ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка навчального модуля з елементами гейміфікації для вивчення методів шифрування»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Тимофій ЄВСЄЄВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Тимофій ЄВСЄЄВ

Керівник: Юрій АБРОСКІН
асистент кафедри ІПЗ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Євсєєву Тимофію Євгеновичу

1. Тема кваліфікаційної роботи: «Розробка навчального модуля з елементами гейміфікації для вивчення методів шифрування»
керівник кваліфікаційної роботи асистент кафедри ІПЗ Юрій АБРОСКІН,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про класичні методи шифрування (шифр Цезаря, шифр Віженера, бінарне кодування, операція XOR), аналіз існуючих навчальних платформ з елементами гейміфікації, а також технічна документація до засобів реалізації: HTML, CSS, JavaScript та хмарних сервісів платформи Firebase.

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

1. Аналіз застосунку для інтерактивного навчання методам шифрування з елементами гейміфікації.

2. Проектування навчального модулю з елементами гейміфікації для вивчення методів шифрування.
Реалізація навчального модуля з елементами гейміфікації для вивчення методів шифрування.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма діяльності.
5. Екранні форми.
6. Апробація результатів дослідження

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

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Аналіз застосунку для інтерактивного навчання методам шифрування з елементами гейміфікації	14.03-21.03.2025	
4	Проектування навчального модулю з елементами гейміфікації для вивчення методів шифрування	22.03-09.04.2025	
5	Реалізація навчального модуля з елементами гейміфікації для вивчення методів шифрування	10.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Тимофій ЄВСЄЄВ
(підпис)

Керівник кваліфікаційної роботи _____ Юрій АБРОСКІН
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 3 табл., 13 рис., 15 джерел.

Мета роботи – підвищення зацікавленості до вивчення криптографічних методів, шляхом навчання з елементами гейміфікації.

Об'єкт дослідження – процес вивчення криптографічних методів у сфері інформаційних технологій.

Предмет дослідження – засоби програмної реалізації гейміфікованого навчального середовища для викладання криптографії.

Короткий зміст роботи: У кваліфікаційній роботі проаналізовано проблематику традиційних підходів до викладання криптографії та досліджено існуючі навчальні платформи, зокрема Cryptohack, Cybersecurity I (Coursera) та CryptoZombies. На основі аналізу було спроектовано клієнт-серверний веб-додаток з використанням хмарних сервісів Firebase для аутентифікації та збереження даних.

Програмно реалізовано чотири навчальні модулі, що охоплюють ключові методи шифрування: шифр Цезаря, шифр Віженера, бінарне кодування та побітову операцію XOR. Кожен модуль поєднує теоретичний матеріал з інтерактивними практичними завданнями та миттєвою перевіркою результатів. Для підвищення мотивації студентів впроваджено механізми гейміфікації: систему балів, досягнення за проходження уроків та візуальні індикатори прогресу. Розроблено користувацький інтерфейс з унікальним тематичним оформленням для кожного уроку.

Проведено розгортання вебсайту на платформі Firebase Hosting з автоматизацією процесу через GitHub Actions. Сферою використання застосунку є навчальний процес у закладах освіти як допоміжний інструмент для викладачів та студентів, а також для самостійного опанування основ криптографії.

КЛЮЧОВІ СЛОВА: КРИПТОГРАФІЯ, ГЕЙМІФІКАЦІЯ, НАВЧАЛЬНИЙ МОДУЛЬ, МЕТОДИ ШИФРУВАННЯ, JAVASCRIPT, FIREBASE, ШИФР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES – Advanced Encryption Standard.

API – Application Programming Interface.

ASCII – American Standard Code for Information Interchange.

BEM – Block-Element-Modifier.

CDN – Content Delivery Network.

CI/CD – Continuous Integration/Continuous Delivery.

CLI – Command-Line Interface.

CSS – Cascading Style Sheets.

CTF – Capture The Flag.

DOM – Document Object Model.

ECC – Elliptic-curve cryptography.

ES – ECMAScript.

HMR – Hot Module Replacement.

HTML – HyperText Markup Language.

IDE – Integrated Development Environment.

JSON-LD – JavaScript Object Notation for Linked Data.

MMO – Massively Multiplayer Online.

NoSQL – Not only SQL.

OOCSS – Object-Oriented CSS.

RSA – Rivest–Shamir–Adleman.

SMACSS – Scalable and Modular Architecture for CSS.

SPA – Single-Page Application.

SSL – Secure Sockets Layer.

UID – Unique Identifier.

UI/UX – User Interface/User Experience.

VS Code – Visual Studio Code.

W3C – World Wide Web Consortium.

WHATWG – Web Hypertext Application Technology Working Group.

XOR – eXclusive OR.

ВСТУП

Актуальність: в умовах цифрової епохи знання з кібербезпеки, зокрема методів шифрування, стають важливою складовою освітнього процесу. Шифрування є ключовим елементом у захисті особистих даних, безпеці комунікацій і забезпеченні конфіденційності інформації як у приватному, так і в державному секторах. З огляду на зростання потреби в фахівцях у сфері інформаційної безпеки, питання доступності та ефективності навчання методам шифрування стає надзвичайно актуальним.

Сучасні цифрові технології відкривають нові можливості для освіти, дозволяючи створювати інтерактивні навчальні середовища, що сприяють кращому засвоєнню складного матеріалу. Одним з таких інноваційних підходів є використання елементів гейміфікації — ігрових механік, інтегрованих у навчальні процеси для підвищення мотивації, залученості та зацікавленості студентів. Гейміфікований підхід до вивчення шифрування дозволяє зробити навіть складні теоретичні поняття легшими для сприйняття, перетворюючи навчання на захопливу пригоду. Отже, використання гейміфікації стане для вчителя підтримкою у прагненні зацікавити дітей та активізувати їхню участь в навчанні, сприяючи розвитку різноманітних інтелектуальних здібностей. Це створить для учнів простір інтерактивного навчання, де вони матимуть можливість практичного застосування знань, вчитись на помилках та їх виправляти. [1].

Розробка вебсайту навчального модуля з елементами гейміфікації для вивчення методів шифрування надає користувачам можливість самостійно опановувати ключові поняття криптографії у зручний час та у власному темпі.

Об`єктом дослідження є процес вивчення криптографічним методам у сфері інформаційних технологій.

Предметом дослідження є засоби програмної реалізації гейміфікованого навчального середовища для викладання криптографії.

Метою роботи є підвищення зацікавленості до вивчення криптографічних методів, шляхом навчання з елементами гейміфікації.

Методи дослідження: першочергово було проаналізований вже існуючі програмні засоби та навчальні платформи для вивчення методів шифрування для формування функціональних та нефункціональних вимог до свого програмного забезпечення. Наступним кроком є проведення огляду засобів реалізації для вебсайту, які будуть відповідати вимогам для функціоналу вебсайту

Наукова новизна роботи визначається наступними функціональними складовими: інтеграція гейміфікації та криптографічного контенту, адаптивний підхід до утримання мотивації.

Практична значущість результатів полягає у можливості використання реалізованого вебсайту для кращого залучення для вивчення методів шифрування, шляхом гейміфікації навчального модулю.

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ІНТЕРАКТИВНОГО НАВЧАННЯ МЕТОДАМ ШИФРУВАННЯ З ЕЛЕМЕНТАМИ ГЕЙМІФІКАЦІЇ

1.1 Проблематика у навчанні методах шифрування

У сучасному інформаційному просторі, де обсяг даних подвоюється кожні кілька років, а кіберзагрози стають дедалі помітними, здатність розуміти та застосовувати методи шифрування перестає бути прерогативою вузького кола фахівців і перетворюється на необхідний інструмент кожного користувача цифрових сервісів [2]. Проте класичні підходи до викладання криптографії, орієнтовані здебільшого на студентів математичних чи інженерних спеціальностей, часто залишають значну частину аудиторії осторонь через свою абстрактність: громіздкі формули, символи з грецького алфавіту та вичерпні доведення теорем спричиняють відчуття «недоступності» теми.

Предметна галузь проекту охоплює освітні технології у сфері криптографії, орієнтовані на інтерактивне, модульне та візуальне навчання. Йдеться не просто про викладання теорії, а про створення середовища, де учень може самостійно пройти шлях від сприйняття до практики. У центрі задачі — пояснення і закріплення роботи класичних алгоритмів: шифру Цезаря, шифру Віженера, бінарного кодування та логічної операції XOR.

Відчуження від практики — ще одна серйозна перешкода на шляху до опанування криптографії. Студентам покажуть доказ безпеки алгоритму, викладуть його псевдокод, але рідко демонструють, як насправді зміна одного символа або біта впливає на результат шифрування. Без власного досвіду “перетягування літер” чи побітових операцій складні алгоритми залишаються абстракцією, а не інструментом для щоденного застосування.

З огляду на це, сучасна педагогіка звертається до концепції «навчання через дію», яку вперше сформулював Девід Колб у рамках своєї теорії Experiential Learning [3]. Цей підхід передбачає, що знання формується в процесі проходження

чотирьох стадій: конкретного досвіду, рефлексії, абстрактної концептуалізації та активного експериментування. Гейміфікація ефективно підтримує кожну із цих стадій [4].

Не менш важливою є проблема мотивації. У традиційних навчальних середовищах учень виконує завдання заради оцінки або абстрактного “засвоєння теми”, але не відчуває жодної азартної зацікавленості. Відсутність миттєвого фідбеку, чіткої системи заохочень і відчуття прогресу перетворює процес навчання на рутинну роботу — і будь-яка невелика помилка стає приводом для розчарування.

Однак для утримання уваги учня недостатньо лише практики — необхідна внутрішня мотивація. Згідно з теорією самодетермінації Десі та Райана, відчуття автономії, компетентності та належності до спільноти стимулює стійкий інтерес до навчання [5]. Таким чином, поєднання практичних вправ із умовами, які задовольняють ці базові психологічні потреби, створює основу для ефективного засвоєння матеріалу.

Щоб подолати ці перешкоди, необхідно створити навчальний простір, який з'єднує ґрунтовність теорії з інтерактивністю практики та яскравими мотивуючими елементами. Тут на допомогу приходить принцип модульності: кожен алгоритм подається як закінчена “міні-історія” — від стислого теоретичного введення до практичного завдання й тесту-квізу. Така структура дозволяє концентруватися на одному понятті за раз, не розгубившись у морі термінів і визначень.

Гейміфікація — це застосування ігрових механік і дизайнерських підходів у неігрових контекстах з метою підвищення мотивації та залученості користувачів [6].

В інтерактивному навчанні учні моделюють реальні життєві ситуації, застосовують ігрові елементи, розбирають кейси та спільно вирішують завдання, що робить процес більш захопливим і водночас підвищує його ефективність [7].

Нарешті, гейміфікація створює відчуття пригоди. Тематичне оформлення в стилі знайомих відеоігор, система балів і рангів, рейтинг між користувачами — усе це формує атмосферу захопливого квесту. Учень не просто виконує вправу, а

“виконує місію”, прагнучи покращити свій результат і перейти до наступного рівня.

Отже, проблема полягає не в самій криптографії, а в тому, як вона подається: сухо, одновимірно, без зв’язку з практикою та без внутрішньої мотивації. Завдання сучасного навчального веб-сайту — розглядати алгоритми як захопливу послідовність невеликих перемог, об’єднаних у єдиний інтерактивний досвід, який перетворює суху теорію на живий інструмент цифрової безпеки.

1.2 Специфіка у вивчені методів шифрування

Специфіка цієї дисципліни полягає в поєднанні абстрактної математичної теорії й практичної реалізації алгоритмів, що вимагає від студента одночасно розвинених аналітичних здібностей, високого рівня формалізації мислення та навичок програмування.

Перш за все, навчання криптографії ґрунтується на строгій математичній базі. Теореми про складність обчислень, властивості арифметики в скінченних полях і групах, теорема Коші—Галоа, доведення коректності алгоритмів — усе це потребує від слухача готовності до роботи з абстракціями та формальними доказами. У класичному університетському курсі період вивчення математичного апарату (теорія чисел, алгебраїчна алгебра, теорія ймовірностей) іноді займає майже половину семестру, тоді як прикладні протоколи (RSA, ElGamal, ECC) розглядаються вже після формування міцного теоретичного фундаменту.

Важливою особливістю є значний обсяг практичних завдань. Шифрування і дешифрування на папері — корисна вправа для розуміння механізмів, але справжня перевірка навичок відбувається у програмному середовищі. Студенти реалізують алгоритми на різних мовах програмування, налагоджують мережеві протоколи обміну ключами, досліджують стійкість рішень до кутових чи сторонніх атак. Така «комп’ютерна практика» перетворює абстрактні формули на працюючий код і дає можливість аналізувати продуктивність, масштабованість та вразливості алгоритмів.

