Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»
(повне найменування вищого навчального закладу)

Відділення телекомунікацій та електронних систем
(назва відділення)

Циклова комісія комп'ютерної інженерії
(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної роботи

бакалавра
(освітній ступінь)

на тему: Розробка навчально-демонстраційного сайту «SkillUpCoding»

Виконав: студент VI курсу, групи КІб-602

Спеціальності **123 Комп'ютерна інженерія**
(шифр і назва, спеціальності)

Володимир БІЛИК
(ім'я та прізвище)

Керівник Володимир ЛІСОВИЙ
(ім'я та прізвище)

Рецензент
(ім'я та прізвище)

Тернопіль – 2025

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
імені ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерної інженерії
Освітній ступінь бакалавр
Освітньо-професійна програма: Комп'ютерна інженерія
Спеціальність: 123 Комп'ютерна інженерія
Галузь знань: 12 Інформаційні технології

ЗАТВЕРДЖУЮ
Голова циклової комісії
комп'ютерної інженерії
_____ Андрій ЮЗЬКІВ
"__" _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

__Білику Володимиру Сергійовичу_
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Розробка навчально-демонстраційного сайту «SkillUpCoding»

керівник роботи Лісовий Володимир Миколайович
(прізвище, ім'я, по батькові)

затверджені наказом Відокремленого структурного «Тернопільський фаховий коледж Тернопільського національного технічного університетут імені Івана Пулюя» від 05 травня.2025 р №4/9-217.

2. Строк подання студентом роботи: 20 червня 2025 року.

3. Вихідні дані до роботи: технічне завдання на створення сайту SkillUpCoding, структура вебсервісу, JSON-файли тестів, вимоги до функціоналу та API, макети сторінок, структура бази MongoDB.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Загальний розділ. Розробка технічного та робочого проєкту. Спеціальний розділ. Економічний розділ. Безпека життєдіяльності, основи охорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   - структурна схема інтерфейсу;
   - структурна схема інтерфейсу роботи зі Snippet;
   - блок схема роботи бази даних;
   - блок схема роботи Snippet ;
   - лістинг компоненту;
   - таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Оксана РЕДЬКВА заст. директора з НВР		
Охорона праці, техніка безпеки та екологічні вимоги	Володимир ШТОКАЛО викладач		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	08.05	
2	Збір і узагальнення інформації	20.05	
3	Написання першого розділу	23.05	
4	Розробка технічного та робочого проекту	28.05	
5	Написання спеціального розділу	3.06	
6	Розрахунок економічної частини	5.06	
7	Написання розділу охорони праці	6.06	
8	Виконання графічної частини	10.06	
9	Оформлення проекту	13.06	
10	Погодження нормоконтролю	17.06	
11	Попередній захист роботи	20.06	
12	Захист кваліфікаційної роботи		

7. Дата видачі завдання: 08 травня 2025 року

Студент _____ Володимир БІЛИК
(підпис) (ім'я та прізвище)

Керівник роботи _____ Володимир ЛІСОВИЙ
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Білик В. С. Розробка навчально-демонстраційного вебсайту SkillUpCoding: кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 123 «Комп'ютерна інженерія». Тернопіль: ВСП «ТФК ТНТУ», 2025. 154 с.

У кваліфікаційній роботі розроблено вебзастосунок «SkillUpCoding», який реалізує функціонал навчальної онлайн-платформи для вивчення основ програмування. Проєкт поєднує статичні сторінки з теоретичними матеріалами (HTML, CSS, JavaScript, Python, C#, алгоритми), інтерактивне тестування, систему облікових записів користувачів та функціонал спільного обміну кодовими фрагментами (snippets).

Клієнтська частина реалізована з використанням HTML5, CSS3, JavaScript. Серверна частина – на Node.js з Express, а збереження даних здійснюється через базу SQLite. Взаємодія реалізована через REST API. Проєкт підтримує JWT-аутентифікацію, валідацію форм та збереження результатів тестування. Застосунок адаптований для десктопних і мобільних пристроїв, з урахуванням вимог з безпеки, охорони праці та ергономіки IT-робочого місця. Хостинг здійснено на GitHub Pages (frontend) та Render.com (backend).

Проведено повний цикл тестування, а також виконано інженерний розрахунок системи освітлення приміщення для забезпечення належних умов праці розробника.

Ключові слова: вебзастосунок, навчальна платформа, JavaScript, Node.js, база даних, тестування, frontend, backend, snippets, безпека, авторизація, REST API.

ANNOTATION

V. Bilyk. Development of the educational-demonstration website “SkillUpCoding”: qualification work for obtaining a Bachelor's degree in Computer Engineering, specialty 123. Ternopil: SEI “TFK TNTU”, 2025. 154 p.

This bachelor’s thesis presents the development of the “SkillUpCoding” web application — an educational platform designed to assist users in learning programming fundamentals. The project combines static educational content (HTML, CSS, JavaScript, Python, C#, Algorithms), interactive testing functionality, a user account system, and a collaborative snippet-sharing feature.

The frontend is developed using HTML5, CSS3, and JavaScript. The backend is built with Node.js and Express, while data storage is handled through an SQLite database. The interaction between client and server is realized via REST API. The platform includes JWT-based authentication, form validation, score tracking, and full responsive design for both desktop and mobile users. The project is deployed via GitHub Pages (frontend) and Render.com (backend).

The work also includes system testing and a detailed engineering calculation of artificial lighting in the developer’s workspace to meet occupational safety standards and ergonomic requirements.

Keywords: web application, educational platform, JavaScript, Node.js, database, testing, frontend, backend, snippets, security, authentication, REST API.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ І СКОРОЧЕНЬ

HTML – HyperText Markup Language (мова розмітки гіпертексту)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

JS – JavaScript (мова сценаріїв)

Node.js – середовище виконання JavaScript на сервері

API – Application Programming Interface (програмний інтерфейс прикладного рівня)

JSON – JavaScript Object Notation (формат обміну даними)

JWT – JSON Web Token (токен для авторизації)

DB – Database (база даних)

SQLite – легковагова реляційна база даних

UI/UX – User Interface / User Experience (інтерфейс користувача / досвід користування)

LED – Light Emitting Diode (світлодіодний)

ВСТУП

Сьогодні все більше людей прагнуть вчитися програмувати та вдосконалювати свої навички. Від класичних текстових посібників і відеоуроків до інтерактивних вправ – веб-ресурсів для вивчення програмування зараз дуже багато. А ще обмін готовими фрагментами коду й обговорення рішень допомагають швидше опанувати новий матеріал., оскільки дають змогу обмінюватися готовими рішеннями та оперативно отримувати зворотний зв’язок.

Сайт поєднує теорію, тести та практичні фрагменти коду в одному місці – і це дає кілька важливих плюсів. Щоб, новачки мають змогу не лише вивчати основи програмування, а й одразу перевіряти свої знання завдяки коротким контрольним питанням. Також розділ для обміну кодом дає поштовх до колективного вдосконалення: кожен студент чи розробник може викладати власні напрацювання, коментувати чужі розв’язання та отримувати підтримку спільноти.

Тому вирішено зробити інтерактивну платформу, що допомагає поєднати навчання й практику в одній системі. У результаті користувачі отримають готові навчальні матеріали, можливість перевірити свої знання за допомогою тестів, а також інструментарій для збереження і спільного обговорення фрагментів програмного коду. Сайт робить навчання кращим: студенти краще розуміють матеріал і охочіше беруться за завдання.

ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

Було проведено аналіз найпопулярніших рішень у сфері онлайн-збереження коротких фрагментів коду, так званих «snippet-ів», і дійшов висновку, що така функціональність є справді необхідною для розробників будь-якого рівня – як початківців, так і досвідчених спеціалістів. Серед сервісів, що дозволяють оперативно зберігати, переглядати й ділитися уривками коду, можна назвати такі як Pastebin, GitHub Gist та інші подібні платформи. Вони стали стандартними інструментами для швидкої роботи з кодом завдяки своїй доступності, простоті інтерфейсу та підтримці підсвічування синтаксису. На мою думку, саме Pastebin і GitHub Gist найкраще представляють цей сегмент, адже дають змогу миттєво публікувати код у відкритому чи приватному режимі, отримувати прямі посилання для зручного поширення, а також здійснювати базову організацію контенту (див. рис. 1.1).

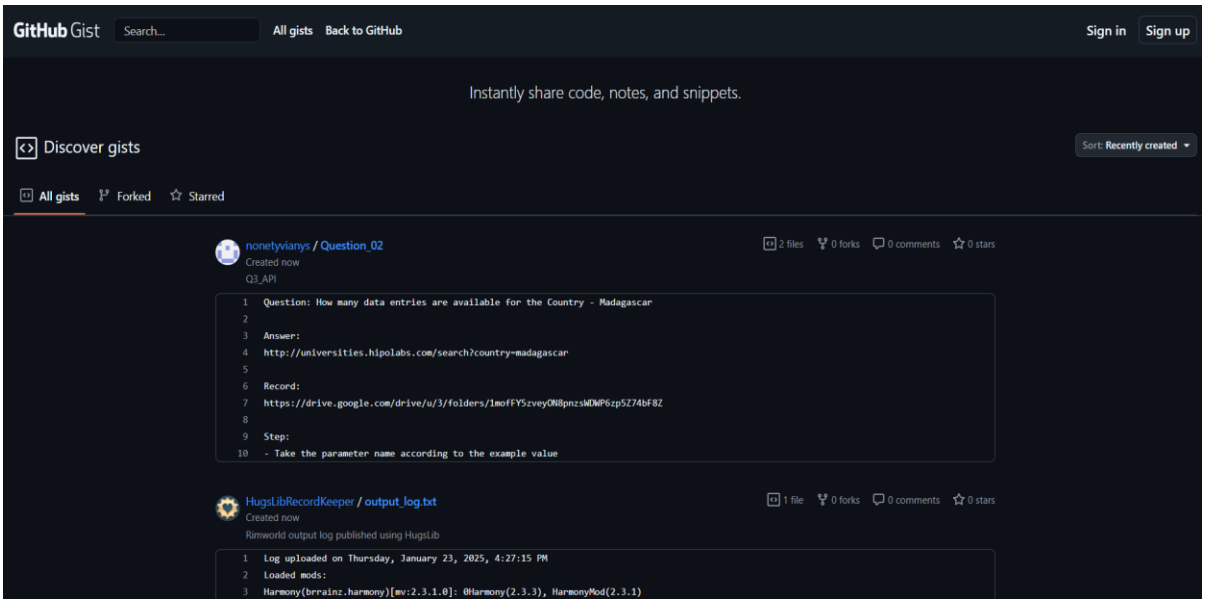


Рисунок 1.1 - GitHub Gist

Втім, незважаючи на свою популярність, ці сервіси мають суттєві обмеження. Вони зосереджуються переважно на функції збереження, але не забезпечують повноцінної взаємодії між користувачами. Наприклад, Pastebin надає лише мінімальний набір функцій – збереження уривку тексту, можливість обрати тип доступу (публічний або приватний), та базову організацію записів. Однак механізми впливу користувачів на контент, такі як система вподобань, рейтингів, сортування за популярністю чи гнучке коментування – відсутні. У GitHub Gist ситуація трохи краща завдяки тісній інтеграції з акаунтом GitHub, однак і тут основна мета полягає у роботі з репозиторіями, веденні проектів та обміні версіями файлів, а не у створенні повноцінної платформи для відкритої публікації snippet-ів із можливістю їх оцінювання та обговорення [2].

Існує ще один напрям використання сховищ для snippet-ів, а саме внутрішньокорпоративні системи. Їх часто налаштовують команди розробників або великі організації, щоб забезпечити приватний обмін кодом між працівниками. Такі системи інтегруються у внутрішні портали або середовища керування завданнями. Їх основною перевагою є конфіденційність – тобто код не виходить за межі внутрішньої мережі. Проте ці рішення також зазвичай мають обмежений функціонал: у них рідко можна побачити візуальні рейтинги найкращих snippet-ів, ефективний пошук за мовами програмування або систему коментарів, орієнтовану на відкриту дискусію [3].

У процесі аналізу зроблено висновок, що всі наявні платформи умовно поділяються на три основні підходи. Перший – це глобальні сервіси типу Pastebin та GitHub Gist, які орієнтовані на максимально швидке збереження коду. Вони чудово підходять для миттєвого обміну, але не мають розгорнутої взаємодії між користувачами. Другий підхід — це внутрішньоорганізаційні

рішення, які переважно зосереджені на захисті доступу, але жертвують відкритістю та гнучкістю функціоналу. Третій варіант — розробка власного вебресурсу, який дозволяє повністю контролювати як зовнішній інтерфейс, так і всі внутрішні процеси: від логіки авторизації до структурованого коментування, сортування та підсвічування коду.

Саме цей третій підхід вирішено реалізувати у кваліфікаційній роботі бакалавра. Створення власного вебсервісу для роботи зі snippet-ами дає змогу врахувати реальні потреби розробника: зручне додавання фрагментів, автоматичне підсвічування синтаксису, вказання мови програмування, рейтингова система, вподобання, сортування за параметрами, а також коментарі з підтримкою відповідей і обговорень. Завдяки такому рішенню можна забезпечити не просто сховище коду, а створити повноцінне мініспівтовариство, де користувачі не лише ділитимуться своїми snippet-ами, а й обговорюватимуть чужі, ставитимуть запитання, даватимуть поради й разом покращуватимуть якість представлених рішень.

Така платформа, орієнтована на навчання і взаємодію, значно розширює стандартні можливості типових snippet-сервісів і створює середовище, яке стимулює не лише обмін знаннями, а й постійне професійне зростання всередині спільноти.

1.2 Технічне завдання

Головною метою кваліфікаційної роботи бакалавра є створення навчально-демонстраційного вебсайту, який надає можливість користувачам з різним рівнем підготовки ознайомлюватися з основами програмування, виконувати інтерактивні тести, ділитися результатами, а також публікувати невеликі, але важливі фрагменти коду, які називаються snippet-ами. Цей сайт

не обмежується лише переглядом коду – він покликаний стати платформою для обміну досвідом, обговорень, а також розвитку навичок у новачків і закріплення знань у досвідченіших користувачів.

Навчальна частина вебсайту передбачає ознайомлення з ключовими мовами програмування, такими як HTML, CSS, JavaScript та Python. Тут не просто демонструється теоретичний матеріал, а надаються інтерактивні приклади, які користувач може виконувати прямо на сайті. Окрім цього, необхідно реалізувати систему створення й публікації snippet-ів. Кожен фрагмент коду супроводжується заголовком, вибором мови програмування і зручним полем для вставлення коду з підсвічуванням синтаксису. Після створення snippet можна буде побачити в загальному списку, відсортувати за датою, кількістю лайків або обраною мовою програмування.

Додаткову цінність ресурсу надає система коментування. Користувачі мають можливість залишати короткі коментарі до snippet-ів, вести діалоги й відповідати один одному у вигляді вкладених коментарів. Так буде реалізовано функціонал обговорення, який дозволятиме глибше аналізувати уривки коду, отримувати зворотний зв’язок і обмінюватися думками щодо оптимізації, помилок або альтернативних підходів до реалізації тієї чи іншої задачі [5].

На сайті необхідно передбачити повноцінну авторизацію з можливістю реєстрації за електронною адресою. При створенні облікового запису користувач вказує свою електронну пошту, пароль та, за бажанням, завантажує фото профілю. У своєму кабінеті можна змінювати особисту інформацію, переглядати історію доданих snippet-ів, бачити свої результати тестування, а також редагувати або видаляти додані матеріали.

Окремий функціональний блок присвячений тестуванню знань. Необхідно реалізувати кілька наборів питань, які охоплюють базові теми

програмування. Після проходження тесту система фіксує результат у вигляді кількості правильних відповідей і часу, витраченого на тест. Ці дані зберігаються в особистому профілі користувача, де він може переглядати свою історію тестування, стежити за прогресом і бачити, як покращуються його результати з часом [7].

Додаткову безпеку буде забезпечено перевіркою унікальності електронної адреси під час реєстрації, захищеним зберіганням паролів, валідацією введених даних і обмеженням доступу до певних функцій лише для авторизованих користувачів. Наприклад, додавати snippet або коментувати можна тільки після входу в систему. Також необхідно реалізувати фільтрацію шкідливого контенту в коментарях.

Важливо зазначити, що вебсайт не лише орієнтований на кінцевого користувача, а й передбачає зручність розгортання і технічної підтримки. Документація проекту включатиме докладну інструкцію з розгортання: зазначено, яку версію Node.js потрібно використовувати, які пакети потрібно встановити, як виглядає рекомендована структура директорій і яким чином запускати сервер. Усе це дозволяє швидко розгорнути сайт як у локальному середовищі, так і на хмарних платформах на кшталт Render чи Heroku.

Опис головного функціоналу також включає пояснення, як зареєструватися, увійти в систему, змінити свій профіль, додати snippet або видалити його. Вказано, яким чином працюватиме система тестів і як буде реалізовано збереження результатів. Також наведено приклади API-запитів (GET, POST, PUT, DELETE) для роботи з базою даних, що дає змогу розробникам швидко зрозуміти логіку системи [8].

Витрати на реалізацію проекту, за попередніми оцінками, не перевищують 30 000 гривень, включаючи хостинг, розгортання середовища і базову підтримку. Загальний обсяг роботи буде становити приблизно 120

годин, що включає дизайн, верстку, серверну логіку, роботу з базою даних, інтеграцію API, тестування і розгортання. Завдяки цьому сайт може стати доступним рішенням для освітніх закладів або приватних курсів з мінімальними витратами. Окупність проекту прогнозується в межах двох-трьох років, особливо за умови поширення платформи серед студентів, викладачів або через партнерства з навчальними ініціативами.

У результаті вебресурс не тільки охоплюватиме навчальний контент, а й стимулюватиме взаємодію між користувачами через публікацію коду, обговорення, лайки, коментарі та особистий прогрес у навчанні. Це створюватиме повноцінну екосистему для самоосвіти і вдосконалення навичок програмування.

1.2.1 Найменування та область застосування

Темою кваліфікаційної роботи бакалавра є створення навчально-демонстраційного сайту під назвою «SkillUpCoding». Це вебплатформа, основна мета якої — сприяти вивченню основ програмування, особливо серед початківців, студентів, слухачів курсів, а також усіх охочих, хто прагне вдосконалити свої навички у зручному форматі. Назва «SkillUpCoding» недаремно включає слово "Skill", адже акцент у цьому проекті зроблено саме на розвитку конкретних практичних умінь. Сайт дає змогу одночасно вчитися, закріплювати знання на практиці і взаємодіяти з іншими користувачами через публікацію та обговорення фрагментів коду.

Платформа «SkillUpCoding» охоплює кілька напрямів практичного застосування. У першу чергу, сайт може використовуватися в освітній сфері. Наприклад, у коледжах, технікумах чи університетах, де викладачі або ментори можуть запропонувати студентам виконувати інтерактивні завдання

та проходити тести. Крім того, вони можуть публікувати демонстраційні приклади коду, які учні зможуть переглядати, коментувати або покращувати. Такий підхід дозволяє не просто ознайомлюватися з теорією, а формувати практичні навички програмування через активну взаємодію.

Ще один напрямок — індивідуальне навчання. Багато користувачів, які самостійно опановують програмування, шукають платформи, де можна практикуватися, бачити чужі приклади та ділитися своїми. Сайт якраз надає таку можливість: користувач має змогу не лише проходити тести чи вивчати розділи з навчальним контентом, а й опублікувати власний snippet — невеликий фрагмент коду, який він вважає корисним або хоче обговорити з іншими. Це дозволяє новачку не лише вчитися, а й одразу застосовувати отримані знання на практиці, перевіряти себе та отримувати зворотний зв’язок [6].

Також сайт може стати у пригоді для командних проектів, спільнот чи гуртків з програмування, де учасники можуть зручно обмінюватися робочими фрагментами коду. Це може бути корисно для мініхакатонів, воркшопів або навіть у рамках невеликих курсів, де необхідно швидко показати своє рішення іншим учасникам. Оскільки на сайті буде реалізовано сортування snippet-ів, можливість ставити лайки, залишати коментарі, вести діалог — це зробить платформу придатною для обговорення альтернативних рішень і підвищення якості написаного коду.

Загальна універсальність платформи полягає в тому, що нею зможуть користуватися як новачки, які роблять свої перші кроки у вивченні HTML, CSS чи JavaScript, так і більш досвідчені програмісти, які бажають структурувати власні напрацювання, вести архів фрагментів, а також поширювати свої "best practices" серед спільноти. Завдяки цьому «SkillUpCoding» охоплюватиме не лише вузьке коло користувачів, а насправді здатен служити освітнім

інструментом для різних категорій аудиторії: від початкового рівня до просунутого.

1.2.2 Призначення розробки

Мета розробки вебсайту полягає у створенні зручної та інтерактивної онлайн-платформи, яка допомагає користувачам різного рівня засвоювати основи програмування та закріплювати отримані знання на практиці. Головна ідея полягає у поєднанні декількох важливих функцій в одному інструменті: теоретичний матеріал, практичні приклади коду, тестування знань, а також можливість ділитися власними розробками та обговорювати їх з іншими учасниками спільноти.

Цей підхід забезпечить комплексне навчання. По-перше, сайт пропонуватиме структурований освітній контент, який охоплює основні розділи: мови розмітки HTML та CSS, основи програмування на JavaScript, базові концепції алгоритмів, а також Python. Кожен з цих розділів включатиме теоретичні пояснення та прості приклади, які користувачі зможуть вивчати самостійно. Весь матеріал буде подаватися послідовно, відповідно до логіки зростаючої складності — від простого до складного [4].

По-друге, користувачі матимуть можливість проходити інтерактивні тести, які дозволять перевірити засвоєний матеріал. Після виконання тестів система автоматично підрахує правильні відповіді, покаже загальний бал, а також надасть зворотний зв’язок, що дозволятиме зрозуміти, які теми потребують повторного опрацювання. Кожен тест буде прив’язаний до конкретного розділу або теми, що дозволятиме здійснювати цільову перевірку знань, не витрачаючи час на повторення того, що вже добре засвоєно.

Окрім тестів і теорії, важливою складовою сайту буде розділ snippet-ів. У цьому розділі кожен користувач зможе опублікувати короткий фрагмент коду з поясненням, зазначенням мови програмування та можливістю отримати зворотний зв’язок від інших. Всі snippet-и будуть виводитися у вигляді карток, де автоматично застосовується підсвічування синтаксису, що значно покращує читабельність. Користувачі зможуть ставити вподобання, коментувати snippet-и, відповідати один одному, а також шукати потрібні приклади за ключовими параметрами, як-от мова програмування чи дата додавання. Таким чином, кожен snippet перетвориться не лише на код, а на точку початку обговорення та навчального діалогу.

Ще один важливий елемент — це особистий профіль користувача. Кожен зареєстрований учасник платформи матиме змогу переглядати свій прогрес, відслідковувати результати проходження тестів, кількість лайків та коментарів під своїми snippet-ами. Це створить ефект мотивації, оскільки користувач бачить, як росте його активність і вплив у спільноті. Крім того, у профілі буде можна змінювати особисту інформацію, додавати фото, редагувати короткий опис «Про себе» та, за потреби, змінювати пароль.

Таким чином, сайт «SkillUpCoding» поєднуватиме навчальні матеріали, тестову перевірку, можливість оприлюднювати та покращувати фрагменти коду й механізми для соціальної взаємодії — усе це зробить платформу не лише інструментом для засвоєння знань, а справжнім середовищем для професійного росту й обміну досвідом.

1.2.3 Вимоги до функціоналу вебсайту

Функціональні вимоги до вебсайту охоплюють декілька ключових напрямків, які разом формують цілісну систему. Однією з найважливіших

частин є система користувачів. Кожен відвідувач сайту зможе пройти реєстрацію, вказавши унікальну електронну адресу, яка перевіряється на дублікати для забезпечення індивідуальності кожного облікового запису. Після створення акаунта користувач отримуватиме доступ до особистого профілю, де зможе редагувати особисту інформацію, змінювати пароль або адресу електронної пошти, а також додавати фото та короткий опис «Про себе». Це створить відчуття персоналізованої участі в платформі.

Іншою важливою складовою буде функціонал тестування. Система передбачає наявність кількох тестів, що охоплюють різні напрямки програмування, наприклад HTML/CSS чи JavaScript. Кожен тест зберігатиме результат — як відсотковий бал, так і час, витрачений на проходження. У профілі користувача доступна історія попередніх результатів, що дозволятиме відстежувати власний прогрес і визначати теми, які варто повторити. Такий підхід мотивуватиме до саморозвитку та дасть можливість об’єктивно оцінювати успішність навчання.

