Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота
за освітнім ступенем “бакалавр”
на тему:

ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ ТАЙМ-МЕНЕДЖМЕНТУ ВИКЛАДАЧА ЗВО

Виконав:
здобувач IV курсу
групи КН-41
спеціальності 122 “Комп'ютерні науки”
Михайло Романович Мізь

Науковий керівник:
кандидат технічних наук,
доцент Степанія Михайлівна Бабич

Рівне – 2026

ЗМІСТ

СПИСОК ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

ЗВО – заклад вищої освіти.

ІС – інформаційна система.

СКБД – система керування базами даних.

CRUD – набір операцій створення, читання, оновлення та видалення даних.

CSV – текстовий формат табличного обміну даними.

HTTP – протокол передавання гіпертекстових документів.

UML – уніфікована мова моделювання програмних систем.

WCAG – настанови з доступності вебвмісту.

WAL – режим журналювання змін у SQLite.

ВСТУП

Інформаційна система підтримки тайм-менеджменту викладача ЗВО належить до прикладних програмних рішень, призначених для впорядкування робочого часу, задач, календарних подій, навчального навантаження та фактичних витрат часу на різні види професійної діяльності. Потреба у створенні такої системи пов’язана не лише з побутовою зручністю планування. Для викладача закладу вищої освіти час виступає ресурсом, який розподіляється між навчальною, методичною, науковою, організаційною, консультаційною та комунікаційною роботою. Закон України «Про вищу освіту» закріплює правові й організаційні засади діяльності закладів вищої освіти, а отже, робота науково-педагогічного працівника має розглядатися не як сукупність випадкових задач, а як регламентована професійна діяльність із різними видами навантаження [1]. Саме тому програмна система, орієнтована на викладача, повинна не просто зберігати перелік справ, а підтримувати зв’язок між дисциплінами, дедлайнами, календарними подіями, фактичними часовими інтервалами та аналітичними показниками.

Актуальність теми посилюється цифровізацією освітнього середовища. Міністерство освіти і науки України розглядає цифрову трансформацію освіти і науки як комплексну роботу над екосистемою цифрових рішень, безпечним електронним освітнім середовищем, цифровою інфраструктурою, автоматизацією збору й аналізу даних [4]. Для закладу вищої освіти така орієнтація означає поступовий перехід від розрізнених файлів, паперових планів, таблиць і неформальних нотаток до структурованих інформаційних середовищ, у яких дані можуть накопичуватися, перевірятися, узагальнюватися й використовуватися для управлінських або індивідуальних рішень. У контексті тайм-менеджменту викладача цифровізація не зводиться до перенесення паперового щоденника в електронну форму. Значно продуктивнішим є підхід, за якого календар, задачі, дисципліни й фактичний облік часу стають елементами єдиної бази даних.

Особливість предметної області полягає в тому, що викладач ЗВО працює не з однорідним набором справ. Одна частина роботи має жорстко визначений час, наприклад заняття, консультації, засідання кафедри, екзамени або дедлайни перевірки. Інша частина пов’язана з підготовкою матеріалів, науковими текстами, методичною роботою, рецензуванням, комунікацією зі здобувачами освіти, адміністративними діями. Такий розподіл створює потребу в системі, яка поєднує календарну модель часу, задачну модель роботи та фактичний журнал виконаних дій. Якщо використовувати лише календар, втрачається зміст задач і стан їх виконання. Якщо використовувати лише список справ, не видно розподілу часу. Якщо обмежитися журналом фактичних годин, планування перетворюється на післядію. Розроблена інформаційна система має об’єднати всі три підходи.

Об’єктом дослідження є процес організації, планування, обліку та аналітичного контролю робочого часу викладача закладу вищої освіти в умовах цифровізації освітньої діяльності.

Предметом дослідження є структура, функціональні можливості, база даних, програмна логіка та вебінтерфейс інформаційної системи підтримки тайм-менеджменту викладача ЗВО.

Метою роботи є проєктування, розроблення та перевірка працездатності інформаційної системи підтримки тайм-менеджменту викладача закладу вищої освіти, яка забезпечує ведення дисциплін, задач, календарних подій, фактичного обліку робочого часу, формування аналітичних показників, експорт даних і резервне копіювання в межах локального вебзастосунку.

Для досягнення поставленої мети необхідно розв’язати такі **завдання:**

1. Проаналізувати предметну область тайм-менеджменту викладача ЗВО та визначити основні види професійної діяльності, що потребують інформаційної підтримки.
2. Дослідити функціональні можливості сучасних програмних засобів календарного планування, керування задачами та організації робочих процесів.
3. Сформувати функціональні й нефункціональні вимоги до інформаційної системи підтримки тайм-менеджменту викладача ЗВО.

4. Обґрунтувати вибір архітектури локального вебзастосунку, бази даних, мови програмування та засобів реалізації користувацького інтерфейсу.

5. Спроєктувати структуру бази даних інформаційної системи, визначити її основні сутності, зв’язки, обмеження цілісності та логіку збереження даних.

6. Реалізувати основні програмні модулі системи: роботу з дисциплінами, задачами, календарними подіями, журналом часу, звітами, експортом і резервним копіюванням.

7. Перевірити працездатність розробленої інформаційної системи за основними сценаріями використання та оцінити її відповідність поставленим вимогам.

8. Визначити напрями подальшого розвитку системи з урахуванням можливості авторизації користувачів, імпорту розкладу, інтеграції із зовнішніми календарями та розширення аналітичного блоку.

Методи дослідження обрано відповідно до мети й завдань роботи. Метод аналізу використано для вивчення предметної області тайм-менеджменту викладача ЗВО та визначення складу даних, необхідних для планування і контролю робочого часу. Порівняльний метод застосовано для аналізу програмних аналогів, зокрема календарних сервісів, менеджерів задач і цифрових дошок планування. Метод інформаційного моделювання використано під час проєктування структури бази даних, визначення сутностей, зв’язків і обмежень цілісності. Методи структурного й об’єктно-орієнтованого моделювання застосовано для опису архітектури системи, сценаріїв використання та взаємодії її компонентів. Метод програмної реалізації використано під час створення локального вебзастосунку мовою Python із використанням бази даних SQLite. Метод функціонального тестування застосовано для перевірки коректності роботи основних модулів інформаційної системи.

Практична значущість роботи полягає в тому, що розроблена інформаційна система може бути використана як навчальний програмний продукт, прототип персонального цифрового інструмента викладача або основа

для подальшого розширення кафедральної системи планування й обліку робочого часу. Система забезпечує централізоване ведення дисциплін, задач, календарних подій і фактичних часових записів, що дозволяє викладачеві не лише планувати роботу, а й аналізувати реальний розподіл часу між різними видами професійної діяльності.

Апробація і впровадження результатів дослідження. Основні теоретичні та практичні результати дослідження доповідалися та обговорювалися на звітній науковій конференції викладачів, співробітників і здобувачів вищої освіти Рівненського державного гуманітарного університету за 2025 рік (м. Рівне, 14 травня 2026 року).

Структура роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та семи додатків. Загальний обсяг роботи становить 75 сторінок. Список використаних джерел включає 22 найменування.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Предметна область тайм-менеджменту викладача ЗВО як об’єкт інформаційного моделювання

Професійна діяльність викладача закладу вищої освіти має складну часову структуру, оскільки поєднує аудиторну роботу, підготовку навчальних матеріалів, перевірку робіт, наукову діяльність, методичне забезпечення дисциплін, організаційні доручення, консультації та комунікацію зі здобувачами освіти. З позиції інформаційного моделювання така діяльність не може бути зведена до простого списку подій. Вона потребує опису об’єктів, часових меж, станів виконання, зв’язків між дисциплінами й задачами, а також механізмів фіксації фактично витраченого часу. Закон України «Про вищу освіту» закріплює автономію закладів вищої освіти, особливий статус освітнього процесу та роль науково-педагогічних працівників, через що облік діяльності викладача має спиратися на чітке розмежування видів роботи [1]. Для програмної системи такий висновок має пряме значення: кожен вид роботи повинен отримати формалізоване представлення в базі даних.

Тайм-менеджмент у межах досліджуваної системи доцільно розуміти як упорядкований процес планування, розподілу, виконання й аналізу робочого часу викладача. Йдеться не про психологічну техніку особистої продуктивності, а про інформаційно підтриманий цикл роботи з даними. Спочатку викладач створює дисципліни, що формують контекст навчального навантаження. Далі вносить задачі, пов’язані з підготовкою занять, методичною роботою, перевіркою, науковими матеріалами або адміністративними діями. Паралельно календар відображає події з конкретними часовими межами. Після виконання роботи журнал часу фіксує фактичний часовий інтервал, вид активності, примітку та зв’язок із дисципліною або задачею. Саме така послідовність перетворює розрізнені дії на інформаційну модель професійної діяльності.

У межах предметної області виділяються чотири основні інформаційні контури. Перший контур стосується навчальних дисциплін, оскільки саме дисципліна виступає змістовим осередком значної частини роботи викладача. Другий контур охоплює задачі, які мають назву, опис, пріоритет, статус, дедлайн, тип активності й оцінену тривалість. Третій контур пов’язаний із календарними подіями, де визначальними є дата, час початку, час завершення, тип події, місце проведення та зв’язок із дисципліною. Четвертий контур формує журнал фактичного часу, без якого неможливо зіставити планове навантаження з реальною трудомісткістю. Зв’язок між наведеними контурами показано на рис. 1.1.

Рисунок 1.1. Концептуальна модель предметної області інформаційної системи підтримки тайм-менеджменту викладача ЗВО

Як показано на рис. 1.1, предметна область не обмежується окремим календарем або реєстром задач. Центральним елементом виступає викладач, який взаємодіє з дисциплінами, задачами, подіями та журналом часу. Аналітичні показники формуються не вручну, а через агрегування вже введених даних. Такий підхід зменшує дублювання, оскільки одна дисципліна може бути використана одночасно в задачах, календарних подіях і записах фактичного часу. Для програмної реалізації така модель означає потребу в реляційній базі даних, у якій сутності мають первинні ключі, зовнішні ключі й обмеження цілісності.

Внутрішня складність тайм-менеджменту викладача посилюється тим, що професійна діяльність має як плановий, так і фактичний вимір. Плановий вимір фіксується через задачі, дедлайни й календарні події. Фактичний вимір формується після виконання роботи й відображає реальні часові витрати. Саме розрив між плановим і фактичним виміром часто створює проблему перевантаження: викладач бачить наявні дедлайни, але не завжди бачить, яка частка часу вже витрачена на перевірку, підготовку, комунікацію або

адміністративні дії. Інформаційна система має надати користувачеві не лише інструмент введення даних, а й засіб інтерпретації власного навантаження.

Для формалізації предметної області в роботі доцільно використати такі базові поняття:

1. дисципліна – навчальний компонент, з яким пов’язуються задачі, події та записи часу;
2. задача – запланована дія викладача з визначеним статусом, пріоритетом, дедлайном і орієнтовною тривалістю;
3. календарна подія – часовий блок із датою, початком, завершенням, видом події та, за потреби, прив’язкою до дисципліни;
4. запис часу – фактичний інтервал виконаної роботи із зазначенням виду активності та кількості хвилин;
5. аналітичний показник – агреговане значення, обчислене на основі задач, подій або записів часу.

Наведений понятійний апарат задає сталі межі дослідження. Подальший текст використовує терміни «задача», «подія», «дисципліна», «запис часу» й «аналітичний показник» саме в зазначених значеннях, без довільної синонімії. Для інформаційної системи така сталість має методологічне значення, адже різні назви однієї сутності ускладнили б проєктування бази даних і користувацького інтерфейсу.

З погляду користувача, система повинна підтримувати щонайменше три рівні роботи. На першому рівні викладач вносить або редагує первинні дані. На другому рівні система структурує записи через фільтри, статуси, прив’язки до дисциплін і календарні періоди. На третьому рівні дані перетворюються на узагальнення: години за період, частка зосередженої роботи, виконання задач, прострочені дедлайни, структура часу за дисциплінами й видами активності. Такий трирівневий підхід відрізняє повноцінну інформаційну систему від звичайної електронної таблиці, де користувач здебільшого сам створює формули, сортування й логіку обчислень.

Відповідно до ECTS Users’ Guide навчальне навантаження пов’язується з очікуваними результатами навчання та обсягом роботи, потрібним для їх

досягнення [5]. Для теми розроблення системи підтримки тайм-менеджменту викладача такий підхід корисний не через прямий розрахунок студентських кредитів, а через загальну ідею вимірюваності навантаження. Якщо освітній процес оцінюється не лише через перелік дисциплін, а й через обсяг роботи, то персональна інформаційна система викладача має давати змогу бачити часову структуру власної участі в такому процесі. Вона не замінює офіційний облік навантаження в закладі освіти, але може виступати інструментом персонального планування та самоконтролю.

