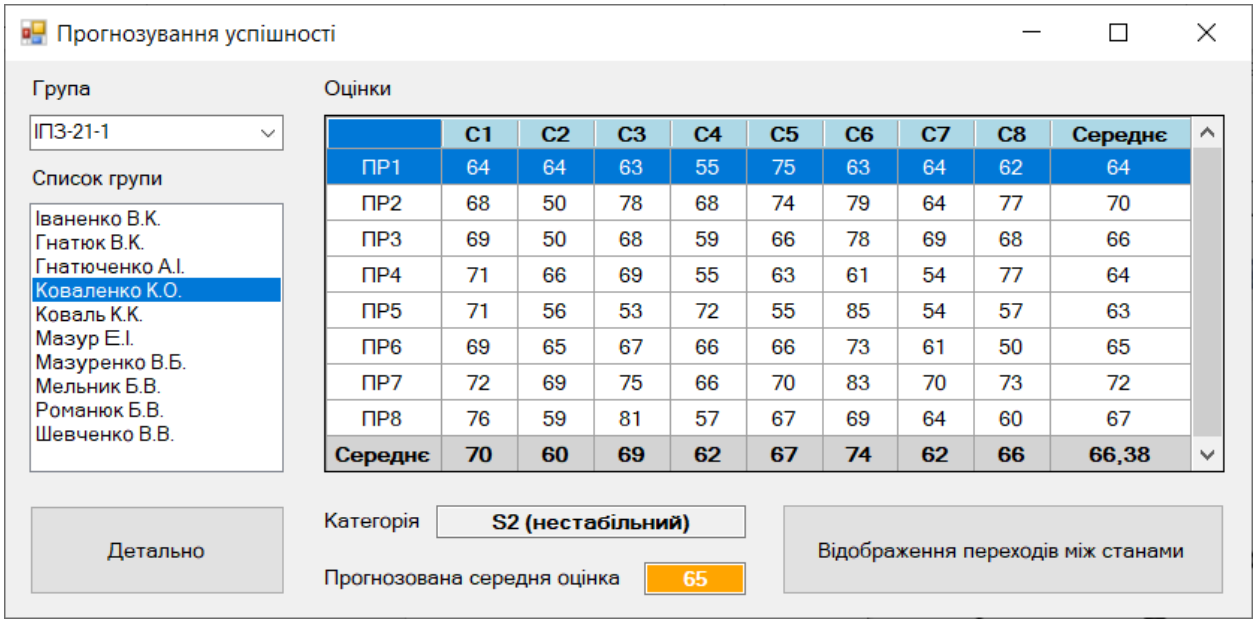
Під станами академічної успішності варто розуміти оцінку в рамках Болонської системи [25], яка відповідає середньому балу, отриманому студентом за певний семестр (таблиця 5.2).

Таблиця 5.2 – Стани академічної успішності студентів

| Шифр | Діапазон балів | Оцінка згідно Болонської системи |
|---|---|---|
| S0 | 90-100 | A |
| S1 | 81-89 | B |
| S2 | 71-80 | C |
| S3 | 61-70 | D |
| S4 | 50-60 | E |

Таким чином, прогнозований середній бал та категорія студента розраховуються на базі побудованого Марківського ланцюгу та обчислених ймовірностей переходів між його станами.

На рисунку 5.3 показано студента, якого було віднесено до категорії «Нестабільний». Як видно з його оцінок, в середніх балах за семестр не спостерігається чіткої тенденції в їх зміні.




| | С1 | С2 | С3 | С4 | С5 | С6 | С7 | С8 | Середнє |
|---|---|---|---|---|---|---|---|---|---|
| ПР1 | 64 | 64 | 63 | 55 | 75 | 63 | 64 | 62 | 64 |
| ПР2 | 68 | 50 | 78 | 68 | 74 | 79 | 64 | 77 | 70 |
| ПР3 | 69 | 50 | 68 | 59 | 66 | 78 | 69 | 68 | 66 |
| ПР4 | 71 | 66 | 69 | 55 | 63 | 61 | 54 | 77 | 64 |
| ПР5 | 71 | 56 | 53 | 72 | 55 | 85 | 54 | 57 | 63 |
| ПР6 | 69 | 65 | 67 | 66 | 66 | 73 | 61 | 50 | 65 |
| ПР7 | 72 | 69 | 75 | 66 | 70 | 83 | 70 | 73 | 72 |
| ПР8 | 76 | 59 | 81 | 57 | 67 | 69 | 64 | 60 | 67 |
| **Середнє** | **70** | **60** | **69** | **62** | **67** | **74** | **62** | **66** | **66,38** |

Рисунок 5.3 – Успішність студента, віднесеного до категорії «Нестабільний»

При натисканні на кнопку «Відображення переходів між станами» демонструється окреме вікно з візуалізацією Марківського ланцюга та з вірогідностями переходів між станами академічної успішності обраного студента (рисунок 5.4).

