МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

КАРАВАН ЄВГЕНІЙ ВІТАЛІЙОВИЧ

Допускається до захисту:
в. о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____О. В. Зелінська
«____» _____2024 р.

РОЗРОБКА СИСТЕМИ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Т. В. Січко, доцент кафедри
інформаційних технологій,
к. т. н., доцент
_____

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця- 2024

АНОТАЦІЯ

Караван Є.В. Розробка системи автоматизації формування розкладу занять. Спеціальність 122 «Комп’ютерні науки», освітня програма «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2024.

У кваліфікаційній (бакалаврській) роботі розглянуто постановку задачі автоматизації формування розкладу занять, проаналізовано існуючі системи та методи створення і автоматизації розкладу. За допомогою мови програмування C# та бази даних була розроблена система, яка використовує генетичний алгоритм для оптимізації процесу. Основна мета системи – забезпечити ефективне та гнучке планування занять у навчальних закладах.

Ключові слова: автоматизація, розклад занять, генетичний алгоритм, C#, база даних.

56 ст. 10 рис., 1 табл., 36 джерел.

ABSTRACT

Karavan Y. Development of a System for Automating the Formation of the Class Schedule. Specialty 122 «Computer Science», educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia 2024.

In the qualification (bachelor's) work, the problem statement of automating class scheduling is considered, existing scheduling systems are analyzed, and methods of creating and automating schedules are discussed. Using the C# programming language and a database, a system for automating the scheduling of classes was developed, which employs a genetic algorithm for optimization. The main purpose of the system is to ensure efficient and flexible scheduling in educational institutions.

ЗМІСТ

ВСТУП

Актуальність теми дослідження очевидна у сучасному контексті вищої освіти, де необхідність ефективного управління навчальним процесом стає все більш актуальною [1]. Зростання числа студентів, різноманітність предметів та гнучкість навчальних програм створюють складні виклики для формування розкладу занять, що відповідає потребам як викладачів, так і студентів [27]. Університети постійно шукають інноваційні рішення для оптимізації цього процесу, а системи автоматизації формування розкладу занять є одним з потенційних рішень. Такі системи дозволяють ефективно використовувати ресурси, уникати конфліктів в графіку та забезпечувати більшу задоволеність усіх учасників навчального процесу [2].

Метою бакалаврської роботи є розробка ефективного інструменту, що допоможе автоматизувати процес складання розкладу занять, забезпечуючи більш ефективне використання ресурсів, таких як аудиторії, викладачі та навчальні матеріали, а також зручність у використанні для всіх учасників навчального процесу.

Завдання дослідження:

1. Провести аналіз сучасних підходів та технологій у сфері автоматизації формування розкладу занять.
2. Визначити вимоги до системи автоматизації та обґрунтувати їх, враховуючи потреби різних учасників навчального процесу.
3. Розробити систему, яка відповідає встановленим вимогам та забезпечує ефективне управління розкладом занять.

Об'єктом дослідження є процес формування розкладу занять в навчальному закладі.

Предметом дослідження є система автоматизації формування розкладу занять, яка розробляється для оптимізації навчального процесу. Це включає в себе аналіз вимог, розробку програмного забезпечення.

Теоретичне значення отриманих результатів полягає в тому, що робота

сприяє подальшому розвитку теорії та практики управління навчальним процесом у вищих навчальних закладах.

Практичне значення результатів бакалаврської роботи полягає в покращенні організації навчального процесу, уникненні конфліктів у графіку занять, підвищенні якості навчання, задоволеності всіх учасників навчального процесу та економії часу і зусиль. Система дозволяє ефективніше планувати заняття, забезпечуючи студентам можливість відвідувати всі необхідні заняття без накладок. Впровадження системи сприятиме підвищенню ефективності навчального процесу, якості освіти та загальної задоволеності учасників.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ ФОРМУВАННЯ РОЗКЛАДУ

1.1 Постановка задачі

Розклад - це організований план, що визначає послідовність та час проведення занять протягом певного періоду. Він відображає графік навчальних заходів у закладах освіти, таких як школи, коледжі та університети, надаючи учасникам чітку структуру для їхнього навчання і роботи [3].

Основними елементами розкладу є [4]:

- предмети: дисципліни, які викладаються протягом семестру або навчального року;
- час: визначені години та дні тижня, коли проводяться заняття;
- аудиторії: приміщення, де відбуваються заняття;
- викладачі: люди, які проводять заняття;
- групи студентів: студенти, які відвідують певні заняття.

Розклад може мати різні форми та варіанти:

- щоденний: визначає порядок проведення занять на кожен день;
- тижневий: охоплює графік занять на весь тиждень;
- семестровий: планує заняття на весь семестр.

Добре спланований розклад допомагає оптимізувати використання ресурсів навчального закладу, запобігати конфліктам та перетинам у графіку занять, а також забезпечувати рівномірне навантаження на викладачів та студентів.

Предмети (дисципліни) у навчальному процесі - це окремі курси або теми, які вивчаються студентами протягом певного періоду. Кожен предмет має свою програму, методи викладання та оцінювання.

Основні форми проведення занять з предметів включають:

- лекції: теоретичні заняття, де викладач подає навчальний матеріал студентам;
- лабораторні заняття: форма навчання, де студенти виконують експерименти або дослідження;
- практичні заняття: застосування теоретичних знань на практиці через виконання завдань або проектів;
- семінари: індивідуальні або групові зустрічі з викладачем для обговорення складних питань або підготовки до іспитів.

Кожна форма занять спрямована на різні аспекти навчання та допомагає студентам краще засвоїти матеріал, розвинути необхідні навички та підготуватися до професійної діяльності.

Час - це фізична величина, що вимірює порядок подій у послідовності. У контексті навчального процесу, час використовується для організації навчального розкладу, планування занять та визначення тривалості кожного етапу навчання. Час може бути представлений у форматі годин, хвилин, днів, тижнів, місяців або навіть років, залежно від потреб конкретного навчального процесу. Ефективне управління часом дозволяє оптимізувати навчальний процес, забезпечувати регулярність та систематичність занять, а також досягати поставлених навчальних цілей у визначений строк.

Аудиторії - це приміщення, спеціально обладнане для проведення навчальних, наукових або інших видів діяльності. У навчальних закладах, аудиторії використовуються для проведення лекцій, практичних занять та інших форм навчання. Кожна аудиторія може мати відповідне обладнання, таке як дошка, проектор, аудіо- та відео обладнання, меблі для сидіння та інше необхідне обладнання для зручного проведення занять. Ефективне використання аудиторій дозволяє забезпечити комфортні та продуктивні умови для навчання та сприяє здійсненню навчального процесу.

Викладачі - це професійні фахівці, які мають знання та досвід у певній галузі і викладають предмети у навчальних закладах. Вони відповідають за

передачу знань, навичок та досвіду студентам, організацію та проведення навчальних занять, оцінювання студентів та ведення наукової роботи у своїй галузі. Викладачі можуть мати різний рівень кваліфікації та досвіду, а також спеціалізуватися на різних предметах та методиках викладання.

Групи студентів - це студенти, які навчаються разом у рамках певного навчального курсу або програми. Групи студентів можуть бути складені за різними критеріями, такими як спеціалізація, рівень знань, рік навчання тощо. У навчальних закладах групи студентів зазвичай відвідують однакові заняття разом з одним або кількома викладачами. Робота з групами студентів дозволяє забезпечити індивідуальний підхід до навчання, сприяти взаємній підтримці та спільному навчанню, а також створювати сприятливу атмосферу для обміну знаннями та досвідом.

Враховуючи вище подану інформацію, необхідно розробити систему автоматизації для формування розкладу занять у навчальному закладі з урахуванням ряду обмежень та вимог [5]. Задача полягає у створенні оптимального розкладу, який би задовольняв наступні критерії та обмеження [6]:

- доступність аудиторій: кожне заняття повинно мати відповідну аудиторію для проведення;
- навантаження викладачів: викладачі повинні мати рівномірне навантаження;
- перетини у заняттях: уникнення перетинів у розкладі занять для груп студентів, щоб уникнути конфліктів та забезпечити комфортний навчальний процес;
- підтримка індивідуальних потреб: врахування спеціальних запитів від груп студентів або викладачів;
- мінімізація часу та зусиль: створення розкладу, який мінімізує загальний час та зусилля, потрібні для його формування та відповідності всім обмеженням.

1.2 Огляд існуючих систем формування розкладу занять

У сучасному світі, особливо в університетському середовищі, ефективне управління розкладом занять вважається ключовим завданням. Системи автоматизації формування розкладу занять допомагають університетам планувати навчальні заходи, розподіляти ресурси та забезпечувати оптимальний навчальний процес. Ось деякі з них:

Ad Astra - це програмне забезпечення для управління розкладом занять та ресурсами в університетах. Воно дозволяє автоматизувати процес формування розкладу занять, враховуючи різноманітні параметри та обмеження, такі як доступність аудиторій, розклад викладачів, побажання студентів тощо. Ad Astra має розширені аналітичні засоби та інтеграцію з іншими системами управління навчальним процесом, що дозволяє університетам ефективно використовувати ресурси та підвищувати якість навчального процесу.

Open EduCat - це відкрите програмне забезпечення для управління навчальними закладами, яке включає модуль для формування розкладу занять. Ця система дозволяє автоматизувати процес планування та розподілу занять для студентів та викладачів університету. Завдяки цій системі університети можуть забезпечити зручний та ефективний розклад занять, що відповідає потребам студентів і викладачів.

Ad Astra

Ad Astra – це система автоматизації формування розкладу занять, призначена для закладів освіти. Вона дозволяє ефективно планувати навчальний процес, забезпечуючи оптимальне використання ресурсів, таких як аудиторії та викладачі. Ad Astra використовує потужні алгоритми для створення розкладу, який відповідає різним критеріям і обмеженням [7].

Інтерфейс системи зображений на рисунку 1.1.

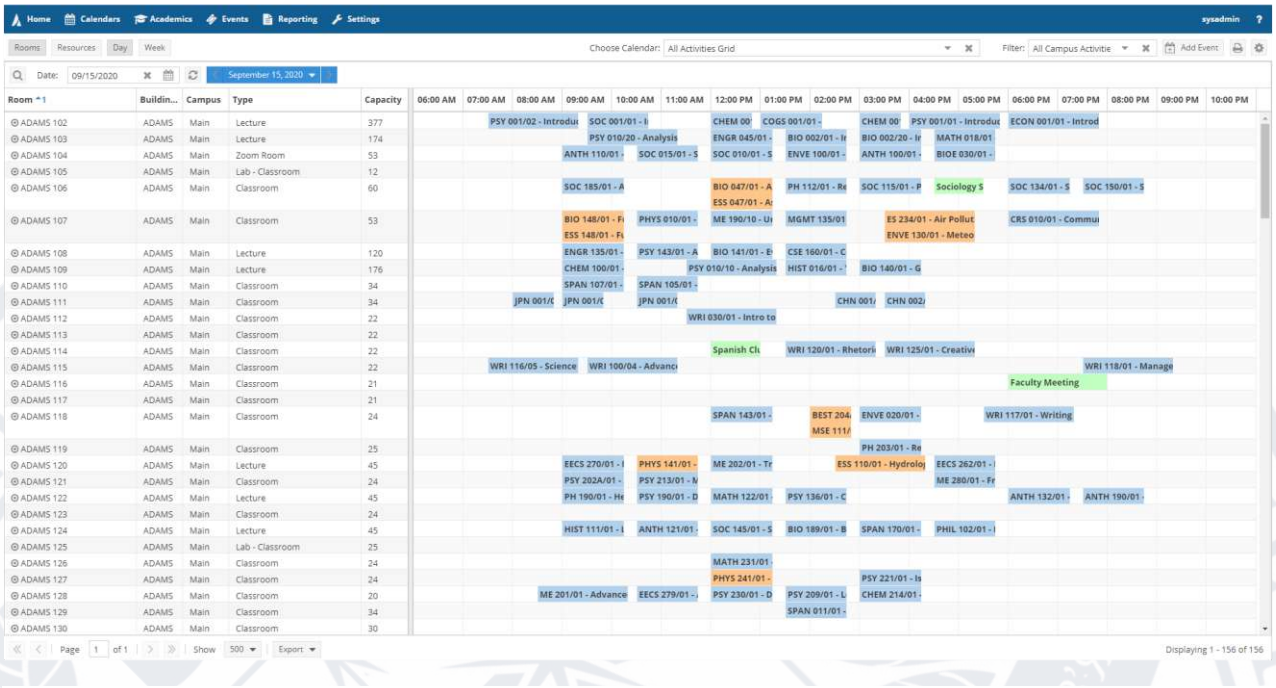

Рисунок 1.1 – Графічний інтерфейс Ad Astra

Ad Astra пропонує різні модулі і функціональні можливості, які включають у себе:

- формування розкладу занять: цей модуль дозволяє університетам створювати розклад занять, враховуючи різні параметри, такі як типи занять, доступність аудиторій, графік викладачів, побажання студентів тощо. Користувачі можуть встановлювати пріоритети, обмеження та умови для формування оптимального розкладу;
- прогнозування ресурсів: цей модуль допомагає університетам прогнозувати потреби у ресурсах, таких як аудиторії, викладачі та інше обладнання, на основі попередніх даних та прогностичних моделей;
- управління аудиторіями: модуль управління аудиторіями дозволяє університетам ефективно керувати доступними приміщеннями, розподіляти їх між різними типами занять, розраховувати максимальну вмістимість та вирішувати конфлікти;

- розрахунок навантаження на викладачів: цей модуль дозволяє автоматично розподіляти навантаження між викладачами, враховуючи їхні пріоритети, розклад занять, кількість годин робочого часу та інші фактори;
- аналітика даних: Ad Astra надає засоби для аналізу даних та генерації звітів, що дозволяє університетам відстежувати ефективність розкладу занять, виявляти проблемні ситуації та приймати обґрунтовані рішення щодо оптимізації навчального процесу;
- інтеграція з іншими системами: Ad Astra має можливість легко інтегруватися з іншими системами університету, такими як системи управління навчальним процесом та обліку студентів, що забезпечує єдність інформації та автоматизацію обміну даними.

