**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

***Спеціальність: 121 «Інженерія програмного забезпечення»***

***Освітньо-професійна програма: «Розробка прошрманого забезпечення »***

***Група: 4РП-08***

# Дипломний проєкт

**здобувача освіти денної форми навчання**
**РП.08.06.000.ДП**

***ГОЛОВАЧОВА АРТЕМА СТАНІСЛАВОВИЧА***

**м. Одеса**
**2025 р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітньо-професійна програма: «**Розробка програмного забезпечення»**

Група: **4РП-08**

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

## Розробка програмного забезпечення для on-line навчання основам кібербезепки

Проєктний матеріал складається з пояснювальної записки на 76 сторінках та графічного (презентаційного) матеріалу на 13 аркушах (слайдах)

Дипломник _____ ( Головачов А.С. )

Керівник _____ ( Залапін О.І. )

**Консультанти:**

з економічного розділу _____ ( Канський М.Ю. )

з розділу охорони праці та техніки безпеки _____ ( Чорновол Н.І. )

з нормоконтролю _____ ( Петрашова В.І. )

старший консультант _____ ( Кривченко Ю.В. )

**До захисту допущений**

Голова циклової комісії _____ ( Кривченко Ю.В. )

Завідувач відділення _____ ( Краснокутська К.Г. )

Захист «26» 06 2025 р. Протокол ЕК № 2

Оцінка ЕК 4/80

Секретар ЕК _____

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"12" 05 2025 р.

# ЗАВДАННЯ

**на дипломний проект**

Здобувачеві освіти *Головачову Артему Станіславовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмного забезпечення для on-line навчання основам кібербезпеки*

затверджена наказом по коледжу від "14" 11 2024 р. № 246

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *Програмне забезпечення реалізувати мовою програмування Typescript з фреймворком Angular у середовищі розробки Visual Studio code; 2. Розмітку реалізувати завдяки мови розмітки HTML; 3. Стилі реалізувати завдяки язику декорування і описання зовнішнього вигляду CSS; 4. Побудову артхітектури проекту реалізувати в графічному редакторі Figma.*

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

*Аналіз сучаних технологій; Проектування архітектури веб-додатку; Реалізація функціоналу додатку; Загальна архітектура проекту; Оптимізація та безпека*

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

*Опис мети проекту; Опис використаних технологій; Архітектура системи, ключові переваги, та загальна взаємодія; Скрішоти головного інтерфейсу, системи аутентифікації, особистого кабінету, розділ "my-course", розділ „about-us". Технічні особливості реалізації додатку та його ключові моменти. Перспективи та напрямки подальшого розвитку.*

1. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| *Основний розділ* | *Залапін О.І.* | | |
| *Економічний розділ* | *Канський М.Ю.* | | |
| *Розділ охорони праці* | *Чорновол Н.І.* | | |
| *Нормоконтроль* | *Петрашова В.І.* | | |
| *Старший консультант* | *Кривченко Ю.В.* | | |

2. Дата видачі завдання 29 квітня 2025р

Керівник Залапін О.І. _____ (підпис)

Завдання прийняв до виконання Головачов А.С. _____ (підпис)

| № s/p | Назва етапів дипломного проекту(роботи) | Термін виконання етапів дипломного проекту (роботи) | Відмітка про виконання |
|---|---|---|---|
| 1. | *Постановка задачі проектування* | 19.05 | виконано |
| 2. | *Аналіз сучасних технології розробки(Angular, Firebase, TypeScript)* | 21.05 | виконано |
| 3. | *Проектування архітектури веб-додатку* | 24.05 | виконано |
| 4. | *Розробка інтерфейсу(UI/UX) та вибір бібліотек* | 27.05 | виконано |
| 5. | *Налаштування Firebase* | 29.05 | виконано |
| 6. | *Реалізація аутентифікації та ролей користувачів* | 02.06 | виконано |
| 7. | *Розробка CRUD-операцій для курсів та навчальних матеріалів* | 04.06 | виконано |
| 8. | *Релізація функціоналу тестування та перевірки знань* | 06.06 | виконано |
| 9. | *Розгортання додатку на Firebase Hosting* | 09.06 | виконано |
| 10. | *Аналіз результатів тестування та ефективності роботи алгоритмів* | 10.06 | виконано |
| 11. | *Виконання графічної частини проекту* | 12.06 | виконано |
| 12. | *Виконання економічних розрахунків* | 13.06 | виконано |
| 13. | *Підготовка пояснювальної записки та оформлення документації* | 14.06 | виконано |
| 16. | *Підготовка доповіді та графічних матеріалів* | 16.06 | виконано |

Дипломник _____ (підпис)

Керівник _____ (підпис)

| Форма | Зона | Поз. | Позначення | Назва | Кіл. | Примітка |
|---|---|---|---|---|---|---|
| | | | | Документація | | |
| | | | РП 08. 06 000. 00 ДП | Дипломний проект | | |
| А4 | | | РП 08. 06 000. 00 ДП ПЗ | Пояснювальна записка | 1 | |

| Зм. | Арк. | № докум. | Підпис | Дата | РП 08. 06 000. 00 ДП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Головачов А.С. | | 16.06.25 | Розробка програмного забезпечення для on-line навчання основам кібербезпеки | Літ. | Аркуш 4 / Аркушів 76 |
| Перевірив | | Залапін О.І. | | 16.06.2 | | | |
| Н. Контр. | | Петрашова В.І. | | 16.06.25 | | ВСП "ОТФК ОНТУ" гр.4РП-08 | |
| Затверд. | | Кривченко Ю.В. | | 18.06 | | | |

# ЗМІСТ

# ВСТУП

Сучасний цифровий світ стрімко розвивається, а разом із цим зростають ризики, пов’язані з кіберзлочинністю, витоком персональних даних та кібератаками. В умовах коли інформаційна безпеки стає критично важливою як для приватних осіб, так і для державних структур та підприємств, актуальним є формування базових знань з кібербезпеки навіть серед користувачів без технічного фаху. Проте традиційні форми навчання вимагають значних ресурсів, а їх доступність для широкої аудиторії залишається обмеженою.

Особливо гостро постає потреба в онлайн – інструментах, що дозволяють самостійно та інтерактивно вивчати основи кіберзахисту. Одним із ефективних рішень у цьому напрямку е створення веб-застосунків.

У рамках дипломного проекту передбачається розробка онлайн-застосунку для навчання основам кібербезпеки, створеного на базі фреймворку Angular з використанням платформи firebase для зберігання даних, аутентифікації та розгортання. Програмне забезпечення включатиме візуальні інтерфейси, адаптивну навігацію, що забезпечить зручність використання на різних пристроях і підвищить ефективність засвоєння матеріалу.

Актуальність даного проєкту обумовлена зростанням масштабів кіберзагроз у всьому світі. У наш час кібератаки — це не лише проблема окремих компаній чи технічних фахівців. Шахрайські листи, фішингові сайти, витоки паролів, злами акаунтів — усе це загрози, з якими стикається кожен користувач Інтернету. Саме тому базові знання з кібербезпеки мають бути доступні кожному, незалежно від професійного рівня чи досвіду.

Пандемія COVID-19 стала переломним моментом у розвитку дистанційного навчання. Масовий перехід на онлайн-формати роботи та освіти продемонстрував, наскільки важливо мати доступ до якісних цифрових освітніх ресурсів. У той же час, ця ситуація показала й вразливість користувачів перед кіберзагрозами — значна частина населення не була готова до безпечної взаємодії в онлайн-середовищі.

# 1 ОСНОВНИЙ РОЗДІЛ

## 1.1 Аналіз сучасних технологій розробки

У сучасних умовах цифрової трансформації, розробка освітніх веб-застосунків потребує ретельного добору, які забезпечують не лише функціональність і продуктивність, а й масштабованість, безпеку та зручність використання. У рамках реалізації дипломного проекту на тему „Розробка програмного забезпечення для on-line навчання основам кібербезпеки” було проаналізовано та обрано комплекс сучасних вебтехнології, до якого увійшли: Angular як фронтенд – фреймворк для побудови SPA, Firebase як хмарна платформа для бекенду, TypeScript як основна мова програмування, а також GitHub як інструмент для організації командної роботи та контролю версій.

Обрані технології дозволяють забезпечити повноцінну взаємодію між користувачем та навчальним середовищем, надавати доступ до персоналізованого контенту, зберігати та аналізувати результати навчання. Особливої актуальності набуває інтеграція цих технології у контексті створення системи з підтримкою користувацької аутентифікації, управління прогресом та динамічного відображення матеріалів відповідно до структури курси.

### 1.1.1 Огляд Angular та його можливостей для SPA-застосунків

Angular – це потужний фреймворк з відкритим кодом, створений компанією Google для побудови складних веб-застосунків, зокрема односторінкових (SPA – Single Page Application). У контексті даного дипломного проекту Angular виконує ключову роль, у реалізації клієнтської частини навчального порталу.

Завдяки модульній архітектурі Angular забезпечує розділення відповідальностей та повторне використання компонентів. Кожна частина інтерфейсу, така як сторінку логіну, перегляд уроку, або тестування, реалізована у вигляді окремого компонента. Angular також дозволяє ефективно управляти маршрутизацією – перемиканням між сторінками без повного перевантаження веб-сторінки. Це забезпечує швидку та плавну взаємодію з інтерфейсом, що є

критично важливим для навчальних платформ, де зручність і швидкість доступу до контенту мають вирішальне значення.

Також Angular має вбудовану підтримку фон, валідації даних серверів для організації бізнес-логіки, а також можливості реактивного програмування через бібліотеку RxJS. У реалізованому проекті ці можливості були використані для побудови форм авторизації, реєстрації, створення динамічних тестів та відображення навчального контенту.

### 1.1.2 Firebase як хмарна платформа для фронтенд-розробника

Firebase – це набір хмарних сервісів від Google, які забезпечують повноцінну backend – інфраструктуру без необхідності створення власного сервера. У цьому проекті Firebase використовується для реалізації настпуних функцій:

- Аутентифікація користувачів (Firebase Authentication): реалізовано можливість реєстрації нових користувачів.
- База даних Firestore: зберігає дані користувача, прогрес, результати, тощо. Структура бази дозволяє організувати навчальні модулі у вигляді колекцій і документів, що дуже зручно для ієрархічних освітніх систем.
- Хостинг (Firebase Hosting): використовується для деплою застосунку у веб, що дозволяє швидко зробити його доступним для користувачів через інтернет.
- Функції безпеки (Firebase rules): дають можливість налаштовувати правила доступу до даних у Firestore залежно від статусу користувача.

Однією з головних переваг Firebase є легка інтеграція з Angular, що значно спрощує взаємодію клієнта з базою даних у реальному часі. Також варто відзначити зручну систему розгортання та масштабованість проєктів без необхідності обслуговування серверів. Це робить Firebase ідеальним вибором для швидкої розробки сучасних веб-застосунків.

### 1.1.3 Інтеграція Angular з Firebase.

У рамках дипломного проекту було застосовано бібліотеку angular/fire для інтеграції з сервісами Firebase. Ця бібліотека є офіційним мостом між двома платформами, надаючи синтаксично зрозумілі методи для взаємодії з базою даних, аутентифікацією, хостинг та аналітикою.

Однією з ключових переваг інтеграції є можливість реалізації реактивного відображення даних, коли будь-які зміни у Firestore одразу ж відображаються у компонентах Angular. Це стало особливо важливим для реалізації розділу з динамічними тестами: користувачі бачать результат в реальному часі, а адміністратор може відслідковувати активність.

### 1.1.4 Огляд TypeScript як основної мови проекту

Усі функціональні компоненти Angular – застосунку були реалізовані з використанням TypeScript – надмножини мови JavaScript, яка додає підтримку статичної типізації. Це дозволило значно зменшити кількість помилок у коді завдяки можливості перевірки типів ще на етапі компіляції, а також покращити структурування програмної логіки.

TypeScript також забезпечує потужні засоби для роботи з об’єктно – орієнтованим підходом: класи, інтерфейси, наслідування тощо. Це дозволило краще організовувати взаємодію між компонентами, сервісами та модулями. Під час реалізації проекту було створено низку інтерфейсів для опису навчальних модулів, користувачів, тестових завдань тощо. Це полегшило роботу над масштабуванням і уніфікацією логіки обробки даних

### 1.1.5 Використання GitHub для контролю версій

У межах реалізації проекту, GitHub був обраний як основна платформа для керування вихідним кодом і координації командної роботи. Репозиторій проекту структурувався відповідно до принципів розробки з використанням системи контролю версій Git. Це дозволило відстежувати історію змін, координувати паралельну роботу над різними модулями, а також ефективно проводити ревізію коду.

Для підтримки порядку у розробці застосовувалася система гілок, що передбачала розмежування стабільної версії(гілка main), і змін які тестувалися (у гілці fix-branch, в нашому випадку).

Функціональність GitHub також активно застосовувалася для документування процесу розробки. Через інструменти GitHub здійснювався контроль за якістю коду, що значно покращило прозорість проекту.

Окрім технічних переваг, GitHub відіграв важливу роль у взаємодії з науковим керівником, оскільки платформа забезпечує можливість наочно демонструвати поточний стан проекту в будь-якій момент часу.

### 1.1.6 Visual Studio Code як основне середовище розробки веб-застосунку

У процесі розробки програмного забезпечення для онлайн-навчання основам кібербезпеки було використано Visual Studio Code(VS Code) – одне з найпопулярніших сучасних середовищ розробки веб-технологій(рис 1.1).

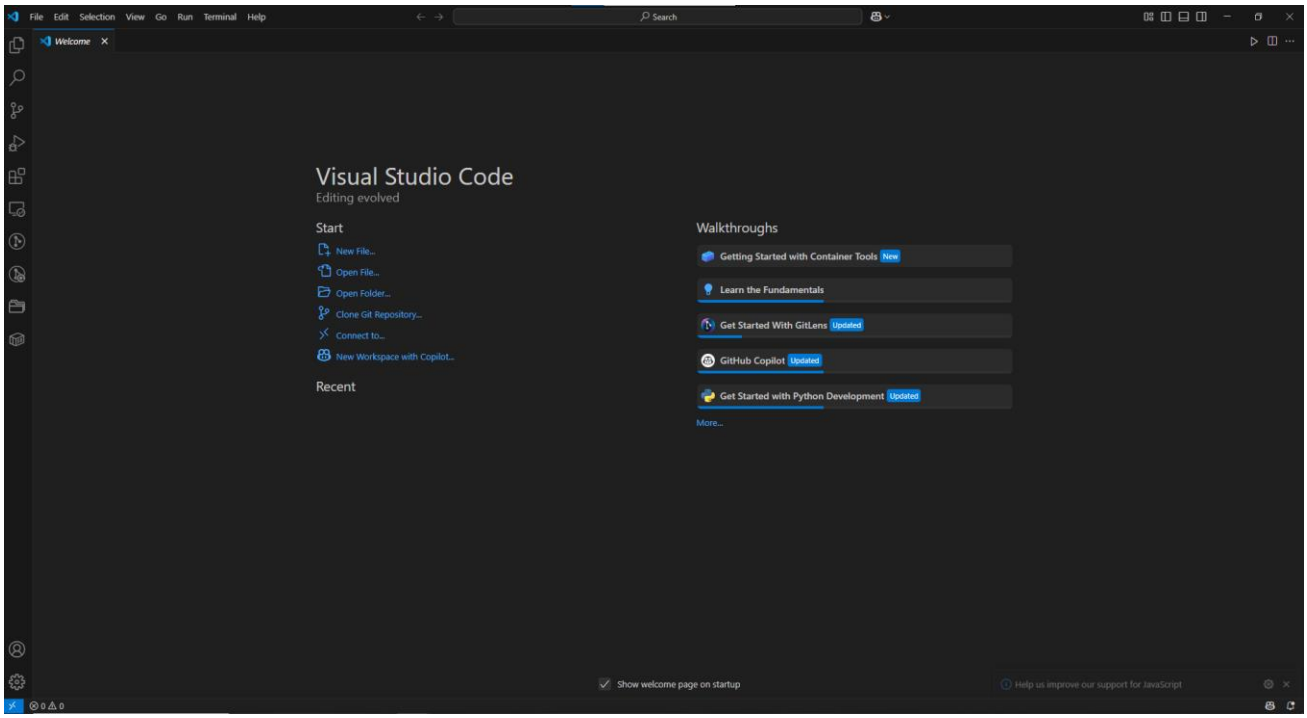



Рисунок 1.1 Середа розробки Visual Studio code

Однією з ключових переваг VS Code є підтримка мов програмування, які використовувалися в проекті – зокрема в TypeScript, HTML і CSS. Середовище забезпечує глибоку інтеграцію з Angular CLI, що дозволяє ефективно

створювати нові компоненти, керувати модулями та проводити компіляцію безпосередньо з вікна редактора. Завдяки розширенням таким як Angulat Language Service, розробник отримує підказки, автозавершення коду, перевірку помилок у шаблонах і можливість навігації між компонентами, що значно прискорює роботу.

У контексті використання Firebase, розширення Firebase Explorer дозволяє інтегрувати функціонал хмарної бази даних безпосередньо в редактор, забезпечує зручний доступ до FireStore, аутентифікації, хостингу та інших сервісів. Це дозволяє відслідковувати зміни в реальному часі.

Також є повний функціонал для роботи з Git та GitHub, у межах VS Code, використовуючи вбудовані інструменти взаємодія з Git та GitHub була моментальна, та легка.

Завдяки VS Code розробка відбувалась у продуктивному, та інтуїтивно зрозумілому середовищі, що забезпечувало високу швидкість та реалізацію поставлених задач.

### 1.1.7 Використання Figma для проєктування структури веб-застосунку

У процесі розробки сучасного веб-застосунку особливо важливим етапом є попереднє планування структури, логіки та візуального вигляду майбутнього інтерфейсу. Саме на цьому етапі формується перше враження про продукт, визначається користувацький досвід (UX), загальна стилістика (UI) та зручність взаємодії користувача з системою. Для забезпечення чіткої візуалізації майбутнього інтерфейсу, узгодження дизайну між усіма учасниками команди та швидкого внесення змін у структуру було використано Figma — потужний онлайн-інструмент для створення прототипів і макетів(рис 1.2).

Figma є однією з найпопулярніших платформ для спільного дизайну, яка дозволяє працювати безпосередньо у браузері без необхідності встановлення додаткового програмного забезпечення. Її інтерфейс інтуїтивно зрозумілий, що робить її доступною навіть для тих користувачів, які не мають глибоких знань у сфері дизайну. Завдяки цьому Figma стала незамінним помічником у підготовчому етапі створення освітнього порталу.

