ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Автоматизація складання розкладів у навчальних закладах за допомогою генетичних алгоритмів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим КОЗЯРИК
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр.ШІД-42
КОЗЯРИК Максим

Керівник: Олександр РАБОТА
д.т.н, професор

Рецензент:
_____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ

Завідувач кафедрою Штучного інтелекту
_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Козярик Максим
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Автоматизація складання розкладів у навчальних закладах за допомогою генетичних алгоритмів

керівник кваліфікаційної роботи д.т.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «25» 02.2025р. № 36

2. Строк подання кваліфікаційної роботи «23» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд сучасних технологій для розробки

5. Перелік графічного матеріалу: *презентація*

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	25.02-05.11.25	Виконав
2	Вивчення принципів роботи генетичних алгоритмів та їх адаптації до задачі формування розкладу	05.11-12.11.25	Виконав
3	Проєктування архітектури вебсистеми: визначення структур бази даних, ролей, логіки	13.11-19.11.25	Виконав
4	Розробка алгоритму генерації розкладу на основі генетичного підходу, тестування	20.11-25.11.25	Виконав
5	Реалізація вебінтерфейсу системи (Flask, HTML/CSS/JS, введення даних, вивід результатів)	27.11-03.12.25	Виконав
6	Розробка модуля візуалізації розкладу, редагування, збереження в SQLite	04.12-10.12.25	Виконав
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.25	Виконав
8	Розробка демонстраційних матеріалів	21.12-23.05.25	Виконав

Здобувач вищої освіти _____ Максим КОЗЯРИК

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ _Олександр РАБОТА

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 64 стор., 13 рис., 20 джерел.

Мета роботи: Розробка програмного засобу для автоматизованого складання розкладу занять у навчальному закладі з використанням генетичного алгоритму з урахуванням обмежень та оптимізаційних критеріїв.

Об'єкт дослідження: Процес формування розкладу в закладах освіти в умовах множинних обмежень (кабінети, викладачі, групи, тип занять тощо).

Предмет дослідження: Методи оптимізації, а саме — застосування генетичних алгоритмів у задачах автоматизованого планування навчального процесу.

Короткий зміст роботи: У дипломній роботі розроблено та реалізовано вебсистему для автоматизації процесу складання розкладу занять у навчальних закладах із використанням генетичних алгоритмів. Основна мета роботи — зменшення часових витрат адміністрації, оптимізація використання ресурсів (аудиторій, викладачів, груп) та дотримання множини обмежень під час формування розкладу. У процесі дослідження було проведено аналіз існуючих програмних рішень, розглянуто основи генетичних алгоритмів і методи їх адаптації до задачі розкладу. Спроєктовано архітектуру клієнт-серверної системи з вебінтерфейсом, реалізовано базу даних для збереження інформації про групи, викладачів, кабінети та предмети. Розроблена система забезпечує генерацію розкладу на основі заданих параметрів і обмежень, має зручний інтерфейс для редагування, перегляду й експорту. Експериментальне тестування підтвердило працездатність та ефективність підходу: зменшено кількість конфліктів, покращено розподіл занять, досягнуто високого покриття обмежень..

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ РОЗКЛАДУ, ГЕНЕТИЧНИЙ АЛГОРИТМ, РОЗКЛАД ЗАНЯТЬ, FLASK, PYTHON, SQLITE, ОСВІТНІ ТЕХНОЛОГІЇ, ОПТИМІЗАЦІЯ, ВЕБІНТЕРФЕЙС.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку цифрових технологій та інформатизації освітнього процесу виникає необхідність в автоматизації рутинних і трудомістких завдань, зокрема — складання розкладів занять у навчальних закладах. Складання розкладу — це складна багатофакторна задача, яка потребує врахування численних обмежень: наявність викладачів, доступність аудиторій, відповідність типів занять, уникнення конфліктів у групах тощо. Традиційне ручне формування розкладу часто вимагає значного часу, не гарантує оптимальності й ускладнює оперативне внесення змін. З розвитком методів штучного інтелекту, зокрема генетичних алгоритмів — одного з ефективних підходів до задачі оптимізації — з'являється можливість створення систем, здатних автоматично формувати коректні й зручні розклади з урахуванням численних факторів. Такі системи дозволяють зменшити навантаження на адміністрацію, мінімізувати конфлікти в розкладі та забезпечити масштабованість освітнього процесу.

Актуальність теми обумовлена зростаючою складністю та варіативністю освітніх програм, що вимагають швидкого, точного та адаптивного складання розкладів. Використання інтелектуальних алгоритмів у цьому процесі сприяє покращенню керованості навчального процесу та оптимізації використання ресурсів.

Метою даної кваліфікаційної роботи є розробка вебсистеми для автоматизованого складання розкладу занять у навчальному закладі з використанням генетичних алгоритмів.

Об'єктом дослідження є процес формування розкладу занять у навчальних закладах.

Предметом дослідження виступає інтелектуальна система, що реалізує автоматизоване складання розкладу з використанням генетичних алгоритмів та сучасних вебтехнологій.

Для досягнення мети було визначено такі завдання:

- проаналізувати наукові джерела щодо методів автоматизованого складання розкладів та можливостей застосування генетичних алгоритмів у системах планування;
- дослідити наявні програмні рішення для генерації розкладів у закладах освіти, визначити їхні переваги та недоліки;
- вивчити принципи побудови генетичних алгоритмів та адаптацію їх операторів (відбір, кросовер, мутація) до задачі розкладу;
- сформулювати функціональні та нефункціональні вимоги до системи автоматизованого складання розкладу;
- розробити архітектуру системи та реалізувати вебінтерфейс для введення навчальних даних, запуску алгоритму, редагування та збереження результатів;
- протестувати систему, оцінити ефективність згенерованих розкладів і якість покриття обмежень, зіставивши результати з ручним плануванням.

Методи дослідження: аналіз літературних джерел та існуючих рішень, моделювання архітектури інформаційної системи, застосування генетичних алгоритмів, реалізація вебінтерфейсу на основі Flask (Python), збереження даних у базі SQLite, тестування якості результатів.

Практичне значення розробки полягає у можливості застосування створеної системи в реальному освітньому середовищі для полегшення процесу планування навчального процесу, що дозволяє зекономити ресурси та підвищити ефективність управління.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У СФЕРІ СКЛАДАННЯ НАВЧАЛЬНИХ РОЗКЛАДІВ

1.1 Проблематика формування розкладів у навчальних закладах

Процес формування розкладу занять у навчальних закладах є однією з найскладніших організаційних задач, що потребує врахування численних факторів і обмежень. З технічної точки зору ця задача належить до класу NP-повних задач — таких, для яких не існує універсального алгоритму, здатного за розумний (поліноміальний) час знайти оптимальне рішення. Внаслідок цього, при кожному додаванні нового елемента (наприклад, ще однієї групи або нового викладача), складність розрахунку зростає експоненційно. Уявімо заклад із 4 навчальними групами, 6 викладачами, 12 предметами й 8 доступними аудиторіями, з яких деякі призначені для лекцій, інші — для лабораторних занять. Кожна група має власні особливості в навчальному плані, а викладачі — обмеження за часом. Якщо врахувати всі можливі комбінації пар, днів, аудиторій і викладачів, отримаємо мільйони потенційних варіантів. Обчислити вручну хоча б частину таких варіантів є нереальним.

Завдання не обмежується лише технічними параметрами — воно має соціальну, педагогічну та психологічну складову. Наприклад, перевантаження студентів кількома складними дисциплінами в один день або поспіль, або призначення пари о 8:00 для викладача, який викладає вечірні курси, може викликати дискомфорт та негативно вплинути на якість навчання. Саме тому задача розкладу є багатовимірною і багатокритеріальною, і її ефективне розв'язання можливе лише за умов використання адаптивних обчислювальних методів, зокрема алгоритмів оптимізації.

Основна складність полягає в необхідності врахування великої кількості обмежень:

- Наявність викладачів: викладач не може одночасно читати лекцію в кількох групах або перебувати в різних аудиторіях.

- Обмеження по кабінетах: заняття мають відповідати типу аудиторії (лекційна, лабораторна тощо) та не дублюватися в часі.
- Групова узгодженість: студенти однієї групи не повинні мати конфліктуючі пари.
- Часові переваги: деякі викладачі або групи мають обмеження за днями чи часом.
- Навчальне навантаження: потрібно рівномірно розподіляти предмети протягом тижня, уникати перевантаження.

У традиційних умовах складання розкладу здійснюється вручну — за допомогою Excel-таблиць, блокнотів, паперових карток або базових електронних шаблонів. У багатьох закладах досі зберігається практика, коли адміністрація або завідувачі кафедр розподіляють пари, виходячи з власного досвіду та загальних шаблонів, часто не маючи змоги враховувати всі можливі обмеження та залежності. При цьому будь-які зміни в навчальному плані, поява нових викладачів або зміни у складі груп вимагають повторного аналізу всього розкладу. Цей процес є вкрай трудомістким і вимагає значної кількості часу та людських ресурсів. Для великих освітніх закладів, які мають десятки навчальних груп, сотні пар на тиждень та десятки викладачів, формування розкладу може тривати тижнями. В результаті можливі часті помилки, накладки пар, перевантаження студентів і викладачів, а також недостатня ефективність використання ресурсів. Ще однією важливою проблемою є складність внесення змін до вже сформованого розкладу. Навіть незначна зміна — наприклад, заміна одного викладача через хворобу — може потребувати повного перегляду ланцюжка занять, оскільки кожна зміна тягне за собою потребу у врахуванні зайнятості аудиторій, інших викладачів, суміжних груп тощо. Без автоматизованих інструментів це часто перетворюється на «ручну кризу», коли навіть невелике коригування вимагає значного перерахунку всього тижневого розкладу.

Ручне планування розкладу в сучасних умовах не відповідає вимогам ефективного управління навчальним процесом і не здатне оперативно реагувати

на динамічні зміни в освітньому середовищі. Такий підхід є не лише застарілим з технологічної точки зору, а й створює численні організаційні труднощі, які мають системний характер. Також важливим є аспект прозорості та об’єктивності. Розклад, сформований вручну, значною мірою залежить від досвіду та інтуїції особи, яка його укладає, що створює ризик упередженості. Відсутність об'єктивних алгоритмічних критеріїв призводить до того, що одні групи або викладачі отримують переваги у вигляді зручного часу занять чи доступу до ресурсів, тоді як інші — відчутно втрачають у якості навчального процесу. Такий стан речей може породжувати незадоволення, конфлікти, а в деяких випадках — скарги до адміністрації навчального закладу.

Особливо критичним є питання раціонального розподілу навчального навантаження. Часто трапляються ситуації, коли у студентів концентруються всі найважчі дисципліни в один день, або викладачі мають значні перерви між парами, що призводить до неефективного використання часу та ресурсів. Ігнорування психофізіологічних особливостей учасників навчального процесу (перевтома, втрата концентрації, стрес) суттєво знижує якість освіти. Брак чітких критеріїв при ручному формуванні розкладу унеможливлює його повторення в однакових умовах, що ускладнює аналіз та аудит рішень. Це також не дає змоги впровадити єдині підходи до оцінки ефективності навчального планування, порівняння між роками або навчальними підрозділами. В умовах сучасного освітнього середовища вкрай необхідне впровадження систематизованого, прозорого та автоматизованого підходу до розкладу, заснованого на використанні алгоритмічних методів та об’єктивних критеріїв. Такий підхід не лише підвищує ефективність управління освітнім процесом, але й створює умови для гнучкого, адаптивного, рівномірного та справедливого розподілу ресурсів.

Брак чітких критеріїв при ручному формуванні розкладу унеможливлює його повторення в однакових умовах, що робить неможливим аудит чи аналіз ефективності рішень. Розклади, складені вручну, зазвичай не мають фіксованої логіки або структури, через що будь-який зовнішній або внутрішній контроль

ускладнюється. Це особливо важливо в умовах впровадження систем управління якістю освіти, де потрібні прозорі й відтворювані процедури планування. Відсутність централізованого алгоритмічного підходу створює хаос в управлінні розкладом. Кожен укладач може використовувати власну методику, що призводить до невідповідностей навіть у межах одного навчального закладу. В умовах частих змін навчальних планів, оновлень програм, появи нових дисциплін або викладачів, система, що не має єдиної логіки генерації, втрачає здатність до швидкої адаптації. При кожному оновленні навчального навантаження адміністрація змушена виконувати значну частину роботи з нуля.

Актуальною стає потреба у систематичному, автоматизованому підході до формування розкладу, який не лише враховує всі задані обмеження, а й забезпечує прозорість, рівномірність, масштабованість і адаптивність системи. Такий підхід базується на формалізованих алгоритмах, що забезпечують стабільну якість розкладу при будь-яких вхідних умовах. Автоматизація дозволяє не лише підвищити ефективність управління навчальним процесом, а й створити умови для гнучкого перерахунку розкладу, швидкого тестування альтернативних варіантів, мінімізації конфліктів і зменшення ризику людських помилок. У підсумку це сприяє стабільності, надійності та об’єктивності в організації освітнього процесу навіть у динамічному середовищі.

1.2 Огляд існуючих програм для автоматичного складання розкладу (FET, Asc Timetables, UniTime)

У відповідь на зростаючу потребу в автоматизованому формуванні навчальних розкладів було розроблено низку програмних продуктів, які частково або повністю вирішують задачу планування занять у навчальних закладах. Вони розрізняються за рівнем складності, функціональними можливостями, способом реалізації алгоритмів, підтримкою локалізації, вимогами до апаратного забезпечення та інтеграційними можливостями. Найбільш відомими серед таких рішень є FET, Asc Timetables та UniTime. Кожна з них має власну цільову аудиторію: FET — для невеликих навчальних закладів і технічно підготовлених

користувачів, Asc Timetables — для широкого кола середніх і великих закладів з потребою у зручному інтерфейсі, UniTime — для комплексних університетських структур із потребами глибокої автоматизації. Вибір відповідного ПЗ залежить не лише від кількості груп, викладачів та аудиторій, а й від організаційної структури закладу, його IT-інфраструктури, бюджету та кадрового потенціалу. Важливим фактором є підтримка оновлень, активність спільноти розробників або користувачів, документація, доступність навчальних матеріалів. Також варто враховувати можливість кастомізації, розширення за допомогою плагінів або API, і навіть наявність функцій резервного копіювання, друку, експорту, а також синхронізації з мобільними пристроями або календарями. Тому комплексний огляд таких систем є важливим етапом перед проєктуванням власного рішення.

FET (Free Timetabling Software) (рис. 1.1) — це потужна безкоштовна система з відкритим вихідним кодом, призначена для автоматичного формування навчальних розкладів. Програма базується на обмеженнях (constraints), які користувач самостійно задає у відповідності до особливостей навчального процесу. До таких обмежень належать наявність викладачів, обмеження по годинах і днях, тип занять, специфіка приміщень, мінімальні або максимальні перерви між парами, обмеження по днях тощо.

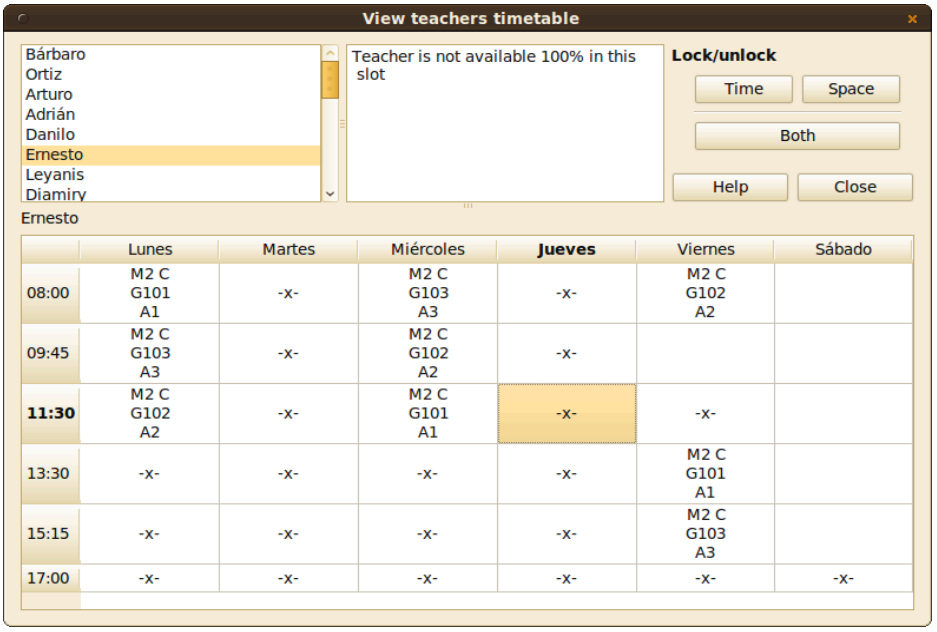


Рисунок 1.1 – Система FET (Free Timetabling Software)

FET підтримує створення складних розкладів навіть для великих освітніх установ із десятками груп, сотнями викладачів та великою кількістю приміщень. Користувач має змогу задавати вагу обмежень (жорсткі та м’які), що дає змогу досягти гнучкого балансу між необхідністю та бажаністю певного параметра. Серед функціоналу FET — генерація розкладу з кількох початкових точок (рандомізація), ведення окремих розкладів для підгруп, підтримка предметів з кількома викладачами, а також експорт у зручні формати (HTML для публікації, XML для інтеграції, CSV для аналізу в Excel). Програма також дозволяє зберігати шаблони і повторно використовувати раніше задані конфігурації. Проте, попри свою функціональність, FET має обмеження. Її інтерфейс є мінімалістичним і не завжди інтуїтивним, що ускладнює використання для користувачів без технічної підготовки. Відсутня інтеграція з сучасними інформаційними системами ЗВО, а також не підтримується повноцінне редагування розкладу після генерації — для зміни потрібно виконувати перегенерацію. Також у системі відсутній вебінтерфейс, що унеможливлює колективну або віддалену роботу з розкладом без зовнішніх інструментів. FET активно використовується в багатьох країнах завдяки своїй відкритості, гнучкості та активній спільноті розробників. Вона особливо підходить для навчальних закладів, які мають обмежений бюджет і можуть адаптуватися до роботи з технічно орієнтованим інтерфейсом.