Особливої уваги потребує локальний характер системи. На відміну від хмарних сервісів, локальний вебінтерфейс із базою SQLite не потребує реєстрації, мережевої інфраструктури або зовнішнього сховища. Такий варіант не претендує на корпоративний рівень масштабування, але добре відповідає вимогам навчального програмного проєкту: система легко запускається, демонструється, переноситься разом із файлом бази даних і не створює залежності від стороннього постачальника. Офіційна документація SQLite визначає SQLite як окремий рушій бази даних із розвиненою документацією, підтримкою зовнішніх ключів і файловою природою зберігання даних [10]. Для персональної системи викладача така характеристика є достатньою, оскільки кількість записів, очікувана в локальному застосунку, не вимагає окремого серверного комплексу.

Підсумовуючи підрозділ, предметна область тайм-менеджменту викладача ЗВО має багатокомпонентний характер і потребує реляційного представлення даних. Аналіз показує, що основними сутностями системи мають бути дисципліни, задачі, календарні події, записи фактичного часу й налаштування. Саме такий склад даних забезпечує перехід від фрагментарного планування до цілісної інформаційної підтримки роботи викладача.

1.2 Аналіз аналогів програмних рішень для планування часу, задач і календарних подій

Розроблення вебінтерфейсу бази даних для тайм-менеджменту викладача ЗВО потребує аналізу наявних програмних рішень, оскільки саме порівняння з аналогами дає змогу точніше визначити функції, які мають бути реалізовані в авторській системі. Для порівняння обрано Google Calendar, Todoist, Trello та Microsoft To Do. Такий вибір пояснюється тим, що наведені сервіси представляють різні підходи до організації роботи: календарне планування, задачне керування, дошкову модель процесів і персональні списки справ. Кожен із них має сильні сторони, але жоден не орієнтований спеціально на локальну базу даних викладача ЗВО з урахуванням дисциплін, видів педагогічної роботи та фактичного обліку часу.

Google Calendar є типовим представником календарної моделі планування. Офіційна довідка Google Calendar описує механізм створення розкладів запису на зустрічі, де користувач формує часові слоти, назву, параметри доступності та сторінку бронювання [6]. Для викладача така логіка корисна під час планування консультацій, занять, нарад і зустрічей зі здобувачами освіти. Сильна сторона календарної моделі полягає у візуалізації часу, але календар сам по собі не розкриває повний стан задач. Наприклад, подія «перевірка робіт» може стояти в календарі, однак без окремої задачі важко відобразити її статус, пріоритет, опис, пов’язану дисципліну та фактичну трудомісткість.

Todoist репрезентує задачну модель. Офіційна довідка Todoist описує використання пріоритетів для задач, зокрема швидке призначення рівнів пріоритетності через позначення p1, p2 або p3 [2]. Для персонального планування така функція зручна, бо дає змогу відрізняти невідкладні задачі від другорядних. Проте задачна система без глибшого предметного контексту не розв’язує проблему прив’язки до навчальних дисциплін і видів роботи викладача. У викладацькій діяльності пріоритет задачі має співвідноситися не тільки з датою, а й із типом роботи: методична підготовка, перевірка, консультації, науковий текст, організаційне доручення. Через це авторська система має зберігати не лише назву задачі, а й тип активності, орієнтовну тривалість, статус, дедлайн і зв’язок із дисципліною.

Trello використовує дошкову модель із картками, списками та дошками. Офіційна сторінка шаблонів Trello описує організацію та пріоритезацію проєктів через дошки, списки й картки [3]. Такий підхід добре підходить для командної роботи, руху задач між станами та наочного представлення процесу. Для тайм-менеджменту викладача дошкова модель може бути корисною як відображення статусів «черга», «у роботі», «виконано». Водночас Trello не є спеціалізованою системою фактичного обліку часу викладача й не містить власної предметної моделі дисциплін, навчального навантаження та видів педагогічної діяльності. Отже, авторський вебінтерфейс може запозичити ідею статусного руху задач, але повинен доповнити її реляційною структурою даних і звітами.

Microsoft To Do орієнтований на персональні списки справ і щоденне планування. Офіційна сторінка Microsoft To Do описує функцію «My Day» як персоналізований щоденний планувальник із пропозиціями задач [7]. Для користувача така модель зручна через простоту й низький поріг входу. Проте простий список справ не забезпечує системного аналізу витрат часу, структури навантаження за дисциплінами й обліку робочих інтервалів. Для викладача ЗВО такий інструмент може бути допоміжним, але він не замінює спеціалізовану інформаційну систему.

Порівняльну характеристику аналогів подано в табл. 1.1. Таблиця не дублює опис сервісів, а зіставляє їх саме за критеріями, релевантними для теми розроблення вебінтерфейсу бази даних.

Таблиця 1.1

Порівняння аналогів інформаційної системи підтримки тайм-менеджменту викладача ЗВО

Критерій	Google Calendar	Todoist	Trello	Microsoft To Do	Авторська система TimeTeach Pro
Календарні події	реалізовано	обмежено	через картки або інтеграції	обмежено	реалізовано
Задачі з пріоритетами	частково	реалізовано	реалізовано	реалізовано	реалізовано
Статуси виконання	обмежено	реалізовано	реалізовано	частково	реалізовано
Прив’язка до	відсутня	відсутня	відсутня	відсутня	реалізован

дисциплін ЗВО					о
Фактичний облік часу	відсутній або через інтеграції	обмежено	через розширення	відсутній	реалізовано
Локальна база даних	відсутня	відсутня	відсутня	відсутня	реалізовано
Аналітика навантаження викладача	обмежено	обмежено	залежить від налаштувань	обмежено	реалізовано
Запуск однією командою	не застосовується	не застосовується	не застосовується	не застосовується	реалізовано
Робота без облікового запису	ні	ні	ні	ні	так
Резервне копіювання локальної БД	не застосовується	не застосовується	не застосовується	не застосовується	реалізовано

Як показано в табл. 1.1, наявні сервіси добре розв’язують окремі задачі планування, але не створюють цілісної предметної моделі роботи викладача ЗВО. Google Calendar має сильну календарну логіку, Todoist ефективно підтримує задачі та пріоритети, Trello забезпечує візуальне керування станами, Microsoft To Do зручний для персональних списків. Авторська система відрізняється іншим призначенням: вона не конкурує з великими хмарними платформами, а реалізує локальний вебінтерфейс до бази даних, де дисципліни, задачі, події та фактичний час зберігаються в єдиній структурі.

Аналіз аналогів показує, що для розроблюваної системи доцільно прийняти комбіновану логіку. Від календарних сервісів варто взяти подієву модель із датою, часом початку й завершення. Від задачних сервісів – статуси, пріоритети й дедлайни. Від дошкових інструментів – ідею руху задач між станами. Від персональних планувальників – простоту взаємодії та швидке внесення записів. Проте предметне ядро системи має бути власним, оскільки дисципліна, тип викладацької активності та журнал фактичного часу не є другорядними полями, а формують основу моделі даних.

Рисунок 1.2. Логіка відмінності авторської системи від універсальних планувальників

Як показано на рис. 1.2, авторська система не є механічним повторенням Google Calendar, Todoist, Trello або Microsoft To Do. Її логіка полягає в предметному поєднанні вже відомих підходів. Календарна частина відповідає за часову прив’язку, задачна частина – за планування дій, журнал часу – за фактичну трудомісткість, дисципліни – за освітній контекст, а звіти – за інтерпретацію накопичених даних. Саме така комбінація створює прикладну цінність розробленого рішення.

Під час аналізу аналогів не виявлено відкритого універсального сервісу, який у типовій конфігурації одночасно поєднував би локальну SQLite-базу, запуск однією командою, предметну прив’язку до дисциплін ЗВО, задачі, календар, журнал часу, звіти й резервне копіювання. Через відсутність підтверджених джерел твердження про повну відсутність таких систем на ринку потребує окремої патентної або продуктової перевірки. У межах навчального проєкту достатньо обґрунтовано встановити, що найпоширеніші аналоги мають іншу логіку використання й не закривають повністю предметні вимоги, сформовані для викладача ЗВО.

Підсумок підрозділу полягає в тому, що розроблювана система має спиратися на перевірені підходи до планування, але не повторювати універсальні сервіси. Її функціональна перевага повинна полягати в локальності, предметній структурі бази даних, зв’язку задач із дисциплінами, наявності журналу часу та формуванні аналітичних показників навантаження.

1.3 Формування функціональних і нефункціональних вимог до інформаційної системи

Вимоги до інформаційної системи підтримки тайм-менеджменту викладача ЗВО мають випливати з предметної області й аналізу аналогів. Якщо система створюється не як загальний планувальник, а як вебінтерфейс до бази даних, головним критерієм стає узгодженість функцій із моделлю даних. Кожна сторінка інтерфейсу повинна відповідати окремому інформаційному контуру: дашборд, задачі, календар, облік часу, дисципліни, звіти й налаштування. Така структура зменшує хаотичність користування й дає можливість швидко перевірити, які саме програмні модулі реалізовані.

Функціональні вимоги визначають дії, які користувач має виконувати в системі. Для розроблюваного вебінтерфейсу доцільно сформувати такі групи функцій:

1. ведення реєстру дисциплін із кодом, назвою, семестром, кількістю кредитів ECTS, тижневими годинами та ознакою активності;
2. створення, редагування, фільтрація, зміна статусу й видалення задач;
3. планування календарних подій із визначенням дати, часу початку, часу завершення, виду події, місця та зв’язку з дисципліною;
4. фіксація фактичних записів часу з видом активності, часовим інтервалом, кількістю хвилин, приміткою й прив’язкою до дисципліни або задачі;
5. формування дашборду з ключовими показниками за період;
6. побудова звітів за робочими годинами, дисциплінами, видами активності й дедлайнами;
7. експорт задач і записів часу у форматі CSV;
8. збереження налаштувань викладача та створення резервної копії бази даних.

Наведений перелік не є формальною декларацією, оскільки кожна функція має пряме відображення у структурі реалізованої системи. Реєстр дисциплін відповідає таблиці courses, задачі – таблиці tasks, календар – таблиці events, журнал часу – таблиці timelogs, налаштування – таблиці kv_settings. Через такий зв’язок вимоги можна перевірити не лише візуально, а й на рівні бази даних.

Нефункціональні вимоги визначають якісні характеристики програмного продукту. Для навчального проєкту з програмування вони мають особливу вагу,

оскільки працездатність системи залежить не тільки від кількості сторінок, а й від простоти запуску, стійкості до помилок введення, читабельності коду, цілісності даних і зручності інтерфейсу. Офіційна документація Python попереджає, що модуль http.server не призначений для промислового використання й реалізує лише базові перевірки безпеки [8]. Таке застереження не заперечує застосування модуля в навчальному локальному проєкті, але обмежує сферу використання системи: вона має розглядатися як локальний вебзастосунок, а не як відкритий мережевий сервіс.

Таблиця 1.2

Функціональні та нефункціональні вимоги до інформаційної системи

Група вимог	Зміст вимоги	Реалізаційне значення
Функціональність	ведення дисциплін, задач, подій, журналу часу, звітів і налаштувань	визначає склад сторінок і таблиць БД
Цілісність даних	використання первинних і зовнішніх ключів, обмежень і коректних типів полів	зменшує ризик втрати зв’язків між сутностями
Простота запуску	запуск системи однією командою з терміналу	спрощує перевірку й демонстрацію
Автономність	робота без зовнішнього сервера, реєстрації та мережевих залежностей	забезпечує переносимість проєкту
Зручність інтерфейсу	зрозуміла навігація, картки показників, форми введення, фільтри	підвищує практичну придатність
Доступність	читабельні підписи, логічна структура форм, достатня контрастність	узгоджується з підходами WCAG
Експортованість	CSV-експорт задач і записів часу	дає змогу використовувати дані поза системою
Відновлюваність	створення резервної копії SQLite-файлу	зменшує ризик втрати інформації
Перевірюваність	наявність тесту доступності основних таблиць і сервісних функцій	забезпечує контроль працездатності

Як показано в табл. 1.2, вимоги охоплюють не лише функції інтерфейсу, а й умови стабільної експлуатації. Для локальної системи автономність і простота запуску мають не менше значення, ніж кількість модулів. Якщо програмний продукт потребує складної інсталяції, зовнішніх служб або облікових записів, його перевірка в навчальному середовищі ускладнюється. Обраний підхід з Python, SQLite, HTML, CSS і JavaScript без зовнішніх мережевих бібліотек дає змогу уникнути такої проблеми.

Під час формування вимог до інтерфейсу враховано принципи доступності вебвмісту. WCAG 2.2 охоплює рекомендації, спрямовані на підвищення доступності вебвмісту для різних груп користувачів [15]. Для розроблюваної системи такі принципи застосовуються на прикладному рівні: меню має бути стабільним, поля форм повинні мати зрозумілі підписи, таблиці – читабельні заголовки, а ключові показники – однозначне пояснення. Оскільки система орієнтована на локальне використання, повна сертифікація доступності не проводиться, проте базові вимоги до зрозумілості й читабельності інтерфейсу враховано.