Основну цінність платформи створюватиме можливість працювати зі snippet-ами — короткими уривками коду, які можна публікувати, переглядати, сортувати й обговорювати. Кожен snippet створюватиметься через спеціальну форму, де користувач вказує заголовок, вставляє код і обирає мову програмування. Після додавання snippet автоматично потраплятиме у список загальнодоступних публікацій, де його можуть бачити інші користувачі. Підсвічування синтаксису забезпечить зручне читання коду. Додатково буде реалізована можливість редагувати або видаляти snippet, якщо це публікація автора. Також будуть реалізовані опції сортування: наприклад, можна переглядати snippet-и за кількістю лайків, датою публікації або фільтрувати їх за мовою програмування, що значно підвищує зручність пошуку.

Важливою частиною буде соціальна взаємодія між користувачами. Для цього кожен snippet підтримуватиме функцію «лайків», які відображатимуть популярність публікації. Крім того, буде передбачено механізм коментування, де кожен учасник зможе залишити відгук, поставити запитання або відповісти на чужий коментар. Така система обміну думками створить основу для навчального діалогу та взаємної допомоги.

Усі результати, пов’язані з тестами, фіксуються та зберігаються у відповідному розділі профілю. Користувач зможе переглянути свої бали та час проходження для кожного тесту окремо. Якщо новий результат кращий за попередній, система замінить старі дані, щоб залишити лише найактуальніші показники. Це дозволить зосередитись на прогресі та покращенні навичок.

Для підтримки порядку та безпеки буде реалізовано адміністративні інструменти. Вони включатимуть перевірку правильності введених даних у формах, контроль за діями користувачів та можливість виявлення помилкових або потенційно небезпечних запитів. Також беде передбачена можливість додавання нових маршрутів, якщо у майбутньому виникне потреба у розширенні функціоналу.

Таким чином, вебсайт охоплюватиме повний набір функцій, необхідних для навчання, самоперевірки та взаємодії у спільноті. Усе це забезпечить не лише комфортне середовище для програмування, а й повноцінну платформу для обміну знаннями та досвідом.

1.2.4 Вимоги до програмної документації

Для забезпечення повноцінного використання і підтримки вебсайту важливо мати зрозумілу, структуровану програмну документацію. Вона має охоплювати кілька важливих аспектів: від базових інструкцій для запуску, до

пояснень реалізованої логіки та опису архітектури. Документація повинна починатися з покрокової інструкції щодо розгортання проекту. У цьому розділі необхідно зазначити мінімальні вимоги до середовища, наприклад, яку саме версію Node.js слід встановити, які додаткові пакети треба підключити через npm і яку структуру папок варто підтримувати. Рекомендується розділяти фронтенд і бекенд по окремих директоріях, а також створити окрему папку для завантажень, якщо передбачається обробка зображень або інших файлів. В інструкції також слід пояснити, як запускати сервер, яку команду використовувати — наприклад, npm start, і як перевірити коректність підключення до бази даних.

Другий важливий розділ повинен містити пояснення головного функціоналу. Наприклад, як користувач може зареєструватися, увійти до системи, змінити інформацію про себе, додати snippet або пройти тест. Якщо передбачено завантаження або редагування фотографій профілю, ці дії також мають бути описані. Так само слід надати базове роз’яснення, як адміністратор або розробник може редагувати тестові запитання чи створювати нові. Це особливо важливо, якщо тести мають бути конфігуровані вручну або мають складну структуру.

Окрему увагу варто приділити опису структур даних і використовуваних моделей. У цьому блоці треба описати, які саме схеми використовуються в базі даних. Наприклад, схема користувача повинна містити поля для імені, електронної адреси, пароля, аватарки та, можливо, додаткових параметрів. Модель snippet-а включає назву, сам код, зазначену мову, кількість лайків, коментарі, а також ID автора. Якщо присутні тести або результати проходження тестів, потрібно описати, як вони зберігаються: яка структура JSON, які поля є обов’язковими, як прив’язується користувач до результатів.

Ще одним блоком програмної документації мають бути лістинги основних частин коду. Тут доречно наводити вивід вмісту основних файлів — наприклад, конфігурації серверного додатку (server.js), прикладів маршрутів, а також частин frontend-функціоналу. Кожен фрагмент має бути коротко прокоментований, щоб було зрозуміло, яку саме функцію він виконує. Наприклад, для frontend можна показати, як працює завантаження snippet-а або валідація форми, а для backend — обробка запитів до API, перевірка токенів авторизації.

Крім того, необхідно згадати адміністративні моменти, пов’язані з обслуговуванням та безпекою. Варто пояснити, як виконуються оновлення системи, які кроки треба зробити при зміні схеми бази даних або при додаванні нового функціоналу. Також варто передбачити процес резервного копіювання — особливо важливо зберігати критичні дані, такі як результати тестів та користувацькі snippet-и. У розділі про безпеку потрібно зазначити, де і як зберігаються паролі (наприклад, у хешованому вигляді за допомогою bcrypt), як буде реалізовано захист від несанкціонованих запитів та що робити у випадку виявлення небажаних або підозрілих дій користувачів.

Наявність такої документації дозволить не лише легко розгорнути і обслуговувати вебпроект, а й забезпечить зрозумілу підтримку як для нових розробників, так і для технічного персоналу в майбутньому.

1.2.5 Техніко-економічні показники

Для обґрунтування доцільності створення навчально-демонстраційного вебресурсу «SkillUpCoding» необхідно розглянути техніко-економічні аспекти проекту. Ці показники охоплюють як часові, так і фінансові параметри

реалізації й утримання системи, а також прогнозують її ефективність з погляду використання.

Орієнтовна загальна тривалість розроблення проекту не повинна перевищувати 120 годин. У цей проміжок входить не лише створення графічного інтерфейсу користувача (frontend) та побудова логіки сервера (backend), але й налаштування та оптимізація бази даних, реалізація тестового функціоналу та системи публікації snippet-ів. Окрім того, у зазначений час включається і фінальний етап: тестування, виявлення помилок та усунення знайдених недоліків, а також перевірка функціонування на різних пристроях.

Вартість впровадження ресурсу залежить від низки факторів, зокрема вибору хостингу, складності середовища розробки, а також залучення сторонніх інструментів. Однак навіть з урахуванням усіх цих змінних повна вартість створення і налаштування системи не перевищує 30 000 гривень. Така сума включає витрати на хмарні сервіси (за необхідності), оплату розробницької роботи та використання базових ресурсів. При цьому є можливість мінімізувати фінансові витрати, використовуючи безкоштовні або бюджетні тарифи хостинг-провайдерів, які повністю задовольняють потреби проекту на початкових етапах з невеликою кількістю користувачів.

Що стосується терміну окупності, за умови належної презентації платформи та її популяризації серед цільової аудиторії (зокрема студентів, викладачів або початківців у сфері програмування), очікується повернення вкладених ресурсів протягом 2–3 років. Цей термін може бути скорочений за рахунок активного просування платформи, впровадження партнерських моделей із закладами освіти або продажу доступу до розширеного функціоналу.

Варто також врахувати потенційні джерела монетизації, які можуть забезпечити регулярний дохід. Наприклад, користувачам можна

запропонувати платну підписку на додаткові функції — поглиблену аналітику результатів тестування, розширений набір прикладів, нові тематичні блоки або розгорнуту систему рекомендацій. Крім того, можливе впровадження партнерських моделей із навчальними платформами або IT-компаніями, що дозволить отримувати доходи від реклами або ліцензійних угод.

Усі ці аспекти демонструють, що реалізація вебсайту «SkillUpCoding» має не лише навчальну цінність, але й економічну доцільність, адже сукупні витрати залишаються в межах обґрунтованого бюджету, а потенційна вигода та окупність у майбутньому виглядають цілком реалістично.

1.2.6 Стадії та етапи розробки

Розробка вебсайту «SkillUpCoding» відбуватиметься поетапно, з поступовим переходом від аналітики та проектування до реалізації та впровадження. На початковому етапі визначені ключові завдання, які сайт має виконувати, та функціональні модулі, необхідні для досягнення цілей. Буде проведено аналіз подібних сервісів, окреслено цільову аудиторію та зібрано технічні й навчальні вимоги. Саме в цей період сформується загальна логіка системи, зокрема її структура, типи користувачів, типи контенту (snippet-и, тести, профілі), а також способи взаємодії між компонентами. Паралельно буде вестися розробка структури бази даних, з акцентом на збереження тестових результатів, коментарів і коду.

Після аналітики та планування розпочнеться налаштування робочого середовища. Буде встановлено Node.js, обрано відповідну базу даних — MongoDB як оптимальний варіант для зберігання гнучких документів. Для зручності візуалізації та підсвічування коду будуть підключені зовнішні бібліотеки, як-от highlight.js, що дозволять покращити вивід snippet-ів на

сторінці. Фронтенд-частина проекту буде ініціалізована окремо з підключенням до основного сервера. Після базових налаштувань середовище буде повністю готове до написання коду.

Фронтенд реалізовуватиметься з урахуванням зручності для кінцевого користувача. Буде створено сторінки для реєстрації, входу та профілю користувача. Усі форми передбачатимуть перевірку введених даних і реакцію на помилки. Також необхідно реалізувати модулі для роботи з snippet-ами — виведення списку, створення нового, редагування, підрахунок лайків і коментарів. Для навчальної частини буде створено кілька сторінок з теоретичними блоками, інтерактивними тестами та системою перевірки правильності відповідей. Зокрема, буде впроваджено механізм миттєвого підрахунку балів після проходження тесту та відображення рекомендацій на основі отриманого результату.

Паралельно з фронтендом будуватиметься серверна логіка. Основні маршрути будуть реалізовані за допомогою Express.js. Буде налаштовано обробку запитів на створення користувача, авторизацію, оновлення даних профілю, а також маршрути для роботи з snippet-ами — зокрема, можливість додати, змінити або видалити фрагмент. Додатково буде реалізовано логіку фіксації тестових результатів, підрахунку балів та збереження результатів у базі даних для подальшого перегляду у профілі.

Після завершення основної розробки проект пройде попереднє тестування. Основну увагу буде приділено перевірці коректності логіки: чи зберігаються дані після реєстрації, чи працює система додавання та редагування snippet-ів, як обробляються помилкові запити, і чи правильно працює механізм тестування. Знайдені помилки будуть виправлені, а також буде здійснено оптимізацію коду для зменшення затримок при обміні даними з сервером.

Фінальним етапом стане розгортання сайту. Буде обрано хостинг-платформу Render для розміщення сервера й бази даних, а також протестовано роботу сайту на різних пристроях. У систему буде додано базові навчальні матеріали та декілька прикладів snippet-ів і тестів, щоб надати стартовий контент для користувачів. При потребі також можливе підключення SSL-сертифіката, щоб забезпечити безпечну передачу даних. Таким чином, проект вийде на фінальну стадію готовності до експлуатації та подальшого розвитку.

1.2.7 Порядок тестування та прийому

Тестування вебсайту «SkillUpCoding» буде проводитися поетапно, із залученням як внутрішніх перевірок, так і зовнішнього фідбеку. Насамперед буде проведена перевірка всього функціоналу на предмет відповідності технічному завданню. Особлива увага приділятиметься коректності роботи ключових модулів, таких як реєстрація користувача, авторизація, додавання фрагментів коду, а також проходження тестів. Перевіриться, як обробляються всі введені дані у формах, чи функціонує захист від некоректного або неповного введення, чи правильно зберігається інформація у базі даних.

Після цього буде протестовано відображення сайту на різних пристроях, зокрема на екранах ноутбуків, планшетів і смартфонів. Тестування проводитиметься у популярних браузерах, серед яких Google Chrome, Mozilla Firefox і Safari. Особливу увагу зверну на правильне виведення коду із підсвічуванням синтаксису, щоб користувачі могли комфортно переглядати навіть складні фрагменти у зручному вигляді.

Окремо будуть перевірятися тести, вбудовані у платформу. Проводитимуться різні сценарії проходження тестів: з повністю правильними відповідями, з випадковими помилками, а також із частковим заповненням.

Після кожного проходження буде контролюватися механізм підрахунку балів, фіксації часу й виведення результату на екран. Важливо буде перевірити, як система реагує, якщо користувач припиняє тест посеред процесу — і чи не виникає при цьому логічних або візуальних збоїв.

Щоб переконатися в зручності інтерфейсу для кінцевого користувача, до тестування будуть залучені сторонні особи — студенти, колеги та інші зацікавлені. Їм буде запропоновано самостійно пройти реєстрацію, переглянути доступні snippet-и, створити власний фрагмент коду та спробувати пройти кілька тестів. Завдяки цьому вдасться зібрати живий фідбек щодо зручності користування, помічених неточностей або можливих покращень.

Завершальним етапом тестування стане презентація готового функціоналу замовнику або комісії. Буде продемонстровано усі основні розділи платформи, включаючи сторінки реєстрації, додавання snippet-ів, коментування, а також перегляд тестових результатів у профілі.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

2.1 Розробка структури сайту і вебсторінок

Для успіху SkillUpCoding було вирішино зосередитися на зрозумілій структурі та логічному розташуванні сторінок. Добре продумана структура задає порядок у навігації та полегшує користувачам роботу з сайтом, його орієнтація у функціоналі, сприйняття контенту та ефективність навчального процесу загалом.

SkillUpCoding створений, щоб навіть повні новачки могли легко почати вивчати програмування. Він має бути простим у користуванні, інтуїтивно зрозумілим, візуально приємним, а також функціональним з технічної точки зору. Було вирішино розділити сайт на тематичні сторінки за мовами та темами, кожна з яких відповідає певному напрямку навчання — мові програмування або супутній темі. Так користувачі не заплутаються і легко знайдуть потрібний контент.

З точки зору функціонального наповнення та зручності користування, на сайті реалізувано низку ключових елементів, які є стандартом сучасних вебресурсів, що орієнтовані на користувача. Серед них — шапка сайту з логотипом і навігаційним меню, яке трансформується в оф-канвасне для мобільної версії; активна панель входу/реєстрації користувачів; окремий розділ з профілем зареєстрованого користувача; функціональні блоки на кожній сторінці; модальні вікна для тестування; кнопки для основних дій користувача та футер із соціальними посиланнями і відомостями про автора.

У загальній структурі сайту виділяється головна сторінка, яка виконує роль стартового екрану. Вона містить назву ресурсу — SkillUpCoding, яка слугує і логотипом, і лінком до індексної сторінки. На головній сторінці також

передбачив короткий опис призначення сайту, візуальні елементи для залучення уваги відвідувача, а також навігаційні кнопки для швидкого переходу до розділів.

Меню на комп’ютері й у мобільній версії дає швидкий доступ до головної сторінки (index.html), сторінки з фрагментами коду (snippets.html), а також до окремих навчальних розділів: алгоритми (algorithms.html), HTML/CSS (html-css.html), JavaScript (javascript.html), Python (python.html) та C# (csharp.html).

Усі сторінки побудовані за однією схемою. Кожна містить заголовок з назвою мови або розділу, вступний текст з поясненням важливості теми, базові теоретичні відомості, приклади коду з коментарями, списки ключових понять і правил, завершальний підрозділ з підсумками, а також інтерактивну кнопку для запуску тесту. Після натискання відкривається модальне вікно з тестовими питаннями, а після проходження тесту — відповідне повідомлення з результатом.

Кожна сторінка також включає кнопки для входу та реєстрації. Для цього реалізовано два модальні вікна: одне — для входу в акаунт, друге — для створення нового користувача. У формі реєстрації передбачено поля для введення імені, електронної адреси та пароля. Форми мають перевірку введених значень, щоб уникнути помилок при реєстрації. Після авторизації відкривається доступ до особистого профілю, де можна переглядати власні дані, змінювати інформацію «Про себе», завантажувати фото профілю, переглядати результати тестів і вийти з акаунту.

У модальному вікні профілю передбачена також можливість перегляду історії проходження тестів, що дозволяє користувачеві стежити за власним прогресом, аналізувати теми, які вже були пройдені, і за потреби повторно повернутися до них.

Питання завантажуються з окремого JSON-файлу tests.json, у якому збережені запитання, варіанти відповідей і правильні варіанти для кожного розділу. Це спрощує масштабування функціоналу та оновлення бази питань без потреби редагувати HTML-код.

Було зроблено сайт повністю адаптивним — усі елементи інтерфейсу (меню, тести, профіль, блоки з текстом і кодом) відображаються коректно на різних пристроях: від комп'ютерів до смартфонів. Для цього застосовано адаптивну верстку з використанням CSS-медіазапитів та мобільного меню, яке з'являється по натисканню на кнопку-бургер.

Клієнтська частина сайту інтегрується з серверною логікою: дані користувачів, результати тестів та зображення зберігаються в базі даних. Уже на етапі створення структури закладено основу для передачі запитів між фронтендом і бекендом, врахувавши, що система має зберігати персональні дані, фото профілю, відповіді на тести.

Для завершення логіки структурування сайту створено загальну структурну схему інтерфейсу, у якій визначив основні сторінки, зв'язки між ними, послідовність навігації та залежності між клієнтськими компонентами. Такий підхід дозволяє легко масштабувати сайт та надалі реалізовувати повноцінну логіку, яку буде описана в наступних розділах.

2.2 Створення та верстка сторінок сайту

Процес створення сайту SkillUpCoding почався із практичного втілення раніше спроектованої структури. На цьому етапі створено кожну окрему вебсторінку, наповнено її контентом, оформлино за допомогою HTML та CSS, а також забезпечено її повну адаптивність під різні типи пристроїв. Важливо було досягти високої якості не лише з точки зору зовнішнього вигляду, але й

з точки зору користувацького досвіду, доступності, логіки розміщення елементів та швидкості завантаження.

Верстку виконав за допомогою HTML5, застосовуючи семантичні теги для зручності навігації та кращої індексації. Усі стилі було винесено в окремий файл styles.css, що значно спростило процес редагування зовнішнього вигляду. Такий підхід дозволив централізовано управляти дизайном і повторно використовувати CSS-класи.

Особливу увагу було приділино мобільній адаптивності. Вона реалізована через використання медіа-запитів у CSS та відсоткового позиціонування блоків. Завдяки єдиній системі CSS-класів і компонуванню блоків за принципами модульності, вдалося уникнути дублювання стилів. Кожен блок сайту структурувався у вигляді окремої секції, що полегшувало орієнтування в коді. Колірна схема, відступи та розміри елементів керувалися через змінні, що зробило оформлення більш гнучким і зручним для подальшого редагування. При цьому розташування елементів відповідало сучасним практикам UX-дизайну.

Кожна сторінка сайту має спільну структуру. У верхній частині розташована шапка з логотипом та навігаційним меню — як у вигляді класичної десктопної панелі, так і як оф-канвас для мобільної версії. Основна частина містить заголовок розділу, текстові пояснення, приклади коду, списки понять та рекомендацій. Також передбачено кнопку запуску тесту, яка відкриває модальне вікно з інтерактивним тестуванням. Крім цього, на кожній сторінці доступне модальне вікно входу та реєстрації, а також вікно профілю користувача з можливістю редагування інформації. Футер сайту включає соціальні посилання, короткий опис автора проекту та рік створення.

Для реалізації повного функціоналу було створено окремі HTML-файли для кожного з навчальних напрямків. Серед них: головна сторінка index.html,

сторінка з фрагментами коду snippets.html, розділ про алгоритми algorithms.html, блок з вивчення HTML/CSS — html-css.html, окрема сторінка для JavaScript — javascript.html, навчальний розділ з Python — python.html, а також сторінка з основами мови C# — csharp.html.

Контент на кожній сторінці відповідає темі: розміщено теоретичні пояснення, приклади з коментарями, списки понять, блоки з кодом, а також тестові запитання. Хоча наповнення відрізняється, структура залишається сталою. Це робить навігацію передбачуваною і зручною для користувача.

Оформлення сторінок реалізувано так, щоб воно було легким для сприйняття і не перевантажувало увагу. Для цього використано контрастні заголовки, структуровані списки, підсвічування коду з фіксованим шрифтом, кольорове виділення додаткової інформації, а також блоки з порадами, які відрізняються візуально.

Адаптивна верстка дозволяє сайту виглядати коректно на всіх пристроях: смартфонах, планшетах, ноутбуках і десктопах. Особливу увагу приділино мобільній версії. Для навігації використано оф-канвас меню, що відкривається натиском на іконку «бургер». Це меню перекриває головний контент і дозволяє легко перемикатися між сторінками. Його закриття реалізовано як по натисканню на кнопку «×», так і по натисканню за межами меню.

Форми входу та реєстрації викликаються через модальні вікна, доступні з будь-якої сторінки. Було реалізовано перевірку валідності email-адрес і відповідність паролів. Після авторизації користувач бачить кнопку «Профіль», що відкриває модальне вікно з інформацією про себе, результатами тестів та можливістю завантаження фото профілю.

Візуальна тема сайту витримана в єдиному стилі: використано шрифт Montserrat, акцентні кольори для важливих елементів (кнопок, заголовків),

світлий фон, контрастні секції, ефект тіні для блоків та плавну анімацію відкриття вікон.

Flexbox і Grid-сітка дозволили точно розташовувати елементи в контейнерах незалежно від розміру екрану. Модальні вікна відкриваються поверх сторінки із затемненням фону, що фокусує увагу користувача на головному вмісті.

Окремо реалізовано кнопку повернення до початку сторінки у вигляді стрілки, що з'являється при прокрутці. Вона фіксується в нижньому правому куті й плавно прокручує сторінку до верху.

Особливу увагу приділино реалізації блоку тестування. Після натискання кнопки «Перевір вивчене» відкривається модальне вікно з динамічним завантаженням тесту з файлу tests.json. Питання відображаються за допомогою JavaScript, а після завершення тесту користувач бачить підсумок з кількістю правильних відповідей. Якщо користувач авторизований, результат зберігається у його профілі.

Реалізовано окрему структуру в JSON-файлі для кожної навчальної сторінки, що дозволяє без змін в HTML додавати нові тести чи оновлювати запитання.

Також продумано інтеграцію верстки з майбутньою серверною логікою: кожен елемент має унікальний ID, що дозволяє легко звертатися до нього з JavaScript. Завдяки цьому можливо без змін у HTML додавати функціональність через JS.

2.3 Розробка структури бази даних

Для реалізації сайту SkillUpCoding обрано сучасну, масштабовану та продуктивну систему зберігання даних — документно-орієнтовану базу

MongoDB. Її переваги в динамічних вебзастосунках очевидні: вона дозволяє гнучко працювати з даними без складних зв’язків, спрощує масштабування, природно поєднується з JavaScript-оточенням, яке використовується і на фронтенді, і на бекенді.

На відміну від реляційних баз, де дані організовано у таблицях, у MongoDB структура побудована на основі колекцій документів. Це дає змогу легко додавати, оновлювати й видаляти записи без необхідності дотримання жорстких схем або реалізації складних операцій об’єднання (join). Кожен об’єкт вирішино зберігати у форматі BSON (розширений JSON), що підтримує більше типів даних і оптимізований для обробки.

Відштовхуючись від функціональних потреб сайту, визначено три основні сутності: користувачі, результати тестування та сесії. У структурі користувача зберігається облікова інформація, включно з іменем, електронною адресою, зашифрованим паролем, а також додаткові дані на кшталт біографії, фото профілю, ролі (наприклад, звичайний користувач або адміністратор) і масиву результатів тестування. Усі ці поля разом формують повний профіль користувача, дозволяючи системі гнучко працювати з персональними даними, забезпечуючи при цьому належний рівень безпеки.

Кожен тестовий результат у структурі користувача зберігається як окремий об’єкт із зазначенням розділу, в якому проходився тест (наприклад, «python» чи «html-css»), кількості правильних відповідей, загальної кількості запитань і дати проходження. Це дозволяє легко формувати історію навчання й відображати її в особистому кабінеті користувача.

Сесії або авторизаційні токени зберігаються окремо. Для кожного такого запису фіксується, кому належить токен (через userId), його строкове значення, а також дати створення й закінчення дії. Це необхідно для авторизації запитів і збереження безпечного середовища при взаємодії з API.

Усі схеми бази даних побудовані за принципом контролю типів даних і обов’язкових полів. Наприклад, поле email проходить валідацію на унікальність і правильний формат. Паролі зберігаються не у відкритому вигляді, а у вигляді хешу, створеного за допомогою бібліотеки bcrypt. Фото профілю не зберігається в базі безпосередньо — лише посилання на нього, яке веде до зовнішнього хмарного сервісу, зокрема Cloudinary.

Архітектура бази враховує ключові принципи: уникається дублювання даних, реалізовано можливість змінювати структуру без порушення функціоналу, передбачено легке додавання нових полів або розширення. Усі важливі поля індексуються (зокрема, email та userId), що забезпечує швидкий пошук і вибірку.