У межах роботи над дипломним проєктом Figma використовувалася для кількох основних цілей. Насамперед — для візуалізації структури сайту: було створено прототип головної сторінки, сторінки логіну та реєстрації, інтерфейс уроку, блок з тестовими завданнями, а також макети сторінок з результатами навчання. Кожен елемент було ретельно продумано, розміщено відповідно до логіки користувацького шляху, що дозволяло заздалегідь побачити, як саме виглядатиме система з погляду кінцевого користувача.

Однією з переваг Figma є її можливості для командної роботи. Усі зміни зберігаються в хмарі в режимі реального часу, тож дизайн можна обговорювати, редагувати та коментувати спільно з викладачем або іншими учасниками команди без необхідності пересилати файли або турбуватися про версію документа. Також ця платформа дозволяє створювати інтерактивні прототипи — тобто наочно демонструвати, як саме працюватимуть кнопки, переходи між сторінками, спливаючі вікна тощо.

Окрім створення макетів, Figma стала своєрідним інструментом для організації думок та ідей. Її використання допомогло краще зрозуміти логіку розміщення контенту, визначити зони, що потребують додаткової уваги, оптимізувати розташування елементів та підвищити загальну зручність користування застосунком. Усе це в комплексі дозволило уникнути багатьох проблем ще до початку написання коду, заощадити час на розробці, та створити більш цілісне і якісне рішення.

Figma стала не просто допоміжним інструментом, а важливим етапом процесу розробки, який дозволив перетворити ідею в конкретну, візуально зрозумілу та структуровано оформлену модель майбутнього застосунку. Figma стала не просто допоміжним інструментом, а важливим етапом процесу розробки, який дозволив перетворити ідею в конкретну, візуально зрозумілу та структуровано оформлену модель майбутнього застосунку. Це дозволило швидше узгоджувати дизайн та уникати помилок на старті. Візуалізація інтерфейсу в Figma допомогла краще продумати навігацію, логіку екранів та зручність взаємодії для кінцевого користувача.

## 1.2 Проектування архітектури веб-додатку

### 1.2.1 Вибір структури проекту (модулі, компоненти, сервіси)

Архітектура Angular додатку побудована на основі сучасних підходів до розробки, з дотриманням принципів SOLID, де S - Single Responsibility Principle, O – Open/Closed Principal, L – Liskov Substitution Priniciple, I – Interface Segregation Principle, D – Dependence Inversion Principle, модульності та чіткого розділення відповідальності між частинами системи. Це не просто набір модулів і компонентів – це гнучка, логічно вибудована структура, яка дозволяє розвивати застосунок, не створюючи технічного боргу. Такий підхід уже давно є стандартом в розробці масштабних веб-додатків, і ми дотрималися його максимально.

Центральне місце у додатку займає AppModule. Він виступає як головна точка входу, яка поєднує в собі всі інші модулі, компоненти, сервіси, директиви, а також налаштування маршрутизації. Саме тут відбувається імпорт усіх ключових Angular – модулів, таких як FormsModule, ReactiveFormsModule, HttpClientModule, і звичайно ж, модулі AngularMaterial, які використовуються для побудови красивого та зручного інтерфейсу. Така організація дозволяє централізовано керувати всіма залежностями та логікою додатку, що значно полегшує подальший розвиток, підтримку і масштабування системи. Крім того, це дуже зручно для командної роботи, кожен розробник чітко бачить межі своїї відповідальності, легко знаходить потрібні частини коду та швидко орієнтується в архітектурі.

Кожна частина функціоналу винесена в окремі компоненти, щоб забезпечити максимальну гнучкість і повторне використання.

- CourseListComponent – це компонент, який відповідає за виведення списку доступних курсів. Він підтримує зручну фільтрацію та пошук, а також інтегрується з системою ролей: в залежності від того чи користувач є викладачем чи студентом, змінюється доступний функціонал. Це дозволяє створювати індивідуальний досвід для кожного типу користувача.

- CourseDetailComponent – надає детальний перегляд вибраного курсу. Користувач може переглядати його секції, додавати курс до свого списку або, навпаки, видаляти. Для викладачів доступне управління змістом курсу – створення, редагування, видалення секцій.
- CourseMakeComponent – редактор курсів. Тут реалізовано створення і редагування з підтримкою динамічних секцій, валідацій форм і навіть завантаження файлів. Усе це побудовано на реактивних формах, що робить інтерфейс швидким, а досвід користування – приємним.
- CourseItemComponent – окремий візуальний елемент списку курсів.Завдяки цьому компоненту ми можемо повторно використовувати логіку відображення курсу в різних місцях додатку, не дублючи код.
- CourseStartComponent – це стартова сторінка розділу “Курси”. Воні надає коротке пояснення, що робити далі, і допомогає користувачеві з першими кроками у взаємодії з цим функціоналом.
- ShoppingCoursesComponent – компонент для роботи з курсами, які користувач додав до своїх. Тут можна переглядати список курсів, вибирати їх для детального перегляду, відкривати секції, завантажувати навчальні матерали.
- ShoppingEditComponent – надає можливість редагувати чи додавати нові секції до курсів. Вбудована інтеграція з реактивними формами дозволяє забезпечити гнучкість та надійність при введені даних.

Профіль користувача: (сделать как список выше)

- MyProfileComponent – відповідає за відображення профілю. Тут користувач бачить свій аватар, біографію, електронну пошту, вік, нікнейм. Також є зручний перехід до режиму редагування.
- MyProfileEditComponent – дозволяє редагувати дані профілю, вибирати аватар, вносити зміни у біо та інші поля. Вся інформація синхронізується з Firebase, що забезпечує надійне збереження змін. При збереженні змін користувач бачить повідомлення про успішне оновлення даних, а також може одразу переглянути результат у власному профілі.

- AuthComponent – універсальний компонент для логіну та реєстрації. Підтримує вибір ролі користувача, обробку можливих помилок, показує індикатор завантаження та дає підказки для покращення UX.
- HeaderComponent – головний навігаційний компонент, який адаптується під роль користувача та його авторизаційний статус. Має інтеграцію з Angular Material Dialog, що покращує взаємодію з користувачем.
- MarqueeBannerComponent – інформаційна стручка, що повідомляє користувача про важливі оновлення, сповіщення тощо. Наприклад про нові курси, або важливі сповіщення про зміну чогось на сайті.
- DropdownDirective – універсальна директива, яка забезпечує зручну роботу з випадаючими меню. Використовується у різних частинах інтерфейсу, що сприяє єдиному UX – стилю
- CourseService – головний сервіс для управління курсами. Реалізує повний CRUD(create - створення, receive - отримання, update - оновлення, delete - видалення), дозволяє додавати курси до особистих, оновлює дані реактивно через Subject.
- ShoppingCoursesService – обробляє особисті курси користувача, дозволяє зберігати вибраний курс, редагувати його структуру, додавати чи видаляти секції. Це зручно для створення персоналізованого навчального досвіду.
- AuthSevice – відповідає за все, що пов’язане з аутентифікацією: логін, реєстрація, збереження сесій, відновлення сесій, ролі користувачів, автологін і автологаут. Працює з Firebase Auth REST API.
- DataStorageService – забезпечує весь обмін даними з Firebase. Саме через цей сервіс відбувається збереження, оновлення та завантаження курсів, профілю та інших даних. Його використання дозволяє централізувати всю логіку роботи з сервером.
- AuthInterceptorService – автоматично додає токен авторизації до кожного HTTP – запиту, що робить роботу з API безпечною та прозорою для розробників. Це знімає необхідність вручну додавати заголовки запитів у кожному комопненті

– AuthGuard та TeacherGuard – використовуються для захисту иашрутів. AuthGuard дозволяє доступ тільки авторизованим користувачам, тоді як TeacherGuard – тільки тим, хто має роль”teacher”. Це забезпечує контроль доступу та дотримання логіки розподілу прав у додатку прав у додатку.

### 1.2.2 Розробка схеми взаємодії клієнтської та серверної частин

Вся взаємодія між Angular – додатком і серверною частиною, яка реалізована на базі Firebase, побудована відповідно до сучасних підходів – з використанням REST API та реактивних потоків даних. Такий підхід забезпечує високу продуктивність, безпеку та зручність у розробці, дозволяючи будувати гнучкі та масштабовані рішення, які легко підтримувати та розширювати (рисунок 1.2).

У нашому додатку всі основні дії х даними – створення, читання, оновлення та видалення (тобто повноцінна підтримка CRUD – функціональності) – виконуються через HTTP – запити Firebase Realtime Database. Ми використовуємо стандартні методи протоколу HTTP, такі як GET, POST, PUT. Та DELETE, які дозволяють ефективно керувати даними в базі.
Ці операції застосовується до таких сутностей, як:

Курси – загальна база курсів, доступна для перегляду й керування (залежно від ролі)

Секції – підрозділи курсу, які можна створювати та редагувати.

Профілі користувачів – особисті дані, які редагуються у відповідному розділі профілю.

Ролі користувачів – визначають права доступу до функціоналу в додатку.

Що важливо, кожен запит до приватних або важливих ресурсів супроводжується авторизаційним токеном, який автоматично додається до заголовків запиту за допомогою HTTP Interceptor. Це забезпечує надійність рівень захисту: лише ті користувачі, які пройшли аутентифікацію, мають доступ до своїх даних або до дій, дозволених їхньою роллю.

Уся структура збереження даних у Firebase є логічною, простою та інтуїтивно зрозумілою. Наприклад:

/courses – тут зберігається загальний каталог курсів.

/my-courses/{userId} – особисті курси кожного користувача, доступні лише йому.

/ profiles/{userId} – профільні дані конкретного користувача.

/userRoles/{userId} – роль, яка визначає доступ до функціоналу (наприклад student або teacher)

Процес аутентифікації реалізовано через Firebase Authentication REST API. Це дозволяє користувачам швидко реєструватися або входити в систему, використовуючи лише emai і пароль. Після успішного входу користувач отримує спеціальний токен - своєрідний”електронний пропуск”, який зберігається у localStorage. Завдяки цьому забезпечує автоматичний логін при повторному відкритті додатку – користувач не повинен вводити логін і пароль щоразу.

Особливістю нашої реалізації є зберігання ролі користувача окремо у базі. Це дозволяє системі при кожному вході підтягувати роль(наприклад студент чи викладач) і відповідно до неї активувати певний функціонал у додатку. Такий підхід забезпечує гнучкість і точність при визначені прав доступу.

Захист даних: багаторівнева безпека

Безпека – один з пріоритетів у побудові нашої взаємодії з сервером. Ми подбали про те, щоб жоден запит до захищених ресурсів не виконувався без перевірки токена авторизація. Це стосується всіх дій із приватними даними: від перегляду профілю до редагування особистих курсів.

Крім того, реалізовано контроль доступу на обох рівнях: як на клієнтський стороні, так і на сервері. З боку клієнту – це Guards, які перевіряють, чи має користувач доступ до певного маршруту. Наприклад, тільки авторизовані користувачі можуть потрапити у розділ профілю, а лише ті, хто має роль teacher, - у розділі створення чи редагування курсів. На серверному рівні додатково використовуються правила доступу Firebase Rules, що забезпечують валідацію користувацьких прав під час кожного запиту до бази. Для кращої прозорості всі спроби доступу логуються в системі.

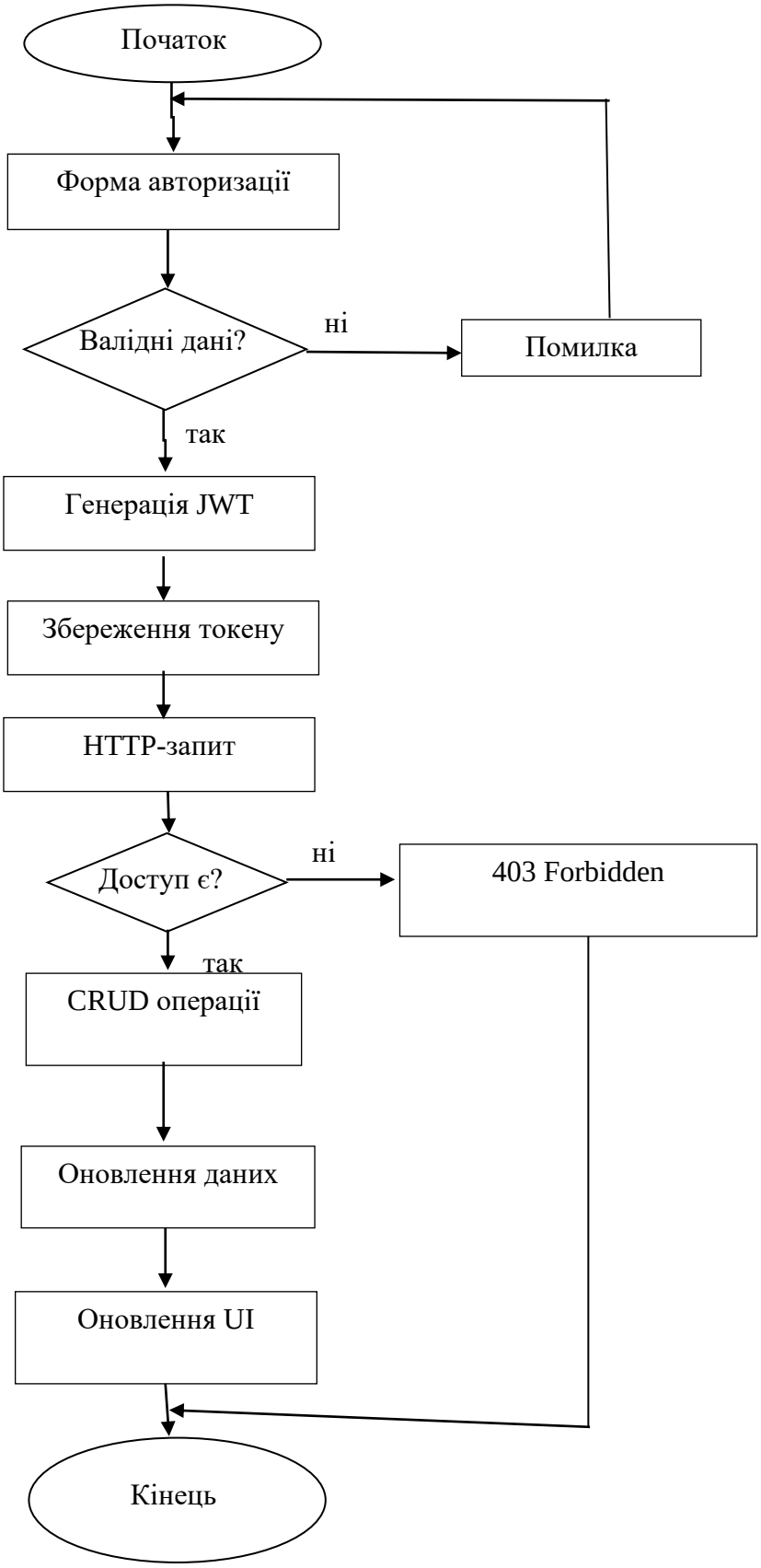



Рисунок 1.2 Алгоритм взаємодії авторизації та CRUD операції

З боку сервера працюють правила безпеки Firebase, які обмежують доступ до бази даних залежно від токена користувача, його ID та ролі. Ця забезпечує двофакторний контроль і значно підвищує безпечність усієї системи.

Реактивність: сучасний користувацький досвід

Один з ключових елементів сучасного фронтенду – це реактивність, тобто здатність інтерфейсу автоматично реагувати на зміни даних. У нашому додатку це реалізовано через RxJS - потужну бібліотеку для роботи з потоками даних.

Дані, які надходять із сервера, зберігаються у відповідних сервісах – це як проміжний кеш або центр обміну інформацією. Сервіси використовують Subject або BehaviorSubject для створення потоків, на які компоненти можуть підписуватися. Завдяки цьому, щойно щось змінюється (наприклад, додається новий курс або змінюється профіль), усі відповідні компоненти отримують оновлення миттєво й автоматично перерисовуються

Це не лише покращує UX, а й дозволяє уникати дублювання коду та повторних HTTP – запитів.

### 1.2.3 Оптимізація стану додатку за допомогою NgRx/RxJS

Однією з ключових переваг використання Angular у великих проектах є можливість реалізувати сучасні реактивні підходи до управління станом і даними. У нашому додатку ця реактивність реалізована за допомогою RxJS бібліотеки, яка є невід'ємною частиною Angular екосистеми. Завдяки RxJS можна побудувати логіку, яка динамічно реагує на зміну даних у реальному часі, що значно підвищує якість користувацького досвіду та спрощує підтримку додатку.

Усі Angular сервіси або сервіс для роботи з курсами профілем користувача або авторизацією – використовують Subject або BehaviorSubject як джерела істини, тобто як централізовані точки, через які відбувається обмін даними між різними частинами додатку.

Наприклад, коли в системі змінюється список курсів, додано новий курс або змінено існуючий, відповідний сервіс оновлює BehaviorSubject. Усі компоненти, які підписані на цей потік даних, миттєво отримують нові значення

й автоматично оновлюють свій інтерфейс. Це означає, що розробників не потрібно вручну оновлювати дані в кожному компоненті – все оновлюється реактивно та узгоджено.

Такий підхід не лише економить час, а й дозволяє уникнути дублювання даних, що в свою чергу зменшує ризик виникнення помилок, пов'язаних із порушенням синхронізації стану.

У додатку активно використовуються оператори RxJS — спеціальні функції для роботи з асинхронними потоками даних у реактивному програмуванні. Вони дозволяють гнучко керувати складними послідовностями подій, такими як обробка користувацьких дій, запитів до серверу чи змін у стані програми. Завдяки RxJS можна легко реалізувати послідовне або паралельне виконання асинхронних операцій, ефективно обробляти помилки, а також відкладати або обмежувати кількість запитів, що надсилаються.

Це особливо важливо для забезпечення плавної роботи додатку, адже правильне управління асинхронністю дозволяє уникнути затримок, дублювання операцій та непередбачуваних збоїв. Таким чином, RxJS оператори допомагають створювати надійні й масштабовані веб-застосунки з чіткою логікою обробки подій і даних у реальному часі.

Серед найбільш уживаних операторів

- take дозволяє контролювати кількість значень, що приходять із потоку, наприклад, брати лише перше оновлення, а далі автоматично відписуватись. Це дуже зручно для запитів або ініціалізацій.
- map - трансформує дані потоку в потрібний формат, дозволяючи адаптувати структуру без змін у джерелі.
- tap – виконує побічні дії без зміни потоку. Наприклад, можна виводити лог у консоль або запускати певну анімацію.
- exhaustMap - особливий корисний у випадках, коли потрібно уникнути паралельного виконання кількох однакових запитів (наприклад, багаторозове натискання на кнопку надсилає лише один запит).