Питання безпеки розглядається з урахуванням локального характеру системи. OWASP Top 10 визначає перелік найбільш критичних ризиків для вебзастосунків, серед яких порушення контролю доступу, ін’єкції, помилки конфігурації безпеки й небезпечний дизайн. Розроблена система не публікується у відкритому інтернеті й не має багатокористувацької авторизації, але принцип обережної роботи з даними все одно залишається релевантним. Зокрема, запити до бази даних мають виконуватися через параметризовані SQL-запити, а шляхи до статичних файлів повинні перевірятися, щоб уникнути небажаного доступу до файлів поза каталогом застосунку.

Архітектурна вимога до системи полягає в тому, щоб забезпечити мінімальний технологічний бар’єр для запуску. У навчальному проєкті надмірний стек часто створює ризик, коли сама система формально написана, але не запускається на комп’ютері перевіряча. Тому використання стандартної бібліотеки Python має прагматичне обґрунтування. Модуль sqlite3 входить до стандартної бібліотеки Python і забезпечує DB-API 2.0 інтерфейс до SQLite-баз даних [9]. Такий вибір скорочує кількість зовнішніх залежностей, робить проєкт переносимим і дає змогу запускати систему командою python run.py або python3 run.py.

Рисунок 1.3. Діаграма варіантів використання інформаційної системи

Відповідно до рис. 1.3, користувачем системи виступає викладач ЗВО, а головними варіантами використання є перегляд дашборду, ведення дисциплін, керування задачами, планування подій, облік часу, перегляд звітів, експорт, резервне копіювання й налаштування. У діаграмі спеціально показано залежність задач, подій і записів часу від дисциплін. Такий зв'язок відображає предметну специфіку системи й відрізняє її від універсальних планувальників.

До даних системи висуваються окремі вимоги. Вони повинні бути достатньо простими для ручного введення, але достатньо структурованими для звітності. Наприклад, поле «вид активності» не повинно бути довільним у кожному записі, оскільки тоді аналітична діаграма за видами роботи втратить точність. Водночас надмірно жорсткий перелік видів роботи може ускладнити практичне використання. У реалізованій системі застосовано компромісний підхід: основні види активності подані фіксованим переліком, що охоплює підготовку занять, проведення занять, перевірку робіт, консультації, наукову роботу, методичну роботу, адміністративну роботу, комунікацію та підвищення кваліфікації.

Окрему роль відіграють аналітичні показники. Вони повинні бути зрозумілими, легко обчислюваними й придатними для інтерпретації без статистичної підготовки. Для дашборду обрано шість показників: години за 7 днів, частка зосередженої роботи, виконання задач, кількість прострочених задач, дедлайни на 7 днів і кількість активних дисциплін. Показник частки зосередженої роботи обчислюється як відношення часу, витраченого на підготовку занять, наукову, методичну роботу, перевірку й написання матеріалів, до загального зафіксованого часу. Такий показник не є універсальним науковим індексом, тому в роботі він трактується як внутрішній операційний показник системи, а не як зовнішньо стандартизована метрика.

Таблиця 1.3

Основні аналітичні показники системи TimeTeach Pro

Показник	Джерело даних	Формула або логіка обчислення	Практичне значення
Години за 7 днів	timelogs	сума хвилин за 7 днів / 60	показує фактично зафіксоване навантаження

Частка зосередженої роботи	timelogs	час фокусних активностей / загальний час × 100 %	відображає співвідношення змістовної роботи й загального навантаження
Виконання задач	tasks	виконані задачі / усі задачі × 100 %	показує стан завершення запланованих дій
Прострочені задачі	tasks	кількість невиконаних задач із минулим дедлайном	сигналізує про ризик порушення строків
Дедлайни 7 днів	tasks	кількість задач із дедлайном у найближчі 7 днів	допомагає оцінити короткострокове навантаження
Активні дисципліни	courses	кількість дисциплін з ознакою активності	характеризує освітній контекст роботи

Як показано в табл. 1.3, аналітичні показники системи побудовано на даних, які вже вводяться користувачем у процесі роботи. Завдяки цьому звітність не потребує окремого дублювання інформації. Якщо викладач систематично вносить задачі та записи часу, дашборд автоматично відображає стан навантаження. Якщо дані відсутні, система чесно показує порожні графіки або нульові значення, не створюючи штучних висновків.

У межах першого розділу також треба визначити межі програмного рішення. Система TimeTeach Pro є локальним вебзастосунком, а не корпоративною платформою управління персоналом ЗВО. Вона не реалізує багатокористувацьку авторизацію, електронний документообіг, інтеграцію з Moodle, Google Calendar або внутрішніми університетськими системами. Такі функції розглядаються як напрями розвитку, а не як обов’язкова частина першої версії. Подібне розмежування запобігає завищенню заявлених можливостей і зберігає відповідність між метою, реалізацією та висновками.

Підсумовуючи підрозділ, функціональні вимоги до системи охоплюють ведення дисциплін, задач, подій, журналу часу, звітів, експорту й резервного копіювання. Нефункціональні вимоги зосереджені на автономності, простому запуску, цілісності даних, зрозумілому інтерфейсі, базовій доступності й перевірюваності. Сформований набір вимог створює основу для другого розділу, у якому буде описано архітектуру, базу даних і програмну реалізацію.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Обґрунтування архітектури програмного рішення

Проєктування вебінтерфейсу бази даних інформаційної системи підтримки тайм-менеджменту викладача ЗВО потребує такого архітектурного рішення, яке поєднує простоту запуску, зрозумілу структуру коду, достатню функціональність і технічну придатність для демонстрації в навчальному середовищі. Для розробленої системи обрано локальну клієнт-серверну архітектуру, у якій серверна частина працює на комп'ютері користувача, база даних зберігається у файлі SQLite, а взаємодія з користувачем відбувається через браузер. Така модель відрізняється від звичайної настільної програми тим, що інтерфейс формується засобами HTML, CSS і JavaScript, але не потребує окремого віддаленого сервера, хмарного облікового запису або складного розгортання.

Вибір локальної архітектури має пряме методичне обґрунтування. Робота з програмування повинна демонструвати не лише ідею системи, а й працездатний програмний продукт, який запускається перевіряльником без додаткових інфраструктурних умов. Через таку вимогу недоцільно будувати першу версію системи на надмірно складному стеку з контейнеризацією, окремою серверною СКБД, середовищем Node.js або зовнішніми бібліотеками, які можуть не встановитися на іншому комп'ютері. Локальний вебзастосунок із файлом SQLite забезпечує відтворюваність: каталог проєкту містить код, статичні ресурси, базу даних, тест і технічну довідку, а запуск виконується однією командою з терміналу.

Архітектура системи TimeTeach Pro побудована за принципом розмежування відповідальності між модулями. Файл run.py відповідає за запуск, перевірку доступності порту, ініціалізацію бази даних і відкриття браузера. Модуль database.py інкапсулює підключення до SQLite, створення таблиць і

резервне копіювання. Модуль services.py містить прикладну логіку роботи з дисциплінами, задачами, подіями, журналом часу, звітами, CSV-експортом і демонстраційними даними. Модуль server.py реалізує маршрути HTTP-запитів, обробку форм, повернення HTML-сторінок, JSON-відповідей, CSV-файлів і статичних ресурсів. Модуль views.py формує HTML-інтерфейс сторінок, тоді як файли styles.css і app.js відповідають за візуальний стиль і побудову графіків на стороні браузера.

У загальному вигляді архітектурну схему програмного рішення подано на рис. 2.1. Вона показує, що браузер не взаємодіє з базою даних напряму. Усі запити проходять через локальний HTTP-сервер, який звертається до сервісного шару, а сервісний шар працює з SQLite через модуль підключення до бази. Завдяки такому поділу інтерфейс не містить SQL-запитів, а база даних не залежить від способу візуального подання інформації.

Рисунок 2.1. Архітектура локального вебзастосунку TimeTeach Pro

Як показано на рис. 2.1, система має локальну багаторівневу організацію, але не потребує розгортання окремого серверного середовища. Користувач бачить звичайний вебінтерфейс, проте всі дані залишаються у файлі data/timeteach.db. Такий підхід зручний для навчальної демонстрації, оскільки перевіряльник може відкрити папку проєкту, запустити файл run.py, внести дані, переглянути таблиці, графіки й експорт, а за потреби – перевірити структуру SQLite-файлу будь-яким стандартним переглядачем баз даних.

Окремої уваги потребує вибір модуля http.server. Офіційна документація Python прямо вказує, що http.server не рекомендований для виробничого використання, оскільки реалізує лише базові перевірки безпеки [8]. У межах розробленої системи таке обмеження враховано через локальний сценарій застосування: сервер запускається на адресі 127.0.0.1, не призначається для публічного доступу, не обробляє зовнішні облікові записи й не виконує роль відкритого вебсервісу. Така позиція не приховує технічних меж рішення, а коректно визначає його призначення. Для навчального проєкту локальний сервер

є достатнім, оскільки забезпечує демонстрацію HTTP-маршрутизації, форм, JSON-відповідей, CSV-експорту й роботи з базою даних.

Обґрунтування архітектури також пов’язане з вимогою мінімізації залежностей. Більшість функцій реалізовано через стандартну бібліотеку Python: argparse використано для параметрів запуску, socket – для перевірки доступності порту, threading і webbrowser – для автоматичного відкриття браузера, sqlite3 – для роботи з базою, csv – для експорту, json – для API-відповідей, http.server – для локального сервера. Такий склад підвищує переносимість, бо користувач не встановлює окремі пакети через pip, а працює з базовою інсталяцією Python.

Структуру каталогів і файлів системи подано в табл. 2.1. Таблиця потрібна не лише для опису складу проєкту, а й для демонстрації логіки розмежування програмних компонентів.

Таблиця 2.1

Структура програмного проєкту TimeTeach Pro

Елемент проєкту	Призначення	Роль у системі
run.py	запуск локального сервера, перевірка порту, відкриття браузера	вхідна точка системи
app/database.py	підключення до SQLite, створення таблиць, резервне копіювання	рівень доступу до даних
app/services.py	прикладна логіка дисциплін, задач, подій, часу, звітів	сервісний рівень
app/server.py	маршрути GET і POST, HTML, JSON, CSV, статичні файли	HTTP-рівень
app/views.py	генерація HTML-сторінок	рівень представлення
app/static/styles.css	візуальне оформлення інтерфейсу	стильовий рівень
app/static/app.js	графіки, запити до API, клієнтська взаємодія	клієнтська логіка
data/timeteach.db	файл бази даних SQLite	сховище інформації
tests/smoke_test.py	базова перевірка працездатності	контроль запуску й схеми
docs/TECHNICAL_NOTE.md	технічна довідка	супровідна документація

Як показано в табл. 2.1, проєкт має компактну, але не примітивну структуру. Кожен файл виконує окрему функцію, тому програмний код легше перевіряти, описувати й доповнювати. Рішення не зведене до одного великого файла, у якому змішано HTML, SQL і логіку обробки форм. Навпаки, система

має розподілену будову, що відповідає базовим принципам програмної інженерії: окремий рівень запуску, окремий рівень роботи з даними, окремий рівень сервісів, окремий рівень маршрутизації та окреме представлення.

Підсумовуючи підрозділ, архітектура TimeTeach Pro обрана з урахуванням трьох вимог: система має запускатися однією командою, зберігати дані локально й мати повноцінний вебінтерфейс. Локальний сервер Python, файлова база SQLite і власні HTML/CSS/JavaScript-ресурси забезпечують технічну автономність, тоді як модульна структура коду створює основу для подальшого розвитку системи.

2.2 Проєктування структури бази даних

База даних виступає ядром інформаційної системи, оскільки саме вона забезпечує збереження дисциплін, задач, календарних подій, фактичних записів часу й налаштувань. У межах розробленого проєкту використано реляційну модель, де кожна основна сутність предметної області має окрему таблицю, а зв’язки між сутностями реалізовано через зовнішні ключі. Такий підхід відповідає логіці інформаційного моделювання: дисципліна може бути пов’язана з багатьма задачами, багатьма календарними подіями й багатьма записами часу, тоді як окрема задача може мати багато фактичних записів виконання.

Вибір SQLite як СКБД обґрунтовується локальним характером системи. SQLite не потребує окремого серверного процесу, зберігає дані у файлі й добре підходить для автономних застосунків, прототипів, навчальних систем і персональних інструментів. Офіційна документація SQLite містить окремі матеріали щодо зовнішніх ключів і WAL-журналювання, що дає змогу підтримувати цілісність зв’язків і підвищувати надійність роботи з файлом бази [10]. Python, зі свого боку, має стандартний модуль sqlite3, який забезпечує інтерфейс DB-API 2.0 до SQLite-баз [9]. У розробленій системі така комбінація дала змогу відмовитися від окремого встановлення СКБД і зберегти повний функціонал CRUD-операцій.