Asc Timetables (рис. 1.2) — одна з найпопулярніших комерційних програм для автоматичного складання навчальних розкладів, яка вже багато років активно використовується в навчальних закладах різних рівнів. Програма має зручний графічний інтерфейс, що дозволяє швидко орієнтуватися навіть недосвідченим користувачам. Підтримується перетягування (drag-and-drop), ручне коригування розкладу, автоматичне заповнення з урахуванням заданих обмежень, кольорове кодування предметів, груп і викладачів. Серед сильних сторін Asc Timetables — можливість створення шаблонів занять, генерація кількох альтернативних варіантів розкладу, налаштування пріоритетів для занять, збереження історії змін, підтримка багатомовного інтерфейсу. Крім того, програмне забезпечення

підтримує імпорт і експорт даних у різних форматах, включаючи PDF, HTML та Excel, що значно спрощує підготовку до друку та розповсюдження серед студентів та викладачів.

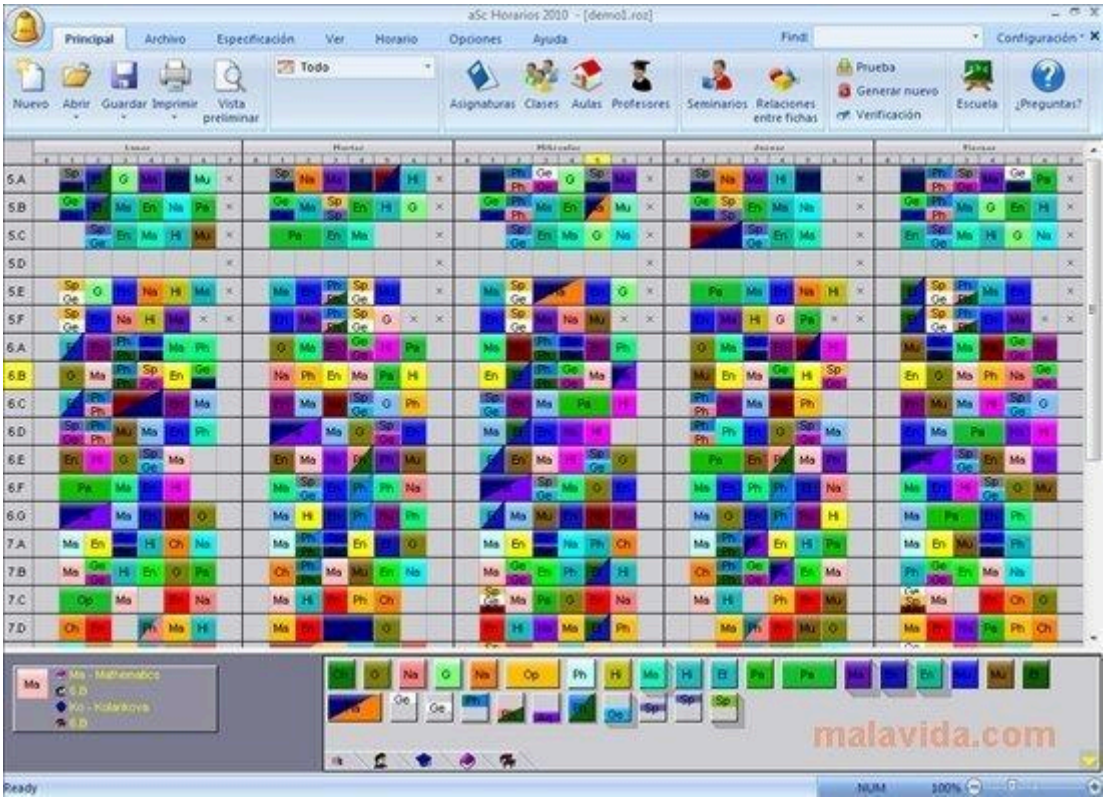


Рисунок 1.2 – Система Asc Timetables

Asc також інтегрується з іншими адміністративними системами закладів освіти (електронний журнал, система управління студентами), а також підтримує хмарні рішення для синхронізації та збереження даних. Програма адаптована до вимог як загальноосвітніх шкіл, так і вищих навчальних закладів, що робить її універсальним рішенням у сфері планування навчального процесу. Недоліками Asc Timetables є її закритий вихідний код, що унеможливлює кастомізацію для глибоко спеціалізованих сценаріїв без участі розробника, а також обмеження безкоштовної версії, яка не включає всі доступні функції. Крім того, у випадку великих навчальних закладів зі складною структурою, система потребує значних зусиль для налаштування і тестування розкладів, що може бути не зовсім ефективно без технічної підтримки.

UniTime (рис. 1.3) — університетська система планування, що створена як комплексне рішення для великих вищих навчальних закладів, які мають потребу в автоматизації не лише розкладів занять, а й управління ресурсами, екзаменами, аудиторіями, а також адміністративним навантаженням. Система є результатом багаторічних досліджень і розробок в Університеті Індіани (США), де вона активно застосовується в реальному навчальному процесі. UniTime має модульну архітектуру, що охоплює чотири основні напрями: планування занять, екзаменів, приміщень та обслуговування студентських запитів.

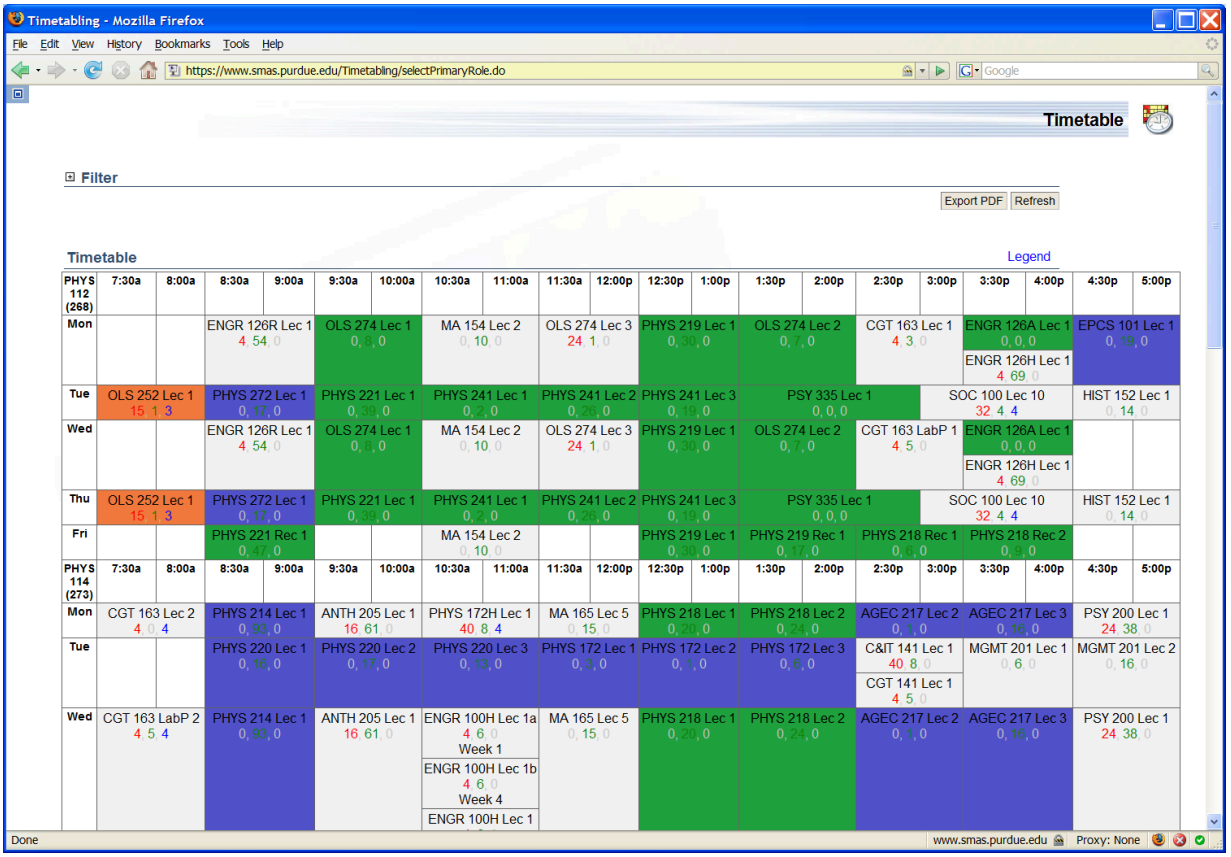


Рисунок 1.3 – Система UniTime

UniTime є веборієнтованою системою, яка використовує сучасні технології Java EE, Hibernate, Spring та базується на PostgreSQL або інших СУБД. Вона передбачає підтримку розмежування ролей користувачів, роботу в кількох часових зонах, імпорт навчального плану, автоматичне виявлення конфліктів, балансування навантаження між викладачами та групами, резервування аудиторій тощо. Крім того, система дозволяє планувати декілька сценаріїв розкладу для вибору

найкращого варіанту. Особливістю UniTime є її здатність інтегруватися з іншими інформаційними системами університету (наприклад, системами управління навчальним процесом, електронними журналами, системами запису студентів), що забезпечує єдину інформаційну екосистему. UniTime також підтримує API для розширення функціональності та налаштування звітності. Серед недоліків — висока складність впровадження, що передбачає налаштування серверного середовища, складну конфігурацію бази даних, необхідність навчання персоналу та підтримки адміністраторів. Крім того, через свою масштабність UniTime не завжди є доцільним вибором для малих або середніх закладів освіти, де потреби в глибокій автоматизації є обмеженими. Проте для великих університетів вона відкриває потужні можливості централізованого управління навчальним процесом, забезпечуючи гнучкість, масштабованість і ефективність планування.

Порівняльна характеристика:

- FET — безкоштовна, розширювана, але неінтуїтивна.
- Asc Timetables — проста у використанні, платна, гарна візуалізація.
- UniTime — масштабована, складна у розгортанні, гнучка та потужна.

Усі ці системи демонструють різні підходи до реалізації задачі планування розкладу. Досвід використання таких рішень свідчить про реальну потребу у створенні більш гнучких, адаптивних та інтелектуальних систем, які поєднують зручність у користуванні, підтримку складних обмежень та здатність швидко адаптуватися до змін. У рамках даної роботи було детально проаналізовано програмні рішення FET, Asc Timetables та UniTime як приклади існуючої практики автоматизації процесу складання навчального розкладу. Аналіз охоплював технічні характеристики, функціональні можливості, обмеження, досвід користувачів та потенціал масштабування. Особливу увагу було приділено алгоритмічній складовій, підтримці складних обмежень, можливості інтеграції з іншими системами, кастомізації та адаптивності до різних освітніх моделей.

Результати аналізу дозволили виділити як сильні сторони (гнучкість, наявність візуального інтерфейсу, підтримка імпорту/експорту даних,

модульність), так і недоліки (відсутність відкритого коду, складність у налаштуванні, нестача адаптивного інтелектуального компонента) кожного з рішень. Ці висновки стали основою для формування вимог до власної системи, яка в даній роботі розробляється на основі генетичного алгоритму. Такий підхід забезпечує адаптивність, масштабованість і здатність до швидкої генерації розкладу з урахуванням реальних обмежень та пріоритетів навчального процесу.

1.3 Основи генетичних алгоритмів та їх застосування у задачах планування

Генетичні алгоритми (ГА) (рис. 1.4) — це клас еволюційних обчислювальних методів, які наслідують принципи біологічної еволюції: спадковість, варіацію та природний добір. Вони моделюють процес адаптації популяцій живих організмів до змін у навколишньому середовищі та переносять ці принципи в математичну і програмну площину для розв’язання задач оптимізації. На відміну від класичних алгоритмів, що базуються на строгих правилах і логіці, генетичні алгоритми здатні знаходити наближені або субоптимальні рішення в умовах високої невизначеності, багатовимірності та великої кількості комбінацій. ГА належать до метаевристичних алгоритмів, які використовують стохастичні механізми (випадковість) для обходу простору рішень. Вони не гарантують знаходження глобального оптимуму, але завдяки багаторазовому «еволюційному» пошуку здатні наближатися до нього з високою ймовірністю. Основна ідея полягає в тому, щоб підтримувати і розвивати множину рішень (популяцію), поступово відкидаючи найгірші та комбінуючи найкращі, з метою формування ще кращих особин у наступних поколіннях.

ГА особливо ефективні в задачах, де традиційні методи не справляються через комбінаційну складність, відсутність лінійних залежностей або велику кількість змінних. Саме тому вони широко застосовуються у сферах, де потрібно враховувати десятки, а іноді й сотні обмежень та умов: логістика, розподіл ресурсів, енергетика, телекомунікації, біоінформатика і, зокрема, — у плануванні навчального розкладу в закладах освіти.

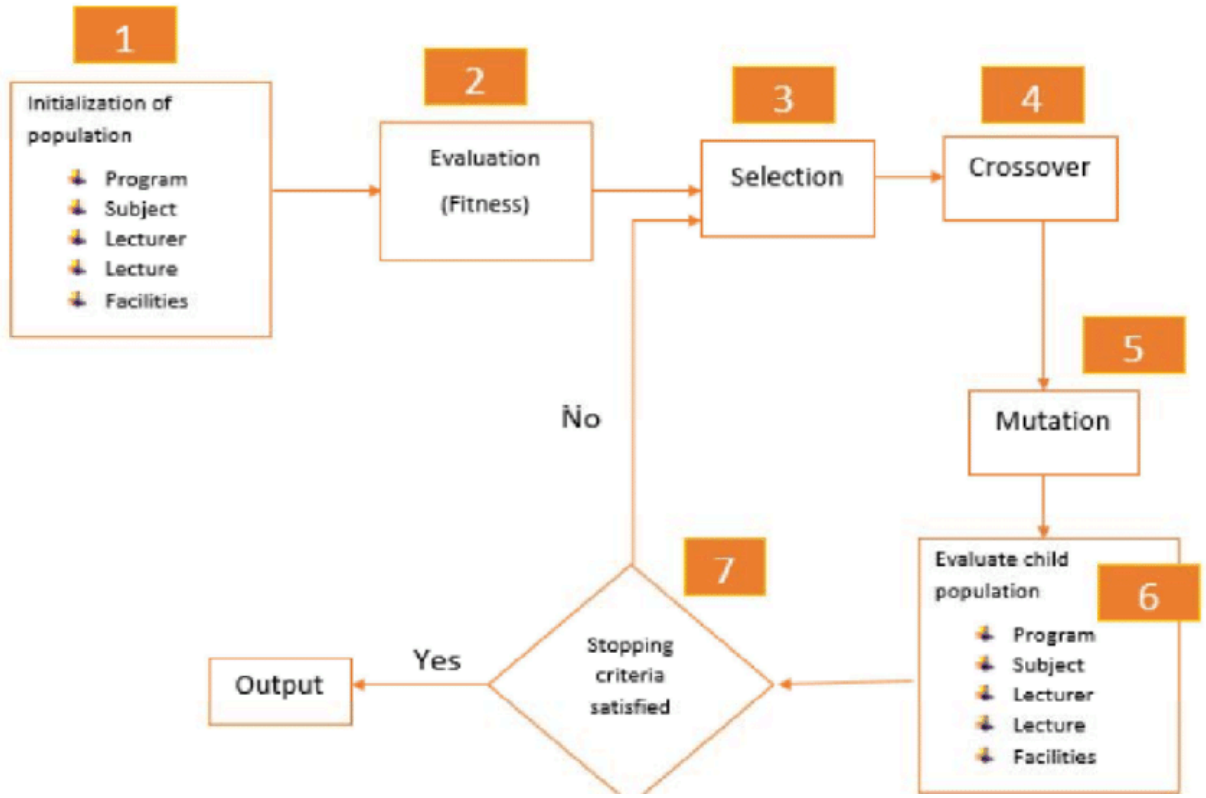


Рисунок 1.4 – Приклад генетичного алгоритму

У таких задачах важко знайти оптимальне рішення за прийнятний час, і саме тут ГА виявляються незамінними. Їхня здатність адаптуватися до складних структур обмежень, швидко генерувати нові варіанти, виявляти конфлікти й мінімізувати їх робить їх ідеальним інструментом для автоматизації складання розкладу. Вони дозволяють не просто автоматизувати процес, а й зробити його гнучким, адаптивним і наближеним до реальних потреб усіх учасників навчального процесу. Генетичний алгоритм працює з множиною потенційних рішень, які називаються популяцією. Популяція складається з багатьох індивідів, кожен з яких репрезентує певне розв'язання задачі — у контексті розкладу це може бути певний набір пар, прив'язаний до конкретних днів, годин, викладачів та аудиторій. Кожне рішення (індивід) кодується у вигляді хромосоми — структури, що містить повну інформацію про конкретний варіант розкладу. Як правило, хромосома представлена у вигляді масиву, де кожен ген відповідає окремій одиниці планування (наприклад, конкретному заняттю), а його значення — це параметри, пов'язані з часом, місцем проведення, типом заняття тощо. Такий

підхід дозволяє чітко формалізувати задачу та забезпечити гнучкість при внесенні змін. Генетичний алгоритм імітує біологічні операції, що відбуваються в природному доборі, і на основі цього виконує пошук оптимального розкладу. Основними операціями є:

- Селекція — вибір найкращих особин (розкладів) на основі функції пристосованості (fitness function), яка оцінює якість розкладу з погляду кількості порушених обмежень.
- Кросовер (схрещування) — об’єднання частин двох батьківських рішень для створення нових комбінацій.
- Мутація — випадкова зміна частини розкладу для підвищення різноманітності популяції та уникнення локального оптимуму.

Процес повторюється в циклі поколінь доти, доки не буде знайдено розклад, що задовольняє задані критерії якості або не буде досягнуто наперед визначеної межі — кількості поколінь чи допустимого часу виконання. На кожному етапі відбираються найбільш придатні розв’язання, які далі схрещуються і мутують, створюючи нову генерацію, що, як правило, має кращі характеристики. Завдяки цьому генетичний алгоритм поступово еволюціонує в напрямі оптимального або близького до оптимального рішення. У деяких випадках застосовується елітарність — стратегія, яка дозволяє зберегти найкращі рішення з попереднього покоління без змін, що прискорює збіжність алгоритму.