Завдяки цим операторам, код стає не лише лаконічним і зрозумілим, ай значно стійкішим до помилок. Вони дозволяють будувати логіку так, щоб не виникали memory leaks (витік пам'яті), не повторювалися запити та не з'являлися неочікувані проблеми у взаємодії з інтерфейсом.

### 1.2.4 Дизайн UI/UX та вибір бібліотек (наприклад Angular Material)

У процесі планування додатку був використаний онлайн-інструмент, під назвою Figma,за допомогою нього були сформовані структури сторінок сайту таких як:

1. Course(головне меню)

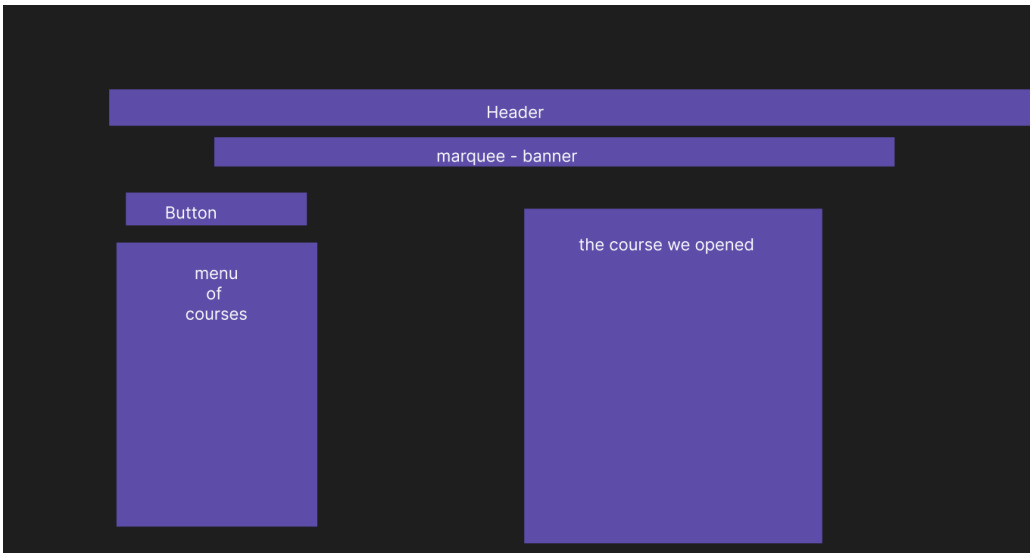



Рисунок 1.3 Головне меню(course)

2. My course(меню наших курсів)

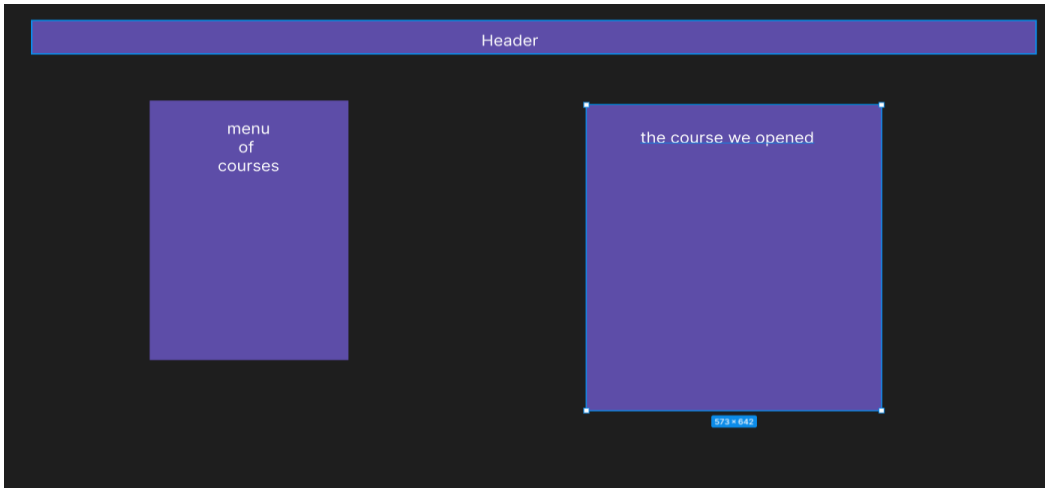



Рисунок 1.4 меню наших курсів

3. About us(сторінка яка описує компанію, та інформація для feedback-у з адмінами)

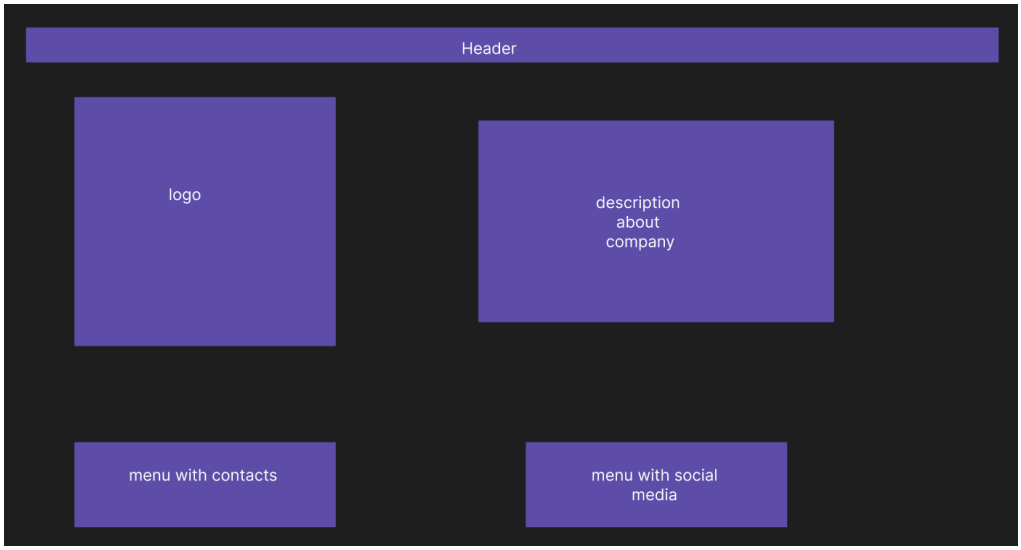



Рисунок 1.5 сторінка нашої компанії

4. My profile(сторінка мого профілю)

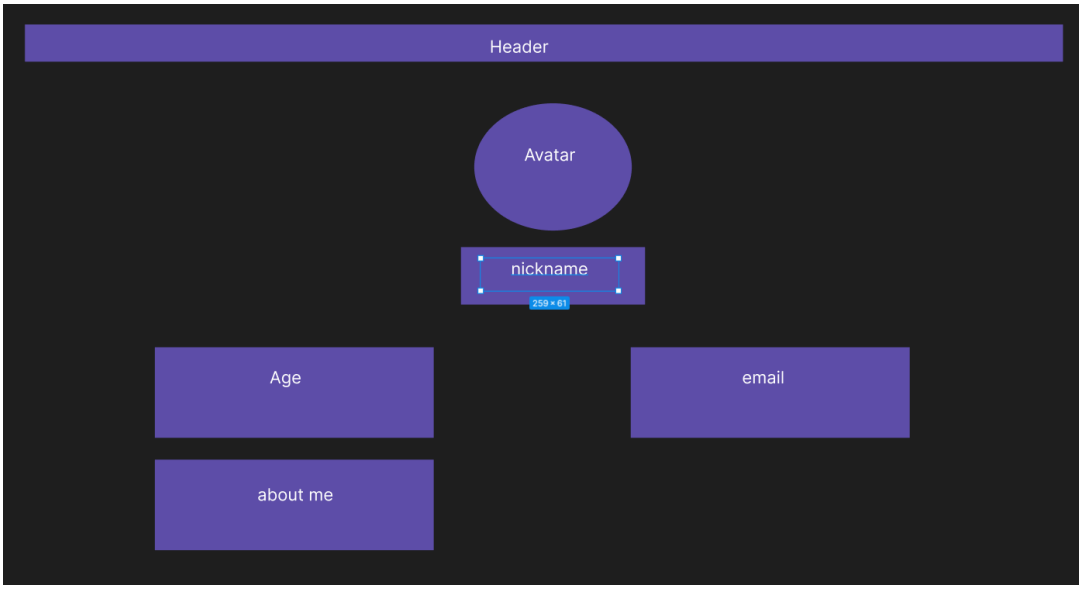



Рисунок 1.6 Сторінка профілю

У процесі створення інтерфейсу додатку було використано поєднання Angular Material, Bootstrap та кастомних стилів, що дозволило сформувати сучасне, візуально привабливе середовище, з яким зручно працювати, і з який приємно сприймати користувачеві.

Angular Material – набір готових UI – компонентів, тісно інтегрований з Angular. Його компоненти використовуються практично повсюдно: від форм

авторизації до списків курсів та редагування профілю. Зокрема, через Material реалізовані:

1.Діалогові вікна для підтвердження або редагування.

2.Форми з інпутами, чекбоксами та селектами.

3.Кнопки різних типів і розмірів.

4.Спінери, які з’являються під час завантаження або очікування відповіді від сервера

Завдяки цьому частини додатку виглядають в єдиному стилі, логічно поєднані між собою і легко читаються незалежно від пристрою. Компоненти, які використовуються, також мають адаптивну поведінку: автоматично підлаштовуються під ширину екрану, змінюють положення елементів, працюють з клавіатурою та екранними рідерами.

Bootstrap – Паралельно з Angular Material у верстці активно застосовується Bootstrap – переважно для керування сітками, позиціонуванням елементів, побудови лейаутів сторінок. Саме Bootstrap дозволяє структурувати контент у колонках, реалізувати гнучке позиціонування блоків, створювати окремі розділи сторінок, які будуть коректно відображатися як на десктопі, так і на мобільних пристроях.

Bootstrap також використовується для окремих UI – елементів, як – от базові кнопки, інформативні повідомлення, маркери, що підсвічують поточний стан, або декоративні елементи сторінок.

У додатку реалізовані також власні стилі, написані вручну з метою надати інтерфейсу індивідуальності та уникнути шаблонності. Особливий вигляд мають:

1.Сторніка профілю, з унікальним розміщенням аватару, біографії, імені.

2. Список курсів – з акцентами на ключових елементах, такими як рейтинг, статус чи тип

3.Банери з повідомленням, які візуально відрізняються від системних .

4.Блоки секцій у курсах, які мають свою логіку відображення, тіні, переходи drag-and-drop стилі.

Інтерфейс спроектовано так, щоб взаємодія з ним була максимально зрозумілою. На кожному етапі користувач отримує візуальні або текстові підказки. Якщо форма заповнена з помилками – з'являється повідомлення, якщо відбувається обробка запиту – виводиться спінер. У місцях, де є логіка з очікуванням (наприклад, збереження або видалення даних), реалізована індикація процесу, щоб уникнути непорозумінь.

У полях форм передбачена підсвітка активних елементів, індикатори валідації та зручні placeholder'и. Для секцій курсів реалізовано можливість drag-and-drop – можна змінювати порядок просто перетягуючи елементи мишкою. У профілі користувача є прев'ю аватара - одразу видно, який файл обрано, ще до збереження.

## 1.3 Реалізація функціоналу додатку

### 1.3.1 Налаштування Firebase(Authentication, Firestore, Hosting)

Firebase Authentication

Для реалізації процесу авторизації та реєстрації користувачів у додатку використовується Firebase Authentication надійне рішення, яке забезпечує повний цикл управління користувачами. Це включає як стандартну реєстрацію за email і паролем, так і всі супутні механізми, необхідні для зручної й безпечної роботи з обліковими записами.

При першому вході в систему користувач вводить свої облікові дані, які надсилаються до Firebase через офіційне REST API. У відповідь додаток отримує токен авторизації, який використовується як підтвердження особи під час подальших запитів до бази даних. Цей токен зберігається у localStorage, завдяки чому сесія зберігається навіть після перезавантаження браузера або повторного запуску програми. Наступного разу, коли користувач відкриває додаток, система автоматично перевіряє наявність токену і, якщо він дійсний, відновлює авторизацію без повторного запиту паролю. Це підвищує зручність користування та забезпечує безперервний доступ до функціональності застосунку.

Крім того, реалізовано функціонал автоматичного виходу(автологауту). Це означає, що після завершення терміну дії токену, користувач автоматично разлогінений – що є важливим з точки зору безпеки.

Ролі користувачів – такі як student чи teacher – не зберігаються у самому токені, а втягуються окремо з бази даних після логіну. Це дозволяє гнучко змінювати права доступу незалежно від механізму авторизації, і не перепідписувати токен при кожній зміні ролі.

Уся інформація, з якою працює додаток – від курсів і секцій до профілів користувачів і їхніх ролей – зберігається в Firebase Realtime Database. Це хмарна NoSql database, яка зберігає дані у вигляді JSON-дерева. Така структура дає змогу створити гнучку, швидку, добре організовану систему зберігання, яку легко масштабувати та обслуговувати.

Для кожного типу даних виділяється окрема гілка:

/courses – всі загальнодоступні курси.

/my-courses/{userId} – курси, які додав конкретний користувач до своїх особистих.

/profiles/{userId} – персональні профілі користувачів з усією інформацією: нікнейм, вік, email, аватар, оротке біо.

/userRoles/{userId} – окрема гілка, яка вказує на роль користувача.

Завдяки цьому дані зберігаються не просто в одному місці, а чітко структуровано. Для кожного користувача створюється окрема приватна область у базі, до якої можуть звертатись лише самі власники. Наприклад, профіль або список особистих курсів доступні лише авторизованому користувачу, який володіє цими даними.

Розгортання (деплой) самого Angular – додатку виконується за допомогою Firebase Hosting – потужного хостингового рішення від Google, розробленого спеціально для односторінкових веб-застосунків (SPA).

Для цього використовується Firebase CLI, через яку генерується фінальна збірка додатку (ng build –prod) і завантажується у хмару. У конфігураційному файлі firebase.json задаються правила маршрутизації, які дозволяють коректно

обробляти всі URL маршрути клієнтського додатку - тобто, якщо користувач заходить напряму на сторінку /course/.../ , Firebase автоматично перенаправляє запит до головного index.html, щоб Angular міг обробити його самостійно.

Firebase Hosting забезпечує високу доступність - незалежно від того, з якої точки світу користувач заходить багато користувачів, вона динамічно розширює свої ресурси, не потребуючи втручання з боку розробника.

Особливу увагу приділено захисту даних користувача. По-перше, всі чутливі налаштування, такі як ключі доступу до Firebase, API-ключі, ID проекту – не зберігаються в коді. Вони розміщені у змінних середовища, завдяки чому не потрапляють до публічного репозиторію, і не можуть бути випадково оприлюднені. По-друге для кожного ресурсу в базу прописані правила доступу. Наприклад, змінювати свій профіль може лише користувач, ID якого відповідає гільці /profiles/{userId}. Аналогічно, ніхто не може видалити чи редагувати курс, якщо не є його автором або не має відповідної ролі (наприклад teacher).Усі запити до Firebase, як на рівні авторизації, так і на рівні бази даних, відбуваються виключно через HTTP. Це виключає можливість перехоплення даним третім особам. (рисунок 1.7, рисунок 1.8, рисунок 1.9).

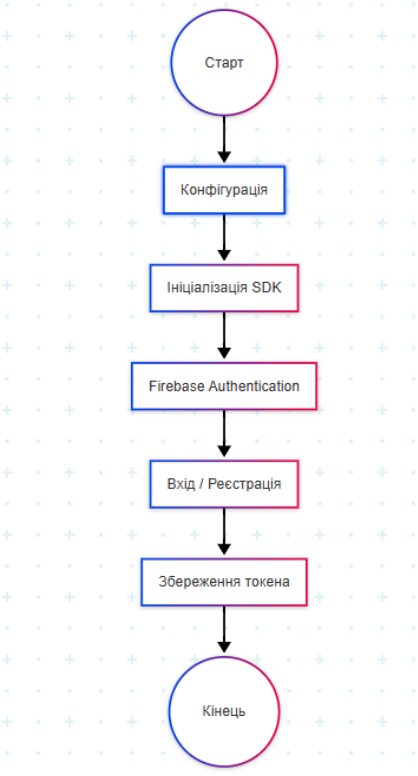



Рисунок 1.7 Аутентифікація та конфігурація

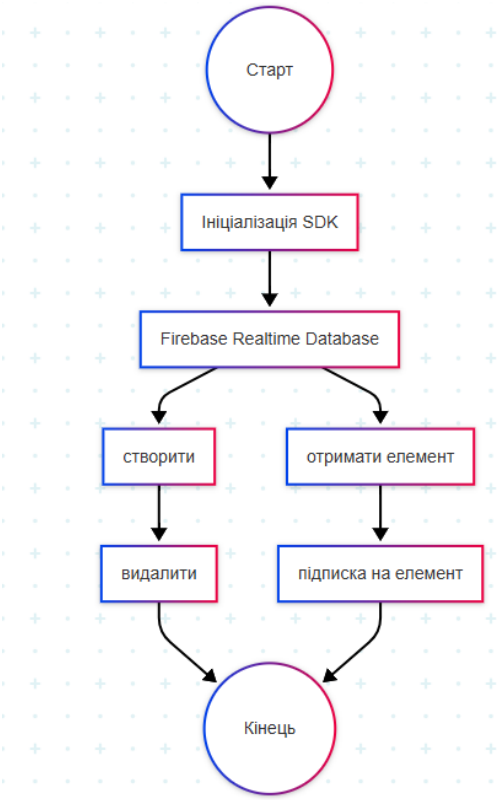



Рисунок 1.8 Робота з базою даних

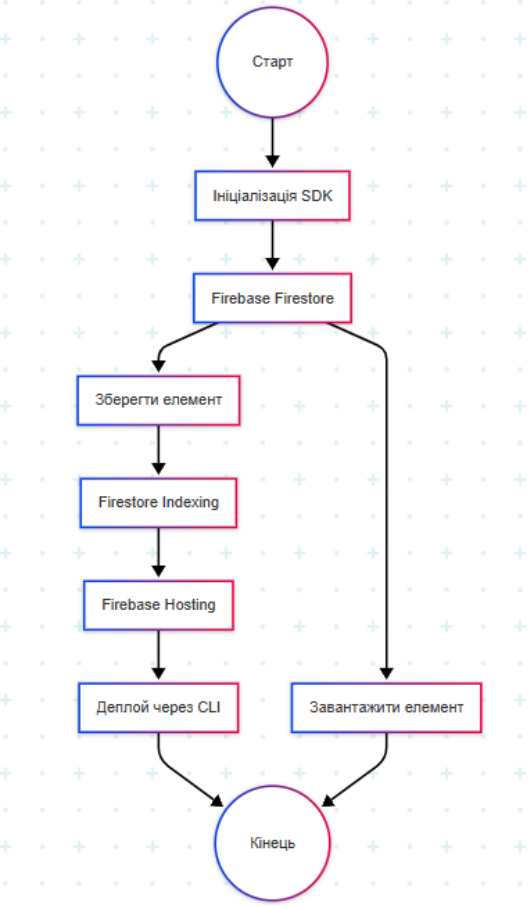



Рисунок 1.9 Firestore та хостинг

### 1.3.2 Розробка CRUD-операцій для роботи з даними

Функціонал управління курсами реалізований з урахуванням ролей користувачів. Додавати нові курси, редагувати існуючи або видаляти їх можуть лише ті користувачі, які мають роль teacher. Ця перевірка відбувається як на рівні клієнта (через route guards) , так і на рівні правил доступу до Firebase Realtime Database.