Ці модулі і функціональні можливості дозволяють університетам ефективно управляти розкладом занять, забезпечуючи оптимальне використання ресурсів та підвищення якості навчального процесу.

Також в Ad Astra доступна підтримка користувачів. Команда підтримки користувачів Ad Astra пропонує широкий спектр послуг, спрямованих на надання допомоги, консультацій та навчання користувачів для ефективного використання програми. Навчання персоналу охоплює всі аспекти використання програми, від основ до продвинутих функцій, і може проводитися як в онлайн-форматі, так і в режимі навчання на місці. Консультації з питань впровадження надають допомогу у розумінні потреб та вимог користувачів для налаштування програми відповідно до їхніх потреб. Технічна підтримка надає допомогу у встановленні та налаштуванні програми, а також відповідає на запитання користувачів.

Команда Ad Astra регулярно випускає оновлення програмного забезпечення з виправленнями помилок, новими функціональностями та іншими поліпшеннями. Підтримка користувачів також включає надання

доступу до цих оновлень та допомогу у їх впровадженні.

Крім того, Ad Astra надає користувачам доступ до різноманітних документів, інструкцій та ресурсів, які допомагають розуміти роботу програми та використання її функціоналу. Ці ресурси сприяють ефективному використанню програми та забезпечують користувачам необхідну інформацію для успішної роботи з нею.

Програма Ad Astra може імпортувати дані про студентів та викладачів із систем управління навчальним процесом університету, дозволяючи автоматично отримувати актуальну інформацію про розклади занять, курси, групи тощо. Після цього Ad Astra може експортувати розклад занять та інші важливі дані назад до систем управління навчальним процесом, щоб забезпечити актуальність даних у всіх системах та уникнути дублювання інформації.

Під час інтеграції з системами управління навчальним процесом здійснюється синхронізація даних між програмою Ad Astra та іншими системами, щоб уникнути розходження даних та забезпечити єдність інформації. Це дозволяє Ad Astra ефективно планувати та розподіляти ресурси для оптимального формування розкладу занять, враховуючи доступні аудиторії, викладачів, курси тощо. Інтеграція з системами управління навчальним процесом допомагає Ad Astra отримувати дані про навчальні плани, курси, групи тощо, що дозволяє створювати розклад занять, який враховує специфіку навчальних програм та потреб студентів. Загалом, ця інтеграція забезпечує обмін даними та єдність інформації між різними системами університету, що сприяє ефективному управлінню навчальним процесом та покращенню якості освіти.

Програма Ad Astra доступна у різних версіях, які враховують специфіку та потреби різних типів університетів. Для великих державних університетів, які мають значну кількість студентів та факультетів, доступна розширена версія програми Ad Astra. Ця версія має розширені функціональні можливості та пристосована для роботи з великим обсягом даних. Приватні коледжі та

університети можуть скористатися меншою версією програми Ad Astra, яка має менший обсяг функціональності та доступна за більш прийнятною ціною. Технічні училища та коледжі можуть скористатися спеціалізованою версією Ad Astra, спрямованою на особливості цих навчальних закладів. Для невеликих підприємств або освітніх закладів з обмеженим бюджетом доступна базова версія Ad Astra з основним набором функцій. Деякі версії Ad Astra можуть бути адаптовані для використання в наукових інститутах та дослідницьких центрах.

Враховуючи різноманітні потреби та особливості різних типів університетів та освітніх закладів, Ad Astra надає різні версії програми з різними рівнями функціональності та можливостей, що дозволяє кожному клієнту знайти оптимальне рішення для своїх потреб.

Апаратне забезпечення та технічні вимоги для програми Ad Astra можуть варіюватися в залежності від обсягу даних, потреб користувачів та обраного рівня функціональності програми. Проте, існують загальні вимоги, які зазвичай встановлюються для встановлення та використання програмного забезпечення. Ось детальніша інформація про апаратне забезпечення та технічні вимоги для Ad Astra:

- для встановлення Ad Astra зазвичай потрібен сервер з операційною системою Windows Server або Linux, залежно від вибору програмного забезпечення. Сервер повинен мати достатню кількість обсягу дискового простору та потужності процесора для забезпечення надійної та ефективної роботи програми;
- Ad Astra використовує реляційну базу даних для зберігання та управління даними. Популярними варіантами для бази даних є Microsoft SQL Server або PostgreSQL. Потрібно встановити та налаштувати базу даних відповідно до вимог програми;
- для доступу до програми Ad Astra зазвичай використовуються звичайні комп'ютери з операційною системою Windows або Mac.

Клієнтські комп'ютери повинні мати сучасні веб-браузери, такі як Google Chrome, Mozilla Firefox або Microsoft Edge, для роботи з веб-інтерфейсом програми;

- для доступу до онлайн-версії Ad Astra, яка може бути хмарним рішенням, потрібне надійне і швидке Інтернет-з'єднання. Швидкість з'єднання повинна бути достатньою для передачі та отримання даних у реальному часі;
- для забезпечення безпеки даних, необхідно встановити та налаштувати відповідні заходи захисту, включаючи файрволи, антивірусне програмне забезпечення та механізми шифрування даних.

Ці технічні вимоги можуть змінюватися в залежності від конкретної версії та конфігурації програми Ad Astra, тому рекомендується уточнювати їх у виробника програмного забезпечення перед встановленням.

Недоліки програми Ad Astra:

- вартість впровадження та підтримки програми Ad Astra може бути високою, особливо для невеликих університетів з обмеженим бюджетом. Придбання ліцензії, навчання персоналу, підтримка та оновлення програмного забезпечення можуть вимагати значних витрат;
- деякі користувачі можуть виявити програму Ad Astra складною для використання через велику кількість функцій та налаштувань. Для користувачів з обмеженим досвідом у використанні комп'ютерних програм це може стати перешкодою;
- університетам може знадобитися постійна технічна підтримка для встановлення, налаштування та підтримки програми Ad Astra. Це може включати в себе як внутрішні ресурси університету, так і залучення зовнішніх фахівців;
- Неспроможність програми адаптуватися до деяких специфічних

потреб або особливостей конкретного університету може бути обмеженням для деяких користувачів. В деяких випадках програмне забезпечення може не мати достатньої гнучкості для вирішення унікальних проблем або сценаріїв використання, що може потребувати додаткових зусиль для знаходження або розробки рішень.

Підсумовуючи вище написану інформацію, Ad Astra - це програмне забезпечення, спеціалізоване у формуванні розкладу занять та управлінні навчальним процесом в університетах та інших освітніх закладах. Ad Astra дозволяє автоматизувати та оптимізувати розклад занять, забезпечуючи ефективну організацію навчального процесу. Незважаючи на свої переваги, Ad Astra також має деякі недоліки, такі як висока вартість, складність використання та залежність від технічної підтримки. Однак, загальною оцінкою є те, що Ad Astra є потужним інструментом для підтримки управління навчальним процесом в освітніх закладах [8].

Open EduCat

Open EduCat - це відкрите програмне забезпечення, спеціально розроблене для управління навчальним процесом в університетах та вищих навчальних закладах. Однією з його ключових функцій є формування розкладу занять, що є критично важливим аспектом для ефективного проведення навчання та організації робочого часу для студентів та викладачів. Крім того, Open EduCat пропонує широкий спектр інших функцій, які сприяють покращенню навчального процесу. Це включає в себе управління студентськими записами, автоматизацію процесу прийому, адміністрування курсів і модулів, контроль відвідуваності та оцінок, а також забезпечення доступу до навчальних матеріалів [9].

Інтерфейс системи зображений на рисунку 1.2.

Рисунок 1.2 – Графічний інтерфейс Open EduCat

Сучасні системи управління навчальним процесом, такі як Open EduCat, надають комплексний підхід до формування розкладу занять, починаючи з планування курсів і предметів. Під час цього етапу адміністрація університету може визначити структуру курсів, обрати необхідні предмети та призначити відповідних викладачів. Це важливо для забезпечення належної організації навчального процесу та визначення потреб у ресурсах.

Автоматизоване створення розкладу є ключовим етапом у процесі оптимізації навчального процесу. Система Open EduCat використовує алгоритми для автоматичного створення розкладу занять на основі різних критеріїв, таких як наявність вільних аудиторій, персональні побажання викладачів та упередження.

Інтеграція з базою даних студентів і викладачів є важливим аспектом для точного визначення кількості учасників кожного заняття та їхнього розподілу. Система забезпечує зв'язок з базою даних університету для автоматичного врахування студентів та викладачів при формуванні розкладу.

Оптимізація використання ресурсів полягає у мінімізації конфліктів у розкладі та максимально ефективному використанні доступних аудиторій та інших ресурсів. Система допомагає уникнути перекриття занять,

оптимізуючи розміщення занять у вільних приміщеннях та забезпечуючи комфортні умови для навчання. Це сприяє підвищенню продуктивності навчального процесу та задоволенню потреб всіх учасників.

Підтримка різних форматів занять в системах управління навчальним процесом, таких як Open EduCat є дуже важливою. Ця можливість дозволяє університетам працювати з різними типами занять, такими як лекції, семінари, лабораторні роботи, практичні заняття та інші. За допомогою Open EduCat можна ефективно планувати різні формати занять для різних курсів і предметів. Наприклад, для технічних дисциплін можуть бути заплановані лабораторні роботи, де студенти мають можливість випробувати теоретичні знання на практиці. Для гуманітарних дисциплін можуть бути організовані семінари або дискусійні групи, що сприяють обговоренню та аналізу матеріалу. Окрім того, Open EduCat може автоматично призначати відповідні аудиторії для лекцій або лабораторних занять з урахуванням їхньої оснащеності та спеціалізації.

Додатково, система може надавати можливість викладачам і студентам взаємодіяти під час занять у різних форматах. Наприклад, вони можуть підтримувати електронні платформи для обговорення матеріалу, завдань або взаємної підтримки під час виконання лабораторних робіт.

Одними з найважливіших аспектів ефективного управління навчальним процесом у вищих навчальних закладах є моніторинг і оновлення розкладу в реальному часі. Open EduCat і подібні системи надають інструменти для постійного контролю за розкладом та оперативного внесення змін у нього. Після створення початкового розкладу система Open EduCat забезпечує можливість моніторингу його виконання в реальному часі. Це дозволяє адміністраторам та викладачам оперативно виявляти будь-які проблеми або несподівані зміни в розкладі та вчасно реагувати на них. Наприклад, якщо викладач захворів або з інших причин не може провести заняття, адміністратор може швидко знайти заміну і оновити розклад без значних перебоїв у навчальному процесі. Це допомагає уникнути непорозумінь та

забезпечує всім учасникам навчального процесу достатньо часу на підготовку до занять.

Крім того, Open EduCat надає можливість зберігати історію змін у розкладі, що дозволяє проводити аналіз ефективності та покращувати процес планування у майбутньому. Аналіз даних про попередні зміни допомагає виявляти тенденції та вдосконалювати стратегії планування.

Завдяки підтримці мобільних пристроїв, користувачі можуть легко переглядати свій розклад, отримувати сповіщення про будь-які зміни або оновлення в розкладі прямо на своїх смартфонах або планшетах. Це забезпечує зручний доступ до інформації про навчальні заняття навіть поза корпусом університету. Крім того, мобільні додатки надають додаткові функції, такі як можливість відмічати присутність на заняттях, переглядати матеріали лекцій або завдання, спілкуватися з викладачами та одногрупниками. Це сприяє покращенню комунікації та співпраці всередині навчального середовища та створює додаткові можливості для навчання та розвитку.