Застосування генетичних алгоритмів до задачі складання навчального розкладу є обґрунтованим завдяки таким перевагам:

- здатність працювати з великою кількістю змінних та обмежень;
- гнучкість при моделюванні специфічних правил (наприклад, «не більше однієї лабораторної на день», «перевага ранкових пар»);
- швидке знаходження прийнятного (але не обов’язково ідеального) рішення в умовах обмеженого часу;
- можливість адаптації до змін вхідних даних без необхідності повного перебудування всієї системи.

У контексті освітніх систем генетичні алгоритми дозволяють створювати динамічні, адаптивні розклади, які враховують численні аспекти функціонування навчального процесу. По-перше, це ресурсні обмеження — наявність вільних аудиторій, доступність викладачів, кількість годин, відведених на певний предмет, тип заняття (лекція, семінар, лабораторна) та сумісність із обладнанням приміщення. По-друге, це педагогічні фактори, зокрема рівномірний розподіл складних дисциплін, уникнення накопичення інтенсивних занять упродовж одного дня або тижня, забезпечення ефективної зміни тематики в межах навчального дня. Психологічні аспекти передбачають уникнення ситуацій, коли студентам або викладачам доводиться починати заняття дуже рано або завершувати надто пізно, мати великі перерви між парами чи повертатися до університету після тривалих «вікон». Важливо також уникати перенавантаження, коли студенти мають, наприклад, чотири або п’ять пар підряд без перерв. Додатково враховуються індивідуальні побажання — як викладачів (зручні дні, вікна для наукової діяльності, відрядження), так і студентів (збалансованість занять за складністю, суміщення з практикою або роботою). Це дозволяє створювати персоналізовані розклади, які відповідають не лише організаційним вимогам, а й забезпечують комфортне освітнє середовище, сприятливе для засвоєння знань та мінімізації стресових ситуацій.

Наприклад, розклад, сформований на основі ГА, може мінімізувати конфлікти, уникати повторів, забезпечувати рівномірний розподіл дисциплін протягом тижня, уникати ситуацій, коли студентам доводиться відвідувати лише одну пару на день або мати великі «вікна» між заняттями. Завдяки випадковому характеру кросоверу та мутації, система досліджує широкий спектр можливих рішень, що дозволяє знаходити нестандартні, але ефективні варіанти. Важливою складовою ГА є правильно визначена функція пристосованості (fitness function). Саме вона виступає головним критерієм для оцінки кожного потенційного рішення та орієнтирує весь процес еволюційного пошуку. У задачах планування розкладу ця функція може включати низку факторів: кількість конфліктів між

парами (наприклад, один викладач у двох аудиторіях одночасно), порушення обмежень щодо ресурсів (відсутність відповідної аудиторії), перевищення максимально дозволеного навантаження на студентів або викладачів, тривалість вікон між парами, наявність занять у небажані години тощо.

Функція пристосованості може враховувати більш тонкі критерії, як-от педагогічні пріоритети (розміщення теоретичних дисциплін у першій половині дня), побажання викладачів (непрацездатність у певні дні), специфіку навчального плану (послідовність предметів) або пріоритети ресурсного використання. Також доцільно застосовувати систему ваг — кожне порушення має різну «вартість», що дозволяє алгоритму фокусуватися на найбільш критичних обмеженнях. Добре налаштована функція пристосованості забезпечує ефективну навігацію в просторі рішень і запобігає ситуаціям, коли алгоритм «застрягає» в локальних мінімумах, не в змозі досягти глобального оптимального варіанта. У поєднанні з іншими операціями — селекцією, кросовером, мутацією — вона забезпечує збалансовану еволюцію популяції та прогресивне покращення розкладу від покоління до покоління.

У результаті, генетичні алгоритми демонструють високу ефективність у практичних задачах планування, де класичні алгоритмічні підходи, як-от жадібні алгоритми, жорсткі обчислювальні правила або табу-пошук, часто не здатні забезпечити задовільний результат у розумний час. З огляду на багатофакторність задачі розкладу, ГА дають змогу створювати персоналізовані, конфліктно-стійкі й адаптивні розклади, що враховують як жорсткі обмеження, так і гнучкі побажання, тим самим виступаючи ідеальним інструментом для реалізації системи автоматизованого формування розкладу занять, розробленої в межах даної роботи.

Висновки до розділу 1

У першому розділі дипломної роботи було проведено комплексний аналіз проблематики складання навчальних розкладів у закладах освіти та розглянуто

існуючі підходи до їх автоматизації. З’ясовано, що задача формування розкладу є складною комбінаційною проблемою з великою кількістю обмежень, які мають як жорсткий, так і м’який характер. У ручному режимі така задача потребує значних ресурсів і часу, не дозволяє оперативно реагувати на зміни та створює ризики помилок і конфліктів.

Огляд наявних програмних рішень, таких як FET, Asc Timetables та UniTime, продемонстрував широкий спектр можливостей, проте жодна з них не є універсальним рішенням для всіх типів закладів. FET відзначається відкритим кодом і гнучкістю, але вимагає технічної підготовки. Asc Timetables є простим у користуванні, але має обмеження в адаптації. UniTime — потужна, проте складна у впровадженні система, яка підходить для великих університетів. Виявлено потребу у створенні більш адаптивного, інтелектуального рішення з високою гнучкістю й можливістю урахування множини змінних і обмежень.

Окрему увагу було приділено аналізу генетичних алгоритмів як ефективного підходу до задач розкладу. Розглянуто ключові етапи їх роботи: ініціалізацію популяції, селекцію, кросовер, мутацію, оцінку функції пристосованості та еволюцію рішень у часі. Генетичні алгоритми показали високу ефективність у задачах, де традиційні методи не справляються або вимагають складної ручної конфігурації. Їх використання дозволяє створити розклади, що відповідають великій кількості взаємопов’язаних вимог, уникають конфліктів і адаптуються до змін у навчальному процесі.

Результати аналізу підтверджують актуальність застосування генетичних алгоритмів у сфері автоматизації навчального планування. Цей підхід забезпечує гнучкість, масштабованість, адаптивність та здатність до швидкого реагування на зміну вхідних параметрів, що є критично важливим для ефективного управління сучасним освітнім процесом..

2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ СКЛАДАННЯ РОЗКЛАДІВ

2.1 Постановка задачі та вимоги до системи

Задача автоматизованого складання навчального розкладу вимагає створення комплексної програмної системи, яка поєднує в собі інструменти збору, обробки, оптимізації та візуалізації навчальних даних із гнучким урахуванням специфічних вимог закладу освіти. Така система повинна моделювати складну мережу взаємопов'язаних сутностей — групи, викладачі, дисципліни, аудиторії, типи занять, розподіл годин тощо — та забезпечувати максимально повне покриття як адміністративних, так і педагогічних вимог. У межах даного проєкту основною метою є побудова адаптивної веборієнтованої інформаційної системи для генерації навчального розкладу. Система базується на використанні генетичного алгоритму — одного з найбільш гнучких методів оптимізації, здатного враховувати широкий спектр обмежень та пріоритетів, що постійно змінюються. Обрана технологічна база (мова програмування Python, фреймворк Flask, СУБД SQLite) дозволяє забезпечити масштабованість, легкість у розгортанні, інтеграцію з іншими сервісами та зручність супроводу.

Проєктна модель також передбачає багаторівневу рольову систему доступу (адміністратор, редактор, викладач), що забезпечує диференційований доступ до функціоналу системи відповідно до повноважень. Адміністратор має можливість повного управління даними та розкладом, редактор — вносити зміни в параметри генерації, а викладач — переглядати результати, залишати побажання або обмеження щодо розкладу. Також реалізовано функцію повторної генерації розкладу без втрати попередніх налаштувань, що дозволяє швидко адаптуватися до змін у навчальному процесі. Вбудовані механізми валідації забезпечують перевірку коректності введених даних, а система журналювання зберігає повну історію змін — з іменами користувачів, датою та змістом правок. Особливу увагу приділено забезпеченню простоти використання для кінцевого користувача.

Сучасний інтерфейс з адаптивною верстkою підтримує роботу з різних пристроїв, має інтуїтивно зрозумілу навігацію, кольорове виділення конфліктів у розкладі та підтримку drag-and-drop елементів. Реалізовано покрокові підказки, що дозволяють користувачам без технічних знань ефективно взаємодіяти із системою.

Сформульована задача охоплює не лише створення ефективного технічного інструмента, а й повномасштабну організаційно-логічну трансформацію всього процесу планування занять у навчальному закладі. Запропонована система виступає як цифровий координатор між адміністрацією, викладачами та студентами, формуючи єдину інформаційну екосистему. Вона орієнтована на досягнення високої швидкості обробки даних, прозорості дій користувачів, об'єктивності формування розкладу на основі складних обмежень та максимальної зручності щоденного використання через сучасний, адаптивний інтерфейс. Система покликана не тільки автоматизувати рутинні операції, а й підвищити загальну ефективність управління навчальним процесом, забезпечивши узгодженість, гнучкість та здатність до оперативного реагування на зміни.

Система має забезпечувати виконання таких функціональних вимог:

- введення вихідних даних: групи, предмети, викладачі, аудиторії, типи занять;
- зв’язування викладачів із предметами та облік їхнього навантаження;
- запуск алгоритму генерації розкладу на основі заданих обмежень;
- візуалізація розкладу у зручному для користувача вигляді (по групах, викладачах, кабінетах);
- редагування та повторна генерація розкладу при зміні параметрів;
- збереження результатів у базу даних для подальшого використання.

До нефункціональних вимог відносяться:

- стабільна робота вебінтерфейсу при великій кількості даних;
- захищеність користувацьких сесій та авторизація адміністраторів;
- адаптивний інтерфейс для використання на різних пристроях;
- висока швидкість генерації навіть при складних умовах планування.

Постановка задачі передбачає формалізацію усіх ключових ресурсів та сутностей, які беруть участь у навчальному процесі. Це, зокрема, аудиторії — з чітким поділом на лекційні, лабораторні, комп'ютерні класи та спеціалізовані приміщення; викладачі — із зазначенням їх кваліфікації, дисциплін, навантаження та обмежень щодо часу; навчальні групи — з прив'язкою до спеціальностей, кількості студентів, типів предметів та інтенсивності навчання; предмети — зі структурованим поділом на типи занять (лекції, практичні, лабораторні) та відповідним розрахунком годин по семестрах. Важливо визначити детальний набір обмежень. Жорсткі обмеження включають, наприклад, заборону дублювання занять у одній аудиторії в той самий час, неможливість призначити викладача одночасно на кілька пар, перевищення граничного навантаження, недоступність ресурсів у певні години. М'які обмеження — це побажання викладачів щодо часу проведення занять, уникнення вікон між парами, баланс навантаження впродовж тижня, урахування зовнішніх факторів (конференції, відрядження тощо). Загальна мета полягає в побудові високоефективної адаптивної моделі, здатної оперативно реагувати на зміни у навчальному процесі, забезпечувати точну генерацію розкладу в межах заданих обмежень і відображати результат у формі, зручній для аналізу, перевірки та редагування. Система має забезпечувати повну автоматизацію ключових етапів планування — від введення вхідних даних до перегляду, експорту та збереження результатів. Важливо, щоб вона мала логічно розділену архітектуру (frontend/backend), масштабувалася як горизонтально (база користувачів), так і функціонально (додавання нових типів занять, обмежень, викладацьких форматів).

Вимога масштабованості передбачає здатність системи працювати як у рамках невеликого навчального закладу (коледж, приватна школа), так і в межах великого університету з кількома факультетами. Керованість включає гнучку систему ролей: адміністратор, редактор, викладач, а також контроль над параметрами доступу до функціоналу. Орієнтація на освітні потреби виражається в підтримці широкого спектра сценаріїв: від класичних денних програм до

модульного навчання, змішаного формату, подвійних спеціальностей, факультативів. Завдяки чіткому формулюванню завдання, глибокій формалізації обмежень та використанню адаптивного генетичного алгоритму, розроблена система має потенціал стати ключовим елементом цифрової трансформації процесу управління навчальним процесом. Вона здатна значно підвищити ефективність роботи адміністрації, знизити навантаження на працівників, забезпечити оперативне реагування на зміни в навчальному середовищі. Крім автоматизованого створення розкладів, система забезпечує прозорість дій усіх учасників, сприяє формуванню єдиного інформаційного простору між керівництвом, викладачами та студентами. Гнучкість, масштабованість та інтуїтивний інтерфейс роблять її універсальним рішенням, яке може бути адаптоване під різні моделі навчання та типи закладів — від шкіл до вищих навчальних закладів. У результаті підвищується організованість, знижується кількість конфліктних ситуацій у розкладі, а загальне враження студентів та викладачів від взаємодії з навчальним процесом значно покращується.

2.2 Архітектура системи: модулі, структура бази даних, інтерфейс

Архітектура системи автоматизації складання розкладу спроектована за принципом багаторівневої клієнт-серверної моделі з чітким розмежуванням відповідальності між її складовими. Основна концепція — це модульність, масштабованість та незалежність кожного компонента, що значно спрощує супровід, розширення й модернізацію системи. Структура системи поділяється на чотири ключові рівні: інтерфейс користувача (frontend), прикладний програмний інтерфейс (API), обчислювальний модуль (генетичний алгоритм) і рівень збереження даних (база даних).

Кожен із цих рівнів виконує окрему функцію: інтерфейс забезпечує взаємодію з користувачем, API слугує для обробки запитів і передачі даних, обчислювальний модуль виконує алгоритмічну обробку та оптимізацію, а база даних відповідає за зберігання та структуровану організацію інформації. Такий

розподіл дозволяє забезпечити паралельну розробку й тестування різних частин системи без порушення загальної логіки, а також адаптувати архітектуру під специфіку конкретного навчального закладу, обсяг даних чи інфраструктурні обмеження. Система реалізована з урахуванням гнучкої взаємодії між компонентами через REST API та розділення логіки клієнтської та серверної частин, що дозволяє легко масштабувати застосунок і впроваджувати додаткові функції без зміни існуючих модулів. Архітектурний підхід дозволяє інтегрувати як локальні, так і хмарні інфраструктури, забезпечуючи продуктивність і стабільність при різному навантаженні.

Інтерфейс користувача (рис. 2.1) є найбільш наближеним до кінцевого користувача елементом архітектури. Його функція полягає в забезпеченні зручного середовища для введення та редагування даних, запуску алгоритму генерації, а також перегляду сформованого розкладу. Через вебінтерфейс користувач (адміністратор, викладач або диспетчер) може пройти автентифікацію, додати нові дисципліни, призначити викладачів, створити групи, обрати доступні аудиторії тощо. Інтерфейс також дає змогу запускати процес генерації розкладу з урахуванням усіх наявних параметрів і переглядати результати у вигляді структурованих таблиць або діаграм. Усі ці запити, що надходять з інтерфейсу, спрямовуються до внутрішнього API, реалізованого у середовищі Flask. Flask API виступає посередником між фронтендом і обчислювальними модулями: він приймає запити від користувача, виконує перевірку даних, взаємодіє з базою даних і передає дані до модуля генетичного алгоритму. Завдяки чітко структурованим маршрутам і обробникам запитів (endpoints), API забезпечує узгоджене функціонування всієї системи, підтримує REST-підхід, а також дозволяє легко інтегрувати додаткові сервіси у майбутньому. Саме цей рівень відповідає за стабільну роботу додатку, логіку обробки інформації та повернення результатів користувачу.

Центральною частиною архітектури є модуль генетичного алгоритму. Він виконує основні обчислення, починаючи з ініціалізації популяції можливих

розкладів, далі — відбір найкращих представників, виконання кросоверу та мутації, обчислення функції пристосованості та створення нового покоління.

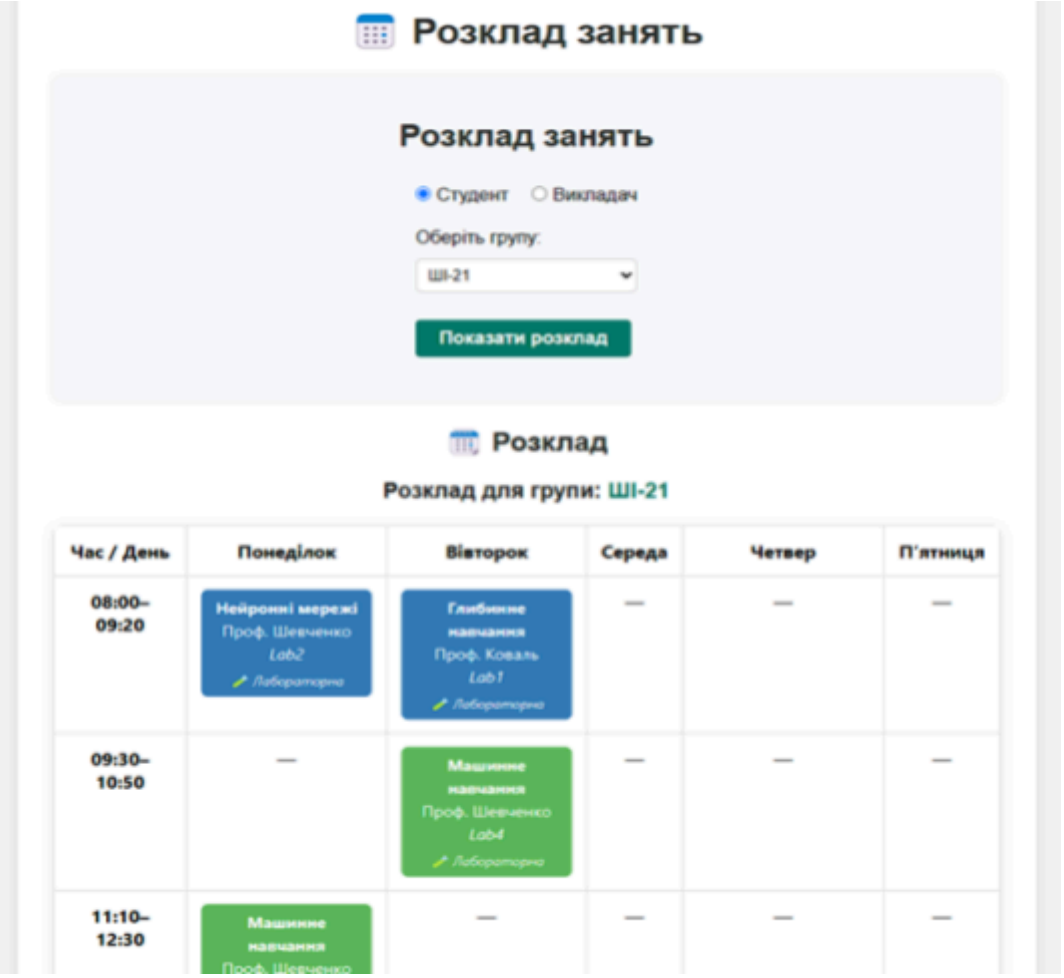


Рисунок 2.1 – Приклад інтерфейсу користувача

Саме тут закладено логіку оптимізації — враховуються усі задані обмеження (ресурсні, часові, педагогічні), а результат оцінюється за кількістю конфліктів і ступенем відповідності бажаним параметрам.