Специфіка викладання методів шифрування полягає у використанні поетапного, модульного підходу. Кожен новий алгоритм (шифр Цезаря, Віженера, AES, RSA, ECC тощо) розглядається спочатку в контексті історичного розвитку, потім — з точки зору математичних основ, а на завершення — у вигляді практичного модуля з лабораторною роботою чи кейсом аналізу реальних вразливостей. Така структура дозволяє поступово нарощувати складність і не переавантажувати студента одразу великою кількістю нових понять.

Криптографія поєднує інформатику, математику, теорію інформації і навіть елементи психології (наприклад, моделі поведінки користувача при виборі паролів, методи соціальної інженерії). Університетські програми зазвичай включають лекції з усіх цих напрямів, а також семінари з юридичних та етичних питань (регулювання шифрування, захист персональних даних).

Нарешті, важливою специфікою є постійна динаміка галузі. Новітні публікації з криптоаналізу та квантова криптографія змушують викладачів оновлювати навчальні матеріали щороку. Студентів навчають не лише відомим алгоритмам, але й методам теоретичного аналізу нових конструкцій, а також інструментам формальної верифікації протоколів (наприклад, TLA^+ чи ProVerif).

Таким чином, вивчення методів шифрування у академічному середовищі поєднує строгі математичні доведення, інтенсивну програмну практику, модульну структуру викладання, міждисциплінарний підхід та оперативне оновлення контенту [22]. Ця синергія забезпечує формування в студентів не лише теоретичних знань, але й практичних умінь, необхідних для розробки, аналізу та впровадження безпечних інформаційних систем.

1.3 Дослідження існуючих аналогів

На сьогоднішній день для інтерактивного опанування методів шифрування пропонуються такі навчальні модулі:

- Cryptohack;
- Cryptography I (Coursera);
- CryptoZombies.

CryptoHack представляє собою веб-платформу, що побудована за принципом «курс + пазли». Кожен модуль поєднує ґрунтовні пояснення з реальними практичними завданнями: від класичних шифрів Цезаря та Віженера до сучасних протоколів RSA й ECC. Гейміфікація, представлена балами, медалями та глобальним рейтингом, створює сильну мотивацію для послідовного проходження вправ, а поступове ускладнення завдань дозволяє учневі розвинути впевненість у власних навичках.

Проте повністю англомовний інтерфейс і необхідність постійного інтернет-з'єднання можуть обмежити доступність для частини користувачів, а жорсткий лінійний формат не дає великої свободи для налаштування курсу під конкретні навчальні цілі. Приклад екранних форм веб додатку Cryptohack на рисунку 1.1.

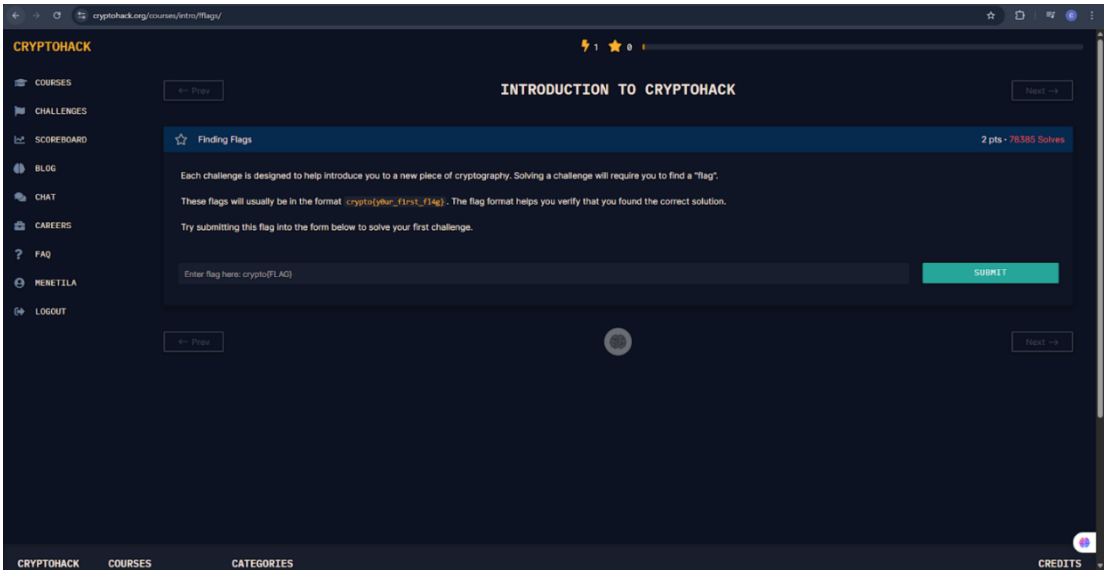


Рис. 1.1 Приклад екранних форм веб платформи Cryptohack.

Переваги використання веб додатку Cryptohack:

- веб додаток має необмежену кількість завдань, постійні оновлення й нові рівні;
- власна мотиваційна система яка влаштована за допомогою балів, CTF-змагань, спільних турнірів;
- реалізована системи підтримки та обмін досвідом у середині спільноти в рамках веб платформи.

Недоліки використання додатку Cryptohack:

складність навчання початківцям методам шифрування через відсутність пояснень принципів та правил з самого початку навчання;

доступні лише англійська та російська мова інтерфейсу;

для доступу до платформи при реєстрації потребуються заздалегідь знань з шифрування методу Цезаря.

Cybersecurity I (Coursera) представляє собою популярний онлайн-курс Стенфорду (інструктор – Д. Боне), присвячений основам теорії криптографії. Підходить для фундаментального розуміння криптографії для початківців. Курс охоплює симетричні і асиметричні методи, саме від теоретичних обґрунтувань до застосування. Він комплексний, добре структурований, але не містить ігрових елементів. Приклад екранних форм веб застосунку Cybersecurity I (Coursera) наведені на рисунку 1.2.

Переваги можна охарактеризувати якісним відео контентом і тестами від провідного професора, глибокою теорії, чіткою програма. Велика кількість допоміжних матеріалів (слайди, посилання). Присутня локалізація інтерфейсу українською мовою, розроблена спільнотою платформи.

Недоліками можна охарактеризувати відсутність гейміфікації (немає балів за швидкість чи змагання між студентами, рівнів, таблиць лідерів), низький рівень інтерактивності уроків поза відео (лише вбудовані у відео запитання та домашні завдання). Доступ до виконання практичних завдань лише за умови оплаченої підписки. Інтерфейс Coursera загалом зручний, але для роботи потрібна реєстрація.

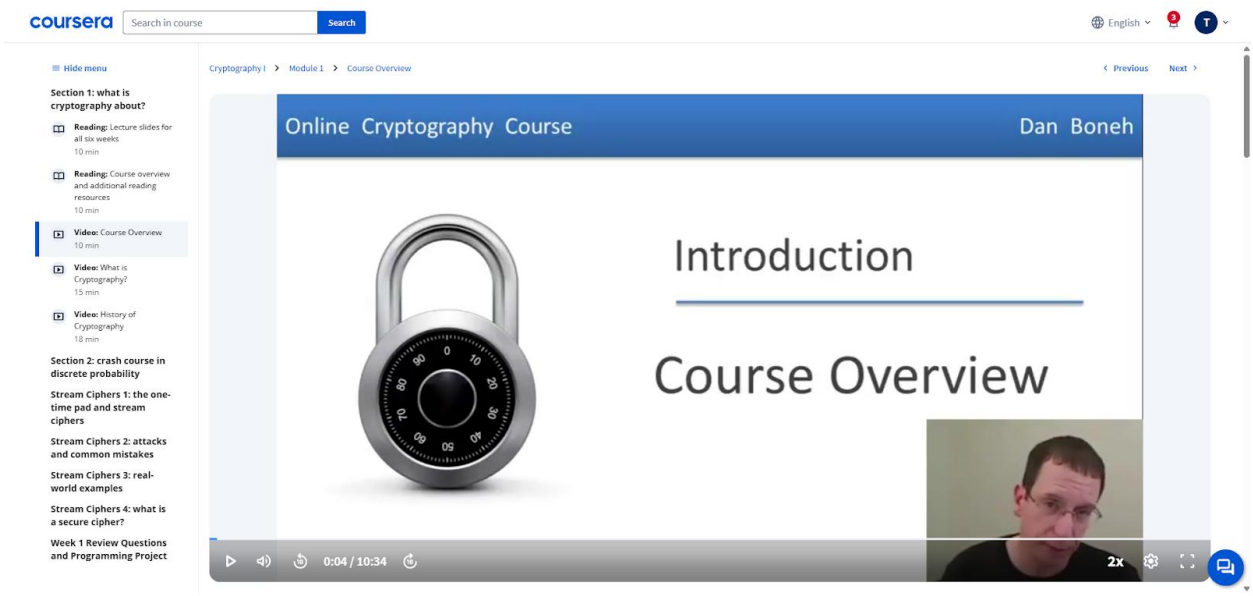


Рис. 1.2 Приклад екранних форм веб застосунку Cybersecurity I (Coursera)

CryptoZombies представляє собою онлайн-школу для вивчення програмування блокчейн і смарт-контрактів на мові Solidity. Уроки побудовані як гейміфікований курс, де користувачі «вирощують» зомбі-збройне плем'я і дізнаються, як писати свій смарт-контракт. У веб-додатку реалізовані гейміфіковане навчання програмуванню блокчейну. Хоча CryptoZombies не присвячена класичним методам шифрування, вона розвиває розуміння криптографічних концепцій у Web3 (хешування, цифрові підписи) інтерактивно. Приклад екранних форм веб застосунку CryptoZombies представлені на рисунку 1.3.

Перевагами є висока мотивація ігровим сюжетом, покрокові онлайн-уроки з миттєвим зворотним зв'язком, вільний доступ. Наявність MMO-елементів, після курсу можна брати участь у змаганнях з іншими гравцями "битва зомбі". Розвинуті елементи гейміфікації, такі як: система «карми», яка підвищується після правильно виконаного завдання, створення персоналізованого аватара, нагороди за проходження завдань у вигляді внутрішньої валюти та інші. Динамічний інтерфейс з візуалізацією ігрофікованих подій.

Недоліки характеризуються фокусом на мові програмування Solidity та блокчейн грі, а не на традиційних методах шифрування. Обмежений набір

криптографії для вивчення. Навчальний курс є менш доречним для вивчення абстрактних методів шифрування. Відсутня локалізація інтерфейсу українською мовою.

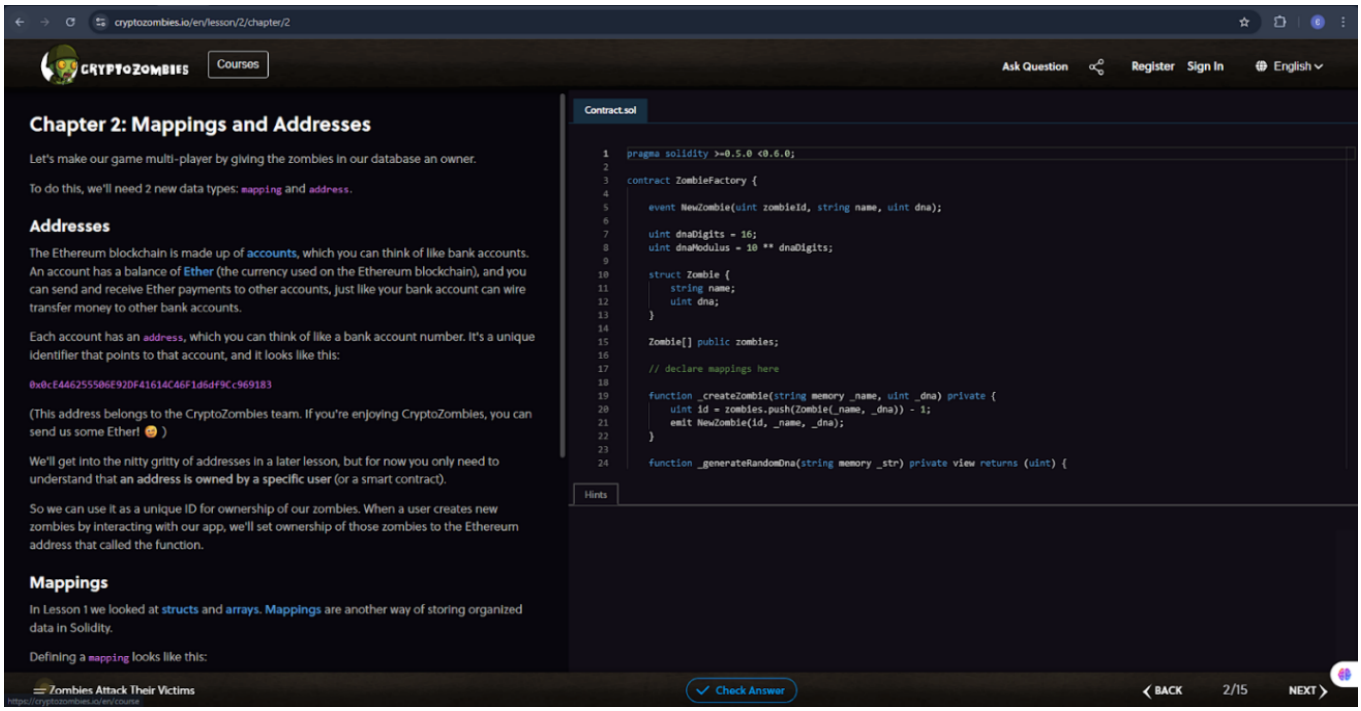


Рис. 1.3 Приклад екранних форм веб застосунку CryptoZombies.