Студентам, викладачам та адміністраторам системи надається зручний і простий у використанні інтерфейс для доступу до всіх необхідних функцій та можливостей системи. Зручний інтерфейс дозволяє користувачам легко переглядати свій розклад, отримувати сповіщення про зміни та оновлення, швидко знаходити необхідну інформацію та виконувати різні дії без зайвих зусиль. Студентам доступні такі функції як перегляд розкладу занять, запис на заняття, перегляд оцінок та завдань, комунікація з викладачами та спілкування з одногрупниками. Зручний інтерфейс дозволяє студентам швидко знаходити необхідну інформацію та взаємодіяти з системою без непотрібних перешкод. Викладачам, з свого боку, доступний інтерфейс для планування занять, оцінювання студентів, завантаження матеріалів та спілкування зі студентами. Це дозволяє викладачам ефективно керувати своїми курсами та спрощує взаємодію зі студентами. Адміністратори, з свого боку, мають доступ до інтерфейсу для управління користувачами,

налаштування параметрів системи, моніторингу даних та інші адміністративні функції. Зручний інтерфейс дозволяє адміністраторам ефективно керувати всіма аспектами системи та швидко реагувати на поточні потреби університету.

Завдяки можливості інтеграції з іншими системами, Open EduCat дозволяє створювати єдину екосистему, що об'єднує різноманітні аспекти управління навчальним процесом та спрощує взаємодію між ними:

- інтеграція з системами управління студентською базою дозволяє автоматично синхронізувати дані про студентів, їхні успіхи, рейтинги та іншу важливу інформацію. Це дозволяє забезпечити цілісність даних та зменшити ризик помилок у веденні студентської інформації;
- інтеграція з бібліотечними системами дозволяє студентам швидко знаходити необхідну літературу та ресурси для вивчення та дослідження. Студенти можуть отримувати доступ до електронних версій книг, журналів та наукових статей через один і той же інтерфейс, що спрощує їхню навчальну діяльність;
- інтеграція з системами електронного навчання дозволяє автоматично реєструвати студентів на курси та надавати доступ до навчальних матеріалів. Студенти можуть отримувати завдання та тести через систему управління навчальним процесом, а потім завантажувати свої відповіді та переглядати результати безпосередньо в системі електронного навчання;
- інтеграція з іншими системами також дозволяє забезпечити обмін даними та спільну роботу між університетом та іншими організаціями або партнерами. Можлива інтеграція з системами стажування, що дозволить автоматично реєструвати студентів на стажування та відстежувати їхні досягнення під час нього;

Підтримка аналізу даних є також важливою функцією в сучасних системах управління навчальним процесом. В Open EduCat ця можливість

дозволяє автоматично збирати дані про використання системи, активність студентів та викладачів, оцінки, результати тестувань, а також інші важливі параметри навчального процесу. Ці дані можуть бути використані для проведення різноманітних аналітичних висновків та візуалізації результатів. Система може проводити аналіз ефективності розкладу занять, виявляючи затримки або перекриття, а також оцінювати популярність та успішність різних курсів та предметів. Це дозволяє адміністраторам виявляти можливі проблеми та приймати заходи для їх вирішення. Дані про успішність студентів можуть використовуватися для аналізу ефективності програм та курсів, виявлення слабких місць та потреб у покращенні навчального матеріалу або методик навчання, що дозволяє університетам покращувати якість освіти та забезпечувати оптимальні умови для навчання студентів. Крім того, дані про активність студентів та їхній успіх можуть використовуватися для індивідуалізації навчального процесу та надання персоналізованих рекомендацій. Система може рекомендувати додаткові матеріали або курси для студентів з певними потребами або цілями навчання.

Хоча Open EduCat є потужною системою управління навчальним процесом зі значними перевагами, вона також має деякі недоліки:

- складність налаштування Open EduCat може становити серйозну перешкоду для університетів, особливо для тих, у кого недостатньо досвіду в адмініструванні систем. Встановлення та налаштування вимагають глибокого розуміння технічних аспектів, що може бути складним завданням для користувачів без відповідного фахового досвіду. Це може призвести до додаткових витрат коштів у випадку необхідності звертатися до зовнішніх консультантів або розробників для допомоги;
- обмежена підтримка з боку спільноти розробників може ускладнити вирішення проблем та оновлення системи. У випадках, коли необхідно вирішити серйозні технічні проблеми або реалізувати нові функції, обмежений доступ до професійної підтримки може

призвести до затримок у розвитку системи та негативно вплинути на її функціональність;

- використання всіх можливостей Open EduCat вимагає від користувачів часу та ресурсів на навчання та ознайомлення з системою. Навчання персоналу та користувачів може вимагати значних зусиль та часу, що може бути не завжди доступним для університетів з обмеженими ресурсами;
- оскільки Open EduCat - це відкрите програмне забезпечення, це може зробити систему вразливою до кібератак та порушень безпеки даних. Незважаючи на заходи захисту, які можуть бути прийняті, ризик вразливості до кіберзлочинності завжди присутній, що може підвищувати рівень ризику для університетів та їхніх даних.

Узагальнюючи, незважаючи на існуючі недоліки, Open EduCat є потужним інструментом для управління навчальним процесом у вищих навчальних закладах. З правильними ресурсами та підтримкою, ця система може ефективно допомогти університетам організувати розклад занять, оптимізувати використання ресурсів та забезпечити оптимальні умови для навчання та розвитку студентів. Зручний інтерфейс, широкі можливості налаштування, підтримка різних форматів занять та інтеграція з іншими системами роблять Open EduCat відмінним вибором для формування розкладу занять у сучасному університеті [10].

Порівняння Ad Astra та Open EduCat

Open EduCat та Ad Astra - це дві існуючі системи для формування розкладу занять в університетах, які мають свої переваги та недоліки.

Порівняння Open EduCat та Ad Astra:

Функціональність:

Open EduCat: ця система має широкий спектр функцій, включаючи планування курсів, автоматизоване створення розкладу, підтримку різних форматів занять, аналітику даних та інтеграцію з іншими системами.

Ad Astra: також надає функціональність для створення розкладу занять,

проте його основний акцент зазвичай спрямований на оптимізацію використання приміщень та ресурсів.

Складність налаштування:

Open EduCat: встановлення та налаштування може бути складним завданням, особливо для користувачів без достатнього досвіду в адмініструванні систем.

Ad Astra: система більш простіша в налаштуванні, що забезпечує швидше впровадження.

Підтримка:

Open EduCat: підтримка може бути обмеженою залежно від спільноти розробників та наявних ресурсів.

Ad Astra: зазвичай надається професійна підтримка від компанії, що розробляє систему, що може забезпечити більш швидке та ефективне вирішення проблем.

Відкритість для помилок безпеки:

Обидві системи можуть бути вразливими до кібератак та порушень безпеки даних. Важливо приділяти належну увагу заходам захисту в обох випадках.

Вартість та витрати:

Open EduCat: є відкритою системою з відкритим вихідним кодом, але може вимагати значних витрат на підтримку та налаштування.

Ad Astra: зазвичай є комерційною системою, що може вимагати плати за ліцензії та підтримку.

Отже, вибір між Open EduCat та Ad Astra буде залежати від потреб та можливостей конкретного університету, його технічного персоналу та фінансових ресурсів. Кожна з цих систем має свої переваги та недоліки, які слід враховувати при виборі.

РОЗДІЛ 2
МЕТОДИ СТВОРЕННЯ ТА АВТОМАТИЗАЦІЇ РОЗКЛАДУ ЗАНЯТЬ

2.1 Опис алгоритму для створення розкладу занять

Метою бакалаврської роботи є створення зручного та оптимального розкладу навчальних занять для закладу вищої освіти, який максимально враховує побажання викладачів та студентів, а також полегшує роботу особи, відповідальної за його формування [11]. Для досягнення цієї мети використовується генетичний алгоритм та цільова функція, яка відображає побажання всіх учасників навчального процесу.

Цільова функція складається з двох основних частин: побажання викладачів, які мають підвищений пріоритет, та побажання студентів [12]. Усі ці дані записуються у вигляді матриці переваг, яка формується на основі зібраної інформації про побажання. Це дозволяє оцінювати розклад за ступенем його відповідності заданим критеріям.

Процес створення розкладу включає кілька етапів [13]:

1. Генерація початкового розкладу: Створюється початковий варіант розкладу, який відповідає базовим обмеженням, таким як наявність аудиторій та викладацького складу.
2. Оцінка розкладу: Генерований розклад оцінюється за допомогою цільової функції, яка враховує матрицю переваг викладачів та студентів. Оцінка допомагає визначити, наскільки розклад відповідає заданим критеріям.
3. Удосконалення розкладу: У випадку, якщо початковий розклад не є оптимальним, він удосконалюється за допомогою генетичного алгоритму. Алгоритм виконує кросовери та мутації, створюючи нові покоління розкладів, кожен з яких оцінюється цільовою функцією. Процес триває до тих пір, поки не буде знайдений оптимальний варіант розкладу.

4. Вивід розкладу: Створений розклад виводиться у зручному для використання форматі, зокрема у вигляді файлів формату XLS, що дозволяє його легко друкувати та використовувати в електронному вигляді.

Цей підхід дозволяє автоматизувати процес формування розкладу занять, робить його гнучким та зручним для всіх учасників навчального процесу, а також забезпечує оптимальне використання ресурсів закладу вищої освіти [14].

2.2 Розробка цільової функції для оптимізації згенерованого розкладу

Для створення оптимального розкладу занять в університеті потрібно розробити цільову функцію, яка допоможе оцінити кожен можливий розклад. Метою є максимально врахувати побажання викладачів та студентів, при цьому уникнувши конфліктів у розкладі [15].

Компоненти цільової функції [16]:

1. Побажання викладачів та студентів:

   Побажання викладачів: Ми використовуємо матрицю побажань викладачів $Pt(i,j)$, де i — це індекс заняття, а j — індекс часового слоту. Кожен елемент цієї матриці показує, наскільки викладач бажає проводити заняття і у часовий слот j.

   Побажання студентів: Аналогічно, ми використовуємо матрицю побажань студентів $Ps(i,j)$, де кожен елемент відображає бажання студентів мати заняття і у часовий слот j.

2. Бінарні змінні $x_{i,j}$:

   $x_{i,j}$ — це змінна, яка дорівнює 1, якщо заняття i призначено у часовий слот j, і 0, якщо ні.

3. Конфлікти [17]**:**

   C — загальна кількість конфліктів у розкладі. Конфлікти можуть включати ситуації, коли одна аудиторія призначена для двох занять одночасно, коли викладач має два заняття в один і той

самий час, або коли групи студентів мають два заняття одночасно.

α — коефіцієнт, який показує, наскільки важливо уникати конфліктів. Цей коефіцієнт визначає, наскільки сильно штрафуються конфлікти у розкладі.

Формула цільової функції виглядає так (2.1):

$$F = w_t \sum_{i=1}^{N} \sum_{j=1}^{M} P_t(i,j) x_{i,j} + w_s \sum_{i=1}^{N} \sum_{j=1}^{M} P_s(i,j) x_{i,j} - aC \rightarrow max \quad (2.1)$$

де w_t — ваговий коефіцієнт для побажань викладачів;

w_s — ваговий коефіцієнт для побажань студентів;

N — кількість занять;

M — кількість часових слотів.

Пояснення формули:

1. Перша частина ($w_t \sum_{i=1}^{N} \sum_{j=1}^{M} P_t(i,j) x_{i,j}$): враховує побажання викладачів. Ми додаємо бали за кожне заняття, яке відбувається у бажаний для викладачів час.
2. Друга частина ($w_s \sum_{i=1}^{N} \sum_{j=1}^{M} P_s(i,j) x_{i,j}$): враховує побажання студентів. Ми додаємо бали за кожне заняття, яке відбувається у бажаний для студентів час.
3. Третя частина ($-aC$): враховує конфлікти. Ми віднімаємо бали за кожен конфлікт у розкладі.

Завдання оптимізації:

Метою є максимізація значення F. Це означає, що ми хочемо отримати розклад, який максимально відповідає побажанням викладачів та студентів і мінімізує конфлікти [17].

2.3 Модель розкладу занять

З огляду на наведені вище дані, ми можемо створити математичну модель розкладу. Розклад буде представлений у форматі, який використовується в Донецькому національному університеті імені Василя

Стуса. Цей формат показано в таблиці 2.1.