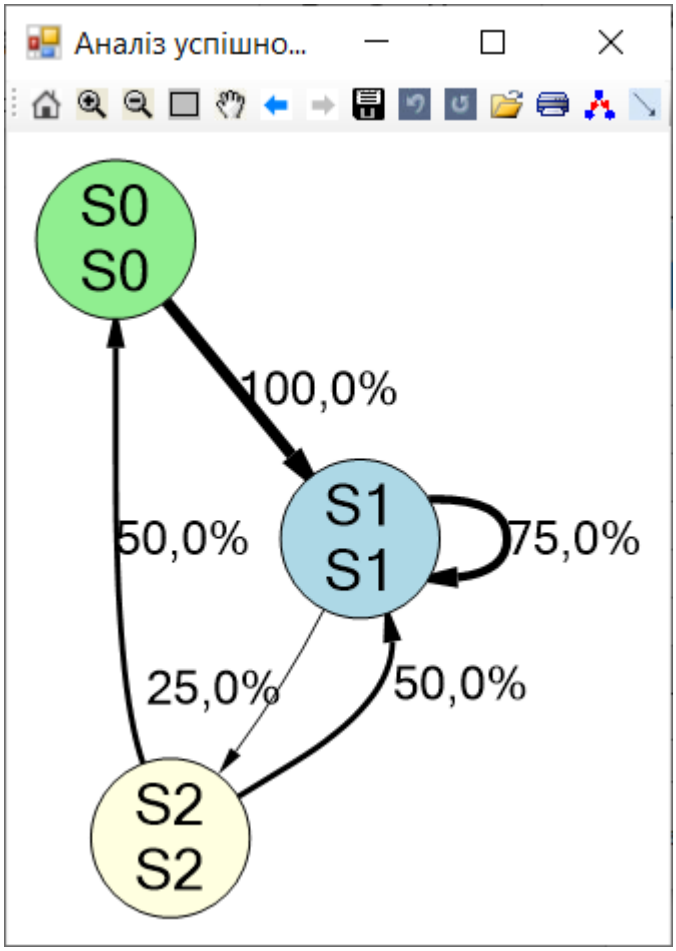


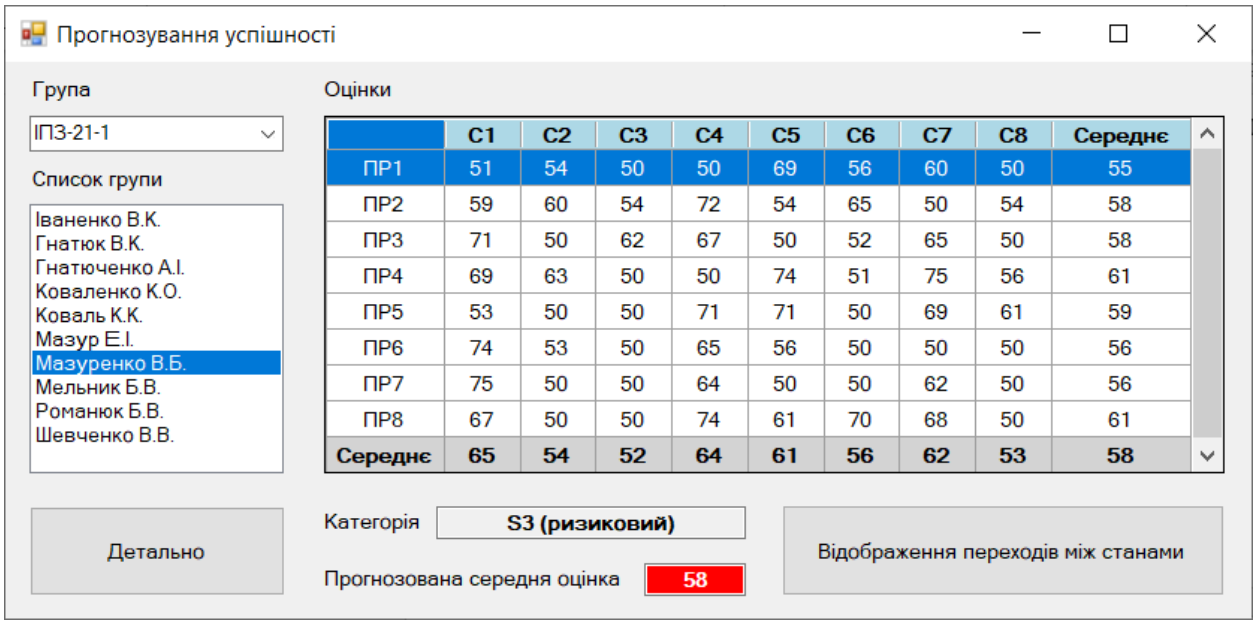

Рисунок 5.4 – Побудова Марківського ланцюга для студента, віднесеного до категорії «Нестабільний»

На рисунку 5.4 наведено стани академічної успішності, в яких перебував студент, та обчислені вірогідності їх зміни на інші (або залишення в поточному стані) протягом одного кроку.

Для візуалізації Марківських ланцюгів використовувалася бібліотека Microsoft Automatic Graph Layout (MSAGL).

Рисунок 5.5 демонструє оцінки студента, віднесеного до категорії «Ризиковий». Отримання такого результату свідчить про те, що кількість

спадів по академічним станам студента перевищує кількість «покращень» цих станів.




| | С1 | С2 | С3 | С4 | С5 | С6 | С7 | С8 | Середнє |
|---|---|---|---|---|---|---|---|---|---|
| ПР1 | 51 | 54 | 50 | 50 | 69 | 56 | 60 | 50 | 55 |
| ПР2 | 59 | 60 | 54 | 72 | 54 | 65 | 50 | 54 | 58 |
| ПР3 | 71 | 50 | 62 | 67 | 50 | 52 | 65 | 50 | 58 |
| ПР4 | 69 | 63 | 50 | 50 | 74 | 51 | 75 | 56 | 61 |
| ПР5 | 53 | 50 | 50 | 71 | 71 | 50 | 69 | 61 | 59 |
| ПР6 | 74 | 53 | 50 | 65 | 56 | 50 | 50 | 50 | 56 |
| ПР7 | 75 | 50 | 50 | 64 | 50 | 50 | 62 | 50 | 56 |
| ПР8 | 67 | 50 | 50 | 74 | 61 | 70 | 68 | 50 | 61 |
| **Середнє** | **65** | **54** | **52** | **64** | **61** | **56** | **62** | **53** | **58** |

Рисунок 5.5 – Успішність здобувача, віднесеного до категорії «Ризиковий»

На рисунку 5.6, своєю чергою, показано ланцюг, що характеризує ймовірності переходів для вищевказаного здобувача. Тут маємо тільки два стани через те, що студент має середні оцінки D та E, віднесені категорій S0 та S1 відповідно.

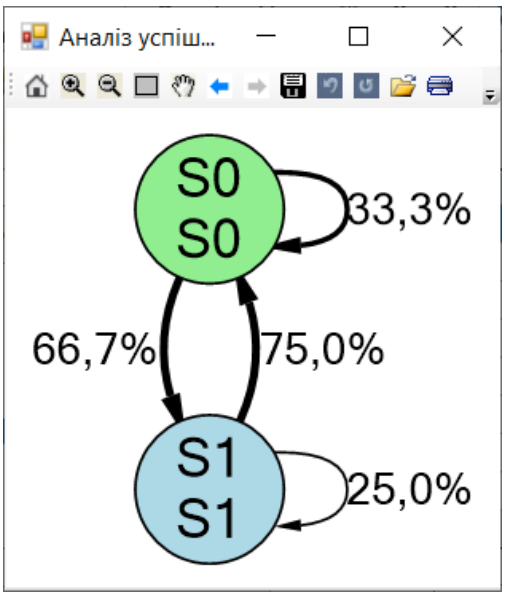
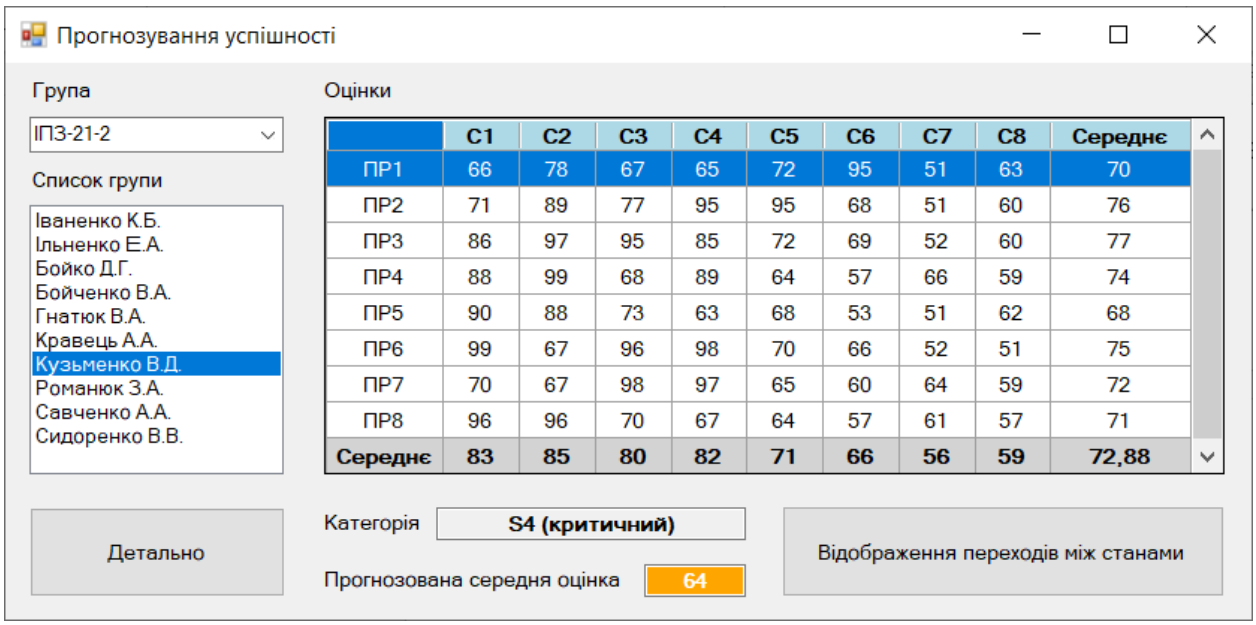



Рисунок 5.6 – Побудова Марківського ланцюга для студента, віднесеного до категорії «Ризиковий»