Розглянувши ключові аспекти кожного з розглянутих навчаоьних модулів – від формату подачі матеріалу й рівня інтерактивності до мови інтерфейсу та наявності елементів гейміфікації була зведена таблиця 1.1, щодо аналізу існуючих рішень для вивчення методів криптографії.

У цій таблиці наведено перелік основних критеріїв, за якими оцінюються аналоги, такі як Cryptohack, Cybersecurity I (Coursera) та CryptoZombies. Такий узагальнений огляд спрямовано на те, щоб допомогти викладачам та зацікавленим особам у вивченні методів криптографії швидко зорієнтуватися серед доступних інструментів і зробити обґрунтований вибір або розробити альтернативне рішення, яке поєднає краші практики з існуючих платформ.

Таблиця 1.1

Зведена таблиця аналізу існуючих застосунків для вивчення методів шифрування та їх порівняння

Показник	**Cryptohack**	**Cryptography I (Coursera)**	**CryptoZombies**
Методика «теорія та практика»	Лише практика	Фокус на теорії	Лише практика
Гейміфікація	CTF-змагання	Відсутні	Стиль MMO гри, змагання, турніри
Мультимодальність	Текст та редактор коду	Відео та текстові нотатки	Інтерактивний IDE
Система зворотного зв'язку	Автоматичне тестування коду	Автоматична перевірка завдань	Автоматичне тестування коду та система «карми»
Локалізація українською мовою	Відсутня	Розроблена спільнотою проекту	Відсутня
Глибина матеріалу з навчання криптографії	Широкий спектр	Широкий спектр	Низький, більший акцент на блокчейні
Доступність на платформах	Web	Web, iOS, Android	Web

Продовження таблиці 1.1

Зведена таблиця аналізу існуючих застосунків для вивчення методів шифрування та їх порівняння

Показник	**Cryptohack**	**Cryptography I (Coursera)**	**CryptoZombies**
Інтерфейс	Інтуїтивний	Інтуїтивний, з багатим змістом та детальною інформацією	Динамічний, інтуїтивний
Доступ	Доступ після реєстрації	Доступ після реєстрації	Відкритий доступ

1.4 Засоби реалізації

В сучасних реаліях розробки навчальних вебсайту необхідний не лише продуманий педагогічний підхід ,але й ретельно підібраний набір інструментів для реалізації. У цьому підрозділі розглядаються засоби реалізації програмного забезпечення з шифруванням та елементами гейміфікації.

1.4.1 Visual Studio Code

Visual Studio Code (VS Code) — міжплатформенний редактор коду, розроблений Microsoft у 2015 році. Він поєднує низький поріг входження з розширеними можливостями через магазин розширень який, дозволяє додавати інструменти для статичного аналізу коду (наприклад, ESLint), автоматичного

форматування (наприклад, Prettier), тестування (Jest, Mocha) та контейнеризації середовища (Docker). Основу VS Code становить Language Server Protocol, що стандартизує взаємодію редактора з мовними серверами, надаючи автодоповнення, перевірку синтаксису та рефакторинг. Завдяки гнучкій архітектурі плагінів, користувачі можуть швидко інтегрувати інструменти для аналізу коду (ESLint), форматування (Prettier), тестування (Jest, Mocha), а також сервіси хмарного розгортання та контейнеризації (Docker).

Налаштування середовища у VS Code здійснюється через JSON-файли (settings.json, tasks.json, launch.json), які дозволяють автоматизувати збірку, тестування та налагодження проєктів. Вбудований термінал та панель Git спрощують робочий процес, зменшуючи потребу у зовнішніх інструментах.

Академічні дослідження зосереджені на порівняльному аналізі продуктивності різних IDE, дослідженні ефективності розширень та їх впливі на споживання пам’яті й час запуску. З результатів досліджень випливає, що добре підібраний набір розширень забезпечує збалансоване навантаження, зберігаючи конкурентоспроможність VS Code порівняно з класичними середовищами редагування коду [9].

1.4.2 CSS

Cascading Style Sheets визначає візуальний та представницький шар вебдодатків, даючи змогу контролювати кольори, шрифти, компонування та адаптивність інтерфейсу. Через механізм каскаду та специфічність селекторів, розробники можуть поєднувати глобальні та локальні стилі, керуючи тим, як правила перекривають одне одного. Значна увага приділяється методологіям організації коду, зокрема BEM (Block-Element-Modifier), SMACSS (Scalable and Modular Architecture for CSS) та OOCSS (Object-Oriented CSS), які спрямовані на зменшення зв’язності між компонентами та полегшення масштабування великих проєктів.

Препроцесори Sass та Less надають можливості змінних, вкладень та міксинів, що покращує структуру коду та дозволяє уникнути дублювання. Сучасні

специфікації CSS включають модулі автоматичної організації елементів на сторінці(flexbox) та техніка каскадних таблиць(grid), які надають інтуїтивні засоби для створення одно- та двовимірних макетів відповідно. Адаптивний дизайн підтримується через медіа-запити, які враховують ширину вікна браузера чи характеристики пристрою, забезпечуючи коректне відображення на мобільних і десктопних платформах.

1.4.3 JavaScript

JavaScript — це інтерпретована, динамічна мова програмування, що забезпечує інтерактивну поведінку вебсторінок та розробку односторінкових додатків (SPA). Регулярно оновлювані специфікації ECMAScript (ES5, ES6/ES2015 та до ES2025) вводять синтаксичні та функціональні покращення, включно з модульною системою, стрілковими функціями, шаблонними рядками, деструктуризацією та async/await для асинхронної роботи. Завдяки об’єктно-орієнтованому підходу та підтримці подій JavaScript інтегрується з DOM API, Fetch API та WebSockets, що відкриває можливості для створення реактивних інтерфейсів та реального часу.

Сучасні фреймворки (React, Vue, Angular) базуються на компонентній архітектурі, наближаючи розробку інтерфейсів до декларативного опису з односпрямованим потоком даних. Статична типізація через TypeScript додає безпеки та покращує досвід розробника, забезпечуючи автодоповнення та виявлення помилок на етапі компіляції. Академічні дослідження у цій сфері зосереджуються на порівнянні продуктивності фреймворків, аналізі time-to-interactive та оптимізації розміру пакетів.

1.4.4 HTML

HyperText Markup Language — це фундаментальна декларативна мова, яка задає структуру вебдокументів. Початкові версії HTML, що з’явилися у середині 1990-х, служили для створення спрощених гіпертекстових зв’язків.

Нинішній стандарт HTML5 пропонує широкий спектр семантичних інструментів. Сучасна специфікація містить низку семантичних елементів: "заголовок", "панель навігації", "головний вміст", "розділ", "стаття", "бічна панель" та "підвал", так і ефективність пошукових систем. Використання цих позначень підвищує читабельність та логічну організацію документа, а також полегшує коректну індексацію пошуковими системами.

Розвиток HTML активно обговорюється спільнотами, такими як Всесвітнім консорціумом мережі (W3C) та Консорціумом постійного розвитку HTML (WHATWG), де формулюються нові специфікації та вдосконалення Сучасна специфікація містить низку семантичних елементів: "заголовок", "панель навігації", "головний вміст", "розділ", "стаття", "бічна панель" та "підвал". Використання цих позначень підвищує читабельність та логічну організацію документа, а також полегшує коректну індексацію пошуковими системами. Це забезпечує інкапсуляцію та повторне використання Атрибути доступності забезпечують підтримку допоміжних технологій, а впровадження мікроданих та формату JSON-LD полегшує розпізнавання ключової інформації пошуковими алгоритмами.. З академічної точки зору HTML розглядається як формальна система зі строгою граматикою та семантикою, яка слідує специфікаціям мови та еволюціонує згідно з реальними потребами вебспільноти.

1.4.5 Node.js

Node.js використовується як середовище виконання JavaScript поза браузером для організації процесу розробки та збірки. Серверне середовище Node.js, побудоване на рушії V8, підтримує неблокуючу модель введення/виведення та цикл подій, що дає змогу обслуговувати багато з'єднань одночасно з низькою затримкою. Менеджер пакетів npm надає доступ до широкого спектра бібліотек для роботи з базами даних, аутентифікації та автоматизації процесів.

Фреймворки Express і Koa створюють засоби для маршрутизації запитів і обробки проміжних рівнів логіки, спрощуючи створення прикладних інтерфейсів

веб-служб. З академічного погляду, аналізують масштабованість, продуктивність вводу/виведення та переваги неблокуючої моделі порівняно з багатопотоковими системами.

Локальний сервер розробки з гарячим перезавантаженням (HMR) дозволив одразу бачити ефект від змін у коді без перезапуску всієї сторінки. Для забезпечення CI/CD процесу було побудовано автоматична інтеграція у GitHub Actions: на кожному пуші виконуються тести та лінтинг, а у випадку успішного проходження надає можливість на автоматичну загрузку вебсайту на Firebase Hosting.

1.4.6 Firebase

Firebase об'єднує в єдину платформу сервіси, необхідні для хмарного розгортання та роботи навчального модуля. Firebase Hosting забезпечує глобальне розповсюдження статичного контенту через CDN з автоматичним SSL-сертифікатом та високою доступністю. Деплой здійснюється за допомогою CLI командою firebase deploy, що робить процес максимально простим та передбачуваним.

Cloud Firestore виступає як NoSQL-база даних у реальному часі. Дані про користувачів такі як: ідентифікатор, ім'я, електронна пошта, а також налаштування активного сеансу зберігаються в окремих колекціях. Підтримка офлайн-режиму та автоматичної синхронізації гарантують збереження результатів навіть при тимчасовій відсутності зв'язку.

Особливу увагу приділено правилам безпеки Firebase Security Rules: вони обмежують права читання та запису, дозволяючи студенту змінювати тільки власні дані та запобігаючи потенційним зловживанням. За потреби модуль аутентифікації Firebase Authentication можна розширити, щоб додати логін через Google або інших провайдерів, не реалізовуючи складний серверний код.

Платформа Firebase від Google пропонує готові серверні сервіси для швидкого запуску веб- і мобільних проєктів. Вона включає бази даних із синхронізацією в реальному часі та можливістю складних запитів, засоби

аутентифікації через електронну пошту чи соціальні мережі, хмарні обчислення, розміщення з глобальною мережею доставки контенту та автоматичну безпеку транспортного рівня.

1.5 Об’єкт, предмет, мета та постановка завдань кваліфікаційної роботи

Об’єктом дослідження дослідження є процес вивчення криптографічним методам у сфері інформаційних технологій.

Предметом дослідження є засоби програмної реалізації гейміфікованого навчального середовища для викладання криптографії.

Метою роботи є підвищення зацікавленості до вивчення криптографічних методів, шляхом навчання з елементами гейміфікації.

Аналізуючи специфіку процесу навчання методам шифрування, а також наявні засоби, платформи та веб ресурси з вивчення шифрування з та без елементів гейміфікації, виникає можливість визначити вимоги до майбутнього програмного рішення. Рішення для вивчення методів шифрування має бути розроблена як веб платформи та забезпечувати такий функціонал:

- використання практичних завдань, у яких користувач застосовує набуті знання для шифрування або дешифрування повідомлень;
- наявність системи підказок та допомоги, яка активується за запитом користувача або після кількох невдалих спроб;
- подання змісту уроку за допомогою короткої ігрофікованої історії під конкретний урок;
- застосування системи балів або винагород за правильне виконання завдань для підвищення мотивації;
- можливість проходження міні-тестів або контрольних завдань для перевірки рівня засвоєння матеріалу.

2 ПРОЕКТУВАННЯ НАВЧАЛЬНОГО МОДУЛЮ З ЕЛЕМЕНТАМИ ГЕЙМІФІКАЦІЇ ДЛЯ ВИВЧЕННЯ МЕТОДІВ ШИФРУВАННЯ

Система представлена як єдине веб-застосування клієнтського типу з використанням хмарного середовища для зберігання даних та автентифікації. На стороні користувача працює комплект HTML-сторінок із програмними сценаріями на JavaScript та каскадними таблицями стилів, а на стороні сервера — хмарні служби для авторизації та сховища користувацьких даних.