База даних системи побудована на основі реляційної моделі, яка забезпечує логічне структурування інформації, простоту реалізації запитів та гарантію цілісності даних. В якості реалізації обрано SQLite — легковагову, вбудовану СУБД, яка не вимагає окремого серверного середовища та забезпечує високу продуктивність для задач середнього масштабу. Такий підхід особливо доцільний для локального або серверного використання в навчальних закладах, де необхідна

стабільність і незалежність від зовнішніх сервісів. Структура бази даних (рис. 2.2) включає ключові сутності освітнього процесу: Teacher — дані про викладачів;

Group — академічні групи з прив'язкою до спеціальностей; Subject — навчальні дисципліни з позначенням типу заняття (лекція, лабораторна); Classroom — аудиторії з зазначенням функціонального призначення; Lesson — основна таблиця, яка поєднує викладача, предмет, аудиторію, групу, час та дату заняття. Додатково реалізовано таблицю ScheduleRule, яка описує навчальне навантаження у вигляді правил — скільки годин на місяць передбачено для кожного предмета в межах кожної групи. Усі зв'язки реалізовані через зовнішні ключі, що забезпечує логічну цілісність і дозволяє використовувати складні запити з приєднанням кількох таблиць.

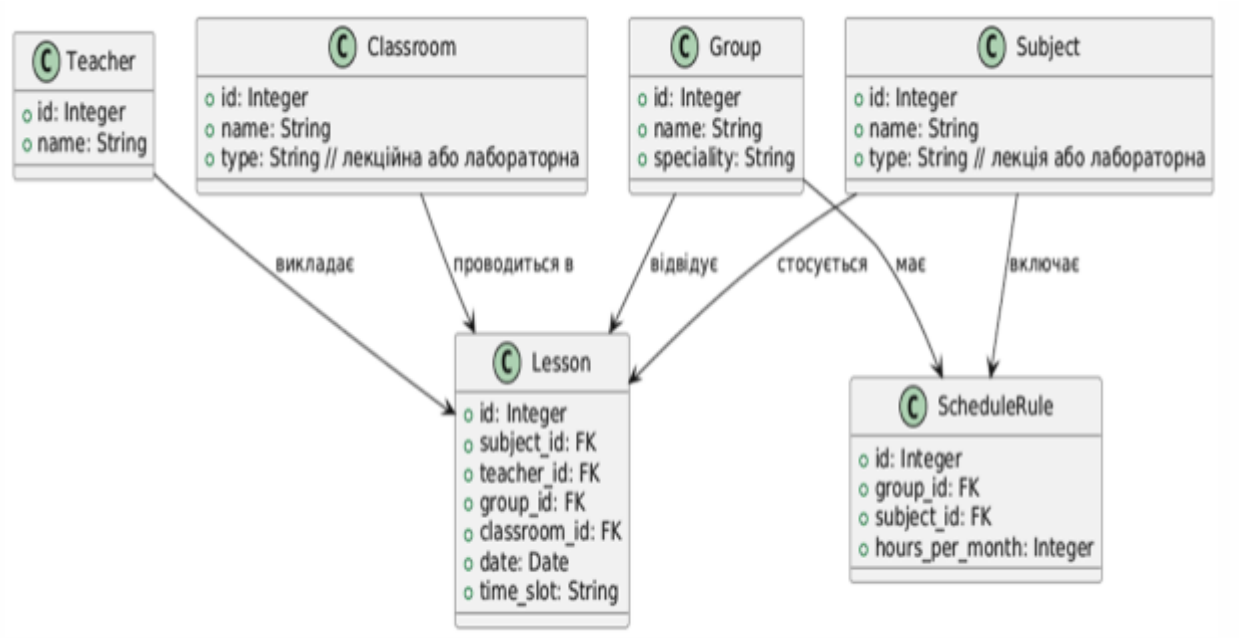


Рисунок 2.2 – Структура бази даних

З боку користувача взаємодія із системою реалізована через зручний вебінтерфейс, побудований з використанням HTML, CSS (зокрема фреймворку Bootstrap) та JavaScript. Інтерфейс забезпечує візуалізацію всіх основних операцій: від введення вихідних даних до перегляду готового розкладу. Адміністратор або викладач може працювати з інформацією у табличній формі, редагувати окремі записи, запускати генерацію нового розкладу, переглядати конфлікти й коментарі

системи щодо розміщення занять. Інтерфейс також підтримує адаптивну верстку, що дозволяє зручно працювати як на настільних комп’ютерах, так і на мобільних пристроях, не втрачаючи при цьому функціональності або читабельності. Такий підхід робить систему доступною для щоденного використання у будь-якому навчальному середовищі.

Загалом архітектура системи (рис. 2.3) дозволяє організувати повноцінний і гнучкий цикл автоматизації складання навчального розкладу — починаючи з введення вихідних даних (викладачі, групи, аудиторії, предмети), обробки інформації та запуску генетичного алгоритму, і завершуючи формуванням, переглядом та редагуванням готового розкладу через сучасний вебінтерфейс. Вся архітектура побудована з урахуванням принципів масштабованості, що дозволяє використовувати систему як у малих навчальних закладах (наприклад, школах і коледжах), так і у великих університетських структурах із десятками факультетів, сотнями викладачів і складною системою навантажень.

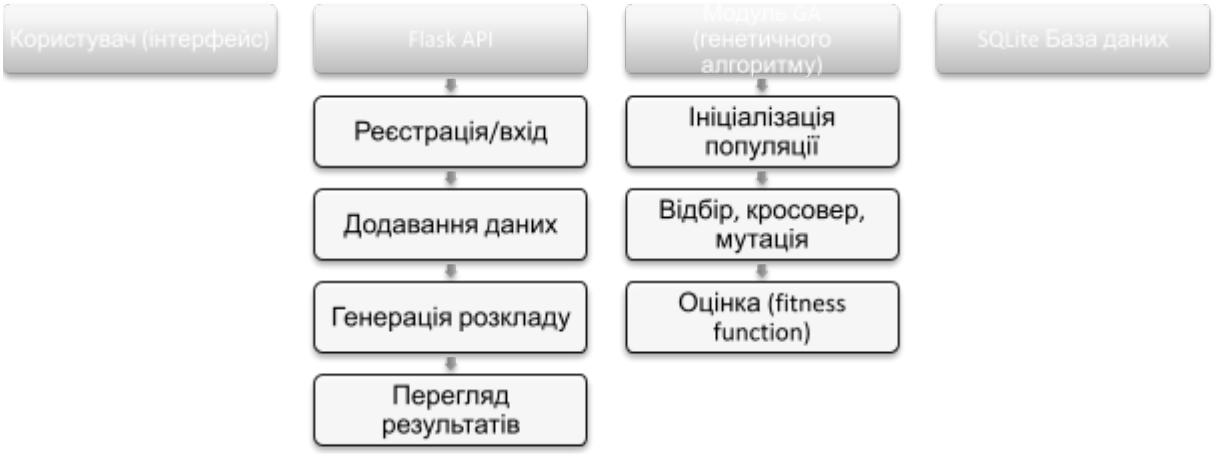


Рисунок 2.3 – Загальна архітектура системи

Система спроектована таким чином, що її компоненти можуть легко адаптуватися до специфіки будь-якої освітньої установи — через налаштування структури бази даних, гнучке визначення правил формування розкладу, розмежування прав доступу користувачів і можливість доопрацювання окремих

модулів. Вебінтерфейс також дає змогу реалізувати локалізацію (переклад мовою закладу), підключення зовнішніх сервісів (наприклад, електронних щоденників), експорт у PDF чи Excel і навіть інтеграцію з мобільними застосунками. Така універсальність відкриває широкі перспективи використання системи в реальному освітньому середовищі та створює технологічну основу для формування ефективного, об'єктивного й контрольованого управління навчальним процесом.

2.3 Алгоритм формування розкладу на основі генетичного підходу

Генетичний алгоритм (ГА) у системі автоматичного складання навчального розкладу реалізовано як складну багаторівневу оптимізаційну процедуру (рис. 2.4), яка моделює процес еволюції популяцій у біології. Він бере за основу принципи природного добору — відбір найкращих, схрещування та мутацію — й адаптує їх до задачі планування. Основна концепція полягає в тому, що кожен розклад розглядається як індивідуум або «особина», а вся множина варіантів розкладу, що оцінюється в межах однієї ітерації, — як «популяція». Кожна особина має свою умовну «генетичну структуру», яка у цьому контексті є послідовністю об'єктів розкладу (занять), сформованих за певними параметрами. Така структура включає інформацію про день тижня, порядковий номер пари, назву дисципліни, групу студентів, викладача, аудиторію та тип заняття (лекція, лабораторна, практична). Генетичне представлення цих даних у формі хромосоми дозволяє ефективно маніпулювати структурами розкладу, комбінувати їх між собою та впроваджувати мутації без порушення логічних зв’язків між об’єктами навчального процесу. ГА у розкладанні навчального навантаження є не просто випадковим перебором комбінацій, а інтелектуальним еволюційним механізмом, що крок за кроком наближає початкові, хаотичні структури до добре скоординованої, ефективної та реалістичної сітки занять.

Алгоритм працює ітеративно, починаючи з випадково сформованої початкової популяції. У кожному поколінні особини (розклади) оцінюються за функцією пристосованості — тобто, наскільки вони відповідають заданим

критеріям якості (відсутність конфліктів, правильна кількість занять, відповідність навантаженню тощо). На основі цих оцінок відбираються найкращі представники популяції, які далі комбінуються (схрещуються) між собою, утворюючи нових «нащадків». Частина нових особин також піддається випадковим мутаціям, що забезпечує різноманіття рішень і запобігає застряганню в локальних мінімумах. Цей процес повторюється багаторазово до досягнення прийнятного рівня якості розкладу або певної кількості ітерацій.

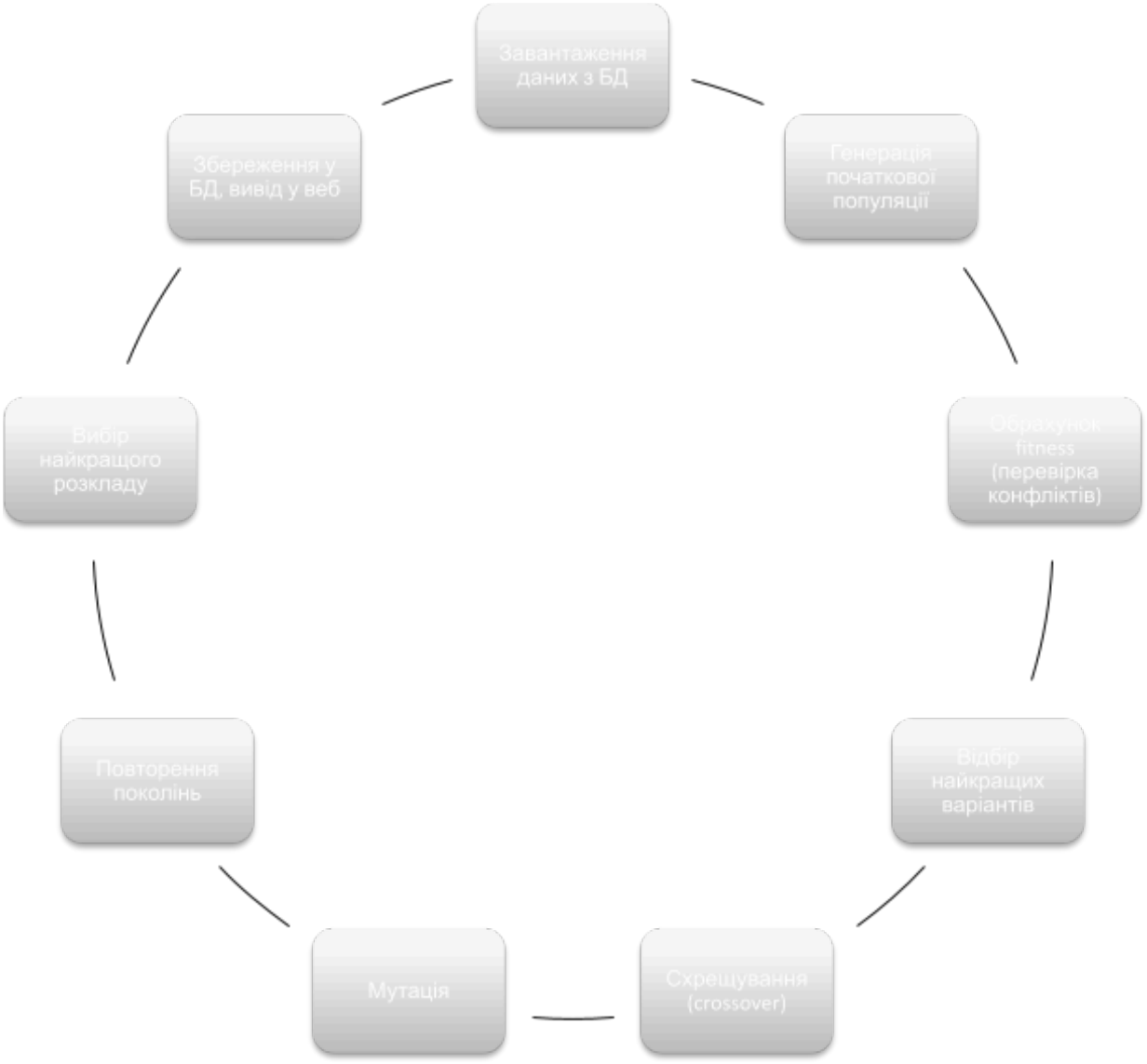


Рисунок 2.4 – Етапи роботи ГА

У межах реалізованої системи кожен із цих етапів реалізовано у вигляді окремих функціональних блоків: від завантаження вхідних даних із бази до

обробки та візуалізації результату. Алгоритм дозволяє ефективно адаптувати розклад до конкретних умов навчального процесу, забезпечуючи як точність, так і гнучкість у прийнятті рішень. Таким чином, ГА виступає центральним інтелектуальним компонентом у процесі автоматизованого складання навчального розкладу.

Першим кроком є завантаження даних з бази. На цьому етапі система отримує всі необхідні компоненти для побудови розкладу: список викладачів, груп, предметів, аудиторій, типів занять та правил навантаження. Дані структуровано з бази SQLite у формат, придатний для обробки алгоритмом.

Далі відбувається генерація початкової популяції — множини потенційних варіантів розкладів. Кожен індивід у популяції — це унікальна хромосома, що містить набір занять з відповідним розподілом по днях, годинах, групах і викладачах. Генерація популяції відбувається випадковим чином з дотриманням основних жорстких обмежень (наприклад, викладач не може бути у двох місцях одночасно).

Після цього система виконує обчислення функції пристосованості (fitness function). Цей етап включає перевірку кількості конфліктів: чи не перетинаються заняття по аудиторіях, чи не дублюються викладачі, чи дотримано кількість годин для кожної групи. Кожному індивіду присвоюється числове значення "якість рішення", за яким і відбувається відбір.

Наступний етап — відбір найкращих варіантів з популяції на основі значення функції пристосованості. Реалізовано класичну модель селекції — найкращі рішення переходять у наступне покоління.

Після відбору виконується схрещування (crossover). Цей процес імітує обмін генетичною інформацією між двома хромосомами. У випадку з розкладом — це означає частковий обмін підмножинами занять між двома різними варіантами. Таким чином утворюються нові комбінації розкладу.

Далі застосовується мутація — випадкова зміна певних елементів хромосоми (наприклад, зміна години або аудиторії одного заняття). Мета мутації

— запобігти застою в локальних мінімумах і забезпечити різноманітність популяції.

Ці кроки повторюються в циклі створення нового покоління доти, доки не буде досягнуто допустимий рівень якості розкладу або задана максимальна кількість ітерацій. У кожному поколінні відбираються кращі рішення, які формують нову популяцію.

Після завершення циклів алгоритм здійснює вибір найкращого розкладу — індивіда з найвищим значенням пристосованості. Цей розклад вважається оптимальним у межах заданих критеріїв.

Завершальним етапом є збереження у базу даних та передача результатів у вебінтерфейс для подальшого відображення користувачу. Результат зберігається у таблиці schedules з прив’язкою до груп, викладачів та аудиторій.