Насамкінець, розглянемо випадок для здобувача категорії «Критичний» (рисунок 5.7) та Марківский ланцюг з вірогідностями зміни станів (рисунок 5.8).




| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | Середнє |
|---|---|---|---|---|---|---|---|---|---|
| ПР1 | 66 | 78 | 67 | 65 | 72 | 95 | 51 | 63 | 70 |
| ПР2 | 71 | 89 | 77 | 95 | 95 | 68 | 51 | 60 | 76 |
| ПР3 | 86 | 97 | 95 | 85 | 72 | 69 | 52 | 60 | 77 |
| ПР4 | 88 | 99 | 68 | 89 | 64 | 57 | 66 | 59 | 74 |
| ПР5 | 90 | 88 | 73 | 63 | 68 | 53 | 51 | 62 | 68 |
| ПР6 | 99 | 67 | 96 | 98 | 70 | 66 | 52 | 51 | 75 |
| ПР7 | 70 | 67 | 98 | 97 | 65 | 60 | 64 | 59 | 72 |
| ПР8 | 96 | 96 | 70 | 67 | 64 | 57 | 61 | 57 | 71 |
| **Середнє** | **83** | **85** | **80** | **82** | **71** | **66** | **56** | **59** | **72,88** |

Рисунок 5.7 – Успішність студента, віднесеного до категорії «Критичний»

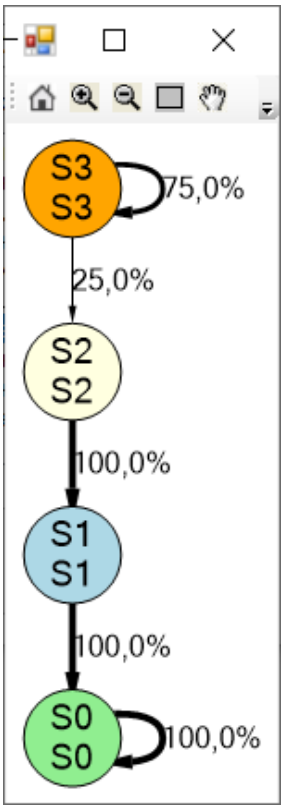



Рисунок 5.8 – Побудова Марківського ланцюга для студента, віднесеного до категорії «Критичний»

Зазначена категорія була призначена студенту через послідовне трикратне погіршення станів академічної успішності. Відповідно, на здобувачів такої категорії деканат має звертати пильну увагу та рекомендувати кураторам здійснювати певні заходи за для стабілізації академічної траекторії.

Варто зазначити, що для категорій «Стабільний» марківскький ланцюг може складатись не більше ніж з двох станів (або взагалі з одного). Також варто зауважити, що категорія «Нестабільний» не несе в собі негативного змісту, а лише наголошує на відсутності певної системи в оцінках здобувача та неможливості віднесення його до трьох інших категорій.

# ВИСНОВКИ

У ході виконання дослідження було створено програмний комплекс для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів, який пропонує можливості обробки та математичного аналізу даних про академічну успішність студентів. Реалізована модель на основі марковських ланцюгів дозволила інтерпретувати зміну рівня успішності як послідовність станів, що еволюціонують у часі, що забезпечило формування матриць переходів та визначення індивідуальних академічних траєкторій здобувачів.

Розроблена база даних у поєднанні з інструментами обробки семестрових оцінок забезпечила функціонування алгоритмів, а застосування процедур нормалізації даних та дискретизації балів підвищило коректність створеної моделі. Використання бібліотеки MSAGL забезпечило наочну візуалізацію переходів між станами, що значно полегшило інтерпретацію результатів і дозволило виявити характерні для студентів тенденції.

У ході експериментів було встановлено, що класифікація студентів за категоріями S1–S4 дає змогу швидко оцінювати загальний навчальний стан та визначати групи ризику.

Програмне забезпечення продемонструвало високу ефективність під час аналізу великих масивів даних, а його інтерфейс забезпечив зручну взаємодію працівників навчального закладу із результатами моніторингу.

Запропонований підхід у подальшому може бути використаний не лише для індивідуального аналізу студентів, а й для оцінки ефективності викладання дисциплін, планування навчального навантаження чи прогнозування навчальних результатів. Підтримка .NET Framework та можливість подальшого розширення функціоналу роблять систему придатною для інтеграції у внутрішню IT-інфраструктуру багатьох закладів освіти.

У підсумку, розроблений програмний комплекс для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів

підтвердив свою практичну цінність як інструмент підтримки прийняття рішень у сфері освіти та продемонстрував, що використання Марковських моделей та засобів візуалізації дозволяє досягти більш глибокого та об’єктивного розуміння процесів формування академічної успішності.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Meyn S. P., Tweedie R. L. Markov Models. Markov Chains and Stochastic Stability. London, 1993. P. 23–53. URL: https://doi.org/10.1007/978-1-4471-3267-7_2

2. Косенко Ю., Носов П., Яковенко Є. Використання ланцюгів Маркова для прогнозування колективної мотивації студентів. *EEJET*. 2010. Т. 3, № 4(45). С. 30–32.

3. Backenköhler M., Wolf V. Student Performance Prediction and Optimal Course Selection: An MDP Approach. *Software Engineering and Formal Methods*. Cham, 2018. P. 40–47. URL: https://doi.org/10.1007/978-3-319-74781-1_3

4. Brezavšček A., Bach M. P., Baggia A. Markov Analysis of Students' Performance and Academic Progress in Higher Education. *Organizacija*. 2017. Vol. 50, no. 2. P. 83–95. URL: https://doi.org/10.1515/orga-2017-0006

5. Using Markov chain model to evaluate medical students' trajectory on progress tests and predict USMLE step 1 scores - a retrospective cohort study in one medical school / L. Wang et al. *BMC Medical Education*. 2021. Vol. 21, no. 1. URL: https://doi.org/10.1186/s12909-021-02633-8

6. Gardner J., Brooks C. Student success prediction in MOOCs. *User Modeling and User-Adapted Interaction*. 2018. Vol. 28, no. 2. P. 127–203. URL: https://doi.org/10.1007/s11257-018-9203-z

7. Baum-Welch Algorithm. *Encyclopedia of Machine Learning and Data Mining*. Boston, MA, 2017. P. 99. URL: https://doi.org/10.1007/978-1-4899-7687-1_59

8. Lagoudakis M. G., Zeugmann T., Sammut C. Viterbi Algorithm. *Encyclopedia of Machine Learning*. Boston, MA, 2011. P. 1025. URL: https://doi.org/10.1007/978-0-387-30164-8_878

9. Lončarević V., Lekić V., Damljanović N. Predicting Student Academic Success with Hidden Markov Models. Proceedings TIE 2024. 2024. P. 68–73. URL: https://doi.org/10.46793/tie24.068l

10. Finite Mixture Modeling. *Finite Mixture and Markov Switching Models*. New York, 2006. P. 1–23. URL: https://doi.org/10.1007/978-0-387-35768-3_1

11. Modeling and predicting students' engagement behaviors using mixture Markov models / R. Maqsood et al. Knowledge and Information Systems. 2022. URL: https://doi.org/10.1007/s10115-022-01674-9

12. IBM SPSS Modeler [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/products/spss-modeler

13. Altair RapidMiner [Електронний ресурс] – Режим доступу до ресурсу: https://rapidminer.com

14. Power BI [Електронний ресурс] – Режим доступу до ресурсу: https://powerbi.microsoft.com

15. Orange Data Mining [Електронний ресурс] – Режим доступу до ресурсу: https://orangedatamining.com

16. Відибіда О. К. Стохастичні моделі. Київ, 2006. 204 с.