Після виконання будь-якої CRUD-операції (create, update, delete), зміни одразу фіксуються у базі Firebase і автоматично розповсюджуються по додатку завдяки підписці на реактивні потоки даних (через BehaviorSubject). Ця забезпечує миттєве оновлення UI для всіх користувачів, які працюють із даними курсів.

Кожен курс має вкладену структуру – він містить списко секцій. Ці секції можна редагувати окрему від самого курсу, додавати або видаляти динамічно. Вся структура курсу зберігається як вкладений об’єкт у базі даних.

Кожна секція – це логічна частина курсу, яка може містити теоретичні матеріали, прикріплені файли, інструкції або практичні завдання. Секції редагуються безпосередньо з інтерфейсу курсу.

При створенні чи оновлені секції користувач може прикріпити файл – наприклад , PDF презентацію, зображення або відео. Ці файли конвертуються у формат base64 і зберігаються безпосередньо в Firebase. Завдяки цьому забезпечується доступність контенту без зовнішніх посилань.

Усі секції зберігаються у вкладеній структурі під відповідним курсом. Це дозволяє ефективно будувати інтерфейс перегляду та редагування курсу, а також здійснювати фільтрацію, пошук або відображення секцій у порядку, що відповідає логіці навчання.

Користувач має доступ до розділу „Профіль”, де він може редагувати особисту інформацію – ім’я, прізвище , нікнейм, вік , email, коротке біо та аватар.

Редагування відбувається безпосередньо в інтерфейсі, із валідацією полів та попереднім переглядом змін. Дані зберігаються у Firebase у вузлі

/profiles/{userId} і завжди доступні з будь-якого пристрою, з якого користувач здійснює вхід.

Для вибору аватару реалізовано інтерактивну галерею з попереднім переглядом, що дозволяє миттєво оновлювати зображення профілю без потреби в завантаженні файлів.

Кожен користувач має можливість додавати вподобані або власні курси до особистого списку (/my-courses/{userId}). У цьому розділі користувач може переглядати повну структуру курсу, працювати з секціями, виконувати їх редагування (якщо має відповідні права).

Усі операції – як додавання курсів або окремий курс, підписані на відповідні стріми даних. Це означає, що будь-які зміни (наприклад, додану нову секцію, видалено курс, змінено назву) автоматично відображаються на інтерфейсі користувача без потреби в ручному оновлені сторінки.

### 1.3.3 Реалізація аутентифікації та ролей користувачів

Процес реєстрації реалізований через Firebase Authentication REST API. Користувач може створити обліковий запис, вписавши email, пароль та роль(student або teacher).

Вибір ролі здійснюється на етапі раєстрації через відповідний елемент форми (наприклад, радіо-кнопки чи селектор).

Після успішної реєстрації обліковий запису створюється у Firebase Authentication, а додаткові дані - ролі користувача, профільні поля тощо – записуються в окрему гілку у Firebase Realtime Database:

/userRoles/{userId} – зберігає роль (наприклад teacher або student)

/profiles/{userId} – зберігає ім’я, нікнейм, аватар, тощо.

Усі дані реєстрації супроводжуються UI–підтримкою - спінери під час запиту, підказки при помилках (наприклад, якщо email вже використовується), інформативні повідомлення після успішної реєстрації.

Після входу (login) користувач отримує JWT-токен (idToken)Б який використовується для ідентифікації при наступних запитах. Цей токен збергіаєтсья у localStorage або sessionStorage – залежно від налаштувань – що

дозволяє реалізувати автоматичний вхід (автологін) при повторному відкритті додатку

Також реалізована функція автоматичного виходу з системи (auto-logout) після завершення терміну дії токену. Таймер логіну запускається при кожній аутентифікації та відслідковує дійсність сесії(рисунок 1.10 рисунок 1.11).

Усі запити до приватних або захищених ресурсів(наприклад, /my-course,/profiles,/userRoles) супроводжується токеном авторизації. Цей токен додається д заголовків кожного HTTP-запиту через спеціально створений HTTP Interceptor.

Інтерцептор атоматично:

1. Додає токен Authorization: Bearer<token> до кожного запиту
2. Перевіряє, чи доступний токен, і за його відсутності перенаправляє користувача на сторінку логіну.

На рівні маршрутизації реалізовано кілька guard-ів:

1.AuthGuard – перевіряє, чи користувач авторизований. Якщо ні – доступи до маршруту заборонений, і користувач перенаправляється до /login

2.TeacherGuars – перевіряє, чи користувач має роль teacher. Якщо роль інша або не визначена, доступ до маршруту забороняється.

Guard-и працюють через окремі сервіси авторизації, які зберігають і транслюють стан користувача (через BechaviorSubject<boolean>), що дозволяє динамічно реагувати на зміну статусу вхід\вихід.

Система ролей побудовано на простій, але гнучкій моделі. Після реєстрації роль користувача записується окремо від облікового запису – у Firebase Realtime Database. Це дає змогу легко розширювати функціональність у майбутньому (наприклад, додати роль "admin").

Дані про роль користувача підтягуються одразу після логіну, разом із профілем. Ці дані зберігаються у клієнтському сервісі (UserService) і розповсюджуються через реактивні потоки (BehaviorSubject<Role>). Це забезпечує централізоване управління доступом і дозволяє швидко реагувати на зміну ролей у режимі реального часу.

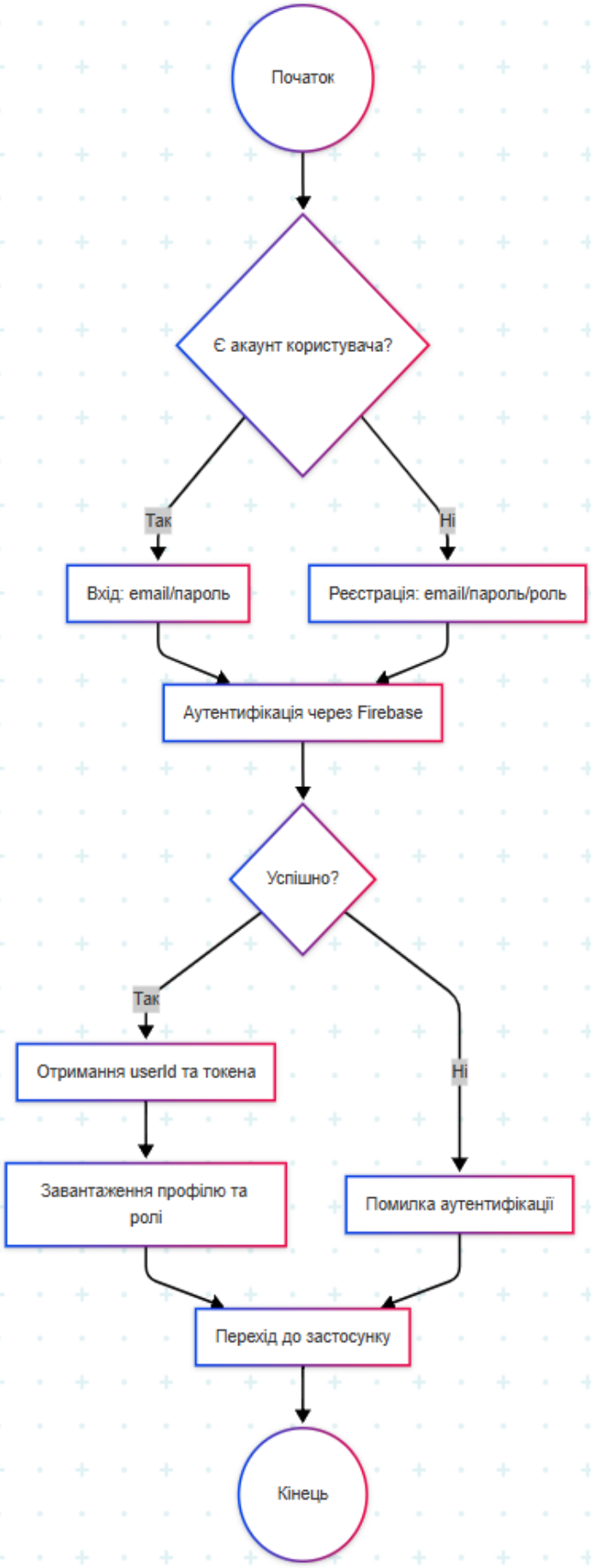



Рисунок 1.10 Реєстрація та вхід корситувача

Компоненти та guard-и підписані на цей стрім, тому автоматично реагують на зміну ролі (наприклад, після переключення користувача).

На клієнт роль впливає на доступність певного функціоналу. Наприклад:

1.Відображення кнопок „створити курс” або „редагувати курс” – лише дял teacher.

2.Доступ до сторінки управління курсами – лише для teacher.

3.Можливість додавати курс до списку – для всіх ролей.

На стороні сервера безпека підтримується через Firebase Security Rules, де прописано які гілки можуть читати/записувати тільки власники або користувачі певної ролі. Наприклад:

```
"rules": {
  "courses": {
    ".read": "auth != null",
    ".write": "auth != null && root.child('userRoles').child(auth.uid).val() === 'teacher'"
  }
}
```

Таким чином, навіть якщо користувач спробує змінити напряму через API, без відповідної ролі це не буде дозволено.

### 1.3.4 Тестування застосунку

Для автоматизованого тестування всієї взаємодії користувача з додатком використовується Cypress. Тести запускаються у реальному браузері та перевіряють повний сценарій — від відкриття сторінки до перевірки стану DOM.

Основні сценарії, що покривається E2E-тестами.

1.Реєстрація нового користувача (включно з вибором ролі, валідацією, повідомленням про успішну реєстрацію).

2.Вхід у систему з валідними та невалідними даними.

3.Робота з курсами: створення, редагування, видалення , перевірка доступності для різних ролей.

4. Взаємодія з профілем користувача: оновлення інформації, вибір аватара, збереження змін.

5.Захист даних: перевірка що student не може  отримати доступ до сторінки для вчителів.

6.Правильність навігації між сторінками, перенаправлення при помилках авторизації

Для кожного тесту Cypress перевіряє як стан URL-адреси, так і зміни в DOM-елементах (рисунок 1.9).

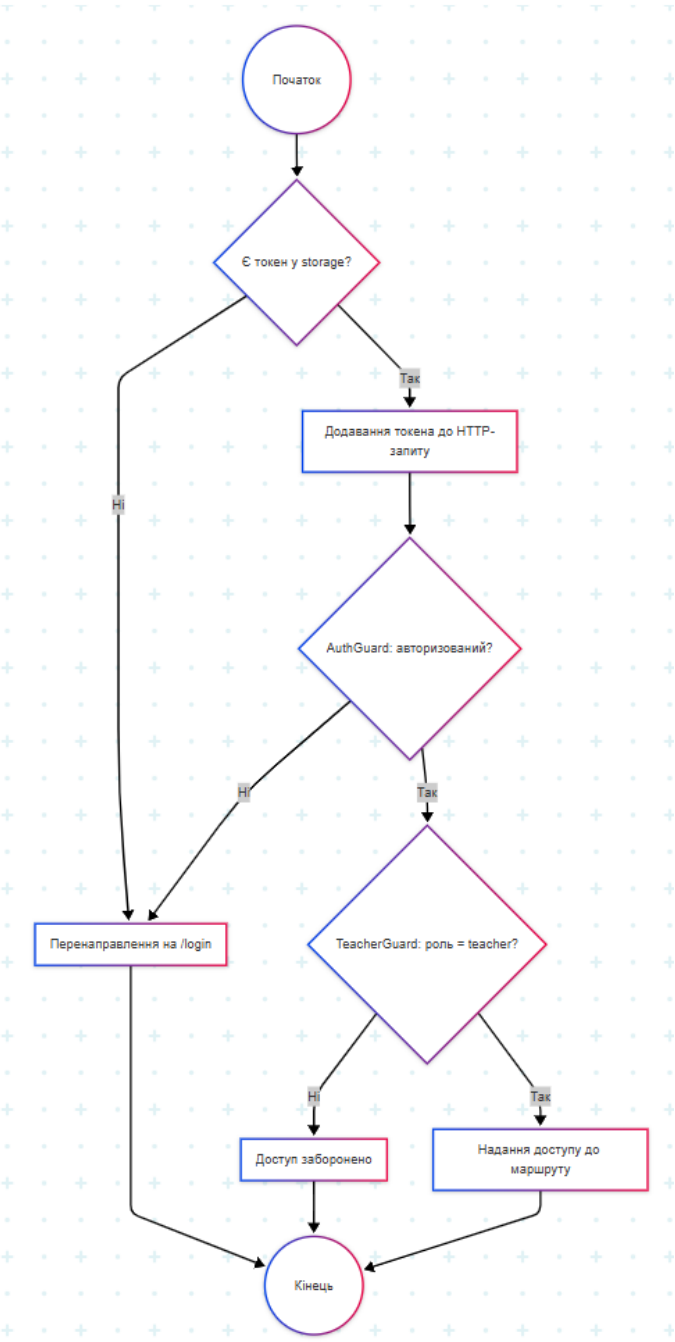



Рисунок 1.11 авторизація при доступі до маршруту

## 1.4 Оптимізація та безпека

### 1.4.1 Лінива загрузка (Lazy Loading) та оптимізація бандлів

Lazy Loading

Для розділення додатку на ізольовані частини та зменшення початкового навантаження використовується механізм Lazy Loading. Основні маршрути додатку (наприклад, /auth, /dashboard, /courses, /profile) розділено на окремі модулі з власними роутерами. Кожен з модулів підключається лише в момент переходу користувача за відповідним маршрутом.

Lazy-модулі оголошуються в основному маршрутизаторі через loadChildren, використовуючи синтаксис динамічного імпорту:

```
{
  path: 'courses',
  loadChildren: () =>
    import('./modules/courses/courses.module').then(m => m.CoursesModule)
}
```

Кожен модуль має власну структуру компонентів, сервісів, маршрутів, що дозволяє ізолювати залежності й спростити масштабування. Також забезпечено передзавантаження найважливіших модулів через PreloadAllModules, що підвищує швидкість при навігації.

Оптимізація бандлів

Angular CLI автоматично виконує оптимізацію під час продакшн-збірки.

Основні етапи:

- Мінімізація: скорочення JavaScript- і CSS-файлів, видалення пробілів, коментарів, спрощення виразів.
- Tree-shaking: видалення не використаного коду з бібліотек і компонентів.
- Compression: готові файли зменшуються за допомогою gzip або brotli (налаштовується у Firebase Hosting або вручну через CI/CD).
- Source maps не включаються у production-збірку для зменшення обсягу та приховування логіки коду.

Для Angular Material використовується асинхронне підключення компонентів, що дозволяє включати лише ті частини бібліотеки, які реально використовуються у відповідному модулі. Анімації підключаються через окремий модуль (BrowserAnimationsModule або NoopAnimationsModule), і можуть бути винесені до lazy-модуля.

Також оптимізовано завантаження зображень і шрифтів — використовуються сучасні формати (webp), відкладене завантаження (loading="lazy"), кешування через Firebase Hosting.

### 1.4.2 База даних (CORS, HTTPS, безпека Firebase)

Перегляньте Angular для бази даних у реальному часі Firebase та автентифікації Firebase, щоб дізнатися про викрадення протоколу HTTPS. Це стосується як запитів до даних, так і дій автентифікації, авторизації, роботи, профів, ролей, курсів.

Доступ до ресурсів Firebase обмежується через конфігурацію CORS. Налаштування неможливе лише запити з дозволених доменів, визначених у проекті Firebase. Небажані або неавторизовані запити з інших джерел блокуються автоматично. Клієнтські запити з Angular-додатком проходять домену, типу перевірки запиту та заголовків перед обробкою.

Правила доступу у Firebase

У Firebase Realtime Database оновлено наступного разу. даних.

– Профіль користувача може переглядати та редагувати лише власника.

– Курси редагування контенту авторів, а також його ідентифікатор збігається з автором курсу.

– Поставляється за рахунок власника.

– Дані особистих курсів користувачів ізольовані, кожен користувач працює лише зі своєю голкою даними.

Ці правила перевіряються на сервері для кожного окремого запиту.

Перед завантаженням ресурсів у HTTP автоматично додається знак вдячності. Це реалізується через механізм перехоплення запитів на клієнті.

Токен зберігається в локальному сховище браузера після входу та використовується до моменту його завершення або видалення.

Після завершення терміну дії токен реалізовано автоматичний вихід користувача з додатком, що запобігає доступ до захищених ресурсів із простим ключем.

Чутливі дані, як-от ключі доступу до Firebase, зберігаються в змінних середовищах. Я не знаю, що робити зі сховищем і кодом. клієнтські інструменти. Це виключає можливість витоку критичної інформації через публічний доступ до публічного коду.

### 1.4.3 Розгортання проекту Firebase Hosting

Angular-додаток збирається в production-режимі з використанням команди ng build --configuration=production. Цей режим активує серію оптимізацій, таких як мінімізація JavaScript- та CSS-файлів, видалення невикористаного коду (tree-shaking), оптимізація ресурсів для швидкого завантаження сторінки та Ahead-of-Time компіляція, що значно пришвидшує старт додатку. У результаті формується фінальна збірка в папці dist, повністю готова до деплою на хостинг або сервер.

Розгортання здійснюється за допомогою CLI-інструменту Firebase CLI. Спочатку виконується авторизація через команду firebase login, потім ініціалізація проєкту через firebase init. Після цього у файлі firebase.json вказується шлях до production-збірки Angular-додатку, наприклад:
*"public": "dist/course-project-starting-project/browser",* а також налаштовуються правила ігнорування:
*"ignore": [ "firebase.json", "/.*", "/node_modules/**" ];*
і механізм переписування маршрутів:
*"rewrites": [ { "source": "**", "destination": "/index.html" } ];*

Це дозволяє реалізувати підтримку SPA: будь-який маршрут автоматично буде переписаний на index.html. Завершується деплой командою firebase deploy, після чого додаток стає доступним в Інтернеті за захищеним HTTPS-посиланням.