Для зв’язку між колекціями використовується ObjectId — внутрішній унікальний ідентифікатор MongoDB. Наприклад, результат тесту можна пов’язати з конкретним користувачем або зберегти його в окремій колекції, зберігаючи при цьому зв’язок через відповідний ID. Це підвищує масштабованість системи і дає змогу ефективно опрацьовувати великі об’єми даних.

Для взаємодії з MongoDB було застосовано бібліотеку Mongoose. Вона дозволяє створювати моделі (Schemas) з вбудованою валідацією, перевіркою типів та автогенерацією timestamp-полів на кшталт createdAt і updatedAt. Кожна сутність має власний файл моделі — наприклад, models/User.js, models/Result.js, models/Session.js. Підключення до бази здійснюється через connection.js, де використфно mongoose.connect() з посиланням на URI бази, що зберігається у .env-файлі.

Безпека при роботі з даними — важливий елемент. Паролі ніколи не зберігаються у відкритому вигляді. Використано bcrypt для створення хешу, а для перевірки — методи порівняння хешів. Авторизація реалізується через

JWT або систему сесій, і вся взаємодія з базою відбувається лише через захищені маршрути API.

На етапі розробки також враховано можливі розширення: додавання рейтингової системи, розширених профілів, медалей за активність чи історії змін. Завдяки гнучкості MongoDB усі ці можливості можна впровадити без потреби в повному переробленні існуючої структури.

2.4 Програмування сайту

Для реалізації SkillUpCoding самостійно створино як фронтенд, так і бекенд — повноцінну клієнт-серверну систему, яка обробляє дані, зберігає інформацію, виконує авторизацію та забезпечує коректну взаємодію між користувачем і вебінтерфейсом. Щоб підтримувати зрозумілу архітектуру, розділино логіку на дві окремі частини: інтерфейсну, яка відповідає за зовнішній вигляд і взаємодію, та бізнес-логіку, яка реалізована на серверній стороні.

Клієнтська частина побудована на базі HTML5 з використанням зовнішніх CSS- і JavaScript-файлів. Кожен розділ сайту — наприклад, Python, JavaScript, HTML/CSS — представлений окремим HTML-файлом, що повторює загальну шаблонну структуру, але містить індивідуальний навчальний контент. Такий підхід дозволив уніфікувати логіку та уникнути дублювання при верстці. Ключові елементи інтерфейсу охоплюють навігаційне меню, мобільну панель, інформаційні секції, кнопки тестування та модальні вікна — зокрема, для реєстрації, входу, профілю та самого тестування.

Для адаптивності інтерфейсу застосовано Flexbox, а також медіа-запити в CSS. Це забезпечує плавне масштабування контенту під будь-який розмір

екрану — від смартфонів до великих моніторів. Модальні вікна відкриваються з анімацією, з затемненням заднього фону, що концентрує увагу користувача на активному елементі.

Всю інтерактивну частину реалізовано у JavaScript-файлі form.js. У ньому знаходиться код, який відповідає за керування модальними вікнами, перемикання між формами входу та реєстрації, перевірку введених даних (наприклад, валідацію email та пароля), збереження авторизованого користувача у сховищі браузера, генерацію тестів за темами, перевірку відповідей, підрахунок балів і відправку результатів на сервер. Уся ця логіка реалізована з використанням подієвих обробників, делегування подій та localStorage — що дозволяє забезпечити плавну взаємодію навіть без постійного з’єднання із сервером.

Серверна частина реалізована на Node.js із використанням фреймворку Express. Це дало змогу легко створити RESTful API, швидко підключити базу MongoDB через бібліотеку Mongoose і реалізувати маршрути для реєстрації, авторизації, роботи з результатами тестів, snippet-ами та зображеннями. На старті ініціалізовано проект через npm і сформував файл package.json, у якому вказав всі залежності.

У розробці були використані бібліотеки express для створення серверу, mongoose — для роботи з базою, cors — для дозволу міждоменних запитів, dotenv — для роботи з файлом конфігурації середовища, bcrypt — для хешування паролів, validator — для перевірки полів на валідність, cloudinary — для зберігання зображень, та nodemon — для зручної розробки з автоматичним перезапуском сервера при зміні файлів.

У файлі server.js підключино основні бібліотеки, створив проміжні обробники запитів, маршрути для обробки форм і API-запитів, реалізував логіку обробки помилок і взаємодії з базою. Маршрути для роботи з

користувачами, snippet-ами, тестами та зображеннями були винесені в окремі модулі.

Модель користувача включає поля для імені, електронної пошти, хешованого пароля, результатів тестування, а також URL зображення профілю. Для захисту паролів використано метод bcrypt.hashSync із сольовим значенням, а для перевірки email-адрес — функції з бібліотеки validator. Усі запити супроводжуються відповідними кодами стану HTTP та JSON-відповідями.

API містить маршрути для реєстрації користувачів, входу, отримання даних профілю, збереження результатів тестів, редагування профілю, видалення зображення та роботи з фрагментами коду. Усі ці маршрути були протестовані в Postman, де перевірино правильність відповіді сервера, формат запитів і обробку помилок. Сервер прослуховує порт 5000 і забезпечує стабільну роботу.

Структура проекту організована відповідно до патерну MVC. Виділино окремі папки для models, routes, controllers і utils, що дозволяє легко орієнтуватися в коді та масштабувати систему. Така архітектура відкриває можливість додавати новий функціонал — зокрема, систему рейтингів, коментарів, додаткові навчальні блоки — без перебудови основного ядра.

2.4.1 Написання клієнтської частини

Клієнтська частина вебсайту SkillUpCoding розроблялась як набір HTML-документів, що об'єднані єдиною структурою, стилем та сценаріями. Головна мета – забезпечити зручний, адаптивний та зрозумілий інтерфейс користувача для вивчення програмування та проходження тестів. Розмітка всіх сторінок є семантично коректною, стилізація реалізована за допомогою

CSS-файлу styles.css, а динамічна взаємодію забезпечино скриптом form.js, який підключається до кожної сторінки.

Інтерфейс складається з логотипу, меню, контенту, футера, а також низки модальних вікон, які відповідають за функціонал реєстрації, входу в обліковий запис, перегляду профілю та проходження тестів. Користувач може взаємодіяти з сайтом як гість, або зареєструватися та мати персоналізований інтерфейс.

Використано такі головні компоненти, які є повторюваними на більшості сторінок:

– шапка з логотипом SkillUpCoding та меню (адаптоване під десктоп і мобільні пристрої);

– кнопки для авторизації, реєстрації та доступу до профілю;

– адаптивне мобільне меню (офканвас);

– головний розділ з інформаційними блоками (напр. Python, JavaScript);

– кнопка для запуску тесту по темі;

– модальні вікна: логін, реєстрація, профіль, тест;

– скрол до гори сторінки;

– футер з інформацією про автора та посиланнями на соцмережі.

Кожна сторінка має загальний шаблон і включає ті ж самі елементи інтерфейсу, але з унікальним наповненням. Наприклад, сторінка python.html містить опис синтаксису мови Python, приклади коду, пояснення до типів даних, умов, циклів, функцій, обробки помилок, а також пояснення щодо ООП, модулів та бібліотек. У кінці кожної теми додається кнопка "Перевір вивчене!", яка відкриває тест (див. рис. 2.1), (див. рис. 2.2).

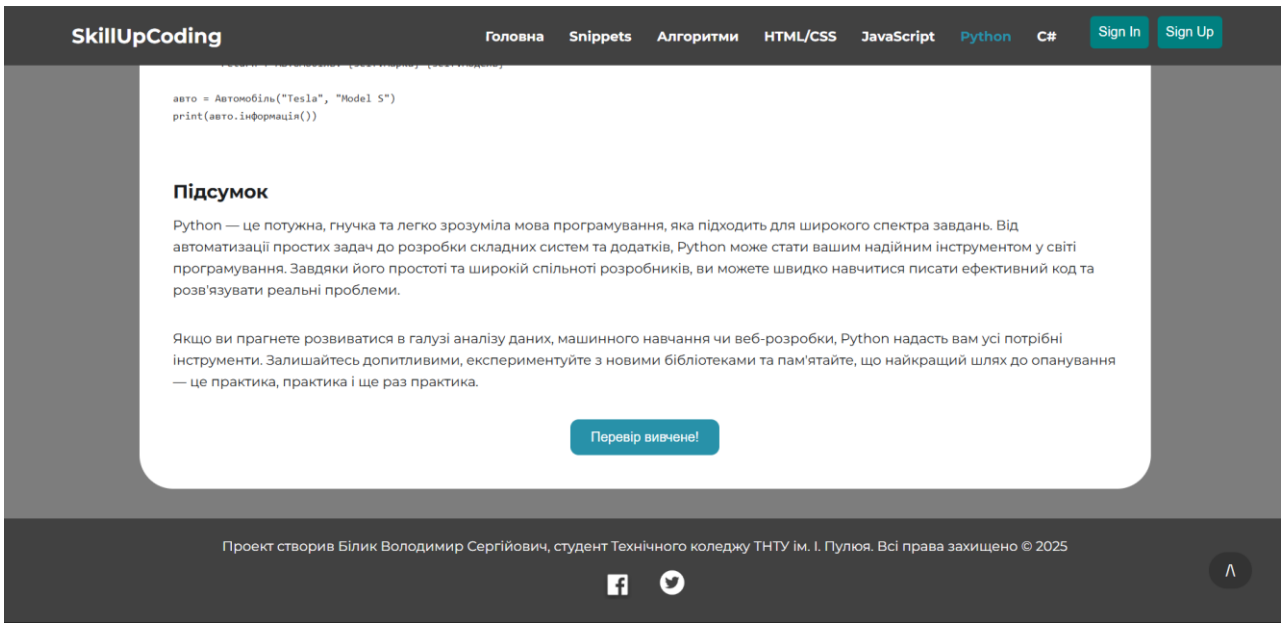


Рисунок 2.1 – Кнопка "Перевір вивчене!" на сторінці python.html

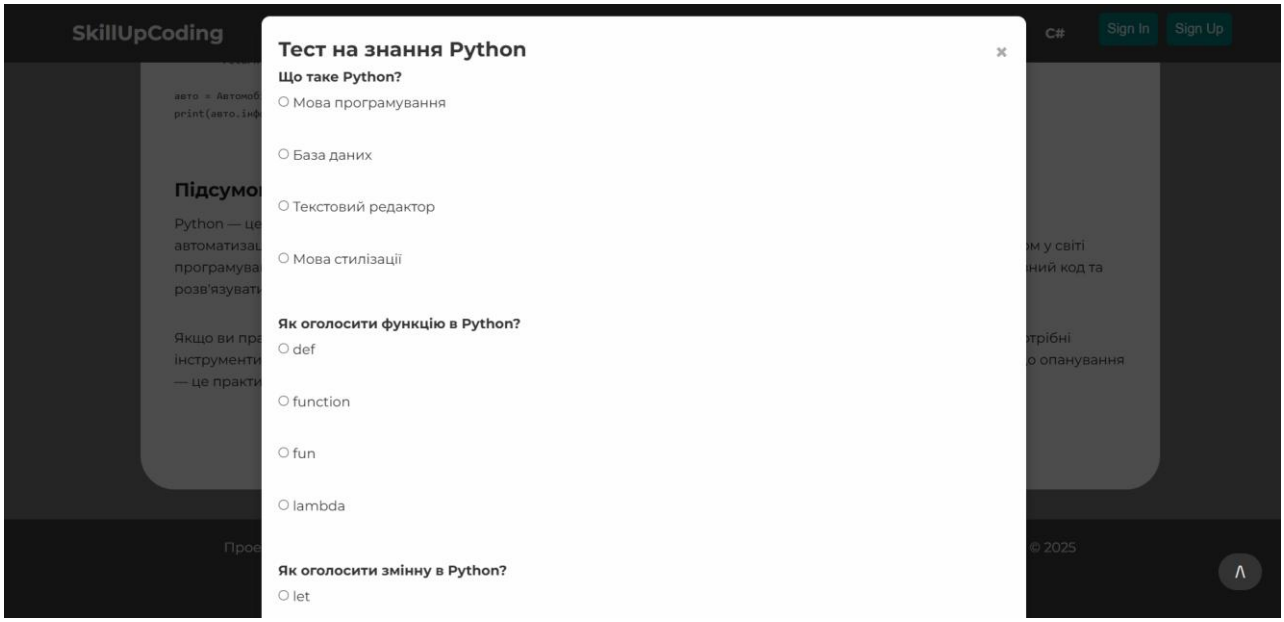


Рисунок 2.2 – Вигляд мобільного меню на сторінці python.html після натискання кнопки

Навігаційну панель на десктопах реалізовано в виді горизонтального меню, яке містить посилання на всі розділи сайту. Для мобільних пристроїв

розроблено офканвас-меню, яке активується іконкою-гамбургером. Це дозволяє зручно користуватися сайтом з будь-якого пристрою.

Усі форми на моєму сайті розроблені інтерактивними: при натисканні кнопки "Sign In" або "Sign Up" відкривається відповідна форма у вигляді модального вікна (див. рис. 2.3). Валідацію полів реалізовано у JavaScript-файлі form.js – перевіряється правильність email, довжина пароля, заповненість полів. Після заповнення форми, користувач "запам’ятовується" локально, і в інтерфейсі відображається кнопка "Профіль" замість кнопок "Sign In" і "Sign Up".

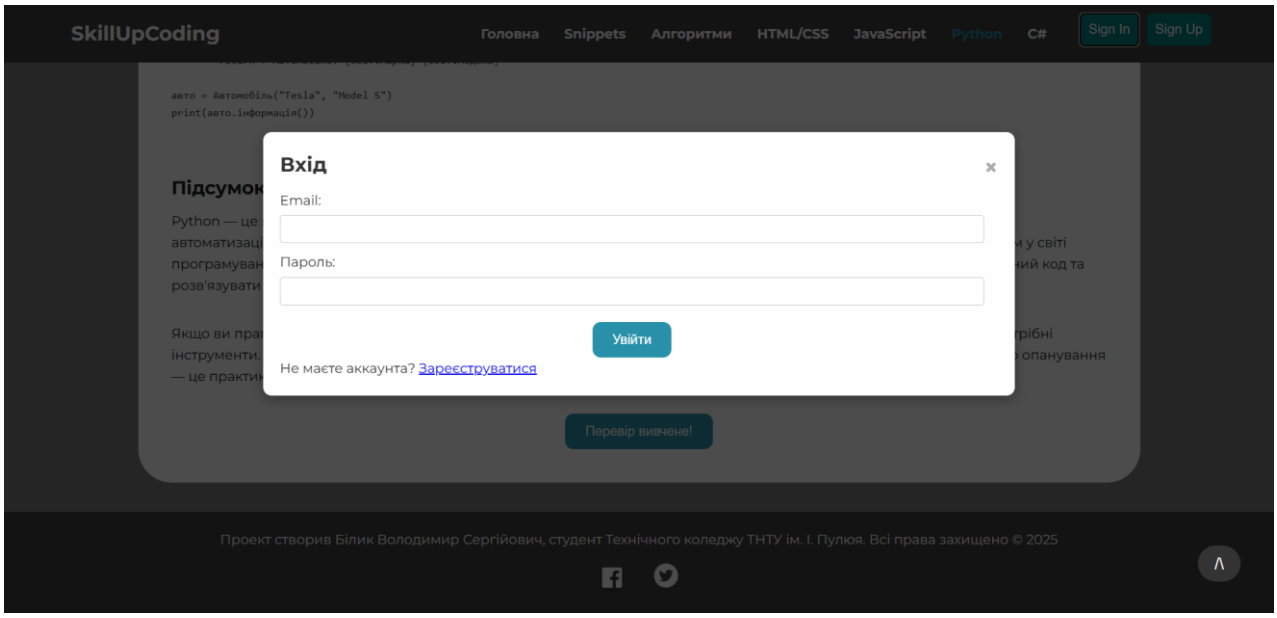


Рисунок 2.3 – Модальне вікно авторизації, яке відкривається після натискання кнопки "Sign In"

У вікні профілю користувач може додати зображення, написати кілька слів про себе та переглянути результати тестів. Цей блок працює з localStorage, що забезпечує збереження даних у межах поточного браузера. Крім того, у

блоці результатів виводяться теми, по яких користувач уже проходив тести, та їхні бали (див. рис. 2.4).

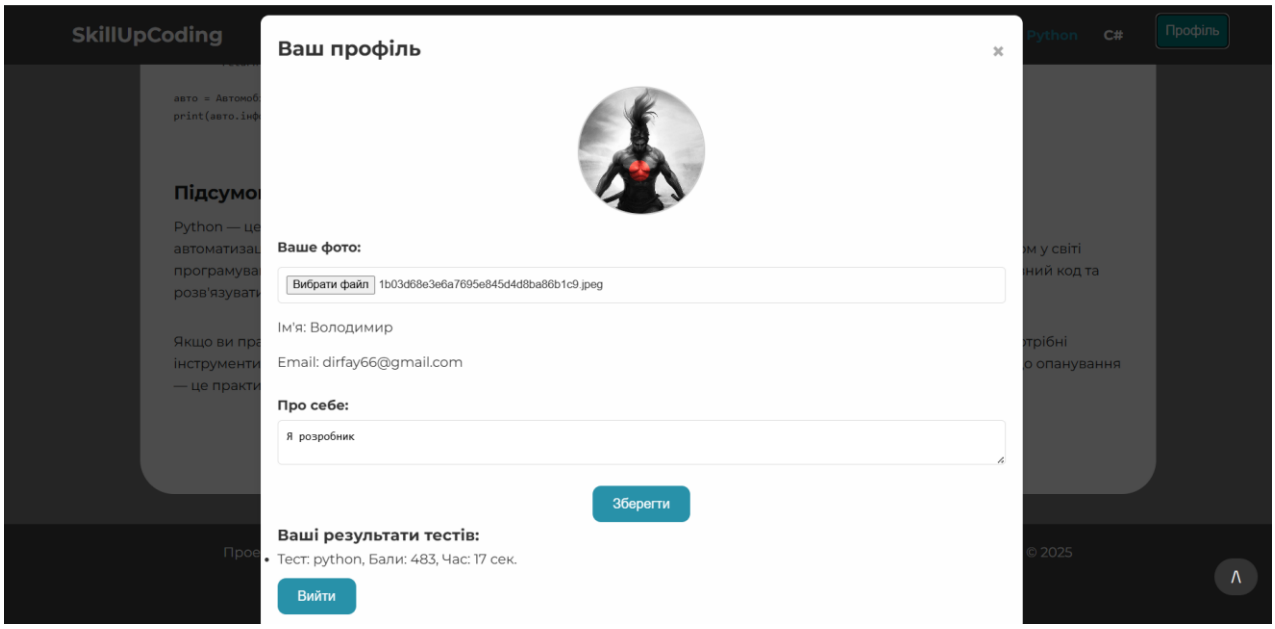


Рисунок 2.4 – Інтерфейс профілю користувача

Тестування знань у системі реалізовано через модальне вікно з адаптивним інтерфейсом, що відкривається під час вибору відповідного розділу. Запитання й варіанти відповідей завантажуються із зовнішнього JSON-файлу (tests.json), що дозволяє гнучко оновлювати вміст тестів без потреби змінювати основну логіку сайту. Користувач по черзі обирає варіанти відповідей, після чого система підраховує правильні й формує загальний бал.

По завершенні тесту в модальному вікні з’являється оцінка з текстовим поясненням — наприклад, якщо відсоток правильних відповідей високий, система виведе «Відмінно!», а при нижчому результаті — «Спробуйте ще раз». Також є кнопка «Пройти тест знову», яка скидає поточні відповіді та дозволяє повторити тестування для кращого засвоєння матеріалу (див. рис. 2.5).

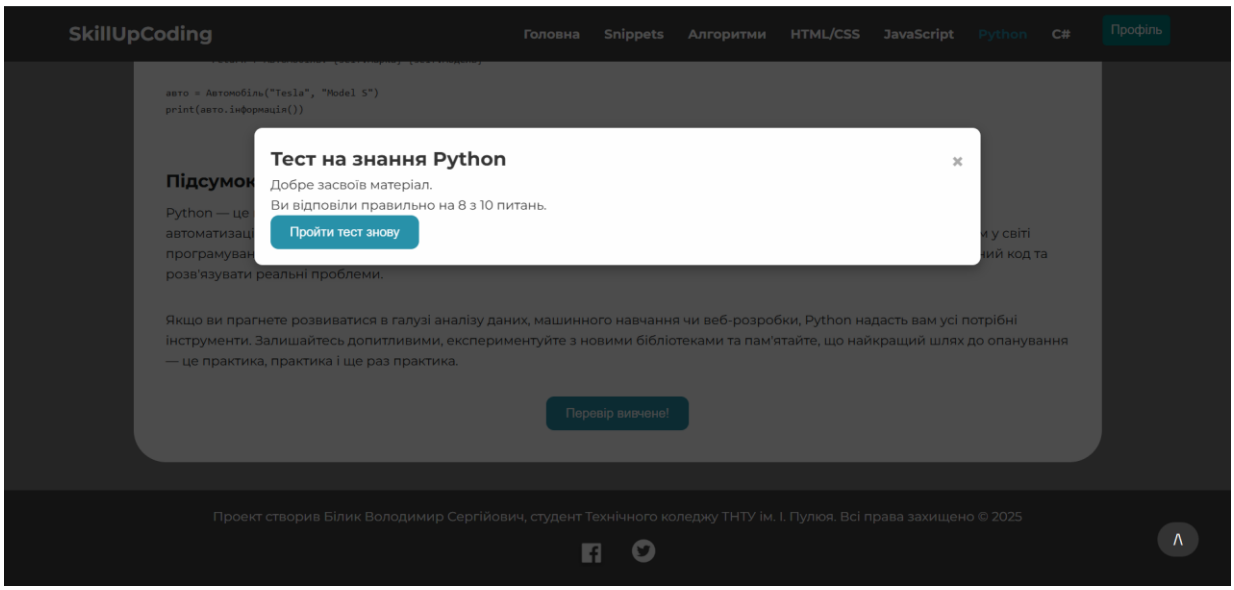


Рисунок 2.5 – Вигляд підсумкового результату

Весь JS-код, що відповідає за обробку форм, модальних вікон, тестів, переходів та авторизації, міститься в одному файлі form.js. У ньому реалізовані функції для:

– відкриття та закриття модальних вікон;
– зчитування даних із форм;
– перевірки email та паролів;
– збереження користувача в localStorage;
– генерації тестів з JSON-структури;
– підрахунку правильних відповідей;
– запису та виводу результатів у профіль;
– зміни інтерфейсу залежно від статусу користувача.

Фронтенд легко підключити до API – він підтримує динамічні запити й змінює контент без перезавантаження сторінки. Вона підтримує динамічну зміну контенту та імітує базову логіку авторизації, що дозволяє користуватися всіма головними функціями навіть у статичному режимі без бекенду.

2.4.2 Написання серверної частини

Серверну частину вебсайту SkillUpCoding реалізовано на базі Node.js з використанням фреймворку Express.js. Це дозволило створити швидкий, масштабований і зручний для підтримки вебсервіс, орієнтований на RESTful-архітектуру. Вибір Express був зумовлений його мінімалізмом, гнучкістю, активною спільнотою та повною сумісністю з MongoDB, яку обрано для зберігання даних.

Розробку почато зі створення окремої директорії для серверної частини та ініціалізації проекту за допомогою npm init, що дозволило згенерувати файл конфігурації package.json. Поступово підключено усі необхідні бібліотеки, кожна з яких виконувала свою роль: express — для запуску самого сервера і побудови маршрутизаторів, cors — для підтримки запитів з фронтенду, mongoose — для взаємодії з MongoDB, dotenv — для роботи з файлами змінних середовища, bcrypt — для безпечного хешування паролів, validator — для валідації введених даних, cloudinary — для збереження зображень у хмарному сховищі, а також nodemon — для автоматичного перезапуску сервера при зміні коду.

Основний серверний файл — server.js. У ньому налаштовано змінні середовища, ініціалізовано застосунок, підключив middleware для обробки запитів у форматі JSON і URL-encoded, налагоджено з’єднання з MongoDB через mongoose і організував прослуховування порту для запуску сервера.

Запити HTTP обробляються через окремі маршрутизатори, розміщені у відповідній директорії routes. Для кожної частини функціоналу створено окремі файли: userRoutes для обробки реєстрації, авторизації, доступу до профілю; productRoutes — для логіки додавання, редагування, видалення та фільтрації продуктів, а також для операцій з кошиком; orderRoutes — для

керування замовленнями та статусами доставки; imageRoutes — для інтеграції з Cloudinary і керування медіафайлами. Усі ці маршрути підключаються до головного застосунку через метод app.use() з відповідними базовими шляхами.