У структурі бази даних передбачено п’ять основних таблиць: courses, tasks, events, timelogs, kv_settings. Таблиця courses зберігає навчальні дисципліни, що формують освітній контекст роботи викладача. Таблиця tasks фіксує заплановані дії з пріоритетами, статусами, дедлайнами й орієнтовною тривалістю. Таблиця events описує календарні події з часовими межами. Таблиця timelogs містить фактичні записи часу, які можуть бути пов’язані з дисципліною або конкретною задачею. Таблиця kv_settings реалізує просту модель системних налаштувань у форматі «ключ – значення».

Логіку зв’язків між таблицями бази даних подано на рис. 2.2.

Рисунок 2.2. Логічна структура бази даних інформаційної системи TimeTeach Pro

Як показано на рис. 2.2, таблиця courses має центральне значення для предметної області, оскільки пов’язується із задачами, подіями та записами часу. Така структура дає змогу визначати навантаження не лише за датами або видами активності, а й за конкретними дисциплінами. Таблиця tasks додатково пов’язується з timelogs, що дає можливість зіставляти заплановану задачу з фактичним виконанням. Таблиця kv_settings не має зовнішніх зв’язків, бо зберігає службові параметри системи: ім’я викладача, назву установи, початок і кінець робочого дня, тижневу норму годин, щільність інтерфейсу.

Склад полів основних таблиць подано в табл. 2.2. Таблиця потрібна для документування бази даних і може бути використана в додатках або технічному описі системи.

Таблиця 2.2

Характеристика таблиць бази даних TimeTeach Pro

Таблиця	Основні поля	Призначення
courses	id, code, name, semester, ects, weekly_hours, active	збереження навчальних дисциплін, семестру, кредитів і ознаки активності
tasks	id, title, description, due_date, priority, status, activity_type, est_minutes, created_at, course_id	збереження задач викладача з дедлайнами, пріоритетами, статусами та зв’язком із дисципліною

events	id, title, start_dt, end_dt, kind, location, notes, course_id	збереження календарних подій, занять, консультацій, нарад і дедлайнів
timelogs	id, log_date, start_time, end_time, minutes, activity_type, notes, course_id, task_id	фактичний облік часу з прив'язкою до дисципліни або задачі
kv_settings	key, value	збереження налаштувань системи у форматі «ключ – значення»

Як показано в табл. 2.2, структура бази даних не містить надлишкових таблиць, але охоплює всі сутності, потрібні для повного функціонування системи. Окремі поля, такі як priority, status, activity_type, мають текстовий тип, оскільки в першій версії системи вони обмежуються на рівні сервісної логіки. Такий підхід спрощує структуру БД і водночас залишає можливість подальшого винесення довідників у окремі таблиці.

Для забезпечення цілісності даних у базі використано зовнішні ключі з дією ON DELETE SET NULL. Така дія обрана свідомо. Якщо користувач видаляє дисципліну, пов'язані задачі, події або записи часу не повинні автоматично зникати, бо вони можуть мати самостійну історичну цінність. Натомість зв'язок із видаленою дисципліною очищується. Аналогічно, якщо видаляється задача, записи фактичного часу зберігаються, але поле task_id очищується. Такий підхід зменшує ризик випадкової втрати журналу роботи.

У базі даних також створено індекси, що прискорюють типові вибірки. Індекс ix_tasks_status_due орієнтований на фільтрацію задач за статусом і дедлайном. Індекси ix_tasks_course, ix_events_course, ix_timelogs_course прискорюють вибірки за дисципліною. Індекс ix_events_start потрібний для календаря, де події обираються за тижневим інтервалом. Індекс ix_timelogs_date_activity підтримує звіти за датою й видом активності. Така індексація не є надмірною, адже відповідає реальним запитам інтерфейсу.

Окремо треба обґрунтувати використання WAL-журналювання. У модулі database.py під час підключення виконується PRAGMA journal_mode = WAL, а також PRAGMA busy_timeout = 5000. Документація SQLite пояснює, що WAL-режим має особливий механізм журналювання і після встановлення зберігається для бази даних [10]. Для локального вебзастосунку така конфігурація корисна,

оскільки читання даних для сторінок і запис через форми можуть відбуватися близько в часі. busy_timeout дає змогу не переривати операцію одразу, якщо база тимчасово зайнята.

Підсумок підрозділу полягає в тому, що база даних TimeTeach Pro спроєктована як компактна реляційна структура, достатня для повноцінного обліку дисциплін, задач, подій, часу й налаштувань. Використання зовнішніх ключів, індексів, WAL-журналювання та окремого модуля підключення забезпечує технічну надійність у межах локального сценарію роботи.

2.3 Обґрунтування вибору засобів програмної реалізації

Вибір засобів програмної реалізації визначався не абстрактною популярністю технологій, а відповідністю вимогам конкретної роботи. Для системи підтримки тайм-менеджменту викладача ЗВО головними критеріями стали автономність, прозорість коду, простота перевірки, відсутність складного розгортання, можливість роботи з локальною базою даних і придатність для створення повноцінного вебінтерфейсу. Унаслідок такого відбору використано Python, SQLite, HTML, CSS, JavaScript і PlantUML.

Python обрано як основну мову програмування через зрозумілий синтаксис, наявність стандартних модулів для HTTP-сервера, роботи з SQLite, CSV, JSON, файлами й аргументами командного рядка. У контексті навчальної роботи окрема перевага Python полягає в тому, що логіку системи можна описати компактно й читабельно, не приховуючи її за великою кількістю фреймворкових абстракцій. Модуль http.server дає змогу побудувати локальний сервер, а модуль sqlite3 – працювати з базою без встановлення зовнішніх пакетів [8; 9].

Порівняння обраних засобів реалізації з альтернативами подано в табл. 2.3. Таблиця демонструє, що вибір технологій не був випадковим, а випливав із вимог до локальності й простоти запуску.

Таблиця 2.3

Обґрунтування вибору засобів програмної реалізації

Компонент	Обране рішення	Можлива альтернатива	Обґрунтування вибору
Мова програмування	Python	JavaScript/Node.js, Java, PHP	стандартні модулі, простий запуск, читабельність коду
Локальний сервер	http.server	Flask, Django, Express	відсутність зовнішніх залежностей, достатність для локальної демонстрації
База даних	SQLite	PostgreSQL, MySQL	файлова БД, автономність, просте резервне копіювання
Інтерфейс	HTML + CSS	готовий UI-фреймворк	контроль стилю, відсутність зовнішніх CDN
Клієнтська логіка	JavaScript	бібліотеки React/Vue	мінімізація залежностей, достатня функціональність
Графіки	Canvas API	Chart.js, D3.js	робота без підключення зовнішніх бібліотек
Діаграми	PlantUML	ручне креслення, draw.io	текстовий опис, відтворюваність, зручність для додатків

Як показано в табл. 2.3, обраний стек має не максимальну промислову потужність, а високу відповідність вимогам навчального програмного проєкту. Якщо метою було б розгортання корпоративної системи для всього університету, доцільним став би інший стек: повноцінний вебфреймворк, серверна СКБД, авторизація, журнал аудиту, ролі доступу й захищене розгортання. Однак для теми вебінтерфейсу бази даних персональної інформаційної системи викладача локальний стек є методично виправданим.

Окрема група вимог стосується безпеки. OWASP Top 10 трактується як стандартний орієнтир для розробників щодо найбільш критичних ризиків вебзастосунків [14]. У розробленій системі не реалізовано публічний доступ і багатокористувацьку модель, проте кілька базових рішень відповідають безпековій логіці: SQL-запити виконуються з параметрами, статичні файли віддаються лише з дозволеного каталогу, шлях до статичного ресурсу перевіряється через resolve(), а сервер за замовчуванням запускається на локальній адресі. Такі заходи не перетворюють навчальний проєкт на захищену промислову платформу, але запобігають найбільш очевидним помилкам локальної реалізації.

Питання доступності інтерфейсу також враховано на рівні проєктних рішень. WCAG 2.2 охоплює рекомендації щодо доступності вебвмісту [15]. У TimeTeach Pro ці підходи проявляються через стабільне ліве меню, контрастні

картки показників, підписані поля форм, зрозумілу ієрархію заголовків, відсутність перевантажених екранів і достатню ширину інтерактивних елементів. Формальної перевірки на відповідність усім критеріям WCAG у межах роботи не проводилось, тому висновок обмежується рівнем урахування базових принципів читабельності й навігаційної зрозумілості.

Підсумовуючи підрозділ, вибір Python, SQLite, HTML, CSS, JavaScript і PlantUML визначається прикладною метою роботи. Обрані засоби забезпечують автономність, читабельність, простоту демонстрації, достатній функціонал і можливість документування системи через формальні діаграми. Обмеження обраного стеку визнано прямо: система призначена для локального використання й навчальної демонстрації, а не для відкритої багатокористувацької експлуатації.

2.4 Проєктування маршрутів, сторінок і користувацької взаємодії

Вебінтерфейс інформаційної системи TimeTeach Pro спроєктовано за модульним принципом: кожна сторінка відповідає окремому функціональному блоку предметної області. Дашборд подає зведені показники, сторінка задач відповідає за планування й контроль виконання, календар організовує події в межах тижня, журнал часу фіксує фактичні витрати, дисципліни формують освітній контекст, звіти забезпечують аналітичне узагальнення, налаштування зберігають персональні параметри системи. Такий поділ зменшує когнітивне навантаження на користувача: викладач не шукає всі функції на одному екрані, а переходить до потрібного розділу через ліве навігаційне меню.

Маршрути системи реалізовано у файлі server.py. Для читання сторінок використовуються GET-запити, для збереження, оновлення й видалення даних – POST-запити. Після виконання операції POST сервер повертає перенаправлення зі статусом 303, що запобігає повторному надсиланню форми під час оновлення сторінки. Для аналітичних графіків передбачено JSON-маршрути, які повертають дані без перезавантаження HTML-сторінки. CSV-експорт

реалізовано окремими маршрутами, що повертають текстовий файл із відповідним заголовком Content-Disposition.

Основні маршрути системи наведено в табл. 2.4. Таблиця дає змогу простежити зв’язок між користувацькими сторінками, HTTP-методами й програмною логікою.

Таблиця 2.4

Основні маршрути вебінтерфейсу TimeTeach Pro

Маршрут	Метод	Призначення	Відповідний модуль
/	GET	дашборд із показниками, найближчими задачами й подіями	dashboard_page, dashboard_summary
/tasks	GET	список, фільтри й форма задач	tasks_page, get_tasks
/tasks/save	POST	створення або редагування задачі	save_task
/tasks/status	POST	зміна статусу задачі	set_task_status
/tasks/delete	POST	видалення задачі	delete_task
/courses	GET	реєстр дисциплін	courses_page, get_courses
/courses/save	POST	створення або редагування дисципліни	save_course
/calendar	GET	тижневий календар подій	calendar_page, get_events
/events/save	POST	створення або редагування події	save_event
/timelog	GET	журнал фактичного часу	timelog_page, get_timelogs
/timelog/save	POST	додавання або редагування запису часу	save_timelog
/reports	GET	сторінка аналітичних звітів	reports_page
/api/workload	GET	дані для графіка робочого часу	workload_by_day
/api/course-share	GET	структура часу за дисциплінами	course_share
/api/activity-share	GET	структура часу за видами активності	activity_share
/api/deadlines	GET	часовий розподіл дедлайнів	deadlines_timeline
/export/tasks.csv	GET	експорт задач у CSV	export_csv
/export/timelogs.csv	GET	експорт журналу часу у CSV	export_csv
/settings	GET	сторінка налаштувань	settings_page
/settings/backup	POST	створення резервної копії БД	backup_database

Як показано в табл. 2.4, система охоплює повний набір операцій для основних сутностей. Маршрути не дублюють один одного, а утворюють логічну карту взаємодії. Для кожної групи даних передбачено сторінку перегляду й окремі маршрути дій. Завдяки такій організації можна перевірити працездатність системи не лише в браузері, а й на рівні коду: кожна дія має свій програмний вхід.

Користувацький сценарій роботи з інтерфейсом починається з дашборду. На головній сторінці користувач бачить назву системи, короткий опис, кнопки швидкої дії, картки показників, графік робочого часу, структуру часу за дисциплінами, найближчі задачі та події. Дашборд не призначений для ручного введення всіх даних; його роль полягає в оперативному огляді ситуації. Якщо показники порожні, система виводить нейтральні повідомлення про відсутність даних, а не генерує штучну аналітику. Така поведінка важлива для академічної доброчесності опису програмного продукту: система показує тільки ті дані, які справді є в базі.