17. The Markov chain. Bristol : Shearsman Books, 2017. 66 p.

18. Bulla J., Berzel A. Computational issues in parameter estimation for stationary hidden Markov models. *Computational Statistics*. 2007. Vol. 23, no. 1. P. 1–18. URL: https://doi.org/10.1007/s00180-007-0063-y

19. Прихована марковська модель [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Прихована_марковська_модель

20. Microsoft Automatic Graph Layout (MSAGL) [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/project/microsoft-automatic-graph-layout/

21. Windows Forms Overview [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/

22. MacKay D. J. C. Information Theory, Inference, and Learning Algorithms. Cambridge : Cambridge University Press, 2003. 628 p. ISBN 978-0-521-64298-9.

23. Jurafsky, D., Martin, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd ed. Upper Saddle River : Pearson Education, 2023.

24. SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://sqlite.org/
25. Bologna Process [Електронний ресурс] – Режим доступу до ресурсу: https://ehea.info/
26. Шамрай О.В., Шаповалова Н. Н. Методичні вказівки до виконання індивідуальних завдань з дисципліни «Інноваційний менеджмент в індустрії програмного забезпечення» для студентів усіх форм навчання за спеціальністю 121 - «Інженерія програмного забезпечення». Криворізький національний університет. 37 с.
27. Варстість електроенергії [Електронний ресурс] – Режим доступу до ресурсу: https://www.novakom.com.ua/tarifs/cf/102.html

# Додаток А – Розрахунок економічної ефективності впровадження програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів

Щоб точно визначити собівартість створення програмного комплексу, необхідно комплексно проаналізувати всі витратні статті, пов'язані з його розробкою, запуском і подальшою підтримкою. До початкових витрат зазвичай відносять роботи зі створення та впровадження системи, придбання необхідного обладнання, програмних ліцензій та використання спеціалізованих інструментів. Далі враховуються оперативні витрати, які охоплюють технічне обслуговування, оновлення, виправлення помилок та оплату роботи персоналу, відповідального за стабільне функціонування програмного комплексу. Окрему частину становлять витрати на підготовку команди – навчальні програми, тренінги та інші заходи, що допомагають ефективно управляти системою.

Вартість системи також розподіляється в часі через амортизацію, що дозволяє визначити середні щорічні витрати на її використання протягом усього терміну служби. До загальної суми додаються вкладення в регулярну підтримку та оновлення програмного забезпечення, а також інші можливі витрати, пов'язані з організацією проєкту – маркетингові заходи, юридичний супровід чи адміністративні послуги. Після обліку всіх цих складових формується повна собівартість програмного комплексу, яку варто коригувати з огляду на зміну цін і зовнішніх умов у перспективі.

Обчислення економічної ефективності розроблюваного програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів здійснювалося згідно відповідних методичних матеріалів [26].

Розрахуємо собівартість створеного програмного комплексу. Для проведення розрахунку використовувалися наступні дані, що наведені у таблиці А.1.

Таблиця А.1 – Стартові показники для розрахунку собівартості програмного комплексу для прогнозування успішності здобувачів вищої освіти

| Найменування | Показник |
|---|---|
| Складність створення програмного комплексу, днів | 45 |
| Ставка програміста (місяць), грн | 35000 |
| Кількість годин в місяці, год | 160 |
| Додаткова зарплатня (%) | 10 |
| Відрахування до соц. фондів (%) | 15 |
| Загальновиробничі витрати компанії (%) | 100 |
| ПДВ (%) | 20 |

При розрахунку вважатимемо, що в одному календарному місяці налічується двадцять робочих днів.

а) Витрати на комплектуючі – оптичні диски для дистрибутивів програмного комплексу:

$$З_k = \sum Ц_k * n_k \qquad (А.1)$$

де $З_К$ – затрати на оптичні диски, грн.;

$Ц_К$ – вартість за 1 оптичний диск, грн.;

$n_k$ – загальна кількість оптичних дисків, шт.

Таким чином:

$$З_k = 25*10 = 250 грн \quad (А.2)$$

Затрати на оптичні диски дорівнюють 250 грн.

б) Затрати на е/е розрахуємо за виразом А.3:

$$B_E = P_E \sum W_i * t_i \quad (А.3)$$

де $P_E$ – поточна вартість одного кВт-год, грн.;

*Wi* – середня прогнозована потужність обладнання, яку буде демонструвати обладнання під час роботи.

Вартість одного кіловату електроенергії в 2025 році становить 4,32 грн [27].

Таким чином, згідно виразу А.4:

$$B_E = 4,32*0,85*160 = 587,52 грн \quad (А.4)$$

витрати електроенергії складуть 587,52 грн.

в) Зарплата обчислюється за виразом А.5:

$$З_{осн} = l_{год} * T_{год} \quad (А.5)$$

де $l_{год}$ – це тарифна ставка виконавця робіт, грн.;

$T_{год}$ – 160 годин – робочі години протягом місяця.

Тарифна ставка виконавця робіт, що створює програмний комплекс, дорівнює 218,75 грн.

Обчислимо зарплату виконавця робіт за виразом А.6:

$$З_{осн} = 218,75*160 = 35000 грн \quad (А.6)$$

Зарплата виконавця робіт складає 35000 грн.

г) Додаткова зарплата обчислюється за виразом А.7:

$$З_{дод} = \frac{З_{осн} * Д\%}{100} \quad (А.7)$$

де $Здод$ – додаткова зарплата виконавця робіт, грн.;

*Д%* – відсоток додаткової зарплати виконавця робіт, що складає десять відсотків.

Обчислимо додаткову зарплату за виразом А.8:

$$З_{дод} = \frac{35000*10}{100} = 3500 грн \quad (А.8)$$

Додаткова зарплата складе 3500 грн.

д) Обсяги відрахувань до соціальних фондів розраховуються за виразом А.9:

$$З_{соц} = \frac{(З_{осн} + З_{дод})*С\%}{100} \quad (А.9)$$

де *Зсоц* – обсяг відрахувань до соціальних фондів, грн.;

*С%* – відсоток необхідних відрахувань у соціальні фонди, що складає 15%.

Розрахуємо відрахування в соціальні фонди за виразом А.10:

$$З_{соц} = \frac{(35000 + 3500) * 15}{100} = 5775 грн \quad (А.10)$$

Відрахування до соціальних фондів дорівнює 5775 грн;

е) Загальновиробничі затрати розраховуються за виразом А.11:

$$З_{заг} = \frac{З_{осн} * Н_1 \%}{100} \quad (А.11)$$

де $З_{заг}$ – загальновиробничі затрати компанії, грн.;

$Н_1 \%$ – планований % загальновиробничих затрат (100%);

Обчислимо загальновиробничі затрати за виразом А.12:

$$З_{заг} = \frac{35000 * 100}{100} = 35000 грн \quad (А.12)$$

Загальновиробничі затрати компанії дорівнюють 35000,00 грн.

Обчислимо виробничу собівартість, провівши адитивні операції за попередніми розрахунками (формула А.13).

$$S_{пп} = 250 + 587,52 + 35000 + 3500 + 5775 + 35000 = 80112,52 грн \quad (А.13)$$

де $S_{пп}$ – розрахована виробнича собівартість, грн.

Отже, виробнича собівартість дорівнює 80112,52 грн.

В представленій таблиці А.2 запропоновано планову калькуляцію щодо виробничої собівартості, та загальної вартості розробки програмного комплексу.

Створення програмного комплексу включало роботи, на виконання яких було потрібно 45 робочих днів.