Для зберігання даних у MongoDB створено три моделі за допомогою mongoose.Schema: User, Product і Order. Модель користувача включає ім’я, email, пароль, роль (адміністратор чи звичайний користувач), список замовлень та кошик. Модель продукту охоплює назву, опис, ціну, категорію та масив із зображеннями. Замовлення містить перелік продуктів, інформацію про користувача, статус доставки, дату, адресу та загальну суму. Всі моделі забезпечують як валідацію, так і зручне оновлення або пошук документів у базі.

У логіці запитів було дотримано принципів REST: запити GET використовуються для отримання даних, POST — для створення нових сутностей, PATCH — для оновлення окремих властивостей, а DELETE — для видалення записів. Наприклад, коли користувач додає продукт до кошика, на сервер надходить POST-запит, у якому обробляються передані дані (ID продукту, ID користувача, кількість), після чого відповідне поле у базі оновлюється.

Інтеграція з Cloudinary дозволила реалізувати функціонал завантаження й видалення зображень напряму з клієнтської частини. Для цього було підключино їхній віджет, що дозволяє завантажувати фото, а також написав логіку видалення старих зображень у разі оновлення або видалення продукту.

Захист даних реалізовано на базовому рівні. Паролі зберігаються у вигляді хешів, а електронні адреси проходять перевірку на коректність. Крім того, доступ до деяких маршрутів обмежений — наприклад, тільки адміністратор має змогу змінювати статус замовлення чи додавати нові продукти. Це гарантує безпеку та контроль за доступом у системі.

2.5 Тестування вебсайту SkillUpCoding

Після завершення розробки як клієнтської, так і серверної частини вебсайту SkillUpCoding, розпочато комплексне тестування системи. Його мета — забезпечити стабільну роботу, зручність для користувачів і відповідність усім технічним вимогам. Тестування охоплювало візуальну перевірку, функціональну логіку, збереження даних у базі, коректність форм, стабільність API та загальний рівень безпеки.

На першому етапі було зосереджено на тестуванні візуальної частини. Було вручну перевірино адаптивність сайту на різних пристроях: смартфонах, планшетах та десктопах. Для цього використовувались інструменти розробника у браузері Google Chrome. Було перевірено, що навігаційне меню відображається правильно, мобільне меню (offcanvas) працює як слід, контент на кожній сторінці виглядає логічно і привабливо. Особливо було звернено увагу на роботу кнопок «Sign In», «Sign Up» та «Профіль», перевіряв відкриття і закриття модальних вікон, а також запуск тестування через кнопку «Перевір вивчене!».

Після візуального тесту перейшов до функціональної перевірки логіки форм. Було протестовано, як система реагує на правильні та неправильні дані при заповненні реєстраційної і логін-форми. Перевірив, чи працює валідація email-адрес, чи відхиляються некоректні паролі, та як система реагує на спробу повторної реєстрації одного й того ж користувача. У профілі перевірив, чи можна змінювати інформацію, додавати фото, а також чи зберігається текст у секції «Про себе».

Особливо ретельно було протестовано модуль проходження тестів. Проходжено тести у всіх п’яти розділах: HTML/CSS, JavaScript, Python, C# і Алгоритми. У кожному з них перевіряв, чи завантажуються питання і варіанти

відповідей, як реагує інтерфейс на вибір, як працює кнопка «Завершити тест», і чи правильно розраховується кількість правильних відповідей. Перевірино, як працює функція повторного проходження тесту, чи зберігаються результати в профілі користувача, та чи видно ці результати після перезавантаження або нового входу.

Також було передбачино ситуацію, коли користувач не відповідає на всі питання — у такому випадку тест не завершується до моменту, поки не буде обрано відповідь на кожне запитання. Результат тесту має відображатися у зрозумілому форматі, із повідомленням про успішність чи неуспішність спроби.

У модулі профілю перевірино відображення всіх збережених результатів. Переконався, що дані користувача — ім’я, email, фото, інформація «Про себе» — зберігаються, і що результат кожного тесту відображається чітко, без дублювання або втрати даних. Було кілька разів оновлено сторінку, виходив з облікового запису та входив знову — усі зміни лишалися актуальними.

На стороні сервера тестовано API за допомогою Postman. Виконано POST-запити на маршрути для реєстрації та входу, GET-запити для отримання даних користувачів, продуктів і замовлень, PATCH-запити для оновлення статусу, а також DELETE-запити для видалення зображень. Перевірялась структура тіла запитів, статуси відповідей (наприклад, 200, 201, 400, 403, 404), формат JSON-відповідей, правильність обробки помилок і загальна швидкість відповіді сервера.

Окремо верифікував дії в самій базі даних через MongoDB Compass. Було переглядено створення документів після реєстрації, оновлення полів при взаємодії з кошиком, появу замовлень після тестів, а також перевіряв збереження профільних фото та даних про користувача.

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція з розміщення сайту в Інтернеті

Після завершення процесу розробки вебресурсу «SkillUpCoding» виникла необхідність забезпечити його постійну доступність для кінцевих користувачів через глобальну мережу Інтернет. З цією метою обрано комбіновану архітектуру розгортання, яка передбачає розміщення клієнтської частини на сервісі GitHub Pages, а серверної частини – на Render.com. Цей підхід дозволяє досягти високого рівня надійності, гнучкості, а також максимально ефективно розділити навантаження між різними платформами.

Клієнтська частина вебсайту, яка відповідає за взаємодію з користувачем, візуалізацію інтерфейсу, відображення тестів, кодування, фрагментів коду (snippets) та інших елементів UI/UX, розроблена із застосуванням HTML5, CSS3 та JavaScript. Ці технології не потребують складного серверного середовища для виконання, що дозволяє розміщувати готові сторінки на сервісах, призначених для розгортання статичних вебсайтів. Одним із найбільш зручних та безкоштовних рішень у цьому напрямку є платформа GitHub Pages, яка надає можливість миттєво опублікувати сайт безпосередньо з репозиторію GitHub, що значно пришвидшує роботу над оновленням ресурсу.

Процес публікації клієнтської частини сайту відбувався за наступним алгоритмом:

– Створення репозиторію з назвою SkillUpCoding на платформі GitHub.

– Ініціалізація Git у локальній директорії проекту та додавання всіх файлів командою "git add . ".

– Створення коміту та відправлення початкової версії сайту до гілки master.

– Встановлення додаткової бібліотеки gh-pages у вигляді dev-залежності, що дозволяє автоматизувати публікацію вмісту з обраної директорії (у даному випадку — public) у спеціальну гілку gh-pages, з якої і здійснюється розгортання GitHub Pages.

– Додавання скрипта "deploy": "gh-pages -d public" до файлу package.json.

– Виконання команди npm run deploy, після якої проект став доступним за публічною URL-адресою - https://dirfay.github.io/SkillUpCoding/.

Завдяки використанню GitHub Pages мені вдалося реалізувати швидкий, безкоштовний та стабільний спосіб публікації фронтенду без необхідності витрат на додатковий хостинг. Будь-які зміни в інтерфейсі можуть бути опубліковані буквально за кілька хвилин шляхом оновлення репозиторію та повторного виконання команди деплою.

Поряд із клієнтською частиною значну роль у роботі проекту відіграє серверна логіка. Серверна частину, вирішино написати мовою програмування JavaScript з використанням фреймворку Express.js, це реалізує функціонал реєстрації та аутентифікації користувачів, обробки запитів, збереження даних до локальної бази даних SQLite, завантаження файлів, а також взаємодію з тестами, коментарями та snippets. Для її розгортання обрано сервіс Render.com — одну з найпопулярніших сучасних платформ для розгортання Node.js-додатків.

Render.com надає низку переваг для проекту:

– Безкоштовний тарифний план для навчальних та невеликих проектів;

– Підтримка автоматичного деплою при кожному оновленні коду в репозиторії GitHub;

– Зручний вебінтерфейс для моніторингу логів, керування залежностями та керування середовищем;

– Надійна серверна інфраструктура з автоматичним відновленням при збоях.

Розгортання серверної частини на Render.com відбувалося за наступною послідовністю:

1. Реєстрація облікового запису на render.com та підключення до облікового запису GitHub.

2. Створення нового Web Service, де обрано назву сервісу SkillUpCoding, тип середовища — Node, та гілку master як джерело публікації.

3. Встановлення головних параметрів:

– Build Command: npm install;

– Start Command: npm start;

– Root Directory залишено порожнім, оскільки серверний код знаходиться у корені репозиторію.

4. Завантаження package.json, де міститься вся необхідна інформація про залежності, включно з бібліотеками як bcrypt, express, sqlite3, cors та іншими.

5. Ініціація процесу деплою, який автоматично запускається після підключення репозиторію.

6. Успішний запуск сервера підтверджується повідомленням у консолі Render (див. рис. 3.1), де вказується, що сервер працює на визначеному порті.

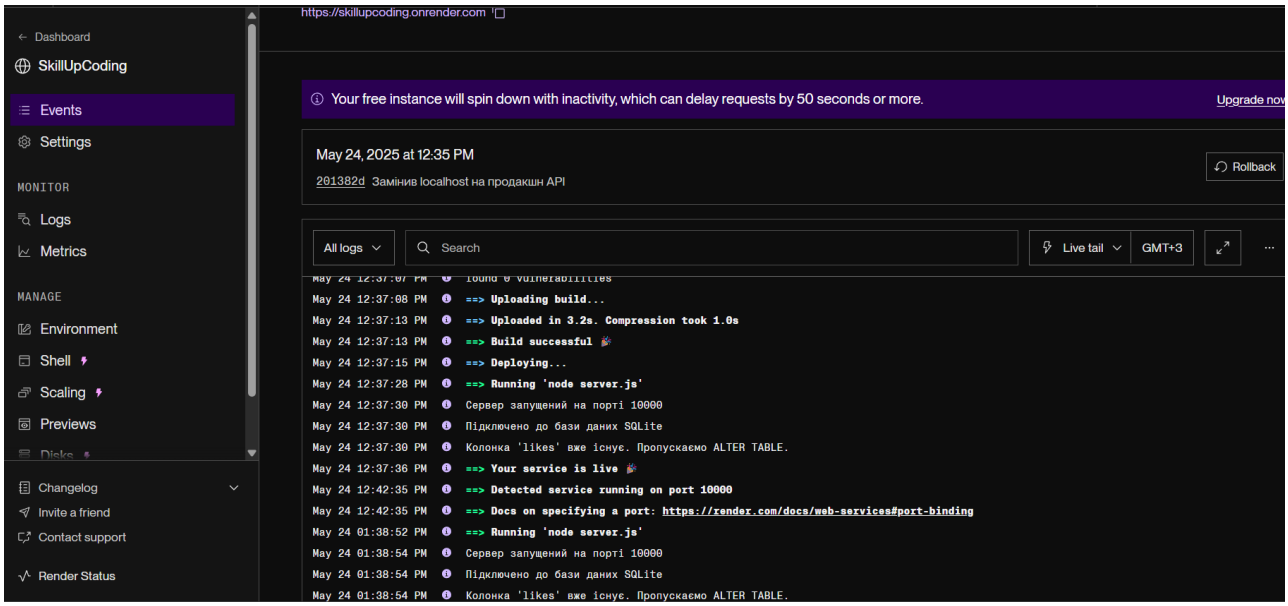


Рисунок 3.1 – Консоль логів Render зі статусом успішного запуску серверної частини сайту

Після успішного розгортання сайт отримав унікальну адресу - https://skillupcoding.onrender.com, яка використовується для всіх запитів до серверної частини. Взаємодія між клієнтом і сервером здійснюється за допомогою HTTP-запитів (GET, POST, PUT, DELETE), а обмін даними — у форматі JSON.

Щоб забезпечити коректну взаємодію між фронтендом і бекендом, налаштовано CORS, що дозволяє клієнту, розміщеному на GitHub Pages, надсилати запити до домену Render. Це є обов’язковою умовою при взаємодії між двома різними доменами. Усі запити авторизації та роботи з персональними даними також захищені за допомогою JWT-токенів, що забезпечує базовий рівень безпеки та ідентифікації користувачів.

В результаті сайт повністю працює онлайн: фронтенд і бекенд обмінюються даними без збоїв, запити обробляються, база даних функціонує,

а інтерфейс коректно оновлюється. Така архітектура розгортання довела свою ефективність під час тестування і є зразковим прикладом сучасного підходу до публікації web-проектів у навчальному середовищі.

3.2 Інструкція з обслуговування та наповнення сайту

Для стабільної та безперебійної роботи сайту «SkillUpCoding» було передбачено повний цикл технічного супроводу, який включає обслуговування як клієнтської, так і серверної частин проекту. Мета такого супроводу — гарантувати безпеку, швидкодію, актуальність контенту та зручність користування платформою.

Клієнтська частина сайту, розроблена за допомогою HTML, CSS та JavaScript, вимагає постійного моніторингу адаптивності інтерфейсу, сумісності з різними браузерами та коректного відображення на пристроях із різними розмірами екрана. Регулярно переглядаю інтерфейс у мобільному, планшетному та десктопному форматах, перевіряючи роботу меню, тестових блоків, форм авторизації та профілю. Усі необхідні зміни вносяться локально, після чого використовую стандартний ланцюжок команд — додавання змінених файлів у Git, фіксація змін через commit, надсилання на GitHub та деплой через GitHub Pages. Завдяки такому підходу оновлення клієнтської частини відбувається швидко, без потреби втручання в хостинг або ручного копіювання файлів.

Окрему увагу було приділено серверній частині, яка реалізована на платформі Node.js у поєднанні з фреймворком Express. Саме серверна логіка відповідає за функціонування акаунтів, зберігання результатів тестування, обробку snippet-ів, зображень, коментарів та іншої інформації, пов’язаної з користувачами. Щоб підтримувати стабільну роботу, регулярно перевіряю

журнали логів у Render — це допомагає виявляти помилки та вчасно усувати їх. У разі зміни функціоналу на сервері оновлюю відповідні файли, зокрема server.js, моделі бази даних або логіку API, після чого зміни надсилаються на GitHub. Завдяки інтеграції GitHub із Render, кожен новий коміт автоматично ініціює процес оновлення серверного застосунку, що забезпечує безперервну роботу без зупинки сервера.

Щоб підтримувати актуальність бібліотек і залежностей, регулярно перевіряю їхні версії через команду npm audit. Це дозволяє уникати потенційно небезпечних вразливостей у пакунках, а також своєчасно оновлювати їх до стабільних версій. Важливим етапом є також періодична перевірка стабільності хостингу Render, особливо якщо використовується безкоштовний тариф, який може призводити до короткочасних затримок у завантаженні.

Контентна частина сайту також потребує активного управління. Зареєстровані користувачі можуть створювати профілі, редагувати інформацію про себе, завантажувати зображення, додавати нові фрагменти коду, залишати коментарі до snippet-ів і вподобувати чужі матеріали. Усі ці дії формують базу даних, що зберігається у вигляді локального файлу database.db, який працює під управлінням SQLite. Забезпечив захист усіх персональних дій через авторизаційні токени JWT, завдяки чому користувач має доступ лише до своїх даних або до публічно доступного вмісту.

Після внесення змін на сайті завжди перевіряю, чи всі функції працюють правильно. Це включає запуск тестів, додавання snippet-ів, зміну профілю, перевірку системи лайків і коментарів, а також відкриття модальних вікон. Якщо виявляються помилки, оперативно їх виправляю через браузерну консоль або за допомогою інструментів налагодження Render. Також важливо регулярно перевіряти, чи дані в localStorage відповідають очікуваним — це

гарантує правильну роботу користувацького профілю після оновлення сторінки або повторного входу.

Щодо оновлення інформаційної частини, періодично додаються нові розділи, теоретичні блоки, питання для тестів, а також приклади коду. Це можна зробити шляхом редагування HTML-сторінок і JSON-файлів із тестовими завданнями. Завдяки структурованості сайту можливо розширювати його функціонал без необхідності повністю переробляти існуючий код. Наприклад, додати нову тему з програмування — достатньо створити новий HTML-файл із шаблоном сторінки та відповідним вмістом, а також додати секцію у файл tests.json.

У рамках резервного копіювання періодично зберігаю файл бази даних у зовнішнє захищене сховище. Це дозволяє убезпечити проект від втрати важливої інформації. Також рекомендовано мати копії ключових конфігураційних файлів, як-от server.js, package.json, .env, щоб мати можливість швидко розгорнути сайт у разі аварії.

Таким чином, підтримка сайту SkillUpCoding включає в себе не лише технічні оновлення та обслуговування, а й щоденну роботу з контентом, моніторинг стабільності, тестування логіки, взаємодію з користувачами, а також забезпечення надійного зберігання даних. Це безперервний процес, який дозволяє платформі залишатися актуальною, безпечною та зручною для кінцевих користувачів.

3.3 Інструкція з популяризації та підтримки сайту

Після завершення етапів розробки та публікації сайту «SkillUpCoding» важливо забезпечити його подальшу життєздатність, пізнаваність і регулярну присутність у цифровому просторі. В умовах сучасної конкуренції в інтернеті

лише створення функціонального ресурсу є недостатнім. Мені необхідно проводити постійну роботу з популяризації, оптимізації, взаємодії з користувачами та технічної підтримки. Саме це дозволяє моєму сайту не просто існувати, а розвиватися, розширювати аудиторію та зміцнювати позиції серед аналогічних платформ.

Процес популяризації сайту охоплює комплекс заходів, що спрямовані на підвищення відвідуваності, покращення індексації у пошукових системах, формування лояльної аудиторії та створення позитивного іміджу онлайн-ресурсу. В першому етапі мені треба розпочати із внутрішньої оптимізації — так званої SEO (Search Engine Optimization). Для цього необхідно переконатися, що кожна сторінка сайту має унікальні та релевантні мета-теги: title, description, keywords. Також забезпечити наявність семантично значущих заголовків, структурованої розмітки HTML, а також читабельних URL-адрес. Важливим етапом є додавання XML-карти сайту, яка допомагає пошуковим системам швидше та ефективніше індексувати сторінки.

Окрім технічної оптимізації, значну роль відіграє контент-маркетинг. Регулярне оновлення контенту, публікація корисних матеріалів, інтерактивних завдань, навчальних кодів та статей на тему програмування дозволяє не лише втримати постійних користувачів, а й залучити нових. Важливо створити розділ або блог, який би регулярно оновлювався актуальними темами: поради для новачків, аналіз помилок у коді, короткі підказки для проходження тестів.

Паралельно з контентом мені слід розгорнути активність у соціальних мережах. Це дозволяє створити постійний канал комунікації з потенційною аудиторією. Платформи, такі як Instagram, Telegram, Facebook та YouTube, можуть бути використані для просування нового контенту, запуску відеоуроків, анонсів нових можливостей, а також інтерактивного спілкування

зі спільнотою. Створення акаунтів сайту у соцмережах мені дає змогу налагодити трафік, який буде перенаправлятися на головний ресурс. Добре працює також стратегія запуску щотижневих або щомісячних розсилок через email-маркетинг із рекомендаціями, аналітикою результатів проходження тестів, новинами про оновлення функціоналу.

Додатковим інструментом для популяризації сайту є проведення партнерських кампаній. Наприклад, можна налагодити співпрацю з освітніми платформами, програмістськими спільнотами, хакатонами або IT-курсами. Це дозволить розширити аудиторію за рахунок зацікавлених осіб із суміжних напрямів.

Не менш важливою складовою є технічна підтримка сайту. Вона передбачає регулярне оновлення залежностей проекту, перевірку журналів логів (особливо на платформі Render), моніторинг продуктивності, швидкості завантаження сторінок, усунення виявлених багів і недопрацювань. Для цього доцільно використовувати інструменти автоматичного моніторингу, такі як UptimeRobot або Google PageSpeed Insights. Зокрема, треба звертати увагу на повідомлення Render про помилки при деплої, щоб оперативно реагувати на збої.

Для забезпечення високої надійності ресурсу та уникнення втрати даних рекомендується реалізувати регулярне резервне копіювання бази даних SQLite. Це можна зробити шляхом автоматизованого копіювання файлу database.db у захищене сховище.

З боку зворотного зв’язку важливо забезпечити можливість комунікації користувачів із розробником або адміністрацією сайту. Це може бути реалізовано через форму зворотного зв’язку, email-підтримку або Telegram-бот. Швидка реакція на звернення користувачів підвищує довіру та репутацію ресурсу.

У підсумку, успішна підтримка й популяризація сайту базується на ключових принципах:

– Постійне оновлення контенту й функціоналу;
– Активність у соцмережах і email-комунікація;
– Впровадження SEO-оптимізації;
– Моніторинг, тестування і технічна підтримка;
– Використання відгуків і зворотного зв’язку користувачів;
– Резервне копіювання критичних даних.

Виконання зазначених дій дає змогу підтримувати сайт «SkillUpCoding» у стабільному робочому стані, забезпечувати постійний приріст користувачів і закріплювати позиції проекту у сфері онлайн-освіти з програмування. Завдяки грамотному управлінню й послідовному розвитку ресурс матиме всі передумови для довготривалої ефективної експлуатації.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини кваліфікаційної роботи бакалавра є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності розробки навчально-демонстраційного сайту «SkillUpCoding». В рамках цього розділу проводиться розрахунок витрат на реалізацію проекту, визначаються його собівартість, економічна ефективність та термін окупності.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

Для визначення загальної тривалості проведення науково-дослідної роботи необхідно узагальнити витрати часу на виконання окремих операцій технологічного процесу, що наведено в таблиці 4.1.

Таблиця 4.1 – Середній час виконання НДР та стадії (операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	2	3	4
1	Аналіз вимог та формулювання технічного завдання на розробку навчально-демонстраційного сайту	Керівник проекту	12
2	Створення макету дизайну вебсайту та затвердження із замовником	Інженер	20
3	Розробка frontend-частини (HTML/CSS/JS)	Інженер	24

Продовження таблиці 4.1

1	2	3	4
4	Розробка backend-частини (Python, Django, бази даних)	Інженер	30
5	Тестування та відлагодження вебсайту	Інженер	16
6	Встановлення та налаштування необхідного ПЗ на сервері	Технік	10
7	Підготовка технічної документації та інструкцій користувача	Інженер	8
8	Здача готового сайту замовнику та навчання персоналу	Керівник проекту	6
Разом			126

Таким чином, загальний час, необхідний для виконання всіх стадій технологічного процесу науково-дослідної роботи з розробки навчально-демонстраційного сайту, становить 126 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Оплата праці являє собою грошовий вираз вартості робочої сили, яка виплачується працівникам за виконану роботу. Заробітна плата визначається з урахуванням тарифних ставок та кількості відпрацьованих годин.

Основна заробітна плата обчислюється за формулою:

$$З_{осн} = Т_с \cdot К_г \quad (4.1)$$

де:

$Т_с$ – тарифна ставка, грн/год;

$К_г$ – кількість відпрацьованих годин.

Рекомендовані тарифні ставки:

– керівник проекту – 260 грн/год;

– інженер – 180 грн/год;

– технік – 100 грн/год.

Розрахунок основної заробітної плати за категоріями працівників здійснено наступним чином:

1. Керівник проекту:

$$З_{осн1} = 260 \cdot (12 + 6) = 4680 \text{ грн.}$$

2. Інженер:

$$З_{осн2} = 180 \cdot (20 + 24 + 30 + 16 + 8) = 17640 \text{ грн.}$$

3. Технік:

$$З_{осн3} = 100 \cdot 10 = 1000 \text{ грн.}$$

Сумарна основна заробітна плата становить:

$$З_{осн} = 4680 + 17640 + 1000 = 23320 \text{ грн.}$$

Додаткова заробітна плата визначається на рівні 12% від основної заробітної плати за формулою:

$$З_{дод} = З_{осн} \cdot К_{дод} \quad (4.2)$$

де:

$К_{дод}$ – коефіцієнт додаткових виплат (приймаємо 0,12).

Отже, додаткова заробітна плата становить:

$$З_{дод} = 23320 \cdot 0,12 = 2798,40 \text{ грн.}$$

Загальні витрати на оплату праці визначаються сумою основної і додаткової заробітної плати:

$$В_{о.п} = З_{осн} + З_{дод} = 23320 + 2798,40 = 26118,40 \text{ грн.}$$

Відрахування на соціальні заходи, відповідно до прикладу, приймаємо на рівні 37,6%. Вони обчислюються за формулою:

$$В_{с.з.} = В_{о.п.} \cdot 0,376 \quad (4.3)$$

Звідси, відрахування на соціальні заходи становлять:

$$В_{с.з.} = 26118,40 \cdot 0,376 = 9812,52 \text{ грн.}$$

Усі отримані результати розрахунків наведено у зведеній таблиці 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці та соціальні відрахування

№ п/п	Категорія працівників	Тарифна ставка, грн/год	К-сть годин	Основна заробітна плата, грн	Додаткова заробітна плата, грн	Відрахування на соц. заходи, грн	Всього витрат на оплату праці, грн
1	2	3	4	5	6	7	8
1	Керівник проекту	260	18	4680	561,60	–	–
2	Інженер	180	98	17640	2116,80	–	–

Продовження таблиці 4.2

1	2	3	4	5	6	7	8
3	Технік	100	10	1000	120,00	–	–
Разом			126	23320	2798,40	9812,52	35930,92

Отже, загальна сума витрат на оплату праці разом з соціальними відрахуваннями становить 35930,92 грн.