Firebase Hosting забезпечує безпечне з’єднання через HTTPS, використання глобальної CDN для швидкого доступу до додатку в будь-якому регіоні та zero-downtime deployment, при якому нова версія стає доступною одразу після деплою без зупинки сервісу. Додаток також може бути інтегрований з іншими сервісами екосистеми Firebase, такими як Firebase Functions, Firebase Storage, Firebase Authentication, Google Analytics тощо.

З метою захисту даних користувачів у застосунку реалізовано автоматичну обробку токенів. В Angular використовується HTTP-інтерцептор, який виглядає наступним чином:

```
@Injectable({ providedIn: 'root' })
export class AuthIntercepotrService implements HttpInterceptor {
constructor(private authService: AuthService) {}
intercept(req: HttpRequest<any>, next: HttpHandler) {
return this.authService.user.pipe(
take(1),
exhaustMap((user) => {
if (!user) {
return next.handle(req);
}
const modifiedReq = req.clone({
params: new HttpParams().set('auth', user.token),
});
return next.handle(modifiedReq);
})
);
}
}
```

Інтерцептор перехоплює всі HTTP-запити та автоматично додає токен автентифікованого користувача у параметри запиту. Це забезпечує авторизований доступ до Realtime Database. У разі відсутності користувача запит

надсилається без змін. Така реалізація дозволяє уникнути ручної передачі токена в кожному запиті та централізовано керувати автентифікацією.

Правила безпеки Realtime Database налаштовані у файлі безпеки Firebase наступним чином:

```
"rules": {
".read": "auth != null",
".write": "auth != null"
}
```

Це правило гарантує, що лише авторизовані користувачі можуть читати або записувати дані в базу. Невідомі або неавторизовані запити автоматично блокуються.

Маршрутизація в Angular-додатку реалізована за допомогою RouterModule. Основні маршрути включають редирект з кореня на /courses, а також вкладені маршрути для роботи з курсами. Виглядає конфігурація маршрутизатора наступним чином:

```
const appRoutes: Routes = [
{ path: '', redirectTo: '/courses', pathMatch: 'full' },
{ path: 'courses', component: CoursesComponent, canActivate: [AuthGuard],
children: [
{ path: '', component: CourseStartComponent },
{ path: 'new', component: CourseMakeComponent, canActivate: [TeacherGuard] },
{ path: ':id', component: CourseDetailComponent, resolve: [CourseResolverService]
},
{ path: ':id/edit', component: CourseMakeComponent, resolve:
[CourseResolverService], canActivate: [TeacherGuard] },]}]
```

Доступ до створення та редагування курсів обмежений для ролі викладача через TeacherGuard, тоді як загальні курси доступні усім авторизованим користувачам через AuthGuard. Це дозволяє побудувати гнучку систему доступу до різних розділів застосунку на основі ролей користувачів.

Інтеграція з Firebase також включає можливість збереження профілю користувача. Після автентифікації дані профілю надсилаються в Realtime Database у вузол profiles з використанням PUT-запиту. Код збереження виглядає наступним чином:

```
storeProfile(profile: any) {
this.authService.user.pipe(take(1)).subscribe((user: User) => {
if (!user) return;
this.http
.put(
https://project-dyplom-cyber-security-default-
rtdb.firebaseio.com/profiles/${user.id}.json?auth=${user.token},
profile
)
.subscribe();
});
}
```

Токен користувача додається до URL як параметр auth, що забезпечує авторизовану передачу даних. Така інтеграція дозволяє зберігати й оновлювати інформацію про користувача, включаючи його ім’я, аватар, роль та інші атрибути профілю.

У сукупності, розгортання Angular-додатку на Firebase Hosting забезпечує високу надійність, безпеку, масштабованість та гнучкість. Завдяки інструментам Firebase CLI, можливостям налаштування SPA-маршрутів, захищеному доступу до бази даних, обробці токенів та інтеграції з іншими сервісами платформи, реалізується повноцінна екосистема для розробки та підтримки сучасного веб-застосунку з ефективною архітектурою. Завдяки вбудованій аналітиці можна відстежувати дії користувачів і покращувати функціональність застосунку. Безперервне розгортання спрощує оновлення продукту та впровадження нових можливостей у реальному часі. Усе це робить Firebase зручним і стабільним. (Рисунок 1.10).

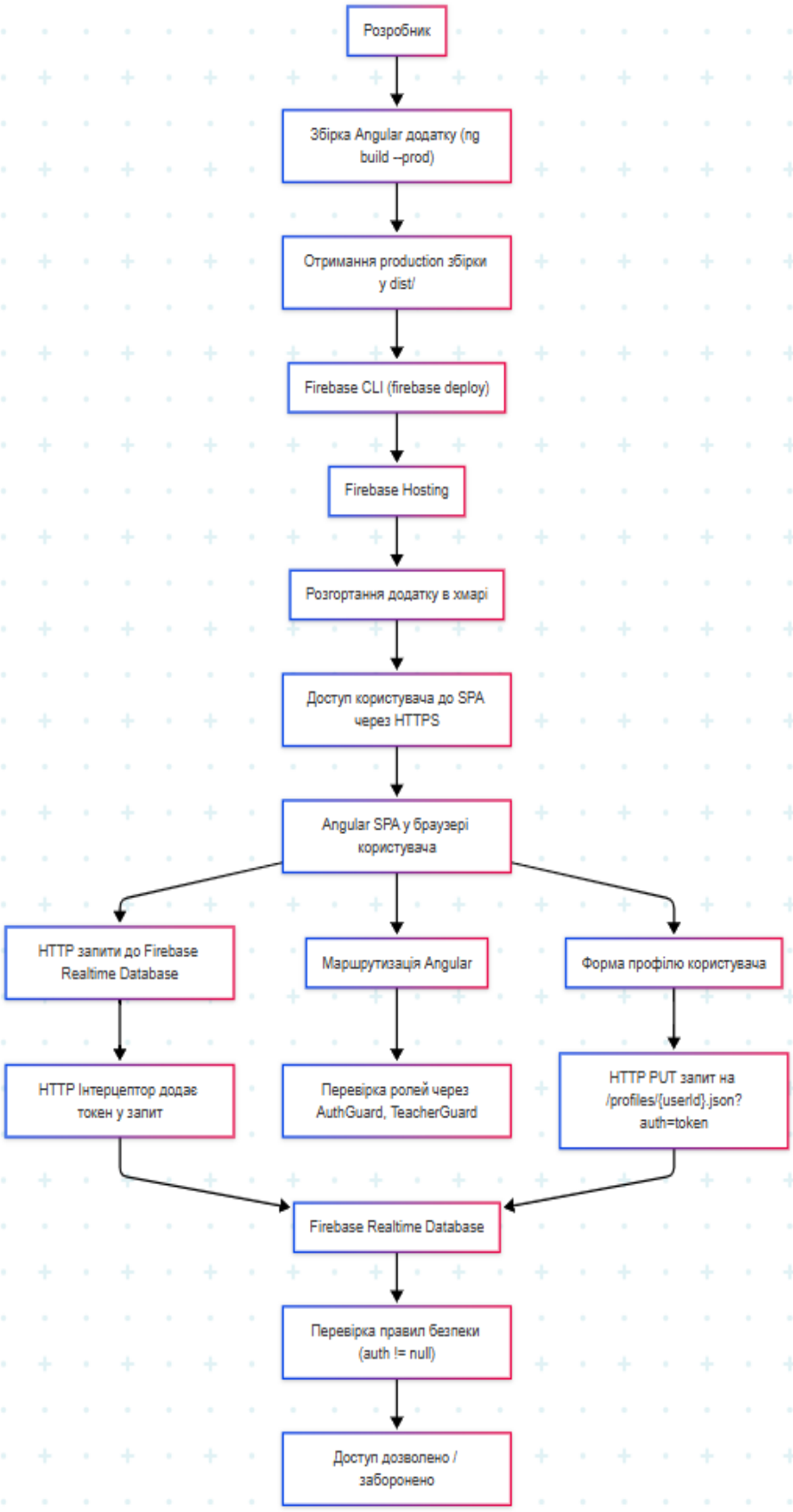



Рисунок 1.10 Процес розгортання Angular SPA на Firebase Hosting з автентифікацією та безпекою

## 1.5 Загальна архітектура проекту

Архітектура розробленого програмного забезпечення ґрунтується на сучасних практиках побудови фронтенд-додатків із залученням Angular як основного фреймворку, а також на хмарних технологіях, зокрема Firebase, що виконує роль бекенд-системи. Основною архітектурною концепцією, що лежить в основі реалізації, є модульність, ізоляція обов'язків між різними частинами застосунку, повторне використання коду та максимальне використання реактивного підходу для роботи з даними та подіями. Згідно з принципами Single Responsibility та Separation of Concerns, кожен структурний елемент, зокрема сервіси, компоненти, гварди, інтерцептори, виконує лише чітко визначений обсяг функцій, уникаючи дублювання логіки чи непотрібних залежностей.

Уся система умовно розділяється на дві основні частини: клієнтську і серверну. Клієнтська частина — це Single Page Application, створена на базі Angular, що відповідає за відображення інтерфейсу, обробку користувацьких подій, реалізацію навігації за допомогою вбудованого механізму маршрутизації, а також взаємодію з сервісами, які інкапсулюють бізнес-логіку, управління станом та HTTP-запити. Серверна частина реалізована на основі інфраструктури Firebase, яка надає такі сервіси як автентифікація користувачів (Firebase Authentication), база даних реального часу (Realtime Database) та хостинг з можливістю деплою готового Angular-додатку з підтримкою SPA-навігації через правила rewrites.

З технічного погляду, вся логіка взаємодії між компонентами, сервісами та зовнішніми джерелами даних побудована з використанням бібліотеки RxJS, яка дозволяє реалізувати реактивний підхід до керування станом. Для широкомовного розповсюдження змін використовуються типи Subject та BehaviorSubject, які дозволяють забезпечити оновлення даних у всіх частинах програми, які на них підписані, з максимальною синхронізацією та узгодженістю.

У рамках клієнтської частини особливу роль відіграють компоненти, які відповідають за рендеринг інтерфейсу, взаємодію з користувачем та підписку на потоки даних із сервісів. Наприклад, компонент CourseListComponent підписується на оновлення масиву курсів через Subject, який генерується у CourseService. Таким чином, будь-яка зміна в списку курсів, що відбувається у сервісі, автоматично транслюється в UI без необхідності ручного оновлення. Крім того, компоненти делегують логіку у сервіси, дотримуючись принципу розділення відповідальностей.

Самі ж сервіси становлять ядро бізнес-логіки проєкту. Наприклад, сервіс CourseService виконує функції додавання, редагування, видалення курсів, зберігає поточний список курсів у пам’яті та надає механізм для реактивного його оновлення. Також існує ShoppingCoursesService, який оперує особистими курсами користувача та обраними секціями, зберігаючи їх у внутрішньому стані та транслюючи зміни через відповідні Subject. Усі запити до серверної частини інкапсульовані у DataStorageService, який, у свою чергу, взаємодіє з AuthService для отримання токену доступу. Відповідно, усі HTTP-запити до Firebase відбуваються з врахуванням авторизації, що є важливим моментом з точки зору безпеки.

Сервіс AuthService реалізує логіку авторизації користувача, використовуючи Firebase Authentication REST API, і підтримує збереження стану користувача у вигляді BehaviorSubject, що дозволяє іншим частинам додатку (наприклад, гвардам) у будь-який момент отримати актуальну інформацію про статус входу або роль користувача.

Особливе місце в архітектурі займають гварди (guards), які відповідають за безпеку навігації. Наприклад, AuthGuard перевіряє, чи користувач взагалі авторизований, тоді як TeacherGuard — чи має він роль викладача. Обидва гварди реалізовані як сервіси, що повертають Observable<boolean | UrlTree>, що дозволяє Angular приймати рішення про дозвіл або блокування переходу до певного маршруту на основі логіки, інкапсульованої в AuthService.

Ще одним важливим компонентом архітектури є інтерцептори, зокрема AuthInterceptor, який автоматично додає токен авторизації до кожного вихідного HTTP-запиту. Таким чином, клієнт не має потреби вручну додавати заголовки чи параметри до запитів — усе це централізовано обробляється на рівні інтерцептора, що значно спрощує підтримку коду й дозволяє уникнути дублювання логіки.

Крім цього, в архітектурі передбачено можливість створення власних директив, які дозволяють розширювати поведінку HTML-елементів. Прикладом є DropdownDirective, що додає можливість створювати кастомні випадаючі меню без потреби повторювати одну й ту ж логіку в кількох компонентах.

Firebase у даному випадку відіграє роль не просто бази даних, а повноцінної серверної платформи, що забезпечує зберігання та синхронізацію даних, керування правами доступу через правила безпеки, автентифікацію користувачів, а також можливість розгортання клієнтського додатку на хостингу з підтримкою SPA-навігації. Завдяки використанню правил доступу .read: *"auth != null" ma .write: "auth != null*", унеможливлюється доступ до критичних даних без проходження процедури входу.

На рівні структури проєкту все об’єднано в головний модуль — AppModule, який відповідає за початкову ініціалізацію застосунку. У деклараціях цього модуля вказано всі основні компоненти, включно з AppComponent, HeaderComponent, CoursesComponent та іншими, а в секції imports — усі допоміжні модулі, необхідні для роботи, зокрема BrowserModule, AppRoutingModule, HttpClientModule тощо. Окремо варто згадати секцію providers, у якій підключаються сервіси, зокрема CourseService, та інтерцептори, як-от AuthInterceptor, що надається через HTTP_INTERCEPTORS з параметром multi: true.

Таким чином, реалізована архітектура відповідає найкращим практикам побудови сучасних Angular-додатків, забезпечує високу гнучкість, масштабованість, розділення відповідальностей і зручність для подальшого тестування, супроводу та розширення. Усі частини системи працюють

узгоджено, завдяки чіткому дотриманню принципів реактивного програмування та максимальної ізоляції логіки, що дозволяє ефективно адаптувати проєкт під нові вимоги в майбутньому.

### 1.5.1 Основні компоненти

1.HeaderComponent

Цей компонент відповідає за головну навігаційну панель, яка відображається на більшості сторінок після входу користувача. Він забезпечує швидкий доступ до основних розділів застосунку — курсів, профілю, кнопки для входу та виходу з облікового запису. Залежно від того, чи авторизований користувач, змінюється відображення елементів: замість кнопки входу показується кнопка виходу, а також іконка профілю.

2.CoursesComponent

Це контейнерний компонент, який не відображає вміст напряму, а містить у собі дочірні компоненти, що відповідають за роботу з навчальними курсами. Його основне завдання — керувати маршрутизацією всередині модуля курсів. Він виступає як загальна оболонка для відображення списку курсів, деталей курсу, редагування тощо. Це компонент-обгортка, що структурує логіку курсового модуля.

3.CourseListComponent

Компонент, який відповідає за відображення повного списку доступних курсів. Він показує курси у вигляді списку або сітки, де кожен елемент — це окрема одиниця навчального матеріалу з короткою інформацією. Якщо користувач є викладачем, він також бачить кнопку для створення нового курсу. Компонент дає змогу обрати курс і перейти до детального перегляду.

4.CourseDetailComponent

Цей компонент відображає детальну інформацію про окремий курс. Коли користувач обирає певний курс зі списку, він потрапляє на сторінку з повним описом курсу, розділами, автором, інформацією про тривалість, кількістю учасників. Користувач може приєднати курс до свого кабінету, а викладач може редагувати або видаляти курс, якщо він є його автором.

5.CourseMakeComponent

Компонент для створення або редагування курсу. Доступний лише користувачам із роллю викладача. Дає змогу ввести заголовок, опис, розділи курсу, прикріпити матеріали. Застосовується як для створення нового курсу, так і для редагування вже наявного. Включає логіку перевірки введених даних та збереження курсу в базі даних.

6.ShoppingCoursesComponent

Компонент особистого кабінету користувача, де зібрані всі курси, які він зберіг або почав проходити. Тут можна переглядати прогрес, відкривати розділи курсів, позначати пройдені теми, продовжити навчання. Це місце, де зберігається власна навчальна колекція користувача, і він має змогу працювати з нею у своєму темпі.

7.MyProfileComponent/MyProfileEditComponent

MyProfileComponent дозволяє переглянути дані свого профілю: ім’я, електронну пошту, роль, фото. Є кнопка для переходу до редагування. MyProfileEditComponent дає змогу змінити особисту інформацію — ім’я, аватар, контактні дані, додаткову інформацію. Зміни зберігаються в базі даних. Ці два компоненти працюють разом, забезпечуючи можливість перегляду й редагування особистого профілю.

8.AuthComponent

Єдиний компонент для авторизації та реєстрації. Працює у двох режимах — вхід до системи та створення нового облікового запису. Користувач перемикається між режимами через інтерфейс. Після успішної авторизації генерується токен, а після реєстрації — створюється новий користувач у базі даних з відповідною роллю. Весь процес реалізовано з використанням Firebase або іншого бекенду.

9.LoadingSpinnerComponent

Універсальний компонент, який відповідає за відображення індикатора завантаження. Він з’являється в момент очікування даних — наприклад, при

завантаженні списку курсів, збереженні змін, вході до системи. Допомагає користувачу зрозуміти, що система працює і обробляє запит.

10.MarqueeBannerComponent

Декоративний компонент, який виводить прокручуваний банер з текстом. Може використовуватись для виведення новин, анонсів курсів, оновлень або мотиваційних повідомлень. Виконує переважно естетичну функцію, додає візуальної динаміки інтерфейсу.

### 1.5.2 Сервіси та їх взаємодія

У структурі застосунку важливу роль відіграють сервіси, що реалізують бізнес-логіку, інкапсулюють доступ до даних і забезпечують зв’язки між різними модулями програми. Кожен сервіс має чітко визначене завдання і співпрацює з іншими через методи, потоки подій та реактивні механізми.

CourseService є основним сервісом для управління курсами. Він реалізує повний набір функцій для додавання, редагування та видалення курсів. Зберігає список курсів локально в пам’яті та транслює зміни в режимі реального часу за допомогою потоків типу Subject. Це дозволяє компонентам, що підписані на ці потоки, миттєво оновлювати інтерфейс після будь-яких змін. Крім того, цей сервіс виступає як джерело правди для інших сервісів — зокрема, передає дані до ShoppingCoursesService, який оперує вже персоналізованим контентом користувача.