Сторінка задач реалізує керування запланованими діями. У користувача є можливість додати назву, опис, дедлайн, пріоритет, статус, вид активності, орієнтовну тривалість і дисципліну. Фільтрація задач виконується за текстовим пошуком, статусом, пріоритетом, дисципліною та дедлайном. Зміна статусу винесена в окрему дію, що дає змогу швидко переводити задачу між станами «черга», «у роботі» й «виконано». Така логіка відповідає практиці персонального планування, але адаптована до викладацького контексту через прив’язку до дисциплін і видів активності.

Календарна сторінка побудована навколо тижневого інтервалу. Події мають назву, початок, завершення, вид, місце, примітку та дисципліну. Тижнева модель обрана через її практичну придатність: викладач часто планує навантаження саме в межах навчального тижня, а не лише окремого дня. Календар підтримує перехід між тижнями, тому користувач може переглядати як поточні, так і майбутні події. Події типу «заняття», «консультація», «нарада», «дедлайн», «екзамен» або «особистий блок» відрізняються змістом, але зберігаються в одній таблиці events, що спрощує структуру БД.

Журнал часу реалізує фактичний облік виконаної роботи. На відміну від задач, які відображають план, журнал фіксує реальний часовий інтервал. Користувач вказує дату, час початку, час завершення, кількість хвилин, вид активності, дисципліну, задачу та примітку. Якщо тривалість не введена вручну, система може обчислити її за початком і завершенням, але значення

нормалізується, щоб уникнути некоректних записів. Завдяки журналу часу система отримує дані для графіків і показників дашборду.

2.5 Проєктування аналітичних показників і звітності

Аналітичний блок системи TimeTeach Pro має прикладне значення, оскільки перетворює накопичені записи на зрозумілі показники робочого навантаження. Якщо система лише зберігала б дисципліни, задачі й події, вона залишалася б електронним реєстром. Наявність звітів переводить її на рівень інструмента підтримки рішень: викладач може оцінити, скільки часу фактично витрачено за останній період, яка частка роботи припадає на підготовку, перевірку або наукову діяльність, які дедлайни наближаються, а які задачі вже прострочені.

Аналітика в системі побудована без зовнішніх статистичних бібліотек. Дані агрегуються SQL-запитами в модулі services.py, повертаються через JSON-маршрути й відображаються у браузері через JavaScript-графіки. Такий підхід відповідає загальній архітектурі проєкту: система не залежить від сторонніх пакетів, але має достатні можливості для візуального подання інформації. Canvas API використано для побудови графіків локально в браузері, що дозволяє уникнути підключення зовнішніх бібліотек візуалізації [17].

Склад основних аналітичних функцій наведено в табл. 2.5. Таблиця показує, які саме дані використовуються для кожного показника й яке практичне значення має результат.

Таблиця 2.5

Аналітичні функції інформаційної системи TimeTeach Pro

Аналітична функція	Джерело даних	Період	Результат
dashboard_summary()	tasks, events, timelogs, courses	7 днів або поточний стан	картки дашборду: години, виконання задач, дедлайни, активні дисципліни
workload_by_day(days)	timelogs	14 днів за замовчуванням	динаміка робочого часу за датами й видами активності
course_share(days)	timelogs, courses	30 днів за замовчуванням	розподіл фактичного часу за дисциплінами

activity_share(days)	timelogs	30 днів за замовчуванням	структура часу за видами активності

Продовження табл. 2.5

deadlines_timeline(days)	tasks	21 день за замовчуванням	кількість невиконаних дедлайнів за датами
export_csv(kind)	tasks або timelogs	залежно від типу експорту	файл CSV для подальшого аналізу

Як показано в табл. 2.5, аналітичний блок охоплює як поточний стан, так і часову динаміку. Дашборд забезпечує швидку оцінку, тоді як сторінка звітів дає розширене уявлення про структуру навантаження. Для навчального проєкту важливо, що показники обчислюються прозоро: вони ґрунтуються на сумуванні хвилин, групуванні за датами, дисциплінами, активностями й дедлайнами, а не на прихованих алгоритмах.

Формула розрахунку фактичних годин за період має простий вигляд: $H = \frac{\sum_{i=1}^{n} m_i}{60}$, де H- кількість годин за вибраний період, m_i- кількість хвилин в окремому записі журналу часу, n- кількість записів за період. Такий показник не потребує складної інтерпретації, але має цінність для контролю навантаження, оскільки переводить окремі записи в узагальнену часову величину.

Частка зосередженої роботи в системі обчислюється як відношення часу, витраченого на змістовні види роботи, до загального зафіксованого часу: $F = \frac{\sum m_f}{\sum m} \cdot 100\%$, де F- частка зосередженої роботи, m_f- хвилини за видами активності, віднесеними до змістовної роботи, m- усі зафіксовані хвилини за період. До таких активностей у системі віднесено підготовку занять, наукову роботу, методичну роботу, перевірку робіт і написання матеріалів. Показник має внутрішній характер і не претендує на нормативну оцінку ефективності викладача. Його призначення – допомогти користувачеві побачити співвідношення між роботою, що потребує концентрації, та загальним обсягом зафіксованих дій.

Показник виконання задач розраховується як відношення задач зі статусом done до загальної кількості задач: $C = \frac{T_d}{T} \cdot 100\%$, де C- відсоток виконання, T_d-

кількість виконаних задач, T- загальна кількість задач. Якщо задач немає, система повертає нульове значення, а не створює некоректний відсоток. Така перевірка потрібна для уникнення ділення на нуль і для коректного відображення порожньої бази.

Логіку формування аналітичних звітів подано на рис. 2.6. Діаграма демонструє шлях даних від журналу часу й задач до графіків, CSV-експорту й карток дашборду.

Рисунок 2.3. Формування аналітичних показників у системі TimeTeach Pro

Як показано на рис. 2.6, аналітична частина системи не створює окремого сховища. Усі звіти формуються з основної бази даних, тому ризик розходження між реєстрами й аналітикою мінімальний. Якщо користувач редагує або видаляє запис часу, відповідні графіки змінюються після нового запиту до API. Така модель підтримує актуальність даних без додаткової синхронізації.

CSV-експорт виконує роль містка між системою й зовнішнім аналізом. Користувач може експортувати задачі або журнал часу, а потім відкрити файл у табличному процесорі. Для навчального проєкту така функція має практичну цінність: вона показує, що система не замкнена у власному інтерфейсі, а дозволяє переносити дані для подальшої обробки. При цьому CSV формується без зовнішніх залежностей через стандартний модуль Python csv.

Підсумовуючи підрозділ, аналітичний блок TimeTeach Pro реалізує дашборд, динаміку робочого часу, структуру часу за дисциплінами, розподіл за видами активності, часову карту дедлайнів і CSV-експорт. Показники мають прозору логіку обчислення й прямо пов’язані з даними, які користувач вносить у систему. Завдяки цьому аналітика не є декоративним елементом інтерфейсу, а виконує функцію інформаційної підтримки планування.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ, ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА РОБОТА З НЕЮ КОРИСТУВАЧА

3.1 Реалізація серверної частини та механізмів взаємодії з базою даних

Програмна реалізація інформаційної системи TimeTeach Pro побудована з урахуванням головної вимоги роботи – система повинна запускатися з терміналу однією командою, відкриватися у браузері, працювати з локальною базою даних і забезпечувати повний цикл операцій над даними. Такий підхід дав змогу уникнути надмірної складності розгортання, але не звів систему до демонстраційного макета. У межах реалізованого проєкту серверна частина виконує роль посередника між браузером, прикладною логікою та SQLite-базою. Користувач працює з формами, таблицями, фільтрами й графіками, тоді як сервер приймає HTTP-запити, перевіряє отримані значення, викликає відповідні сервісні функції, читає або змінює записи в базі даних і повертає результат у вигляді HTML-сторінки, JSON-відповіді або CSV-файлу.

Вхідною точкою системи є файл run.py. Його призначення не обмежується механічним запуском сервера. Файл перевіряє доступність обраного порту, ініціалізує структуру бази даних, за потреби створює початкові демонстраційні записи та відкриває браузер після старту локального сервера. Така логіка підвищує зручність перевірки, оскільки користувачеві не потрібно окремо запускати базу даних, налаштовувати адресу сервера або шукати головну сторінку вручну. У типовому сценарії достатньо перейти до каталогу проєкту й виконати команду python3 run.py, після чого застосунок стає доступним за локальною адресою http://127.0.0.1:8000.

Серверна частина реалізована у файлі server.py, де описано обробку GET- і POST-запитів. GET-маршрути використовуються для відображення сторінок, отримання даних для графіків і завантаження CSV-файлів. POST-маршрути відповідають за створення, редагування, зміну статусу й видалення записів.

Після успішного виконання POST-операції сервер повертає перенаправлення зі статусом 303, що відповідає практиці «post-redirect-get» і запобігає повторному надсиланню форми після оновлення сторінки. Для навчального програмного проєкту така деталь має значення, бо демонструє не лише наявність форм, а й коректну організацію циклу користувацької взаємодії.

У серверній частині застосовано розмежування між маршрутизацією та прикладною логікою. Файл server.py не повинен безпосередньо містити весь набір SQL-запитів, оскільки така практика швидко ускладнює супровід коду. Натомість основні операції винесено до services.py, де зосереджено функції збереження дисциплін, отримання задач, зміни статусів, формування подій, записів часу, звітів, експорту й резервного копіювання. Файл database.py відповідає за встановлення з'єднання з SQLite, увімкнення зовнішніх ключів, створення таблиць та індексів. Поділ відповідальності між файлами створює чітку внутрішню архітектуру, показану раніше на рис. 2.1, і дає змогу перевіряти кожен рівень системи окремо.

У роботі з базою даних застосовано параметризовані SQL-запити. Їх використання має не лише технічну, а й методичну цінність, оскільки демонструє безпечнішу практику роботи з користувацькими значеннями. З огляду на локальний характер системи загрози мають інший масштаб, ніж у відкритому вебсервісі, але принцип захищеного програмування не повинен зникати. OWASP Web Security Testing Guide розглядає тестування безпеки вебзастосунків як окрему дисципліну, що охоплює перевірку проєктних рішень, реалізації та експлуатаційних умов [18]. У TimeTeach Pro така логіка реалізована на базовому рівні: запити до бази не будуються через конкатенацію рядків із введеними користувачем значеннями, статичні файли обмежуються каталогом static, а сервер призначений для локального запуску.

Окрему роль у серверній частині відіграє обробка форм. У системі передбачено форми для дисциплін, задач, подій, журналу часу й налаштувань. Після отримання форми сервер перетворює дані запиту у словник, передає їх у сервісну функцію й очікує нормалізований результат. Наприклад, під час

збереження задачі система перевіряє назву, приводить пріоритет і статус до допустимих значень, обробляє дату дедлайну, переводить орієнтовну тривалість у числовий формат, а порожній ідентифікатор дисципліни перетворює на NULL. Без такої проміжної логіки база даних швидко накопичувала б неузгоджені значення, а звіти втрачали б точність.

У системі реалізовано також спеціальні маршрути для JSON-відповідей. Вони потрібні не для сторінок, а для графіків. Наприклад, динаміка робочого часу за днями передається у браузер через маршрут /api/workload, структура часу за дисциплінами – через /api/course-share, розподіл за видами активності – через /api/activity-share, часовий розподіл дедлайнів – через /api/deadlines. Такий поділ дає змогу не перезавантажувати всю сторінку під час побудови графіка й робить аналітичну частину гнучкішою. Дані готуються на сервері, а їхнє візуальне подання формується клієнтським кодом.

Ключові серверні функції подано в табл. 3.1. Таблиця узагальнює не весь код, а ті програмні елементи, які забезпечують повний цикл роботи системи.

Таблиця 3.1

Основні серверні та сервісні функції TimeTeach Pro

Функція або група функцій	Модуль	Призначення
main()	run.py	запуск системи, перевірка порту, ініціалізація БД, відкриття браузера
init_db()	database.py	створення таблиць, індексів, початкових налаштувань
get_connection()	database.py	відкриття з’єднання з SQLite та ввімкнення службових параметрів
save_course(), get_courses()	services.py	створення, редагування й отримання дисциплін
save_task(), get_tasks(), set_task_status()	services.py	повний цикл роботи із задачами
save_event(), get_events()	services.py	створення й отримання календарних подій
save_timelog(), get_timelogs()	services.py	фіксація фактичного часу й отримання журналу
dashboard_summary()	services.py	формування показників операційної панелі
workload_by_day(), course_share()	services.py	агрегування даних для графіків
export_csv()	services.py	формування CSV-файлів для задач і журналу часу
backup_database()	database.py	створення резервної копії SQLite-файлу

Як показано в табл. 3.1, серверна частина охоплює як базові CRUD-операції, так і додаткові функції, без яких система залишалася б неповною:

дашборд, звіти, експорт і резервне копіювання. Такий набір підтверджує, що TimeTeach Pro є не макетом сторінок, а завершеним локальним застосунком із повним циклом роботи з даними.

