«НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

Фаховий молодший бакалавр
(освітньо-професійний ступінь)

на тему: Розробка онлайн-платформи для вивчення мови програмування TypeScript з використанням Nest.js

проєкт захищений в ЕК « __ » _____2025 р.

Виконав студент IV курсу, групи ***ОК-***
ОПП *«Обслуговування комп'ютерних систем та мереж»*
Спеціальності *123 Комп'ютерна інженерія*
Лендєл Василь Сергійович
(прізвище, ім'я по батькові)

Керівник		*Селемонавічус А.А*
	(підпис)	(ім'я прізвище)
Нормоконтролер		*Любомира Кужій*
	(підпис)	(ім'я прізвище)
Рецензент		
	(підпис)	(ім'я прізвище)
Голова ЕК		*Олег Гіщак*
	(підпис)	(ім'я прізвище)
Члени ЕК		*Любомира Кужій*
	(підпис)	(ім'я прізвище)
		Андрій Селемонавічус
	(підпис)	(ім'я прізвище)

Дипломний проєкт захищений в ЕК « __ » _____2025 р.

з оцінкою « _____ »

Львів 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА» ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА» НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Циклова комісія	***Комп'ютерних систем і мереж***
Освітньо-професійний ступінь	***Фаховий молодший бакалавр***
Освітньо-професійна програма	***Обслуговування комп'ютерних систем та мереж***
Спеціальність	***123 Комп'ютерна інженерія***

ЗАТВЕРДЖУЮ
Завідувач відділення
«Комп'ютерних систем і мереж»
_____ Володимир СТАХІВ
«____» _____ 2025 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Лендєлу Васию Сергійовичу
(прізвище, ім'я та по батькові)

1. Тема проєкту *Розробка онлайн-платформи для вивчення мови програмування TypeScript з використанням Nest.js*

керівник проєкту *Селемонавічус Андрій Альбідасович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом директора від «*20*» *березня 2025* року № *20 - ст*

2. Строк подання студентом проєкту «*10*» *червня* 2025 року

3. Вихідні дані до проєкту

3.1 В якості сови програмування використати мову TypeScript:

3.2 В якості бази даних використати PostgreSQL

3.3 Розобити веб-платформу з використанням фреймворку Nest.j

4. Зміст розрахунково-пояснювальної записки

4.1 Огляд онлайн платформа для вивчення програмування

4.2 Розробка онлайн платформа для вивчення мови програмування TypeScript

4.3 Проєктування веб-сайту

4.4 Техніко-економіче обҐунтування

4.5 Охорона праці та безпека життєдіяльності

5. Перелік графічного матеріалу

5.1.	*Лист 1 Приклад онлайн платформи*
5.2.	*Лист 2 Архітектура програми*
5.3.	*Лист 3 Схема бази даних*
5.4.	*Лист 4. Кошторис витрат на розробку проєктного рішення*

6 Консультанти розділів проєкту

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		Завдання видав	Завдання отримав
Техніко-економічне обґрунтовування	*Тетяна Підкуймуха*		
Охорона праці та безпека життєдіяльності	*Роман Томків*		

7. Дата видачі завдання « *01»квітня 2025* року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Термін виконання	Примітка
1	*Огляд веб-платформ для аналізу*		
2	*Вибір технологій*		
3	*Проєктування архітектури*		
4	*Написання програми*		
5	*Розрахунок техніко-економічного аспекту*		
6			

Студент _____ Василь Лендєл
(підпис) (ім'я, прізвище)

Керівник проєкту _____ Андрій Селемонавічус
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка дипломного проєкту: 46 сторінки, 3 рисунка, 3 таблиці, 12 посилань, 4 демонстраційних креслення.

Об’єкт проектування: діяльність студента в освітньому процесі.

Мета проєкту – створення веб-платформи для вивчення програмування з інтерактивними завданнями, авторизацією, особистими кабінетами та системою перевірки рішень, що працює на основі серверного фреймворку Nest.js.

Метод проектування – еалізація багаторівневої клієнт-серверної архітектури з використанням Nest.js (бекенд), TypeScript та Prisma ORM для взаємодії з базою даних PostgreSQL. Клієнтська частина реалізована за допомогою React.js. API побудоване за принципами REST. Додаток розгорнутий на хмарній платформі Render з інтегрованою CI/CD-підсистемою на базі GitHub Actions.

Галузь використання – системи онлайн-освіти, зокрема сервіси дистанційного навчання з акцентом на інформатику та програмування.

NEST.JS, TYPESCRIPT, PRISMA ORM, POSTGRESQL, REACT, ОНЛАЙН-ПЛАТФОРМА, REST API, CI/CD, RENDER, GITHUB ACTIONS.

ЗМІСТ

Вступ

Актуальність теми дипломної роботи зумовлена стрімким розвитком інформаційних технологій та зростаючим попитом на фахівців у галузі програмування. У зв’язку з цим зростає інтерес до онлайн-освіти, яка дозволяє опановувати нові знання незалежно від місця перебування. Однак не всі існуючі освітні платформи задовольняють вимоги щодо якості навчання, гнучкості інтерфейсу та технічної реалізації. Тому актуальною є розробка сучасної, зручної та масштабованої онлайн-платформи для вивчення програмування з використанням сучасного технологічного стеку, зокрема фреймворку Nest.js, який забезпечує модульність, продуктивність і підтримку TypeScript.

Метою дослідження є розробка веб-платформи, яка дозволяє користувачам вивчати основи програмування в інтерактивній формі, з можливістю проходження тестів, виконання практичних завдань та отримання зворотного зв’язку.

Для досягнення мети поставлені такі завдання:

- провести аналіз існуючих онлайн-платформ для вивчення програмування.
- Визначити функціональні та нефункціональні вимоги до майбутньої системи.
- Розробити архітектуру веб-додатку з використанням Nest.js.
- Реалізувати базовий функціонал платформи, включаючи авторизацію, доступ до навчального контенту, виконання завдань і перевірку результатів.
- Провести тестування системи та оцінити її ефективність і зручність у використанні.

Об’єктом дослідження є процес онлайн-навчання в галузі програмування.

Предметом дослідження є методи та засоби розробки веб-платформ для навчання програмуванню з використанням Nest.js.

У дипломній роботі застосовано такі методи дослідження: аналіз літературних джерел і готових рішень, порівняльний аналіз функціоналу, методи проєктування інформаційних систем, розробка програмного забезпечення, тестування, моделювання процесів взаємодії користувача з системою.

Структура роботи включає вступ, три розділи, висновки, список використаних джерел та додатки. У першому розділі розглянуто теоретичні основи розробки онлайн-платформ, проведено аналіз аналогічних систем. У другому — описано проєктування платформи, визначено вимоги, побудовано архітектуру та структуру бази даних. У третьому розділі подано реалізацію та тестування системи, проаналізовано її функціональність та перспективи подальшого розвитку.

Фактологічну основу дослідження становлять наукові та технічні джерела, документація до Nest.js, результати аналізу функціоналу існуючих освітніх онлайн-платформ (Codecademy, FreeCodeCamp, SoloLearn, Codewars, LeetCode), а також практичні матеріали, отримані в процесі розробки власного програмного забезпечення.