4.3 Розрахунок матеріальних витрат

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів на їх ціну і обчислюються за формулою:

$$M_{B_i} = q_i \cdot p_i \tag{4.4}$$

де:

q_i – кількість витраченого матеріалу і-го виду;

p_i – ціна матеріалу і-го виду.

Загальні матеріальні витрати визначаються за формулою:

$$З_{м.в.} = \sum M_{B_i} \tag{4.5}$$

Для реалізації кваліфікаційної роботи бакалавра розробки навчально-демонстраційного сайту необхідно використати наступні матеріали та ресурси, перелік та вартість яких наведено у таблиці 4.3.

Таблиця 4.3 – Зведені розрахунки матеріальних витрат для розробки навчально-демонстраційного сайту

№ п/п	Найменування матеріалів та ресурсів	Од. вим.	Кількість	Ціна за одиницю, грн	Загальна сума, грн
1	Доменне ім'я сайту (.com)	шт.	1	500	500
2	Хостинг (1 рік оренди)	шт.	1	2500	2500
3	Ліцензія на програмне забезпечення (JetBrains IDE)	шт.	1	4000	4000
4	Ліцензія на Adobe Photoshop (1 рік)	шт.	1	8000	8000
5	SSL-сертифікат (на 1 рік)	шт.	1	1000	1000
6	Використання сторонніх API (Google, сервіс аутентифікації)	шт.	1	3000	3000
7	Комплект витратних матеріалів (офісний папір, картриджі, папки для документації)	комплект	1	1500	1500
8	Технічна література (довідники з програмування, розробки вебінтерфейсів)	шт.	3	600	1800
Разом					22300

Таким чином, загальна сума матеріальних витрат на реалізацію кваліфікаційної роботи бакалавра з розробки навчально-демонстраційного сайту «SkillUpCoding» становить 22300 грн.

4.4 Розрахунок витрат на електроенергію

Витрати на електроенергію розраховуються за формулою:

$$З_е = W \cdot T \cdot S \tag{4.6}$$

де:

W – потужність обладнання, кВт;

T – час роботи обладнання, год;

S – вартість електроенергії за 1 кВт·год, грн.

Згідно з підходом, який використано у прикладі, приймаємо такі значення:

– тривалість роботи обладнання – 38 год;

– потужність обладнання – 0,7 кВт/год;

– вартість 1 кВт·год електроенергії – 7 грн.

Виконавши розрахунок, отримуємо витрати на електроенергію:

$$З_е = 0,7 \cdot 38 \cdot 7 = 186,20 \text{ грн}$$

Таким чином, витрати на електроенергію для реалізації дипломного проекту становлять 186,20 грн. Отриманий результат демонструє економічність використання обладнання та раціональність робочого часу в процесі виконання завдань дипломного проекту.

4.5 Визначення транспортних затрат

Транспортні витрати розраховуються як частка від загальної суми матеріальних витрат і становлять, за рекомендаціями, від 8 до 10%. Вони визначаються за формулою:

$$ТВ = З_{м.в.} \cdot (0,08 \dots 0,1) \quad (4.7)$$

де:

$З_{м.в.}$ – загальні матеріальні витрати, грн.

З урахуванням раніше розрахованої загальної суми матеріальних витрат, що становить 22300 грн, приймаємо транспортні витрати на рівні 8%:

$$ТВ = 22300 \cdot 0,08 = 1784 \text{ грн.}$$

Отже, транспортні витрати на реалізацію кваліфікаційної роботи бакалавра становлять 1784 грн.

4.6 Розрахунок суми амортизаційних відрахувань

Комп'ютерне обладнання та оргтехніка належать до четвертої групи основних фондів. Мінімально допустимий термін їх використання згідно з нормативними вимогами становить 2 роки.

Сума амортизаційних відрахувань за час роботи обладнання розраховується за формулою:

$$A = \frac{БВ \cdot НА \cdot Т}{150\%} \quad (4.8)$$

де:

БВ – балансова вартість обладнання (30000 грн);

НА – норма амортизації (4%), прийнята відповідно до нормативів;

Т – кількість годин роботи обладнання (126 год).

Підставляємо значення і розраховуємо:

$$A = \frac{30000 \cdot 0,04 \cdot 126}{150} = 100,80 \text{ грн}$$

Таким чином, сума амортизаційних відрахувань, що припадає на час роботи над проектом, становить 100,80 грн.

4.7 Обчислення накладних витрат

Накладні витрати — це витрати, які не пов’язані безпосередньо з технологічним процесом розробки навчально-демонстраційного сайту, проте виникають при його реалізації. До таких витрат належать витрати на адміністрування, організаційні витрати, оренду приміщень, комунальні платежі та інші управлінські витрати.

Накладні витрати можуть становити 20–60% від загальної суми основної та додаткової заробітної плати і визначаються за формулою:

$$\text{НВ} = \text{В}_{\text{о.п.}} \cdot (0,2 \ldots 0,6) \quad (4.9)$$

де:

$\text{В}_{\text{о.п.}}$ — загальні витрати на оплату праці, грн.

Для даного проекту приймаємо накладні витрати на рівні 40% від загальних витрат на оплату праці (основна та додаткова зарплата), які становлять 26118,40 грн.

Розрахунок накладних витрат:

$$НВ = 26118,40 \cdot 0,4 = 10447,36 \text{ грн}$$

Отже, сума накладних витрат для реалізації кваліфікаційної роботи бакалавра становить 10447,36 грн.

4.8 Складання кошторису витрат та визначення собівартості НДР

Кошторис витрат є підсумковим документом, який об'єднує всі види витрат, що були попередньо розраховані для реалізації кваліфікаційної роботи бакалавра. Він відображає всі витрати, пов’язані з виконанням роботи над проектом та служить для визначення загальної собівартості науково-дослідної роботи (НДР).

Загальна собівартість визначається за формулою:

$$С_в = В_{о.п.} + В_{с.з.} + З_{м.в.} + З_{е.} + ТВ + А + НВ \quad (4.10)$$

де:

$В_{о.п.}$ – загальні витрати на оплату праці (основна + додаткова заробітна плата);

$В_{с.з.}$ – відрахування на соціальні заходи;

$З_{м.в.}$ – матеріальні витрати;

$З_{е.}$ – витрати на електроенергію;

ТВ – транспортні витрати;

А – амортизаційні відрахування;

НВ – накладні витрати.

Проведені розрахунки зведено у таблицю 4.4.

Таблиця 4.4 – Кошторис витрат на реалізацію кваліфікаційної роботи бакалавра

№ п/п	Стаття витрат	Сума, грн.	В % до загальної суми
1	Витрати на оплату праці (основна + додаткова)	26 118,40	36,9 %
2	Відрахування на соціальні заходи	9 812,52	13,9 %
3	Матеріальні витрати	22 300,00	31,5 %
4	Витрати на електроенергію	186,20	0,3 %
5	Транспортні витрати	1 784,00	2,5 %
6	Амортизаційні відрахування	100,80	0,1 %
7	Накладні витрати	10 447,36	14,8 %
Разом	Загальна собівартість НДР	70 749,28	100,0 %

Таким чином, загальна собівартість реалізації кваліфікаційної роботи бакалавра з розробки навчально-демонстраційного сайту становить 70749,28 грн.

4.9 Розрахунок ціни НДР

Ціна науково-дослідної роботи визначається з урахуванням собівартості виконання роботи, запланованого рівня рентабельності та податку на додану вартість (ПДВ), ставка якого становить 20%.

Розрахунок ціни здійснюється за формулою:

$$Ц = С_в \cdot (1 + Р_{рен}) \cdot (1 + ПДВ) \quad (4.11)$$

де:

Ц – ціна НДР, грн;

$С_в$ – собівартість виконання НДР, грн;

$Р_{рен}$ – рівень рентабельності (приймаємо 30%);

ПДВ – податок на додану вартість (20%).

Використовуючи попередньо розраховану собівартість проекту (70749,28 грн), визначаємо ціну науково-дослідної роботи таким чином:

$$Ц = 70749,28 \cdot 1,3 \cdot 1,2 = 110368,08 \text{ грн}$$

Отже, ціна виконання кваліфікаційної роботи бакалавра з розробки навчально-демонстраційного сайту становить 110368,08 грн, що включає собівартість, плановий рівень рентабельності та ПДВ у розмірі 20%.

4.10 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність проекту є комплексною характеристикою результативності використання ресурсів, витрачених на його реалізацію. Для визначення ефективності кваліфікаційної роботи бакалавра необхідно розрахувати такі економічні показники, як прибуток, економічна ефективність та термін окупності.

Прибуток визначається за формулою:

$$П = Ц - С_в \quad (4.12)$$

де:

П – прибуток, грн;

Ц – ціна науково-дослідної роботи, грн;

$С_в$ – собівартість виконання роботи, грн.

Розрахунок прибутку:

$$П = Ц - С_в = 110368,08 - 70749,28 = 39618,80 \text{ грн}$$

Економічна ефективність ($Е_р$) визначається відношенням прибутку до собівартості виконання робіт і обчислюється за формулою:

$$Е_р = \frac{П}{С_в} \quad (4.13)$$

Розрахунок економічної ефективності:

$$Е_р = \frac{П}{С_в} = \frac{39618,80}{70749,28} = 0,56$$

Термін окупності капітальних вкладень ($Т_р$) визначається за формулою:

$$Т_р = \frac{1}{Е_р} \quad (4.14)$$

Розрахунок терміну окупності:

$$Т_р = \frac{1}{Е_р} = \frac{1}{0,56} = 1,79 \text{ року}$$

Допустимим вважається термін окупності до 5 років. В даному випадку отриманий термін окупності 1,79 року є задовільним і підтверджує економічну доцільність реалізації кваліфікаційної роботи бакалавра.

Проведені розрахунки узагальнені у таблиці 4.5.

Таблиця 4.5 – Техніко-економічні показники кваліфікаційної роботи бакалавра розробки навчально-демонстраційного сайту «SkillUpCoding»

№	Показник	Одиниця виміру	Значення
1	Собівартість	грн.	70749,28
2	Плановий прибуток	грн.	39618,80
3	Ціна	грн.	110368,08
4	Економічна ефективність	–	0,56
5	Термін окупності	роки	1,79

Таким чином, загальна ціна розробки кваліфікаційної роботи бакалавра становить 110368,08 грн. При цьому, економічна ефективність має високе значення (0,56), що свідчить про вигідність реалізації проекту та його окупність протягом 1,79 року.

5 ОХОРОНА ПРАЦІ ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

5.1 Розрахунок системи штучного освітлення для приміщення, де здійснюється розробка сайту

У сучасному цифровому світі праця програміста, веброзробника чи будь-якого іншого спеціаліста з інформаційних технологій передбачає тривале перебування за комп'ютером. У процесі створення сайту, розробки коду, налагодження інтерфейсів або аналізу результатів тестування ключову роль відіграє саме зорове навантаження. Саме тому освітлення робочого місця розробника є не просто фактором комфорту, а елементом, який напряму впливає на ефективність, продуктивність і здоров'я людини. Робота за екраном монітора передбачає значне навантаження на зір, що при неправильному освітленні може швидко призводити до перевтомлення, головного болю та погіршення концентрації. Недостатнє освітлення також спричиняє відблиски на моніторі та тіні на робочих поверхнях, що може змушувати людину займати незручну позу і додатково навантажувати м'язово-скелетну систему.

У кваліфікаційній роботі бакалавра «SkillUpCoding» правильне освітлення є однією з ключових вимог, що впливає на якість і швидкість розробки сайту. Освітлення робочого місця має відповідати державним нормам України, зокрема ДБН В.2.5-28:2018 «Природне і штучне освітлення» [1].

Згідно з вимогами ДБН, у приміщеннях, де провадиться робота з відеодисплейними терміналами, до яких належать монітори комп'ютерів, рівень освітленості на робочій поверхні має становити не менше ніж 500 люксів (лк). Цей показник забезпечує достатню яскравість для комфортного

зорового сприймання тексту, графіки, елементів інтерфейсу, коду, що відображаються на екрані.

Важливим є й дотримання однорідності освітлення, що означає уникнення різких перепадів яскравості на різних ділянках приміщення. Високий контраст між світлими і темними зонами може спричинити втому очей та зниження концентрації. Тому в нормативних документах також вказується коефіцієнт нерівномірності освітлення, який для ІТ-приміщень не повинен перевищувати 1,3.

Ще одним обов’язковим параметром є відсутність сліпучих відблисків, які виникають при неправильному розташуванні джерел світла відносно монітора. Вони можуть негативно впливати на зір і викликати дискомфорт. У нормативних рекомендаціях наголошується на необхідності уникати прямого потрапляння світла в очі працівника або на екран, де працює програміст.

Вимоги також охоплюють питання типу джерел світла. Найбільш рекомендованими для офісної та інженерної праці є світлодіодні (LED) світильники, які забезпечують високу світлову віддачу, мають тривалий ресурс служби та не мерехтять, що важливо при тривалому зоровому навантаженні.

Таким чином, усі вищезазначені норми регламентують, яким має бути безпечне й ефективне освітлення для праці з комп’ютером. При реалізації кваліфікаційної роботи бакалавра «SkillUpCoding» ці нормативи були враховані під час вибору світильників, визначення їхньої кількості та розміщення у приміщенні.

Розробка кваліфікаційної роботи бакалавра – навчально-демонстраційного сайту «SkillUpCoding» – проводилася у реальних умовах, у приміщенні, яке слугувало повноцінним робочим простором протягом усього циклу створення вебресурсу. Саме це середовище було взято за основу для

проведення точного розрахунку освітлення. У приміщенні були створені необхідні умови для роботи: стабільний Інтернет, комп’ютерне обладнання та достатнє природне освітлення.

Розміри приміщення — 4 м × 5 м, висота стелі — 2,7 м, що є типовим для офісних або житлових приміщень. Стіни та стеля світлого кольору, підлога середньої світлості. Відповідно, коефіцієнт використання світлового потоку прийнято n = 0,6. Робоче місце (стіл з комп’ютером) розташовано біля вікна таким чином, щоб максимально використовувати природне світло зліва та зменшити відблиски на екрані.

Світлодіодні світильники (LED 600×600 мм, потужністю 36 Вт, світловий потік 3200 лм) встановлені на стелі в чотири поздовжні ряди, по два світильники в кожному ряду. Це дозволяє рівномірно покрити всю площу приміщення освітленням та уникнути зон із недостатньою яскравістю.

У межах кожного ряду світильники встановлені з інтервалом 1,5 метра між собою. Відстань між паралельними рядами становить 1,15 метра, що забезпечує рівномірний розподіл світлового потоку в обох напрямках — як уздовж, так і впоперек приміщення. Завдяки такій конфігурації виключається поява «мертвих зон» або надлишково освітлених ділянок.

Робоче місце з комп’ютером розташовано в межах охоплення щонайменше двох суміжних світильників, що дозволяє досягти нормативного рівня освітленості (500 лк) без додаткових джерел світла. Це положення забезпечує візуальний комфорт та мінімізує навантаження на зір при тривалій роботі.

Таке розташування дозволяє досягти рівномірного розподілу світлового потоку по площі всього приміщення та виключає утворення темних або надмірно освітлених ділянок. Монтаж світильників на рівномірній висоті по

центральній осі кімнати дозволяє забезпечити ідеальний кут падіння світла, що особливо важливо для роботи з екранами.

Загалом, враховуючи площу приміщення, його геометрію, тип джерел світла та особливості розміщення меблів і техніки, дана кімната є цілком придатною для виконання проектної ІТ-діяльності. Всі характеристики були враховані при подальших інженерних розрахунках системи штучного освітлення, що дозволило забезпечити освітленість у межах нормативних 500 лк.

Для забезпечення нормативного рівня освітлення в приміщенні, де здійснюється розробка навчально-демонстраційного сайту «SkillUpCoding», було здійснено технічно обґрунтований розрахунок кількості світильників, необхідних для створення комфортних умов роботи за комп'ютером. Розрахунок проводиться на основі методики, передбаченої ДБН В.2.5-28-2018 «Природне і штучне освітлення», з використанням формули визначення кількості джерел світла на площу приміщення.

Для досягнення нормативних 500 лк використовуються LED-світильники потужністю 36 Вт (3200 лм).

Розрахунок кількості світильників проводиться за формулою:

$$N = \frac{E \cdot S \cdot k}{F \cdot n \cdot Z} \tag{5.1}$$

де:

– E — нормативна освітленість, лк (500 лк);
– S — площа приміщення, м² (20 м²);
– k — коефіцієнт запасу (1,5);
– F — світловий потік одного світильника, лм (3200 лм);

– n — коефіцієнт використання світлового потоку (0,6);

– Z — коефіцієнт нерівномірності освітлення (1,1).

Підставляючи всі значення у формулу, отримаємо:

$$N = \frac{500 \cdot 20 \cdot 1{,}5}{3200 \cdot 0{,}6 \cdot 1{,}1} = \frac{15000}{2112} \approx 7{,}1 \quad (5.2)$$

Розрахункова кількість світильників — 8 одиниць. На практиці було використано саме 8 світильників, що відповідає нормативним вимогам та забезпечує достатню рівномірність і інтенсивність освітлення в приміщенні. Правильна конфігурація їх розміщення дозволила уникнути будь-яких темних зон та забезпечила комфортні умови для роботи.

Таким чином, використані 8 LED-світильників повністю відповідають розрахункам та забезпечують необхідний нормативний рівень освітленості для комфортної роботи. Проведені вимірювання і спостереження підтвердили, що система освітлення, змонтована відповідно до наведених розрахунків і рекомендацій, ефективно забезпечує нормативну освітленість і створює сприятливі умови для комфортної та продуктивної роботи за комп'ютером.

Після виконання розрахунків кількості світильників для забезпечення нормативного освітлення, наступним важливим етапом стало проектування схеми їх розміщення. Раціональне розташування джерел світла в приміщенні є не менш важливим за саму їх кількість. Неправильне розміщення світильників може спричинити появу тіней, «мертвих зон» без належного освітлення або навпаки — надмірно яскравих ділянок, що викликають зорове перенапруження.

Світильники розміщені на стелі симетрично у 4 ряди по два, з чітко визначеними відстанями між ними (1,5 м між парами вздовж короткої сторони і 1,15 м між рядами вздовж довгої сторони), що дозволило досягти максимально ефективного освітлення у всьому приміщенні.

Світильники розміщені симетрично, що забезпечує рівномірне освітлення приміщення і комфортну роботу за комп’ютером. Це рішення стало результатом практичних вимірювань, спостережень і попереднього моделювання схеми освітлення, завдяки чому вдалося досягти якісного освітлення без перевантаження очей під час тривалої роботи.

Усі відстані, розташування світильників та орієнтація робочого місця представлені у графічній схемі планування, що наведена нижче (див. рис. 5.1):

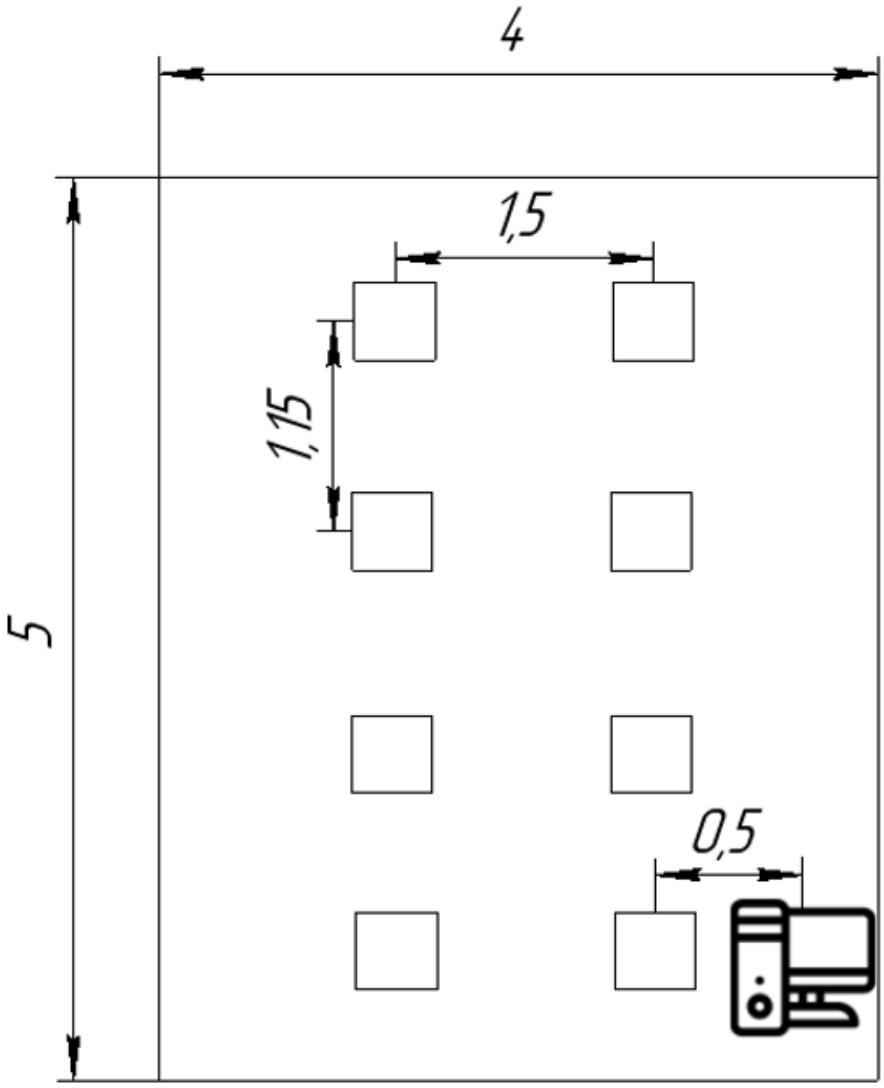


Рисунок 5.1 – Схема розміщення світильників у приміщенні розробки сайту SkillUpCoding

Практичні вимірювання підтвердили, що реальна освітленість у приміщенні відповідає нормативним розрахункам, особливо у зоні робочого місця. Використання світлодіодних світильників забезпечує енергоефективність, безпеку та комфорт.

Таким чином, освітлення приміщення відповідає всім нормам і забезпечує комфортні умови для роботи з комп'ютером.

5.2 Основні державні нормативні акти в сфері ІТ

Інформаційні технології в сучасному світі відіграють ключову роль у розвитку всіх сфер економіки, освіти, науки та культури. В Україні, як і в більшості інших держав, функціонування та розвиток ІТ-сфери регулюється цілою низкою державних нормативних актів та стандартів, спрямованих на забезпечення її ефективності, безпеки та відповідності міжнародним вимогам.

Основною метою нормативно-правового регулювання ІТ-діяльності є створення прозорих, передбачуваних умов для здійснення підприємницької діяльності, захисту інтересів усіх учасників ринку інформаційних технологій, а також забезпечення інформаційної безпеки держави та її громадян.

До основних державних нормативних актів, що регламентують діяльність у сфері інформаційних технологій, належать:

1. Закон України «Про інформацію» — закон визначає загальні принципи інформаційних відносин у державі, регулює право власності на інформацію, забезпечення доступу до інформації та встановлює відповідальність за її незаконне використання. Закон також встановлює основні засади захисту персональних даних та приватності користувачів.

2. Закон України «Про захист інформації в інформаційно телекомунікаційних системах» — цей нормативний акт визначає правові основи захисту інформації від несанкціонованого доступу, пошкодження або витоку в інформаційних системах різного рівня. Документ встановлює вимоги щодо створення, використання та сертифікації систем захисту інформації.

3. Закон України «Про електронні комунікації» — закон регулює питання надання послуг електронних комунікацій, включаючи роботу операторів мобільного зв'язку, інтернет-провайдерів, стандарти якості послуг та їх доступність для користувачів. Документ сприяє модернізації телекомунікаційної інфраструктури та впровадженню інноваційних технологій.

4. Закон України «Про електронну комерцію» — закон визначає принципи здійснення комерційних операцій через мережу Інтернет, захист прав споживачів у електронній торгівлі, вимоги до укладання електронних договорів та їх виконання, а також відповідальність сторін у разі порушення встановлених норм.