2.1 Проектування загальної архітектури системи

Система реалізована за клієнт-серверною схемою з використанням хмарних сервісів Firebase (автентифікації та бази даних) на боці сервера. Фронтенд представляє собою набір веб-сторінок (HTML/JavaScript/CSS), що працюють у браузері користувача, а всі серверні функції надані Firebase («Authentication», «Firestore Database»). Тобто фактично використовується монолітна архітектура: усі компоненти зосереджені в межах одного клієнтського додатку. Для розділення коду застосовано ES-модулі та окремі сценарії (програмні скрипти) для кожного розділу шифрування. Систему можна розбити на такі функціональні блоки:

- модуль аутентифікації: забезпечує реєстрацію/вхід користувача через email-пароль або Google (Firebase Auth);
- модуль навігації: головна сторінка та меню з кнопками для вибору завдання;
- модуль навчальних вправ: окремі сторінки для завдань з різних методів шифрування (Цезар, Віженер, бінарні коди тощо) та пов’язана з ними логіка перевірки рішень;
- модуль гейміфікації: відповідає за нарахування балів, відображення індикатора прогресу та збереження досягнень (запис у базу Firestore).

Кожен модуль взаємодіє через чітко окреслені інтерфейси: наприклад, модуль вправ отримує введення від користувача, перевіряє відповідь та у разі успіху викликає функції модуля гейміфікації для оновлення балів і досягнень.

У клієнтському коді використано патерн спостерігача: подія onAuthStateChanged від Firebase Auth сповіщає про зміну стану авторизації і оновлює UI. Для стилізації застосовано бібліотеку TailwindCSS і окремі CSS-файли з ігровими шрифтами та кольорами (наприклад, шрифт «Press Start 2P» для стилю ретро-ігри). Компоненти виконані таким чином, щоб логіка прикладного рівня була відділена від презентації – HTML відповідає за макет, CSS за оформлення, а JS за обробку подій і взаємодію з сервером.

2.2 Проектування сценаріїв використання системи

Головним користувачем начального модуля є учень. Для ілюстрації сценаріїв системи застосована діаграма випадків використання (use-case діаграму) на рисунку 2.1, на якій учень пов’язаний з переліченими кейсами:

- реєстрація та вхід зумовлена тим, що користувач може створити акаунт або увійти через Google;
- перегляд навчальних модулів здійснюється після входу користувача бачить головне меню з переліком доступних завдань.
- - вивчення матеріалу зумовлена тим, що кожен розділ містить теоретичні пояснення шифру й практичні завдання (шифрування, дешифрування, виконання задач).
- отримання зворотного зв’язку характеризується тим, що після вводу відповіді система перевіряє результат і відображає повідомлення про успіх або помилку.
- нарахування балів та досягнень представляє собою успішне виконання завдань користувач отримує бали (score), а після повного проходження модуля – відповідне досягнення

- перегляд досягнень у будь-який момент, таким чином учень може переглянути набрані відзнаки.

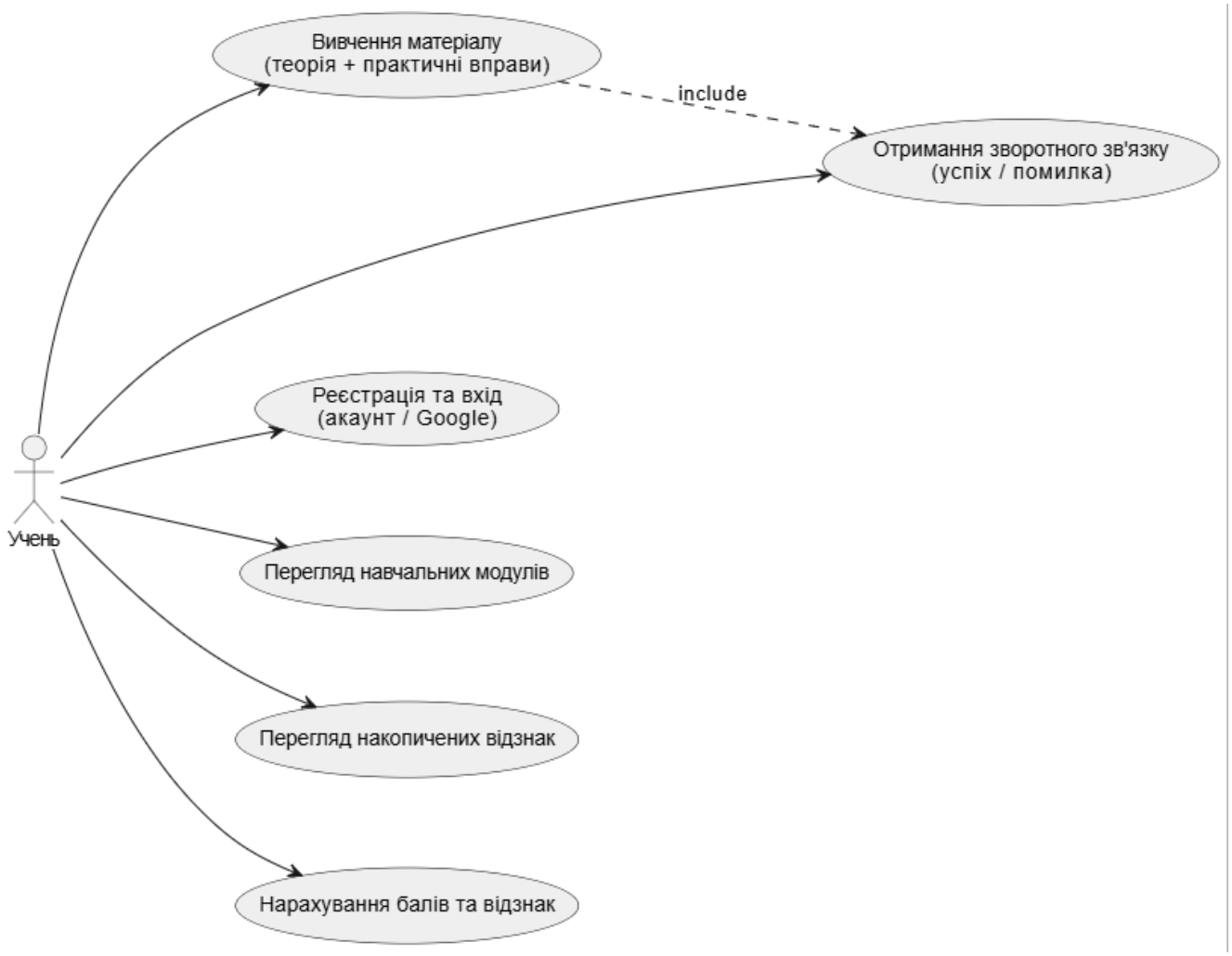


Рис. 2.1 Діаграма випадків використання

2.3 Функціональні та не функціональні вимоги

Першим кроком у процесі проектування є формалізація вимог. Функціональні та нефункціональні вимоги окреслюють, що система повинна робити та яким чином. Для зручності читання нижче зведені таблиці функціональних вимог у таблиці 2.1 та нефункціональних вимог у таблиці 2.2.

Таблиця 2.1

Зведена таблиця функціональних вимог для вебсайту із вивчення методів шифрування та їх порівняння

№	**Вимога**	**Опис**
1	Аутентифікація користувача	Система повинна дозволяти реєстрацію та вхід через електронну пошту/пароль або обліковий запис Google.
2	Перегляд меню уроків	Користувач може проходити уроки: читати теорію, вводити відповіді на задачі, переглядати перевірений результат.
3	Виконання завдання	Користувач може проходити уроки: читати теорію, вводити відповіді на задачі, переглядати перевірений результат.
4	Нарахування балів і досягнень	За кожен правильно виконаний етап нараховуються бали, а також повне виконання уроку присвоює досягнення.
5	Збереження прогресу	Отримані досягнення та балли повинні зберігатися у БД (Firebase Firestore) для кожного користувача.

Таблиця 2.2

Зведена таблиця нефункціональних вимог для вебсайту із вивчення методів шифрування та їх порівняння

№	Вимога	Опис
1	Інтерфейс надихнений відеоіграми	Інтерфейс має бути інтуїтивно зрозумілим, з єдиним ігровим стилем та достатньо великими елементами управління.
2	Швидкодія та відгук	Сторінки повинні швидко реагувати на дії користувача (затримка обробки не більше кількох сотень мс).
3	Безпека	Доступ до персональних даних (нараховані досягнення) має бути можливий лише після успішної автентифікації.
4	Масштабованість	Оскільки додаток працює з Firebase, він автоматично підтримує масштабування без необхідності власного сервера.

Наведені у таблиці 2.1 функціональні вимоги визначають основні сценарії взаємодії користувача з навчальним модулем.

По-перше, підсистема аутентифікації гарантує надійний доступ через електронну пошту або обліковий запис Google, що забезпечує захищеність персональних даних і зручність входу.

По-друге, інтерфейс головного меню дає змогу одразу після входу отримати огляд усіх доступних уроків, що сприяє інтуїтивній навігації.

Третя вимога закладає основу педагогічного процесу: студент послідовно вивчає теорію й переходить до практичних вправ з одразу інтегрованою перевіркою результатів.

Далі, механізм нарахування балів і відзнак стимулює мотивацію та відчуття прогресу. І нарешті, збереження прогресу у хмарній базі Firestore дозволяє відновити роботу з будь-якого пристрою без втрати отриманих досягнень.

Нефункціональні вимоги (таблиця 2.2) визначають стандарти якості. Інтуїтивно зрозумілий та єдиний ігровий стиль підвищує залученість, а швидка реакція інтерфейсу дає змогу забезпечити рівень задоволеності від використання.

Високий рівень безпеки оберігає персональні дані від несанкціонованого доступу, а хмарне масштабування дає змогу працювати з великою кількістю користувачів без втрати продуктивності. У сукупності ці вимоги формують надійну, зручну й ефективну платформу для вивчення методів шифрування з елементами гейміфікації.

2.4 Проектування структури бази даних

Сучасна система зберігання даних побудована на основі хмарного документного сховища Firestore, що належить до класу безреляційних систем(NoSQL). Кожному зареєстрованому користувачу відповідає окремий документ у колекції «users», де ключем слугує унікальний ідентифікатор (uid), згенерований під час автентифікації. У межах цього документа зберігається адреса електронної пошти(email), яка також надходить від служби авторизації, та масив рядкових ключів відзнак — «achievements». Для ілюстрації взаємодії вебсторінки із базою данних наведена діаграма класів на рисунку 2.2.

Цей підхід забезпечує компактність моделі і водночас гарантує гнучкість: додавання нової відзнаки відбувається за допомогою атомарної операції об’єднання масиву, що виключає дублювання записів і зберігає цілісність даних.

Оскільки весь навчальний контент розміщений у клієнтській частині, база виконує роль виключно сховища персонального прогресу. Така мінімалістична

модель дозволяє уникнути зайвих зв’язків між колекціями та спрощує масштабування: із зростанням кількості користувачів не виникає необхідності в розширенні ресурсів серверної частини, оскільки обробка запитів відбувається в хмарі. Запити на запис і читання здійснюються за єдиним шаблоном, що сприяє підтримці коду на високому рівні однорідності. Крім того, використання документно-орієнтованої структури спрощує можливість подальшого розширення моделі даних (наприклад, додавання поля «дата останньої активності» чи «рівень складності»), не порушуючи логіку вже існуючих записів. Така архітектура не лише забезпечує надійність і доступність, а й відповідає вимогам сучасних освітніх додатків із елементами гейміфікації.

Рис. 2.2 Діаграма класів

У представленій діаграмі класів відображено дві ключові сутності предметної області навчального модуля: клас User, що моделює документ користувача у базі, та клас FirestoreService, який забезпечує взаємодію з хмарним сховищем. Клас User містить три приватні властивості: uid (унікальний рядок-ідентифікатор), email (рядок із адресою електронної пошти) і achievements (список рядкових ключів отриманих відзнак). Така модель відповідає принципам

докладної предметної області: усі персональні дані й прогрес користувача зберігаються централізовано та окремо від логіки обробки.

Клас FirestoreService реалізує публічні методи створення та отримання об’єктів User (createUser, getUser) і метод addAchievement() для атомарного доповнення масиву відзнак у документі. Зазначена сервісна сутність відповідає принципу єдиної відповідальності: вона унеможливлює прямий доступ класу User до хмарного сховища й забезпечує чітке розмежування між моделлю даних і протоколом взаємодії з базою. Асоціація зі стрілкою «створює та оновлює» демонструє керовану взаємодію сервісу з екземплярами класу User і підкреслює, що всі зміни документа виконуються через засоби FirestoreService.

Такий підхід відповідає стандартам уніфікованого моделювання: класи мають чіткі назви й розділені секції для властивостей і методів, а зв’язки позначені у вигляді асоціацій. Модель легко піддається розширенню: у майбутньому можна додати класи для аналітики, логування чи обробки помилок, з’єднавши їх аналогічними асоціаціями із FirestoreService або безпосередньо з User. Зберігаючи мінімалізм та модульність, така структура забезпечує надійну основу для подальшої розробки й тестування системи.