Підсумовуючи підрозділ, серверна реалізація системи побудована на чіткому розмежуванні запуску, маршрутизації, прикладної логіки, роботи з базою та формування представлення. Така будова забезпечує зрозумілість, перевірюваність і технічну цілісність програмного продукту. Подальший аналіз має перейти до клієнтської частини, оскільки саме інтерфейс визначає практичну зручність роботи викладача з базою даних.

3.2 Реалізація вебінтерфейсу та основних функціональних модулів

Користувацький інтерфейс TimeTeach Pro реалізовано як локальний вебінтерфейс, що відкривається у браузері після запуску серверної частини. Його завдання полягає не тільки в естетичному поданні даних, а й у правильній організації професійної роботи викладача. Через таку вимогу інтерфейс побудовано навколо кількох стабільних зон: лівого навігаційного меню, верхньої інформаційної панелі, основної робочої області, карток показників, таблиць, форм і графіків. Користувач не переходить між випадковими вікнами, а працює в єдиній системі з однаковою логікою розташування елементів.

Візуальна структура головної сторінки відповідає операційній природі дашборду. У верхній частині подано назву системи, коротке пояснення призначення, кнопки швидкого переходу до створення задачі або події. Нижче розміщено картки з ключовими показниками: години за 7 днів, частка зосередженої роботи, виконання задач, прострочені задачі, дедлайни на 7 днів і активні дисципліни. Така організація дає змогу користувачеві одразу побачити стан робочого навантаження, не відкриваючи окремі сторінки. Після карток подано графічні блоки, списки найближчих задач і подій. Головна сторінка тому виконує роль короткого аналітичного огляду, а не звичайної заставки.

Сторінка задач реалізує повний цикл роботи з плановими діями. У верхній частині розміщено фільтри за текстом, статусом, пріоритетом, дисципліною та дедлайном. Основна таблиця показує назву задачі, дисципліну, дедлайн, пріоритет, статус, вид активності й орієнтовну тривалість. Поруч із кожною задачею передбачено дії для редагування, зміни статусу й видалення. Форма створення задачі структурована так, щоб користувач міг швидко внести зміст роботи, але водночас не пропустив поля, потрібні для аналітики. Саме тому задача має не тільки назву й дедлайн, а й пріоритет, статус, тип активності та зв’язок із дисципліною.

Сторінка календаря відображає події в тижневому інтервалі. Така модель відповідає звичній логіці навчальної роботи викладача, де тиждень часто виступає базовою одиницею планування. Події мають дату й час, назву, вид, місце проведення, примітки та прив’язку до дисципліни. Користувач може переходити між тижнями, бачити заплановані заняття, консультації, наради, екзамени або персональні робочі блоки. Календарна сторінка не дублює задачі, а доповнює їх часовою організацією: задача вказує, що потрібно зробити, а подія показує, коли саме відбудеться дія або блок роботи.

Журнал часу реалізує фактичну сторону тайм-менеджменту. Користувач вносить дату, початок і завершення роботи, кількість хвилин, вид активності, дисципліну, задачу та примітку. Таке подання дає змогу зберігати не лише факт виконання, а й структуру витрат часу. Наприклад, підготовка лекції, перевірка практичних робіт і консультація зі здобувачем можуть бути пов’язані з однією дисципліною, але відрізняються за видом активності. Саме журнал часу створює основу для звітів, тому його структура має бути достатньо деталізованою.

Сторінка дисциплін виконує роль предметного довідника. У ній зберігаються код дисципліни, назва, семестр, кількість кредитів ECTS, тижневі години та ознака активності. Така модель не є надмірною для персональної системи, оскільки викладач може працювати з різними дисциплінами в різних семестрах, а не лише з поточними задачами. Ознака активності дає змогу відокремити поточні дисципліни від тих, що вже завершені, не видаляючи їх із

бази. У майбутньому така структура може бути розширена довідником освітніх програм, груп і типів занять.

Сторінка звітів узагальнює дані, введені в інших модулях. Вона містить графік динаміки робочого часу, структуру часу за дисциплінами, розподіл за видами активності та часову карту дедлайнів. Графіки побудовані без зовнішніх бібліотек, через JavaScript і Canvas API. MDN описує Canvas API як засіб малювання графіки через JavaScript, придатний, зокрема, для візуалізації даних [17]. Така реалізація відповідає загальній вимозі автономності: система не звертається до зовнішнього CDN і не потребує інтернету для побудови графіків.

Загальну структуру сторінок і функціональних модулів показано на рис. 3.1. Діаграма фіксує не технічну архітектуру, а композицію інтерфейсу, через яку користувач взаємодіє з базою даних.

Рисунок 3.1. Структура вебінтерфейсу користувача TimeTeach Pro

Як показано на рис. 3.1, вебінтерфейс не є набором ізольованих сторінок. Модулі пов’язані між собою через спільні дані: дисципліни використовуються в задачах, календарі й журналі часу; задачі можуть уточнювати записи фактичного виконання; журнал часу формує основу звітів; налаштування впливають на персоналізацію системи. Така внутрішня зв’язність підтверджує, що інтерфейс відображає реальну модель даних, а не лише декоративну навігацію.

Практична реалізація вебінтерфейсу TimeTeach Pro передбачає поділ системи на окремі функціональні сторінки, кожна з яких відповідає певному сценарію роботи викладача. Після переходу до відповідного розділу користувач отримує доступ не лише до форми введення даних, а й до інструментів фільтрації, перегляду, редагування або експорту інформації. Така побудова забезпечує логічну послідовність роботи із системою: спочатку викладач формує перелік дисциплін, далі створює задачі й календарні події, після виконання роботи фіксує фактичні часові інтервали, а потім аналізує накопичені дані у звітному модулі.

Рисунок 3.2. Сторінка керування задачами TimeTeach Pro

Як показано на рис. 3.2, сторінка задач призначена для оперативного створення, класифікації та контролю поточних справ викладача. Форма додавання задачі містить поля для назви, дисципліни, дедлайну, пріоритету, статусу, очікуваної тривалості, типу активності та опису. Завдяки цьому задача в системі не зводиться до короткої текстової нотатки, а перетворюється на структурований запис бази даних. Особливе значення має можливість прив’язки задачі до конкретної дисципліни, оскільки це дозволяє надалі аналізувати, на які навчальні компоненти витрачається найбільше часу. Блок фільтрації за назвою, статусом, пріоритетом, дисципліною та дедлайном забезпечує швидкий пошук потрібних записів навіть за умови накопичення значної кількості задач.

Рисунок 3.3. Сторінка календаря занять і подій TimeTeach Pro

На рис. 3.3 подано сторінку календаря, яка реалізує календарну модель планування робочого часу викладача. Користувач може додати подію із зазначенням назви, дисципліни, часу початку й завершення, типу події, локації та приміток. Така структура є зручною для фіксації занять, консультацій, засідань кафедри, нарад, екзаменів або персональних блоків часу. Тижнева сітка дозволяє візуально оцінити розподіл навантаження протягом конкретного періоду. Наявність кнопок переходу до попереднього, поточного й наступного тижня робить календар придатним не лише для поточного контролю, а й для короткострокового планування майбутньої роботи.

Рисунок 3.4. Сторінка фактичного обліку робочих інтервалів TimeTeach Pro

Як видно з рис. 3.4, модуль обліку часу призначений для фіксації фактично виконаної роботи. На відміну від календаря, який відображає заплановані події,

журнал часу показує реальні часові інтервали, витрачені на певний вид активності. Користувач зазначає дату, час початку, час завершення, кількість хвилин, тип активності, дисципліну, пов’язану задачу та короткі нотатки щодо виконаної роботи. Над формою розміщено оперативні показники: кількість записів, сумарна кількість годин і середня тривалість одного інтервалу. Це дозволяє викладачеві швидко оцінити фактичне навантаження без переходу до розгорнутого звіту. Додатковою перевагою є можливість експорту записів у форматі CSV.

Рисунок 3.5. Сторінка ведення дисциплін і навчального навантаження TimeTeach Pro

Рис. 3.5 демонструє сторінку дисциплін, яка виконує функцію довідника навчальних компонентів. Для кожної дисципліни передбачено введення коду, назви, семестру, кількості кредитів ECTS, годин на тиждень і статусу активності. Цей модуль є базовим для всієї системи, оскільки дисципліни використовуються в задачах, календарних подіях, журналі часу та звітах. Завдяки такому рішенню система може не просто зберігати окремі події або завдання, а формувати зв’язану модель навчального навантаження викладача. Табличне подання дисциплін у нижній частині сторінки дає змогу швидко переглянути вже внесені записи та надалі використовувати їх у суміжних модулях.

Рисунок 3.6. Сторінка звітів та аналітики навантаження TimeTeach Pro

На рис. 3.6 показано сторінку звітів, яка узагальнює накопичені дані про фактичний час, структуру активностей, дисципліни та майбутні дедлайни. Цей модуль виконує аналітичну функцію і перетворює введені користувачем записи на показники, придатні для оцінювання навантаження. Передбачено окремі блоки для аналізу робочих годин за останні 14 днів, частки дисциплін за останні 30 днів та експорту даних у CSV-форматі. За відсутності записів система

коректно повідомляє користувача про недостатність даних для побудови графіків. Це свідчить про наявність базової логіки перевірки інформаційного наповнення перед формуванням аналітичного результату.

Рисунок 3.7. Сторінка налаштувань локального стенду TimeTeach Pro

Рис. 3.7 відображає сторінку налаштувань, призначену для персоналізації локального середовища роботи. Користувач може задати ПІБ викладача, назву ЗВО або підрозділу, початок і завершення робочого дня, норму годин, а також щільність інтерфейсу. Налаштування зберігаються у таблиці `kv_settings` наявної SQLite-бази, що забезпечує їх повторне використання після перезапуску системи. Наявність цього модуля підкреслює прикладний характер програмного продукту, оскільки система адаптується до конкретного користувача й не залишається статичним демонстраційним макетом.

Узагальнюючи зміст рис. 3.2-3.7, можна стверджувати, що розроблений вебінтерфейс охоплює всі основні сценарії тайм-менеджменту викладача ЗВО. Сторінка дисциплін створює інформаційну основу для прив’язки навчального навантаження; сторінка задач забезпечує планування поточних справ; календар відображає часову структуру занять і подій; журнал часу фіксує фактичне виконання роботи; звіти забезпечують аналітичне узагальнення; налаштування дають змогу персоналізувати систему. Взаємодія цих модулів підтверджує, що TimeTeach Pro є не набором окремих форм, а цілісною інформаційною системою підтримки тайм-менеджменту викладача закладу вищої освіти.

Для оцінювання реалізованих функціональних модулів доцільно узагальнити їх у табл. 3.2. У таблиці показано, яку практичну дію забезпечує кожна сторінка та які дані вона використовує.

Таблиця 3.2

Реалізація основних функціональних модулів системи TimeTeach Pro

Модуль	Основні дії користувача	Дані, що використовуються

Дашборд	перегляд показників, найближчих задач, подій і графіків	задачі, події, дисципліни, журнал часу
Задачі	створення, редагування, фільтрація, зміна статусу, видалення	tasks, courses
Календар	планування подій, перехід між тижнями, перегляд занять і дедлайнів	events, courses
Облік часу	фіксація фактичної роботи, зв’язок із задачею або дисципліною	timelogs, tasks, courses
Дисципліни	ведення навчальних дисциплін, семестру, кредитів і активності	courses
Звіти	перегляд графіків, структури часу, дедлайнів, експорт CSV	timelogs, tasks, courses
Налаштування	зміна профілю викладача, параметрів робочого дня, резервне копіювання	kv_settings, файл SQLite

Як показано в табл. 3.2, кожен модуль системи має конкретне функціональне навантаження й працює з визначеними таблицями бази даних. Така відповідність між сторінками та сутностями підвищує прозорість системи. Перевіряльник може послідовно відкрити кожний модуль, виконати типову дію й переконатися, що база даних змінюється коректно.

Інтерфейс також ураховує базові принципи доступності. WCAG 2.2 визначає підходи до того, як робити вебвміст доступнішим для користувачів із різними потребами [15]. У TimeTeach Pro такі принципи реалізовано на рівні читабельності, контрасту, сталого розташування навігації, зрозумілих підписів форм, логічної ієрархії заголовків і достатнього розміру інтерактивних елементів. Формальної сертифікації доступності система не проходила, але базові вимоги до сприйняття й навігації враховано під час проєктування сторінок.

Підсумовуючи підрозділ, вебінтерфейс TimeTeach Pro реалізує повний користувацький цикл: огляд стану навантаження, планування задач, календарне розміщення подій, фіксацію фактичного часу, ведення дисциплін, перегляд звітів, експорт і резервне копіювання. Інтерфейс має єдину композицію, предметну логіку й достатню візуальну якість для використання як готового програмного продукту в межах навчальної роботи.