1 ОГЛЯД ОНЛАЙН ПЛАТФОРМА ДЛЯ ВИВЧЕННЯ ПРОГРАМУВАННЯ

1.1 Сучасний стан розвитку онлайн-освіти

.Інформаційна епоха, в якій ми живемо, суттєво змінила підходи до навчання Освіта перестала бути прив’язаною до класичних навчальних аудиторій. Завдяки глобальному розвитку цифрових технологій, освіта стала доступною в будь-якому куточку світу через інтернет. Онлайн-освіта дозволяє проходити курси з провідних університетів, опановувати затребувані професії, зокрема у сфері IT, без відриву від роботи чи зміни місця проживання. Одним із найдинамічніших напрямів онлайн-освіти є навчання програмуванню. Попит на кваліфікованих програмістів -постійно зростає, а з ним — і інтерес до навчання в цій сфері. Серед переваг онлайн освіти: гнучкий графік, персоналізований підхід, миттєвий доступ до великої кількості навчальних ресурсів, інтерактивність, можливість самоперевірки та -контролю власного прогресу. Популяризації онлайн-навчання посприяв і COVID коли освітні установи були змушені перейти на дистанційний формат. Цей ,19 перехід стимулював подальший розвиток EdTech-сектору та змусив багатьох ,переосмислити важливість цифрових навичок. Багато компаній, таких як Google ,Microsoft, Amazon, створили власні освітні платформи для підготовки IT-фахівців що також посилило конкуренцію в цій галузі.У результаті з’явилися сотні ,навчальних онлайн-ресурсів, які пропонують вивчення різних мов програмування фреймворків, алгоритмів, структур даних. Ці курси охоплюють широкий спектр користувачів — від початківців до професіоналів, які хочуть підвищити кваліфікацію або змінити спеціалізацію. Серед найвідоміших міжнародних платформ — Coursera, Udemy, edX, Codecademy, FreeCodeCamp, LeetCode, а також платформи, орієнтовані виключно на практику, як-от Codewars. Водночас, не всі сервіси здатні поєднувати якісний теоретичний контент з ефективною практичною реалізацією, що створює попит на нові рішення у цій сфері. Багато курсів пропонують лише відеолекції без інтерактивних вправ або навпаки — практичні

завдання без супроводу теоретичних пояснень. Це створює розрив між засвоєнням знань і реальним застосуванням навичок у задачах. Ще однією тенденцією останніх років є розвиток мобільного навчання (mobile learning) — користувачі прагнуть мати доступ до навчального контенту через смартфони, що диктує нові вимоги до дизайну платформ і адаптивності. Також набувають популярності елементи гейміфікації (бали, досягнення, рейтинги), які покращують мотивацію та залучення ,студентів. В Україні також зростає кількість освітніх онлайн-проєктів. Наприклад ,«Prometheus», «Coursera для університетів» у межах держпрограми, «Моя освіта» ITEA» — приклади національних ініціатив, які дозволяють безкоштовно або на» -комерційній основі вивчати програмування, цифрову грамотність та інші IT напрями. Таким чином, сучасна онлайн-освіта є потужним інструментом розвитку .особистості, а вивчення програмування — однією з її найзатребуваніших галузей ,Проте ефективність навчання напряму залежить від якості платформи, її зручності .технічної реалізації та здатності поєднувати теорію з практикою

:Основні вимоги до ефективної освітньої платформи

Ефективна онлайн-платформа для навчання програмуванню має відповідати певним критеріям, які забезпечують якість освітнього процесу, зручність користування та мотивацію до навчання. Ці вимоги можна розділити на функціональні, технічні та методичні.

Функціональні вимоги:

1. Інтерактивність: можливість виконувати завдання в режимі реального часу, отримувати миттєвий фідбек, переглядати приклади рішень;

2. Особистий кабінет: відображення прогресу, перегляд виконаних завдань, персональні цілі;

3. Панель адміністратора: для керування користувачами, курсами, статистикою навчання.

Технічні вимоги:

1. Надійність та масштабованість: система повинна витримувати навантаження великої кількості користувачів;

2. Безпека: захист персональних даних, безпечне зберігання налаштувань користувачів;

3. Адаптивний інтерфейс: підтримка різних розмірів екранів, включно з мобільними пристроями.

Методичні вимоги:

1. Структурованість курсу: поділ на модулі з логічним переходом від простого до складного;

2. Актуальність контенту: використання сучасних прикладів, завдань, інструментів;

3. Зворотний зв’язок: коментарі до неправильних відповідей, пояснення рішень.

Для підтримки мотивації користувача важливо, щоб навчання було не лише інформативним, а й цікавим, з елементами взаємодії, співпраці або змагання. Це особливо актуально в умовах самостійного навчання, коли відсутній фізичний контакт з викладачем або групою.

1.2Огляд технологій для створення онлайн-платформ

Створення сучасної онлайн-платформи для навчання вимагає використання гнучких і потужних технологій як на стороні клієнта, так і сервера. Одним із найпоширеніших підходів є архітектура «клієнт-сервер», де frontend (FE) відповідає за інтерфейс користувача, а backend (BE) — за бізнес-логіку, обробку даних, автентифікацію та збереження інформації.

Frontend

Для реалізації інтерфейсу користувача сьогодні найчастіше використовуються сучасні JavaScript-фреймворки:

1. React — один з найпопулярніших інструментів для розробки односторінкових застосунків (SPA). Завдяки компонентній архітектурі React

дозволяє розділяти інтерфейс на незалежні блоки з власною логікою, що спрощує повторне використання коду. Широко підтримується спільнотою, має величезну кількість бібліотек та плагінів.

3. Vue.js — легкий у вивченні, швидкий фреймворк, який ідеально підходить для невеликих або середніх проєктів. Забезпечує простоту інтеграції в існуючі HTML-сторінки, має двосторонню прив’язку даних і підтримує модульну структуру.

Angular — повноцінний фреймворк від Google з потужною екосистемою та підтримкою TypeScript. Підходить для масштабних застосунків, де важливо мати сувору архітектуру, типізацію та шаблонізацію.

Усі ці технології базуються на JavaScript або його надбудовах, забезпечуючи зручність створення динамічного та інтерактивного інтерфейсу. У контексті онлайн-платформи для вивчення програмування важливою перевагою є можливість швидкої реакції інтерфейсу, адаптивного дизайну, а також реалізації інтерактивних компонентів (редактори коду, перевірка відповідей, веб-сокети тощо).

Backend

На серверній частині (backend) можуть застосовуватись різні мови та фреймворки:

- Node.js — середовище виконання JavaScript на сервері, яке дозволяє створювати високопродуктивні, асинхронні веб-застосунки. Завдяки використанню єдиної мови (JavaScript) на FE та BE забезпечується швидка розробка та легке навчання для новачків. Node.js ідеально підходить для реального часу (наприклад, оновлення результатів чи чатів у режимі live).
- Django (Python) — високорівневий веб-фреймворк, що дозволяє створювати складні системи за короткий час завдяки принципу «DRY» (Don’t Repeat Yourself). Django має потужну ORM, чудову документацію та активну спільноту. Python є мовою, що легко читається, тому особливо підходить для освітніх проєктів.

- Laravel (PHP) — один з найпопулярніших PHP-фреймворків, який пропонує елегантну синтаксичну структуру та широкий набір інструментів для розробки веб-додатків. Добре підходить для класичних сайтів, панелей адміністратора, систем управління контентом (CMS).
- Spring Boot (Java) — промисловий стандарт для великих, надійних веб-систем. Дає змогу реалізувати складну бізнес-логіку з високим рівнем безпеки та масштабованості. Java традиційно вважається мовою корпоративного рівня, і Spring Boot активно використовується в банківських та освітніх системах.

Обґрунтування вибору технології Nest.js

У межах цього дипломного проєкту обрано Nest.js — прогресивний бекенд-фреймворк, що базується на TypeScript і використовує архітектурні принципи Angular. Nest.js поєднує переваги Node.js із чіткою структурою, типовою для складних застосунків.

Переваги Nest.js у контексті навчальної онлайн-платформи:

1. TypeScript — забезпечує типізацію, що важливо для масштабованих проєктів;

2. Модульна архітектура — спрощує підтримку, тестування і розширення системи;

3. Інтеграція з ORM — зокрема TypeORM чи Prisma, для роботи з базами даних;

4. REST і GraphQL API — дозволяють створювати гнучкі інтерфейси для взаємодії;

5. Системи контролерів, сервісів, декораторів — дозволяють логічно розділити відповідальність у коді;

6. Можливості авторизації та валідації — готові рішення для побудови безпечного доступу.