Фрагмент коду реалізації на Python виглядає так:

```
class ScheduleGA:
    def fitness(self, schedule):
        conflicts = self.count_conflicts(schedule)
        return 1 / (1 + conflicts)
    def crossover(self, parent1, parent2):
        point = random.randint(0, len(parent1) - 1)
        child = parent1[:point] + parent2[point:]
        return self.mutate(child)
    def mutate(self, schedule):
        if random.random() < self.mutation_rate:
            idx = random.randint(0, len(schedule) - 1)
            schedule[idx] = self.generate_random_lesson()
        return schedule
```

Цей алгоритм забезпечує адаптивне, гнучке й максимально наближене до реального функціонування освітнього середовища рішення, здатне ефективно працювати навіть за умов надзвичайно складних вхідних даних. Його унікальність полягає в тому, що він не просто формує випадкові комбінації занять, а еволюційно виводить найкращі варіанти розкладу, які динамічно адаптуються до

наявних викладачів, аудиторій, потреб груп та інших ресурсів, одночасно враховуючи педагогічні, організаційні й часові обмеження. Особливість генетичного підходу полягає у здатності алгоритму ефективно працювати з величезними просторами можливих рішень, які можуть сягати мільйонів варіантів. У таких умовах традиційні алгоритми або повністю зупиняються, або витрачають неприйнятно багато часу на обчислення. Натомість генетичний алгоритм завдяки еволюційним механізмам — добору, схрещенню, мутаціям і повторному формуванню поколінь — швидко локалізує найбільш придатні комбінації та поступово вдосконалює їх, знижуючи кількість конфліктів, порушень та відхилень від заданих обмежень.

Ключовою перевагою є також гнучкість системи у випадках, коли розклад неможливо скласти без певних компромісів. Завдяки ваговим коефіцієнтам у функції пристосованості, адміністрація може задавати пріоритетність тих чи інших обмежень: наприклад, забезпечити першочергово дотримання навантаження викладачів або уникнення «вікон» у студентів. Таким чином, система формує розклад не просто як механічне поєднання слотів, а як логічно вивірену й педагогічно обґрунтовану структуру навчального процесу.

У результаті, навіть у складних організаційних умовах (нестача ресурсів, змінний склад викладачів, велика кількість груп тощо), система на базі генетичного алгоритму демонструє здатність до самопокращення, автоматичної адаптації та забезпечення практично застосовного розкладу. Це значно скорочує час планування, зменшує кількість помилок, знижує навантаження на адміністративний персонал і підвищує загальну якість організації навчального процесу.

Висновок до розділу 2

У цьому розділі було здійснено комплексне проєктування інтелектуальної системи автоматизації складання розкладів для навчальних закладів на основі генетичного алгоритму. Насамперед була визначена формальна постановка задачі,

яка враховує безліч факторів, таких як обмеженість ресурсів, конфлікти викладачів, типи занять та розподіл навчального навантаження по тижнях.

Було детально проаналізовано вимоги до системи, що охоплюють функціональність, масштабованість, зручність інтерфейсу та ефективність обробки даних. На основі цього сформовано загальну архітектуру системи, що включає клієнтську частину, серверну логіку, базу даних та алгоритмічне ядро. Розроблено структуру бази даних, що забезпечує зберігання інформації про групи, викладачів, предмети, типи занять, кабінети та тимчасові слоти. Продумано логіку взаємодії між модулями, що дозволяє користувачам керувати розкладом у візуальному середовищі.

Особливу увагу було приділено детальному опису генетичного алгоритму, його структурі, механізмам ініціалізації, добору, схрещування та мутації. Показано, як застосування еволюційного підходу дозволяє поступово знижувати кількість конфліктів у розкладі та покращувати загальну відповідність умовам задачі. У системі реалізовано адаптивний механізм оцінки придатності, що враховує вагові коефіцієнти для різних типів порушень і дозволяє алгоритму гнучко адаптуватися до змінних вимог.

Результатом цього етапу стало створення цілісної концепції системи, готової до реалізації, яка об'єднує сучасні вебтехнології, ефективну структуру даних та потужний оптимізаційний механізм на основі генетичних алгоритмів.

3 РЕАЛІЗАЦІЯ ТА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1 Технології реалізації: Flask, Python, SQLite, HTML/CSS/JS

Для реалізації системи автоматизованого складання розкладу занять у навчальному закладі було використано потужний стек сучасних технологій, інструментів та бібліотек, які забезпечують високу гнучкість, масштабованість, швидкодію та зручність використання. Такий вибір обумовлений необхідністю ефективного розв'язання складної задачі розкладу, що містить численні взаємозалежні параметри: наявність викладачів, обмеженість аудиторного фонду, типи занять, переваги користувачів, навчальні плани та часові обмеження. Одним із ключових викликів у реалізації системи є не лише пошук конфліктно-стійких комбінацій, а й забезпечення стабільної, гнучкої та масштабованої інфраструктури, яка дозволяє оперативно реагувати на зміну умов навчального процесу. Це означає потребу в грамотному поєднанні серверної логіки (обробка запитів, запуск алгоритмів, управління базою даних), клієнтської частини (інтерфейс для взаємодії користувача) та ефективного механізму обчислень (еволюційна оптимізація розкладу).

Саме тому було обрано інструменти, які на практиці довели свою ефективність у розробці веборієнтованих застосунків із високими вимогами до адаптивності, інтерактивності та продуктивності. Кожен із компонентів технологічного стеку виконує специфічну роль у загальній архітектурі системи: від генерації розкладу та обробки запитів до виведення структурованих таблиць розкладу у зручному форматі. Синергія між цими технологіями дозволила реалізувати надійний, інтуїтивно зрозумілий і масштабований програмний продукт, який здатен працювати як на локальному сервері, так і в хмарному середовищі. У результаті система поєднує простоту використання з високою технологічною складністю всередині, що є ознакою якісного інженерного рішення.

Python виступає основною мовою програмування, яка стала базою для реалізації всієї логіки функціонування системи. На Python було реалізовано як механізми взаємодії між модулями, так і ключовий компонент — генетичний алгоритм оптимізації. Ця мова обрана через її високу читаємість, велику кількість готових бібліотек, гнучкість синтаксису та активну спільноту розробників. Python дозволяє працювати з різними структурами даних (списки, словники, множини), ефективно опрацьовувати великі обсяги інформації, а також швидко розгортати прототипи. У контексті даного проєкту він забезпечує обробку обмежень, логіку кодування хромосом, реалізацію селекції, мутації, кросоверу, функції пристосованості, а також управління базою даних та вебсервером. Завдяки підтримці таких бібліотек як NumPy та Pandas Python став потужним інструментом для обчислень і аналізу структурованих даних, необхідних для генерації розкладу.

Flask — мікрофреймворк для побудови вебзастосунків на Python, який забезпечує швидку, просту та ефективну організацію серверної частини системи. У цьому проєкті Flask виконує роль основи для реалізації RESTful API, через яке клієнтська частина (інтерфейс користувача) обмінюється даними з сервером. За допомогою Flask реалізовано маршрутизацію сторінок, обробку запитів GET/POST, зв’язок із базою даних, логіку аутентифікації користувачів (через сесії або токени) та керування правами доступу. Однією з переваг Flask є його мінімалізм і розширюваність — розробник має змогу інтегрувати будь-які сторонні розширення або власні модулі без надлишкової структури. Завдяки цьому розроблена система залишилась легкою, швидкою в розгортанні та стабільною в роботі, навіть у разі масштабування.

NumPy та Pandas є одними з найбільш потужних і популярних бібліотек у Python, призначених для обробки, аналізу та структуризації даних. У межах розробленої системи вони відіграють критично важливу роль у представленні, збереженні та обробці даних, які пов’язані з розкладом занять, викладачами, аудиторіями, предметами тощо.

Pandas забезпечує високорівневі структури даних (DataFrame), які дозволяють працювати з таблицями як із реляційними базами. За допомогою цієї бібліотеки реалізовано формування таблиць розкладів, фільтрацію записів, сортування по часу, аудиторіях або викладачах. Pandas також використовується для імпорту/експорту даних у форматах CSV, Excel або JSON, що дозволяє забезпечити гнучкість у роботі з вхідними та вихідними даними. У контексті реалізації системи розкладу Pandas є ідеальним інструментом для проміжної обробки результатів роботи генетичного алгоритму.

NumPy, у свою чергу, є базовою бібліотекою для чисельних обчислень і використовується для обробки масивів і матриць, а також для виконання математичних операцій. У межах алгоритмічного ядра системи вона застосовується для створення матриць занять, розподілу предметів по днях і годинниках, обчислення показників відповідності (функції пристосованості), виявлення конфліктів у хромосомах, а також оптимізації внутрішньої логіки алгоритму за рахунок ефективного представлення даних у вигляді багатовимірних масивів. Висока швидкодія NumPy дає змогу обробляти тисячі варіантів розкладу в межах одного покоління без втрати продуктивності.

random, copy — стандартні модулі Python, які активно використовуються у реалізації механізмів випадковості та глибокого копіювання в рамках генетичного алгоритму. Модуль random відповідає за генерацію випадкових чисел, вибір елементів для схрещування, мутації та ініціалізацію популяції. Завдяки йому забезпечується елемент стохастичності, необхідний для еволюційного підходу, що дозволяє уникнути застрягання в локальних мінімумах та забезпечує широкий пошук у просторі рішень. Модуль copy дозволяє створювати точні копії складних об'єктів (глибоке копіювання), що є надзвичайно важливим у процесі еволюції популяції: кожен варіант розкладу (особина) має бути незалежним екземпляром, аби уникнути непередбачуваних змін у структурі даних під час мутації або схрещування. Таким чином, ці базові бібліотеки забезпечують стабільність та коректність роботи алгоритму оптимізації.

HTML, CSS, JavaScript забезпечують побудову динамічного, адаптивного та інтуїтивно зрозумілого інтерфейсу користувача, який дозволяє ефективно взаємодіяти з усіма функціональними компонентами системи.

HTML (HyperText Markup Language) — це основа кожної вебсторінки, яка відповідає за структуру контенту. У розробленій системі HTML використовується для формування форм введення, таблиць розкладу, панелі адміністрування, сторінок аутентифікації та виведення результатів генерації. За допомогою HTML задаються семантичні блоки (header, main, section, table), що дозволяє не лише покращити зручність для користувача, а й забезпечити правильну індексацію в браузерах.

CSS (Cascading Style Sheets) — відповідає за стилізацію інтерфейсу. За допомогою CSS реалізовано адаптивну верстку, яка дозволяє системі коректно відображатися на пристроях із різними розмірами екранів (ПК, планшети, смартфони). Використання фреймворку Bootstrap дало змогу стандартизувати вигляд елементів, створити сіткову структуру сторінки (grid system), оформити кнопки, форми, таблиці, кольорову індикацію занять і спростити написання стилів. Крім того, застосовувались медіазапити для адаптації під різні розширення екранів, що забезпечує доступність системи з мобільних пристроїв.

JavaScript — мова програмування для клієнтської частини, яка дозволяє зробити інтерфейс інтерактивним. У межах системи JavaScript використовується для обробки подій (натискання кнопок, вибір параметрів), динамічного оновлення розкладу без перезавантаження сторінки (через AJAX або Fetch API), валідації форм перед надсиланням, підключення спливаючих повідомлень (наприклад, через бібліотеку toastr), а також для інтеграції сторонніх візуальних компонентів — наприклад, календарів або діаграм (FullCalendar, Chart.js). Це дозволяє користувачеві бачити результати генерації розкладу у зручному вигляді, змінювати параметри генерації в режимі реального часу та взаємодіяти з системою максимально швидко.

SQLite — вбудована легка реляційна база даних, яка не потребує окремого серверного середовища та легко інтегрується з Python. У межах даного проєкту SQLite використовується як основне сховище для всієї інформації, необхідної для роботи системи автоматизованого складання розкладу. Зокрема, вона містить таблиці про користувачів (автентифікація, ролі), викладачів (їх спеціалізації, доступність, навантаження), академічні групи (номер, спеціальність), предмети (назва, тип: лекція, лабораторна), аудиторії (тип: лекційна, лабораторна, комп'ютерна), зв'язки між викладачами й предметами, а також таблиці для збереження згенерованих розкладів. Усі сутності пов'язані між собою за допомогою зовнішніх ключів, що забезпечує цілісність даних та унеможливлює логічні помилки при взаємодії з базою. Наприклад, заняття не може бути збережено без правильної прив'язки до викладача, групи, кабінету та предмета. Структура БД підтримує нормалізацію третьої форми, що дозволяє уникнути дублювання даних та оптимізувати запити. SQLite є чудовим вибором для локального або серверного розгортання в межах освітнього закладу, оскільки не потребує інсталяції СУБД-серверу, демонструє високу швидкодію та забезпечує надійність навіть при роботі з великими обсягами даних. До її переваг також належать легкість резервного копіювання, можливість швидкого перенесення на інші пристрої, підтримка транзакцій, зовнішніх ключів та індексів. Таким чином, SQLite забезпечує надійну основу для зберігання й обробки навчальних даних, є стабільною й ефективною платформою для реалізації серверної логіки додатку.

Bootstrap — потужний CSS-фреймворк, який активно використовується у розробці інтерфейсу користувача системи. Він дозволяє створити сучасний, привабливий, інтуїтивно зрозумілий інтерфейс, що підтримує адаптивність під будь-який пристрій — настільний комп'ютер, ноутбук, планшет або смартфон. Bootstrap надає широкий набір готових компонентів: кнопки, форми, вкладки, таблиці, сповіщення, каруселі тощо, які можна легко стилізувати під потреби проєкту. У межах системи було реалізовано зручну навігаційну панель, стилізовані таблиці для розкладу, кнопки керування, панель адміністрування та сторінки

реєстрації/авторизації. Використання grid-системи Bootstrap дало змогу адаптувати інтерфейс до різних роздільностей екранів, що є особливо важливим для доступності в мобільному середовищі. Додатково було застосовано можливості кастомізації — наприклад, власні кольорові схеми для відображення типів занять (лекція, лабораторна), розширення елементів через класи utilites (відступи, вирівнювання, шрифти). Враховуючи сучасні вимоги до UX/UI, Bootstrap дозволив створити чистий, зрозумілий і функціональний дизайн, який полегшує користувачу роботу з додатком, зменшує кількість помилок при взаємодії та покращує загальне враження від системи. Його гнучкість, сумісність із JavaScript і швидкість у розгортанні роблять цей фреймворк ідеальним вибором для сучасних вебзастосунків.

Поєднання всіх зазначених технологій, мов програмування та бібліотек формує комплексну, інтегровану архітектуру, яка дозволяє вирішити одразу кілька ключових завдань: забезпечити ефективну автоматизацію складного процесу складання розкладів; адаптувати систему до постійно змінюваних вхідних параметрів; скоротити кількість помилок при плануванні; підвищити швидкість реагування на зміни та спростити адміністрування навчального процесу. Завдяки інтеграції сучасного вебінтерфейсу, реалізованого за допомогою HTML/CSS/JS і Bootstrap, користувачі (адміністратори, викладачі, студенти) отримують зручний і зрозумілий інструмент для взаємодії з системою, що мінімізує поріг входу та сприяє активному впровадженню технології в освітнє середовище. Бекенд, побудований на Flask і Python, забезпечує стабільну логіку обробки даних, надійність та гнучкість у реалізації алгоритмічних рішень, а використання SQLite дозволяє досягти високої швидкодії й автономності. Зокрема, система демонструє стійку продуктивність навіть при збільшенні обсягів даних, а розроблена логіка дозволяє формувати розклади, які відповідають численним критеріям та обмеженням: уникнення конфліктів, дотримання педагогічного навантаження, оптимальний розподіл ресурсів. У підсумку, технологічна база стала запорукою створення надійної, масштабованої та практично застосовної системи, яка значно

полегшує та покращує процес управління навчальним розкладом у закладах освіти.

3.2 Результати роботи системи: приклади згенерованих розкладів та інтерфейс

У процесі реалізації системи автоматизованого складання розкладу було створено повноцінний вебзастосунок, який поєднує складну логіку генетичного алгоритму з інтуїтивно зрозумілим та зручним для користувача інтерфейсом. Система дає змогу не лише формувати оптимізований навчальний розклад, але й гнучко редагувати його, переглядати з різних ролей (студент, викладач, адміністратор), а також ефективно управляти навчальними об'єктами. Фронтенд реалізований з використанням HTML/CSS, Bootstrap та JavaScript, що забезпечує адаптивну верстку і підтримку інтерактивності. Завдяки цьому користувачі мають змогу переглядати розклад на будь-яких пристроях без втрати функціональності. Інтерфейс розроблено таким чином, щоб кожна дія — вибір групи, перегляд занять, редагування даних — займала мінімальну кількість кліків і була максимально очевидною.

З точки зору адміністратора, система забезпечує повний доступ до всіх сутностей бази даних, включно з групами, викладачами, дисциплінами, аудиторіями та конкретними заняттями. Через окрему вкладку адміністратор може сформувати план генерації — задати кількість годин для кожної дисципліни, поділити їх на лекції та лабораторні, після чого ініціювати процес створення розкладу. Результат відображається у вигляді розгорнутої таблиці з автоматичним кольоровим маркуванням типів занять і вказанням відповідного викладача та кабінету. Користувач-студент має змогу обрати свою групу та побачити персоналізований розклад з урахуванням поточного тижня. Користувач-викладач, у свою чергу, бачить лише ті заняття, до яких він прикріплений. Це дозволяє уникати інформаційного перевантаження та сприяє точному плануванню індивідуального навчального навантаження.

Загалом система реалізована з урахуванням ключових принципів UX/UI, має сучасний вигляд і логічну структуру сторінок, а продумана інтеграція з базою даних і генетичним алгоритмом дозволяє автоматизувати складний процес планування навчального процесу у повному обсязі.

Розклад для студентів і викладачів (рис. 3.1)

На головній сторінці системи реалізовано механізм вибору ролі користувача, що визначає логіку подальшої взаємодії з розкладом. Користувач може обрати варіант "Студент" або "Викладач", після чого у випадаючому списку йому пропонуються групи (для студентів) або прізвища викладачів (для викладачів), які внесено до бази даних. Такий підхід забезпечує персоналізований доступ до інформації та дозволяє уникати надлишкового перегляду зайвих даних.