Таблиця 6.2 – Демонстрація планової калькуляції

| Пункти калькуляції | Сума, грн. |
|---|---|
| CD-диски (Комплектуючи вироби) | 250 |
| Кошти на електроенергію: | 587,52 |
| Основна зар. плата | 35000 |
| Додаткова зар. плата | 3500 |
| Розмір відрахувань в соц. фонди | 5775 |
| Загальновиробничі витрати підприємства | 35000 |
| Виробнича собівартість в розрахунку на 1 місяць | 80112,52 |
| Собівартість розробки системи багатофакторного оцінювання навчальних досягнень. | 180235,17 |

Отже, собівартість створеного програмного комплексу для прогнозування успішності здобувачів вищої освіти складає 180235,17 грн.

Для розрахунку економічного ефекту від застосування програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів можна використати наступне математичне рівняння:

$$E_{еф} = V * Q \qquad \text{(А.13)}$$

де *Еефф* – економічна ефективність розробки програмного комплексу для прогнозування успішності здобувачів вищої освіти;

*V* – оціночний економічний ефект з розрахунку на одного студента, який формується на основі очікуваної користі від раннього виявлення студентів, які потребують уваги через зниження навчального рівня;

*Q* – загальна кількість студентів в закладі вищої освіти, успішність яких аналізує зазначений програмний комплекс.

Цей підхід дозволяє оцінити загальний економічний вплив системи, виходячи з припущення, що кожен заклад вищої освіти отримує певну вартісну користь від використання програмного комплексу.

Однак, важливо зазначити, що реальний економічний ефект може бути складно точно оцінити через різноманітність факторів та необхідність урахування індивідуальних відмінностей студентів, а також динаміки ринку освітніх послуг.

Також слід врахувати, що ці фактори можуть змінюватися в часі та в залежності від специфіки галузі.

Обчислення ефективності створеного програмного комплексу прогнозування успішності здобувачів вищої освіти можливе за умови визначення зазначених вище змінних. Їх значення докладно представлені в таблиці А.3.

Таблиця А.3 — Значення змінних для обчислення економічної ефективності від впровадження програмного комплексу

| Параметр | Значення |
|---|---|
| Вартісний ефект на одного студента на місяць | 100 |
| Загальна кількість студентів | 370 |

Для розрахунку терміну окупності програмного комплексу необхідно використати формулу, яка виглядає наступним чином:

$$T = \frac{C}{E_{еф}} \tag{А.15}$$

де Т – термін окупності програмного комплексу прогнозування успішності здобувачів вищої освіти, міс.;

С – загальні витрати на розробку та впровадження включають всі витрати, пов'язані з проектуванням, розробкою, тестуванням, впровадженням та підтримкою системи, грн.,

$Е_{еф}$ – місячний економефект, грн.

З огляду на це, економічний ефект від впровадження програмного комплексу складе

$$Еефф = 100 * 370 = 37000 \text{ грн} \quad (А.16)$$

Виконаємо розрахунок терміну окупності програмного комплексу згідно виразу:

$$Т = 180235,17/37000 \approx 4,87 \text{ місяця} \quad (А.17)$$

Таким чином, собівартість програмного комплексу для прогнозування успішності здобувачів вищої освіти дорівнює 180235,17 грн, економічний ефект від її впровадження складає 37000 грн на місяць, витрачені кошти на розробку окупляться приблизно протягом п’яти місяців. З огляду на це, можна відзначити доцільність роботи над програмним комплексом для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів і його впровадження, а також доцільність використання зазначених програмних засобів в умовах вищих навчальних закладів.

# Додаток Б – Вихідний код програмного комплексу для прогнозування успішності здобувачів вищої освіти на основі Марківських ланцюгів