Таблиця 2.1 Модель розкладу занять в ДонНУ

День тижня	Час навчання	Група студентів			
		Дисципліна	Тип заняття	Аудиторія	Викладач

Визначення первинного ключа в даній таблиці не є очевидним, оскільки це залежить від підходу до вирішення завдання. У роботі первинним ключем обрано комбінацію з дисципліни, типу заняття та групи студентів, у якій це заняття буде проводитись, оскільки саме ця інформація визначає процес навчання з точки зору студента. Варто зазначити, що викладач також є важливим полем, але оскільки один предмет можуть викладати кілька викладачів, це робить викладача неунікальним ключем. Також слід враховувати, що один предмет може викладатися у кількох групах. Щоб уникнути проблем, ми обираємо таку модель, оскільки інші моделі можуть спричинити додаткові труднощі [18].

Така модель, як описано в таблиці 2.1, потребує спрощення та оптимізації. Всю інформацію про розклад ми будемо представляти у вигляді паралелепіпеда. Спочатку ми розділимо її на чотири групи, об'єднавши певні поля, як показано у формулах 2.2, 2.3, 2.4, 2.5.

$$X_1 =< \text{День тижня} > \quad (2.2)$$

$$X_2 =< \text{Час навчання} > \quad (2.3)$$

$$X_3 =< \text{Аудиторія} > \quad (2.4)$$

$$Z =< \text{Викладач} - (\text{Дисципліна} - \text{Тип заняття}) - \text{Група студентів} > \quad (2.5)$$

Дисципліна і Тип заняття (далі Дисципліна) об’єднуються, оскільки їх поєднання однозначно визначає одиницю навчального процесу. День, Пара і Аудиторія залишаються окремими, оскільки вони відображають фізичні обмеження, порушення яких вказує на некоректний розклад [19]. Наприклад, якщо в одній аудиторії одночасно призначено два заняття. Поля Дисципліна

– Група – Викладач будуть розташовані у вузлах паралелепіпеда, що у реальності важко уявити. Тому запис у списку розкладу в моделі виглядатиме, як показано у формулі (2.6).

$$s_n = (D_i, T_j, R_k, N_l) \tag{2.6}$$

де s_n — розклад;

D_i— день тижня;

T_j — час навчання;

R_k— аудиторія;

N_l — об’єднання < Викладач – Дисципліна – Група студентів >.

Виходячи з наведеної інформації, можна визначити розмірність паралелепіпеда, який матиме три виміри з четвертим у вузлах. Розмірність куба буде статичною, оскільки навчальний тиждень складається з 6 днів, а в один день не може бути більше 6 пар. Кількість аудиторій буде братися з бази даних аудиторного фонду факультету [20].

Графічне зображення розкладу можна побачити на рисунку 2.1.

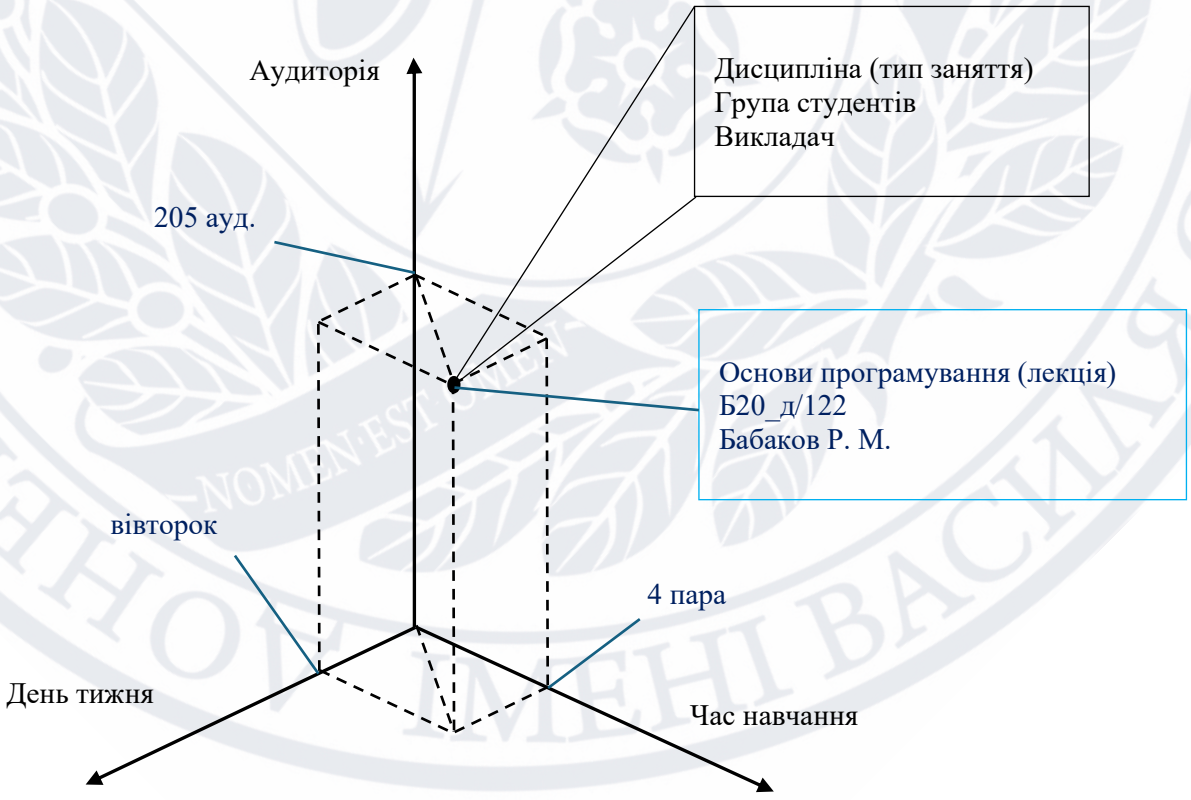

Рисунок 2.1 – Графічне уявлення розкладу занять

Для кращого розуміння рисунку 2.1 розглянемо приклад. Припустимо, ключ включає лекцію з дисципліни "Основи програмування", яку буде проводити викладач Бабаков Роман Маркович у групи Б20_д/122. Лекція буде проходити в аудиторії 205 у вівторок на четвертій парі. Отже, як показано на рисунку 2.1, у паралелепіпеді за координатами (2, 4, 205), що відповідає другому дню тижня (вівторок), четвертій парі та відповідній аудиторії, буде розташована комірка:

<Основи програмування (лекція) – Б20_д/122 – Бабаков Р. М.>

Очевидно, що в цей час і в цій аудиторії ніхто інший не може бути, оскільки це місце в паралелепіпеді вже зайняте.

2.4 Розробка алгоритму для генерації розкладу занять

Після вибору моделі розкладу та цільової функції, слід розглянути метод оптимізації розкладу. Це метод, який дозволить покращити розклад таким чином, щоб він відповідав вимогам, визначеним у цільовій функції [21].

Для виконання цього завдання обрано еволюційний генетичний алгоритм. Основна ідея цього методу полягає в тому, що кожна наступна генерація розкладів формується на основі попередніх ітерацій. Після цього відбувається відбір, під час якого всі менш вдалі розклади відкидаються. Таким чином, цільова функція покращується без втручання користувача, оскільки ця модель є автономною. Якість отриманих результатів буде постійно зростати [22].

Загальні кроки, які реалізує генетичний алгоритм, наступні [28]:

1. Створення першої генерації розкладів випадковим чином.

   На початку генерується початкова популяція розкладів, які можуть бути абсолютно випадковими.

2. Оцінка цих розкладів цільовою функцією.

   Кожен розклад оцінюється за допомогою цільової функції, яка враховує побажання викладачів та студентів, а також уникнення конфліктів.

3. Генерація нового “потомства” з попередньої ітерації**.**

   На основі кращих розкладів з попередньої генерації створюються нові розклади. Це включає в себе операції схрещування та мутації, які додають варіативність у популяцію.

4. Відкидання гірших розкладів.

   Після генерації нової популяції найгірші розклади відкидаються, залишаючи тільки найбільш оптимальні.

5. Повторення кроків 2-4 до досягнення заданої точності або перевищення максимальної кількості ітерацій.

Ці кроки повторюються до тих пір, поки розклад не буде відповідати заданим критеріям або поки кількість ітерацій не перевищить допустимий максимум. Для зручності алгоритм можна представити у вигляді схеми, як показано на рисунку 2.2:

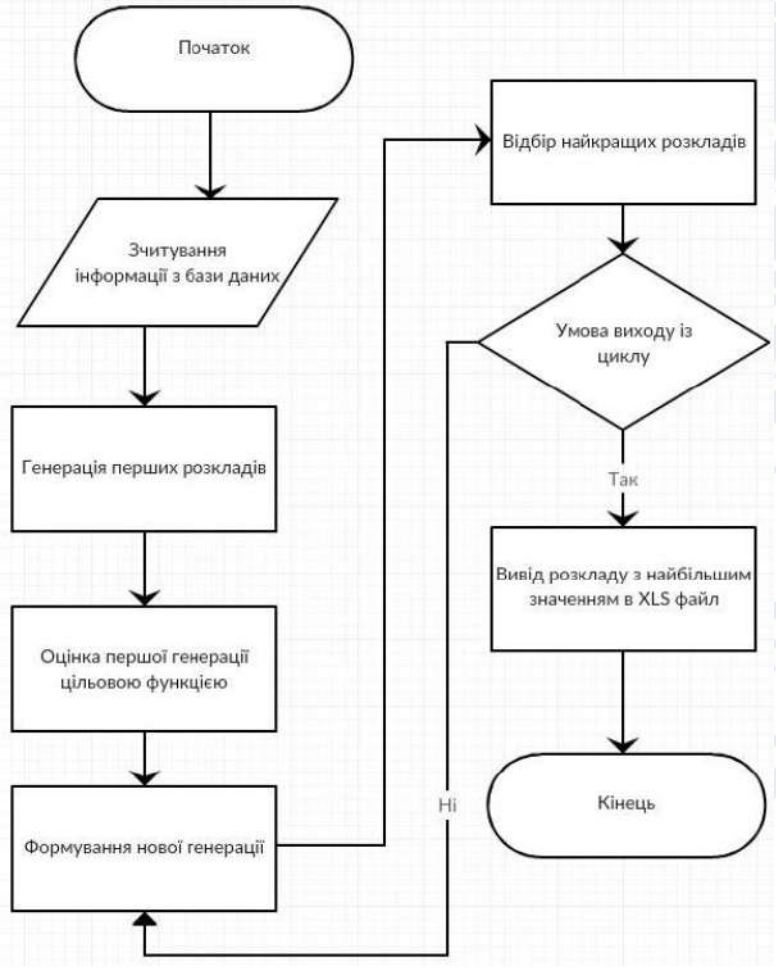

Рисунок 2.2 - Блок-схема генетичного алгоритму

Таким чином, генетичний алгоритм дозволяє автоматизувати процес створення оптимального розкладу занять, враховуючи всі необхідні критерії та обмеження.

Створення першої генерації

Перший етап генетичного алгоритму полягає в створенні початкової популяції розкладів. Цей етап є критично важливим, оскільки від якості початкових розкладів залежить ефективність подальших ітерацій алгоритму [23].

Кроки створення першої генерації:

1. Визначення початкових параметрів.

   Перед тим, як розпочати генерацію, потрібно визначити розмір популяції. Це число розкладів, які будуть створені на першій ітерації. Зазвичай розмір популяції визначається експериментально і може змінюватися в залежності від конкретних вимог та обмежень університету.

2. Випадкове призначення занять.

   Кожен розклад у початковій популяції створюється випадковим чином. Це означає, що заняття (лекції, семінари, лабораторні роботи) розподіляються по доступним слотам (час і аудиторія) випадково. Враховуються основні обмеження, такі як:

   - Кількість пар в день;
   - Доступність аудиторій;
   - Розподіл викладачів.

3. Уникнення конфліктів.

   Навіть на етапі випадкової генерації важливо мінімізувати кількість конфліктів. Наприклад, двом заняттям не можна призначити одну і ту ж аудиторію одночасно, і викладач не може бути присутнім на двох заняттях одночасно. Для цього можна використовувати базові алгоритми перевірки на

конфлікти, які будуть застосовані до кожного створеного розкладу.

4. Запис розкладу до списку згенерованих розкладів.

   Після створення розкладу він додається до списку згенерованих розкладів. Це дозволяє зберегти всі створені розклади для подальшої оцінки та відбору. Запис включає всі деталі розкладу: дисципліни, тип занять, групи, викладачі, аудиторії, дні та пари. Важливо забезпечити унікальність кожного записаного розкладу, щоб популяція була максимально різноманітною.

Блок-схема створення першої генерації розкладу занять зображена на рисунку 2.3.