5. Закон України «Про захист персональних даних» — цей закон визначає основні принципи збору, обробки, зберігання, передачі та захисту персональних даних. Він особливо важливий для IT-компаній, які розробляють програмні продукти, що працюють з персональною інформацією користувачів. Закон відповідає міжнародним стандартам, зокрема GDPR, забезпечуючи захист прав та свобод людини.

6. Закон України «Про кібербезпеку» — закон встановлює правові основи захисту державних інформаційних ресурсів, а також регулює діяльність органів державної влади, підприємств та організацій щодо протидії

кіберзагрозам і забезпечення стійкості критичної інформаційної інфраструктури.

7. Постанова Кабінету Міністрів України «Про затвердження вимог у сфері інформаційної безпеки» — постанова містить конкретні технічні та організаційні вимоги до систем інформаційної безпеки, що використовуються у державних органах та установах. Вона визначає основні механізми контролю і заходи забезпечення захисту інформаційних ресурсів держави.

8. Державні стандарти України (ДСТУ) — для ІТ-галузі особливе значення мають державні стандарти (ДСТУ), які встановлюють вимоги до якості програмних продуктів, інформаційних систем, технологій зберігання та передачі даних, кібербезпеки, шифрування інформації, та інших аспектів роботи ІТ-сфери. До найважливіших відносяться ДСТУ ISO/IEC 27001 (інформаційна безпека), ДСТУ ISO 9001 (система управління якістю в ІТ), ДСТУ ISO/IEC 12207 (життєвий цикл програмного забезпечення).

Окрему увагу в контексті забезпечення якості програмних продуктів та послуг слід приділити саме стандарту ДСТУ ISO/IEC 12207, що визначає процеси та процедури управління повним життєвим циклом програмного забезпечення. Даний стандарт регламентує основні етапи: планування, аналіз вимог, проектування, кодування, тестування, впровадження та супровід програмних систем. Використання цього стандарту дозволяє досягти високого рівня якості та надійності створюваного програмного забезпечення, а також ефективного управління ризиками та витратами на всіх етапах його розробки та використання.

Наступною важливою групою нормативних актів є міжнародні стандарти серії ISO/IEC 27000, впроваджені в Україні як ДСТУ ISO/IEC 27000. Вони стосуються інформаційної безпеки та управління ризиками в

інформаційних системах і технологіях. Серед них особливе місце займає стандарт ДСТУ ISO/IEC 27001, який описує системи менеджменту інформаційної безпеки (СМІБ) та надає чіткий механізм побудови і підтримки ефективної системи захисту інформації. Застосування цього стандарту в IT-компаніях сприяє створенню надійного середовища, що відповідає міжнародним вимогам і дозволяє успішно вести діяльність як на внутрішньому, так і на зовнішньому ринках.

Важливо також згадати стандарт ДСТУ ISO/IEC 20000, який регулює систему управління IT-послугами (IT Service Management). Цей стандарт є ключовим для організацій, які займаються наданням IT-послуг. Він визначає принципи та процедури забезпечення високої якості сервісу, чітке управління ресурсами, відповідність потребам користувачів, а також ефективну взаємодію між замовниками та виконавцями IT-послуг.

Окреме значення для сфери IT має Постанова Кабінету Міністрів України № 869 «Про затвердження Загальних вимог до кіберзахисту об'єктів критичної інформаційної інфраструктури». Ця постанова встановлює комплекс вимог до захисту державних інформаційних ресурсів, а також критично важливої інфраструктури, яка включає об'єкти енергетики, фінансової сфери, транспорту, комунікацій та інших важливих галузей. Документ забезпечує єдині підходи до реалізації кіберзахисту в масштабах всієї країни та визначає відповідальність за впровадження необхідних заходів.

Додатково до цих нормативних актів варто також згадати Рішення Ради національної безпеки і оборони України «Про Стратегію кібербезпеки України», яка затверджена Указом Президента України. Стратегія визначає основні напрями та пріоритети державної політики у сфері кібербезпеки, забезпечення інформаційного суверенітету, боротьби з кіберзлочинністю та захисту прав громадян в інформаційному просторі.

Таким чином, державна політика України у сфері інформаційних технологій базується на комплексній нормативній базі, яка включає як національні закони та постанови уряду, так і міжнародні стандарти, імплементовані у вітчизняне законодавство. Суворе дотримання цих актів і стандартів дозволяє забезпечити стабільність, безпеку та конкурентоспроможність вітчизняного IT-сектору на міжнародному рівні, а також сприяє формуванню сучасного інформаційного суспільства.

ВИСНОВОК

У ході виконання кваліфікаційної роботи бакалавра було спроектовано, реалізовано, протестовано та задокументовано повноцінний навчально-демонстраційний вебсайт SkillUpCoding, основною метою якого є сприяння ефективному засвоєнню основ програмування через поєднання теоретичного матеріалу, інтерактивних тестів і обміну фрагментами коду між користувачами.

В межах проекту реалізовано:

– інтуїтивно зрозумілу, адаптивну структуру сайту з підтримкою мобільних пристроїв;

– розділи з теорією та практичними прикладами по HTML, CSS, JavaScript, Python, алгоритмах тощо;

– модуль тестування знань із фіксацією результатів у профілі користувача;

– систему облікових записів з автентифікацією, персональними профілями та історією навчання;

– функціонал публікації та обговорення snippet-ів — фрагментів коду з підсвічуванням синтаксису;

– сучасну клієнт-серверну архітектуру на базі HTML, CSS, JavaScript, Node.js, Express і MongoDB;

– захист даних користувачів та перевірку на валідність введення;

– технічну документацію та інструкції з розгортання, обслуговування й адміністрування системи.

Під час реалізації проекту були враховані державні вимоги з охорони праці, зокрема щодо освітлення робочого місця розробника. Проведено розрахунок кількості світильників для забезпечення нормативної освітленості

приміщення, а також дотримано екологічних вимог у частині вибору енергоефективного обладнання.

У рамках економічного обґрунтування доведено доцільність реалізації проекту як із навчальної, так і з комерційної точки зору. Загальна вартість розробки вклалася у прийнятний бюджет, а термін окупності проекту — не більше 2–3 років.

Таким чином, поставлені цілі кваліфікаційної роботи бакалавра досягнуто повністю. Розроблений сайт відповідає технічним вимогам, є зручним для користувачів та має потенціал до подальшого розвитку й масштабування в освітніх або комерційних цілях. Проект SkillUpCoding може ефективно застосовуватись у навчальних закладах, на онлайн-курсах або в IT-спільнотах для підтримки практичного навчання програмуванню.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДБН В.2.5-28:2018. Природне і штучне освітлення : державні будівельні норми України. Київ : Мінрегіон України, 2018. 74 с.

2. Шевчук В. Я. Основи веб-програмування : навч. посіб. Львів : Видавництво ЛНУ ім. Івана Франка, 2020. 224 с.

3. Ляшенко О. О., Кривко А. В. Практичне програмування JavaScript. Харків : ФОП Коваленко, 2021. 196 с.

4. Crockford D. JavaScript: The Good Parts. Sebastopol : O'Reilly Media, 2008. 176 p.

5. Поліщук М. І. HTML і CSS: основи створення сучасних веб-сайтів. Київ : Академвидав, 2022. 312 с.

6. Web Content Accessibility Guidelines (WCAG) 2.1 [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/WCAG21/. – Дата звернення: 01.06.2025.

7. Mozilla Developer Network (MDN). Web Docs: HTML, CSS, JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org. – Дата звернення: 01.06.2025. – Заголовок з екрану.

8. GitHub Pages Documentation. Hosting static websites [Електронний ресурс]. – Режим доступу: https://pages.github.com. – Дата звернення: 01.06.2025.

ДОДАТКИ

Додаток А. Лістинг файлу form.js

```
const scrollToTopBtn = document.getElementById("scrollToTopBtn");

window.onscroll = () => {
  scrollToTopBtn.classList.toggle("show", window.scrollY > 300);
};

scrollToTopBtn.addEventListener("click", () => {
  window.scrollTo({ top: 0, behavior: "smooth" });
});

document.querySelectorAll('a[href^="#"]:not([href="#"])').forEach((anchor) => {
  anchor.addEventListener("click", (e) => {
    e.preventDefault();
    const href = anchor.getAttribute("href");
    const target = document.querySelector(href);
    if (target) {
      target.scrollIntoView({ behavior: "smooth" });
    }
  });
```

```
});

document.addEventListener("DOMContentLoaded", () => {
  const currentPage = window.location.pathname.split("/").pop();
  const allMenuLinks = document.querySelectorAll(
    ".desktop-menu a, .offcanvas-menu a"
  );
  allMenuLinks.forEach((link) => {
    link.classList.remove("active");
    if (link.getAttribute("href") === currentPage) {
      link.classList.add("active");
    }
  });

  const goToLanguagesBtn = document.getElementById("go-to-languages-btn");
  if (goToLanguagesBtn) {
    goToLanguagesBtn.addEventListener("click", () => {
      const internetSection = document.getElementById("Internet");
      if (internetSection) {
        internetSection.scrollIntoView({ behavior: "smooth" });
      }
    });
  }
```

```
const signInButton = document.getElementById("sign-in-button");
const signUpButton = document.getElementById("sign-up-button");
const profileButton = document.getElementById("profile-button");

const signInButtonM = document.getElementById("sign-in-button-m");
const signUpButtonM = document.getElementById("sign-up-button-m");
const profileButtonM = document.getElementById("profile-button-m");

const updateAuthButtons = () => {
  const isLoggedIn = localStorage.getItem("isLoggedIn") === "true";

  if (isLoggedIn) {
    if (signInButton) signInButton.style.display = "none";
    if (signUpButton) signUpButton.style.display = "none";
    if (profileButton) profileButton.style.display = "block";

    if (signInButtonM) signInButtonM.style.display = "none";
    if (signUpButtonM) signUpButtonM.style.display = "none";
    if (profileButtonM) profileButtonM.style.display = "block";
  } else {
    if (signInButton) signInButton.style.display = "block";
    if (signUpButton) signUpButton.style.display = "block";
```

```
    if (profileButton) profileButton.style.display = "none";

    if (signInButtonM) signInButtonM.style.display = "block";
    if (signUpButtonM) signUpButtonM.style.display = "block";
    if (profileButtonM) profileButtonM.style.display = "none";
  }
};
updateAuthButtons();

let isLoginMode = false;

if (signInButton) {
  signInButton.onclick = (e) => {
    e.preventDefault();
    isLoginMode = true;
    openRegistrationModal();
  };
}
if (signUpButton) {
  signUpButton.onclick = (e) => {
    e.preventDefault();
    isLoginMode = false;
    openRegistrationModal();
```

```
  };
}

if (signInButtonM) {
  signInButtonM.onclick = (e) => {
    e.preventDefault();
    isLoginMode = true;
    openRegistrationModal();
  };
}
if (signUpButtonM) {
  signUpButtonM.onclick = (e) => {
    e.preventDefault();
    isLoginMode = false;
    openRegistrationModal();
  };
}

const regModal = document.getElementById("registration-modal");
const authModalTitle = document.getElementById("auth-modal-title");
const switchToLogin = document.getElementById("switch-to-login");
const regForm = document.getElementById("registration-form");
```

```
  const closeRegModalBtn = document.getElementById("close-registration-
modal");


  const openRegistrationModal = () => {
    if (!regModal) return;
    regModal.style.display = "block";
    toggleAuthMode();
  };


  if (closeRegModalBtn) {
    closeRegModalBtn.onclick = () => {
      regModal.style.display = "none";
    };
  }


  const toggleAuthMode = () => {
    if (isLoginMode) {
      authModalTitle.textContent = "Вхід";
      regForm.querySelector("button").textContent = "Увійти";
      switchToLogin.innerHTML =
        'Не маєте аккаунта? <a href="#" id="to-login-link">Зареєструватися</a>';
      document.getElementById("name-field").style.display = "none";
      document.getElementById("name").removeAttribute("required");
```

```
    } else {
      authModalTitle.textContent = "Реєстрація";
      regForm.querySelector("button").textContent = "Зареєструватися";
      switchToLogin.innerHTML =
        'Вже маєте аккаунт? <a href="#" id="to-login-link">Увійти</a>';
      document.getElementById("name-field").style.display = "block";
      document.getElementById("name").setAttribute("required", "required");
    }
  };

  regForm.addEventListener("submit", (e) => {
    e.preventDefault();
    const name = document.getElementById("name").value.trim();
    const email = document.getElementById("email").value.trim();
    const password = document.getElementById("password").value.trim();

    if (isLoginMode) {

fetch("https://skillupcoding.onrender.com/login", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ email, password }),
})
```

```
  .then((res) => res.json())
  .then((data) => {
    if (data.userId && data.token) {
      localStorage.setItem("userId", data.userId);
      localStorage.setItem("userName", data.name);
      localStorage.setItem("userEmail", data.email);
      localStorage.setItem("authToken", data.token);
      localStorage.setItem("isLoggedIn", "true");

      updateAuthButtons();
      regModal.style.display = "none";

      window.location.reload();

    } else {
      alert("Помилка входу: " + data.error);
    }
  })
  .catch((err) => console.error("Помилка входу:", err));
    } else {
fetch("https://skillupcoding.onrender.com/register", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
```

```
  body: JSON.stringify({ name, email, password }),
})
  .then((res) => res.json())
  .then((data) => {
    if (data.userId && data.token) {
      localStorage.setItem("userId", data.userId);
      localStorage.setItem("userName", data.name);
      localStorage.setItem("userEmail", data.email);
      localStorage.setItem("authToken", data.token);
      localStorage.setItem("isLoggedIn", "true");

      updateAuthButtons();
      regModal.style.display = "none";

      alert("Реєстрація успішна! Ви вже авторизовані.");

      window.location.reload();
    } else if (data.error) {
      alert("Помилка реєстрації: " + data.error);
    } else {
      console.log("Неочікувана відповідь при реєстрації:", data);
    }
  })
```

```
.catch((err) => console.error("Помилка реєстрації:", err));
  }
});

document.body.addEventListener("click", (e) => {
  if (e.target && e.target.id === "to-login-link") {
    e.preventDefault();
    isLoginMode = !isLoginMode;
    toggleAuthMode();
  }
});

const profileModal = document.getElementById("profile-modal");
const closeProfileModalBtn = document.getElementById("close-profile-modal");
const logoutButton = document.getElementById("logout-button");

if (profileButton) {
  profileButton.onclick = (e) => {
    e.preventDefault();
    openProfileModal();
  };
}
if (profileButtonM) {
```

```
  profileButtonM.onclick = (e) => {
    e.preventDefault();
    openProfileModal();
  };
}

const openProfileModal = () => {
  if (!profileModal) return;
  profileModal.style.display = "block";

  fetch("https://skillupcoding.onrender.com/user-info", {
    headers: {
      Authorization: "Bearer " + localStorage.getItem("authToken"),
    },
  })
    .then((res) => res.json())
    .then((data) => {
      if (data.error) {
        console.error("Помилка при завантаженні користувача:", data.error);
        document.getElementById("user-name").textContent = "Невідомо";
        document.getElementById("user-email").textContent = "Невідомо";
      } else {
        document.getElementById("user-name").textContent =
```

```
        data.name || "Невідомо";
      document.getElementById("user-email").textContent =
        data.email || "Невідомо";

      if (data.profilePhoto) {
        document.getElementById("profile-photo").src = data.profilePhoto;
      } else {
        document.getElementById("profile-photo").src = "";
      }

      if (data.aboutMe) {
        document.getElementById("aboutMe").value = data.aboutMe;
      } else {
        document.getElementById("aboutMe").value = "";
      }
    }
  })
  .catch((err) => console.error("Помилка завантаження user-info:", err));

const userId = localStorage.getItem("userId");
if (userId) {
  fetch(`https://skillupcoding.onrender.com/user-results/${userId}`, {
    headers: {
```

```
        Authorization: "Bearer " + localStorage.getItem("authToken"),
      },
    })
      .then((res) => res.json())
      .then((data) => {
        if (Array.isArray(data)) {
          const resultsList = document.getElementById("results-list");
          resultsList.innerHTML = "";
          data.forEach((result) => {
            const li = document.createElement("li");
            li.textContent = `Тест: ${result.testName}, Бали: ${result.score}, Час:
${result.timeTaken} сек.`;
            resultsList.appendChild(li);
          });
        } else {
          console.error(
            "Отримані дані не є масивом або є помилкою:",
            data
          );
        }
      })
      .catch((err) =>
        console.error("Помилка завантаження результатів:", err)
```

```
      );
    }
  };

  if (closeProfileModalBtn) {
    closeProfileModalBtn.onclick = () => {
      profileModal.style.display = "none";
    };
  }

  if (logoutButton) {
    logoutButton.onclick = () => {
      logout();
      profileModal.style.display = "none";
    };
  }

const logout = () => {
  localStorage.removeItem("isLoggedIn");
  localStorage.removeItem("userId");
  localStorage.removeItem("userName");
  localStorage.removeItem("userEmail");
  localStorage.removeItem("authToken");
```

```
  updateAuthButtons();
  window.location.reload();
};

  const profileForm = document.getElementById("profile-form");
  if (profileForm) {
    profileForm.addEventListener("submit", (e) => {
      e.preventDefault();
      const aboutMe = document.getElementById("aboutMe").value.trim();
      const photoFile = document.getElementById("uploadPicture").files[0];

      if (!aboutMe) {
        alert('Поле "Про себе" обов’язкове для заповнення.');
        return;
      }

      const formData = new FormData();
      formData.append("aboutMe", aboutMe);
      if (photoFile) {
        formData.append("profilePhoto", photoFile);
      }

      fetch("https://skillupcoding.onrender.com/update-profile", {
```

```
      method: "POST",
      headers: {
        Authorization: "Bearer " + localStorage.getItem("authToken"),
      },
      body: formData,
    })
      .then((res) => res.json())
      .then((data) => {
        if (data.success) {
          alert("Профіль успішно оновлено!");
          if (data.profilePhoto) {
            document.getElementById("profile-photo").src = data.profilePhoto;
          }
        } else {
          alert("Помилка оновлення профілю: " + data.error);
        }
      })
      .catch((err) => console.error("Помилка оновлення профілю:", err));
  });
}

window.onclick = function (event) {
  if (event.target === regModal) {
```

```
    regModal.style.display = "none";
  }
  if (event.target === profileModal) {
    profileModal.style.display = "none";
  }
};

let testsData = {};

fetch("data/tests.json")
  .then((response) => response.json())
  .then((data) => {
    testsData = data;
    console.log("Тести завантажені:", testsData);
    addEventListenersForTests();
  })
  .catch((error) => console.error("Помилка при завантаженні JSON:", error));

const addEventListenersForTests = () => {
  const sections = ["html-css", "javascript", "python", "csharp", "algorithms"];

  sections.forEach((section) => {
    const openTestBtn = document.getElementById(`go-to-${section}-test`);
```

```
    if (openTestBtn) {
      openTestBtn.onclick = () => openTestModal(`test-modal-${section}`,
section);
    }

    const closeModalBtn = document.getElementById(`close-modal-${section}`);
    if (closeModalBtn) {
      closeModalBtn.onclick = () => closeTestModal(`test-modal-${section}`);
    }

    const submitTestBtn = document.getElementById(`submit-test-${section}`);
    if (submitTestBtn) {
      submitTestBtn.onclick = () => submitTest(section);
    }
    const retryTestBtn = document.getElementById(`retry-test-${section}`);
    if (retryTestBtn) {
      retryTestBtn.onclick = () => retryTest(section);
    }
  });

  window.onclick = (event) => {
    if (
      event.target.classList &&
```

```
      event.target.classList.contains("modal")
    ) {
      event.target.style.display = "none";
    }
  };
};

let testStartTime = {};

const openTestModal = (modalId, testName) => {
  const modal = document.getElementById(modalId);
  if (!modal)
    return console.error(`Модальне вікно ${modalId} не знайдено.`);

  modal.style.display = "block";
  loadTestQuestions(testName);
  testStartTime[testName] = Date.now();
};

const closeTestModal = (modalId) => {
  const modal = document.getElementById(modalId);
  if (!modal) return;
  modal.style.display = "none";
```

```
};

const loadTestQuestions = (testName) => {
  if (!testsData || !testsData.tests) {
    console.error("Тести ще не завантажені або JSON-файл не знайдено.");
    return;
  }
  const selectedTest = testsData.tests.find((t) => t.section === testName);
  if (!selectedTest) {
    console.error(`Тест для розділу ${testName} не знайдено.`);
    return;
  }
  const container = document.getElementById(`test-questions-${testName}`);
  if (!container)
    return console.error(`Контейнер для питань ${testName} не знайдено.`);

  container.innerHTML = selectedTest.questions
    .map(
      (q, idx) =>
        `<div class="question-block">
          <p>${q.question}</p>` +
        q.options
          .map(
```

```
          (opt, optIdx) =>
            `<label>
              <input type="radio" name="question${idx}" value="${optIdx}">
              ${opt}
            </label><br>`
        )
        .join("") +
      `</div>`
  )
  .join("");

  container.style.display = "block";
  const submitBtn = document.getElementById(`submit-test-${testName}`);
  if (submitBtn) submitBtn.style.display = "block";
  const resultContainer = document.getElementById(
    `test-result-${testName}`
  );
  if (resultContainer) resultContainer.style.display = "none";
  const retryBtn = document.getElementById(`retry-test-${testName}`);
  if (retryBtn) retryBtn.style.display = "none";
};

const calculateScore = (testName) => {
```

```
  const testData = testsData.tests.find((t) => t.section === testName);
  if (!testData)
    return { scorePercentage: 0, correctAnswers: 0, totalQuestions: 0 };

  let correctAnswers = 0;
  testData.questions.forEach((question, idx) => {
    const selectedOption = document.querySelector(
      `input[name="question${idx}"]:checked`
    );
    if (
      selectedOption &&
      parseInt(selectedOption.value) === question.correctAnswer
    ) {
      correctAnswers++;
    }
  });

  const total = testData.questions.length;
  const percentage = (correctAnswers / total) * 100;
  return { scorePercentage: percentage, correctAnswers, totalQuestions: total };
};

const showResult = (testName) => {
```

```
    const { scorePercentage, correctAnswers, totalQuestions } =
      calculateScore(testName);
    let message;
    if (scorePercentage <= 20) {
      message = "Ти погано засвоїв матеріал.";
    } else if (scorePercentage <= 40) {
      message = "Тобі потрібно краще опрацювати матеріал.";
    } else if (scorePercentage <= 60) {
      message = "Середній рівень знань.";
    } else if (scorePercentage <= 80) {
      message = "Добре засвоїв матеріал.";
    } else {
      message = "Ти прекрасно засвоїв матеріал.";
    }

    const resultEl = document.getElementById(`test-result-${testName}`);
    if (!resultEl) return;
    resultEl.innerHTML = `<p>${message}</p>
      <p>Ви відповіли правильно на ${correctAnswers} з ${totalQuestions}
питань.</p>`;
    resultEl.style.display = "block";

    const questionsCtn = document.getElementById(`test-questions-${testName}`);
```

```
if (questionsCtn) questionsCtn.style.display = "none";
const submitBtn = document.getElementById(`submit-test-${testName}`);
if (submitBtn) submitBtn.style.display = "none";
const retryBtn = document.getElementById(`retry-test-${testName}`);
if (retryBtn) retryBtn.style.display = "block";

const userId = localStorage.getItem("userId");
if (!userId) {
  console.log("Користувач не авторизований, результат не збережено.");
  return;
}

let baseScore = Math.round((scorePercentage / 100) * 500);
let timeTaken = Math.floor((Date.now() - testStartTime[testName]) / 1000);
let timeBonus = Math.max(0, 100 - timeTaken);
let totalScore = baseScore + timeBonus;

fetch("https://skillupcoding.onrender.com/submit-result", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    Authorization: "Bearer " + localStorage.getItem("authToken"),
  },
```

```
      body: JSON.stringify({
        userId: parseInt(userId),
        testName,
        score: totalScore,
        timeTaken,
      }),
    })
      .then((res) => res.json())
      .then((data) => {
        console.log("Результат тесту збережено:", data);
      })
      .catch((err) => console.error("Помилка збереження результату:", err));
  };

  const submitTest = (testName) => {
    showResult(testName);
  };

  const retryTest = (testName) => {
    loadTestQuestions(testName);
  };
});
```

```
function loadSnippets(sortValue = "dateDesc") {
  let url = "https://skillupcoding.onrender.com/snippets";

  if (sortValue === "likesDesc") {
    url += "?sort=likesDesc";
  } else if (sortValue === "dateAsc") {
    url += "?sort=dateAsc";
  } else if (sortValue === "languageAsc") {
    url += "?sort=languageAsc";
  } else {
    url += "?sort=dateDesc";
  }

  fetch(url)
    .then((res) => res.json())
    .then((data) => {
      renderSnippets(data);
    })
    .catch((err) => console.error("Помилка завантаження snippets:", err));

  const closeSnippetModalBtn = document.getElementById("close-snippet-
modal");
  if (closeSnippetModalBtn) {
```

```
    closeSnippetModalBtn.onclick = () => {
      const modal = document.getElementById("snippet-modal");
      if (modal) modal.style.display = "none";
    };
  }

  const addCommentBtn = document.getElementById("modal-addCommentBtn");
  if (addCommentBtn) {
    addCommentBtn.onclick = () => {
      const text = document.getElementById("modal-commentText").value.trim();
      if (!text) return;
      addComment(currentSnippetId, text);
      document.getElementById("modal-commentText").value = "";
    };
  }
}

function renderSnippets(snippets) {
  const container = document.getElementById("snippetsList");
  if (!container) return;
  container.innerHTML = "";

  const currentUserId = localStorage.getItem("userId") || "";
```

```
snippets.forEach((snippet) => {
  const div = document.createElement("div");
  div.classList.add("snippet-card");

  div.style.margin = "10px 0";
  div.style.padding = "10px";

  const langClass = snippet.language
    ? `language-${snippet.language.toLowerCase()}`
    : "language-plaintext";

  div.innerHTML = `
    <h4>${snippet.title} <small>(${snippet.language || "NoLang"})</small></h4>
    <p>Автор: ${snippet.userName}</p>
    <pre><code class="${langClass}">${snippet.codeText}</code></pre>
  `;

  const likeContainer = document.createElement("div");
  likeContainer.classList.add("snippet-like-container");
  likeContainer.onclick = () => likeSnippet(snippet.id);

  const likeCount = document.createElement("span");
```

```
likeCount.classList.add("snippet-like-count");
likeCount.textContent = snippet.likes || 0;

const likeIcon = document.createElement("img");
likeIcon.classList.add("snippet-like-icon");
likeIcon.src = "img/like.png";

likeContainer.appendChild(likeCount);
likeContainer.appendChild(likeIcon);
div.appendChild(likeContainer);

if (parseInt(snippet.userId) === parseInt(currentUserId)) {
  const editBtn = document.createElement("button");
  editBtn.textContent = "Edit";
  editBtn.classList.add("nav-btn");
  editBtn.style.marginRight = "8px";
  editBtn.onclick = () => editSnippet(snippet);

  const delBtn = document.createElement("button");
  delBtn.textContent = "Delete";
  delBtn.classList.add("nav-btn");
  delBtn.onclick = () => deleteSnippet(snippet.id);
```

```
      div.appendChild(editBtn);
      div.appendChild(delBtn);
    }

    const viewCommentBtn = document.createElement("button");
    viewCommentBtn.textContent = "Переглянути / Коментувати";
    viewCommentBtn.classList.add("nav-btn");
    viewCommentBtn.style.margin = "8px";

    viewCommentBtn.onclick = () => openSnippetModal(snippet);

    div.appendChild(viewCommentBtn);

    container.appendChild(div);
  });

  hljs.highlightAll();
}

let currentSnippetId = null;

function openSnippetModal(snippet) {
  currentSnippetId = snippet.id;
```

```
  const modal = document.getElementById("snippet-modal");
  if (!modal) return;

  document.getElementById("modal-snippet-title").textContent = snippet.title;
  document.getElementById("modal-snippet-author").textContent =
snippet.userName || "Невідомо";
  const codeEl = document.getElementById("modal-snippet-code");
  codeEl.textContent = snippet.codeText;

  modal.style.display = "block";

  loadComments(snippet.id);
}

function createSnippet() {
  const token = localStorage.getItem("authToken");
  if (!token) {
    alert("Спершу увійдіть у систему!");
    return;
  }

  const titleEl = document.getElementById("snippetTitle");
```

```
const codeEl = document.getElementById("snippetCode");
const langEl = document.getElementById("snippetLang");

if (!titleEl || !codeEl) {
  console.log("Елементи не знайдені: #snippetTitle, #snippetCode");
  return;
}

const title = titleEl.value.trim();
const codeText = codeEl.value.trim();
const language = langEl.value.trim();

if (!title || !codeText) {
  alert("Title і CodeText обов'язкові!");
  return;
}

fetch("https://skillupcoding.onrender.com/snippets", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    Authorization: "Bearer " + token,
  },
```

```
    body: JSON.stringify({ title, codeText, language }),
  })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        alert("Snippet створено!");
        titleEl.value = "";
        codeEl.value = "";
        langEl.value = "";
        loadSnippets();
      } else {
        alert("Помилка: " + (data.error || ""));
      }
    })
    .catch((err) => console.error("Помилка createSnippet:", err));
}

function deleteSnippet(snippetId) {
  const token = localStorage.getItem("authToken");
  if (!token) {
    alert("Увійдіть, щоб видалити snippet.");
    return;
  }
```

```
  if (!confirm("Видалити snippet?")) return;

  fetch(`https://skillupcoding.onrender.com/snippets/${snippetId}`, {
    method: "DELETE",
    headers: {
      Authorization: "Bearer " + token,
    },
  })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        alert("Snippet видалено.");
        loadSnippets();
      } else {
        alert("Помилка при видаленні: " + data.error);
      }
    })
    .catch((err) => console.error("Помилка deleteSnippet:", err));
}

function editSnippet(snippet) {
  const token = localStorage.getItem("authToken");
  if (!token) {
```

```
  alert("Спершу увійдіть!");
  return;
}
const newTitle = prompt("Новий заголовок:", snippet.title);
if (newTitle === null) return;
const newLang = prompt("Мова:", snippet.language || "");
if (newLang === null) return;
const newCode = prompt("Новий codeText:", snippet.codeText);
if (newCode === null) return;

fetch(`https://skillupcoding.onrender.com/snippets/${snippet.id}`, {
  method: "PUT",
  headers: {
    "Content-Type": "application/json",
    Authorization: "Bearer " + token,
  },
  body: JSON.stringify({
    title: newTitle,
    codeText: newCode,
    language: newLang,
  }),
})
  .then((res) => res.json())
```

```
    .then((data) => {
      if (data.success) {
        alert("Snippet оновлено!");
        loadSnippets();
      } else {
        alert("Помилка оновлення: " + data.error);
      }
    })
    .catch((err) => console.error("Помилка editSnippet:", err));
}

function likeSnippet(snippetId) {
  const token = localStorage.getItem("authToken");
  if (!token) {
    alert("Будь ласка, увійдіть, щоб лайкати Snippet.");
    return;
  }

  if (!snippetId) {
    console.error("Немає ID snippet'a для лайку!");
    return;
  }
```

```
  fetch(`https://skillupcoding.onrender.com/snippets/${snippetId}/like`, {
    method: "POST",
    headers: {
      Authorization: "Bearer " + token,
    },
  })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        loadSnippets();
      } else {
        alert("Помилка лайку: " + (data.error || "Невідома"));
      }
    })
    .catch((err) => {
      console.error("Помилка likeSnippet:", err);
      alert("Сталася помилка при виконанні запиту на лайк.");
    });
}

function timeAgo(isoDateString) {
  const date = new Date(isoDateString);
  const now = new Date();
```

```
let diffInSeconds = (now - date) / 1000;

if (diffInSeconds < 60) {
  return "менше хвилини тому";
}

let minutes = Math.floor(diffInSeconds / 60);
if (minutes < 60) {
  return minutes + " хв. тому";
}

let hours = Math.floor(minutes / 60);
if (hours < 24) {
  return hours + " год. тому";
}

let days = Math.floor(hours / 24);
if (days < 30) {
  return days + " дн. тому";
}

let months = Math.floor(days / 30);
if (months < 12) {
```

```
    return months + " міс. тому";
  }

  let years = Math.floor(months / 12);
  return years + " р. тому";
}

function loadComments(snippetId) {
  fetch(`https://skillupcoding.onrender.com/snippets/${snippetId}/comments`)
    .then(res => res.json())
    .then(data => {
      renderComments(data, snippetId);
    })
    .catch(err => console.error("Помилка завантаження коментарів:", err));
}

function renderComments(comments, snippetId) {
  const container = document.getElementById("commentsList");
  if (!container) return;
  container.innerHTML = "";

  comments.forEach(comment => {
    const div = document.createElement("div");
```

```
div.style.marginLeft = comment.parentId ? "30px" : "0";

const dateStr = timeAgo(comment.dateCreated);

div.innerHTML = `
  <p>
    <strong>${comment.userName}</strong> &bull;
    <small>${dateStr}</small>
  </p>
  <p>${comment.commentText}</p>
`;

const replyBtn = document.createElement("button");
replyBtn.classList.add("nav-btn");
replyBtn.textContent = "Відповісти";
replyBtn.style.marginRight = "5px";

replyBtn.onclick = () => {
  const text = prompt("Ваш коментар (відповідь):");
  if (text) {
    addComment(snippetId, text, comment.id);
  }
```

```
};
div.appendChild(replyBtn);

const currentUserId = localStorage.getItem("userId") || "";
if (parseInt(comment.userId) === parseInt(currentUserId)) {
  const editBtn = document.createElement("button");
  editBtn.classList.add("nav-btn");
  editBtn.textContent = "Редагувати";
  editBtn.style.marginRight = "5px";
  editBtn.onclick = () => {
    const newText = prompt("Новий текст:", comment.commentText);
    if (newText !== null) {
      editComment(comment.id, newText, snippetId);
    }
  };
  div.appendChild(editBtn);

  const delBtn = document.createElement("button");
  delBtn.classList.add("nav-btn");
  delBtn.textContent = "Видалити";
  delBtn.style.marginRight = "5px";
  delBtn.onclick = () => {
    if (confirm("Видалити коментар?")) {
```

```
          deleteComment(comment.id, snippetId);
        }
      };
      div.appendChild(delBtn);
    }

    container.appendChild(div);
  });
}

function addComment(snippetId, commentText, parentId = null) {
  const token = localStorage.getItem("authToken");
  if (!token) {
    alert("Спершу увійдіть!");
    return;
  }
  fetch(`https://skillupcoding.onrender.com/snippets/${snippetId}/comments`, {
    method: 'POST',
    headers: {
      "Content-Type": "application/json",
      Authorization: "Bearer " + token
    },
    body: JSON.stringify({ commentText, parentId })
```

```
  })
    .then(res => res.json())
    .then(data => {
      if (data.success) {
        loadComments(snippetId);
      } else {
        alert("Помилка додавання коментаря: " + (data.error || ""));
      }
    })
    .catch(err => console.error("Помилка addComment:", err));
}

function editComment(commentId, newText, snippetId) {
  const token = localStorage.getItem("authToken");
  if (!token) {
    alert("Спершу увійдіть!");
    return;
  }
  fetch(`https://skillupcoding.onrender.com/comments/${commentId}`, {
    method: 'PUT',
    headers: {
      "Content-Type": "application/json",
      Authorization: "Bearer " + token
```

```
    },
    body: JSON.stringify({ newText })
  })
    .then(res => res.json())
    .then(data => {
      if (data.success) {
        loadComments(snippetId);
      } else {
        alert("Помилка редагування: " + (data.error || ''));
      }
    })
    .catch(err => console.error("Помилка editComment:", err));
}

function deleteComment(commentId, snippetId) {
  const token = localStorage.getItem("authToken");
  if (!token) {
    alert("Спершу увійдіть!");
    return;
  }
  fetch(`https://skillupcoding.onrender.com/comments/${commentId}`, {
    method: 'DELETE',
    headers: {
```

```
      Authorization: "Bearer " + token
    }
  })
    .then(res => res.json())
    .then(data => {
      if (data.success) {
        loadComments(snippetId);
      } else {
        alert("Помилка видалення коментаря: " + (data.error || ""));
      }
    })
    .catch(err => console.error("Помилка deleteComment:", err));
}

document.addEventListener("DOMContentLoaded", () => {
  const snippetListEl = document.getElementById("snippetsList");
  if (snippetListEl) {
    loadSnippets("dateDesc");

    const createBtn = document.getElementById("createSnippetBtn");
    if (createBtn) {
      createBtn.addEventListener("click", createSnippet);
    }
```

```
    const sortSelect = document.getElementById("sortSnippets");
    const sortApplyBtn = document.getElementById("sortSnippetsApply");

    if (sortApplyBtn && sortSelect) {
      sortApplyBtn.addEventListener("click", () => {
        const sortValue = sortSelect.value;
        loadSnippets(sortValue);
      });
    }
  }
});
```