***Form.cs***

```
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using System.Collections.Generic;

namespace MarkovChainStudentSuccess
{
    public partial class Form1 : Form
    {
        private DataTable studentsData;
        private int currentStudentId = -1;


        public Form1()
        {
            InitializeComponent();
            InitializeDatabase();
            LoadGroups();
            SetupDataGridView();
            SetupTextFields();


            textBoxCategory.ReadOnly = true;
            textBoxCategory.TextAlign =
HorizontalAlignment.Center;
            textBoxCategory.Font = new
Font(textBoxCategory.Font, FontStyle.Bold);
        }

        private void SetupTextFields()
        {
            // Налаштування текстових полів
            textBoxGrade.ReadOnly = true;
            textBoxGrade.TextAlign = HorizontalAlignment.Center;
            textBoxGrade.Font = new Font(textBoxGrade.Font,
FontStyle.Bold);
        }

        private void InitializeDatabase()
        {
```

```
            try
            {
                DB.InitializeDatabase();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка ініціалізації бази
даних: {ex.Message}",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void LoadGroups()
        {
            try
            {
                var groups = DB.GetGroups();
                comboBoxGroups.DataSource = groups;
                comboBoxGroups.DisplayMember = "Name";
                comboBoxGroups.ValueMember = "Id";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження груп:
{ex.Message}",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void SetupDataGridView()
        {
            dataGridViewGrades.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
            dataGridViewGrades.AllowUserToAddRows = false;
            dataGridViewGrades.RowHeadersVisible = false;
            dataGridViewGrades.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        }

        private void comboBoxGroups_SelectedIndexChanged(object
sender, EventArgs e)
        {
            if (comboBoxGroups.SelectedValue != null &&

int.TryParse(comboBoxGroups.SelectedValue.ToString(), out int
groupId))
            {
                LoadStudents(groupId);
            }
        }
```

```
        private void LoadStudents(int groupId)
        {
            try
            {
                studentsData = DB.GetStudents(groupId);
                listBoxStudents.DataSource = studentsData;
                listBoxStudents.DisplayMember = "Name";
                listBoxStudents.ValueMember = "Id";

                if (studentsData.Rows.Count > 0)
                {
                    listBoxStudents.SelectedIndex = 0;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження
студентів: {ex.Message}",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

                        private void
listBoxStudents_SelectedIndexChanged(object sender, EventArgs e)
                        {
                            if (listBoxStudents.SelectedItem ==
null)
                            {
                                textBoxGrade.Text =
string.Empty;
                                textBoxCategory.Text =
string.Empty;
                                dataGridViewGrades.DataSource =
null;
                                return;
                            }

                            var selectedRow =
(listBoxStudents.SelectedItem as DataRowView)?.Row as DataRow;
                            if (selectedRow == null) return;

                            int studentId =
Convert.ToInt32(selectedRow["Id"]);
                            currentStudentId = studentId; //
Оновлюємо поточний ID студента

                            try
                            {
                                // Завантажуємо та відображаємо
оцінки в сітці
```

```
                                LoadStudentGrades(studentId);

                                // Отримуємо оцінки для аналізу
                                var grades =
DB.GetStudentGrades(studentId);
                                var gradesBySubject = new
Dictionary<string, List<int>>();

                                // Групуємо оцінки за предметами
                                foreach (DataRow row in
grades.Rows)
                                {
                                    string subject =
row["subject"].ToString();
                                    if
(!gradesBySubject.ContainsKey(subject))
                                    {
                                        gradesBySubject[subject]
= new List<int>();
                                    }

                                    if (row["grade"] !=
DBNull.Value && int.TryParse(row["grade"].ToString(), out int
grade))
                                    {

gradesBySubject[subject].Add(grade);
                                    }
                                }

                                // Виконуємо аналіз
                                if (gradesBySubject.Count > 0)
                                {
                                    // Обчислюємо загальний
середній бал
                                    double totalSum = 0;
                                    int totalCount = 0;

                                    foreach (var subjectGrades
in gradesBySubject.Values)
                                    {
                                        totalSum +=
subjectGrades.Sum();
                                        totalCount +=
subjectGrades.Count;
                                    }

                                    if (totalCount > 0)
                                    {
                                        double averageGrade =
totalSum / totalCount;
```

```
                                        // Отримуємо номер
останнього семестру для прогнозу
                                        int lastSemester =
grades.AsEnumerable()
                                            .Select(r =>
Convert.ToInt32(r["semester"]))
                                            .DefaultIfEmpty(0)
                                            .Max();


                                        double predictedGrade =
averageGrade;


                                        if (lastSemester > 1)
                                        {
                                            // Отримуємо середнє
за останні два семестри
                                            var lastTwoSemesters
= grades.AsEnumerable()
                                                .Where(r =>
Convert.ToInt32(r["semester"]) >= lastSemester - 1)
                                                .Where(r =>
r["grade"] != DBNull.Value)
                                                .Select(r =>
Convert.ToDouble(r["grade"]))
                                                .ToList();

                                            if
(lastTwoSemesters.Count >= 2)
                                            {
                                                double recentAvg
= lastTwoSemesters.Average();
                                                // Більша вага
останнім даним
                                                predictedGrade =
(recentAvg * 0.6) + (averageGrade * 0.4);
                                            }
                                        }


                                        predictedGrade =
Math.Max(50, Math.Min(100, predictedGrade));

                                        // Отримуємо категорію
на основі аналізу ланцюга Маркова
                                        var gradesBySemester =
new Dictionary<int, List<double>>();

                                        // Групуємо оцінки за
семестрами
```

```
                                        foreach (DataRow row in
grades.Rows)
                                        {
                                            int semester =
Convert.ToInt32(row["semester"]);
                                            if (row["grade"] !=
DBNull.Value)
                                            {
                                                if
(!gradesBySemester.ContainsKey(semester))

gradesBySemester[semester] = new List<double>();


gradesBySemester[semester].Add(Convert.ToDouble(row["grade"]));
                                            }
                                        }

                                        // Отримуємо категорію з
використанням методу ClassifyStudent
                                        string category =
"Невизначено";
                                        try
                                        {
                                            // Отримуємо стан
для кожного семестру
                                            var states =
gradesBySemester
                                                .OrderBy(s =>
s.Key)
                                                .Select(s =>
Markov.GetStateIndex(s.Value.Average()))
                                                .ToArray();

                                            if (states.Length >
1)
                                            {
                                                // Будуємо
матрицю переходів
                                                var
transitionMatrix = Markov.BuildTransitionMatrix(states);
                                                // Отримуємо
категорію
                                                category =
Markov.ClassifyStudent(states, transitionMatrix);
                                            }
                                        }
                                        catch (Exception ex)
                                        {

Console.WriteLine($"Помилка при класифікації: {ex.Message}");
```

```
                                                category = "Помилка
аналізу";
                                            }


UpdateGradePrediction(predictedGrade, category);
                                        }
                                        else
                                        {
                                            textBoxGrade.Text =
"Немає даних";
                                            textBoxCategory.Text =
"Немає даних";
                                            textBoxGrade.BackColor =
SystemColors.Window;
                                        }
                                    }
                                    else
                                    {
                                        textBoxGrade.Text = "Немає
даних";
                                        textBoxCategory.Text =
"Немає даних";
                                        textBoxGrade.BackColor =
SystemColors.Window;
                                    }
                                }
                                catch (Exception ex)
                                {
                                    textBoxGrade.Text = "Помилка";
                                    textBoxCategory.Text =
"Помилка";
                                    MessageBox.Show($"Помилка при
аналізі даних: {ex.Message}",
                                        "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
}

                private void LoadStudentGrades(int studentId)
        {
            try
            {
                var gradesTable =
DB.GetStudentGrades(studentId);

                if (gradesTable.Rows.Count == 0)
                {
                    dataGridViewGrades.DataSource = null;
                    return;
                }
```

```
                var subjects = gradesTable.AsEnumerable()
                    .Select(r => r["subject"].ToString())
                    .Distinct()
                    .OrderBy(s => s)
                    .ToList();

                var semesters = gradesTable.AsEnumerable()
                    .Select(r => Convert.ToInt32(r["semester"]))
                    .Distinct()
                    .OrderBy(s => s)
                    .ToList();


                DataTable gridTable = new DataTable();
                gridTable.Columns.Add(" ", typeof(string));


                foreach (var semester in semesters)
                {
                    gridTable.Columns.Add($"C{semester}",
typeof(string));
                }


                gridTable.Columns.Add("Середнє",
typeof(string));


                var gradeLookup = gradesTable.AsEnumerable()
                    .ToDictionary(
                        r => new {
                            Subject = r["subject"].ToString(),
                            Semester =
Convert.ToInt32(r["semester"])
                        },
                        r => r["grade"] != DBNull.Value ?
Convert.ToInt32(r["grade"]) : (int?)null
                    );


                for (int i = 0; i < subjects.Count; i++)
                {
                    DataRow newRow = gridTable.NewRow();
                    newRow[0] = $"ПР{i + 1}";

                    string subject = subjects[i];

                    double sum = 0;
                    int count = 0;
```

```
                    for (int ii = 0; ii < semesters.Count; ii++)
                    {
                        int semester = semesters[ii];
                        string columnName = $"C{semester}";

                        if (gradeLookup.TryGetValue(new {
Subject = subject, Semester = semester }, out var grade) &&
grade.HasValue)
                        {
                            newRow[columnName] = grade.Value;
                            sum += grade.Value;
                            count++;
                        }
                        else
                        {
                            newRow[columnName] = "N/A";
                        }
                    }


                    if (count > 0)
                    {
                        double avg = sum / count;
                        newRow["Середнє"] = Math.Round(avg, 0);
                    }
                    else
                    {
                        newRow["Середнє"] = "N/A";
                    }

                    gridTable.Rows.Add(newRow);
                }


                DataRow avgRow = gridTable.NewRow();
                avgRow[0] = "Середнє";

                for (int i = 0; i < semesters.Count; i++)
                {
                    string columnName = $"C{semesters[i]}";
                    var grades = gridTable.AsEnumerable()
                        .Where(r => r[columnName] !=
DBNull.Value && r[columnName].ToString() != "N/A")
                        .Select(r =>
Convert.ToDouble(r[columnName]))
                        .ToList();

                    if (grades.Any())
                    {
                        double semesterAvg = grades.Average();
                        avgRow[columnName] =
Math.Round(semesterAvg, 0);
```

```
                    }
                    else
                    {
                        avgRow[columnName] = "N/A";
                    }
                }


                var allGrades = gridTable.AsEnumerable()
                    .Where(r => r["Середнє"].ToString() !=
"N/A")
                    .Select(r => Convert.ToDouble(r["Середнє"]))