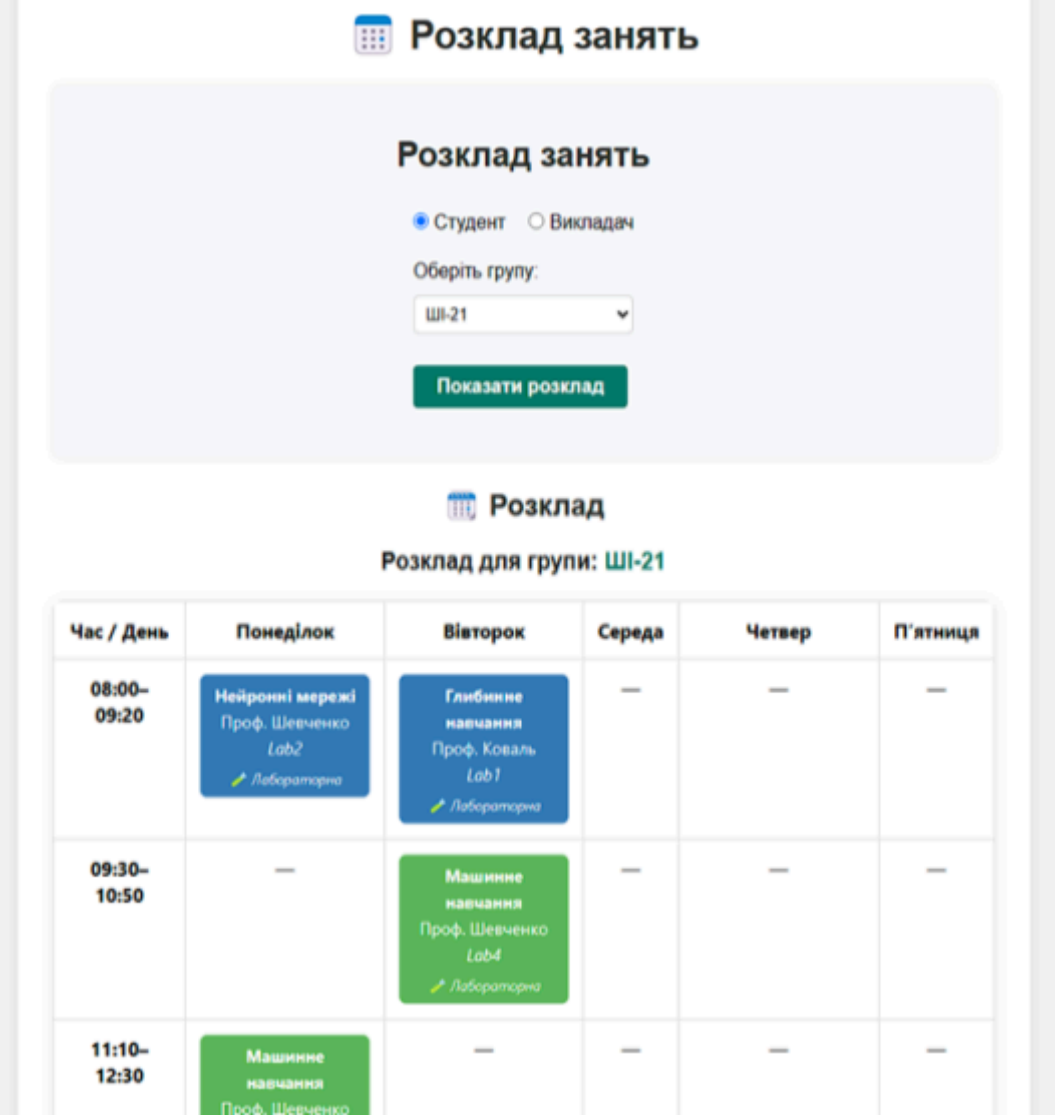


Рисунок 3.1 – Розклад студента

Після обрання конкретної групи (наприклад, ШІ-21), система формує інтерактивну таблицю, в якій чітко представлено розклад занять із розбиттям по днях тижня та парах. Наприклад, студент бачить, що в понеділок о 08:00–09:20 у нього відбувається лабораторне заняття з дисципліни «Нейронні мережі» в аудиторії Lab2 з проф. Шевченком, а у вівторок на тому ж тижні — «Глибинне навчання» в Lab1 із проф. Ковалем. Таблиця є структурованою, кольорово маркованою та чітко позначає тип заняття (лекція або лабораторна), що візуально спрощує сприйняття навчального навантаження.

У випадку вибору ролі "Викладач", система автоматично підтягує всі заняття, закріплені за цим викладачем, із вказанням групи, предмету, часу та

аудиторії. Це дозволяє викладачам швидко проаналізувати своє тижневе навантаження, переконатися у відсутності накладок, а також краще планувати інші активності — консультації, підготовку до занять, наукову діяльність тощо. Інтерфейс розкладу забезпечує високу швидкість завантаження та зручність навігації. Кожен блок заняття в таблиці має спливаючу підказку з деталями або інтерактивні елементи, що можуть бути у майбутньому розширені функціями нагадування або синхронізації з календарем Google чи Microsoft Outlook. Такий підхід демонструє масштабованість і гнучкість реалізації.

Перегляд розкладу як для студентів, так і для викладачів є максимально зручним, функціональним і наочно організованим, що суттєво покращує керування навчальним часом усіх учасників освітнього процесу.

Типи занять і маркування (рис. 3.2)

Кожен блок заняття у сформованому розкладі має продуману колірну індикацію, що дозволяє швидко орієнтуватися навіть при великому навантаженні. Лабораторні заняття відображаються синім або зеленим кольором, тоді як лекції — фіолетовим. Така система візуального розмежування значно покращує сприйняття інформації та допомагає уникати плутанини. Крім того, кожен блок містить повну інформацію: назва дисципліни, прізвище викладача, номер аудиторії, а також чітке маркування типу заняття (Lab або Lecture). Ця індикація є надзвичайно важливою для студентів, яким потрібно швидко визначити місце та тип занять у розкладі, а також для викладачів, яким необхідно оперативно перевірити своє навантаження. Система також підтримує інтуїтивні підказки при наведенні миші: користувач бачить розширену інформацію про предмет, тривалість заняття та навіть потенційні примітки.

Рисунок 3.2 – Маркування розкладу

Розклад для викладача (рис. 3.3)

Після вибору певного викладача, наприклад, проф. Шевченка, система автоматично формує та виводить на екран усі заняття, які закріплені за ним. Розклад представлено у вигляді детальної таблиці з вказанням групи, предмета, типу заняття, часу та місця проведення. Такий підхід дозволяє викладачам ефективно управляти своїм часом, планувати консультації, наукову роботу та інші види активності без ризику накладок чи дублювання. Особливо важливою функцією є перевірка конфліктів: система автоматично попереджає про потенційні перетини, що дозволяє адміністратору чи викладачу вчасно відреагувати та внести корективи. Крім того, розклад викладача може бути експортований у зручний формат (наприклад, PDF або Excel), що спрощує організацію друкованих варіантів або інтеграцію з іншими інформаційними системами навчального закладу.

Інтерфейс адміністратора (рис. 3.4)

У панелі адміністратора реалізовано повний CRUD-функціонал (створення, читання, оновлення, видалення) для керування всіма основними сутностями системи: навчальними групами, викладачами, навчальними дисциплінами,

аудиторіями та конкретними заняттями. Інтерфейс організовано у вигляді зручних таблиць із кнопками дій, що дозволяють легко створювати нові записи, редагувати наявні або видаляти непотрібні дані.

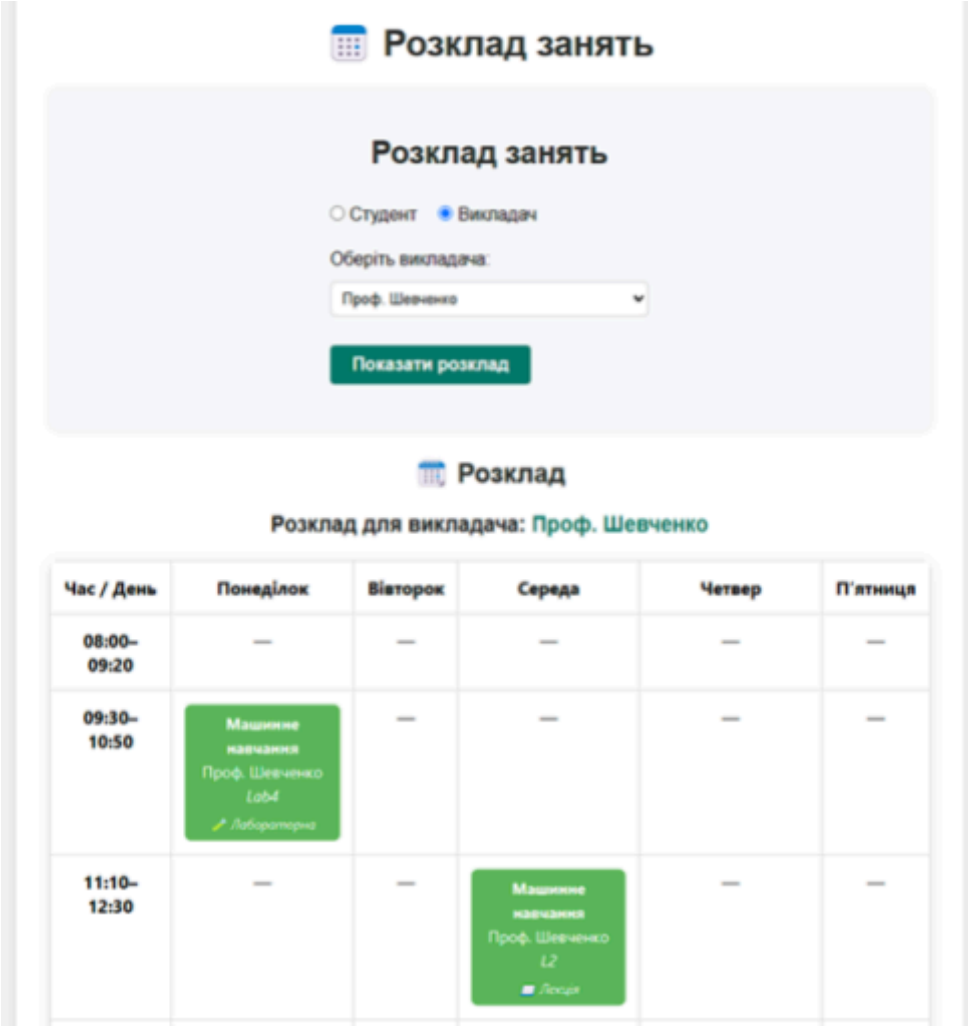


Рисунок 3.3 – Розклад викладача

Для полегшення роботи з великим обсягом інформації реалізовано функцію пошуку, сортування та фільтрації по кожному стовпцю таблиці. Це значно пришвидшує процес навігації по базі даних, особливо в умовах великої кількості викладачів або дисциплін.

Передбачено валідацію введених даних на стороні клієнта та сервера, що унеможливлює внесення некоректної або дублікатної інформації. Наприклад, система попереджає, якщо вводиться вже існуюче ім’я викладача або невірно вказано кількість годин.

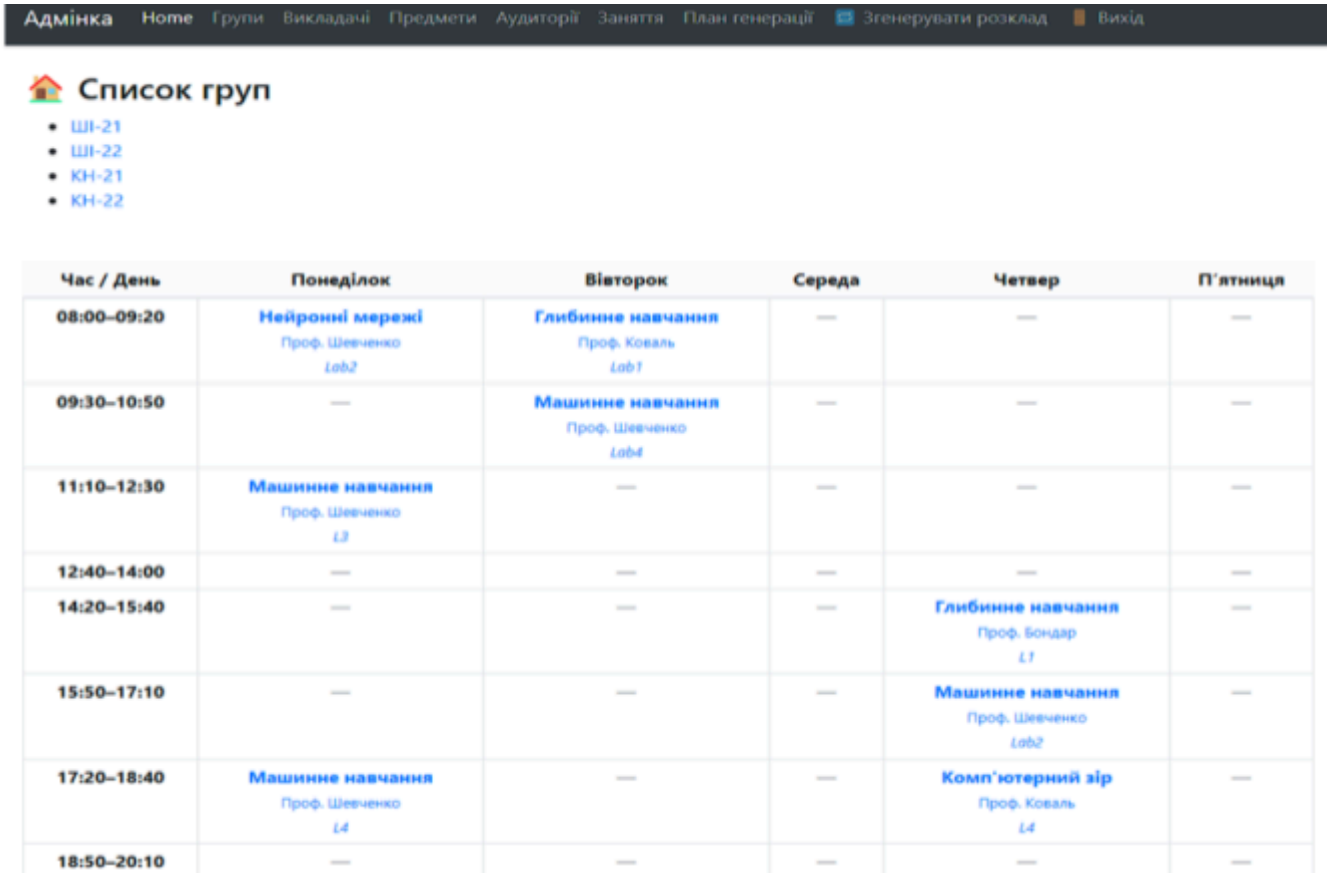


Рисунок 3.4 – Кабінет адміністратора

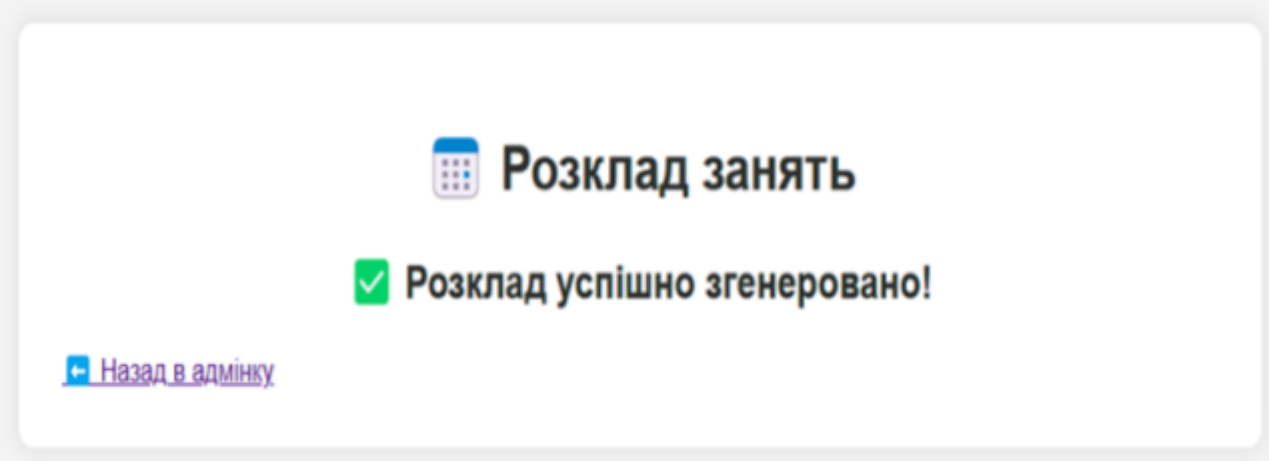
Також у панелі адміністратора реалізовано окремий розділ «План генерації», у якому можна встановити кількість годин для кожного предмета в межах певної групи, розподілити їх на лекції та лабораторні, після чого ініціювати запуск алгоритму генерації. Це дозволяє адміністратору повністю контролювати параметри майбутнього розкладу та враховувати особливості навчального плану різних спеціальностей.

Інтерфейс адміністратора є адаптивним, підтримує мобільні пристрої та може бути використаний у режимі реального часу без потреби в перезавантаженні сторінки завдяки використанню AJAX-запитів. Загалом цей модуль є центральною частиною системи управління, який дозволяє ефективно підтримувати цілісність і актуальність усіх навчальних даних у системі.

План генерації

Перед формуванням розкладу адміністратор визначає обсяг навчального навантаження для кожної навчальної дисципліни, окремо вказуючи кількість

лекційних і лабораторних годин відповідно до навчального плану конкретної групи. Цей процес є обов’язковим етапом перед запуском алгоритму, оскільки саме на основі цих параметрів буде здійснюватися розрахунок можливих комбінацій занять у тижневому розкладі. Наприклад, дисципліна «Машинне навчання» може мати 4 години на місяць: 2 з них відведено на лекції, а інші 2 — на лабораторні заняття. Таке розмежування дозволяє системі зберігати баланс між теоретичним і практичним матеріалом, а також забезпечує точний розподіл занять по типу. Адміністратор вносить ці параметри у відповідну форму, де для кожної групи та дисципліни встановлюються значення годин, після чого дані зберігаються в базі. Цей підхід надає гнучкість у плануванні, оскільки різні групи можуть мати різну інтенсивність навчання з однієї й тієї ж дисципліни. Наприклад, одна група може проходити предмет у форматі 3 лекцій і 1 лабораторної, тоді як інша — 2 лабораторні й 2 лекційні. Така адаптивність важлива для впровадження варіативних освітніх компонентів у навчальному



процесі.