2.5 Проектування інтерфейсу користувача

Інтерфейс системи слід реалізувати у вигляді кількох веб-сторінок зі спільним стилем в ігровій тематиці. Основні сценарії взаємодії з UI такі:

- сторінка входу (index.html): користувач бачить вітальний екран із назвою модуля і формою для входу/реєстрації. Є поля для електронної пошти і пароля та кнопка входу через Google. Після успішного входу відображається повідомлення і відбувається перехід на головне меню.;
- Головне меню (menu.html): містить логотип і перелік доступних уроків як кнопки (наприклад, «шифр Цезаря», «шифр Віженера» тощо). Кожній кнопці присвоєно тематичну іконку. Є також кнопка перегляду досягнень (модальне

вікно), де виводяться назви вже отриманих бейджів. Якщо користувач натискає кнопку уроку, відкривається відповідна сторінка завдання

- сторінка завдання (наприклад, cesar.html): зверху є кнопка «Назад» до меню. Далі послідовно розташовані блоки теорії та практичні кроки. Кожен крок містить інструкцію (текст) і поле для вводу відповіді. Кнопки типу «Перевірити відповідь» запускають скрипт, що перевіряє введення. Результат (правильно/неправильно) відображається кольором або повідомленням. Вгорі сторінки – прогрес-бар і лічильник балів, які оновлюються динамічно після кожної правильного кроку. Якщо усі кроки пройдено, показується фінальне повідомлення про успішне закінчення уроку.

Вимоги до UI/UX можно сформулювати наступним чином - інтерфейс повинен бути виконаний в єдиному стилі з урахуванням ігрових метафор: використано ігрові шрифти та яскраві градієнти, що відповідають тематиці кожного шифру. Наприклад, для шифру Цезаря застосовано шрифт у стилі ретро-ігрової консолі. Сторінки адаптивні завдяки CSS (meta viewport), всі елементи керування достатньо великі для зручної роботи на мобільних пристроях. Завдання формулюються простою мовою, система дає підказки (наприклад, обмеження введення лише латинськими літерами) і одразу дає зворотний зв’язок. Важлива вимога – швидкий відгук: перевірка відповіді відбувається миттєво (скрипт працює на клієнті), а прогрес миттєво оновлюється. У дизайні передбачено високу контрастність ігрового інтерфейсу (різкий контраст фону і тексту) для кращої читабельності.

2.6 Гейміфікаційні механізми

Система включає основні елементи гейміфікації, що підвищують мотивацію користувача [6]. Зокрема:

- бали які під час виконання завдань за кожну правильну відповідь нараховуються бали. У коді це змінна «score», яка оновлює свій відображуваний

лічильник після успішних кроків. бали служать миттєвим показником успіху в поточному завданні.

- Досягнення визначені набором констант ALL_ACHIEVEMENTS – ключі та назви ігрових нагород за проходження модулів (наприклад, CESAR MASTER, VIGENERE HUNTER тощо). Після завершення всіх кроків певного уроку викликається функція saveAchievementToFirestore() із відповідним ключем (наприклад, "cesar_all_done"), і ця ачивка зберігається у полі achievements документа користувача у Firestore. Учень може переглянути набрані ачивки в меню (відкривається модальне вікно зі списком назв бейджів). Реалізовано також секретне досягнення GODMODE, яке отримується автоматично, якщо у користувача вже є всі основні бейджі (логіка перевіряється функцією checkSecretAchievement()).
- Індикатор прогресу: у кожному уроці є графічний прогрес-бар, що показує відсоток виконаних кроків (міняє ширину в залежності від відсотка виконання). При досягненні 100% він сигналізує про завершення уроку.
- Декоративний дизайн: кожен урок оформлено у стилі тематичної гри (напр., шрифти і кольорові фони створюють атмосферу «кіберпанку» або «ретро комп’ютерів»). Це створює додаткову ігрову мотивацію – за проходження уроку користувач ніби «відкриває» відповідну ігрову тему.

Таким чином, реалізовано базові принципи гейміфікації – нагороди за успіхи, відчуття прогресу та змагальний елемент (змагання із самим собою по балах). Як відзначають дослідники, такі ігрові механіки роблять навчальний процес більш захоплюючим і ефективним [5].

Сучасні дослідження підтверджують, що особливо ефективними є адаптивні гейміфікаційні підходи. Наприклад, Zourmpakis та інші у своїй роботі аналізують вплив реалізації та адаптованих ігрових елементів на мотивацію студентів, демонструючи позитивні результати в науковій освіті. Це підкреслює потенціал для подальшого розвитку навчального модуля, де гейміфікаційні елементи могли б динамічно підлаштовуватися під прогрес та потреби користувача [10].

3 РЕАЛІЗАЦІЯ НАВЧАЛЬНОГО МОДУЛЮ З ЕЛЕМЕНТАМИ ГЕЙМІФІКАЦІЇ ДЛЯ ВИВЧЕННЯ МЕТОДІВ ШИФРУВАННЯ

3.1 Опис реалізації програмного забезпечення

Модуль реалізовано як веб-додаток за клієнт–серверною моделлю, де клієнтська частина (браузер) відповідає за відображення інтерфейсу та обробку введених користувачем даних, а серверна частина (Firebase) зберігає результати та забезпечує авторизацію [11]. Програмний модуль складається з кількох HTML-сторінок (меню та окремих уроків) і пов’язаних із ними JavaScript-скриптів, які реалізують алгоритми шифрування, а також CSS-стилів для оформлення. Головна сторінка надає навігацію між уроками, кожен урок відкривається у власному браузерному вікні (сторінці) з полями для введення даних і кнопками керування. Загальну структуру можна описати багаторівнево: рівень представлення (HTML/CSS-сторінки та елементи введення), рівень логіки (JavaScript-скрипти, що містять алгоритми шифрування та перевірки) і рівень даних (Firebase, що зберігає досягнення й результати користувача).

Оскільки програмний модуль орієнтований на браузер, інтеграція з Firebase забезпечує збереження балів, авторизацію та синхронізацію результатів. Прикладом служить скрипт script_achievements.js, який опрацьовує досягнення і взаємодіє з базою даних (через Firebase API) для оновлення інформації про бали. Архітектурно це дозволяє розділити відображення та обробку даних: браузерна частина відповідає за швидку взаємодію із користувачем, а бекенд – за надійне збереження та обробку результатів.

Обрана клієнт–серверна архітектура веб-додатку забезпечує кросплатформеність і гнучкість розгортання (використання Firebase Hosting) [11]. Перевагою є простота розробки інтерфейсу та можливість миттєвого оновлення контенту, а зберігання даних у хмарі підвищує надійність роботи. Серед недоліків – залежність від мережі при зверненні до сервера та потенційні затримки при

передачі даних. Незважаючи на це, така модель обрана через її ефективність для освітнього сервісу: користувач може займатися з будь-якого пристрою, а дані його сесій надійно зберігаються.

3.2 Алгоритмічна реалізація

У цьому підрозділі розглянуто реалізацію ключових алгоритмів шифрування, використаних у кожному з уроків. Для наочності наведено псевдокоди основних шифрів: Цезаря, Віженера, бінарного кодування і XOR.

3.2.1 Шифр Цезаря

Мета уроку: ознайомити користувача з простим замінним шифром Цезаря, продемонструвати, як зміщення букв здійснює шифрування і дешифрування.

Опис алгоритму: Шифр Цезаря – це замінний симетричний алгоритм шифрування, де кожна літера відкритого тексту замінюється на літеру, зсунуту на фіксовану кількість позицій по алфавіту [12].

У формулі 3.1 продемонстровано алгебраїчний принцип дії шифру Цезаря:

$$E_n(x) = (x + n)\ mod\ N, \tag{3.1}$$

де x - числове представлення символу відкритого тексту;

n - ключ шифру (зсув), $1 \leq n \leq N - 1$;

N - розмір алфавіту (наприклад, $N = 26$ для англійської абетки);

операція «$mod\ N$» - взяття залишку від ділення на N , що забезпечує циклічність алфавіту.

Алгоритм простий і вимагає мінімальних обчислювальних ресурсів, тому часто використовується для навчальних цілей. Для розшифрування виконується зворотний зсув на той самий ключ.

Система порівнює отриманий зашифрований текст із правильним (наприклад, генерується очікуваний результат за відомим алгоритмом з даним ключем). Якщо користувач правильно обрав параметри, видається повідомлення про успішне шифрування/дешифрування. Користувач також може виконати зворотну операцію, подаючи зашифрований текст і ключ для розшифрування, щоб переконатися у коректності кодування.

3.2.2 Шифр Віженера

Мета уроку: продемонструвати складніший полілалфавітний алгоритм шифрування, що використовує рядок-ключ і набір таблиць (квадрат Віженера), поглибити знання про кластери зсувів. Опис алгоритму: шифр Віженера – це поліалфавітний шифр, який поєднує послідовність шифрів Цезаря з різними зсувами, що визначаються буквами ключа [13]. Ключове слово повторюється вздовж тексту, і для кожної букви відкритого тексту береться відповідна літера ключа.

У формулі 3.2 продемонстровано алгебраїчний принцип дії шифру Цезаря:

$$E_i = (P_i + K_i)\ mod\ N, \tag{3.2}$$

де P_i - числовий індекс $i-$ го символу відкритого тексту;

K_i- числовий індекс $i-$ го символу розширеного ключа;

N - розмір алфавіту (наприклад, $N = 26$ для англійської абетки);

операція «mod N» - взяття залишку від ділення на N , що забезпечує циклічність алфавіту.

Таким чином для кожного символу використовується власний зсув, що робить шифр значно складнішим для злому за допомогою частотного аналізу. Дешифрування виконується аналогічно: зашифрований символ знаходять у рядку таблиці по ключовій літері, а відповідний стовпчик дає вихідний символ. Це й є класичним представленням шифру Віженера в термінах модульної арифметики.

3.2.3 Бінарне шифруваня

Мета уроку: навчити користувача переводити текстову інформацію в двійковий формат і навпаки, ознайомити з основами кодування символів (наприклад, ASCII-кодування).

Опис алгоритму: Бінарне шифрування перетворює текст у послідовність двійкових чисел. Кожен символ відкритого тексту перетворюється у свій кодовий номер за стандартом ASCII (або Unicode), після чого кожен код представлено у двійковому вигляді (зазвичай 8 бітів).

У формулах 3.3 та 3.4 продемонстровано алгебраїчний принцип дії бінарного шифрування:

$$a_i = ASCII(p_i), a_i \in [0{,}255], \tag{3.3}$$

де a_i - числовий код символу p_i за стандартом ASCII;

$ASCII(*)$ - функція, що повертає код символу.

Далі цей код подається у вигляді восьмибітного рядка у формулі 3.4:

$$B = b_{1,1}, b_{1,2}, \dots, b_{1,8}, \dots, b_{n,1}, \dots, b_{n,8}, \tag{3.4}$$

де B - послідовність бітів для всього тексту;

$b_{i,j} \in \{0,1\}$ – j-й біт восьмирозрядного представлення коду a_i;

індекси i та j – пробігають $i = 1..n, j = 1..8$.

Отримані бітові послідовності порівнюють з еталонними; при невідповідності формату або кодування виводиться повідомлення про помилку.

Наприклад, літера «А» має ASCII-код 65, що у двійковому вигляді дорівнює 01000001. Для відновлення тексту з двійкового рядка послідовність бітів ділять на групи по 8 і переводять кожну групу назад у символ.

Бінарне шифрування ілюструє перетворення даних із текстового формату в двійковий і навпаки, що є основою комп'ютерної обробки інформації. Перевагою цього уроку є практичність (кожен символ чітко відображається у двійковому вигляді), а також зрозумілість алгоритму перекладу. Однак недоліками можуть бути необхідність жорсткого формату (фіксована довжина коду, обмеження символів ASCII) та можливі помилки при введенні.

3.2.4 Шифр XOR

Мета уроку: показати, як працює побітова операція XOR для шифрування даних, продемонструвати властивість зворотності: повторна операція з тим самим ключем відновлює початковий текст.

Опис алгоритму: Шифр XOR ґрунтується на побітовій операції виключного АБО (XOR) між бітами відкритого тексту та бітами ключа. Якщо ключ менший за повідомлення, він циклічно повторюється.

У формулі 3.5 продемонстровано алгебраїчний принцип дії шифру XOR:

$$A \oplus B = A\overline{B} + \overline{A}B, \tag{3.5}$$

де A – перший вхідний біт;

B - другий вхідний біт;

$\overline{A}$ - логічне «НІ» відносно A (інверсія);

$\overline{B}$ - логічне «НІ» відносно B (інверсія);

$A\overline{B}$ - добуток, що дорівнює 1 коли $A = 1$ і $B = 0$;

$\overline{A}B$ - добуток, що дорівнює 1 коли $A = 0$ і $B = 1$;

«+» - логічне АБО;

«$\oplus$» - операція виключного АБО (XOR), що дає 1 лише коли $A \neq B$.