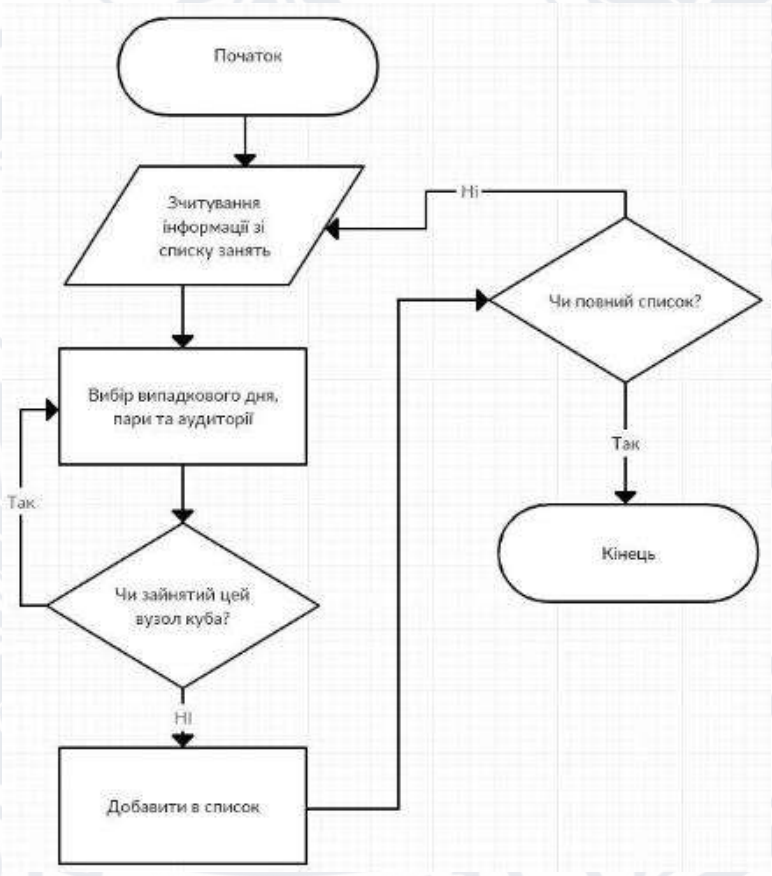

Рисунок 2.3 - Блок-схема генерації першого розкладу занять

Як приклад, припустимо, що у нас є такі дані: 4 дисципліни (математика, фізика, інформатика, хімія), кожна з яких має по 2 заняття на тиждень, 3 групи студентів (Група А, Група В, Група С), і по 2 викладачі для

кожної дисципліни. Також у нашому розпорядженні є 4 аудиторії. Розклад потрібно скласти на 5 днів на тиждень з 4 парами на день.

На першому етапі створення першої генерації розкладів визначаємо розмір популяції, наприклад, 20 розкладів. Для кожного розкладу випадково призначаємо заняття для кожної групи. При цьому переконуємося, що кожне заняття призначене в межах доступних слотів (час і аудиторія). Наприклад, для першого розкладу, заняття з математики для Групи А можуть бути призначені на понеділок на першій парі в аудиторії 101, а для Групи В на вівторок на другій парі в аудиторії 102. Важливо уникати конфліктів, таких як подвійне використання аудиторії або викладача. Якщо випадково призначене заняття викликає конфлікт (наприклад, один викладач має бути одночасно в двох різних аудиторіях), алгоритм коригує призначення, щоб уникнути цього конфлікту.

Після створення кожного розкладу, він додається до списку згенерованих розкладів. Кожен розклад включає всі деталі: дисципліни, тип занять, групи, викладачі, аудиторії, дні та пари. Наприклад, розклад може виглядати так: "Математика, лекція, Група А, Викладач 1, Понеділок, Перша пара, Аудиторія 101". Таким чином, після завершення цього етапу, ми маємо набір початкових розкладів, готових для подальшої оптимізації за допомогою генетичного алгоритму.

Отже, на першому етапі ми отримуємо першу генерацію розкладів, яка є відправною точкою для подальших ітерацій генетичного алгоритму. Важливо мінімізувати початкові конфлікти, щоб прискорити процес пошуку оптимального рішення.

Змішування та створення нової генерації

Змішування та створення нової генерації є ключовим етапом у генетичному алгоритмі, що дозволяє поступово покращувати розклад занять, відбираючи та комбінуючи найкращі рішення з попередньої генерації [29].

Цей процес включає кілька основних кроків:

1. Відбір батьків: На цьому етапі вибираються розклади (батьки) з поточної генерації, які будуть використовуватися для створення нових розкладів (нащадків). Відбір здійснюється на основі їхньої оцінки цільовою функцією. Розклади з кращими оцінками мають більшу ймовірність бути вибраними для змішування.
2. Кросовер (схрещування): Після відбору батьків здійснюється процес кросоверу, де двоє батьків комбінуються для створення одного або більше нащадків. Це може бути реалізовано різними способами, наприклад, одноточковим або багатоточковим кросовером. Основна ідея полягає в тому, щоб нові розклади успадковували частину характеристик від кожного з батьків.
3. Мутація: Для забезпечення різноманітності в популяції застосовується мутація, де випадкові зміни вносяться в деякі розклади. Це допомагає уникнути локальних максимумів і покращує здатність алгоритму знаходити оптимальне рішення. Мутація може полягати у випадковому перенесенні заняття в інший час або аудиторію.
4. Формування нової генерації: Нові розклади (нащадки) додаються до популяції, і процес повторюється до досягнення заданої кількості генерацій або до отримання розкладу, який задовольняє всі вимоги.

Блок-схема змішування та створення нової генерації відображена на рисунку 2.4.

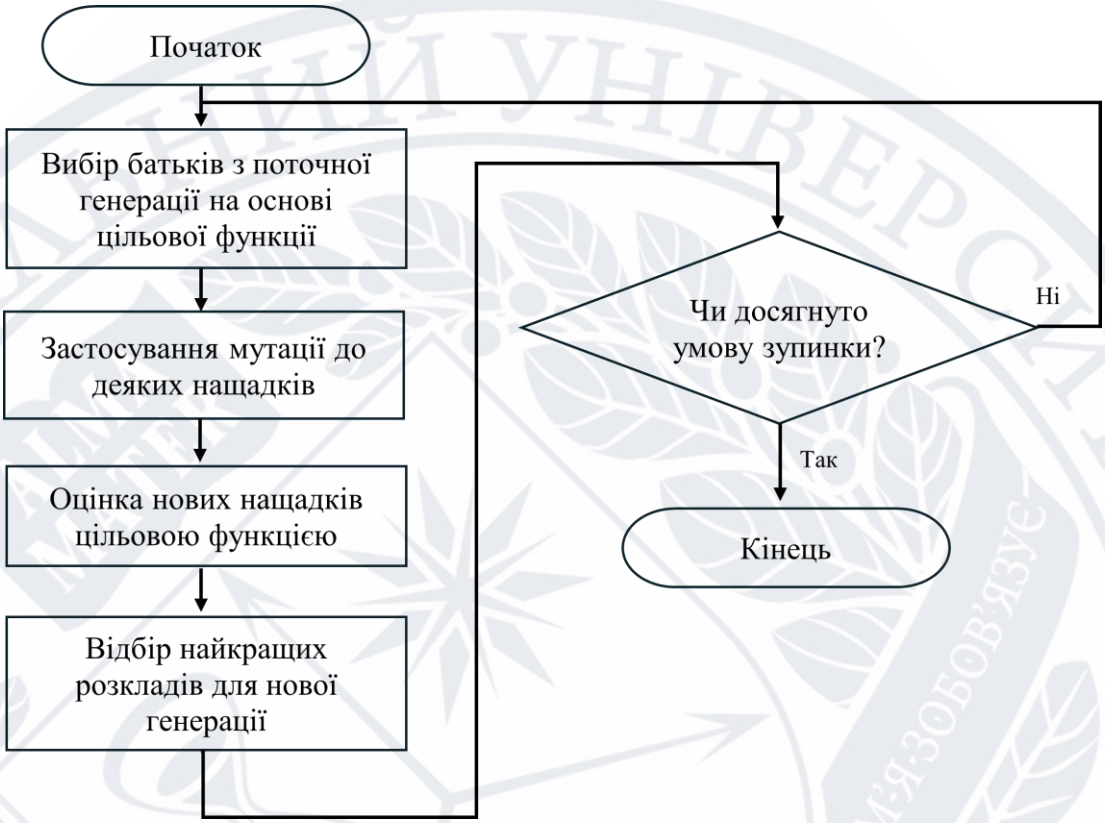

Рисунок 2.4 - Блок-схема змішування та створення нової генерації

Селекція та струс

Після того, як була сформована нова популяція розкладів шляхом схрещування та мутації, настає етап селекції та струсу, які грають важливу роль у вдосконаленні результатів генетичного алгоритму.

Селекція включає в себе відбір найкращих кандидатів для наступного покоління розкладів. Після оцінки кожного розкладу за допомогою цільової функції відбираються ті, які відповідають заданим критеріям краще за всіх. Ці кращі розклади стають батьківськими для наступного покоління розкладів.

Струс є етапом, який дозволяє внести випадкові зміни в обраних розкладах. Це може включати зміни в днях тижня, часі проведення занять, аудиторіях та інших параметрах. Мета струсу полягає в тому, щоб розширити різноманіття в популяції розкладів та сприяти дослідженню нових варіантів. Випадкові зміни можуть допомогти алгоритму уникнути зачеплення в локальних мінімумах та забезпечити кращий розвиток рішень.

Ці два етапи разом допомагають забезпечити ефективний та швидкий процес оптимізації, щоб знайти найкращий розклад занять у відповідності з вимогами та обмеженнями [30].

Завершення генетичного алгоритму

Вихід з генетичного алгоритму може відбутися за кількома критеріями:

- досягнення заданої точності: цей критерій використовується, коли відомо, який рівень точності необхідний для вирішення задачі. Це може бути досягнення певного значення цільової функції або покращення розкладу до заданого рівня якості;
- досягнення максимальної кількості ітерацій: алгоритм може бути встановлений на певну кількість ітерацій, після якої він автоматично завершить свою роботу. Це може бути корисно для управління часом обчислень та обмеження ресурсів;
- стабілізація результатів: якщо після певної кількості ітерацій результати алгоритму не змінюються або змінюються дуже мало, можна вважати, що алгоритм зійшов на плато і далі його виконання не має сенсу;
- задоволення заданих обмежень: іноді може виникнути ситуація, коли знайдений розклад відповідає усім обмеженням, що були задані для задачі. У цьому випадку алгоритм може бути завершений.

Ці критерії гарантують ефективне та адаптивне завершення роботи генетичного алгоритму, забезпечуючи оптимальність результатів та враховуючи різноманітні умови та обмеження задачі формування розкладу занять [31].

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ

3.1 Вибір середовища розробки

Для реалізації системи автоматизації формування розкладу занять було обрано мову програмування C# з наступних причин [32]:

- простота та зручність використання: C# є мовою високого рівня з зрозумілим та логічним синтаксисом, що дозволяє швидко освоїти основи програмування, що спрощує написання та підтримку коду;
- об'єктно-орієнтоване програмування (ООП): C# підтримує принципи ООП, такі як інкапсуляція, наслідування та поліморфізм. Це дозволяє створювати модульний, розширюваний та підтримуваний код, що важливо для великих проектів;
- потужні інструменти та бібліотеки: C# має широкий набір стандартних бібліотек у .NET Framework та .NET Core, які забезпечують функціональність для роботи з файлами, мережевими протоколами, графічними інтерфейсами та базами даних;
- підтримка баз даних: C# забезпечує зручні інструменти для роботи з різними типами баз даних, включаючи MySQL, завдяки технологіям ORM (Object-Relational Mapping), таким як Entity Framework;
- безпека та управління пам'яттю: C# має вбудовані механізми управління пам'яттю та безпеки, що підвищує надійність додатків;
- підтримка та спільнота: C# має велику та активну спільноту розробників, що забезпечує наявність великої кількості ресурсів,

таких як документація, форуми та навчальні матеріали.

Для зберігання та управління даними системи було обрано систему управління базами даних MySQL з наступних причин [33]:

- продуктивність: MySQL відома своєю високою продуктивністю та швидкістю обробки запитів, що критично важливо для системи, яка працює з великими обсягами даних про розклади, заняття та ресурси;
- надійність та стабільність: MySQL є однією з найбільш надійних та стабільних СУБД, яка забезпечує високу доступність даних та мінімальні простої завдяки вбудованим механізмам реплікації та резервного копіювання;
- масштабованість: MySQL підтримує горизонтальне масштабування, що дозволяє додавати нові сервери для розподілу навантаження та підвищення продуктивності системи;
- зручність використання: MySQL має зручний та інтуїтивно зрозумілий інтерфейс, підтримує стандартний мову запитів SQL, що дозволяє швидко створювати, модифікувати та управляти базами даних;
- сумісність: MySQL легко інтегрується з різними мовами програмування та середовищами розробки, включаючи C# та Visual Studio, забезпечуючи безшовну роботу між додатком та базою даних;
- безкоштовна ліцензія: MySQL доступна за ліцензією GPL (General Public License), що дозволяє безкоштовно використовувати її у багатьох проектах, що важливо для освітніх установ та невеликих підприємств.