Таким чином, Nest.js дозволяє реалізувати надійну, масштабовану платформу з усіма необхідними функціональними модулями для навчання: реєстрація та вхід, курси, тести, перевірка коду, інтеграція з базами даних, аналітика прогресу тощо.

Висновок

Кожна з розглянутих мов програмування має свої сильні сторони:

1. JavaScript (Node.js, Nest.js) — Ідеальний вибір для швидкої розробки повного стеку. Завдяки використанню однієї мови як на фронтенді, так і на бекенді, зменшується час розробки та спрощується командна взаємодія. Nest.js, як структурований фреймворк для Node.js, дозволяє створювати масштабовані, модульні додатки з чіткою архітектурою. Також активно підтримує TypeScript, що підвищує стабільність коду.

2. Python (Django) — Відомий своєю простотою, елегантним синтаксисом та "батарейками в комплекті". Django дозволяє дуже швидко створити повнофункціональний веб-застосунок з адміністративною панеллю, системою аутентифікації, ORM та шаблонами. Підходить для проєктів, де потрібна висока швидкість створення MVP, а також де важлива обробка даних, аналітика або інтеграція з ML-модулями.

3. PHP (Laravel) — Laravel є найпопулярнішим фреймворком у PHP-екосистемі, забезпечує зручну структуру проєкту, підтримку шаблонів Blade, маршрутизацію, ORM (Eloquent), а також інструменти для аутентифікації, валідації та API. Низький поріг входження, широке поширення хостингів для PHP, велика кількість плагінів — усе це робить Laravel добрим вибором для невеликих або середніх освітніх проєктів..

4. Java (Spring Boot) — надійність, масштабованість, корпоративна підтримка. Один із найнадійніших виборів для великих, критичних до безпеки або продуктивності систем. Spring Boot дозволяє створювати корпоративні REST API, добре масштабується, підтримує мікросервісну архітектуру, інтеграцію з базами даних, брокерами повідомлень (Kafka, RabbitMQ), системами безпеки (OAuth2,

JWT). Ідеально підходить для державних, банківських, освітніх систем з високим навантаженням

У рамках цієї дипломної роботи, враховуючи вимоги до швидкої розробки, інтерактивності та навчального характеру продукту, вибір Nest.js є оптимальним.

1.3Аналіз існуючих рішень

Існує багато онлайн-платформ, які дозволяють вивчати програмування. Однак більшість із них фокусуються або лише на подачі теорії, або лише на практиці. Нижче наведено короткий аналіз відомих рішень (табл . 1.1)

Таблиця 1.1 - аналіз відомих онлайн-платформ:

Платформа	**Фокус**	**Теорія**	**Практика**	**Переваги**	**Недоліки**
FreeCodeCamp	Освіта	+	+	Безкоштовна, структурована	Простий дизайн, англомовна
Codecademy	Освіта	+	+	Інтерактивність, різні курси	Платний доступ до більшості курсів
Codewars	Практика	–	+	Гейміфікація, рейтинг	Відсутність курсів, англійська
LeetCode	Практика	–	+	Алгоритми, підготовка до співбесід	Складні задачі, мало пояснень

Таким чином, хоча такі сервіси як Codewars та LeetCode є чудовими для практики, вони не надають теоретичних пояснень і не підходять початківцям. Навпаки, FreeCodeCamp і SoloLearn більше орієнтовані на теорію, але часто мають обмежену інтерактивність або занадто прості завдання.

У дипломному проєкті планується об’єднати переваги цих платформ:

1. Послідовна подача теорії;

2. Миттєве застосування знань у практичних завданнях;

3. Вбудований редактор коду з перевіркою;

4. Мотиваційна система балів та рейтингів;

5. Підтримка української мови;

6. Гнучка архітектура з Nest.js, яка дозволяє додавати нові курси, завдання, перевірки.

Завдяки поєднанню всіх вищенаведених характеристик, платформа стане ефективним інструментом як для початківців, що тільки розпочинають шлях у програмуванні, так і для досвідчених користувачів, які хочуть вдосконалити навички, готуватись до співбесід або практикувати алгоритмічне мислення..

2 РОЗРОБКА ОНЛАЙН ПЛАТФОРМА ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ TYPESCRIPT

2.1 Редактор коду VS Code

Visual Studio Code (VS Code) — це потужне, легке та безкоштовне середовище розробки, створене компанією Microsoft. Воно є кросплатформним (працює на Windows, macOS, Linux) і підтримує більшість популярних мов програмування. Особливо добре реалізована підтримка JavaScript, TypeScript, Node.js — технологій, що лежать в основі розробки дипломного проєкту.

На відміну від важких IDE, таких як WebStorm, IntelliJ IDEA або Eclipse, VS Code вирізняється високою швидкістю запуску, низьким споживанням ресурсів і гнучкістю в налаштуванні. Його можна легко адаптувати під конкретні потреби проєкту або стилю роботи команди завдяки великому вибору плагінів та інструментів

Основні переваги використання VS Code:

1. Вбудована інтеграція з Git і GitHub

- Можна здійснювати коміти, переглядати зміни в реальному часі, створювати гілки, вирішувати конфлікти, писати повідомлення до комітів, не виходячи з редактора.
- Підтримка розширень для GitLens, Git Graph, GitHub Copilot значно спрощує командну роботу.

2. Широкий каталог розширень (extensions)

- Доступ до більш ніж 30 000 розширень: ESLint, Prettier, REST Client, Docker, Prisma, NestJS Snippets, GraphQL, Live Server тощо.
- Можна легко налаштувати підтримку CI/CD, API запитів, дебаггінгу Docker-контейнерів тощо.

3. Інтегрований термінал

- Працює як повноцінна консоль (bash, zsh, PowerShell), з якої можна запускати сервер, тестувати API, керувати пакетами (npm/yarn/pnpm), запускати міграції та CI-скрипти.

4. Потужні засоби налагодження

- Підтримка debugging для Node.js, TypeScript, Chrome, REST-запитів.
- Можливість ставити breakpoints, переглядати змінні, стеки викликів, локальні середовища — все це зручно вбудовано у вкладку Debug.

5. Інтелектуальне автозаповнення (IntelliSense)

- Завдяки інтеграції з TypeScript та JSDoc-підказками, VS Code пропонує автодоповнення, навігацію по коду, підсвічування помилок, що значно пришвидшує розробку.

6. Підтримка форматування та стандартизації коду

- Лінтери й форматери (ESLint, Prettier) автоматично перевіряють стиль, синтаксис та потенційні помилки коду під час написання, що особливо корисно в командній роботі.

7. Підтримка середовищ і робочих просторів

- Можливість створювати конфігурації для різних середовищ (розробка, тестування, продакшн), запускати скрипти з launch.json, перемикатися між робочими просторами тощо.

Особливо зручно використовувати VS Code у поєднанні з Nest.js, оскільки офіційна документація фреймворку передбачає шаблони, які легко адаптуються під структуру проєкту у цьому редакторі. Крім того, підтримка TypeScript робить роботу з типами, підказками та автозаповненням ще зручнішою, що покращує якість коду та прискорює розробку.

У межах дипломного проєкту VS Code було основним інструментом при написанні backend частини та frontend.

Завдяки підтримці сучасних стандартів та сумісності з усіма ключовими технологіями проєкту, Visual Studio Code значно полегшив процес розробки, зробивши його швидким, комфортним і більш контрольованим. Це середовище ідеально підходить як для індивідуальної, так і для командної роботи над складними вебпроєктами.