Додаток Б. Лістинг модального вікна < RegistrationModal />

```
<div id="registration-modal" class="modal">
  <div class="modal-content">
    <span id="close-registration-modal" class="close-btn">&times;</span>
    <h2 id="auth-modal-title">Реєстрація</h2>
    <form id="registration-form">
      <div id="name-field">
        <label for="name">Ім'я:</label>
        <input type="text" id="name" name="name" required />
      </div>
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required />
      <label for="password">Пароль:</label>
      <input type="password" id="password" name="password" required />
      <button type="submit" class="main-page-btn">Зареєструватися</button>
    </form>
    <p id="switch-to-login">
      Вже маєте аккаунт? <a href="#" id="to-login-link">Увійти</a>
    </p>
  </div>
</div>
```

Додаток В. Лістинг модального вікна <ProfileModal />

```
<div id="profile-modal" class="modal">
    <div class="modal-content">
      <span id="close-profile-modal" class="close-btn">&times;</span>
      <h2>Ваш профіль</h2>
      <form id="profile-form">
        <img
          id="profile-photo"
          src=""
          alt="Фото профілю"
          style="
            width: 150px;
            height: 150px;
            object-fit: cover;
            display: block;
            margin-bottom: 15px;
          "
        />
        <label for="uploadPicture">Ваше фото:</label>
        <input type="file" id="uploadPicture" name="profilePhoto" />
        <p>Ім'я: <span id="user-name"></span></p>
        <p>Email: <span id="user-email"></span></p>
```

```
      <label for="aboutMe">Про себе:</label>
      <textarea id="aboutMe" placeholder="Напишіть про себе..."></textarea>
      <button type="submit" class="main-page-btn">Зберегти</button>
    </form>
    <div id="user-results">
      <h3>Ваші результати тестів:</h3>
      <ul id="results-list"></ul>
    </div>
    <button id="logout-button" class="main-page-btn">Вийти</button>
  </div>
</div>
```

Додаток Г. Лістинг модального вікна <TestModalAlgorithms />

```
<div id="test-modal-html-css" class="modal">
    <div class="modal-content">
      <span id="close-modal-html-css" class="close-btn">&times;</span>
      <h2>Тест на знання HTML/CSS</h2>
      <div id="test-questions-html-css"></div>
      <div id="test-result-html-css" style="display: none"></div>
      <button id="submit-test-html-css" class="main-page-btn">
        Завершити тест
      </button>
      <button
        id="retry-test-html-css"
        class="main-page-btn"
        style="display: none"
      >
        Пройти тест знову
      </button>
    </div>
  </div>
```

Додаток Д. Лістинг файлу server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const bcrypt = require('bcrypt');
const sqlite3 = require('sqlite3').verbose();
const jwt = require('jsonwebtoken');
const path = require('path');
const multer = require('multer');

const upload = multer({ dest: 'uploads/' });

const app = express();
const port = process.env.PORT || 3000;

app.use(bodyParser.json());
app.use(cors());

app.use(express.static(path.join(__dirname, 'public')));
app.use('/uploads', express.static(path.join(__dirname, 'uploads')));

const secretKey = 'secret_key';
```

```
const dbPath = path.join(__dirname, 'database.db');
const db = new sqlite3.Database(dbPath, (err) => {
  if (err) {
    console.error('Не вдалося підключитися до бази даних', err);
  } else {
    console.log('Підключено до бази даних SQLite');
  }
});

db.run(`
  CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password TEXT NOT NULL,
    aboutMe TEXT,
    profilePhoto TEXT
  )
`);

db.run(`
  CREATE TABLE IF NOT EXISTS results (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    userId INTEGER NOT NULL,
    testName TEXT NOT NULL,
    score INTEGER NOT NULL,
    timeTaken INTEGER NOT NULL,
    FOREIGN KEY(userId) REFERENCES users(id)
  )
`);

db.run(`
  CREATE TABLE IF NOT EXISTS snippets (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    userId INTEGER NOT NULL,
    title TEXT NOT NULL,
    codeText TEXT NOT NULL,
    language TEXT,
    dateCreated TEXT,
    FOREIGN KEY(userId) REFERENCES users(id)
  )
`);

db.run(`
  CREATE TABLE IF NOT EXISTS comments (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    snippetId INTEGER NOT NULL,
    userId INTEGER NOT NULL,
    parentId INTEGER,
    commentText TEXT NOT NULL,
    dateCreated TEXT NOT NULL,
    FOREIGN KEY(snippetId) REFERENCES snippets(id),
    FOREIGN KEY(userId) REFERENCES users(id)
  )
`);

db.all("PRAGMA table_info(snippets)", [], (err, columns) => {
  if (err) {
    console.error("Помилка PRAGMA:", err);
  } else {
    const hasLikes = columns.some(col => col.name === 'likes');
    if (!hasLikes) {
      db.run("ALTER TABLE snippets ADD COLUMN likes INTEGER
DEFAULT 0", (alterErr) => {
        if (alterErr) {
          console.log("Помилка при створенні колонки 'likes':", alterErr.message);
        } else {
          console.log("Колонка 'likes' успішно додана!");
```

```
        }
      });
    } else {
      console.log("Колонка 'likes' вже існує. Пропускаємо ALTER TABLE.");
    }
  }
});

function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.status(401).json({ error: 'Токен не надано' });
  jwt.verify(token, secretKey, (err, user) => {
    if (err) return res.status(403).json({ error: 'Неправильний токен' });
    req.user = user;
    next();
  });
}

app.post('/register', (req, res) => {
  const { name, email, password } = req.body;
  if (!name || !email || !password) {
    return res.json({ error: 'Будь ласка, заповніть всі поля' });
```

```
  }
  const hashedPassword = bcrypt.hashSync(password, 10);


  const stmt = db.prepare('INSERT INTO users (name, email, password) VALUES
(?, ?, ?)');
  stmt.run(name, email, hashedPassword, function(err) {
    if (err) {
      if (err.message.includes('UNIQUE constraint failed')) {
        res.json({ error: 'Користувач з таким email вже існує.' });
      } else {
        res.json({ error: err.message });
      }
    } else {
      const newUserId = this.lastID;
      const token = jwt.sign({ userId: newUserId }, secretKey);
      res.json({
        userId: newUserId,
        name,
        email,
        token
      });
    }
  });
```

```
  stmt.finalize();
});

app.post('/login', (req, res) => {
  const { email, password } = req.body;
  if (!email || !password) {
    return res.json({ error: 'Будь ласка, заповніть всі поля' });
  }
  db.get('SELECT * FROM users WHERE email = ?', [email], (err, user) => {
    if (err) return res.json({ error: err.message });
    if (!user) {
      return res.json({ error: 'Користувача не знайдено.' });
    }
    const isValidPassword = bcrypt.compareSync(password, user.password);
    if (!isValidPassword) {
      return res.json({ error: 'Невірний пароль.' });
    }
    const token = jwt.sign({ userId: user.id }, secretKey);
    res.json({
      userId: user.id,
      name: user.name,
      email: user.email,
      token
```

```
    });
  });
});

app.post('/submit-result', authenticateToken, (req, res) => {
  const { userId, testName, score, timeTaken } = req.body;
  if (!userId || !testName || score === undefined || timeTaken === undefined) {
    return res.json({ error: 'Неправильні дані' });
  }
  db.get(
    'SELECT id, score FROM results WHERE userId = ? AND testName = ?',
    [userId, testName],
    (err, row) => {
      if (err) return res.json({ error: err.message });
      if (!row) {
        const insertStmt = db.prepare(
          'INSERT INTO results (userId, testName, score, timeTaken) VALUES (?, ?,
?, ?)'
        );
        insertStmt.run(userId, testName, score, timeTaken, function(e2) {
          if (e2) return res.json({ error: e2.message });
          res.json({ resultId: this.lastID });
        });
```

```
        insertStmt.finalize();
      } else {
        if (score > row.score) {
          const updateStmt = db.prepare(
            'UPDATE results SET score = ?, timeTaken = ? WHERE id = ?'
          );
          updateStmt.run(score, timeTaken, row.id, function(e3) {
            if (e3) return res.json({ error: e3.message });
            res.json({ resultId: row.id });
          });
          updateStmt.finalize();
        } else {
          res.json({
            message: 'Стара спроба була краща або рівна, нічого не змінено.'
          });
        }
      }
    }
  );
});

app.get('/user-results/:userId', authenticateToken, (req, res) => {
  const userId = req.params.userId;
```

```
  if (parseInt(userId) !== req.user.userId) {
    return res.status(403).json({ error: 'Доступ заборонено' });
  }
  db.all('SELECT * FROM results WHERE userId = ?', [userId], (err, rows) => {
    if (err) return res.json({ error: err.message });
    res.json(rows);
  });
});

app.get('/user-info', authenticateToken, (req, res) => {
  const userId = req.user.userId;
  db.get('SELECT name, email, aboutMe, profilePhoto FROM users WHERE id =
?', [userId], (err, row) => {
    if (err) return res.json({ error: err.message });
    if (!row) return res.json({ error: 'Користувача не знайдено.' });
    res.json({
      name: row.name,
      email: row.email,
      aboutMe: row.aboutMe || '',
      profilePhoto: row.profilePhoto || ''
    });
  });
});
```

```
app.post('/update-profile', authenticateToken, upload.single('profilePhoto'), (req, res) => {
  const userId = req.user.userId;
  const aboutMe = req.body.aboutMe || '';
  let photoPath = null;
  if (req.file) {
    photoPath = '/uploads/' + req.file.filename;
  }
  db.run(
    'UPDATE users SET aboutMe = ?, profilePhoto = ? WHERE id = ?',
    [aboutMe, photoPath, userId],
    function(err) {
      if (err) return res.json({ error: err.message });
      res.json({ success: true, profilePhoto: photoPath });
    }
  );
});

app.post('/snippets', authenticateToken, (req, res) => {
  const { title, codeText, language } = req.body;
  const userId = req.user.userId;
  if (!title || !codeText) {
```

```
    return res.json({ error: 'Потрібно поле title і codeText.' });
  }
  const dateCreated = new Date().toISOString();

  db.run(`
    INSERT INTO snippets (userId, title, codeText, language, dateCreated)
    VALUES (?, ?, ?, ?, ?)
  `,
  [userId, title, codeText, language, dateCreated],
  function(err) {
    if (err) return res.json({ error: err.message });
    res.json({ success: true, snippetId: this.lastID });
  });
});

app.get('/snippets', (req, res) => {
  const sort = req.query.sort;
  let sql = `
    SELECT s.*, u.name as userName
    FROM snippets s
    JOIN users u ON s.userId = u.id
  `;
```

```
  if (sort === 'likesDesc') {
    sql += ' ORDER BY s.likes DESC';
  } else if (sort === 'dateAsc') {
    sql += ' ORDER BY s.dateCreated ASC';
  } else if (sort === 'languageAsc') {
    sql += ' ORDER BY s.language ASC';
  } else {
    sql += ' ORDER BY s.dateCreated DESC';
  }

  db.all(sql, [], (err, rows) => {
    if (err) return res.json({ error: err.message });
    res.json(rows);
  });
});

app.put('/snippets/:id', authenticateToken, (req, res) => {
  const snippetId = req.params.id;
  const { title, codeText, language } = req.body;
  db.get('SELECT userId FROM snippets WHERE id = ?', [snippetId], (err, row)
=> {
    if (err) return res.json({ error: err.message });
    if (!row) return res.json({ error: 'Snippet не знайдено' });
```

```
    if (row.userId !== req.user.userId) {
      return res.json({ error: 'Це snippet іншого користувача.' });
    }
    db.run(`
      UPDATE snippets
      SET title = ?, codeText = ?, language = ?
      WHERE id = ?
    `,
    [title, codeText, language, snippetId],
    function(e2) {
      if (e2) return res.json({ error: e2.message });
      res.json({ success: true });
    });
  });
});

app.delete('/snippets/:id', authenticateToken, (req, res) => {
  const snippetId = req.params.id;
  db.get('SELECT userId FROM snippets WHERE id = ?', [snippetId], (err, row)
=> {
    if (err) return res.json({ error: err.message });
    if (!row) return res.json({ error: 'Snippet не знайдено' });
    if (row.userId !== req.user.userId) {
```