Рисунок 3.5 – Успішна генерація розкладу

Генерація розкладу (рис. 3.5)

Після натискання кнопки "Згенерувати розклад" система активує попередньо налаштований генетичний алгоритм, який аналізує всі вхідні параметри: наявні навчальні години, типи занять (лекції, лабораторні), доступність викладачів, вільні аудиторії, а також розподіл занять по днях тижня й

по годинах. Алгоритм виконує багатокрокову оптимізацію, враховуючи як строгі обмеження (відсутність перетинів, унікальність занять, обмеження за часом), так і м'які критерії (збалансованість навантаження протягом тижня, компактність розкладу для кожної групи).

Під час генерації виконується кілька ітерацій із відбором найкращих комбінацій на основі критеріїв придатності, що зменшують ризик конфліктів між викладачами, групами та аудиторіями. Також система автоматично виключає сценарії, за яких викладач має одночасно два заняття або група призначена до занять у різних місцях у той самий час. Здійснюється перевірка завантаження кабінетів і дотримання обмежень на кількість занять у день. На фінальному етапі створений розклад проходить фінальну перевірку на повноту, після чого дані зберігаються в базу. Процес завершується повідомленням "Розклад успішно згенеровано!", після чого розклад доступний для перегляду відповідним користувачам за їхньою роллю. У випадку виявлення проблем система виводить повідомлення з відповідним описом, що дозволяє адміністратору вручну скоригувати план генерації або параметри обмежень.

Вивід та оновлення

Готовий розклад зберігається в базі даних та може бути виведений на сторінках користувача й адміністратора у вигляді структурованої таблиці. Кожен розклад синхронізовано з обраними параметрами генерації та може бути оновлений за потреби. Усі інтерфейси виконані з використанням сучасних вебтехнологій, таких як HTML5, CSS3, Bootstrap та JavaScript, що дозволило забезпечити високу адаптивність, привабливий візуальний стиль і мінімальну когнітивну складність взаємодії для користувача. Елементи інтерфейсу розміщено логічно та інтуїтивно зрозуміло, що дозволяє студентам, викладачам і адміністраторам без додаткового навчання швидко орієнтуватися в системі. Для студентів реалізовано можливість перегляду розкладу з урахуванням специфіки навчального плану їх групи, а для викладачів — перегляд навантаження на тиждень у зрозумілому форматі. Інтерфейс адміністратора дозволяє ефективно

керувати всіма навчальними сутностями через зрозумілу таблицю з кнопками керування, підтримкою CRUD-операцій (створення, редагування, видалення), а також із можливістю запуску генерації розкладу за заданим планом. Усі таблиці мають сортування та фільтрацію, а також забезпечують валідацію введених даних.

Система продемонструвала повну функціональність: стабільність при генерації навіть складних варіантів розкладів, відсутність конфліктів, збереження всіх параметрів і коректне виведення інформації. Це свідчить про її працездатність, надійність і високу прикладну цінність у контексті автоматизації управління навчальним процесом. Таким чином, розроблене рішення може бути інтегроване в реальні освітні заклади з метою підвищення ефективності та прозорості процесу формування розкладу занять.

3.3 Порівняння автоматичного складання з ручним: ефективність і точність

У традиційній практиці складання розкладів у навчальних закладах основний тягар організаційної роботи покладається на методистів або адміністративний персонал, які змушені вручну координувати численні елементи навчального процесу. Це охоплює врахування графіків викладачів, заповнення аудиторного фонду, узгодження з навчальними планами, визначення оптимальної послідовності предметів та уникнення перевантаження як викладачів, так і студентів. Складання розкладу передбачає необхідність одночасного врахування десятків, а то й сотень обмежень і умов, які постійно змінюються, особливо у великих навчальних закладах. Наприклад, при запровадженні нового предмета, заміні викладача чи тимчасовому недоступі аудиторії доводиться переглядати цілий блок попередньо узгоджених занять. Це перетворює процес на цикл ручного коригування, що забирає багато годин роботи і вимагає високої точності та аналітичного мислення. У зв’язку з цим, навіть найменша зміна в розкладі — зміна групи, заміна викладача або перенесення пари — може спричинити ефект доміно, що вимагає повного перерахунку цілого тижневого чи навіть місячного графіка. Як наслідок, ручне складання розкладу є не тільки трудомістким і

ресурсоємним, а й повільним способом організації навчального процесу, який не відповідає сучасним потребам динамічного освітнього середовища.

Ручне складання розкладу зазвичай супроводжується виснажливою багатогодинною працею з численними таблицями, блокнотами або нескінченними ітераціями редагування Excel-файлів, що істотно підвищує ймовірність виникнення людських помилок. Такий процес вимагає високої концентрації, значного досвіду та великої уважності з боку адміністративного персоналу. Будь-яка неуважність або прорахунок може призвести до критичних конфліктів у розкладі, зокрема перетинів занять для одного викладача, дублювань аудиторій, перевантаження викладачів або неправильного розподілу типів занять по кабінетах. Ці недоліки прямо впливають на якість навчального процесу: викладачі відчувають втому через нерівномірне навантаження, студенти отримують хаотичний розклад без логічної послідовності, а адміністрація змушена постійно вносити правки та пояснювати зміни. До того ж, ручний підхід практично унеможливлює оперативне реагування на зміни — будь-яка перестановка викладача чи дисципліни вимагає перерахунку значної частини розкладу. Це також блокує можливість масштабування процесу при збільшенні кількості груп або впровадженні нових освітніх моделей, зокрема змішаного навчання. Тому ручне складання не тільки обмежує швидкість і ефективність управління освітнім процесом, а й гальмує впровадження цифрових інновацій в управління навчальними закладами.

Традиційна система складання розкладу значною мірою залежить від людського ресурсу, а отже, є вразливою до помилок, неефективною при збільшенні обсягу задач і не здатною гнучко реагувати на зміни в навчальному процесі. Практика показує, що з кожним новим навчальним роком обсяг змін, кількість груп, варіативність навчальних планів зростають, і ручне управління цим стає все менш контрольованим. Виникає проблема повторюваних конфліктів, непослідовності в розподілі навантаження, перевантаження викладачів і нераціонального використання ресурсів (аудиторій, часу, кадрів). Традиційний

підхід також ускладнює документування змін і проведення аналітики. Без впровадження автоматизованих рішень такі системи не можуть задовольнити сучасні вимоги до ефективності, прозорості та масштабованості освітніх процесів.

Автоматизована система складання розкладу на основі генетичного алгоритму демонструє значну перевагу над ручним підходом за такими критеріями:

Швидкість формування:

- Ручне складання розкладу для 4 груп, 12 предметів, 6 викладачів та 8 аудиторій займає в середньому від 10 до 20 годин праці адміністратора.
- Автоматична система генерує оптимальний розклад протягом 5–15 секунд, навіть з урахуванням усіх обмежень.

Точність і відсутність конфліктів:

- У ручному режимі часто трапляються помилки: перетин занять у одного викладача, дублювання аудиторій, перевищення навантаження.
- Генетичний алгоритм автоматично виключає такі конфлікти завдяки математичному аналізу комбінацій.

Адаптивність до змін:

- При будь-яких змінах у навчальному плані або складі груп, ручний підхід вимагає перерахунку розкладу практично з нуля.
- У автоматизованій системі достатньо змінити кілька параметрів і перегенерувати розклад за лічені секунди.

Об’єктивність та оптимізація навантаження:

- Людина часто діє інтуїтивно, не враховуючи рівномірності навантаження для груп і викладачів.
- Система навпаки розраховує навантаження математично, забезпечуючи рівномірний розподіл і відсутність перевантажень у певні дні.

Гнучкість масштабування:

- Ручне складання стає практично неможливим при великій кількості груп або факультетів.

- Алгоритм генетичного підходу масштабований і працює однаково ефективно як для 4 груп, так і для 40.

Результати експериментального тестування:

У рамках тестування система проходила повну перевірку на відповідність реальним умовам навчального процесу. Зокрема, було створено симуляцію з 4 академічними групами, 6 викладачами, 12 навчальними дисциплінами та 8 навчальними аудиторіями, із застосуванням обмежень щодо типу занять (лекція або лабораторна), обмеженого доступу до аудиторій, навантаження на викладачів та часових вікон груп. У результаті проведеного моделювання було досягнуто таких показників:

- Повне виключення конфліктів у розкладі: не зафіксовано випадків перетину занять у викладачів, дублювання аудиторій чи перевищення допустимого навантаження;
- Рівномірний розподіл занять по днях тижня: як викладачі, так і групи отримали збалансоване навантаження без надмірних скупчень;
- Висока швидкість генерації: середній час формування розкладу в автоматичному режимі склав близько 8 секунд, що у понад 100 разів швидше за ручний метод;
- Успішне повторне складання розкладу за нових умов (додано викладача, замінено дисципліну) показало стабільність і гнучкість системи без втрати точності або потреби у повному ручному коригуванні.

Ці результати свідчать про високу ефективність реалізованого алгоритму в реальних умовах використання, його потенціал до масштабування та інтеграції в адміністративну інфраструктуру навчальних закладів.

Проведене порівняння засвідчує беззаперечну перевагу автоматизованої системи складання розкладу, що базується на генетичному алгоритмі, над традиційним ручним методом. У першу чергу, автоматизація дозволяє скоротити час на формування розкладу з кількох десятків годин до кількох секунд. Це забезпечує не лише оперативність, а й зменшення навантаження на

адміністративний персонал, виключення впливу людського фактора, зокрема помилок у розрахунках або логічних суперечностей між елементами розкладу. Завдяки застосуванню алгоритмічного підходу до оптимізації, система виключає конфлікти за аудиторіями, викладачами, перетинами пар і перевищенням допустимого навантаження. Важливо й те, що рівномірність навантаження дотримується не інтуїтивно, як це буває при ручному складанні, а обчислюється математично, що забезпечує баланс між днями тижня та учасниками навчального процесу. У випадках змін у структурі предметів, замінах викладачів або оновлення навчального плану, система дозволяє швидко оновити параметри та згенерувати новий актуальний розклад без потреби у повному перерахунку вручну.

Автоматизована система вирізняється надзвичайно високим рівнем адаптивності та масштабованості, що дозволяє ефективно впроваджувати її в найрізноманітніших навчальних контекстах — від невеликих шкіл і коледжів до великих університетських комплексів, а також міжфакультетських програм із сотнями груп і викладачів. Гнучка архітектура платформи забезпечує безпроблемне масштабування без втрати продуктивності або точності алгоритмів. Система також підтримує модульну інтеграцію з іншими цифровими освітніми сервісами, включно з електронними журналами, системами керування навчанням (LMS), платформами віддаленого навчання, внутрішніми CRM і ERP-системами. Завдяки цьому відкриваються нові горизонти для комплексної цифровізації навчального процесу, уніфікації обліку й моніторингу, покращення прозорості управлінських рішень і підвищення загальної ефективності адміністрування закладу. Така інтеграція дозволяє автоматизувати не лише складання розкладів, а й значну частину супровідної документації та комунікації, що критично важливо в умовах сучасного освітнього середовища, орієнтованого на дані та цифрову трансформацію.

Автоматизована система складання розкладу на основі генетичних алгоритмів демонструє не лише суттєво вищу ефективність і точність у порівнянні з ручними методами, але й забезпечує повну адаптивність до динамічних змін

навчального процесу. Її логіка дозволяє автоматично враховувати десятки параметрів, від типу занять до навантаження викладачів, і реагувати на зміни в режимі реального часу. Важливо, що ця система не вимагає постійного втручання людини — усі процеси оптимізації й перевірки виконуються математично, що мінімізує людський фактор та підвищує достовірність кінцевого результату. Крім того, можливість масштабування дозволяє використовувати цю систему як для одного факультету, так і для всього університету або освітнього консорціуму. Інтеграція з іншими цифровими сервісами, такими як електронні журнали, системи керування навчанням (LMS) чи системи внутрішньої аналітики, відкриває шлях до створення повноцінного цифрового освітнього середовища нового покоління, де прийняття рішень ґрунтується на точних даних і алгоритмах оптимізації, а не на інтуїції чи ручних правках..

Висновок до розділу 3

У третьому розділі було детально розглянуто процес реалізації системи автоматичного складання розкладів у навчальних закладах на основі генетичних алгоритмів. Було описано використані технології, такі як Flask, Python, SQLite, HTML/CSS/JS, що забезпечили ефективну побудову веб-інтерфейсу та логіки обробки даних. Представлено візуальні приклади роботи системи, які підтверджують правильність реалізації, зручність користування та коректність відображення згенерованих розкладів.

Особливу увагу приділено порівнянню між ручним та автоматичним підходами до формування розкладів. У ході аналізу було доведено, що автоматизований підхід дозволяє суттєво зменшити трудові витрати, знизити ризик помилок, забезпечити масштабованість, гнучкість, адаптивність до змін, а також точне врахування множини обмежень, які виникають у реальних навчальних умовах. Система не лише виконує технічну функцію генерації розкладу, а й стає частиною ширшої цифрової екосистеми управління навчальним процесом.

Реалізована система демонструє високу ефективність і практичну цінність для закладів освіти, підтверджуючи доцільність використання методів штучного інтелекту, зокрема генетичних алгоритмів, для вирішення складних задач планування.

ВИСНОВКИ

У межах виконаної кваліфікаційної роботи було реалізовано інтелектуальну вебсистему для автоматизованого складання навчального розкладу в закладах освіти з використанням генетичних алгоритмів. Система дозволяє ефективно враховувати складну сукупність обмежень і вимог — від ресурсних до педагогічних та організаційних.

У результаті дослідження досягнуто наступних основних результатів:

- проведено детальний аналіз проблематики ручного формування розкладу та особливостей існуючих програмних рішень (FET, Asc Timetables, UniTime), що виявило потребу в більш гнучкому й адаптивному підході;
- досліджено принципи роботи генетичних алгоритмів, зокрема їх придатність до задачі багатофакторного планування;
- розроблено архітектуру клієнт-серверної системи з модульним розподілом — інтерфейс, API, база даних та обчислювальний модуль;
- реалізовано базу даних із підтримкою сутностей: групи, викладачі, предмети, аудиторії, типи занять та правила навантаження;
- впроваджено інтерфейс для адміністрування, редагування розкладу та запуску алгоритму з урахуванням заданих параметрів;
- проведено експериментальне тестування системи, яке показало зменшення конфліктів, рівномірний розподіл занять і значне прискорення процесу складання розкладу порівняно з ручним способом.

Результати підтвердили, що використання генетичного підходу забезпечує високу адаптивність системи до змін вхідних даних, масштабованість під різні типи навчальних закладів, а також суттєве зниження адміністративного навантаження.

Практичне значення роботи полягає в можливості впровадження системи в реальні умови освітнього середовища, де вона здатна покращити організацію навчального процесу, зробити його прозорим, ефективним і зручним для всіх учасників — студентів, викладачів та адміністрації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Голуб О.І. Штучний інтелект: навч. посібник. – Київ: КНЕУ, 2020. – 312 с.

2. Михайленко А.О. Генетичні алгоритми в задачах оптимізації. – Харків: ХНУРЕ, 2019. – 228 с.

3. Жуков І.М. Інформаційні технології в освіті: підручник. – Львів: Видавництво ЛНУ, 2021. – 356 с.

4. Ситник К.М. Інтелектуальні системи: теорія і практика. – Дніпро: ДНУ, 2020. – 290 с.

5. Попов В.В. Алгоритми та структури даних. – Київ: Наукова думка, 2018. – 276 с.

6. Давиденко С.О. Основи проектування інформаційних систем. – Київ: Каравела, 2021. – 198 с.

7. Ярошенко О.В. Системний аналіз і моделювання. – Харків: НТУ "ХПІ", 2020. – 312 с.

8. Goldberg D.E. Genetic Algorithms in Search, Optimization and Machine Learning. – Addison-Wesley, 1989. – 412 p.

9. Mitchell M. An Introduction to Genetic Algorithms. – MIT Press, 1998. – 226 p.

10. Haupt R.L., Haupt S.E. Practical Genetic Algorithms. – Wiley-Interscience, 2004. – 256 p.

11. Holland J.H. Adaptation in Natural and Artificial Systems. – MIT Press, 1992. – 208 p.

12. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – Pearson, 2020. – 1136 p.

13. Eiben A.E., Smith J.E. Introduction to Evolutionary Computing. – Springer, 2015. – 300 p.

14. Згурська О.П. Програмне забезпечення інформаційних систем. – Київ: КНУ, 2020. – 248 с.

15. FET – Free Timetabling Software. [Електронний ресурс]. – Режим доступу: https://timetabling.org

16. Asc Timetables. [Електронний ресурс]. – Режим доступу: https://www.asctimetables.com

17. UniTime – Comprehensive Academic Scheduling Solution. [Електронний ресурс]. – Режим доступу: https://unitime.org

18. Flask Web Framework. [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com

19. Python Programming Language. [Електронний ресурс]. – Режим доступу: https://www.python.org

20. ISO/IEC 25010:2011 Systems and Software Quality Requirements and Evaluation (SQuaRE). – International Organization for Standardization, 2011.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Автоматизація складання розкладів у навчальних закладах за допомогою генетичних алгоритмів

Виконав студент 4 курсу
групи ШІ-42
Козярик Максим
Керівник роботи
Викладач
Работа О.В

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи –розробка програмного засобу для автоматизованого складання розкладу занять у навчальному закладі з використанням генетичного алгоритму з урахуванням обмежень та оптимізаційних критеріїв.

Об'єкт дослідження – процес формування розкладу в закладах освіти в умовах множинних обмежень (кабінети, викладачі, групи, тип занять тощо).

Предмет дослідження – методи оптимізації, а саме — застосування генетичних алгоритмів у задачах автоматизованого планування навчального процесу.