ShoppingCoursesService обслуговує особисті курси користувача. Він зберігає обрані користувачем секції курсів та прогрес проходження. Коли користувач додає певну секцію до свого навчального простору, сервіс оновлює відповідну колекцію і транслює зміни через реактивний потік. Таким чином реалізується відображення змін у реальному часі на сторінках профілю або особистих курсів.

DataStorageService відповідає за всю логіку взаємодії з віддаленим бекендом, зокрема Firebase. Він виконує HTTP-запити для збереження і завантаження курсів, секцій, профілів. Особливістю цього сервісу є його інтеграція з AuthService при кожному запиті отримується актуальний токен

авторизації, який додається до URL або заголовків запиту. Таким чином гарантується, що всі дії з базою виконуються від імені конкретного користувача з відповідними правами доступу.

AuthService реалізує всю логіку авторизації та автентифікації користувача через Firebase Auth REST API. Після успішного входу в систему зберігається інформація про користувача у вигляді об’єкта, що транслюється через BehaviorSubject. Це дозволяє будь-якому компоненту або сервісу в реальному часі реагувати на зміну стану користувача (наприклад, оновлення інтерфейсу після входу або виходу). Також реалізовано механізм автоматичного входу, що відновлює сесію користувача на основі даних з localStorage.

### 1.5.3 Гварди

Для забезпечення захисту доступу до певних сторінок застосунку використовуються Angular-гварди — спеціальні сервіси, які перехоплюють навігацію. Основні два типи гвардів, які використовуються:
AuthGuard — перевіряє, чи авторизований користувач. Якщо ні — перенаправляє його на сторінку входу. Таким чином жодна захищена сторінка не може бути переглянута без попередньої автентифікації.

TeacherGuard — додатковий рівень захисту, який перевіряє роль користувача. Якщо користувач не є викладачем, він не отримає доступ до сторінок створення чи редагування курсів. Це дозволяє обмежити можливість змінювати вміст тільки для авторизованих викладачів.

### 1.5.4 Маршрутизація

Уся маршрутизація налаштована в окремому модулі AppRoutingModule. Структура маршрутів ієрархічна та підтримує вкладеність. Для кожного шляху задається відповідний компонент, а також умови доступу у вигляді гвардів. Наприклад, розділ з курсами має власний батьківський маршрут, у якому вкладені маршрути для перегляду детальної інформації про курс або його створення. Це дозволяє реалізувати більш складну логіку інтерфейсу та

покращити контроль над навігацією, забезпечуючи динамічну зміну контенту без повного перезавантаження сторінки.

Також реалізовано перенаправлення (redirect) для обробки невалідних або застарілих шляхів, що запобігає помилкам навігації. Активне посилання в меню автоматично підсвічується за допомогою директиви routerLinkActive, що покращує зручність користування. Для оптимізації продуктивності використовується lazy loading — модулі завантажуються лише тоді, коли користувач переходить до відповідного маршруту. Параметри маршруту дозволяють передавати ідентифікатори або додаткові дані без необхідності зберігати їх у глобальному стані. Також обробляються route-помилки (наприклад, 404), забезпечуючи користувача інформативними повідомленнями при недоступності сторінки.

### 1.5.5 Реактивність і керування станом

Весь стан застосунку керується за допомогою бібліотеки RxJS, що дозволяє організувати реактивні потоки даних і ефективно оновлювати інтерфейс. Основні типи використовуваних потоків:

- Subject використовується для трансляції подій, таких як додавання курсу або секції.
- BehaviorSubject дозволяє зберігати поточний стан (наприклад, авторизованого користувача) і миттєво передавати його новим підписникам.

Таке використання дозволяє досягти реактивного зв’язку між сервісами і компонентами, забезпечуючи синхронізацію інтерфейсу з даними у реальному часі. Наприклад, якщо курс додається у CourseService, це миттєво відображається в усіх компонентах, які підписані на потік coursesChanged.

### 1.5.6 Інтеграція з Firebase

- Firebase використовується як основний хмарний бекенд для застосунку. Інтеграція охоплює кілька ключових напрямів:

– Authentication для входу та реєстрації користувачів через REST API. Токен, отриманий після входу, використовується для автентифікації при доступі до бази даних.
– Realtime Database зберігає всі дані: курси, секції, профілі, прогрес користувача. Завдяки гнучким правилам безпеки можна чітко обмежити доступ до даних на основі ролей та автентифікації.
– Firebase Hosting використовується для розгортання SPA-застосунку на Angular. Налаштування rewrites дозволяє завжди повертати index.html для підтримки клієнтської маршрутизації навіть при прямому переході за посиланням. Також використано Firebase Security Rules для реалізації багаторівневого контролю доступу до даних у реальному часі.

### 1.5.7 Безпека

Безпека реалізована на кількох рівнях. По-перше, Angular-гварди фільтрують доступ до маршруту ще до того, як компонент буде завантажений. По-друге, всі запити до Firebase супроводжуються токеном, який отримується після входу в систему. По-третє, в Realtime Database Firebase налаштовані правила доступу, що дозволяють читати або змінювати дані лише авторизованим користувачам. Наприклад, тільки ті, хто увійшов до системи, можуть створювати або редагувати курси.

Крім того, усі передані дані шифруються за допомогою HTTPS, що унеможливлює перехоплення інформації третім особам. Регулярно перевіряється активність користувачів для виявлення підозрілих дій, таких як багаторазові невдалі спроби входу. Система має обмеження на сесії — у разі тривалої бездіяльності виконується автоматичний вихід. Також реалізовано захист від XSS-атак шляхом використання Angular-сантайзерів, які очищують небезпечний HTML. Всі дії користувача логуються, що дозволяє відслідковувати історію змін і швидко реагувати на потенційні загрози.
Для додаткового захисту використовується перевірка ролей користувача, яка забезпечує розмежування прав доступу до різних функцій системи в залежності від контексту використання**.**

### 1.5.8 UI/UX та дизайн

Інтерфейс створений із використанням Angular Material, що забезпечує сучасні, адаптивні UI-компоненти, такі як кнопки, діалоги, інпут-форми та вкладки. Також застосовується Bootstrap для верстки і гнучкої layout-сітки.

Використано ретельне тестування на різних пристроях, що гарантує коректну роботу інтерфейсу на всіх екранах. Для додання індивідуальності використано кастомні стилі: це drag-and-drop реалізація у списках, інтерактивні прев'ю зображень, кольорова схема тощо. Взаємодія з користувачем спрощена за допомогою анімацій та плавних переходів, що робить роботу з додатком ще зручнішою. UX-компоненти реалізовані з урахуванням зручності користувача: спінери відображають стан завантаження, модальні вікна запитують підтвердження дій, поля форм мають валідацію в режимі реального часу, щоб зменшити кількість помилок і покращити загальний досвід. Додатково, система сповіщень інтегрована для своєчасного інформування користувача про стан системи або будь-які зміни.

### 1.5.9 Тестування

Тестування застосунку охоплює як юніт-рівень, так і end-to-end перевірки. Для юніт-тестування використовується зв'язка Jasmine + Karma, що дозволяє протестувати окремі компоненти, сервіси, гварди на коректність логіки та обробку подій.

Для e2e-тестування застосовано Cypress — сучасний фреймворк для автоматизації користувацьких сценаріїв. Наприклад, перевіряється можливість входу користувача в систему, створення нового курсу викладачем, додавання секції, а також обробка випадків несанкціонованого доступу. Це гарантує, що основні функції програми залишаються стабільними при змінах у коді.

Також проводиться перевірка правильності відображення елементів інтерфейсу та навігації між сторінками. Тести охоплюють як позитивні, так і негативні сценарії взаємодії з додатком. Це дозволяє мінімізувати ризики помилок у продакшн-версії та покращити загальну надійність програмного забезпечення.

# 2 ЕКОНОМІЧНИЙ РОЗДІЛ

## 2.1. Резюме

В даному дипломному проекті було розроблено програмний продукт розробка програмного забезпечення для on-line навчання основам кібербезепки.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

## 2.2 Розрахунок ціни програмного продукту нормативним методом

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1. Каталог аналогів

| Найменування ПП | Обсяг функції ПП – $V_o$, усл. Машинних командах. |
|---|---|
| 3. ПП СУБД | 2500 - 9800 |

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі таблиці 2.2

Таблиця 2.2. Vo в умовних машинних командах

| Обсяг ПП, тис.умов.машинних команд | Норма часу, люд/год |
|---|---|
| 2.00 | 244 |

На підставі отриманого значення визначається укрупнена норма часу на розробку аналога програмного забезпечення, яка коректується поправочним

коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8:

$T^a$ = 244 х 0,8 = 195,2 (люд/годин)

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

- Розробка технічного завдання

$$T_{ТЗ} = T^a p * L_1 * K_H \quad (1)$$

- Розробка технічного проекту

$$T_{ТП} = T^a p * L_2 * K_H \quad (2)$$

- Розробка робочого проекту

$$T_{РП} = T^a p * L_3 * K_H * K_T \quad (3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (таблиця 2.2);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (таблиця 2.3);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (таблиця 2.4).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості

| Код стадії | Ступінь новизни | | |
|---|---|---|---|
| | А | Б | В |
| ТЗ ($L_1$) | 0,15 | 0,12 | 0,12 |
| ТП ($L_2$) | 0,16 | 0,15 | 0,11 |
| РП ($L_3$) | 0,55 | 0,58 | 0,61 |

Таблиця 2.4. Значення поправочного коефіцієнта

| Код ступеня новизни | Ступінь новизни | Значення $К_Н$ |
|---|---|---|
| А | Принципово нові ПП | 1,75 – 1,2 |
| Б | ПП – розвиток визначеного параметричного ряду | 1,0 – 0,8 |
| В | ПП маючий аналог | 0,7 |

Таблиця 2.5. Значення коефіцієнта ступеня використання

| Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, % | Значення $К_Т$ |
|---|---|
| 60 і вище | 0,6 |
| 40-60 | 0,7 |
| 20-40 | 0,8 |
| До 20 | 0,9 |

Розрахунок трудомісткості по кожному етапу окремо:

- Трудомісткість технічного завдання

  $Т_{ТЗ} = Т^а р * L_1 * K_Н = 195,2 * 0,12 * 0,7 = 16,4$ (люд/годин)

- Трудомісткість розробки технічного проекту

  $Т_{ТП} = Т^а р * L_2 * K_Н = 195,2 * 0,11 * 0,7 = 15$ (люд/годин)

- Трудомісткість розробки робочого проекту

  $Т_{РП} = Т^а р * L_3 * K_Н * K_Т = 195,2 * 0,61 * 0,7 * 0,8 = 66,6$ (люд/годин)

Для подальших розрахунків необхідно визначити кількість папера, витраченого на кожен етап. $N_{тз}$= 3 (стр), $N_{тп}$= 19 (стр), $N_{рп}$= 28 (стр), $N_{пз}$= 15 (стр) – технічне завдання, розробка технічного проекту, розробка робочого проекту, пояснювальна записка відповідно. Розрахунок зведений у таблицю 2.6

Таблиця 2.6. Розрахунок трудомісткості ПП

| Найменування етапів | Розрахунок, години. | | |
|---|---|---|---|
| 1 | Розробка ПП | Контроль керівника | Нормоконтроль |
| 1.ТЗ | $Т_{Ртз}$=16,4 | $Т_{кк}=0,7*N_{тз}=0,7*3=2,1$ | $Т_{нк}=0,15*N_{тз}=0,15*3=0,45$ |
| 2.Розробка ТП | $Т_{Ртп}$=15 | $Т_{кк}=0,7 * N_{тп}=0,7*19=13,3$ | $Т_{нк}=0,15*N_{тп}=0,15*19=2,85$ |
| 3.Розробка РП | $Т_{Ррп}$= 66,6 | $Т_{кк}=0,7 * N_{рп}=0,7*28 =$ $=19,6$ | $Т_{нк}=0,15*N_{рп}=0,15*27,5=$ $=4,2$ |
| 4.Розробка пояснювальної записки | $Т_{пз}$=1,5 $N_{пз}=1,5*16=$ $=24,75$ | $Т_{кк}=0,7 * N_{тз}=0,7*15=10,5$ | $Т_{нк}=0,15*N_{пз}=0,15*15=2,25$ |
| Усього, в т.ч.: | $Т_{пп}$=154,75 | | |
| на розробку | $\sum Т_р$=99,5 | | |

Продовження таблиці 2.6

| | | | |
|---|---|---|---|
| - контроль керівника | | $\sum T_{хк}=45,5$ | |
| нормоконтроль | | | $\sum T_{нк}=9,75$ |

### 2.3. Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку програмного забезпечення. Розрахунок основної заробітної плати виконавців приведений у таблиці 6.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2025 року – 8000 гривень; мінімальну погодинну тарифну ставку – 46 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

| Найменування робіт | Трудомісткість робіт, роб.години | Годинна тарифна ставка,грн.. | Розрахунок, грн. |
|---|---|---|---|
| 1.Розробка ПП | 99,5 | 83,00 | 8258,5 |
| 2.Контроль керівника | 45,5 | 55,00 | 2502,5 |
| 3.Нормоконт-роль | 9,75 | 47,50 | 463,125 |
| Усього (З о | - | - | ∑Зо = 11221,125 |

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

| Найменування матеріальних витрат | Тип, модель | Кількість, шт | Ціна одиниці, грн. | Вартість, грн. |
|---|---|---|---|---|
| Папір А1 | - | - | - | - |
| Папір А4 | арткуш | 65 | 1,5 | 97,5 |
| Разом | - | - | - | $В_{м\,і} = 97,5$ |
| Транспортно заготівельні Витрати 10% | | | | $В_{тр\_з} = 0,1\times В_{м1} = 9,75$ |
| Усього | | | | $В_{м}=В_{м1}+В_{тр\_з}= 106,80$ |

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

| Стаття витрат | Значення, грн. | Формула розрахунку |
|---|---|---|
| 1. Матеріали | 106,80 | В м (див. табл. 2.8) |
| 2. Основна заробітна плата | 11221,125 | З о (див. табл. 2.7) |
| 3.Додаткова заробітна плата | 1683,6 | *Зд = 0,15× Зо =0,15*11221,125=1683,6 |
| 4.Відрахування до єдиного фонду соціального внеску | 2839 | $В_{є.с.в.}$=0,22× ($З_о$ + $З_д$) =0,22*(11221,125+1683,6)=2839 |
| 5. Накладні витрати | 3 366 | $В_{нак.}$ = 0,3*$З_о$ = 0,3*11221,125 |
| 6. Повна собівартість | 19516,525 | $С_{пов}$= $В_м$ +$З_о$ + $З_д$ +$В_{є.с.в.}$ + $В_{нак.}$= 106,80+11221,125+1683,6+2839+3 366 |

Розмір прибутку, що включається в ціну, визначається по наступній формулі:

П=(Спов*Р)/100 = (*19516,5) / 100 = 1951,6525 (грн), (2.6)

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

Цо =Спов + П= 19516,525+ 1951,6525 = 21484,1050 (грн); (2.7)

Податок на додану вартість визначається по наступній формулі:

ПДВ = 0.2 * Цо = 0,2 * 19516,525= 3903,305 (грн); (2.8)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

Цр =Цо + ПДВ= 19516,525+ 3903,305 = 23419,83 (грн); (2.9)

# 3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ

Організація охорони праці та безпеки на виробництві завжди була дуже важлива для збереження здоров'я працівників. З метою забезпечення безпечних умов праці спеціально створюються алгоритми дій та інструкції з охорони праці щодо виконання робочих завдань, які містять чітко визначені правила та послідовності дій. Кожен працівник повинен бути ознайомлений з цими інструкціями ще до початку виконання своїх обов'язків, проходити відповідні інструктажі.

## 3.1. Основні шкідливі чинники, що впливають на програміста

Одним із основних шкідливих чинників, що впливають на працівників, які здійснюють професійну діяльність із застосуванням комп'ютерної техніки, є статичне навантаження на опорно-руховий апарат. Тривале перебування у сидячому положенні протягом робочого дня, що у середньому становить від 7 до 10 годин, негативно позначається на функціональному стані м'язів, порушує кровообіг і спричиняє деформаційні зміни хребетного стовпа. Нерухоме положення тіла зумовлює надмірне навантаження на окремі групи м'язів, порушує кровообіг. Нерухоме положення тіла зумовлює надмірне навантаження на окремі групи м'язів, зокрема в ділянці шийного та грудного відділів хребта, плечового поясу, попереку. Хронічне м'язове перенапруження, відсутність регулярної зміни положення тіла та активного розслаблення м'язів можуть спричинити розвиток різноманітних захворювань.

Наступним фактором, що суттєво впливає на стан здоров'я працівника, є зорове навантаження. У процесі програмування, обробки графічної інформації та читання з екрана монітора працівник тривалий час фокусує зір на об'єктах, розташованих на близькій відстані. Найбільш розповсюдженим наслідком є синдром комп'ютерного зору, що проявляється у вигляді наступних симптомів: сухість очей, сльозотеча, біль в очах. Виникнення цих симптомів пов'язане зі зниженням частоти кліпання, надмірною яскравістю екрану. Також це може

спричинити зниження гостроти зору та підвищенну втому очей, що негативно впливає на загальну працездатність.

### 3.2. Організація робочого місця

Організація робочого місця працівника, що здійснює професійну діяльність за персональним комп'ютером, має відповідати вимогам ергономіки, техніки безпеки та санітарно-гігієнічним нормативам. Незалежно організоване робоче місце сприяє зниженню впливу шкідливих чинників,, зменшенню ризику виникнення професійних захворювань і підвищенню продуктивності праці.
Робоче місце повинно бути укомплектоване меблями та обладнанням, конструкція яких забезпечує можливість індивідуального регулювання відповідно до антропометричних особливостей працівника. Висота робочого столу має становити від 680 до 760 мм, а глибина стільниці – не менше 800мм для розміщення монітора на рекомендованій від очей відстані. Офісне крісло має мати регульовану висоту сидіння, спинку з підтримкою поперекового відділу хребта, підлокітники, механізм зміни кута нахилу спинки та обертання. Конструкція крісла має забезпечувати підтримку ніг у положенні, за якого колінні суглоби утворюють кут 90-110 градусів, а ступні повністю спираються на підлогу або спеціальну підставку.

Розташування монітора має забезпечувати кут зору в межах 30-35 градусів нижче горизонтальної лінії погляду. Відстань від очей до екрана має бути не меншою за 600 мм. Центр екрана бажано розташувати трохи нижче рівня очей, щоб уникнути надмірного напруження м'язів шиї та очей. Монітор повинен мати чітке зображення без мерехтіння, з достатнім рівнем контрастності та яскравості, що не викликає зорового дискомфорту.