Для реалізації системи автоматизації формування розкладу занять було обрано середовище розробки Visual Studio з наступних причин [34]:

- підтримка мови програмування C#: Visual Studio є офіційним

середовищем розробки для C#, забезпечуючи повну інтеграцію з .NET Framework та .NET Core;

- інтеграція з MySQL: Visual Studio підтримує підключення до MySQL через плагіни, такі як MySQL for Visual Studio, дозволяючи легко працювати з базою даних безпосередньо з середовища розробки;
- зручність використання: Visual Studio надає зручний інтерфейс з потужними інструментами для розробки, такими як IntelliSense, візуальний дизайнер форм, менеджер пакетів NuGet та інші;
- інструменти для налагодження та тестування: Visual Studio включає потужні інструменти для налагодження та тестування додатків, такі як дебагер, що дозволяє відстежувати виконання коду в режимі реального часу.

Це поєднання забезпечує високу продуктивність, надійність, масштабованість, зручність використання та сумісність, що робить його оптимальним вибором для даного проекту [24].

3.2 Структура бази даних

Основна мета бази даних для програми автоматизації формування розкладу занять в університеті – забезпечити збереження та ефективне управління інформацією. Структура бази даних спроектована таким чином, щоб підтримувати цілісність даних та забезпечувати необхідні зв'язки між таблицями [35]. Опис таблиць, що містяться в базі даних:

Таблиця Departments:

Таблиця Department зберігає інформацію про кафедри університету.

- department_id (INT, PK): Унікальний ідентифікатор кафедри. Використовується для однозначного розрізнення кафедр;
- name (VARCHAR): Назва кафедри. Це поле містить повну назву кафедри.

Таблиця Specialties:

Таблиця Specialty містить дані про спеціальності, які пропонуються на різних кафедрах. Вона дозволяє визначити, які спеціальності закріплені за кожною кафедрою.

- specialty_id (INT, PK): Унікальний ідентифікатор спеціальності;
- name (VARCHAR): Назва спеціальності;
- department_id (INT, FK): Ідентифікатор кафедри, до якої належить спеціальність.

Таблиця Courses:

Таблиця Course зберігає інформацію про курси студентів різних спеціальностей.

- course_id (INT, PK): Унікальний ідентифікатор курсу;
- number (INT): Номер курсу (1, 2, 3, 4 тощо);
- specialty_id (INT, FK): Ідентифікатор спеціальності, до якої належить курс.

Таблиця Groups:

Таблиця Group зберігає дані про групи студентів, які навчаються на кожному курсі.

- group_id (INT, PK): Унікальний ідентифікатор групи;
- name (VARCHAR): Назва групи;
- course_id (INT, FK): Ідентифікатор курсу, до якого належить група;

Таблиця Subgroups:

Таблиця Subgroup містить інформацію про підгрупи студентів, які можуть бути створені для практичних або лабораторних занять.

- subgroup_id (INT, PK): Унікальний ідентифікатор підгрупи;
- name (VARCHAR): Назва підгрупи;
- group_id (INT, FK): Ідентифікатор групи, до якої належить підгрупа.

Таблиця Subjects:

Таблиця Subject містить дані про навчальні предмети, які викладаються в кожному семестрі в різних спеціальностей.

- subject_id (INT, PK): Унікальний ідентифікатор предмету;
- name (VARCHAR): Назва предмету;
- specialty_id (INT, FK): Ідентифікатор спеціальності, до якої належить предмет;
- semester (INT): Семестр, у якому викладається предмет.

Таблиця Type_lessons:

Таблиця Type_lesson містить дані про типи занять, такі як лекції, практичні та лабораторні заняття.

- type_lesson_id (INT, PK): Унікальний ідентифікатор типу заняття;
- name (VARCHAR): Назва типу заняття (лекція, практика, лабораторна).

Таблиця Classrooms:

Таблиця Classroom містить інформацію про аудиторії, які використовуються для проведення занять.

- classroom_id (INT, PK): Унікальний ідентифікатор аудиторії;
- name (VARCHAR): Назва аудиторії;
- capacity (INT): Кількість посадкових місць у аудиторії;
- type (VARCHAR): Тип аудиторії (лекційна, лабораторна, комп'ютерна).

Таблиця Professors:

Таблиця Professor містить дані про викладачів, які працюють на кафедрі.

- professor_id (INT, PK): Унікальний ідентифікатор викладача;
- name (VARCHAR): Ім'я викладача;
- department_id (INT, FK): Ідентифікатор кафедри, на якій працює викладач.

Таблиця Professor_time_work:

Таблиця Professor_time_work містить інформацію про графік роботи викладачів.

- id_professor_time_work (INT, PK): Унікальний ідентифікатор запису графіку роботи;
- day_of_week (VARCHAR): День тижня;
- works_lessons (VARCHAR): Пари, в які викладач може працювати;
- professor_id (INT, FK): Ідентифікатор викладача.

Таблиця Professor_Subject:

Таблиця Professor_Subject містить дані про призначення викладачів на навчальні предмети.

- professor_subject_id (INT, PK): Унікальний ідентифікатор призначення;
- professor_id (INT, FK): Ідентифікатор викладача;
- subject_id (INT, FK): Ідентифікатор предмету;
- type_lesson_id (INT, FK): Ідентифікатор типу заняття.

Таблиця Subject_plans:

Таблиця Subject_plans містить інформацію про навчальний план для кожної навчальної дисципліни.

- subject_plan_id (INT, PK): Унікальний ідентифікатор запису.
- subject_id (INT, FK): Ідентифікатор предмету.
- type_lesson_id (INT, FK): Ідентифікатор типу заняття.
- weekly_hours (INT): Кількість годин на тиждень для даного типу занять.

Таблиця Schedules:

Таблиця Schedule містить інформацію про розклад занять для кожної групи студентів.

- schedule_id (INT, PK): Унікальний ідентифікатор запису

розкладу;

- day_of_week (VARCHAR): День тижня;
- time_slot (INT): Час навчання (1 пара - 6 пара);
- group_id (INT, FK): Ідентифікатор групи;
- subgroup_id (INT, FK, NULLABLE): Ідентифікатор підгрупи (може бути NULL, якщо заняття для всієї групи);
- subject_id (INT, FK): Ідентифікатор предмету;
- type_lesson_id (INT, FK): Ідентифікатор типу заняття;
- classroom_id (INT, FK): Ідентифікатор аудиторії;
- professor_id (INT, FK): Ідентифікатор викладача.

На рисунку 3.1 зображена загальна схема бази даних з зв'язками між таблицями.

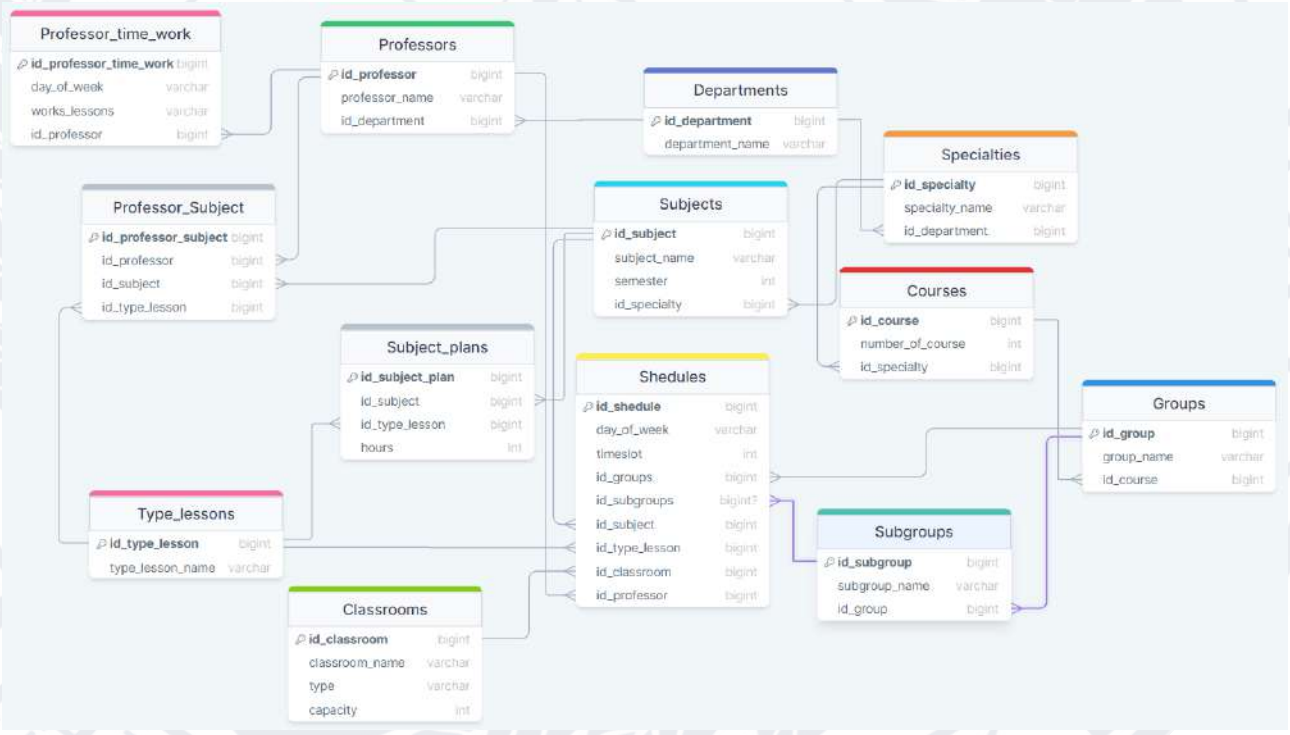

Рисунок 3.1 – Схема бази даних

Розроблена база даних забезпечує структуроване збереження всіх необхідних даних для автоматизації формування розкладу занять в університеті. Вона дозволяє ефективно управляти необхідною інформацією. Така структура забезпечує цілісність даних та полегшує процес формування розкладу, враховуючи всі необхідні обмеження та вимоги.

3.3 Архітектура системи

Архітектура системи розроблена таким чином, щоб забезпечити ефективне виконання всіх необхідних функцій на одному пристрої. Передбачається, що система буде використовуватись одним користувачем, відповідальним за формування розкладу занять [25]. Схема архітектури системи автоматизації формування розкладу занять зображена на рисунку 3.2:

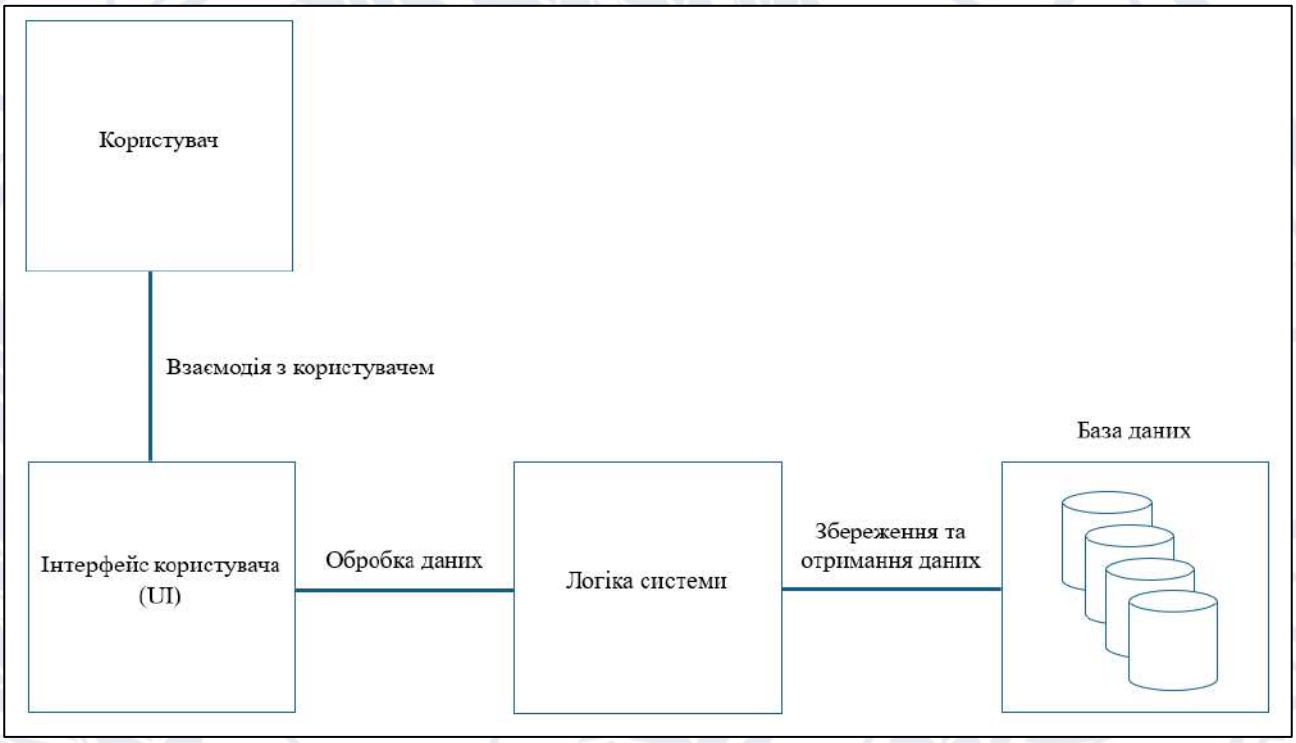

Рисунок 3.2 – Схема архітектури системи

Основні компоненти системи включають [26]:

- інтерфейс користувача (UI): інтерфейс, який дозволяє користувачеві взаємодіяти із системою для введення даних, перегляду інформації та управління процесом формування розкладу. Інтерфейс розроблений з використанням Windows Forms на мові програмування C#. Це дозволяє створювати зручні та інтуїтивно зрозумілі графічні інтерфейси з підтримкою сучасних UI/UX концепцій;
- логіка додатку: основний модуль, що виконує всю бізнес-логіку системи. Логіка додатку реалізована з використанням .NET Framework, що забезпечує високу продуктивність і надійність.