АКТУАЛЬНІСТЬ ТЕМИ

1.Традиційне складання розкладів вручну є трудомістким, займає багато часу та часто призводить до конфліктів занять, неефективного використання ресурсів (аудиторій, викладачів).

2.Необхідність оптимізації — зростання кількості груп, викладачів та видів занять (лекції, лабораторні) унеможливлює ефективне ручне планування.

3.Сучасні освітні заклади потребують: автоматизації процесу формування розкладу; врахування численних обмежень та пріоритетів; підвищення гнучкості та точності розкладу.

4.Генетичні алгоритми є ефективним інструментом вирішення задач комбінаторної оптимізації, таких як розклад — дозволяють знаходити найбільш збалансовані рішення.

5.Розробка інтелектуальної системи формування розкладу на основі генетичних алгоритмів є актуальною, своєчасною та значущою для автоматизації освітнього процесу.

ЗАДАЧИ ДИПЛОМНОЇ РОБОТИ

1.Провести аналізна укових джерел щодо методів автоматизованого складання розкладів та можливостей використання генетичних алгоритмів у системах планування.

2.Дослідити наявні програмні рішення для генерації розкладів у закладах освіти, визначити їхні переваги та недоліки.

3.Вивчити принципи побудови генетичних алгоритмів, адаптацію операторів (відбір, кросовер, мутація) до задачі розкладу.

4.Сформулювати функціональні та нефункціональні вимоги до системи автоматизованого складання розкладу з урахуванням обмежень аудиторій, занять і викладачів.

5.Розробити архітектуру системи та реалізувати вебінтерфейс для: введення навчальних даних (групи, пари, викладачі); запуску алгоритму формування розкладу; редагування та збереження результатів.

6.Провести тестування системи, оцінити ефективність згенерованих розкладів та якість покриття обмежень, зіставивши з ручним плануванням.

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:

•Реєстрація та авторизація користувачів (адміністратор).

•Додавання даних про: групи, викладачів, кабінети, типи занять; зв’язки міжвикла дачами та предметами.

Інтерфейс для генераціїрозкладу на основіобранихпараметрів.

•Реалізаціягенетичного алгоритму з урахуванням: типу занять (лекція, лабораторна), доступностіресурсів (аудиторій, викладачів).

•Можливістьперегенераціїрозкладу при змінівхіднихданих.

•Збереженнярезультатів у БД.

•Панель адміністратора для керуваннякористувачами, кабінетами, предметами, групами, викладачами.

•Виведеннярозкладу у виглядітаблиці по групах, з можливістюредагування.

Нефункціональні вимоги:

•Підтримказахищенихсесій та обліковихзаписів.

•Збереженняданих у БД (наприклад, SQLite абоPostgreSQL).

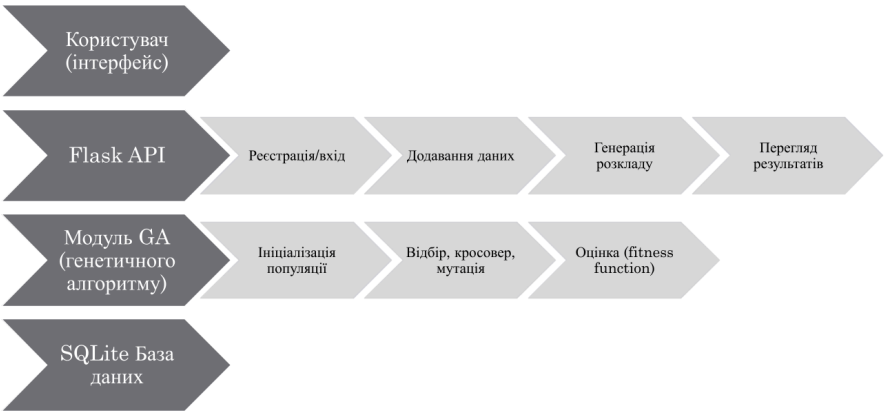
•Швидкереагування та оптимізація часу генерації.

•Зрозумілий та зручнийUX/UI інтерфейс — мінімумкліків, максимум наочності.

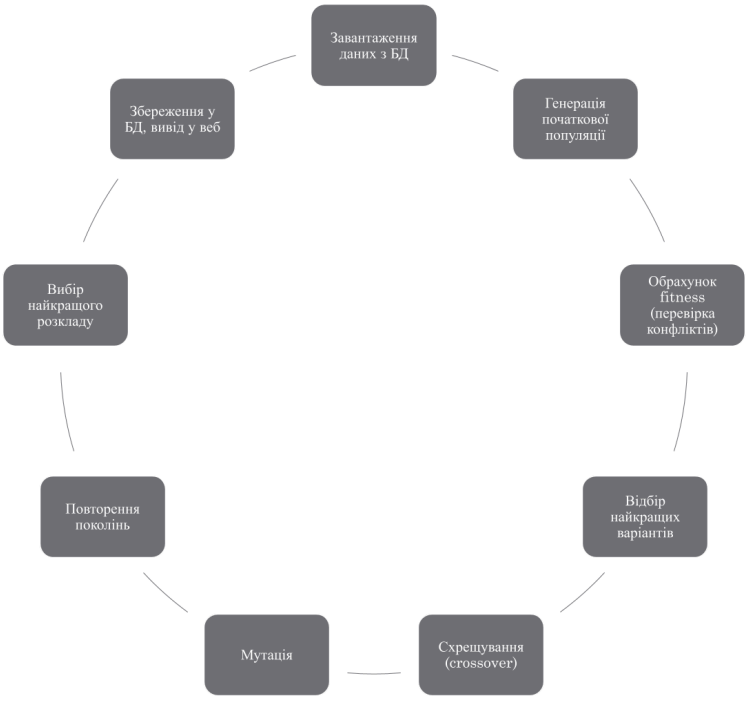
Технології, мови та бібліотеки

Мова програмування	Python	Основна логіка, реалізація алгоритму GA
Фреймворк	Flask	Веб-сервер для API та інтерфейсу
Бібліотека	NumPy, Pandas	Обробка масивів, структурування даних
Бібліотека	random, copy	Генерація популяцій, мутацій
Веб-інтерфейс	HTML/CSS, JS	Візуалізація, введення дани
БД	SQLite	Збереження розкладів, користувачів
UX/UI	Bootstrap	Сучасний зручний інтерфейс

Архітектура веб-застосунку



ЯК ПРАЦЮЄ ГЕНЕТИЧНИЙ АЛГОРИТМ



СТРУКТУРА БАЗИ ДАННИХ

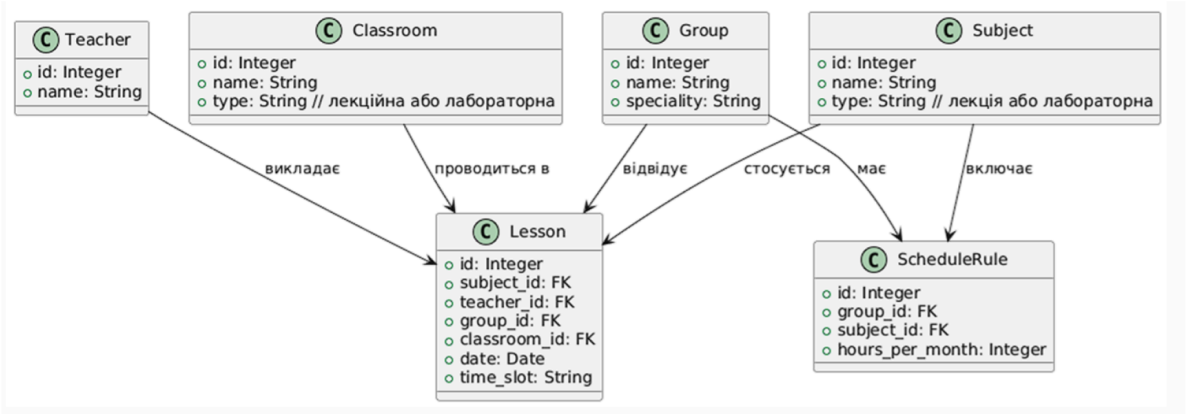


Рис. 0

ПОРІВНЯННЯ З РУЧНИМ ПЛАНУВАННЯМ

Критерій	Ручне планування	Генетичний алгоритм
Швидкість	Повільно	Миттєво
Конфлікти	Часті	Мінімізовані
Масштабованість	Обмежена	Висока
Гнучкість до змін	Низька	Висока
Людський фактор	Високий ризик	Відсутній

ПРИКЛАД ЗГЕНЕРОВАНОГО РОЗКЛАДУ

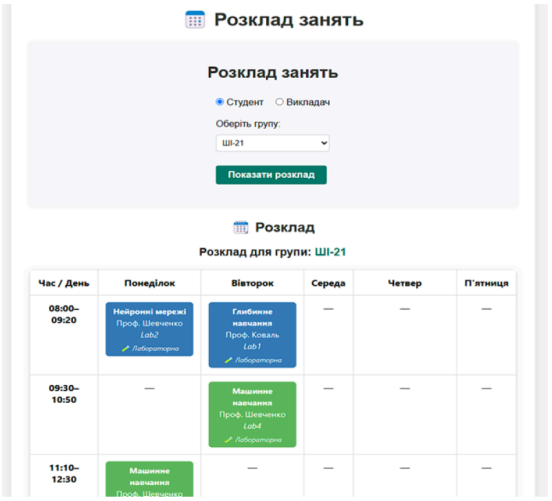


Рис. 1
Розклад студента

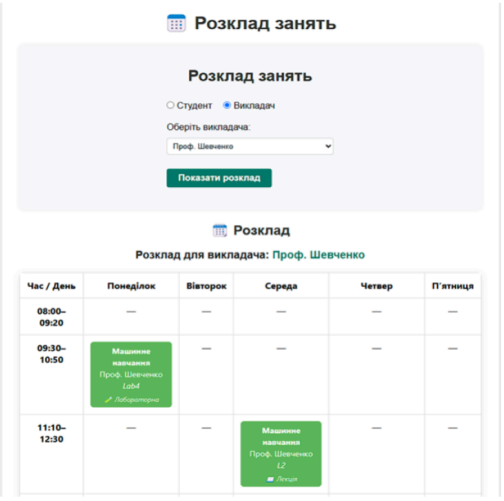


Рис. 2
Розклад викладача

ГОЛОВНА

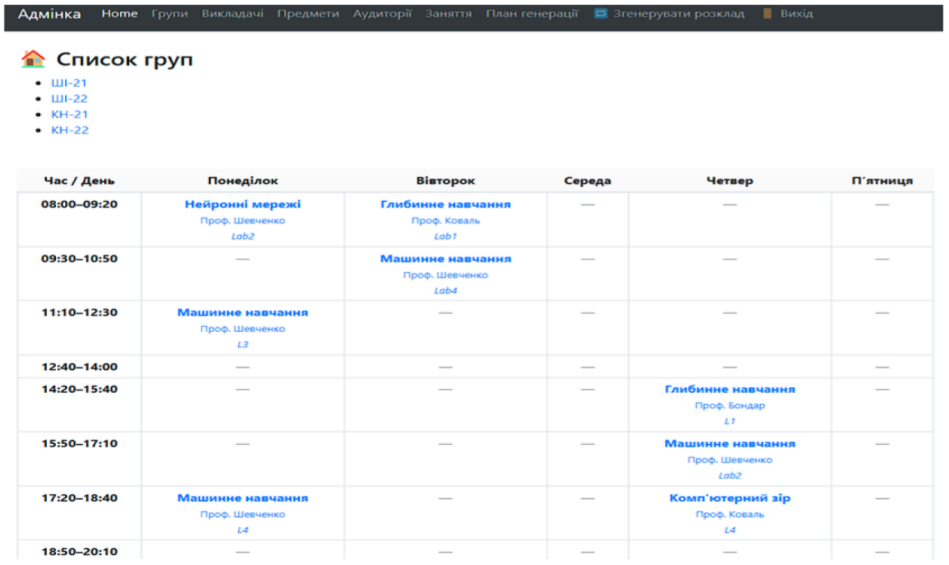


Рис. 3
Головна сторінка

СТОРІНКИ ІНФОРМАЦІЙ

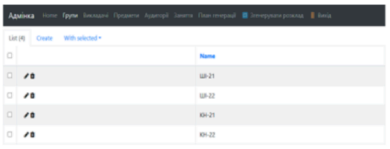


Рис. 4
Сторінка - Групи

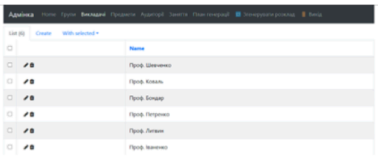


Рис. 5
Сторінка - Викладачі

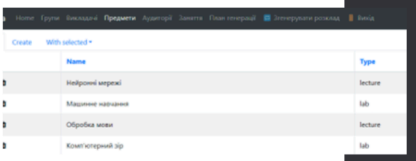


Рис. 6
Сторінка - Предмети

Рис. 7
Сторінка - Аудиторії

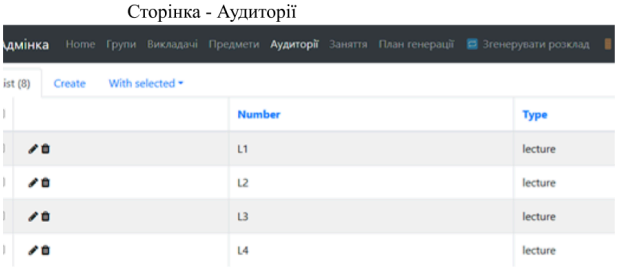
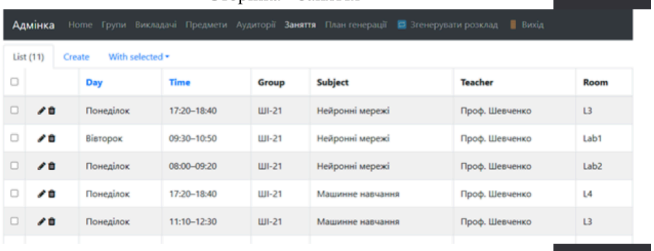


Рис. 8
Сторінка - Заняття



ПЛАН ГЕНЕРАЦІЇ ТА ГЕНЕРАЦІЯ

План генерації

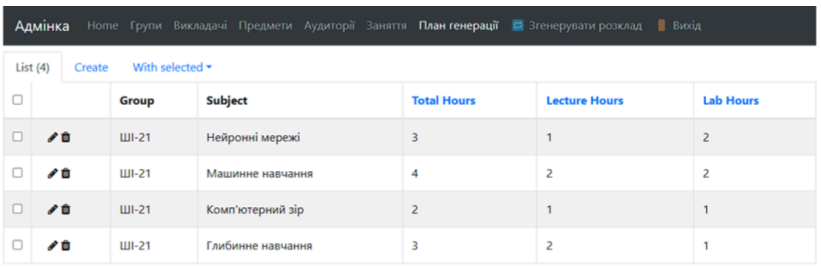
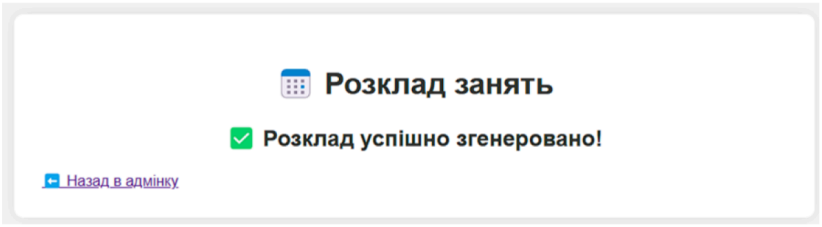


Рис. 9



Виконана генерація

Рис. 10

ВИГЛЯД РОЗКЛАДУ ДЛЯ ЮЗЕРА

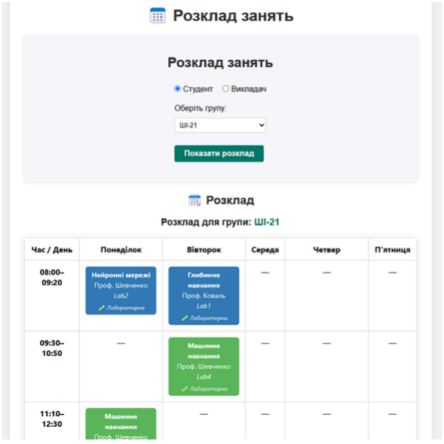


Рис. 11
Розклад студента

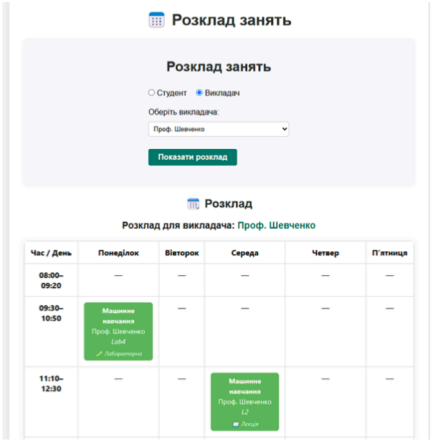


Рис. 12
Розклад викладача

ВИСНОВКИ

У результаті виконаної дипломної роботибуло реалізовано систему автоматизованого складання розкладу занять у навчальних закладах із використанням генетичного алгоритму. Проведено аналіз літературних джерел і наявних рішень, що дозволило підтвердити актуальність задачі розкладу в умовах складни хобмежень. На основі цього було спроєктовано архітектуру системи з розділенням на клієнтську та серверну частину, реалізовану засобами Python (Flask), HTML/CSS, JavaScript, а також базою даних SQLite.

Сформульовано та реалізовано функціональні й не функціональні вимоги, що забезпечили повноцінну роботу з розкладом: введення груп, викладачів, кабінетів і предметів, налаштування правил, запуск алгоритму та відображення результатів у вигляді зручного календаря. Генетичний алгоритм дозволив ефективно оптимізувати розклад, мінімізуючи конфлікти та дотримуючись встановлених обмежень.

Розроблена система підтвердила свою працездатність, стабільність та прикладну цінність. Її використання значно зменшує часові витрати адміністрації навчального закладу, покращує керованість навчальним процесом і є придатною до масштабування.