2.2 Мова програмування JavaScript (Node.js)

JavaScript — одна з найпопулярніших і найуніверсальніших мов програмування у веброзробці. Спочатку вона використовувалася виключно для створення клієнтської логіки в браузері, але з появою середовища виконання Node.js JavaScript почав широко застосовуватись і для серверної частини. Це дозволило створювати повноцінні вебзастосунки з єдиною мовою як на фронтенді, так і на бекенді, що значно зменшує складність розробки, полегшує командну роботу та пришвидшує впровадження змін.

Node.js — це високопродуктивне середовище виконання JavaScript-коду на сервері, що базується на рушії V8 від Google. Воно побудоване на асинхронній, подієво-орієнтованій моделі, що дозволяє ефективно обробляти тисячі одночасних запитів без блокування основного потоку виконання. Завдяки цим властивостям, Node.js ідеально підходить для реактивних систем, API-серверів, реального часу, чату, онлайн-редакторів, а також навчальних платформ, де важливо забезпечити швидку відповідь і масштабованість при роботі багатьох користувачів.

Переваги Node.js у контексті створення онлайн-платформи:

1. Одна мова — JavaScript використовується на всіх рівнях застосунку — від інтерфейсу користувача (React, Vue) до серверної логіки (Node.js), що спрощує розробку, навчання та підтримку проєкту.

2. Асинхронність: Node.js дозволяє ефективно обробляти численні запити без блокування потоку виконання. Завдяки event-loop і callback-архітектурі, Node.js

чудово масштабується і справляється з великою кількістю одночасних користувачів, запитів до бази даних або інших зовнішніх сервісів

3. Розвинена екосистема npm (Node Package Manager) - Існує понад мільйон бібліотек і пакетів, доступних через npm. Це дозволяє швидко додавати до застосунку функціональність, наприклад: логування (winston, pino), авторизацію (jsonwebtoken, passport), роботу з базами даних (prisma, mongoose), тестування (jest, mocha) тощо.

4. Можливість використання сучасних стандартів ECMAScript - Node.js підтримує сучасні версії JavaScript (ES6+), включаючи async/await, модулі, класи, стрілочні функції, а також має глибоку інтеграцію з TypeScript, що дозволяє писати більш надійний, типізований код

5. Для Node.js доступні численні фреймворки, що спрощують створення вебзастосунків — від мінімалістичного Express до архітектурно побудованого Nest.js, який забезпечує модульність, тестованість і масштабованість

У рамках даної дипломної роботи Node.js виступає як серверна платформа, на основі якої реалізовано всю backend-логіку освітньої платформи — зокрема, реєстрацію користувачів, авторизацію, перевірку завдань, зберігання даних у базі, логування подій тощо. У поєднанні з Nest.js було реалізовано чітко структурований, масштабований і безпечний серверний застосунок, що відповідає сучасним стандартам веброзробки.

Таким чином, використання JavaScript у зв’язці з Node.js дало змогу забезпечити ефективність, стабільність і швидкість розробки освітньої онлайн-платформи.

2.3 Фреймворк Nest.js

Nest.js — це прогресивний серверний фреймворк для Node.js, створений з урахуванням потреб розробки масштабованих і підтримуваних веб-додатків. Він

побудований на основі TypeScript і використовує архітектурні підходи, притаманні фреймворку Angular, зокрема модульність, інжекцію залежностей та використання декораторів. Завдяки цьому Nest.js забезпечує чітку структуру проєкту і спрощує командну роботу над складними застосунками.

Головною особливістю Nest.js є його модульна архітектура, що дозволяє розділити функціонал на незалежні частини — модулі. Кожен модуль містить контролери (обробники HTTP-запитів), сервіси (бізнес-логіку), DTO (data transfer objects), middleware, інтерсептори та, за потреби, підключення до бази даних. Такий підхід значно полегшує масштабування проєкту, його тестування та повторне використання коду.

Основні переваги Nest.js:

1. TypeScript за замовчуванням — забезпечує типізацію, автозаповнення, кращу підтримку IDE, що покращує якість коду.

2. Підтримка REST та GraphQL API — фреймворк дозволяє створювати як RESTful, так і GraphQL API з мінімальними зусиллями.

3. Інтеграція з ORM — можливість гнучко взаємодіяти з базами даних через Prisma, TypeORM, MikroORM. У проєкті використано Prisma для декларативного моделювання даних та автоматичного створення SQL-міграцій

4. Безпека та авторизація — завдяки middleware, guards (@UseGuards()), pipes, Nest.js дозволяє реалізувати JWT-автентифікацію, RBAC (role-based access control), валідацію даних.

5.Підтримка WebSocket, Microservices — дозволяє створювати реальногочасові сервіси або масштабувати систему за допомогою мікросервісної архітектури (через Redis, NATS, Kafka).

6. Підключення до Swagger (OpenAPI) — автоматично генерує документацію для API через @nestjs/swagger

У межах дипломного проєкту Nest.js був обраний як головний фреймворк для реалізації backend-частини навчальної платформи. Його структура дозволила створити окремі модулі для автентифікації користувачів, керування курсами, завданнями, оцінками, що відповідає вимогам до масштабованості системи. Наприклад, модулі:

- AuthModule — реалізує автентифікацію через JWT, логін/реєстрацію
- UsersModule, - управління користувачами (роль, email, ім’я, ID)
- CoursesModule, LessonsModule, AssignmentsModule — створення й редагування курсів, уроків і завдань
- SubmissionsModule — перевірка коду студентів
- BlogModule — модуль для публікацій та коментарів

Модулі реалізовано незалежно, що дозволяє в майбутньому легко додавати нові функції без змін у вже існуючому коді.

Крім того, Nest.js має хорошу документацію та активну спільноту, що значно полегшує процес розробки, вирішення типових проблем та підтримки проєкту. У поєднанні з Node.js та базою даних PostgreSQL, Nest.js забезпечує надійну архітектуру та зручне середовище для створення складних навчальних рішень.
Nest.js — це сучасний фреймворк, який поєднує простоту Node.js із силою архітектурного підходу Angular. Його впровадження в дипломному проєкті дозволило досягти високої організованості коду, зменшити технічний борг і спростити масштабування платформи. Nest.js не лише відповідає вимогам до безпечної та надійної backend-розробки, але й забезпечує швидке впровадження нових функцій у систему. У поєднанні з PostgreSQL і Prisma, це рішення створює основу для повнофункціональної освітньої системи нового покоління.

2.4 PostgreSQL

PostgreSQL — це потужна об'єктно-реляційна система управління базами даних з відкритим кодом, яка активно використовується в промислових, освітніх

та наукових проєктах по всьому світу. Вона підтримує стандарт SQL, а також надає розширені можливості для збереження, обробки та аналізу даних. PostgreSQL відома своєю стабільністю, масштабованістю та активною спільнотою розробників.

У контексті веб-розробки PostgreSQL часто обирають за:

1. Надійність — система забезпечує ACID-сумісність (атомарність, узгодженість, ізоляція, довговічність), що є критично важливим при збереженні користувацьких даних;

2. Розширюваність — підтримка власних типів даних, функцій, процедур та розширень (наприклад, PostGIS для геоданих);

3. Високу продуктивність — оптимізатори запитів, індексація, партиціювання, підтримка паралельного виконання;

4. Гнучку схему безпеки — управління доступом реалізовано на рівнях ролей, схем, таблиць і навіть окремих колонок. Підтримується audit-логування, SSL-з'єднання, а також шифрування даних у процесі й у спокої.

У рамках розробки дипломного проєкту PostgreSQL використовується як основна система для збереження даних навчальної платформи. Зокрема, в базі зберігається інформація про:

1. зареєстрованих користувачів (студентів, викладачів, адміністраторів);

2. створені курси та теми;

3 .завдання та тести;

4. результати перевірки рішень;

5.оцінки й прогрес користувачів.