Цей метод простий і швидкий, але стає стійким лише за умови використання довгого випадкового ключа (одноразового полотна).

Після виконання операції XOR користувач може обрати відновлення тексту за допомогою тієї ж функції. Система порівнює отриманий результат із початковим. У разі невідповідності повідомляється про помилку. Додатково, можуть бути задачі з байтовими масивами або «секретним повідомленням», де правильна відповідь підтверджується зворотним застосуванням XOR.

Вибір цього алгоритму в навчальному модулі обумовлений демонстрацією бітових операцій та поняттям симетричного шифрування, що збільшує загальне уявлення учнів про криптографічні методи.

3.3 Діаграма класів

У основі реалізації навчального модуля лежить ієрархічна модель об’єктів, що поєднує загальні механізми навігації та оцінювання з вузькоспрямованими алгоритмами шифрування. Абстрактний клас Lesson визначає базову поведінку кожного уроку: він містить властивості title (назва уроку), currentSection (індекс поточного етапу) і score (накопичені бали).

Через методи showNextSection(), validateInput(input) та completeLesson() забезпечується перехід між теоретичною частиною й практичними завданнями, перевірка коректності введення користувачем і зберігання результатів. Така уніфікована структура дозволяє вбудовувати нові алгоритми, не дублюючи код навігації та логіки підрахунку балів.

Конкретні шифри реалізовано в похідних класах, які перевизначають абстрактні методи для власного алгоритму. У CaesarLesson зосереджено функціонал простого циклічного зсуву символів, у VigenereLesson — послідовного застосування різних зсувів за ключовим словом у поліалфавітному шифрі, BinaryLesson демонструє перетворення тексту у бінарний формат ASCII і навпаки, а XorLesson ілюструє операцію виключного АБО для симетричного шифрування.

Кожний із цих класів містить логіку перевірки відповідності введеного користувачем результату очікуваному та нарахування балів за кожен етап, що сприяє збалансованому освітньому процесу: від простого до складнішого.

Інформація про поточного користувача зберігається в класі User, який має ідентифікатор, контактні дані й масив отриманих досягнень. Самі досягнення описує клас Achievement із властивостями (key, label та description). Зв’язок «один-до-багатьох» між User та Achievement реалізується через службовий компонент AchievementManager (або «службу взаємодії з базою даних»), що відповідає за запис і зчитування прогресу.

Нарешті, клас MainMenu узгоджує графічний інтерфейс: він формує перелік доступних уроків, налаштовує обробку натиснень і відображає вже розблоковані відзнаки, підтримуючи відчуття безперервного прогресу. На рисунку 3.1 наведено діаграму класів системи.

Такий модульний підхід забезпечує не лише легкість розширення (достатньо створити новий спадкоємний клас для додаткового алгоритму), а й підтримку єдиної логіки взаємодії. Використання абстрактного базового класу дозволяє централізовано змінювати загальні процеси навігації або оцінювання, не порушуючи специфічних реалізацій.

Окрім того, чіткий розподіл відповідальностей між компонентами сприяє більшому ступеню тестованості: виправлення або оптимізація одного алгоритму не впливає на роботу інших. Тому слід зауважити, що обрана класова структура гарантує як гнучкість розвитку модуля, розділення логіки, повторне використання коду та масштабованість.

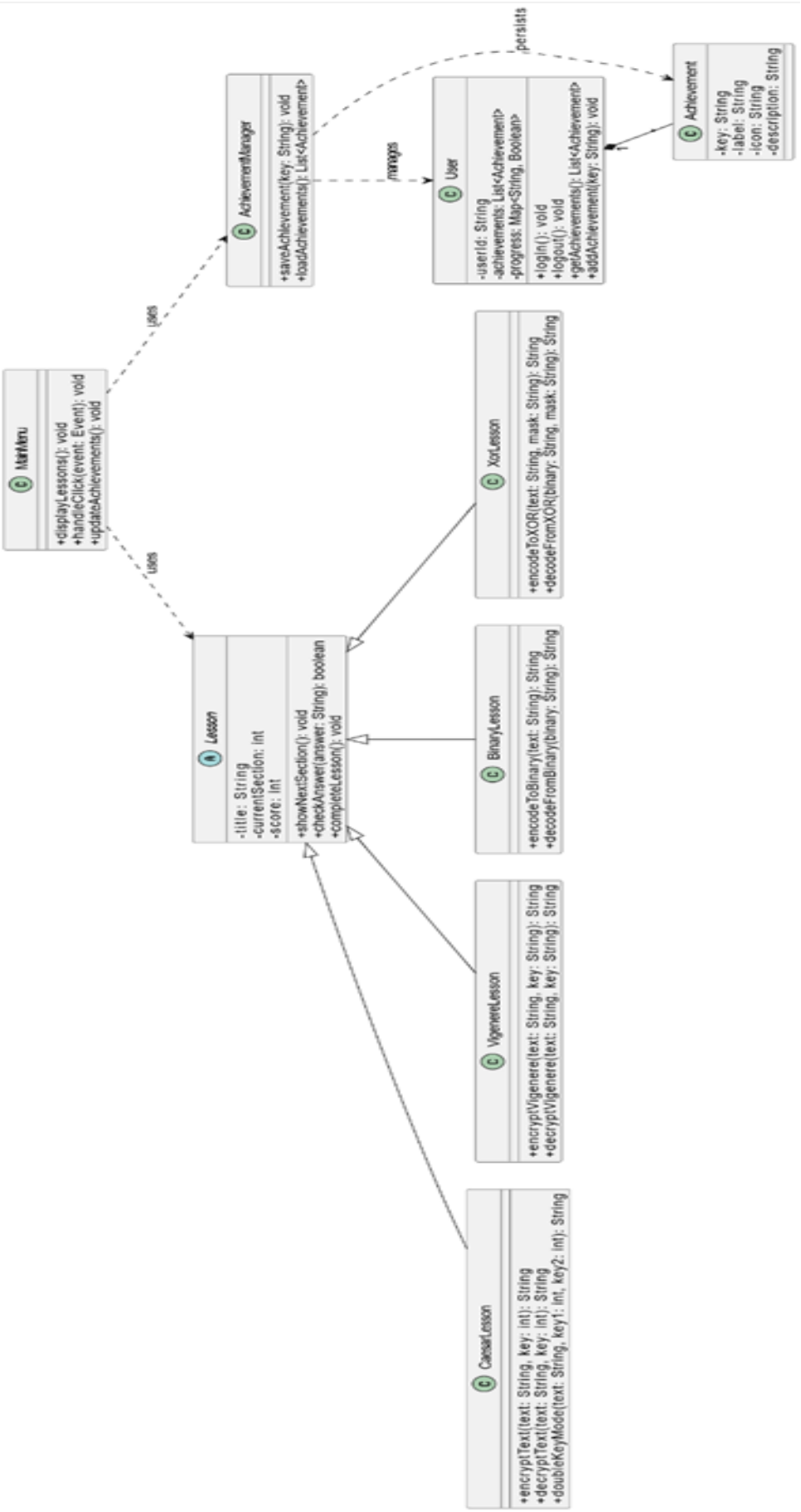


Рис 3.1 Діаграма класів

3.4 Інтерфейс користувача

Інтерфейс програмного модуля розроблено у вигляді низки веб-сторінок з формами для вводу та виводу даних. Інтерфейс кожного уроку реалізовано як інтерактивну веб-сторінку з наступними елементами:

- кнопка «На головну» (повернення в меню) – розташована у верхньому лівому куті. При натисканні відкривається головне меню в новому вікні браузера, що дозволяє користувачу завершити поточний урок;
- кнопка «Правила» – викликає модальне вікно з інструкціями. Воно містить текстове пояснення або підказки щодо того, як виконувати вправи в уроці. Модальне вікно можна закрити кнопкою «Закрити» чи подібним елементом;
- індикатор рахунку і рангу – у верхній частині екрану відображає поточні набрані бали (очок) і рейтинг (ранг) користувача. Значення оновлюються після кожного завдання. Наприклад, після успішної операції шифрування користувач отримує певну кількість балів, а індикатор імені рангу може змінити колір або букву (A, B, C тощо);
- індикатор прогресу – полоса або шкала, яка відображає етап проходження уроку. Наприклад, на початку уроку прогрес низький, і з кожним виконаним завданням смужка наповнюється, візуально показуючи завершеність;
- текстові поля для вводу/виводу – на практичних етапах уроку з’являються поля вводу, куди користувач вводить повідомлення (наприклад, відкритий текст або ключ) та отримує результати шифрування/дешифрування. Поля мають описовий текст для підказки формату вводу;
- кнопки дій – найбільш поширені – «Далі», «Почати практику», «Перевірити», «Завершити урок». Кнопка «Далі» використовується в інформаційних секціях для переходу між вступними частинами уроку. «Почати практику» переводить до активного завдання. Кнопка «Перевірити» запускає перевірку правильності введеного рішення (шифру/дешифрування) та викликає функцію обробки в програмному модулі уроку. У відповідь на перевірку

з’являється текстове повідомлення-зворотний зв’язок («Правильно!» або пояснення помилки) у відведеному для цього полі. Після проходження останнього завдання з’являється кнопка «Завершити урок», що переводить користувача на фінальну сторінку з підсумковим рейтингом;

- Сценарії навігації: уроки побудовані послідовно. Спочатку користувач ознайомлюється з правилами (модальне вікно) і вступом (статичний блок тексту). Коли готовий, він натискає «Почати практику». Далі доступні практичні завдання: після введення даних і натискання «Перевірити» система відображає результат і надає можливість перейти до наступного завдання (кнопка «Далі»). Такий цикл повторюється для всіх етапів. Після завершення останнього завдання з’являється підсумкове повідомлення з оцінкою (рангом) і кнопка для повернення в меню;
- загальний принцип навігації є лінійним і зрозумілим: користувач рухається від вступної частини до практичних блоків і на завершення повертається в меню. Усі кнопки чітко промарковані, а динамічні підказки (повідомлення про помилку) допомагають уникнути невірного вводу та зрозуміти правила шифрування тексту. Для кожного уроку збережено однакову структуру інтерфейсу, відрізняються лише тематичне оформлення (наприклад, оформлення в стилі певної гри чи тематики).

Урок із шифру Цезаря оформлено в ретро-комп’ютерній стилістиці, що одразу налаштовує на атмосферу класичних криптографічних експериментів. Приклад екранних форм уроку «шифр Цезаря» продемонстровано на рисунку 3.2. На початку користувач бачить вступну секцію з коротким поясненням принципу шифрування та єдиною кнопкою «Почати практику», що запускає перше завдання. Саме завдання передбачає два поля для введення: перше — відкритий текст, а друге — числовий ключ (зсув у межах від 0 до 25). Натискання кнопки «Зашифрувати» викликає обробник, який перевіряє коректність введених даних (чи є текст і чи знаходиться число в прийнятному діапазоні), після чого виводить результат у спеціальному полі. Якщо введення некоректне, система видає повідомлення «Введіть текст і ключ!». Після вірного рішення програма збільшує індикатор прогресу та додає бали до лічильника, а кнопка «Далі» стає доступною для

переходу до наступного етапу. Далі користувач виконує зворотне розшифрування, що також перевіряється автоматично, і лише після цього урок завершується підсумковим тестом з трьома запитаннями та виводом загальної кількості набраних балів.