Цей модуль включає алгоритми формування розкладу, валідацію даних та інші необхідні операції;

- база даних: система управління базами даних MySQL використовується для зберігання всіх даних, необхідних для формування розкладу. База даних включає таблиці для зберігання інформації про кафедри, спеціальності, курси, групи, підгрупи, предмети, викладачів, аудиторії та розклад занять. Вибір MySQL обумовлений її високою продуктивністю, надійністю та зручністю використання;

Процес взаємодії компонентів:

1. Введення даних: Користувач вводить дані через інтерфейс користувача (UI). Це можуть бути дані про нові кафедри, спеціальності, курси, групи, предмети, викладачів та аудиторії.
2. Збереження даних: Введені дані передаються до логіки додатку, яка перевіряє їх на коректність та зберігає у базі даних MySQL. Всі дані зберігаються з урахуванням встановлених зв'язків між таблицями для забезпечення цілісності інформації.
3. Формування розкладу: Коли всі необхідні дані введені, користувач запускає процес формування розкладу. Логіка додатку виконує генетичний алгоритм, який генерує оптимальний розклад занять. Алгоритм враховує всі обмеження та побажання, визначені користувачем, та генерує розклад з мінімальною кількістю конфліктів.
4. Перегляд та редагування розкладу: Згенерований розклад відображається в інтерфейсі користувача. Користувач може переглядати розклад для різних груп, викладачів та аудиторій. У разі необхідності, розклад можна редагувати вручну через інтерфейс користувача.
5. Збереження розкладу: Після завершення редагування, остаточний розклад зберігається у базі даних MySQL. Користувач може

експортувати розклад у зручному форматі для подальшого використання.

Переваги вибраної архітектури

- простота та зручність: Використання C# та WPF забезпечує простоту реалізації та зручність використання інтерфейсу;
- висока продуктивність: .NET Framework та MySQL забезпечують високу продуктивність та надійність роботи системи;
- легкість підтримки та розширення: Об'єктно-орієнтований підхід та модульна структура забезпечують легкість підтримки та можливість подальшого розширення функціональності системи;
- єдине робоче середовище: Всі компоненти системи встановлені та працюють на одному пристрої, що спрощує управління та підтримку.

Загалом, обрана архітектура забезпечує всі необхідні умови для ефективного формування розкладу занять в університеті, враховуючи обмеження на використання системи одним користувачем.

3.4 Опис основних класів системи

Для автоматизації формування розкладу занять в університеті, розглянемо три основні Класи [36]:

- DatabaseManager: клас, що отримує дані з бази даних і записує дані в базу даних;
- ScheduleGenerator: клас, що формує розклад занять з урахуванням всіх вимог і обмежень, оптимізує сформований розклад за допомогою генетичного алгоритму і перевіряє його на правильність;
- ExcelExporter: Клас, що зберігає розклад у форматі Excel.

Клас DatabaseManager:

Призначення: Цей клас відповідає за взаємодію з базою даних. Він отримує всі необхідні дані для генерації розкладу і зберігає розклад у базу даних після його формування. Основна мета класу - забезпечити зручний і ефективний інтерфейс для роботи з базою даних.

Методи:

- GetClassrooms: Повертає список всіх аудиторій з бази даних;
- GetProfessors: Повертає список всіх викладачів з бази даних;
- GetGroups: Повертає список всіх студентських груп з бази даних;
- GetSubjects: Повертає список всіх навчальних предметів з бази даних;
- GetDepartments: Повертає список всіх кафедр з бази даних;
- GetSpecialties: Повертає список всіх спеціальностей з бази даних;
- GetCourses: Повертає список всіх курсів з бази даних;
- GetSubgroups: Повертає список всіх підгруп студентів з бази даних;
- GetTypeLessons: Повертає список всіх типів занять з бази даних;
- GetProfessorTimeWork: Повертає розклад роботи викладачів;
- GetProfessorSubjects: Повертає призначення викладачів на навчальні предмети;
- GetSubjectPlans: Повертає навчальні плани для кожної дисципліни;
- SaveInSchedule: Зберігає дані у базу даних.

Призначення:

- GetClassrooms, GetProfessors, GetGroups, GetSubjects, GetDepartments, GetSpecialties, GetCourses, GetSubgroups, GetTypeLessons, GetProfessorTimeWork, GetProfessorSubjects, GetSubjectPlans: ці методи призначені для отримання відповідних даних з бази даних. Вони використовуються для завантаження

початкових даних, необхідних для генерації розкладу;

- SaveInSchedule: забезпечує збереження даних у базі після їх редагування.

Клас ScheduleGenerator:

Призначення: Цей клас відповідає за генерацію розкладу занять з урахуванням всіх вимог і обмежень. Він також здійснює оптимізацію розкладу за допомогою генетичного алгоритму і перевіряє його на правильність. Основна мета класу - створити ефективний розклад, який задовольняє всі встановлені обмеження і вимоги.

Методи:

- LoadData: завантажує необхідні дані для формування розкладу занять з бази даних через DatabaseManager. Для цього викликає методи отримання даних;
- GenerateInitialSchedule: генерує початковий розклад, базуючись на завантажених даних. Використовує отримані дані для створення першого варіанту розкладу;
- OptimizeSchedule: оптимізує початковий розклад за допомогою генетичного алгоритму, враховуючи всі обмеження та вимоги. Застосовує генетичний алгоритм для покращення початкового розкладу;
- ValidateSchedule: перевіряє розклад на правильність, забезпечуючи виконання навчального плану і відсутність конфліктів. Перевіряє, чи відповідає розклад всім встановленим вимогам;
- GetSchedule: повертає сформований і оптимізований розклад.

Клас ExcelExporter:

Призначення: Цей клас відповідає за збереження розкладу у форматі Excel. Він приймає розклад у вигляді вхідних даних і зберігає його у вигляді файлу Excel, форматуючи дані відповідно до форми розкладу занять у

Донецькому національному університеті імені Василя Стуса. Основна мета класу - надати можливість зберігати і переглядати розклад у зручному і широко використовуваному форматі.

Методи:

- ExportScheduleToExcel: експортує переданий розклад у Excel файл за вказаним шляхом. Форматує дані відповідно до заданих вимог (назва групи, день тижня, час заняття, дисципліна, тип заняття, аудиторія, викладач).

Ці класи та методи забезпечують функціональність для отримання даних з бази даних, генерацію і оптимізацію розкладу, перевірку розкладу на правильність і збереження його у форматі Excel.

3.5 Формування тестового розкладу занять

Інтерфейс програми зображений на рисунку 3.3 і складається з основного вікна, яке містить чотири вкладки: “Аудиторії”, “Викладачі”, “Сформувати розклад” і “Редагувати розклад”.

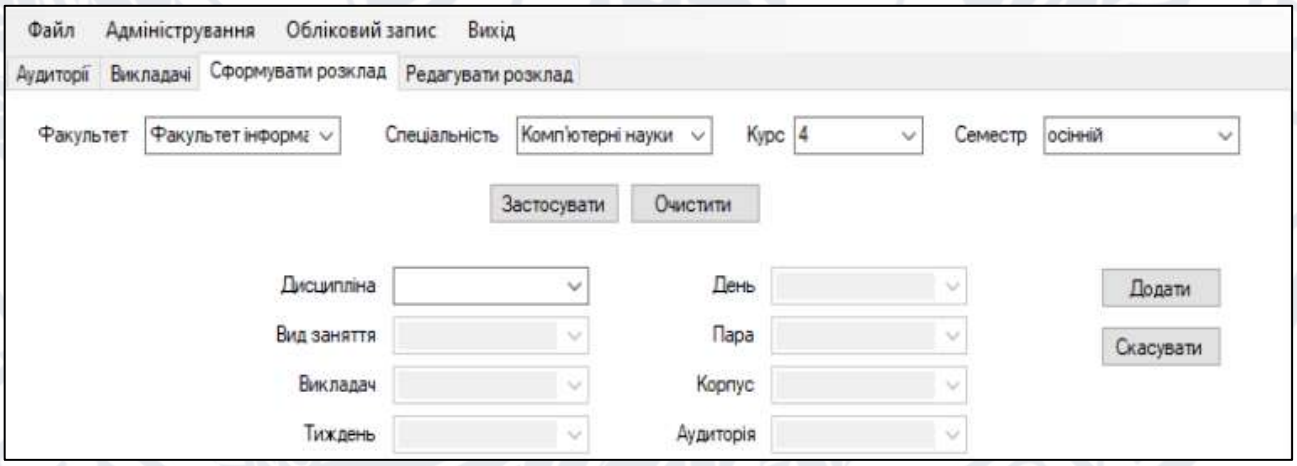

Рисунок 3.3 – Інтерфейс користувача

На вкладці “Аудиторії” користувач може переглянути зайнятість конкретної аудиторії або знайти всі вільні аудиторії певного типу, з певним обладнанням. Вкладка “Викладачі” надає інформацію про графік роботи викладача. Користувач може фільтрувати інформацію за конкретною дисципліною, яку викладач веде. На вкладці “Сформувати розклад”

користувач може створити розклад для закладу освіти. Перед початком формування розкладу потрібно обрати факультет, спеціальність, курс та семестр, на якому буде складено розклад. Після цього потрібно натиснути кнопку "Застосувати" і перейти до додавання навчальних занять до розкладу. Крім того, передбачено можливість редагування розкладу на вкладці “Редагувати розклад”.

Після успішної генерації розкладу, є можливість зберегти його у файлі формату XLSX на комп'ютері в папці з програмою. Це зроблено для зручності подальшого використання. Приклад збереженого розкладу занять зображений на рисунку 3.4:

День тижня	Час навчання	Б20_д/122			
		Дисципліна	Тип заняття	Аудиторія	Викладач
Понеділок	III	Технології TextMining та WebMining	Лек.	205-Кристал/ Microsoft Teams	Ніколюк П.К., проф.
	IV	Технології TextMining та WebMining	Лаб. (А)	303 А-Кристал/ Microsoft Teams	Поремський Ю.В.
		Технології TextMining та WebMining	Лаб. (В)	304-Кристал/ Microsoft Teams	Ніколюк П.К., проф.
	V	Технології TextMining та WebMining	Лаб. (Б)	303 А-Кристал/ Microsoft Teams	Поремський Ю.В.
		Технології TextMining та WebMining	Лаб. (Г)	304-Кристал/ Microsoft Teams	Ніколюк П.К., проф.
Вівторок	II	Прикладні комп`ютерні технології	Лек.	307-Кристал/ Microsoft Teams	Січко Т.В., доц.
	III	Технології розподілених та паралельних обчислень	Лек.	205-Кристал/ Microsoft Teams	Бабаков Р.М., проф.
	V	Інтелектуальний аналіз даних та методи штучного інтелекту	Лек.	Microsoft Teams	Ротштейн О.П., проф.
	VI	Інтелектуальний аналіз даних та методи штучного інтелекту	Лаб. (А)	Microsoft Teams	Ротштейн О.П., проф.
Четвер	III	Технології розподілених та паралельних обчислень	Лаб. (Г)	303 А-Кристал/ Microsoft Teams	Бабаков Р.М., проф.
	IV	Технології розподілених та паралельних обчислень	Лаб. (Б)	303 А-Кристал/ Microsoft Teams	Бабаков Р.М., проф.
		Інтелектуальний аналіз даних та методи штучного інтелекту	Лаб. (В)	Microsoft Teams	Ротштейн О.П., проф.
	IV	Технології розподілених та паралельних обчислень	Лаб. (В)	303 А-Кристал/ Microsoft Teams	Бабаков Р.М., проф.
		Іноземна мова професійного спрямування	Пр. (А)	128-Кристал/ Microsoft Teams	Лозинська Л. Ф., доц.
		Прикладні комп`ютерні технології	Лаб. (Б)	304-Кристал/ Microsoft Teams	Січко Т.В., доц.