Взаємодія між серверною частиною (Nest.js) та PostgreSQL реалізована за допомогою ORM-бібліотеки Prisma, яка дозволяє працювати з базою даних на рівні об’єктів, а не сирих SQL-запитів. Це значно полегшує розробку, підвищує читабельність коду, зменшує кількість потенційних помилок та дозволяє автоматизувати генерацію запитів і взаємодію з таблицями бази даних. Завдяки

вбудованим можливостям Prisma, таким як автоматичне оновлення типів, генерація клієнта та керування міграціями, розробник може швидко адаптувати модель даних до змін у бізнес-логіці та оперативно оновлювати структуру бази без ризику втрати цілісності даних.

Схема бази даних була спроєктована з урахуванням принципів нормалізації, що забезпечує уникнення надлишковості та дублювання даних. У моделі враховано чіткі логічні зв’язки між сутностями, зокрема між курсами, уроками, користувачами, завданнями та відправленими рішеннями. Крім того, у структурі передбачено підтримку авторських блогів і коментарів, що створює додаткові можливості для взаємодії користувачів із платформою. Система побудована так, щоб бути масштабованою — як у плані зростання обсягів даних, так і розширення функціональності (наприклад, через додавання нових типів навчального контенту, аналітики або адміністративних інструментів).

Таким чином, обрана архітектура на базі Nest.js + Prisma + PostgreSQL не лише відповідає сучасним вимогам до якості, гнучкості та надійності веброзробки, але й забезпечує зручність супроводу, тестування та майбутнього розширення проєкту.

.

3 ПРОЄКТУВАННЯ ВЕБ-САЙТУ

3.1 Загальна структура

Перш ніж перейти до реалізації функціональних можливостей, важливо визначити загальну структуру програмного забезпечення, яка забезпечує масштабованість, підтримку та зрозумілу організацію коду. У межах дипломного проєкту було використано фреймворк **Nest.js**, що сам по собі базується на модульній архітектурі та слідує принципам SOLID, DI (інжекція залежностей), MVC.

Проєкт умовно поділений на логічні частини:

- prisma/ — конфігурація бази даних і файли міграцій;
- src/ — вихідний код застосунку;
- modules/ — функціональні модулі системи (auth, user, course, task тощо);
- common/ — загальні утиліти, типи, константи, які використовуються по всьому проєкту;

Кожен модуль реалізовано згідно зі стандартною структурою Nest.js, що включає:

- DTOs (dtos/) — класи для валідації вхідних даних;
- Controllers — обробка HTTP-запитів (GET, POST, PUT, DELETE);
- Services — бізнес-логіка;
- Repositories — взаємодія з базою даних через Prisma.

Такий підхід забезпечує уніфікацію структури, спрощене тестування та зручність у масштабуванні, адже нові модулі можуть створюватися за аналогічною схемою без порушення архітектури.

Приклад підключення до бази даних:

src > database > prisma.service.ts > ...

```ts
import { Injectable, OnModuleDestroy, OnModuleInit } from '@nestjs/common';

import { PrismaClient } from '@prisma/client';

@Injectable()
export class PrismaService
  extends PrismaClient
  implements OnModuleInit, OnModuleDestroy
{
  public constructor() {
    super();
  }

  public async onModuleInit(): Promise<void> {
    await this.$connect();
  }

  async onModuleDestroy(): Promise<void> {
    await this.$disconnect();
  }
}
```

Приклад авторизації з використанням Jwt Token:

```
@Injectable()
export class AuthService {
  public constructor(
    private readonly userService: UserService,
    private readonly tokenService: TokenService,
    private readonly storageService: StorageService,
  ) {}

  public async signUp(dto: CreateUserDto, file?: Express.Multer.File) {
    const user = await this.userService.findByEmail(dto.email);

    if (user) {
      throw new BadRequestException(EXCEPTION.USER_ALREADY_EXISTS);
    }

    const hashedPassword = await bcrypt.hash(dto.password, 10);

    if (file) {
      const fileId = CryptoUtil.generateUUID();

      await this.storageService.uploadFile(fileId, file.buffer, file.mimetype);

      dto.imageUrl = fileId;
    }

    const newUser = await this.userService.create({
      ...dto,
      password: hashedPassword,
    });

    const token = await this.tokenService.generateToken(newUser.id);

    return {
      user: userMapper(newUser),
      token,
    };
  }

  public async signIn(dto: LoginUserDto) {
    const user = await this.userService.findByEmail(dto.email);

    if (!user) {
      throw new BadRequestException(EXCEPTION.USER_NOT_FOUND);
    }

    const isPasswordValid = await bcrypt.compare(dto.password, user.password);

    if (!isPasswordValid) {
      throw new BadRequestException(EXCEPTION.INVALID_PASSWORD);
    }

    const token = await this.tokenService.generateToken(user.id);

    return {
      user: userMapper(user),
      token,
    };
  }
}
```

Приклад репозиторія для взаємодії з БД:

```
@Injectable()
export class CodeSubmissionRepository extends BaseRepository<
  CodeSubmission | null,
  CreateCodeSubmissionDto,
  UpdateCodeSubmissionDto
> {
  public constructor(private readonly prismaService: PrismaService) {
    super();
  }

  public async create(dto: CreateCodeSubmissionDto): Promise<CodeSubmission> {
    return await this.prismaService.codeSubmission
      .create({
        data: dto,
      })
      .catch((error) => {
        throw new InternalServerErrorException(error.message);
      });
  }

  public async find(): Promise<CodeSubmission[]> {
    return await this.prismaService.codeSubmission.findMany().catch((error) => {
      throw new InternalServerErrorException(error.message);
    });
  }

  public async findById(id: string): Promise<CodeSubmission | null> {
    return await this.prismaService.codeSubmission
      .findUnique({
        where: {
          id,
        },
      })
      .catch((error) => {
        throw new InternalServerErrorException(error.message);
      });
  }

  public async update(
    id: string,
    updates: UpdateCodeSubmissionDto,
  ): Promise<CodeSubmission> {
    return await this.prismaService.codeSubmission
      .update({
        where: {
          id,
        },
        data: updates,
      })
      .catch((error) => {
        throw new InternalServerErrorException(error.message);
      });
  }

  public async delete(id: string): Promise<CodeSubmission> {
    return await this.prismaService.codeSubmission
      .delete({
        where: {
          id,
        },
      })
      .catch((error) => {
        throw new InternalServerErrorException(error.message);
      });
  }
}
```

Висновок: файлова структура побудована з дотриманням принципів розділення відповідальності, що дає змогу легко масштабувати застосунок, тестувати окремі модулі та додавати нові функції без порушення існуючої логіки.

3.2 Розробка бази даних

Для забезпечення збереження, обробки та зв’язків даних, що використовуються на освітній онлайн-платформі, було розроблено реляційну базу даних на основі PostgreSQL з використанням ORM Prisma. Завдяки чіткій структурі сутностей, реалізовано підтримку ролей користувачів, навчального контенту, завдань, перевірок, блогу та коментарів. Всі моделі описані у файлі schema.prisma.

Підключення до бази даних реалізоване в Prisma наступним чином:

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

Основні сутності бази даних:

1. User

Зберігає дані про зареєстрованих користувачів платформи. Має поля: email, password, role, createdAt, а також зв’язки з CodeSubmission (надіслані відповіді) та Comment (коментарі в блозі). Кожен користувач має унікальний id.

2. Author

Модель для авторів курсів. Автори можуть створювати один або кілька курсів (зв’язок "один-до-багатьох"). Автор також має bio, imageUrl, дату створення.

3. Course

Курс — центральна одиниця навчання. Має назву, опис, мову, рівень складності та зв’язок з автором. Кожен курс містить кілька уроків (Lesson[]).

4. Lesson

Містить інформацію про конкретні навчальні модулі всередині курсу. Має зв’язок із відповідним курсом та список завдань для перевірки знань — CodeAssignment[].