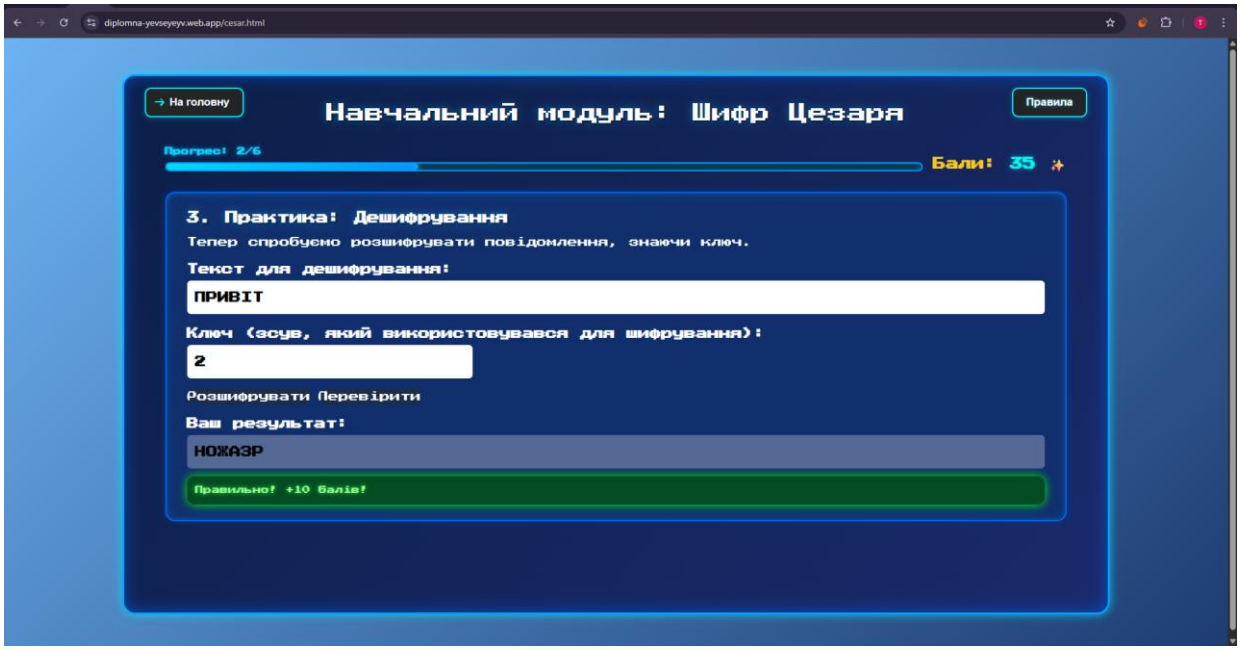


Рис 3.2 Приклад екранних форм уроку «шифр Цезаря»

Урок шифру Віженера продовжує знайомство з криптографічними алгоритмами, виконаний у стилістичному оформлені відеогри «Devil May Cry 3», пропонуючи користувачу готичне оформлення інтерфейсу: темний фон із декоративним шрифтом створює відчуття таємничості, яке властиве серії відеоігор «Devil May Cry», натхнене поемою Данте Аліг’єрі. Приклад екранних форм уроку «шифр Віженера» продемонстровано на рисунку 3.3. Після натискання «Почати практику» відкривається сторінка з двома полями — для відкритого тексту та ключового слова. Ключ автоматично циклічно повторюється до довжини повідомлення. Кнопки «Зашифрувати» та «Розшифрувати» по черзі викликають відповідні функції, що перевіряють правильність дій користувача: у разі успіху спрацьовує функція нарахування балів і змінюється рейтингова панель із позначеннями «S», «A», «B» тощо. Після проходження всіх вправ автоматично

з’являється спливаюче вікно з текстом «Вітаємо, мисливець! Ти пройшов урок у стилі Devil May Cry. Твій фінальний ранг: S» — так реалізовано отримання досягнення. Таким чином у цьому уроці поєднано складнішу технічну логіку поліалфавітного шифрування з наочною візуалізацією процесу й винагородою за успіхи.

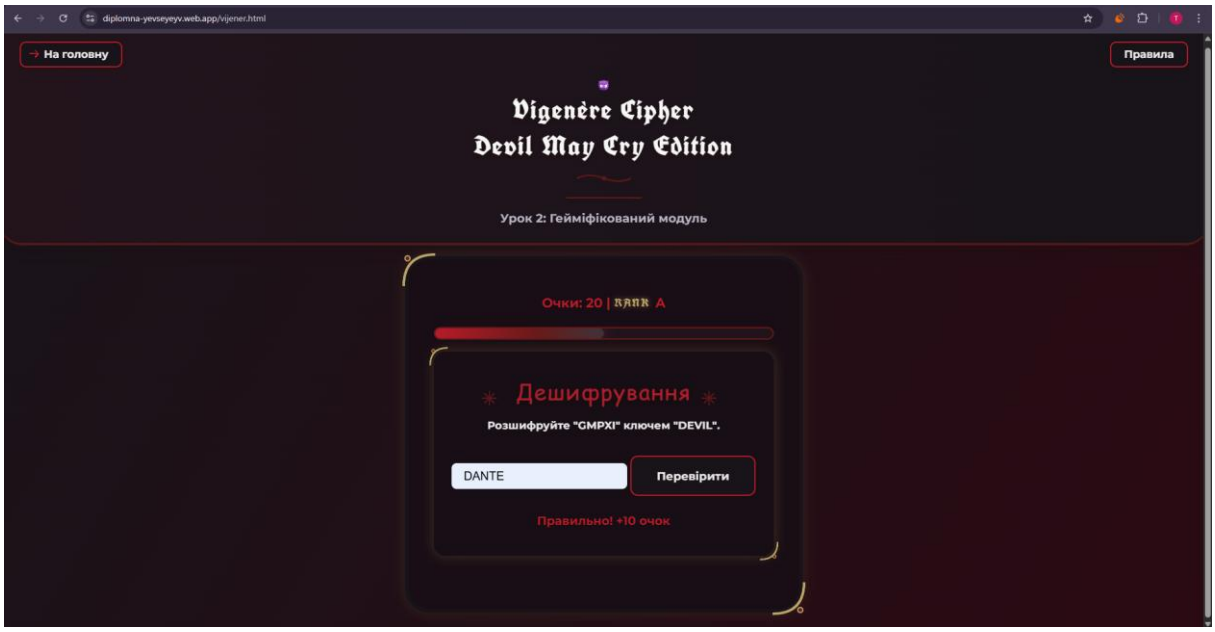


Рис 3.3 Приклад екранних форм уроку «шифр Віженера»

Урок бінарного кодування, виконаний у стилі відеогри «Minecraft», знайомить із перетворенням тексту в двійковий формат і навпаки. Приклад екранних форм уроку «бінарне кодування» продемонстровано на рисунку 3.4. Інтерфейс вирізняється піксельними елементами та кольоровою гамою рудо-зелених відтінків, що нагадують ігровий світ. Основна сторінка має два послідовні блоки: перший — поля для введення текстового повідомлення та кнопка «Кодувати», другий — поле для введення бінарного рядка та кнопка «Декодувати». Після натискання «Кодувати» система обчислює ASCII-код кожного символу, перетворює його в 8-бітний рядок і виводить результат із групуванням по байтах. У разі введення невірного формату — наприклад, букви в бінарному полі — з’являється повідомлення «Невірний формат». За успішне виконання кожного

кроку користувач отримує «досвід» — значення XP, що відображається у вигляді окремої смужки-індикатора.

Такий підхід підкреслює практичність уроку та дозволяє наочно відчути, як працює цифрове кодування.

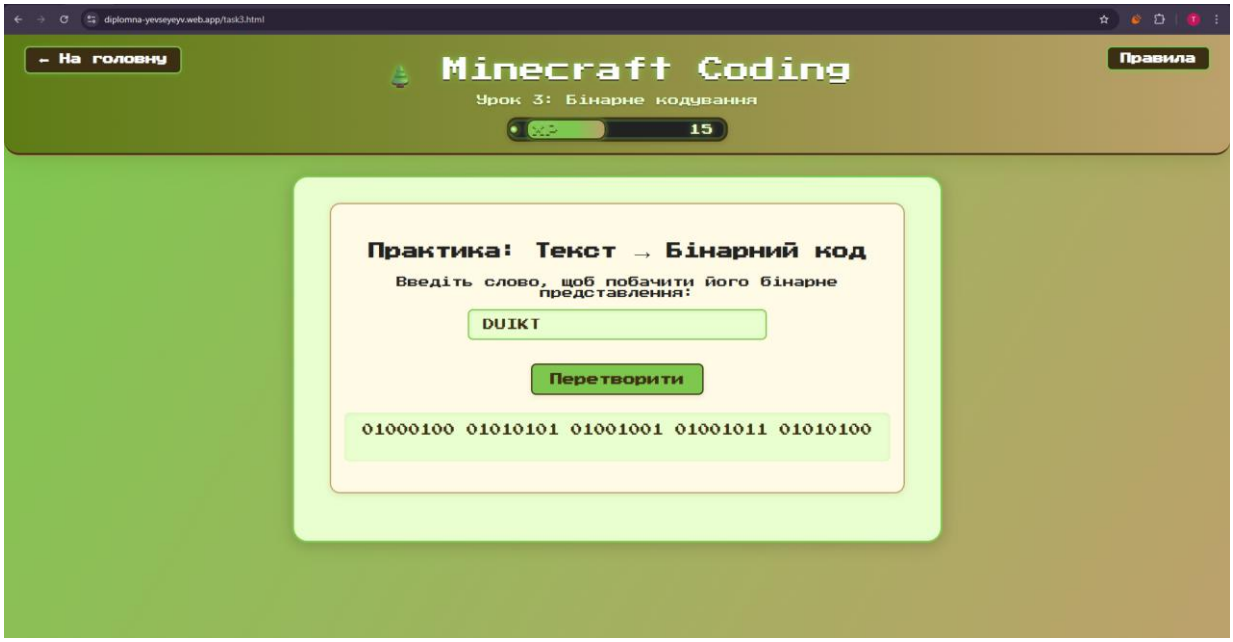


Рис 3.4 Приклад екранних форм уроку «бінарне кодування»

Нарешті, урок шістнадцяткового кодування з використанням побітової операції XOR поєднує концепцію симетричного шифру виконаний у візуальному оформлені відеогри «Cyberpunk 2077» з неоновими елементами кіберпанк-стилю. Приклад екранних форм уроку «шифрування XOR» продемонстровано на рисунку 3.5. Користувач вводить вихідний текст і ключове слово (або символ), після чого кнопка «Зашифрувати» запускає побітову операцію XOR над кожним байтом та перетворює результат у двозначний шістнадцятковий формат. Потім у полі шістнадцяткового коду можна перевірити результат функцією «Розшифрувати»: якщо вихідний текст відновлено вірно, програма нараховує бали й оновлює «RAM-бар» — індикатор, що імітує заповнення оперативної пам’яті. По завершенні обох етапів з’являється підсумковий екран із титулом «XOR-шифрувальник» та кнопкою повернення до меню. Ця структура уроку демонструє властивість

зворотності операції XOR і водночас мотивує користувача через візуальні ефекти та нарахування очок.

Рис 3.5 Приклад екранних форм уроку «шифрування XOR»

Детальний інтерфейс кожного уроку забезпечує плавну взаємодію користувача з системою. Наявність індикаторів прогресу й балів стимулює поступове проходження завдань. Стандартизовані елементи керування (кнопки, поля вводу, повідомлення) роблять користувацький досвід передбачуваним і зручним. Завдяки підказкам і повідомленням про результати користувач отримує негайний зворотний зв’язок, що підвищує ефективність навчання.

Інтерфейс екрану аутентифікації виконано у сучасному неоновому стилі з темним фоном і яскравими жовто-блакитними акцентами, що налаштовують на атмосферу інтерактивного навчання й гейміфікації. У центрі екрана розміщено картку входу/реєстрації з м’якими заокругленими кутами та легким світінням по контуру. Приклад екранних форми аутентифікації представлено на рисунку 3.6.

На початку користувач бачить заголовок модуля CryptoGame: сучасний навчальний модуль та підзаголовок Вивчення сучасних методів шифрування, що підтверджують тему додатку. Нижче — два поля для введення:

- електронна пошта (тип email);
- пароль мінімальна довжина якого 6 символів (тип password).

Під полями — велика кнопка Зареєструватися із градієнтною заливкою. При натисканні на неї система перевіряє коректність введених даних:

- чи відповідає формат електронної пошти стандартному шаблону.
- чи містить пароль щонайменше 6 символів.

Якщо перевірка не пройшла, під формою з’являється повідомлення червоним шрифтом, наприклад: «Введіть коректний email і пароль не менше 6 символів». Успішна реєстрація приводить до автоматичного входу в систему та перенаправлення на головне меню.

Під кнопкою розташоване текстове посилання Вже є акаунт? Увійти, що відкриває форму входу за аналогічною логікою перевірки даних.