3.3 Сценарії роботи користувача з інформаційною системою

Практична цінність програмної системи виявляється не лише через перелік реалізованих модулів, а передусім через сценарії роботи користувача. Для TimeTeach Pro основним користувачем є викладач ЗВО, який планує власну роботу, фіксує виконання й аналізує часове навантаження. Через таку постановку задачі сценарії мають охоплювати не випадкові кліки інтерфейсом, а послідовні професійні дії: створення дисциплін, внесення задач, планування календарних подій, облік фактичного часу, перегляд дашборду й формування звіту.

Перший типовий сценарій пов’язаний із початковим налаштуванням системи. Після запуску викладач відкриває сторінку налаштувань і вказує власне ім’я, назву закладу освіти, параметри робочого дня та тижневу норму годин. Далі створює перелік активних дисциплін, зазначаючи код, назву, семестр, кількість кредитів і тижневі години. Така дія формує предметний контекст для подальших задач і записів часу. Якщо дисципліни не створити, система все одно працюватиме, але аналітика за навчальними компонентами буде неповною.

Другий сценарій охоплює планування задач. Викладач переходить на сторінку «Задачі», створює нову задачу, вказує дедлайн, пріоритет, статус, вид активності, орієнтовну тривалість і дисципліну. Наприклад, задача може стосуватися підготовки лекції, перевірки лабораторних робіт, написання методичних рекомендацій або підготовки матеріалів до наукової публікації. Після створення задача з’являється в загальному списку й може бути відфільтрована за статусом, пріоритетом або дисципліною. Якщо дедлайн наближається, задача враховується в показнику «Дедлайни 7 днів» на дашборді.

Третій сценарій стосується календарного планування. На сторінці календаря викладач створює подію з датою, часом початку й завершення, видом події, місцем і дисципліною. Подія може позначати заняття, консультацію, засідання, екзамен або персональний робочий блок. Такий сценарій особливо корисний для розмежування задач і подій: задача описує запланований

результат, а календарна подія резервує часовий інтервал. Якщо викладач має задачу «підготувати презентацію до лекції», то в календарі може бути створений окремий блок «підготовка лекційного матеріалу». У звітах події та фактичний час не змішуються, що зберігає точність аналітики.

Четвертий сценарій охоплює фактичний облік виконаної роботи. Після завершення певного блоку роботи викладач відкриває сторінку «Облік часу» й створює запис: дата, початок, завершення, тривалість, вид активності, дисципліна, пов’язана задача та примітка. Якщо робота виконувалася в межах конкретної задачі, зв’язок із нею дозволяє надалі зіставити план із фактичним виконанням. Якщо запис стосується загальної діяльності, наприклад адміністративної комунікації, прив’язка до задачі може бути порожньою, але вид активності все одно зберігається. Така гнучкість потрібна, бо не кожна професійна дія викладача заздалегідь оформлюється як задача.

П’ятий сценарій стосується аналізу навантаження. Коли в системі накопичено записи часу, викладач відкриває дашборд або сторінку звітів. Дашборд показує короткі показники, а звіти дають змогу побачити динаміку робочого часу, структуру навантаження за дисциплінами, розподіл за видами активності та дедлайни. Якщо в певному періоді переважає адміністративна робота, користувач бачить таку структуру через графік. Якщо одна дисципліна займає непропорційно багато часу, розподіл за дисциплінами дає підставу скоригувати планування. Система не робить нормативних висновків замість викладача, але надає впорядковані дані для самостійної оцінки.

Повний сценарій роботи користувача показано на рис. 3.8. Діаграма демонструє послідовність дій від запуску системи до аналізу звітів і резервного копіювання.

Рисунок 3.8. Сценарій роботи викладача з інформаційною системою TimeTeach Pro

Як показано на рис. 3.8, робота з системою має циклічний характер. Користувач не завершує взаємодію після внесення однієї задачі, а постійно повертається до планування, виконання, обліку й аналізу. Саме така циклічність відповідає логіці тайм-менеджменту: дані планування уточнюються фактичними записами, а фактичні записи впливають на подальше планування.

Для перевірки повноти користувацьких сценаріїв доцільно подати їх у вигляді таблиці. Табл. 3.3 фіксує основні сценарії, очікуваний результат і критерій успішності.

Таблиця 3.3

Основні сценарії роботи користувача з TimeTeach Pro

Сценарій	Дії користувача	Очікуваний результат	Критерій успішності
Початкове налаштування	введення профілю викладача й параметрів робочого дня	налаштування збережено в БД	дані відображаються після перезапуску
Створення дисципліни	заповнення коду, назви, семестру, кредитів і годин	дисципліна з’являється в реєстрі	дисципліна доступна у формах задач, подій і часу
Створення задачі	введення назви, дедлайну, пріоритету, статусу й дисципліни	задача збережена й показана в таблиці	задача врахована в дашборді та фільтрах
Планування події	введення часу початку й завершення події	подія показана в календарі	подія відображається в потрібному тижні
Фіксація часу	введення дати, тривалості, активності й дисципліни	запис часу збережено	години враховано в графіках і звітах
Зміна статусу задачі	переведення задачі в інший стан	статус оновлено	відсоток виконання змінюється
Експорт CSV	вибір експорту задач або журналу часу	формується CSV-файл	файл містить актуальні записи
Резервне копіювання	запуск створення копії БД	створено файл резервної копії	копія збережена в окремому каталозі

Як показано в табл. 3.3, кожен сценарій має вимірюваний результат. Це дає змогу оцінювати систему не суб’єктивно, а через перевірку конкретних дій. Якщо дисципліна створена, вона повинна з’явитися у відповідному списку. Якщо запис часу додано, він має вплинути на графік. Якщо задача виконана, дашборд повинен змінити показник виконання.

Підсумовуючи підрозділ, сценарії роботи користувача підтверджують практичну завершеність системи. TimeTeach Pro підтримує не окрему операцію, а повний цикл персонального тайм-менеджменту викладача: налаштування, планування, календаризацію, фактичний облік, аналіз, експорт і резервне копіювання. Така послідовність створює основу для перевірки працездатності системи.

3.4 Перевірка працездатності та результати функціонального тестування

Перевірка працездатності TimeTeach Pro виконувалася з позиції навчального програмного проєкту, де головним завданням є підтвердження того, що система запускається, створює базу даних, відкриває вебінтерфейс, виконує основні операції з даними й коректно формує звіти. Для такої системи недоцільно заявляти повноцінне промислове тестування безпеки, навантажувальне тестування або сертифікаційну перевірку доступності. Натомість обґрунтованим є функціональне тестування основних сценаріїв, перевірка структури бази, тестування запуску й ручна перевірка інтерфейсу.

У каталозі проєкту передбачено файл tests/smoke_test.py, який виконує базову перевірку системи. Такий тест не замінює повний набір модульних тестів, але швидко виявляє критичні помилки: відсутність таблиць, неможливість підключення до бази, помилки створення демонстраційних даних або некоректну роботу базових сервісних функцій. Документація Python щодо unittest визначає цей модуль як інструментарій для створення та запуску тестів [19]. У межах TimeTeach Pro така можливість використовується як основа для перевірки найважливіших елементів після розпакування проєкту.

Функціональне тестування виконувалося за логікою «дія – очікуваний результат – фактичний результат». Такий підхід придатний для програмної роботи, оскільки безпосередньо пов’язує вимоги з перевіркою. Наприклад, якщо у вимогах зазначено створення задач, тестовий сценарій має містити створення

задачі, її відображення в таблиці, зміну статусу, участь у показнику виконання й можливість експорту. Якщо заявлено роботу календаря, треба перевірити створення події, відображення в тижневому інтервалі й коректність часу початку та завершення.

Результати функціонального тестування наведено в табл. 3.4. У ній подано ключові перевірки, потрібні для підтвердження працездатності основних модулів.

Таблиця 3.4

Результати функціонального тестування TimeTeach Pro

Перевірка	Очікуваний результат	Фактичний результат	Статус
Запуск python3 run.py	локальний сервер стартує на 127.0.0.1:8000	сервер запускається, браузер відкриває дашборд	виконано
Створення бази даних	файл data/timeteach.db створено автоматично	файл створюється після першого запуску	виконано
Наявність таблиць	у БД є courses, tasks, events, timelogs, kv_settings	таблиці створюються функцією init_db()	виконано
Створення дисципліни	запис з’являється в реєстрі дисциплін	дисципліна зберігається й доступна у формах	виконано
Створення задачі	задача відображається в таблиці задач	запис створюється, фільтри працюють	виконано
Зміна статусу задачі	статус оновлюється без дублювання запису	показник виконання на дашборді змінюється	виконано
Створення події	подія показана в календарі потрібного тижня	календар відображає подію за датою	виконано
Додавання запису часу	запис зберігається й впливає на звіти	години з’являються в аналітичних графіках	виконано
JSON API графіків	API повертає структуровані дані	браузер будує графіки без зовнішніх бібліотек	виконано
CSV-експорт	формується файл задач або журналу часу	CSV завантажується з актуальними даними	виконано
Резервне копіювання	створюється копія SQLite-файлу	копія формується в каталозі резервів	виконано

Як показано в табл. 3.4, основні функції системи виконуються коректно. Перевірка охоплює не лише запуск, а й роботу з усіма ключовими сутностями. Окрема цінність тестування полягає в тому, що воно підтверджує зв’язок між введенням даних і звітністю. Якщо створюється запис часу, він не просто зберігається в таблиці, а впливає на графіки й показники. Якщо змінюється

статус задачі, дашборд відображає нове співвідношення виконаних і невиконаних задач.

Перевірка роботи з базою даних включала перегляд структури таблиць, коректність зовнішніх ключів, індексів і службових параметрів. SQLite підтримує файлову модель зберігання, що не потребує окремого серверного процесу [9]. Для TimeTeach Pro така особливість означає, що тестування можна виконати без встановлення додаткової СКБД. Достатньо запустити систему, внести кілька записів і переконатися, що вони залишаються після перезапуску. Постійність даних після закриття сервера підтверджує, що система працює не як тимчасовий інтерфейс, а як застосунок із реальним сховищем.

Окрема перевірка стосувалася коректності фільтрів і дат. У задачах перевірялась фільтрація за статусом, пріоритетом, дисципліною й дедлайном. У календарі перевірявся перехід між тижнями. У журналі часу перевірялася фільтрація за періодом і дисципліною. Для маршрутизації й обробки параметрів URL застосовується стандартна логіка розбору запитів, сумісна з підходами Python urllib.parse, який призначений для розбору URL на компоненти [20]. Коректна робота параметрів потрібна для того, щоб користувач міг застосовувати фільтри без втрати поточного стану сторінки.

Перевірка інтерфейсу виконувалася за кількома критеріями: читабельність елементів, логічність навігації, відповідність сторінок заявленим функціям, відсутність перевантаження форм, зрозумілі підписи, наявність повідомлень у разі відсутності даних. У системі передбачено нейтральні порожні стани для графіків і списків. Якщо користувач ще не створив записів часу, система не показує хибну динаміку, а повідомляє, що дані відсутні. Така поведінка підвищує достовірність інтерфейсу й не створює ілюзії наявності аналітики там, де дані ще не введені.

Тестування резервного копіювання підтвердило можливість створення копії SQLite-файлу. Для локальної системи така функція має особливу практичну вагу, оскільки база даних зберігається в одному файлі. У разі помилкового видалення даних, перенесення проєкту або демонстрації роботи на іншому

комп’ютері користувач може використати резервну копію як окремий файл. Такий механізм не є повноцінною системою версій, але для персонального локального застосунку він достатній.

Підсумовуючи підрозділ, функціональне тестування підтвердило працездатність системи TimeTeach Pro за основними сценаріями. Система запускається, створює базу даних, зберігає й відображає дисципліни, задачі, події та записи часу, формує дашборд, будує графіки, експортує CSV і створює резервну копію. Межі тестування визначено коректно: перевірено локальний навчальний застосунок, а не промислову багатокористувацьку платформу.

3.5 Оцінювання якості реалізації та напрями подальшого розвитку системи

Оцінювання якості реалізації TimeTeach Pro доцільно виконувати з урахуванням мети розроблення. Система створювалася як локальний вебінтерфейс бази даних для підтримки тайм-менеджменту викладача ЗВО, а не як корпоративна платформа для автоматизованого управління всіма працівниками університету. Через таке обмеження якість реалізації слід оцінювати за відповідністю вимогам, працездатністю, повнотою функціоналу, цілісністю даних, зрозумілістю інтерфейсу, автономністю запуску й можливістю подальшого розвитку.

Першим критерієм якості є функціональна повнота. У системі реалізовано всі модулі, визначені в першому розділі: дисципліни, задачі, календар, журнал часу, дашборд, звіти, експорт, налаштування й резервне копіювання. Кожен модуль має власну сторінку й відповідні серверні маршрути. Така відповідність між вимогами та реалізацією означає, що система не втратила логіку проєктування під час програмування. Особливо показовим є зв’язок між журналом часу й аналітикою: дані, введені користувачем, автоматично впливають на графіки й показники, а не існують як ізольований реєстр.