5. CodeAssignment

Модель для кодувальних завдань. Містить вступний код (starterCode), код для тестування (testCode) та опис. Має зв’язок із уроком (Lesson) і з відповідями студентів (CodeSubmission[]).

6. CodeSubmission

Відповідає за збереження надсланих студентами відповідей на завдання. Містить submittedCode, result (наприклад, «успішно» або «помилка»), submittedAt. Пов’язаний з User (студент) та CodeAssignment.

7. BlogPost та Comment

Реалізують розділ блогу. BlogPost — це публікація з контентом, зображенням та датою створення. Коментарі (Comment) прив’язуються як до автора (User), так і до поста (BlogPost), формуючи зв’язки типу "багато-до-одного".

Приклад моделі з коду (CodeSubmission):

```
model CodeSubmission {
  id              String   @id @default(uuid())
  codeAssignmentId String
  studentId       String
  submittedCode   String
  result          String
  submittedAt     DateTime @default(now())
```

```
  assignment CodeAssignment @relation(fields: [codeAssignmentId], references: [id])
  student   User          @relation(fields: [studentId], references: [id])
}
```

Структура зв’язків у базі:

1. User ←→ CodeSubmission (один-до-багатьох)
2. Course ←→ Lesson (один-до-багатьох)
3. Lesson ←→ CodeAssignment (один-до-багатьох)
4. CodeAssignment ←→ CodeSubmission (один-до-багатьох)
5. Author ←→ Course (один-до-багатьох)
6. BlogPost ←→ Comment ←→ User (двосторонні відношення)

Усі зв’язки реалізовані через Prisma @relation, а ідентифікатори — як UUID, що забезпечує глобальну унікальність і масштабованість.

Висновок

Запропонована модель бази даних є логічно структурованою та оптимізованою під функціональні потреби онлайн-платформи для навчання програмуванню. Її структура охоплює всі ключові сутності — користувачів, курси, уроки, завдання, відправлені рішення, авторів і блог-контент. Це дозволяє ефективно зберігати як теоретичний матеріал, так і практичні результати взаємодії студентів із системою, включаючи перевірку завдань та комунікацію через коментарі.

Завдяки використанню Prisma ORM реалізується зручний доступ до бази даних із чітким типізованим інтерфейсом, що значно пришвидшує розробку, мінімізує помилки при роботі з SQL-запитами та забезпечує підтримку складних зв’язків між таблицями. Вбудовані механізми генерації клієнта, автоматичного

створення міграцій та можливість використання seed-даних дають змогу швидко адаптувати структуру до змін у бізнес-логіці або масштабуванні проєкту.

Prisma також добре інтегрується з Nest.js, дозволяючи дотримуватись принципів модульності, інкапсуляції та залежностей за допомогою DI-контейнера. Це забезпечує чисту архітектуру, легкість тестування, гнучкість підтримки та високу масштабованість. У поєднанні з PostgreSQL, Prisma ORM дозволяє реалізувати високопродуктивну та стабільну систему з підтримкою ACID-транзакцій, складних фільтрів і агрегацій.

Таким чином, обрана технологія моделювання даних повністю відповідає вимогам сучасної веброзробки, є безпечною, розширюваною та готовою до продуктивного використання в реальному середовищі.

.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Ефективна розробка програмного забезпечення потребує не лише технічної реалізації, а й попереднього економічного аналізу. У цьому розділі представлено розрахунок основних витрат, пов’язаних зі створенням, тестуванням та запуском MVP-версії онлайн-платформи для вивчення програмування.

4.1 Розрахунок витрат на розробку програмного продукту

Витрати на розробку та впровадження програмних засобів (К) включають:

К = К1 + К2, (5.1)

де К1 – витрати на розробку програмного продукту, грн.; К2 – витрати на налагодження та дослідну експлуатацію програмного засобу на сервері, грн.

Витрати на розробку програмного засобу охоплюють:

1. Витрати на оплату праці розробників (З);
2. Нарахування на зарплату (НЗ);
3. Витрати на куповані вироби (ВК);
4. Накладні витрати (НВ);
5. Інші витрати (Інші).

Для розробки програмного продукту були задіяні:

1. Керівник проєкту (К);
2. Студент-дипломник (СД);
3. Консультант з охорони праці (КОП);
4. Консультант з економічної частини (КЕ).

Оплата праці за годину:

1. Керівник проєкту – 118,13 грн;
2. Консультант з економіки – 118,13 грн;
3. Консультант з охорони праці – 103,48 грн;

4. Студент-дипломник – 8,72 грн (1569,6 грн / 180 год).

Витрати на оплату праці:

- Зкп = 1 * 14 * 118,13 = 1 653,82 грн;
- Зке = 1 * 1 * 118,13 = 118,13 грн;
- Зкоп = 1 * 1 * 103,48 = 103,48 грн;
- Зсд = 1 * 180 * 8,72 = 1 569,60 грн;

Всього: 3 444,99 грн

Таблиця 5.1 – Розрахунок витрат на оплату праці

Спеціальність розробника	**Кількість**	**Години**	**Ставка (грн)**	**Сума (грн)**
Керівник проєкту	1	14	118,13	1 653,82
Консультант з економіки	1	1	118,13	118,13
Консультант з охорони праці	1	1	103,48	103,48
Студент-дипломник	1	180	8,72	1569,6
Разом	4	236		3 444,99

Нарахування на зарплату (22%): Нз = 1 875,43 * 0,22 = 412,60 грн

Витрати на куповані вироби:

1. Папір А4 Zoom — 229,00 грн

2. Друк пояснювальної записки — 210,00 грн

3. Папка для проєкту — 157,80 грн

4. Транспортно-заготівельні витрати (10%) — 59,68 грн

- Разом: 656,48 грн

Накладні витрати (30%):

Нв = 3 615,49 * 0,30 = 1 084,65 грн

Інші витрати (10%):

Інші = (3615,49 + 795,41 + 1084,65 + 656,48) * 10 / 90 = 683,56 грн

Загальні витрати на розробку К1:

К1 = 3 615,49 + 795,41 + 656,48 + 1 084,65 + 683,56 = 6 835,59 грн'

4.2 Розрахунок витрат на налагодження та дослідну експлуатацію на сервері

Вартість віддаленого серверу, що використовується для тестування та розгортання системи:

1. Хмарний сервер (Render VPS) — 1 міс.: 400 грн

2. База даних PostgreSQL (Render) — 250 грн

3. CI/CD через GitHub Actions (умовно): 150 грн

4. Інші витрати на підтримку (Інтернет, електроенергія тощо): 275,80 грн

- Разом К2: 1 075,80 грн

Загальні витрати на реалізацію:

К = К1 + К2 = 6 835,59 + 1 075,80 = 7 911,39 грн

Таблиця 5.2 – Кошторис витрат на реалізацію програмного продукту

Найменування елементів витрат	**Сума (грн)**
Витрати на оплату	3

праці	615,49
Нарахування на зарплату	795,41
Витрати на куповані вироби	656,48
Накладні витрати	1 084,65
Інші витрати	683,56
Налагодження і запуск на сервері	1 075,80
Разом	7 911,39

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

Забезпечення безпечних умов праці та дотримання вимог охорони праці є обов’язковою складовою будь-якої професійної діяльності, незалежно від її характеру чи сфери. Навіть офісна або дистанційна праця, яка на перший погляд здається нешкідливою, має свої потенційні ризики для здоров’я та життєдіяльності працівника.

Зокрема, робота програміста або фахівця, що працює з інформаційними технологіями, пов’язана з тривалим перебуванням за комп’ютером, що призводить до статичного навантаження на опорно-руховий апарат, перевтоми зорового аналізатора, нервово-емоційного перенапруження, а також впливу електромагнітного випромінювання та недостатньої фізичної активності.