Рисунок 3.4 – Приклад згенерованого розкладу занять

Згідно з отриманим розкладом, всі вимоги дотримані належним чином: відсутні будь-які повтори, і ні один викладач чи група студентів не перебувають у двох різних місцях одночасно. Аудиторії відповідно вибрані в залежності від чисельності студентів та типу занять.

ВИСНОВКИ

У бакалаврській роботі було проведено комплексне дослідження та розробка системи автоматизації формування розкладу занять для університету. Було проведено вивчення та аналіз існуючих систем формування розкладу занять у вищих навчальних закладах, таких як Ad Astra та Open EduCat. Аналіз показав, що хоч ці системи є потужними інструментами для автоматизації формування розкладу занять, вони мають свої обмеження та недоліки, що можуть ускладнювати процес формування розкладу та його оптимізацію.

Для вирішення складних завдань, пов'язаних з формуванням розкладу занять у вищих навчальних закладах, були розглянуті нові методи та алгоритми, які забезпечують ефективне та оптимальне розподілення навчальних занять. У цьому контексті був використаний генетичний підхід, який є одним з найбільш потужних інструментів оптимізації. Головною метою використання генетичного підходу було створення такого розкладу, який би враховував всі специфічні вимоги та обмеження, що існують у конкретному університеті. Це включало в себе не лише уникнення конфліктів у графіку викладачів та груп студентів, але й забезпечення оптимального використання доступних аудиторій. Такий підхід дозволяє створювати більш збалансовані та ефективні розклади, що відповідають потребам навчальних закладів.

Створена система автоматизації формування розкладу занять була реалізована з використанням передових технологій програмування, зокрема, Windows Forms для розробки інтерфейсу користувача та MySQL для створення бази даних. Використання Windows Forms дозволило створити інтуїтивно зрозумілий і зручний інтерфейс, що сприяє ефективній взаємодії з програмою. Інтерфейс користувача був спроектований з урахуванням потреб користувачів, забезпечуючи зручний доступ до всіх функцій системи. У свою чергу, використання бази даних MySQL було обумовлене потребою у надійному зберіганні та оптимальному управлінні даними системи. MySQL

відома своєю високою надійністю та швидкодією, що робить її ідеальним вибором для використання в системах, які потребують стабільної та ефективної роботи.

Такий підхід до реалізації системи дозволив забезпечити не лише зручність та ефективність у використанні для користувачів, але й надійність та оптимальну продуктивність системи в цілому. Результатом є стабільна та функціональна система, яка відповідає сучасним вимогам та сприяє ефективному формуванню розкладу занять у навчальних закладах.

Отже, система автоматизації формування розкладу занять з використанням генетичного алгоритму є ефективним та надійним інструментом для оптимізації розкладу університету. Її впровадження може покращити якість навчання, сприяти раціональному використанню ресурсів та задовольнити потреби користувачів.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Островська Г. В., Мічківський С. М. Формування розкладу заліково-екзаменаційної сесії у вищих навчальних закладах. Матеріали наукової конференції професорсько-викладацького складу, наукових працівників і здобувачів наукового ступеня за підсумками науково-дослідної роботи за період 2017–2018 рр. (16–17 травня 2019 р.): у 2-х томах. Том 2. Вінниця: Донецький національний університет імені Василя Стуса, 2019. С. 110-111.
2. Мулява І. Я. Система формування розкладу навчального занять з використанням суб'єктивних переваг. Міжнародний науковий журнал. 2016. № 7. С. 22-27.
3. Методична система забезпечення навчання у ВНЗ. 2020. URL: http://moodle.ipo.kpi.ua/moodle/file.php/903/9_lecture.pdf (дата звернення 24.03.2024).
4. Мачинська Н. І., Стельмах С. С. Сучасні форми організації навчального процесу у вищій школі. 2012. URL: http://baltijapublishing.lv/omp/index.php/bp/catalog/download/330/9145/19071-1?inline=1 (дата звернення 24.03.2024).
5. Інформаційно-аналітична система управління навчальним процесом ЗВО / Ю.В. Триус, І.В. Стеценко, І.В. Герасименко, В.Г. Гриценко // Інформаційні технології в освіті: Збірник наукових праць. Випуск 9.– Херсон: Видавництво ХДУ, 2011. – С. 39-48.
6. Мельник О. О. Одноетапні задачі теорії розкладів у багаторівневій системі планування: автореф. дис. На здобуття наук. ступеня канд. техн. наук: спец. 05.13.06 «Інформаційні технології» / Олена Олексіївна Мельник, Національний університет «Львівська політехніка». – Львів, 2013. – 26 с.
7. Astra Schedule. URL: https://www.aais.com/astra-schedule (last accessed: 14.04.2024).

8. A. Noaman and F. Ahmed, "ERP systems functionalities in higher education", Procedia Computer Science, vol. 65, pp. 385-395, 2015. URL: https://core.ac.uk/download/pdf/82168931.pdf (last accessed: 14.04.2024).
9. OpenEducat. URL: https://openeducat.org (last accessed: 14.04.2024).
10. A. Althunibat, B. M. Al-mahadeen, F. Altarawneh and F. A. Al-Qarem, "The Acceptance of using Enterprise Resource Planning (ERP) System in Higher Education: A Case Study of Jordanian Universities", 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), pp. 315-318, 2019.
11. Сіпко О.М. Груповий та індивідуальний підходи для визначення вимог суб'єктів навчального процесу / Матеріали Другої Міжн. наук.-техн. конф. "Обчислювальний інтелект-2013". Черкаси, 2013. 241 с.
12. Сіпко О.М. Врахування вимог студентів і викладачів в задачі складання розкладу занять / Сіпко О.М., О.В. Котик // VII Міжн. школа-семінар «Теорія прийняття рішень»: праці школи-семінару, 29 вересня – 4 жовтня 2014 р. – Ужгород: УжНУ, 2014. – С. 236.
13. Войтко Б. С., Мічківський С. М. Система формування розкладу навчальних занять з використанням платформи 1С: ПІДПРИЄМСТВО. Матеріали всеукраїнської науково-практичної конференції для студентів, аспірантів та молодих вчених. Вінниця, 2020. 194 с.
14. Романченко І.С. Еволюційні методи оптимізації та їх використання у військовій галузі досліджень: монографія / І.С. Романченко, С.Л. Борисюк, М.М. Потьомкін // Центр. НДІ Збройних Сил України. – Житомир: Рута, 2015. – 127 с.: рис., табл. – Бібліогр.: с. 118-127.
15. Снитюк В.Є., Сіпко О.М. Формування цільової функції в задачі складання розкладу занять у ЗВО на основі суб'єктивних переваг. Матеріали 16-ї Міжн. конф. System Analysis and Information Technologies (SAIT 2014). Київ, 2014. 263-264 с.
16. Снитюк В.Є., Сіпко О.М. Аспекти формування цільової функції в задачі складання розкладу занять у ЗВО / Матеріали Дев'ятої Міжн. наук.-

практ. конф. «Математичне та імітаційне моделювання систем. (МОДС – 2014)». Київ-Жукін, 2014. 349-353 с.

17. Снитюк В.Є., Сіпко О.М. Еволюційний метод розв’язання задачі складання розкладу занять з використанням функції штрафу / Матеріали ІІ Міжн. конф. «Інформаційні технології та взаємодії», КНУ імені Тараса Шевченка. Київ, 2015. 109-111 с.
18. Леонова М.В. Моделювання задач складання розкладу занять у ЗВО: огляд та різні підходи до розв’язування / Вісник Запорізького національного університету. 2013. № 1. 52-59 с.
19. Деканова М. В. Математична модель і алгоритм побудови розкладу навчальних занять університету. Вісник Полоцького державного університету. Серія С. 2013. № 12. С. 24-33.
20. Шпак А. В. Математична модель формування навчальних потоків з метою оптимізації використання лекційно-практичного аудиторного фонду. Автоматика та інформатика. 2012. № 1-2. 51 с.
21. Безуглий М. О., Секірін О. І. Методи підвищення ефективності складання розкладу в умовах навчального закладу. Матеріали міжнародної науково–технічної конференції студентів, аспірантів та молодих вчених "Комп'ютерна та програмна інженерія – 2015". Донецький національний технічний університет, 2015.
22. Чорний О.П. Формування графіка процесу навчання фахівців інженерних спеціальностей / О.П. Чорний, Ю.В. Лашко, Т.П. Коваль // Інженерні та освітні технології в електротехнічних і комп’ютерних системах. – 2013. – №4.
23. Снитюк В. Є., Сіпко О. М. Технологія еволюційного формування розкладів у закладах вищої освіти. Київ: Видавець ФОП Піча Ю. В., 2022. 136 с.
24. Рубан І. В., Дуденко С. В., Бусигін Ю. В., Колмиков М. М., Трублін О. А. Аналіз сучасного програмного забезпечення для автоматизації процесу складання розкладу навчальних занять. Системи

обробки інформації. 2013. № 8 (115). С. 305–310.

25. Проншина К. О., Голуб Б. Л., Вєтрова Д. В. Програмна система формування розкладу занять у закладі вищої освіти. Математичні машини і системи. 2019. № 4. URL: http://www.immsp.kiev.ua/publications/articles/2019/2019_4/04_Golub_19.pdf (дата звернення 24.03.2024).
26. Астістова Т. І. Розробка системи керування розкладом занять у вищому навчальному закладі. Вісник Київського національного університету технологій та дизайну. 2016. № 4 (100). С. 13–14.
27. Sipko O.M. Aspects of formulation of the objective function in the problem of scheduling in higher educational institutions based on subjective preferences / Sipko O.M., Snytyuk V.Ye. // Nauka i Studia. – Polska, Przemyśl: Sp. z o.o. «Nauka i studia». – 2014. – № 15 (125). – P. 49-51.
28. Odim O.M. On the Fitness Measure of Genetic Algorithm for Generating Institutional Lecture Timetable / O.M. Odim, B.O. Oguntunde, O.O. Alli // Journal of Emerging Trends in Computing and Information Since. – 2013. – Vol. 4 – № 4. – pp. 436-444.
29. Gong D W, Jing S, Miao Z. A Set-Based Genetic Algorithm for Interval Many-Objective Optimization Problems [J]. IEEE Transactions on Evolutionary Computation, 2018, 22(99):47-60.
30. Tsai S E. Project Scheduling with Resource Constraints by Fuzzy Gantt Chart and Genetic Algorithm [J]. Journal of Aeronautics, Astronautics and Aviation, 2019, 51(4 Vol. 51 No. 4):391-402.
31. Egwali A.O. Synthesis-Algo: An Inclusive Timetable Generat-ing Algorithm / A.O. Egwali, F.A. Imouokhome// African 123 Journal of Computing & ICT. – 2012. – Vol. 5. – №4. – P. 92-100.
32. Everything You Need to Know About C#. URL: https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c- (last accessed: 05.04.2024).
33. Visual Studio. URL: https://visualstudio.microsoft.com/ (last accessed:

05.04.2024).

34. MySQL. URL: https://www.mysql.com/ (last accessed: 05.04.2024).
35. Vahideh Panahi and Nima Jafari Navimipour, "Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators", Concurrency and Computation: Practice and Experience, pp. e5218, 2019.
36. Antony E. Phillips et al., "Integer programming methods for large-scale practical classroom assignment problems", Computers & Operations Research, vol. 53, pp. 42-53, 2015.
37. K. Saule, U. Indira, B. Aleksander, Z. Gulnaz, M. Zhanl, I. Madina, et al., "Development of the Information and Analytical System in the Control of Management of University Scientific and Educational Activities", Acta Polytechnica Hungarica, vol. 15, no. 4, pp. 27-44, 2018.
38. I. Uvalieva, Z. Garifullina, A. Utegenova, S. Toibayeva and B. Issin, "Development of intelligent system to support management decision-making in education", 2015 6th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), pp. 1-7, May 2015.
39. Taha H.A. Operations research. An introduction / H.A. Taha. – Pearson education limited, 2017. – 849 p.
40. Etminaniesfahani A, Ghanbarzadeh A, Marashi Z. Fibonacci indicator algorithm: A novel tool for complex optimization problems [J]. Engineering Applications of Artificial Intelligence, 2018, 74(SEP.):1-9.

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

__

Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)