Другим критерієм є автономність. Система запускається локально, не потребує окремої СКБД, хмарного облікового запису, зовнішнього сервера або сторонніх бібліотек для базового функціонування. Така властивість має значну перевагу для навчальної роботи: програмний продукт можна перевірити на іншому комп'ютері без складного налаштування. Разом із тим автономність має природне обмеження: система не синхронізує дані між пристроями й не підтримує одночасну роботу кількох користувачів. Тому локальність треба розглядати не як універсальну перевагу, а як свідомо обрану характеристику першої версії.

Третім критерієм є цілісність даних. Структура SQLite-бази передбачає первинні ключі, зовнішні ключі, індекси й службові обмеження. Вибір ON DELETE SET NULL дає змогу зберігати історію задач, подій і записів часу навіть у разі видалення дисципліни або задачі. Така поведінка відповідає логіці персонального обліку: видалення довідникового елемента не повинно автоматично знищувати фактично виконану роботу. У майбутньому таку модель можна посилити через журнал змін, але вже в першій версії вона забезпечує прийнятний рівень цілісності.

Четвертим критерієм є зручність інтерфейсу. Візуально система має єдину стилістику, стабільну навігацію, чіткі картки показників, табличне подання даних і зрозумілі форми. Вона не перевантажена зайвими візуальними ефектами, але має достатньо виразний вигляд для демонстрації готового програмного продукту. Вимоги доступності враховано на базовому рівні через читабельність, контрастність, підписані поля та логічну ієрархію сторінок, що узгоджується з загальними підходами WCAG 2.2 [15].

П'ятий критерій стосується супроводу й розширюваності. Завдяки поділу коду на run.py, database.py, services.py, server.py, views.py, styles.css і app.js система може розвиватися без повного переписування. Якщо потрібно додати нову сутність, наприклад групи здобувачів або типи занять, зміни можна послідовно внести до бази, сервісного рівня, маршрутів і представлення. Якщо

потрібно перейти на інший серверний підхід, структура сервісів і БД може бути частково збережена.

Узагальнене оцінювання якості реалізації подано в табл. 3.5. У таблиці показано сильні сторони системи та обмеження, які мають бути враховані в подальшому розвитку.

Таблиця 3.5

Оцінювання якості реалізації TimeTeach Pro

Критерій	Реалізований стан	Оцінка
Функціональна повнота	реалізовано дисципліни, задачі, календар, облік часу, звіти, експорт, резервне копіювання	відповідає меті роботи
Автономність	система працює локально й запускається однією командою	сильна сторона першої версії
Цілісність даних	використано SQLite, зовнішні ключі, індекси, WAL-журналювання	достатньо для персонального застосунку
Зручність інтерфейсу	реалізовано дашборд, меню, форми, таблиці, графіки	придатно для практичного використання
Безпека	параметризовані SQL-запити, локальний запуск, обмеження статичних файлів	достатньо для локального сценарію
Масштабованість	багатокористувацький режим відсутній	напрям подальшого розвитку
Інтеграції	зовнішні календарі й освітні платформи не підключені	напрям подальшого розвитку
Тестування	виконано базове функціональне тестування й smoke-test	потребує розширення для промислового рівня

Як показано в табл. 3.5, система відповідає заявленій меті, але має чітко окреслені межі. Найсильнішими сторонами є автономність, повний набір персональних функцій, зв’язок між сутностями й наявність аналітики. Основні обмеження стосуються масштабування, авторизації, інтеграцій і розширеного тестування. Таке розмежування дозволяє уникнути завищених висновків і водночас показує реальний потенціал розвитку.

Напрями подальшого розвитку системи логічно поділити на функціональні, технічні й організаційні. До функціональних належать імпорт розкладу занять, повторювані події, шаблони задач, розширена робота з групами, нагадування про дедлайни, звіти за семестрами й порівняння планового та

фактичного часу. До технічних належать авторизація, багатокористувацький режим, міграція на серверний фреймворк, журнал аудиту, резервне копіювання за розкладом, розширений набір автоматизованих тестів. До організаційних належать адаптація системи до кафедрального використання, узгодження довідників видів роботи з внутрішніми положеннями ЗВО та підготовка інструкції користувача.

У разі розвитку до корпоративного рівня першочерговими мають стати авторизація та розмежування ролей. Викладач, завідувач кафедри й адміністратор системи не повинні мати однакові права доступу. Також потрібно буде перейти від локального http.server до повноцінного серверного середовища, оскільки документація Python прямо обмежує виробниче використання http.server [8]. Перехід до фреймворку на зразок Flask або Django, окремої серверної СКБД і захищеного розгортання може бути виправданий лише тоді, коли система вийде за межі персонального локального застосунку.

Підсумовуючи підрозділ, реалізація TimeTeach Pro відповідає поставленим вимогам і забезпечує повноцінну роботу локальної інформаційної системи підтримки тайм-менеджменту викладача ЗВО. Система має завершений набір функцій для персонального використання, однак зберігає потенціал для розвитку в напрямі багатокористувацької платформи з інтеграціями, авторизацією та розширеною аналітикою.

ВИСНОВКИ

У роботі здійснено проєктування, програмну реалізацію та перевірку працездатності інформаційної системи підтримки тайм-менеджменту викладача закладу вищої освіти. Відповідно до теми дослідження основну увагу зосереджено не лише на створенні окремого вебінтерфейсу або бази даних, а на формуванні цілісного прикладного програмного рішення, яке забезпечує інформаційну підтримку планування, обліку та аналізу робочого часу викладача. Такий підхід відповідає специфіці професійної діяльності науково-педагогічного працівника, оскільки його робочий час розподіляється між навчальною, методичною, науковою, організаційною, консультаційною та комунікаційною роботою.

У процесі аналізу предметної області встановлено, що тайм-менеджмент викладача ЗВО має складнішу структуру порівняно зі звичайним персональним плануванням. Викладач працює не лише з окремими справами чи календарними подіями, а з багаторівневою системою професійних задач, пов’язаних із дисциплінами, дедлайнами, навчальним навантаженням, комунікацією зі здобувачами освіти та фактичними витратами часу. Саме тому інформаційна система підтримки тайм-менеджменту повинна поєднувати три взаємопов’язані компоненти: календарну модель часу, задачну модель роботи та журнал фактичного виконання дій.

Порівняння програмних аналогів показало, що наявні цифрові інструменти, зокрема календарні сервіси, менеджери задач і системи організації робочих процесів, ефективно розв’язують окремі завдання планування, але не повністю враховують специфіку діяльності викладача ЗВО. Google Calendar зручний для календарного планування, Todoist і Microsoft To Do - для ведення задач, Trello - для візуальної організації процесів, однак ці інструменти не забезпечують повноцінного зв’язку між дисциплінами, задачами, фактичними часовими записами та аналітикою навантаження. Це обґрунтувало потребу в розробленні спеціалізованої інформаційної системи.

У роботі сформовано функціональні та нефункціональні вимоги до інформаційної системи. До основних функціональних вимог віднесено ведення дисциплін, створення й редагування задач, фіксацію календарних подій, облік фактичного робочого часу, формування аналітичних показників, експорт даних і резервне копіювання. До нефункціональних вимог належать простота запуску, локальне збереження даних, зручність інтерфейсу, зрозуміла структура сторінок, стабільність роботи, мінімальна залежність від зовнішніх сервісів і можливість подальшого розширення програмного продукту.

Обґрунтовано вибір архітектури локального вебзастосунку, який запускається з терміналу однією командою та відкривається в браузері. Такий підхід є доцільним для навчального й демонстраційного проєкту, оскільки не потребує складного серверного розгортання, зовнішнього хостингу або створення хмарної інфраструктури. Використання бази даних SQLite дозволяє зберігати всю інформацію у локальному файлі, що спрощує перенесення системи, резервне копіювання й перевірку працездатності. Застосування Python як мови програмування забезпечує достатню гнучкість реалізації серверної логіки, оброблення даних і формування аналітичних показників.

Спроєктовано структуру бази даних інформаційної системи. Основними сутностями визначено дисципліни, задачі, календарні події, часові записи та службові елементи, необхідні для експорту й резервного копіювання. Встановлення зв’язків між цими сутностями забезпечує логічну цілісність системи: задачі можуть бути пов’язані з дисциплінами, часові записи - з конкретними видами діяльності, а календарні події - з плановими часовими інтервалами. Така структура дає змогу не лише зберігати інформацію, а й виконувати її аналітичне узагальнення.

Реалізовано основні програмні модулі інформаційної системи. Система забезпечує додавання, перегляд, редагування та видалення дисциплін, задач, подій і записів фактичного часу. Окрему роль відіграє аналітичний блок, який дозволяє оцінювати розподіл часу за видами діяльності, виявляти прострочені задачі, аналізувати навантаження та формувати оперативне уявлення про стан

виконання роботи. Наявність експорту даних і резервного копіювання підвищує практичну цінність системи, оскільки користувач отримує можливість зберігати результати роботи та переносити їх для подальшого використання.

Перевірка працездатності системи за основними сценаріями використання засвідчила відповідність реалізованого рішення поставленим вимогам. Система коректно виконує базові операції з даними, забезпечує взаємозв’язок між основними сутностями, дає змогу працювати з календарем, задачами й журналом часу, а також формує аналітичні показники. Отримані результати підтверджують, що розроблений програмний продукт може бути використаний як функціональний прототип інформаційної системи підтримки тайм-менеджменту викладача ЗВО.

Практична цінність розробленої системи полягає в її прикладній спрямованості, простоті запуску та адаптації до реальних потреб викладача. Вона може використовуватися як навчальний проєкт, демонстраційний програмний продукт або основа для створення більш складної кафедральної чи університетської системи планування робочого часу. Перевагою реалізованого рішення є те, що система не залежить від зовнішніх хмарних сервісів, працює локально, має зрозумілу структуру й може бути поступово розширена.

Подальший розвиток інформаційної системи доцільно пов’язати з упровадженням авторизації користувачів, розмежуванням ролей, імпортом розкладу занять, інтеграцією з Google Calendar або іншими календарними сервісами, додаванням сповіщень, розширенням аналітичних звітів, формуванням тижневих і місячних підсумків навантаження, а також можливістю використання системи не лише одним викладачем, а групою користувачів у межах кафедри або освітнього підрозділу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цифрова трансформація освіти і науки / Міністерство освіти і науки України. URL: https://mon.gov.ua/tag/tsifrova-transformatsiya-osviti-i-nauki (дата звернення: 07.05.2026).
2. Python 3 Documentation / Python Software Foundation. URL: https://docs.python.org/3/ (дата звернення: 07.05.2026).
3. The Python Tutorial / Python Software Foundation. URL: https://docs.python.org/3/tutorial/index.html (дата звернення: 07.05.2026).
4. sqlite3 — DB-API 2.0 interface for SQLite databases / Python Software Foundation. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 07.05.2026).
5. Flask Documentation / Pallets. URL: https://flask.palletsprojects.com/ (дата звернення: 07.05.2026).
6. Quickstart — Flask Documentation / Pallets. URL: https://flask.palletsprojects.com/en/stable/quickstart/ (дата звернення: 07.05.2026).
7. API — Flask Documentation / Pallets. URL: https://flask.palletsprojects.com/en/stable/api/ (дата звернення: 07.05.2026).
8. Jinja Documentation / Pallets. URL: https://jinja.palletsprojects.com/ (дата звернення: 07.05.2026).
9. Template Designer Documentation — Jinja Documentation / Pallets. URL: https://jinja.palletsprojects.com/en/stable/templates/ (дата звернення: 07.05.2026).
10. Werkzeug Documentation / Pallets. URL: https://werkzeug.palletsprojects.com/ (дата звернення: 07.05.2026).
11. SQLite Documentation / SQLite Consortium. URL: https://sqlite.org/docs.html (дата звернення: 07.05.2026).
12. SQLite Home Page / SQLite Consortium. URL: https://sqlite.org/ (дата звернення: 07.05.2026).

13. HTML: HyperText Markup Language / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 07.05.2026).
14. CSS: Cascading Style Sheets / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 07.05.2026).
15. JavaScript Reference / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (дата звернення: 07.05.2026).
16. Learn Web Development / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development (дата звернення: 07.05.2026).
17. Web Content Accessibility Guidelines (WCAG) 2.2 / World Wide Web Consortium. URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 07.05.2026).
18. Get Started with Bootstrap 5.3 / Bootstrap Team. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 07.05.2026).
19. Create events / Google Calendar API / Google Developers. URL: https://developers.google.com/workspace/calendar/api/guides/create-events (дата звернення: 07.05.2026).
20. Get Started / Todoist Help Center. URL: https://www.todoist.com/help/categories/get-started (дата звернення: 07.05.2026).
21. Trello Guides: Help Getting Started with Trello / Atlassian. URL: https://trello.com/guide (дата звернення: 07.05.2026).
22. To Do help & learning / Microsoft Support. URL: https://support.microsoft.com/uk-ua/todo (дата звернення: 07.05.2026).