До обов'язкових умов також належить відповідне освітлення. Загальна освітленість у робочій зоні повинна бути в межах 300-500 лк. Робоче місце повинно бути оснащене комбінованим освітленням: загальним (стельовим), та місцевим (настільна лампа). Джерела світла не повинні створювати відблисків на поверхні екрана.

Параметри мікроклімату приміщення мають відповідати встановленим нормам: температура повітря – 21–23°C, відносна вологість – 40–60%. Рекомендується застосовувати системи вентиляції або кондиціювання з можливістю підготовки повітря (очищення, нагріву, охолодження та зволоження/осушення).

Для підтримки оптимального мікроклімату найкраще підходять:

-Припливно-витяжні вентиляційні системи з рекуперацією тепла та фільтрацією повітря.

-Системи кондиціювання (канальні, VRV/VRF або прецизійні), здатні контролювати температуру та вологість.

Додатково можна використовувати зволожувачі або осушувачі повітря, якщо основна система не забезпечує необхідний рівень вологості.

Електробезпека є критично важливим елементом забезпечення безпечних умов праці для працівників, які використовують комп'ютерну техніку. Основні небезпечні фактори включають:

1.Ризик ураження електричним струмом:

-Може виникнути при контакті з несправними елементами живлення

-Небезпека підвищується при пошкодженні ізоляції кабелів

-Особливо небезпечні роботи з відкритими корпусами обладнання

2.Пожежна небезпека:

-Короткі замикання в електромережах

-Перегрів обладнання при тривалій експлуатації

-Несправності блоків живлення

3.Інші ризики:

-Вихід з ладу мережевого обладнання

-Пошкодження даних через перепади напруги

-Електромагнітне випромінювання

Заходи з електробезпеки:

1.Вимоги до обладнання:

-Усі пристрої повинні мати справний стан

-Обов'язкова наявність заводських маркувань

-Наявність інструкцій з експлуатації

-Відповідність чинним стандартам (ДСТУ, ГОСТ)

2.Організаційні заходи:

-Регулярні технічні огляди обладнання

-Систематичні перевірки стану електропроводки

-Навчання персоналу правилам електробезпеки

3.Технічні заходи захисту:

-Використання стабілізаторів напруги

-Застосування джерел безперебійного живлення

-Монтаж систем заземлення

-Встановлення пристроїв захитного відключення

4.Додаткові вимоги для приміщень з ПК:

-Заборона самостійного ремонту під напругою

-Обмеження доступу до електрощитів

-Дотримання норм щільності розміщення обладнання

-Контроль температури та вентиляції

Персональні заходи захисту

-Використання ізольованого інструменту при роботах

-Застосування діелектричних килимків

-Дотримання правил особистої гігієни (сухі руки при роботі)

Важливо пам'ятати, що навіть у офісних приміщеннях з комп'ютерною технікою необхідно дотримуватись усіх норм електробезпеки для запобігання аварійним ситуаціям та травматизму.

### 3.3 Пожежна безпека

Пожежна безпека є невід’ємною складовою загальної системи охорони праці на підприємстві, у тому числі в умовах організації праці програмістів. Незважаючи на відсутність безпосереднього контакту з відкрити джерелами вогню, діяльність пов’язана з експлуатацією значної кількості електронного

обладнання, створює потенційно небезпечні умови щодо виникнення пожежонебезпечних ситуацій.

З метою запобігання виникненню пожежі всі електричні пристрої повинні бути розміщені з дотриманням встановлених нормативів на достатній відстані від легкозаймистих предметів. Всі експлуатовані електротехнічні засоби повинні проходити регулярну перевірку на технічну справність та відповідати чинним стандартам і вимогам електробезпеки.

У кожному робочому приміщенні необхідно передбачити наявність первинних засобів пожежогасіння, зокрема вогнегасників вуглекислотного ($CO_2$) або порошкового типу типу, призначених для ліквідації загоряння електроустановок, які перебувають під напругою.

Необхідною умовою є наявність схеми евакуації із зазначенням основних і запасних маршрутів виходу, а також послідовності дій у разі виникнення загоряння або задимлення. План евакуації повинен бути розміщений на видному місці та відповідати встановленим вимогам.

# ВИСНОВКИ

У ході виконання дипломного проекту було створено сучасний веб-додаток на тему "Розробка програмного забезпечення для online-навчання основам кібербезпеки", що відповідає вимогам до якості програмного забезпечення та поставленому завданню. Платформа дозволяє інтерактивно й доступно вивчати матеріали з будь-якого пристрою та в будь-який час. Проєкт реалізовано з використанням Angular для фронтенду та Firebase для бекенду, що забезпечує швидкодію, безпеку (через Firebase Security Rules), та масштабованість. Важливими технічними досягненнями стали впровадження Lazy Loading, Tree Shaking і реактивних підходів за допомогою RxJS, що оптимізувало продуктивність інтерфейсу навіть за обмежених мережевих ресурсів.

Функціональність включає створення навчальних курсів, керування ними та профілями користувачів через динамічні форми без потреби в розробниках. Тестування проводилося на всіх рівнях, що підтвердило стабільність та надійність роботи системи під навантаженням. Платформа має економічні переваги: зниження витрат на навчання, гнучкий доступ до матеріалів і можливість масштабування без додаткових витрат. У перспективі передбачається розширення функціоналу, розробка мобільної версії та інтеграція з іншими освітніми системами. Проєкт довів, що створення ефективної освітньої платформи можливе силами однієї особи, і має практичну цінність для реального впровадження.

Під час реалізації проекту були використані сучасні технології: фреймворк Angular для створення зручного інтерфейсу та хмарна платформа Firebase для зберігання даних, аутентифікації та безпеки. Серед ключових технічних рішень — Lazy Loading для завантаження модулів лише за потреби, Tree Shaking для зменшення розміру збірки, та RxJS для реалізації реактивної логіки. Особливу увагу приділено безпеці: завдяки Firebase Security Rules кожен користувач має доступ лише до власних даних. Проведене тестування підтвердило надійність і стабільність роботи навіть за умов навантаження.

# ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ


1. Mozilla Developer Network (MDN). JavaScript and Web Technologies – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org – Дата звернення: 14.05.2025
2. NgRx Documentation. State Management for Angular – [Електронний ресурс]. – Режим доступу: https://ngrx.io/docs – Дата звернення: 25.03.2025
3. RxJS Official Documentation. Reactive Programming with JavaScript – [Електронний ресурс]. – Режим доступу: https://rxjs.dev/guide – Дата звернення: 07.02.2025
4. Google Cloud. Firebase Security Rules – [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/rules – Дата звернення: 19.05.2025
5. Мироненко В. В. Основи веб-розробки: HTML, CSS, JavaScript – [Електронний ресурс]. – Режим доступу: https://library.kname.edu.ua/opac/index.php?url=/notices/index/IdNotice:21401/ – Дата звернення: 28.01.2025
6. Пархоменко О. М. Програмування вебзастосунків засобами Angular – [Електронний ресурс]. – Режим доступу: https://openarchive.nure.ua/handle/document/17419 – Дата звернення: 09.03.2025
7. Коваль С. В. Технології розробки інтернет-застосувань – [Електронний ресурс]. – Режим доступу: https://lpnu.ua/education/majors/subject/DDPGS/6.050101.01/8/2020/ua/full/2/14559 – Дата звернення: 04.05.2025
8. Гриценко А. О. Інформаційна безпека в хмарних сервісах – [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/handle/123456789/41620 – Дата звернення: 16.04.2025
9. Коваленко І. І. Основи тестування програмного забезпечення – [Електронний ресурс]. – Режим доступу: https://card.opu.ua/uk/catalog/book/116218 – Дата звернення: 12.02.2025

# ДОДАТОК А. Лістинги основних класів та сервісів Angular

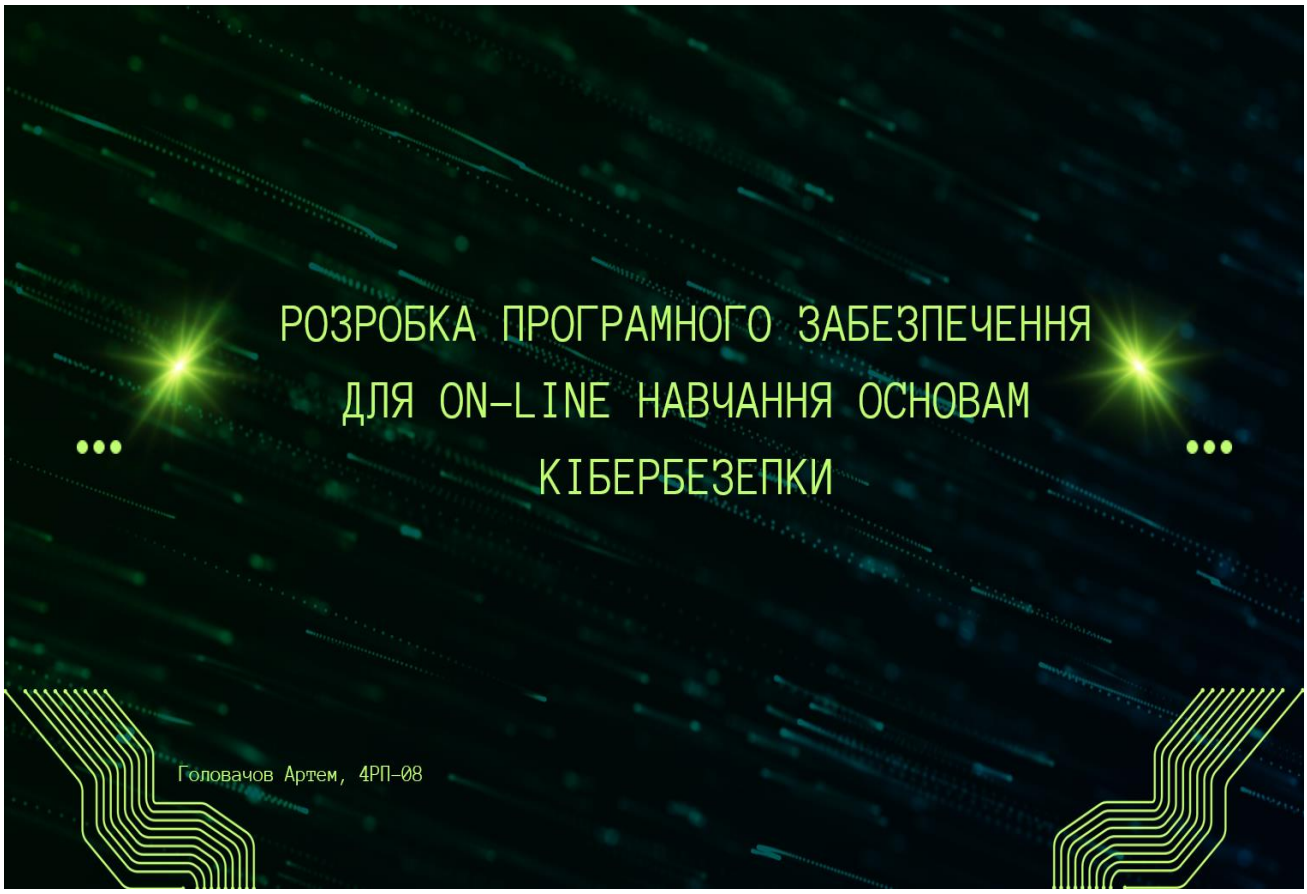
```
/**
 * Головний модуль застосунку
 */
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    CoursesComponent,
    CourseListComponent,
    CourseDetailComponent,
    // ...інші компоненти...
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    AppRoutingModule,
    HttpClientModule,
    // ...Material та інші модулі...
  ],
  providers: [
    provideAnimationsAsync(),
    CourseService,
    ShoppingCoursesService,
    {
      provide: HTTP_INTERCEPTORS,
      useClass: AuthIntercepotrService,
      multi: true,
    },
  ],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

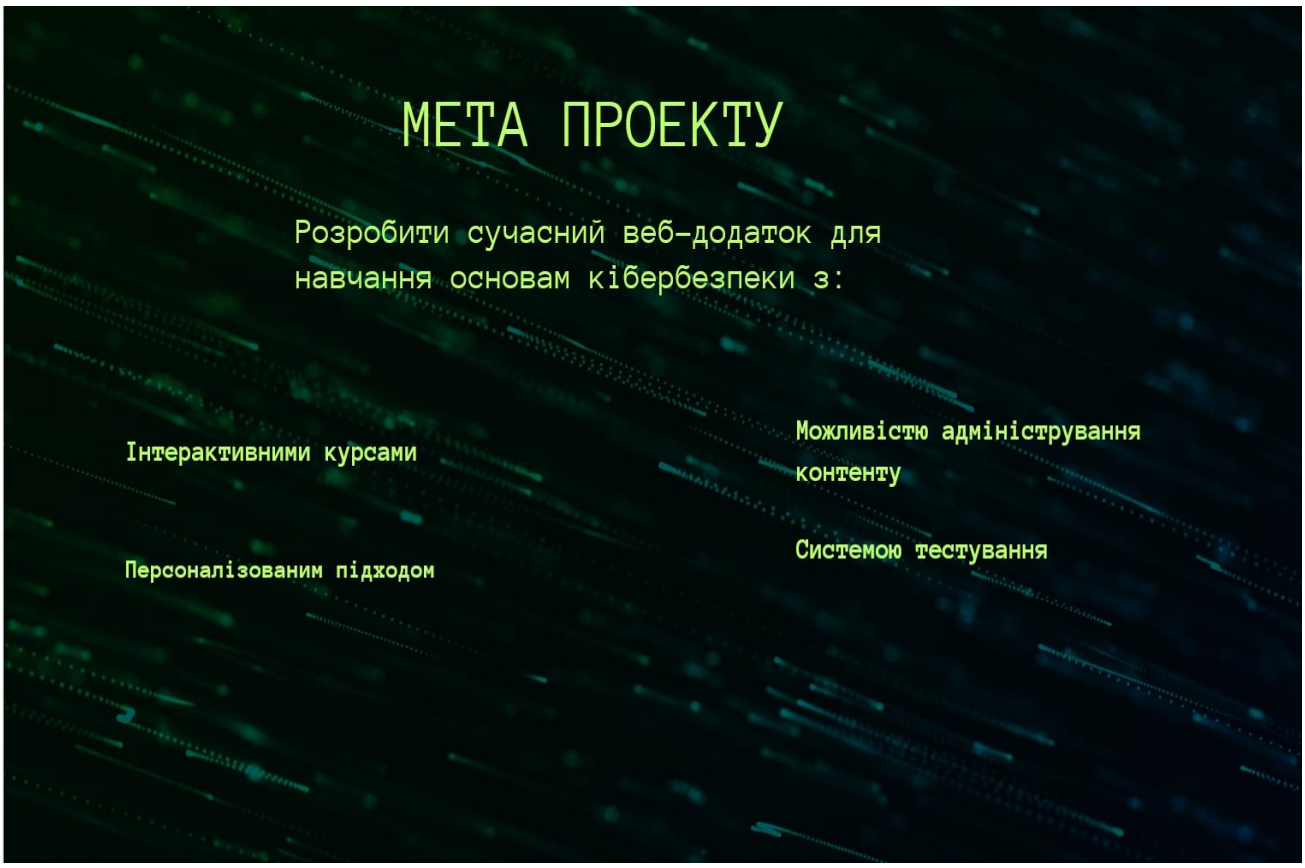
```
/**
 * Головний компонент застосунку
 */
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent implements OnInit {
  constructor(private authService: AuthService) {}

  ngOnInit(): void {
    this.authService.autoLogin();
  }
}
/**
 * Маршрутизація застосунку
 */
const appRoutes: Routes = [
  { path: '', redirectTo: '/courses', pathMatch: 'full' },
  {
    path: 'courses',
    component: CoursesComponent,
    canActivate: [AuthGuard],
    children: [
      { path: '', component: CourseStartComponent },
      { path: 'new', component: CourseMakeComponent, canActivate: [TeacherGuard] },
      { path: ':id', component: CourseDetailComponent, resolve: [CourseResolverService] },
      { path: ':id/edit', component: CourseMakeComponent, resolve: [CourseResolverService],
canActivate: [TeacherGuard] },
    ],
  },
  { path: 'my-courses', component: ShoppingCoursesComponent, canActivate: [AuthGuard] },
  { path: 'about-us', component: AboutUsComponent },
  {
```

```
    path: 'my-profile',
    canActivate: [AuthGuard],
    children: [
      { path: '', component: MyProfileComponent },
      { path: 'edit', component: MyProfileEditComponent },
    ],
  },
  { path: 'auth', component: AuthComponent },
];
/**
 * Сервіс аутентифікації (фрагмент)
 */
@Injectable({ providedIn: 'root' })
export class AuthService {
  user = new BehaviorSubject<User>(null);
  userRole = new BehaviorSubject<string>(null);

  constructor(private http: HttpClient, private router: Router) {}
  signup(email: string, password: string, role: string = 'student') { /* ... */ }
  login(email: string, password: string) { /* ... */ }
  autoLogin() { /* ... */ }
  logout() { /* ... */ }
  fetchUserRole(userId: string, token: string) { /* ... */ }
  saveUserRole(userId: string, role: string, token: string) { /* ... */ }
}
/**
 * Інтерцептор для додавання токена авторизації до HTTP-запитів
 */
@Injectable({ providedIn: 'root' })
export class AuthIntercepotrService implements HttpInterceptor {
  constructor(private authService: AuthService) {}
  intercept(req: HttpRequest<any>, next: HttpHandler) {
    return this.authService.user.pipe(
      take(1),
      exhaustMap((user) => {
```