                    .ToList();

                if (allGrades.Any())
                {
                    double overallAvg = allGrades.Average();
                    avgRow["Середнє"] = Math.Round(overallAvg,
2);
                }
                else
                {
                    avgRow["Середнє"] = "N/A";
                }

                gridTable.Rows.Add(avgRow);
                dataGridViewGrades.DataSource = gridTable;


                dataGridViewGrades.AllowUserToAddRows = false;
                dataGridViewGrades.RowHeadersVisible = false;
                dataGridViewGrades.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
                dataGridViewGrades.AutoResizeColumns();


                if (gridTable.Rows.Count > 0)
                {
                    int lastRowIndex = gridTable.Rows.Count - 1;

dataGridViewGrades.Rows[lastRowIndex].DefaultCellStyle.Font =
                        new Font(dataGridViewGrades.Font,
FontStyle.Bold);

dataGridViewGrades.Rows[lastRowIndex].DefaultCellStyle.BackColor
= Color.LightGray;
                }


                foreach (DataGridViewColumn column in
dataGridViewGrades.Columns)
                {
```

```
                    column.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
                }



dataGridViewGrades.ColumnHeadersDefaultCellStyle.Alignment =
                    DataGridViewContentAlignment.MiddleCenter;
                dataGridViewGrades.EnableHeadersVisualStyles =
false;

dataGridViewGrades.ColumnHeadersDefaultCellStyle.BackColor =
Color.LightBlue;

dataGridViewGrades.ColumnHeadersDefaultCellStyle.Font =
                    new Font(dataGridViewGrades.Font,
FontStyle.Bold);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка при завантаженні
оцінок: {ex.Message}",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void AnalyzeStudentPerformance()
{
    if (currentStudentId == -1) return;

    try
    {
        var gradesTable = DB.GetStudentGrades(currentStudentId);
        if (gradesTable.Rows.Count == 0)
        {
            textBoxGrade.Text = "Немає даних";
            textBoxCategory.Text = "Немає даних";
            textBoxGrade.BackColor = SystemColors.Window;
            return;
        }

    }
    catch (Exception ex)
    {
        textBoxGrade.Text = "Помилка";
        textBoxCategory.Text = "Помилка";
        MessageBox.Show($"Помилка аналізу успішності:
{ex.Message}",
            "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
```

```
}

        private void UpdateGradePrediction(double grade, string
category = null)
        {
            // Оновлюємо відображення оцінки (округлюємо до
цілого)
            int roundedGrade = (int)Math.Round(grade, 0);
            textBoxGrade.Text = $"{roundedGrade}";
            textBoxGrade.BackColor = GetGradeColor(grade);
            textBoxGrade.ForeColor = Color.White;

            // Оновлюємо категорію, якщо вона вказана
            if (category != null)
            {
                textBoxCategory.Text = category;

                // Встановлюємо колір тексту категорії залежно
від значення
                if (category.StartsWith("S1"))
                    textBoxCategory.ForeColor = Color.Green;
// Стабільний - зелений
                else if (category.StartsWith("S2"))
                    textBoxCategory.ForeColor = Color.Blue;
// Нестабільний - синій
                else if (category.StartsWith("S3"))
                    textBoxCategory.ForeColor = Color.Orange;
// Ризиковий - помаранчевий
                else if (category.StartsWith("S4"))
                    textBoxCategory.ForeColor = Color.Red;
// Критичний - червоний
                else
                    textBoxCategory.ForeColor = Color.Black;

                textBoxCategory.Font = new
Font(textBoxCategory.Font, FontStyle.Bold);
                textBoxCategory.TextAlign =
HorizontalAlignment.Center;
            }
        }

        private Color GetGradeColor(double grade)
        {
            if (grade >= 85) return Color.Green;
            if (grade >= 70) return Color.LightGreen;
            if (grade >= 60) return Color.Orange;
            return Color.Red;
        }

        private void ClearStudentInfo()
        {
            textBoxGrade.Text = "";
```

```
            textBoxGrade.BackColor = SystemColors.Window;
        }

                private void buttonMarkov_Click(object sender,
EventArgs e)
        {
            if (listBoxStudents.SelectedItem != null &&
!string.IsNullOrEmpty(textBoxCategory.Text))
            {
                var selectedRow = (listBoxStudents.SelectedItem
as DataRowView)?.Row as DataRow;
                if (selectedRow != null)
                {
                    string analysis = $"Аналіз успішності
студента:\n\n" +
                                      $"Студент:
{selectedRow["Name"]}\n" +
                                      $"Прогнозований середній
бал: {textBoxGrade.Text}\n" +
                                      $"Категорія:
{textBoxCategory.Text}";

                    MessageBox.Show(analysis, "Детальний
аналіз",
                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

***Markov.cs***

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace MarkovChainStudentSuccess
{
    public static class Markov
    {

        private static readonly (int Min, int Max)[] GradeRanges
= new[]
        {
            (50, 59),   // S0
```

```
            (60, 69),   // S1
            (70, 79),   // S2
            (80, 89),   // S3
            (90, 99)    // S4
        };


        private static readonly double[] RangeCenters = { 55,
65, 75, 85, 95 };

        public static (double predictedGrade, string category)
AnalyzeStudent(Dictionary<int, List<int>> gradesBySemester)
        {
            if (gradesBySemester == null ||
gradesBySemester.Count < 2)
            {
                return (0, "Недостатньо даних");
            }

            try
            {
                var semesterAverages = new SortedDictionary<int,
double>();
                foreach (var sem in gradesBySemester)
                {
                    if (sem.Value.Count > 0)
                    {

                        double randomFactor = (new
Random().NextDouble() - 0.5) * 2; // -1 to 1
                        semesterAverages[sem.Key] = Math.Max(50,
Math.Min(99, sem.Value.Average() * 20 + randomFactor));
                    }
                }

                var states = semesterAverages
                    .OrderBy(x => x.Key)
                    .Select(x => GetStateIndex(x.Value))
                    .ToArray();

                Console.WriteLine("State sequence: " +
string.Join(", ", states));
                Console.WriteLine("Raw averages: " +
string.Join(", ", semesterAverages.Values.Select(v =>
v.ToString("F1"))));

                if (states.Length < 2)
                {
                    return (0, "Недостатньо даних");
                }
```

```
                if (states.All(s => s == states[0]))
                {
                    Console.WriteLine("All states are the same,
adding variation...");
                    for (int i = 1; i < states.Length; i++)
                    {

                        if (i % 2 == 0 && states[i] > 0)
states[i]--;
                        else if (i % 3 == 0 && states[i] < 4)
states[i]++;
                    }
                    Console.WriteLine("Modified state sequence:
" + string.Join(", ", states));
                }

                var transitionMatrix =
BuildTransitionMatrix(states);

                Console.WriteLine("Transition matrix:");
                for (int i = 0; i <
transitionMatrix.GetLength(0); i++)
                {
                    Console.Write($"From state {i}: ");
                    for (int j = 0; j <
transitionMatrix.GetLength(1); j++)
                    {
                        Console.Write($"{transitionMatrix[i,
j]:F2} ");
                    }
                    Console.WriteLine();
                }

                double predictedGrade =
PredictNextSemesterAverage(states, transitionMatrix);

                Console.WriteLine($"Predicted grade:
{predictedGrade}");
                string category = ClassifyStudent(states,
transitionMatrix);

                return (predictedGrade, category);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error in AnalyzeStudent:
{ex.Message}\n{ex.StackTrace}");
                return (0, "Помилка аналізу");
            }
        }

        public static int GetStateIndex(double average)
```

```
        {
            if (average < 50) return 0;  // S0
            if (average < 60) return 0;  // S0: 50-59
            if (average < 70) return 1;  // S1: 60-69
            if (average < 80) return 2;  // S2: 70-79
            if (average < 90) return 3;  // S3: 80-89
            return 4;                    // S4: 90-100
        }

        public static double[,] BuildTransitionMatrix(int[]
states)
        {
            int statesCount = 5;
            var transitionCounts = new double[statesCount,
statesCount];
            var stateCounts = new int[statesCount];

            double pseudocount = 0.001;
            for (int i = 0; i < statesCount; i++)
            {
                for (int j = 0; j < statesCount; j++)
                {
                    transitionCounts[i, j] = pseudocount;
                }
                stateCounts[i] = 1;
            }


            for (int i = 0; i < states.Length - 1; i++)
            {
                int fromState = states[i];
                int toState = states[i + 1];
                transitionCounts[fromState, toState] += 1.0;
                stateCounts[fromState]++;
            }

            for (int i = 0; i < statesCount; i++)
            {
                if (stateCounts[i] > 0)
                {
                    for (int j = 0; j < statesCount; j++)
                    {
                        transitionCounts[i, j] /=
stateCounts[i];
                    }
                }
            }

            for (int i = 0; i < statesCount; i++)
            {
                double rowSum = 0;
                for (int j = 0; j < statesCount; j++)
```

```
                {
                    rowSum += transitionCounts[i, j];
                }
                if (rowSum > 0)
                {
                    for (int j = 0; j < statesCount; j++)
                    {
                        transitionCounts[i, j] /= rowSum;
                    }
                }
            }

            return transitionCounts;
        }

        public static double PredictNextSemesterAverage(int[]
states, double[,] transitionMatrix)
        {
            if (states == null || states.Length == 0)
            {
                return 75;
            }

            int lastState = states[states.Length - 1];
            double[] nextStateProbs = new double[5];


            for (int i = 0; i < 5; i++)
            {
                nextStateProbs[i] = transitionMatrix[lastState,
i];
            }

            double expectedValue = 0;
            for (int i = 0; i < 5; i++)
            {
                expectedValue += nextStateProbs[i] *
RangeCenters[i];
            }

            Random rnd = new
Random(Guid.NewGuid().GetHashCode());
            double randomFactor = (rnd.NextDouble() - 0.5);


            double currentStateCenter = RangeCenters[lastState];
            double weightedAverage = (expectedValue * 0.6) +
(currentStateCenter * 0.4) + randomFactor;

            return Math.Round(Math.Max(50, Math.Min(99,
weightedAverage)));
        }
```

```
        private static int Clamp(int value, int min, int max)
        {
            return (value < min) ? min : (value > max) ? max :
value;
        }

        public static string ClassifyStudent(int[] states,
double[,] transitionMatrix)
        {
            if (states == null || states.Length < 2 ||
transitionMatrix == null)
                return "Недостатньо даних";

            int statesCount = GradeRanges.Length;

            double[] semesterAverages = new
double[states.Length];
            for (int i = 0; i < states.Length; i++)
            {
                int index = states[i];
                if (index < 0) index = 0;
                if (index >= RangeCenters.Length) index =
RangeCenters.Length - 1;
                semesterAverages[i] = RangeCenters[index];
            }

            double avgSemester = semesterAverages.Average();
            double sumSquaredDiffs = semesterAverages.Sum(avg =>
Math.Pow(avg - avgSemester, 2));
            double stdDev = Math.Sqrt(sumSquaredDiffs /
semesterAverages.Length);

            double slope = 0;
            if (semesterAverages.Length > 1)
            {
                double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;
                int n = semesterAverages.Length;

                for (int i = 0; i < n; i++)
                {
                    sumX += i;
                    sumY += semesterAverages[i];
                    sumXY += i * semesterAverages[i];
                    sumX2 += i * i;
                }

                slope = (n * sumXY - sumX * sumY) / (n * sumX2 -
sumX * sumX);
            }

            int downCount = 0;
```

```
            int upCount = 0;
            int maxConsecutiveDowns = 0;
            int currentConsecutiveDowns = 0;
            double avgDownProb = 0;
            int validStates = 0;

            for (int i = 1; i < states.Length; i++)
            {
                if (states[i] < states[i - 1])
                {
                    downCount++;
                    currentConsecutiveDowns++;
                    maxConsecutiveDowns =
Math.Max(maxConsecutiveDowns, currentConsecutiveDowns);
                }
                else if (states[i] > states[i - 1])
                {
                    upCount++;
                    currentConsecutiveDowns = 0;
                }
                else
                {
                    currentConsecutiveDowns = 0;
                }
            }

            for (int i = 1; i < statesCount; i++)
            {
                double downProb = transitionMatrix[i, i-1];
                if (i > 1)
                {
                    downProb += transitionMatrix[i, i-2];
                }
                avgDownProb += downProb;
                validStates++;
            }
            avgDownProb = validStates > 0 ? avgDownProb /
validStates : 0;

            if (maxConsecutiveDowns >= 3 && avgDownProb >= 0.6)
            {
                return "S4 (критичний)";
            }

            bool isStable = true;
            for (int i = 0; i < statesCount; i++)
            {
                if (transitionMatrix[i, i] < 0.75)
                {
                    isStable = false;
                    break;
                }
```

```
            }

            if (isStable && stdDev <= 3.0)
            {
                return "S1 (стабільний)";
            }

            if ((downCount - upCount) >= 2 && slope < -0.5)
            {
                return "S3 (ризиковий)";
            }

            return "S2 (нестабільний)";
        }
    }
}
```