```
      return res.json({ error: 'Недостатньо прав для видалення.' });
    }
    db.run('DELETE FROM snippets WHERE id = ?', [snippetId], function(e2) {
      if (e2) return res.json({ error: e2.message });
      res.json({ success: true });
    });
  });
});

app.post('/snippets/:id/like', authenticateToken, (req, res) => {
  const snippetId = req.params.id;
  db.run('UPDATE snippets SET likes = likes + 1 WHERE id = ?', [snippetId],
function(err) {
    if (err) return res.json({ error: err.message });
    if (this.changes === 0) {
      return res.json({ error: 'Snippet не знайдено.' });
    }
    res.json({ success: true });
  });
});

app.post('/snippets/:snippetId/comments', authenticateToken, (req, res) => {
  const snippetId = req.params.snippetId;
```

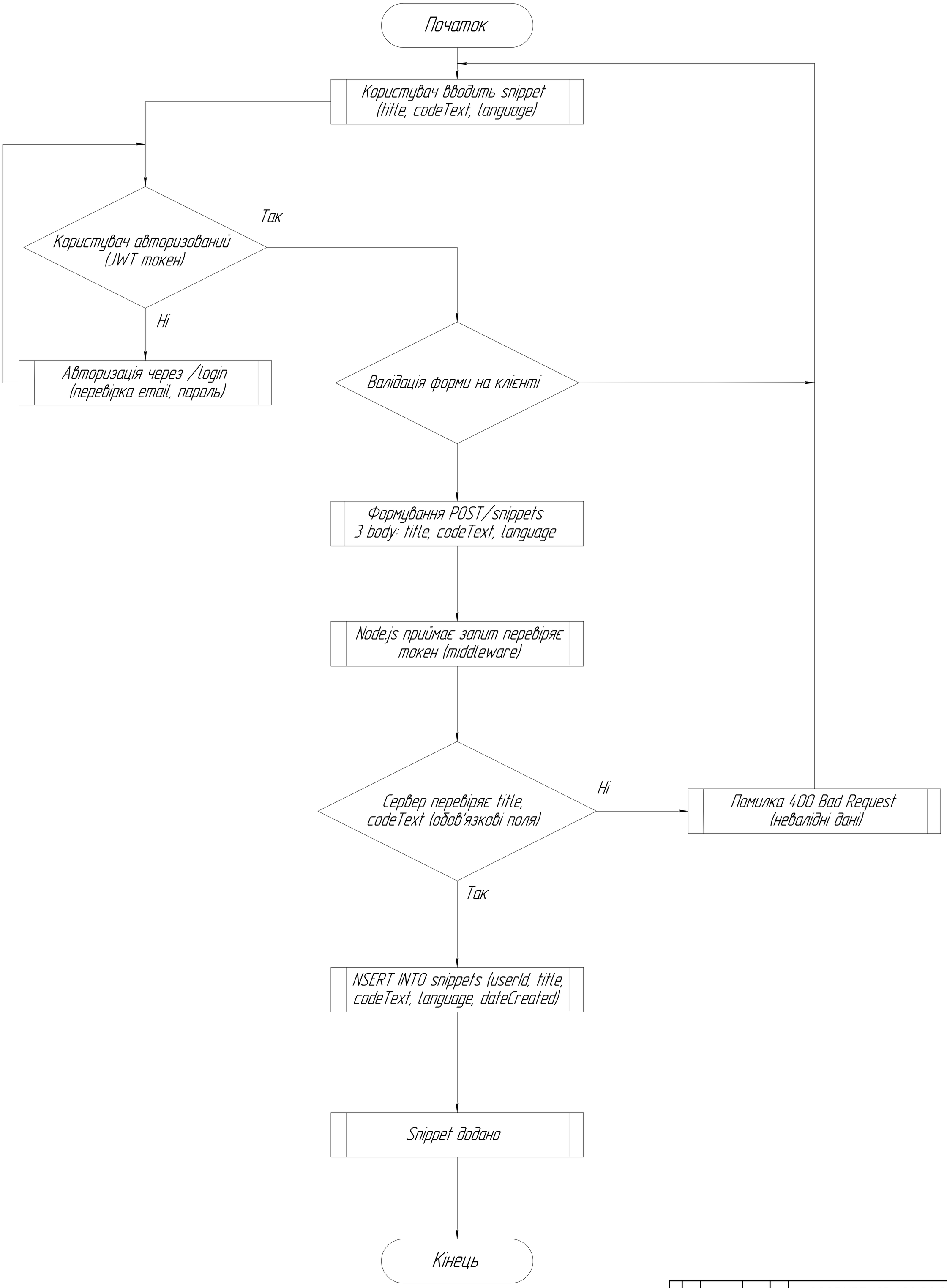
```
  const userId = req.user.userId;
  const { commentText, parentId } = req.body;

  if (!commentText) {
    return res.json({ error: 'Текст коментаря не може бути порожнім.' });
  }
  const dateCreated = new Date().toISOString();

  db.run(`
    INSERT INTO comments (snippetId, userId, parentId, commentText,
dateCreated)
    VALUES (?, ?, ?, ?, ?)
  `,
  [snippetId, userId, parentId || null, commentText, dateCreated],
  function(err) {
    if (err) return res.json({ error: err.message });
    res.json({ success: true, commentId: this.lastID });
  });
});

app.get('/snippets/:snippetId/comments', (req, res) => {
  const snippetId = req.params.snippetId;
```

```
  const sql = `
    SELECT c.*, u.name as userName
    FROM comments c
    JOIN users u ON c.userId = u.id
    WHERE c.snippetId = ?
    ORDER BY c.dateCreated ASC
  `;
  db.all(sql, [snippetId], (err, rows) => {
    if (err) return res.json({ error: err.message });
    res.json(rows);
  });
});

app.delete('/comments/:commentId', authenticateToken, (req, res) => {
  const commentId = req.params.commentId;
  const userId = req.user.userId;

  db.get('SELECT userId FROM comments WHERE id = ?', [commentId], (err,
row) => {
    if (err) return res.json({ error: err.message });
    if (!row) return res.json({ error: 'Коментар не знайдено.' });

    if (row.userId !== userId) {
```

```
      return res.json({ error: 'Недостатньо прав для видалення.' });
    }

    db.run('DELETE FROM comments WHERE id = ?', [commentId], function(e2)
{
      if (e2) return res.json({ error: e2.message });
      res.json({ success: true });
    });
  });
});

app.put('/comments/:commentId', authenticateToken, (req, res) => {
  const commentId = req.params.commentId;
  const userId = req.user.userId;
  const { newText } = req.body;

  if (!newText) {
    return res.json({ error: 'Текст не може бути порожнім при редагуванні.' });
  }

  db.get('SELECT userId FROM comments WHERE id = ?', [commentId], (err,
row) => {
    if (err) return res.json({ error: err.message });
```

```
    if (!row) return res.json({ error: 'Коментар не знайдено.' });
    if (row.userId !== userId) {
      return res.json({ error: 'Недостатньо прав для редагування.' });
    }

    db.run(`
      UPDATE comments
      SET commentText = ?
      WHERE id = ?
    `,
    [newText, commentId],
    function(e2) {
      if (e2) return res.json({ error: e2.message });
      res.json({ success: true });
    });
  });
});

app.listen(port, () => {
  console.log(`Сервер запущений на порті ${port}`);
});
```

Блок-схема роботи Snippet



Зм.	Арк.	№ докум.	Підпис	Дата	2025.КРБ.123.602.02.00.01 БС	
Розроб.		Білик В.С.			Розробка навчально-демонстраційного сайту «SkillUpCoding» Блок схема	Літ. / Маса / Масштаб
Перев.		Лісовий В.М				Арк. / Аркушів 1
Т.контр.						ВСП ТФК ТНТУ КІб-602 м. Тернопіль
Н.контр.		Приймак В.А				
Затв.						

Копіював Формат А1

Первинне застосування

Довідковий №

Підп. і дата

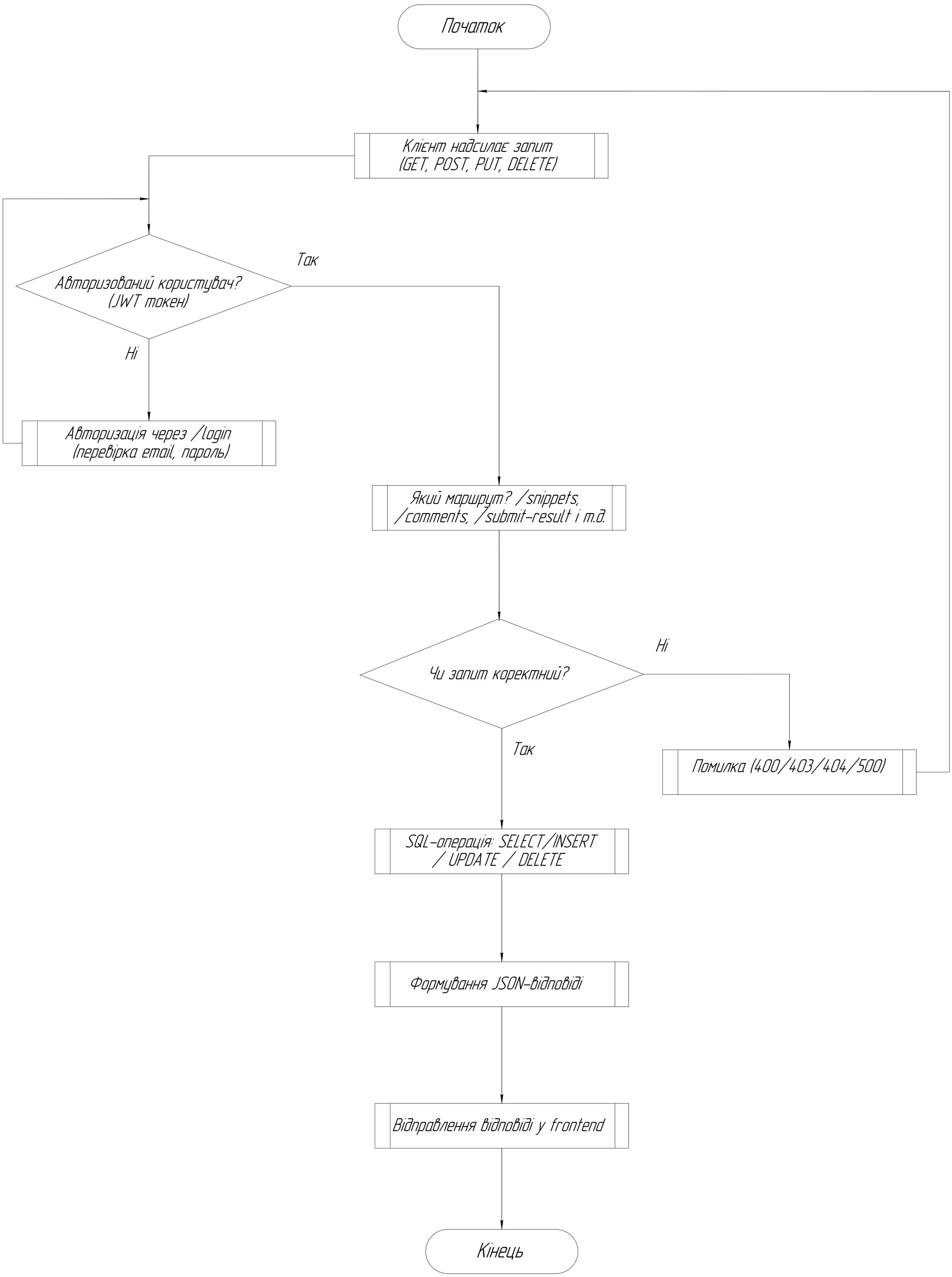
Інв. № дубл.

Взам. інв. №

Підп. і дата

Інв. № орг.

Блок-схема роботи бази даних



Зм.	Арк.	№ докум.	Підпис	Дата	2025.КРБ.123.602.02.00.02 БС
Розроб.		Білик В.С.			Розробка навчально-демонстраційного сайту «SkillUpCoding» Блок схема
Перев.		Лісовий В.М			Літ. / Маса / Масштаб
Т.контр.					Арк. / Аркушів 1
Н.контр.		Приймак В.А			ВСП ТФК ТНТУ Кіб-602 м. Тернопіль
Затв.					

Копіював

Формат А1

Первинне застосування

Довідковий №

Інв. № дубл.

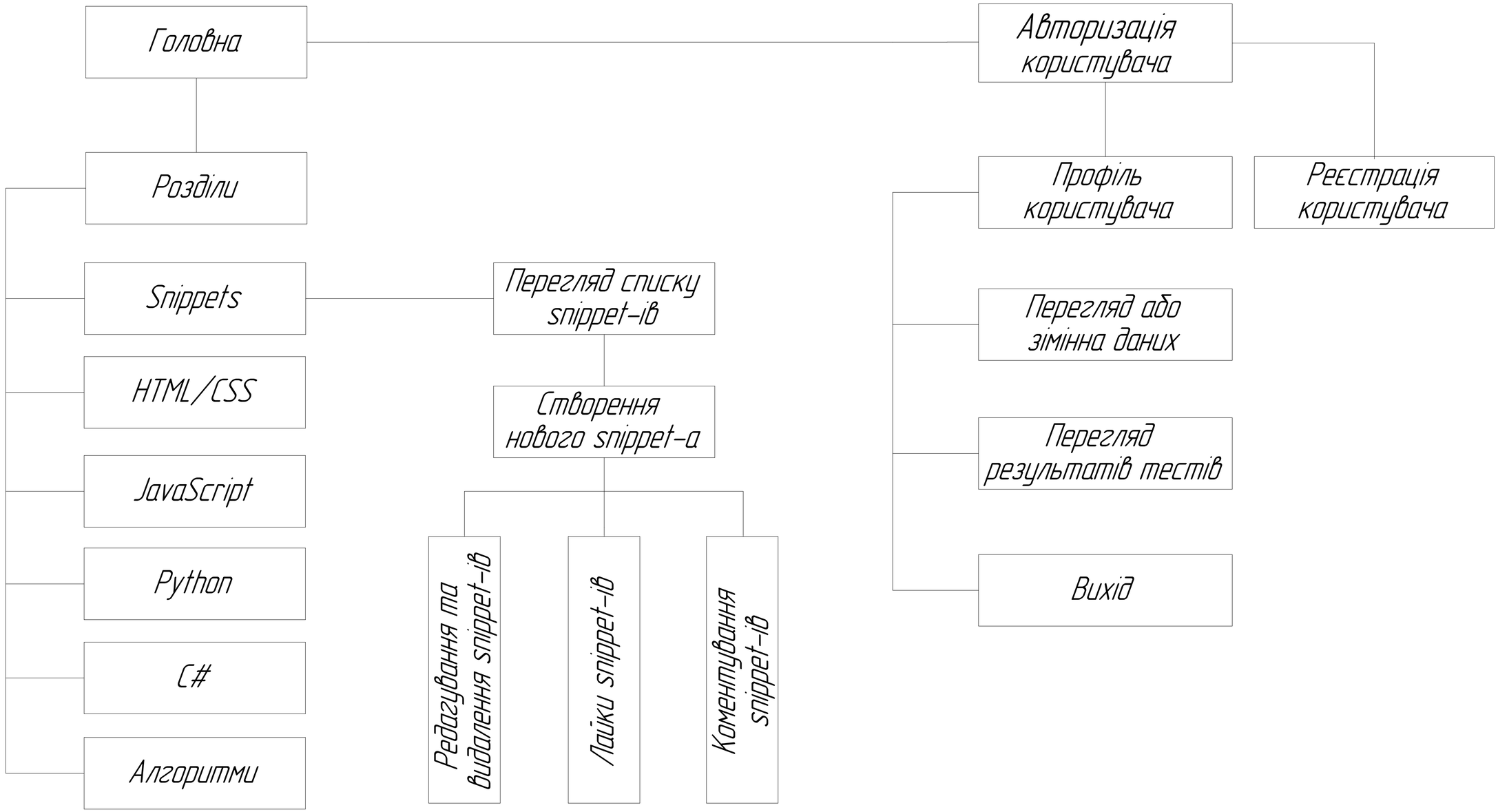
Підп. і дата

Зам. інв. №

Підп. і дата

Інв. № орг.

Структурна схема інтерфейсу



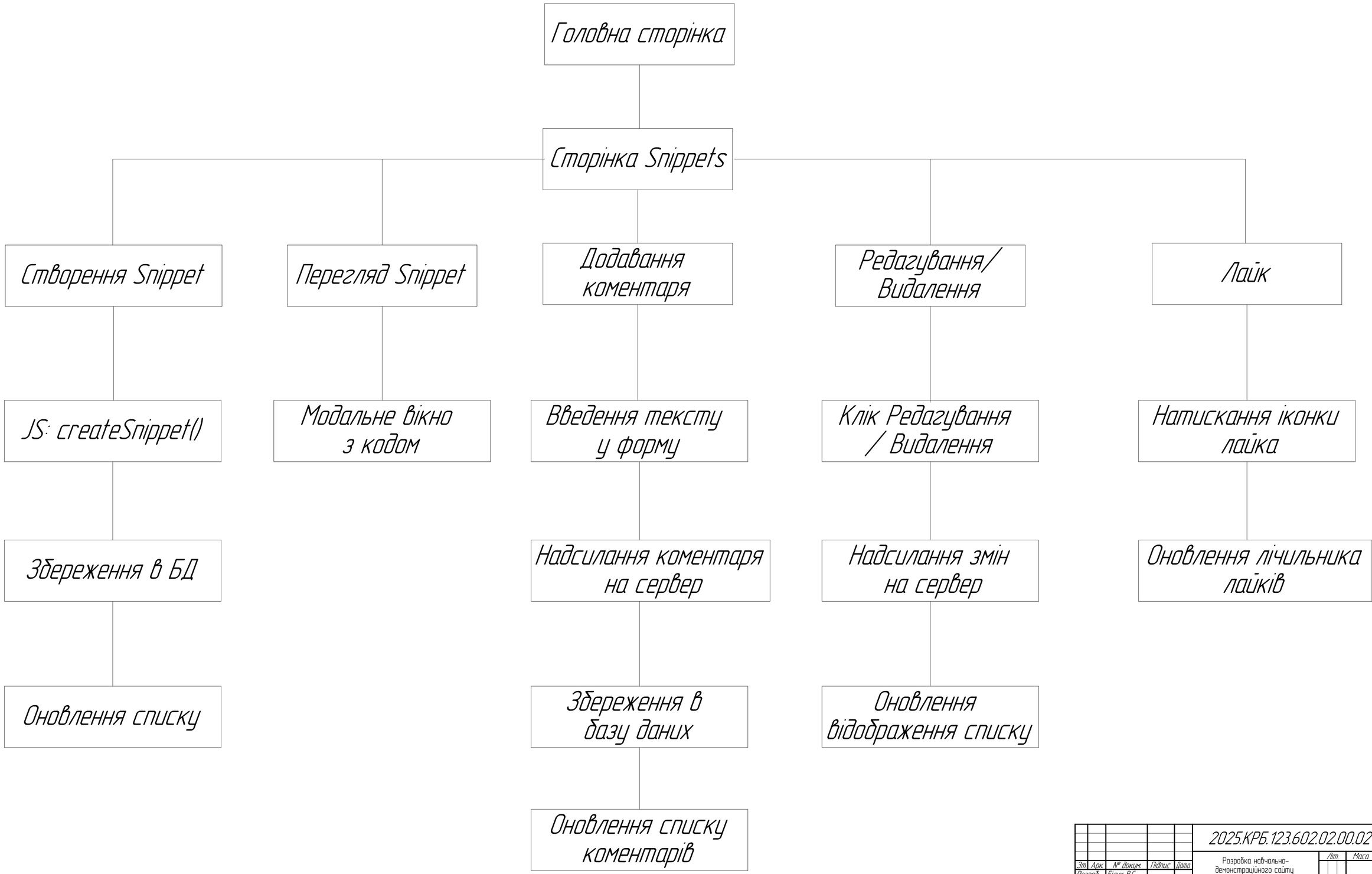
Зм.	Арк.	№ докум.	Підпис	Дата	2025.КРБ.123.602.02.00.01 СС	
Розроб.		Білик В.С.			Розробка навчально-демонстраційного сайту «SkillUpCoding» Схема структурна	Літ. / Маса / Масштаб
Перев.		Лісовий В.М				Арк. / Аркушів 1
Т.контр.						ВСП ТФК ТНТУ КІБ-602 м. Тернопіль
Н.контр.		Прийомак В.А				
Затв.						

Копіював

Формат А1

Інв. № орг. | Підп. і дата | Взам. інв. № | Інв. № дубл. | Підп. і дата | Первинне застосування

Структурна схема інтерфейсу роботи зі Snippet-ами



Зм.	Арк.	№ докум.	Підпис	Дата	2025.КРБ.123.602.02.00.02 СС		
Розроб.		Білик В.С.			Розробка навчально-демонстраційного сайту «SkillUpCoding» Схема структурна	Літ.	Маса	Масштаб
Перев.		Лісовий В.М						
Т.контр.						Арк.	Аркушів 1	
Н.контр.		Приймак В.А				ВСП ТФК ТНТУ КІБ-602 м. Тернопіль		
Затв.								

Копіював

Формат А1

Інв. № орг. | Підп. і дата | Взам. інв. № | Інв. № дубл. | Підп. і дата | Первинне застосування

Лістинг функції renderSnippets(snippets)

```
function renderSnippets(snippets) {
  const container = document.getElementById("snippetsList");
  if (!container) return;
  container.innerHTML = "";

  const currentUserId = localStorage.getItem("userId") || "";

  snippets.forEach((snippet) => {
    const div = document.createElement("div");
    div.classList.add("snippet-card");

    const langClass = snippet.language
      ? `language-${snippet.language.toLowerCase()}`
      : "language-plaintext";

    div.innerHTML = `
      <h4>${snippet.title} <small>(${snippet.language || "NoLang"})</small></h4>
      <p>Автор: ${snippet.userName}</p>
      <pre><code class="${langClass}">${snippet.codeText}</code></pre>
    `;

    const likeContainer = document.createElement("div");
    likeContainer.classList.add("snippet-like-container");
    likeContainer.onclick = () => likeSnippet(snippet.id);

    const likeCount = document.createElement("span");
    likeCount.classList.add("snippet-like-count");
    likeCount.textContent = snippet.likes || 0;

    const likeIcon = document.createElement("img");
    likeIcon.classList.add("snippet-like-icon");
    likeIcon.src = "img/like.png";
```

```
function renderSnippets(snippets) {
  const container = document.getElementById("snippetsList");
  if (!co
 likeContainer.appendChild(likeCount);
    likeContainer.appendChild(likeIcon);
    div.appendChild(likeContainer);
  if (parseInt(snippet.userId) === parseInt(currentUserId)) {
      const editBtn = document.createElement("button");
      editBtn.textContent = "Edit";
      editBtn.classList.add("nav-btn");
      editBtn.onclick = () => editSnippet(snippet);
      const delBtn = document.createElement("button");
      delBtn.textContent = "Delete";
      delBtn.classList.add("nav-btn");
      delBtn.onclick = () => deleteSnippet(snippet.id);
      div.appendChild(editBtn);
      div.appendChild(delBtn);
    }
    const viewCommentBtn = document.createElement("button");
    viewCommentBtn.textContent = "Переглянути / Коментувати";
    viewCommentBtn.classList.add("nav-btn");
    viewCommentBtn.onclick = () => openSnippetModal(snippet);
    div.appendChild(viewCommentBtn);
    container.appendChild(div);
  });
  hljs.highlightAll();
}
```

2025.КРБ.123.602.02.00.00 ТП					
Зм. Арк.	№ докум.	Підпис	Дата	Розробка навчально-демонстраційного сайту «SkillUpCoding» Текст програми	Літ. / Маса / Масштаб
Розроб.	Білик В.С.				
Перев.	Лісовий В.М				
Т.контр.					Арк. / Аркушів 1
Н.контр.	Приймак В.А				ВСП ТФК ТНТУ КІб-602 м. Тернопіль
Затв.					

Первинне застосування | Довідковий № | Підп. и дата | Інв. № дубл. | Зам. инв. № | Підп. и дата | Інв. № ор.

Копіював   Формат А1

Таблиця техніко-економічних показників

№ п/п	Назва показника	Значення	№ п/п	Назва показника	Од. виміру	Значення
1	Мова розмітки	HTML, CSS, JavaScript	6	Трудомісткість розробки	год.	126
2	Backend-технології	Python, Django	7	Собівартість	грн.	70749,28
3	Менеджер пакетів	pip, npm	8	Плановий прибуток	грн.	39618,80
4	Система керування версіями	Git	9	Економічна ефективність	–	0,56
5	Загальний об'єм проекту без модулів	~200 KB	10	Термін окупності	рік	1,79