```
      if (!user) return next.handle(req);

      const modifiedReq = req.clone({
        params: new HttpParams().set('auth', user.token),
      });

      return next.handle(modifiedReq);
    })
  );
 }
}
/**
 * Сервіс для роботи з курсами (фрагмент)
 */
@Injectable({ providedIn: 'root' })
export class CourseService {
 coursesChanged = new Subject<Course[]>();
 private courses: Course[] = [];

 setCourses(courses: Course[]) {
  this.courses = courses;
  this.coursesChanged.next(this.courses.slice());
 }
 getCourses() {
  return this.courses.slice();
 }
 getCourse(index: number) {
  return this.courses[index];
 }
 addCourse(course: Course) { /* ... */ }
 updateCourse(index: number, newCourse: Course) { /* ... */ }
 deleteCourse(index: number) { /* ... */ }
}
```

# ДОДАТОК Б. Слайди мультимедійної презентації



# Використані технології

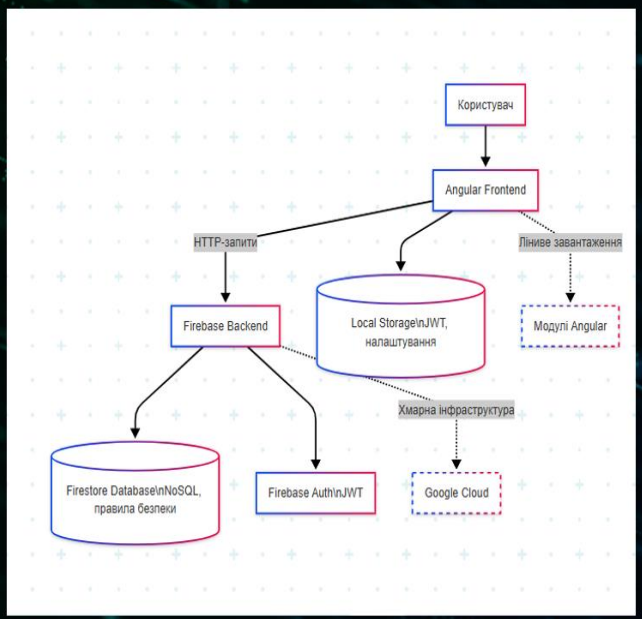
**Frontend:**
- Angular 15+
- TypeScript
- RxJS для стану
- Angular Material UI

**Backend:**
- Firebase Authentication
- Cloud Firestore
- Firebase Hosting
- Firebase Security Rules

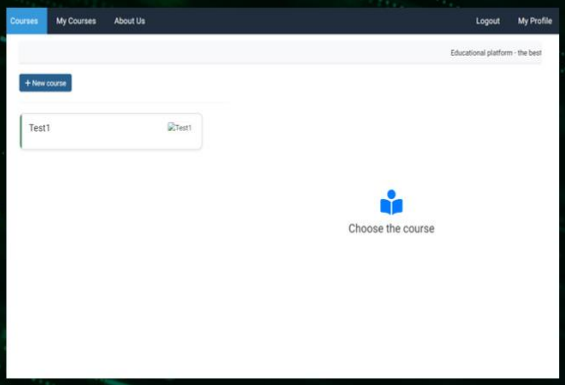
# АРХІТЕКТУРА СИСТЕМИ

Архітектура додатка:
- Frontend: Angular (SPA, модулі, реактивність).
- Backend: Firebase (Firestore, Auth, JWT, безпека).
- Інфраструктура: Google Cloud (масштабованість, надійність).

Ключові переваги:
- Швидкість (ліниве завантаження).
- Безпека (JWT, правила доступу).
- Гнучкість (NoSQL, хмарні рішення).



# ГОЛОВНИЙ ІНТЕРФЕЙС

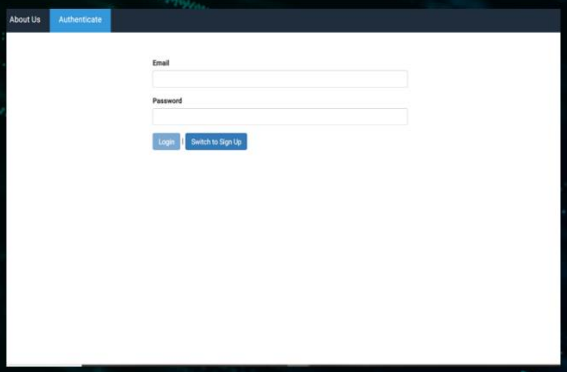



Головна сторінка зі списком курсів

- Навігаційне меню (Courses, My Courses, About Us)
- Кнопка "+ New course" для створення нового курсу
- Список доступних курсів (наприклад, "Test1")
- Панель управління (Logout, My Profile, Manage)

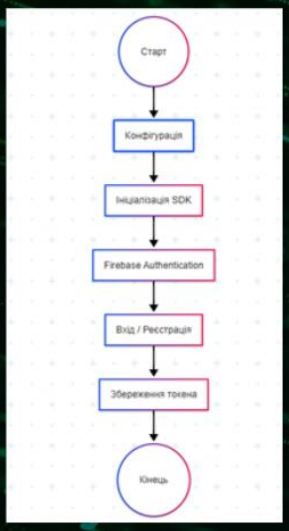
# Система аутентифікації

- Введення email та пароля
- Вибір ролі (студент/викладач)
- Перемикання між формами реєстрації та входу
- Захищене з'єднання з Firebase Authentication

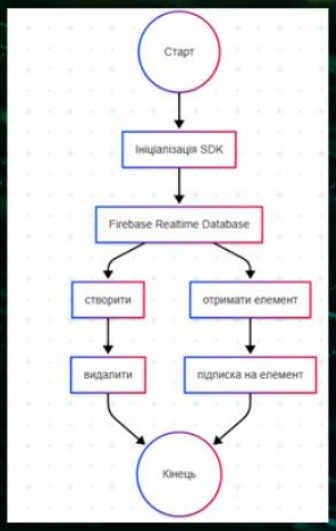



Форма реєстрації/авторизації

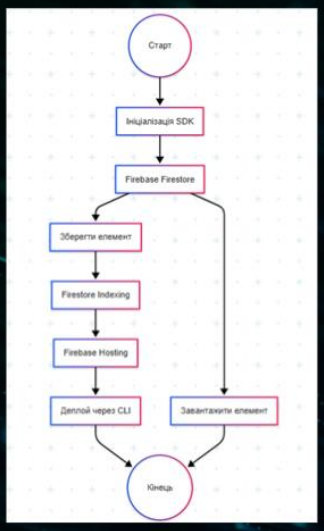
## Налаштування Firebase(Authentication, Firestore, Hosting)



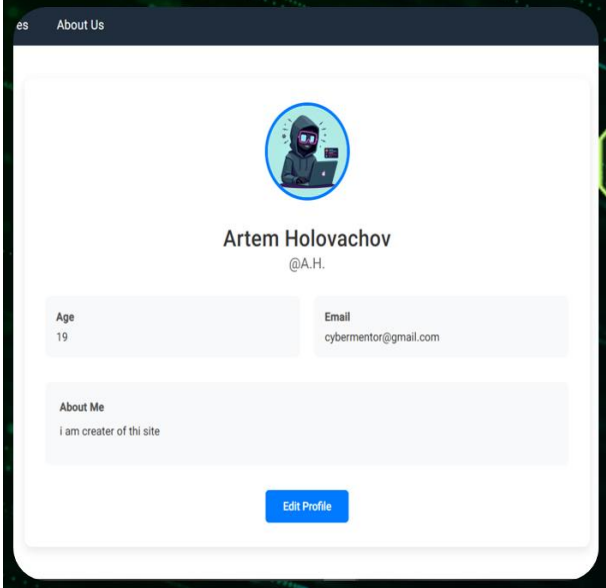

Аутентифікація та конфігурація




Робота з базою даних




Firestore та хостинг

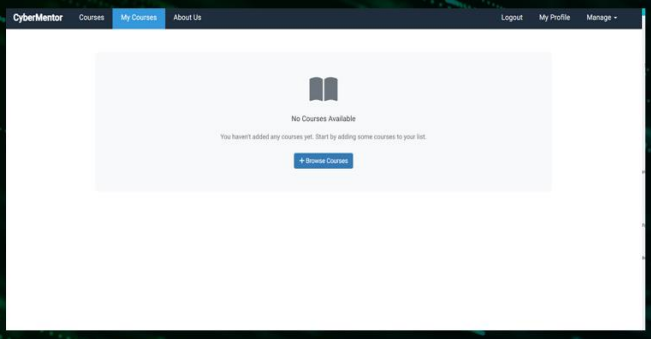



Сторінка профілю користувача

## ОСОБИСТИЙ КАБІНЕТ

- ВІДОБРАЖЕННЯ ОСОБИСТОЇ ІНФОРМАЦІЇ (ІМ'Я, НІКНЕЙМ, ВІК, EMAIL)
- СЕКЦІЯ "ABOUT ME" ДЛЯ ДОДАТКОВОЇ ІНФОРМАЦІЇ
- КНОПКА "EDIT PROFILE" ДЛЯ РЕДАГУВАННЯ ДАНИХ
- ІНТЕГРАЦІЯ З FIREBASE ДЛЯ ЗБЕРЕЖЕННЯ ДАНИХ

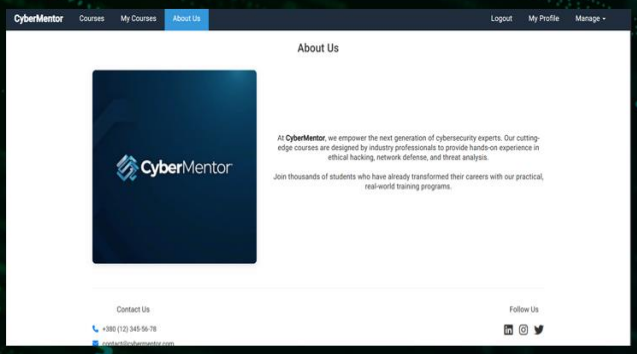
# РОЗДІЛ "MY COURSES"




Сторінка з особистими курсами

- Повідомлення про відсутність курсів (якщо немає доданих)
- Кнопка "+ Browse Courses" для переходу до каталогу
- Можливість додавання курсів до особистого списку
- Відображення прогресу проходження

# СТОРІНКА "ABOUT US"




Інформаційна сторінка про платформу

- Опис місії та цілей платформи
- Інформація про авторів курсів
- Контактні дані для зв'язку
- Мотиваційний контент для користувачів

# ТЕХНІЧНІ ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ

КЛЮЧОВІ МОМЕНТИ:

- ВИКОРИСТАННЯ ANGULAR ДЛЯ ШВИДКОГО РЕАКТИВНОГО ІНТЕРФЕЙСУ
- FIREBASE ЯК БЕЗСЕРВЕРНИЙ БЕКЕНД (AUTH, FIRESTORE, HOSTING)
- РЕАЛІЗАЦІЯ JWT ДЛЯ БЕЗПЕЧНОЇ АВТЕНТИФІКАЦІЇ
- ОПТИМІЗАЦІЯ ЗАВАНТАЖЕННЯ (LAZY LOADING МОДУЛІВ)

# ПЕРСПЕКТИВИ РОЗВИТКУ

Список ідей:

- Інтеграція з LMS системами (Moodle, Google Classroom)
- Додавання механізму відеоконференцій для консультацій
- Розробка мобільного додатку на Ionic
- Система автоматичної перевірки практичних завдань


посилання на веб-інтерфейс

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

# РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

*Головачова Артема Станіславовича*

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Залапін Олексій Ігорович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмного забезпечення для on-line навчання основам кібербезпеки*

Обсяг розрахунково-пояснювальної записки *76* сторінок

Обсяг графічної (презентаційної) частини *12* аркушів (слайдів)

## ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

*Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений розробці програмного забезпечення для on-line навчання основам кібербезпеки та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації*

б) характеристика виконання кожного розділу дипломного проекту

*Пояснювальна записка складається з основного розділу, економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.*

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

*Графічна частина складається з 12 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.*

г) перелік позитивних якостей дипломного проекту *Наявність компонентів для анімацій, відображення індикаторів завантаження та інтерактивних форм сприяє покращенню користувацького досвіду. Реалізація реактивного підходу за допомогою RxJS дозволяє компоненти автоматично обновлюватися у відповідь на зміни даних. Це забезпечує синхронізацію працездатності між різними частинами застосунку*

д) основні недоліки дипломного проекту *Більш детальний розбір покриття тестами та показники продуктивності могли б підвищити впевненість у стабільності системи. Варто було б розглянути сценарії негативного тестування чи edge cases. Деякі логічні блоки (наприклад, налаштування аутентифікації, обробка ролей, синхронізація даних) описані по кілька разів у різних контекстах. Графічна частина проекту слабка, схеми алгоритмів виконані без дотримання вимог*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Задовільно*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис: _____

« 23 » 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

# ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

*Головачова Артема Станіславовича*

(прізвище, ім`я та по батькові)

Спеціальність: *121 "Інженерія програмного забезпечення"*

Освітньо-професійна програма: *«Розробка програмного забезпечення»*

Тема дипломного проекту: *Розробка програмного забезпечення для on-line навчання основам кібербезпеки*

## ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 75 сторінок. У пояснювальній записці наведено етапи розробки програмного забезпечення для on-line навчання основам кібербезпеки. Графічна частина складається з 12 слайдів мультимедійної презентації, які також містять схеми, алгоритми, діаграми, що передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проектом: *Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Головачов А.С. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Головачов А.С. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою. Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту*

**ДОЗВІЛ**
**НА РОЗМІЩЕННЯ**
**ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ**
**(ДИПЛОМНОГО ПРОЕКТУ)**
**В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»**

Ми, що нижче підписалися,

***Головачов Артем Станіславович***
здобувач освіти гр. 4РП-08, та

***Залапін Олексій Ігорович***,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

***«Розробка програмного забезпечення для on-line навчання основам кібербезпеки» (автор роботи – Головачов А.С., керівник роботи – Залапін О.І.)***

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Головачов А.С. /

Керівник _____ / Залапін О.І. /

«16» червня 2025 р.

# ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4РП-08

*Головачова Артема Станіславовича*

на тему *Розробка програмного забезпечення для online навчання основам кібербезпеки*

Висновок відповідальної особи за проведення нормоконтролю:

*пояснювальна записка до дипломного проєкту виконана з деякими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування*

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 18.06.2025 р. значення коефіцієнту подібності в роботі становить 9,39%, коефіцієнт цитування – 0,89%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

**Попередня експертиза (малий захист) дипломного проєкту**

здобувача (здобувачки) освіти *Головачова А.С.* (П.І.Б.)

проведена « *20* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/18/2025
Дата редагування 6/18/2025

Документ прийнятий

# Звіт подібності

## метадані

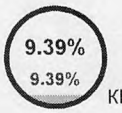
Назва організації
**Odesa Technical Professional College of Odesa National University of Technology**
Заголовок
**Розробка програмного забезпечення для online навчання основам кібербезпеки**

Автор
**Головачов Артем Станіславович**
Науковий керівник / Експерт
**Залапін Олексій Ігорович**

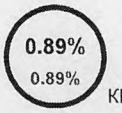
підрозділ
**Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"**

## Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

**25**
Довжина фрази для коефіцієнта подібності 2

**12426**
Кількість слів

**104949**
Кількість символів

## Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

| | | |
|---|---|---|
| Заміна букв | ß | 16 |
| Інтервали | A→ | 1 |
| Мікропробіли | · | 39 |
| Білі знаки | ß | 0 |
| Парафрази (SmartMarks) | a | 57 |

## Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

| ПОРЯДКОВИЙ НОМЕР | НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ) | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| 1 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content | 77 0.62 % |
| 2 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content | 77 0.62 % |
| 3 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content | 66 0.53 % |
| 4 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content | 56 0.45 % |
| 5 | https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download | 43 0.35 % |

| | | |
|---|---|---|
| 6 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content | 43 0.35 % |
| 7 | https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download | 40 0.32 % |
| 8 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content | 31 0.25 % |
| 9 | https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download | 28 0.23 % |
| 10 | https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content | 26 0.21 % |

## з домашньої бази даних (0.35 %)

| ПОРЯДКОВИЙ НОМЕР | ЗАГОЛОВОК | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| 1 | Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності<br>[redacted] 6/12/2025<br>Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету") | 37 (4) 0.30 % |
| 2 | Розробка web-застосунку для генерації повідомлень із використанням технологій штучного інтелекту<br>[redacted] 6/14/2025<br>Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету") | 7 (1) 0.06 % |

## з програми обміну базами даних (0.00 %)

| ПОРЯДКОВИЙ НОМЕР | ЗАГОЛОВОК | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|

## з Інтернету (9.04 %)

| ПОРЯДКОВИЙ НОМЕР | ДЖЕРЕЛО URL | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| **1** | **https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content** | **551 (20) 4.43 %** |
| **2** | **https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download** | **83 (2) 0.67 %** |
| **3** | **https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content** | **69 (5) 0.56 %** |
| 4 | https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download | 62 (4) 0.50 % |
| 5 | https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download | 58 (3) 0.47 % |
| **6** | **https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download** | **51 (3) 0.41 %** |
| 7 | https://card-file.ontu.edu.ua/bitstreams/8999d5af-6274-44f4-ae78-d23e08048d38/download | 45 (5) 0.36 % |
| 8 | https://card-file.ontu.edu.ua/bitstreams/12d5c0ab-e979-48f2-a8ec-d5fc31f71fd5/download | 44 (6) 0.35 % |
| 9 | https://roytuts.com/angular-spring-boot-security-jwt-authentication-and-authorization/ | 31 (4) 0.25 % |
| 10 | https://www.c-sharpcorner.com/article/interceptor-in-angular/ | 30 (2) 0.24 % |
| 11 | https://peerdh.com/uk/blogs/programming-insights/implementing-jwt-authentication-with-angular-routing-and-guards-1 | 25 (2) 0.20 % |
| 12 | https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download | 14 (1) 0.11 % |
| 13 | https://developer.okta.com/blog/2020/01/21/angular-material-login | 14 (1) 0.11 % |

| | | |
|---|---|---|
| 14 | https://blog.flicher.net/ionic-4-user-registration-login-tutorial/ | 12 (2) 0.10 % |
| 15 | https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download | 10 (1) 0.08 % |
| 16 | https://card-file.ontu.edu.ua/bitstreams/d170b7e7-9f64-4cae-8636-2f0a585386fa/download | 10 (1) 0.08 % |
| 17 | https://stackoverflow.com/questions/76204932/how-to-use-canactivatefn-in-angular-16-via-constructor-dependency-injection | 8 (1) 0.06 % |
| 18 | https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download | 6 (1) 0.05 % |

## **Список прийнятих фрагментів** (немає прийнятих фрагментів)

| ПОРЯДКОВИЙ НОМЕР | ЗМІСТ | КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

**Спеціальність: 121 «Інженерія програмного забезпечення»**
Освітньо-професійна програма: «Розробка прошрманого забезпечення **»** **Група:** 4РП- 08

Дипломний **проєкт**
**здобувача освіти денної форми навчання**
РП.08.06.000.ДП

ГОЛОВАЧОВА
АРТЕМА СТАНІСЛАВОВИЧА

м. Одеса
2025 р.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

**Спеціальність: 121 «Інженерія програмного забезпечення»**
Освітньо-професійна програма: **«Розробка програмного забезпечення»**
**Група: 4РП-** 08

ПОЯСНЮВАЛЬНА ЗАПИСКА
(
Розробка

програмного

забезпечення
для
on
-
line

навчання основам
кібербезепки

)до дипломного проєкту на тему:
_____
(
7
6
)_____
(
1
3
)Проєктний **матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах)**