Крім того, сучасне робоче місце програміста включає велику кількість електронного обладнання (комп’ютери, блоки живлення, монітори, периферійні пристрої), що створює ризики коротких замикань, перегріву, електричних ударів або пожежі у разі неправильного використання. Особливої уваги вимагає ергономіка робочого місця, що суттєво впливає на довготривалу працездатність і попередження професійних захворювань.

5.1 Електробезпека

Для забезпечення безпечної та ефективної експлуатації ЕОМ та ВДТ необхідно дотримуватись певних принципів та правил. Перш за все, регулярні щомісячні діагностичні огляди та чистка пристроїв допоможуть уникнути накопичення пилу, які можуть спричинити утворення електростатичного струму та пробою. Під час оглядів також необхідно оцінювати стан радіокомпонентів та замінювати їх на компоненти аналогічних номіналів за необхідності, щоб забезпечити найвищу продуктивність пристроїв. Крім того, монтаж електропроводів, кабелів та ЕОМ, а також їх ремонт та обслуговування має проводитись за допомогою діелектричних засобів та при повному відімкненні від

загальної мережі для усунення можливості утворення струмів короткого замикання або струмів перенавантаження.

Електропроводи та кабелі, а також ЕОМ з ВДТ і ПП мають відповідати виконанням та ступеню захисту класу зони за НПАОП 40.1-1.32-01. Для забезпечення безпеки працівників, які працюють з ЕОМ та персональними комп'ютерами, необхідно дотримуватись вимог електробезпеки, встановлених нормативними документами, зокрема: «Правила улаштування електроустановок» (ПУЕ) та «Правилах охорони праці під час експлуатації електронно-обчислюваних машин» (НПАОП 0.00-1.28-10.). Також важливо враховувати, що лінія електромережі для живлення ЕОМ з ВДТ має бути виконана як окрема групова три провідна мережа з фазового, нульового та захисного провідника. Усі провідники мають відповідати вимогам НПАОП 40.1-1.32-01. Нульовий захисний провідник має бути прокладений від стійки групового розподільного щита. ЕОМ з ВДТ має під'єднуватись до електромережі лише за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення.

У приміщенні, де одночасно експлуатується або обслуговується більше ніж п'ять персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до звичайної двопровідної електромережі, в тому числі — з використанням перехідних пристроїв. Загалом, дотримання цих принципів та правил забезпечить безпечну та ефективну роботу ЕОМ та ВДТ.

5.2 Правила техніки безпеки при роботі за комп’ютером

Під час роботи на комп'ютері можуть діяти шкідливі фактори, які можуть вплинути на здоров'я користувача. Інформаційна безпека та електробезпека забезпечуються за допомогою ряду профілактичних заходів. Один з найважливіших аспектів є саме електробезпека. Для усунення ризику ураження струмом необхідно розміщувати обладнання та кабелі відповідно до вимог безпеки. Це може бути досягнуто за допомогою захисного заземлення, використання безпечних розеток та електропроводки, розрахованих на потужність системи, а також ізоляції всіх проводів.

Для забезпечення ефективності та продуктивності роботи комп'ютера важливо регулярно чистити внутрішні частини від пилу, користуватися окремими вогнетривкими столами для комп'ютерів та іншого устаткування. Щоб запобігти іскрінню, необхідно рідше вставляти та виймати вилки з розеток.

Освітлення на робочому місці повинно відповідати характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення, фоном та контрастом об'єкта і фону. Потрібно забезпечити рівномірне розподілення яскравості на моніторі та навколишньому просторі, відсутність різких тіней, відблисків та стабільну освітленість під час роботи. Вибирати оптимальну спрямованість світлового потоку та необхідний склад світла. Для забезпечення здоров'я користувача можна використовувати спеціальні світильники та підсвічування, які забезпечують оптимальні умови для зорової роботи.

Правила безпеки при роботі за комп'ютером:

• Увімкніть кондиціонер у приміщенні, щоб уникнути перегрівання пристроїв та забезпечити комфортну температуру для користувача.

1. Переконайтесь у стабільному розташуванні обладнання на столі. Не забудьте, що нестабільна підставка для монітора може призвести до його падіння та пошкодження. Відкрийте монітор так, щоб було зручно спостерігати екран -

прямо (не збоку) і трохи зверху вниз, з нахилом екрана, його нижній край ближче до користувача.

2. Перевірте загальний стан обладнання, справність електропроводки, кабелів, вилок, розеток та заземлення захисного екрана. Якщо щось несправне, виправте це перед роботою.

3. Налаштуйте освітлення робочого місця. Краще мати якісне та достатнє освітлення, щоб зменшити навантаження на очі та запобігти їх перевтомленню.

4. Регулюйте та фіксуйте висоту крісла та зручний нахил спинки для користувача. Важливо, щоб користувач почував себе комфортно та міг працювати тривалий час без відчуття дискомфорту.

5. Під’єднайте необхідне обладнання до системного блоку. Кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері, щоб уникнути пошкодження пристроїв та пошкодження даних.

6. Увімкніть комп'ютерне обладнання послідовно: монітор, системний блок, принтер (якщо потрібно друкувати). Це дозволить уникнути перезавантаження системи та забезпечити її стабільну роботу.

7. Налаштуйте яскравість монітора та контрастність. Не робіть зображення надто яскравим, щоб не перевтомлювати очі. Також важливо відпочивати та робити паузи під час тривалої роботи за комп'ютером, щоб не перенапружувати очі та запобігти втомі.

5.3 Пожежна безпека

Пожежна безпека об'єкта - це важливий стан об'єкта, про який регламентом визначено імовірність виникнення та розвитку пожежі та впливу на людей її небезпечних факторів, а також забезпечується захист матеріальних цінностей. Цей стан регулюється нормативними актами, зокрема ДБН В.1.1-7-2002 "Пожежна безпека об'єктів будівництва", ДСТУ 2272:2006 "Пожежна безпека. Терміни та

визначення основних понять", НАПБ А.01.001-2004 "Правила пожежної безпеки в Україні".

Важливою складовою пожежної безпеки є належне ознайомлення всіх працівників з правилами пожежної безпеки, проходження протипожежних інструктажів та перевірки знань з питань пожежної безпеки. Зокрема, автоматична пожежна сигналізація повинна завжди бути у ввімкненому, черговому стані.

Приміщення для роботи повинно відповідати нормам, зазначеним у НАПБ А.01.001.-2004 про будівлі та приміщення для ЕОМ. Зокрема, підлога та стіни в такому приміщенні повинні входити в групу горючості Г1. Заборонено зберігати легкозаймисті та горючі речовини в будь-яких кількостях в приміщенні з ЕОМ. Важливо не залишати ЕОМ без нагляду під час роботи та вимикати їх від глобальної електромережі по закінченню роботи з ними. На робочому місці має бути встановлений легкий доступ до двох газових вогнегасників як засобів первинної пожежної безпеки. Меблі та обладнання повинні розміщуватися таким чином, щоб забезпечувався вільний евакуаційний прохід до дверей виходу з приміщення (завширшки не менше 1 м). Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати. Документи, папір та інші горючі матеріали слід зберігати на відстані не менше 1 м від електрощитів і електрокабелів; 0,5 м від електросвітильників; 0,6 м від сповіщувачів пожежної сигналізації та 0,15 м від приладів центрального водяного опалення. Засоби протипожежного захисту слід утримувати у справному стані. Відстань від найбільш віддаленого місця до вогнегасника не повинна бути більшою за 20 м.