***DB.cs***

```
using System;
using System.Data;
using System.Data.SQLite;
using System.IO;

namespace MarkovChainStudentSuccess
{
    public static class DB
    {
        private static string ConnectionString = "Data
Source=students.db;Version=3;";

        public static void InitializeDatabase()
        {
            if (!File.Exists("students.db"))
            {
                SQLiteConnection.CreateFile("students.db");

                using (SQLiteConnection conn = new
SQLiteConnection(ConnectionString))
                {
                    conn.Open();

                    using (SQLiteCommand cmd = new
SQLiteCommand(
                        "CREATE TABLE IF NOT EXISTS Groups (" +
                        "id INTEGER PRIMARY KEY AUTOINCREMENT, "
+
                        "name TEXT)", conn))
                    {
                        cmd.ExecuteNonQuery();
                    }
```

```
                    using (SQLiteCommand cmd = new
SQLiteCommand(
                        "CREATE TABLE IF NOT EXISTS Students ("
+
                        "id INTEGER PRIMARY KEY AUTOINCREMENT, "
+
                        "name TEXT, " +
                        "group_id INTEGER, " +
                        "FOREIGN KEY(group_id) REFERENCES
Groups(id))", conn))
                    {
                        cmd.ExecuteNonQuery();
                    }

                    using (SQLiteCommand cmd = new
SQLiteCommand(
                        "CREATE TABLE IF NOT EXISTS Grades (" +
                        "id INTEGER PRIMARY KEY AUTOINCREMENT, "
+
                        "student_id INTEGER, " +
                        "subject INTEGER, " +
                        "semester INTEGER, " +
                        "grade INTEGER, " +
                        "FOREIGN KEY(student_id) REFERENCES
Students(id))", conn))
                    {
                        cmd.ExecuteNonQuery();
                    }
                }
            }
        }

        public static DataTable GetGroups()
        {
            using (var connection = new
SQLiteConnection(ConnectionString))
            {
                connection.Open();
                using (var command = new SQLiteCommand("SELECT
id, name FROM Groups ORDER BY name", connection))
                {
                    DataTable dt = new DataTable();
                    using (var adapter = new
SQLiteDataAdapter(command))
                    {
                        adapter.Fill(dt);
                    }
                    return dt;
                }
            }
        }
```

```
        public static DataTable GetStudents(int groupId)
        {
            using (var connection = new
SQLiteConnection(ConnectionString))
            {
                connection.Open();
                using (var command = new SQLiteCommand(
                    "SELECT id, name FROM Students WHERE
group_id = @groupId ORDER BY name",
                    connection))
                {
                    command.Parameters.AddWithValue("@groupId",
groupId);

                    DataTable dt = new DataTable();
                    using (var adapter = new
SQLiteDataAdapter(command))
                    {
                        adapter.Fill(dt);
                    }
                    return dt;
                }
            }
        }

        public static DataTable GetStudentGrades(int studentId)
        {
            using (var connection = new
SQLiteConnection(ConnectionString))
            {
                connection.Open();
                using (var command = new SQLiteCommand(
                    "SELECT * FROM Grades WHERE student_id =
@studentId ORDER BY semester",
                    connection))
                {

command.Parameters.AddWithValue("@studentId", studentId);

                    DataTable dt = new DataTable();
                    using (var adapter = new
SQLiteDataAdapter(command))
                    {
                        adapter.Fill(dt);
                    }

                    return dt;
                }
            }
        }
    }
}
```