Приміщення, у яких розміщені ПЕОМ, слід оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-1,4 чи ВВК-2 або один ВВПА-400 на три ПЕОМ, але не менше ніж один вогнегасник зазначених типів на приміщення. Крім того, рекомендується забезпечити додатковий простір для працівників, який забезпечить їм додаткові можливості для руху та роботи з ПЕОМ, а також забезпечити доступ до відповідних джерел електроживлення.З метою забезпечення повної пожежної безпеки рекомендується

проводити щорічні перевірки на відповідність пожежним нормам та стандартам.Вуглекислотний вогнегасник ВВК-1,4 наведено на рисунку 5.1

Рисунок 5.1 – Вогнегасник вуглекислотний ВВК-1,4

5.4 Виробниче приміщення та робоче місце

Згідно з вимогами ДСанПіН 3.3.2.007-98, приміщення, в яких планується робота з ВДТ, повинні відповідати проєктній документації будинку, погодженій з уповноваженими державними органами. Якщо ви плануєте працювати з відомчими документами та матеріалами на комп'ютері, то вам необхідно взяти до уваги правила розташування робочого місця, які забезпечать вам максимальний комфорт та безпеку під час роботи.

Розміщення робочих місць з ВДТ у підвальних приміщеннях та на цокольних поверхах заборонено. Якщо ваше робоче місце знаходиться в приміщенні, то зверніть увагу на доступність основних умов, необхідних для вашої продуктивної роботи. На одного працівника площа робочого місця має становити не менше ніж 6,0 м, а об'єм - не менше ніж 20,0 м.

У залежності від того, яку роботу ви виконуєте, має бути враховано чинні санітарні норми освітлення, температури, відносної вологості повітря, сили таступеня вібрації, звукового шуму, вогнестійкості, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря.

Щоб забезпечити високий рівень комфорту та безпеки під час роботи з комп'ютером, на кожну кімнату, де працюють співробітники, повинні бути наявні елементи природного та штучного освітлення відповідно до ДБН В.2.5-28-2006 Для досягнення максимального рівня безпечності та охорони праці при роботі з комп’ютером виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації й вогнегасниками. На вікнах повинні бути встановлені сонцезахисні плівки.

При розташуванні елементів робочого місця користувача ПК необхідно враховувати робочу позу користувача, простір для розміщення користувача, можливість огляду елементів робочого місця, можливість ведення записів, розміщення документації та матеріалів, якими користуватиметься працівник. Робочі місця з ПК мають бути розташовані від стіни з вікнами на відстані не менше ніж 1,5 м, від інших стін - на відстані не менше ніж 1 м. Недопустиме таке розташування ПК, при якому працівник повернений обличчям або спиною до вікон кімнати або до задньої частини ПК, у яку вмонтовані вентилятори.Загальні рекомендації до робочої пози та робочого місця наведені на рисунку 5.2

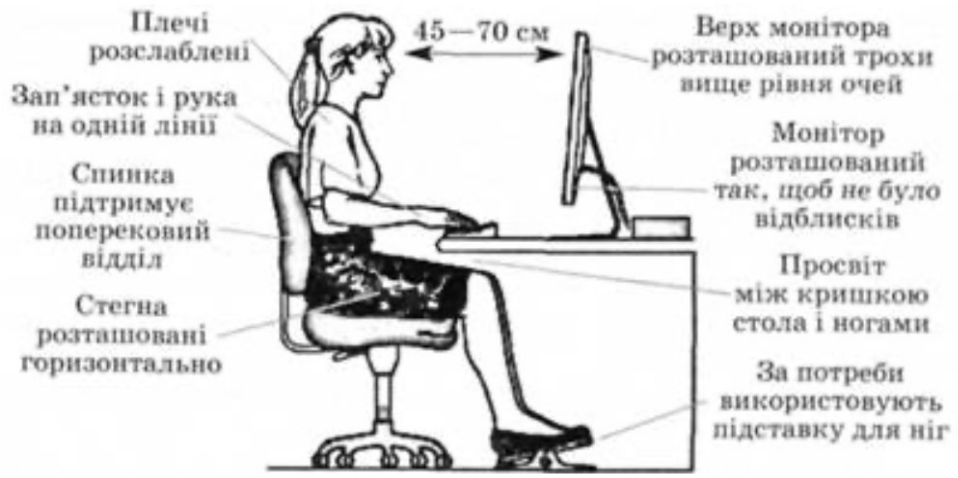

Рисунок 5.2 – Рекомендації до робочої пози та робочого місця

ВИСНОВКИ

У дипломній роботі було розроблено та проєктно обґрунтовано створення онлайн-платформи для вивчення програмування з використанням сучасних вебтехнологій, зокрема фреймворку Nest.js. Основна мета роботи — реалізація зручної, функціональної та масштабованої системи, що дозволяє користувачам опановувати програмування в інтерактивному форматі — була досягнута.

У ході виконання роботи було проаналізовано сучасний стан розвитку онлайн-освіти, особливо у сфері інформаційних технологій. Встановлено, що попит на онлайн-навчання програмуванню постійно зростає, водночас більшість існуючих рішень або складні для початківців, або не забезпечують ефективного поєднання теорії з практикою. Це обґрунтовує актуальність розробки нових освітніх рішень.

На основі зібраних вимог було проведено системне проєктування архітектури майбутнього вебзастосунку, визначено функціональну структуру, обрано мову програмування (JavaScript/TypeScript) та стек технологій. Для реалізації серверної частини було використано Nest.js, що дозволило застосувати модульну архітектуру, розділення логіки по функціональних блоках, зручну інтеграцію з базою даних (PostgreSQL) та підтримку REST API.

У результаті було реалізовано MVP-версію онлайн-платформи з наступним функціоналом:

1. реєстрація та авторизація користувачів;
2. створення курсів, уроків і практичних завдань (кодових вправ);
3. надсилання рішень, перевірка й збереження результатів;
4. можливість адміністрування контенту;
5. інтеграція CI/CD для автоматичного розгортання.

Під час розробки використано систему контролю версій (Git), ORM Prisma, а також хмарну платформу Render для деплою проєкту.

У розділі з техніко-економічного обґрунтування було здійснено оцінку вартості розробки MVP. Розрахунки показали, що загальна вартість створення платформи, включаючи трудовитрати, хостинг і тестування, становить приблизно 7911,39 грн. Додатково оцінено витрати на щомісячну підтримку проєкту, які можуть складати 1 000–1 500 грн. Це свідчить про економічну доцільність реалізації такого продукту на практиці.

Враховуючи результати дипломної роботи, можна зробити висновок, що обрана платформа відповідає сучасним вимогам до вебдодатків, а обрані інструменти дозволяють у майбутньому легко масштабувати функціонал, інтегрувати нові модулі, зберігаючи стабільність і зручність використання. Отже, створена система може стати основою для повноцінного навчального сервісу в галузі ІТ-освіти.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. — [Офіційний стандарт].

2. NestJS Documentation. — [Електронний ресурс]. — Режим доступу: https://docs.nestjs.com

3. TypeScript Handbook. — [Електронний ресурс]. — Режим доступу: https://www.typescriptlang.org/docs/

4. PostgreSQL Documentation. — [Електронний ресурс]. — Режим доступу: https://www.postgresql.org/docs/

5. ReactJS Documentation. — [Електронний ресурс]. — Режим доступу: https://react.dev/

6. Prisma ORM Documentation. — [Електронний ресурс]. — Режим доступу: https://www.prisma.io/docs/

7. Render Hosting Platform. — [Електронний ресурс]. — Режим доступу: https://render.com/

8. GitHub Actions Documentation. — [Електронний ресурс]. — Режим доступу: https://docs.github.com/en/actions

9. Codewars — онлайн-платформа для практики програмування. — [Електронний ресурс]. — Режим доступу: https://www.codewars.com/

10. LeetCode — онлайн-сервіс з підготовки до технічних інтерв'ю. — [Електронний ресурс]. — Режим доступу: https://leetcode.com/

11. Мікросервісна архітектура: теорія і практика [Електронний ресурс] / М. Фаулер. — Режим доступу: https://martinfowler.com/articles/microservices.html 屹HYPERLINK "https://dsp.gov.ua/"

12. НПАОП 0.00-1.28-10. Правила охорони праці під час роботи з ЕОМ. — [Офіційне видання].