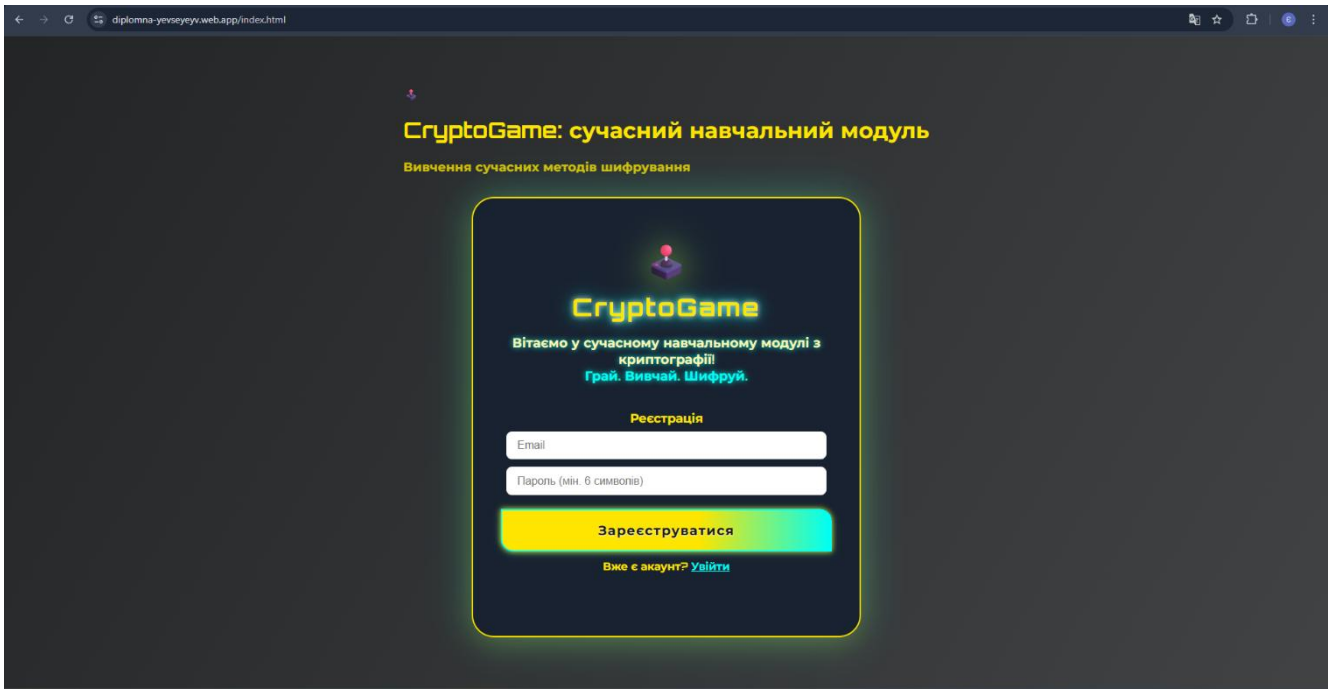


Рис. 3.6 Екранна форма аутентифікації

Головне меню оформлено в стилі футуристичного кабінету з неоновим блакитним заголовком на темному тлі, що створює відчуття цифрової панелі керування. У верхній частині сторінки зліва розміщено логотип модуля, праворуч — адресу електронної пошти користувача та кнопка «вийти». Це дозволяє швидко перевірити поточний акаунт і завершити сеанс.

Приклад екранних форми головного меню представлено на рисунку 3.7.

Центральним елементом є блок вибору завдань із заголовком Оберіть завдання: та кнопкою. Досягнення, що відкриває вікно перегляду здобутих ранґів та балів. Нижче перелічено чотири завдання у вигляді великих неонових кнопок:

- завдання 1: «Шифр Цезаря» (позначка замка вказує на необхідність пройти початкову автентифікацію);
- завдання 2: «Шифр Віженера» (відкрито для виконання після проходження першого);
- завдання 3: «Бінарне кодування (Minecraft)»;
- завдання 4: «Шифрування з ключем».

Кожна кнопка підсвічується при наведенні курсора та викликає відповідний урок. Закриті завдання мають зменшену яскравість і блокуються до виконання попередніх кроків. Під списком завдань розміщено футер із авторським підписом: «© 2025 CryptoGame | Бакалаврська робота Євсєєва Тимофія»

Рис. 3.7 Екрані форми головного меню

3.5 Діаграма активності

Нижче наведено послідовні кроки (приклади сценаріїв) для роботи користувача з кожним із уроків.

Перший сценарій (урок Цезаря):

- крок перший: Користувач відкриває браузерне вікно та заходить на головну сторінку модуля;

крок другий: він вибирає пункт меню «Цезар», відкривається сторінка уроку з полями «Відкритий текст» і «Ключ (зсув)»;

- крок третій: у текстове поле вводиться повідомлення (наприклад, “Привіт”). У поле «Ключ» вводиться число (наприклад, 3);
- крок четвертий: Натискає кнопку «Зашифрувати». JavaScript-скрипт виконує шифрування по алгоритму Цезаря і показує результат у нижньому полі;
- крок п’ятий: система автоматично перевіряє результат за умови, якщо зашифрований текст правильний, з’являється повідомлення «Правильно! +10 балів». В іншому випадку – «Неправильно, спробуйте ще раз»;
- крок шостий: користувач може натиснути «Розшифрувати», щоб повернути текст; при цьому проводиться зворотна перевірка правильності;
- крок сьомий: Після завершення уроку програма додає бали за всі правильно виконані кроки у загальну таблицю результатів та відображає оновлене досягнення (наприклад, «Цезар-майстер») на сторінці «Досягнення».

Другий Сценарій (урок Віженера):

- крок перший: з головної сторінки модулю користувач переходить до уроку «Віженер»;

крок другий: на новій сторінці вводить відкритий текст (наприклад, “Криптографія”) і текстовий ключ (наприклад, “КЛЮЧ”);

- крок третій: натискає «Зашифрувати»: застосовується метод Віженера і виводиться шифротекст у відведене поле;
- крок четвертий: система порівнює результат з очікуваним, якщо співпадає – виводить «Правильно! +10 балів»; якщо ні – «Неправильно»;

- крок п’ятий: може спробувати розшифрувати, натиснувши кнопку «Розшифрувати». Сценарій дешифрування аналогічний;
- крок шостий: по завершенні уроку сумарні бали та інформація про виконані завдання оновлюються в профілі користувача.

Третій Сценарій (урок бінарного кодування):

- крок перший: користувач вибирає урок «Бінарний» («Майнкрафт-бінарний»);
- крок другий: у поле «Текст» вводиться фраза (наприклад, “Hello”);
- крок третій: натискає кнопку «Кодувати»: скрипт переводить кожен символ у двійковий код і показує послідовність бітів;
- крок четвертий: якщо введено правильний текст, відображається повідомлення «Правильно! +10 балів». Інакше з’являється «Невірний переклад»;
- крок п’ятий: користувач потім очищує поле результату та вводить в нього власний бінарний рядок, наприклад, «01001000 01100101 01101100 01101100 01101111»;
- крок шостий: натискає «Декодувати» після чого програма перевіряє, чи цей рядок повертає початковий текст. У разі успіху – знову нараховуються бали.
- крок сьомий: наприкінці бачить загальний бал за цей урок та інформацію про набрані досягнення.

Четвертий Сценарій (урок XOR):

- крок перший: користувач переходить до уроку «XOR» з головного меню;
- крок другий: уводить у поле «Текст» будь-який рядок (наприклад, “ТАЄМНИЙ”). У поле «Ключ» вводить один символ (наприклад, “А”);
- крок третій: натискає «Зашифрувати» після чого система по кожному символу виконує шифрування по ключу і показує результат;
- крок четвертий: перевіряє правильність зашифрованого рядка, отримує повідомлення з балами;

- крок п’ятий: далі вводить отриманий зашифрований текст у поле «Текст» (або очищає поле і вводить його заново), той самий символ ключа “A”, і натискає «Розшифрувати» ;
- крок шостий: переконується, що програма повернула початковий текст, отримує відповідне повідомлення;
- крок сьомий: загальні бали за урок і оновлені досягнення відображаються в інтерфейсі (наприклад, «XOR-шифрувальник»).

Сценарії демонструють крок за кроком, як користувач взаємодіє з кожним модулем, отримуючи миттєвий відгук і бали. Для кращої візуалізації сценарію роботи вебсайту, на рисунку 3.8 наведено діаграму активності. Перевагою таких сценаріїв є наочна послідовність дій та можливість самоконтролю («шифруй – перевіряй – розшифровуй»), що сприяє кращому розумінню матеріалу. Загалом, подання сценарію роботи користувача підкреслює інтерактивність модуля та педагогічну цінність підходу, де кожна дія підтверджується результатом і коментарем системи.

Крім цього, кожен урок реалізовано окремим блоком коду з чітким поділом на процедури шифрування, дешифрування та перевірки коректності введених даних. Реєстрація обробників натискання кнопок «Зашифрувати» і «Розшифрувати» здійснюється через функції-слухачі подій, а всі дії з оновлення інтерфейсу — від показу повідомлень до зміни смужки прогресу та підрахунку балів — винесено у допоміжний скрипт.

Збереження результатів у хмарній базі організовано за допомогою асинхронних запитів із блоками «спроба–перехоплення» для обробки помилок, що у разі збою виводить «Помилка з’єднання із сервером». Такий підхід забезпечує єдину точку контролю для перевірки даних і спрощує додавання нових уроків.

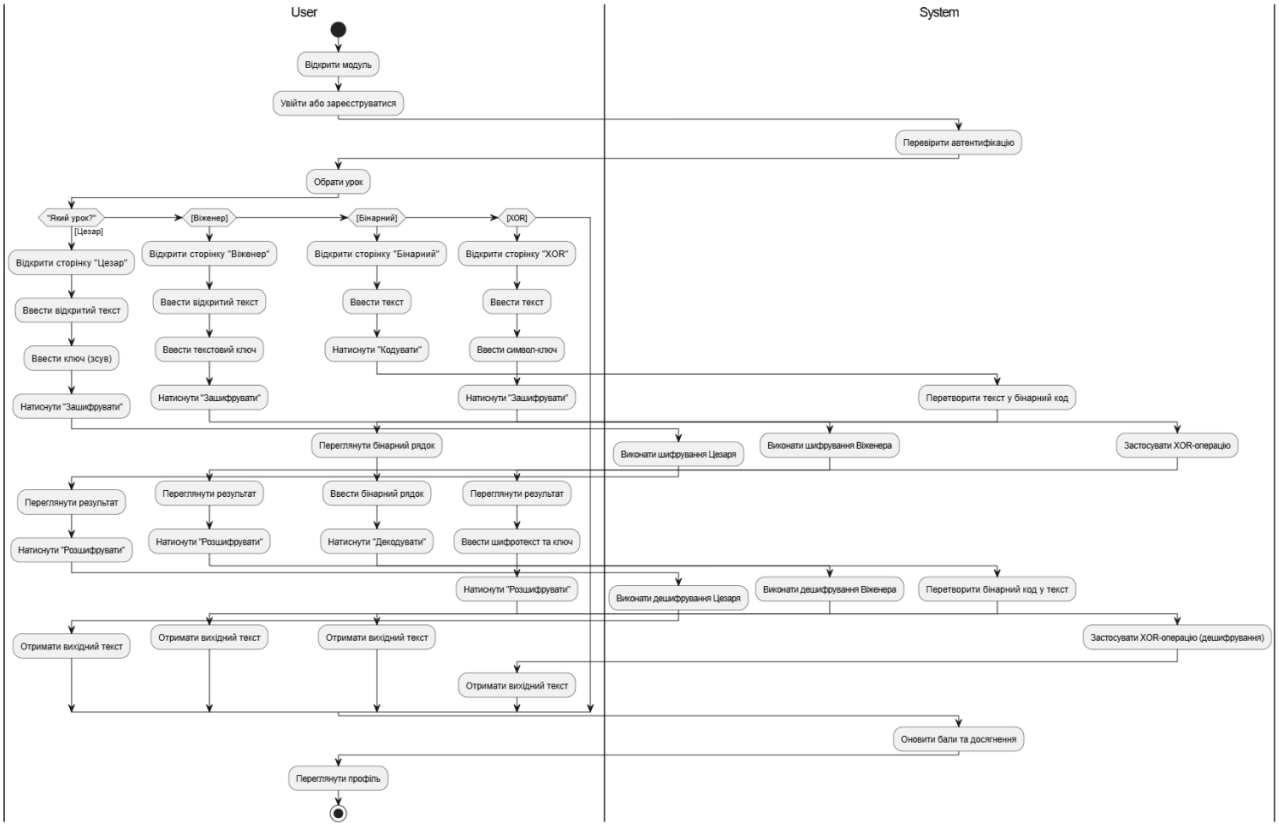


Рис 3.8 Діаграма активності

3.6 Інтеграція з Firebase

Для авторизації користувачів застосовано сервіс Firebase Authentication. Підтримується реєстрація та вхід за email/паролем, а також вхід через обліковий запис Google. У коді це реалізовано за допомогою методів Firebase Auth (файл firebase.js): функції createUserWithEmailAndPassword та signInWithEmailAndPassword для класичного логіна і реєстрації, і використання GoogleAuthProvider для входу через Google. Зокрема, при натисканні кнопки входу через Google створюється провайдер new GoogleAuthProvider(), після чого викликається signInWithPopup(auth, provider), що відкриває вікно авторизації через обліковий запис Google. Успішний вхід повертає об’єкт user, який містить унікальний ідентифікатор (UID) та електронну пошту користувача. Змінюючи стан авторизації, обробник onAuthStateChanged вмикається автоматично – він викликає функцію loadProgress() (для завантаження раніше збережених даних) або оновлює інтерфейс користувача при виході/вході.

Після успішного входу через Google або email, у коді використовується UID поточного користувача (auth.currentUser.uid) як ключ для всіх операцій із базою даних Firestore. Наприклад, при виконанні завдання викликається функція saveAchievementToFirestore(achievementId), яка отримує поточного користувача і за допомогою setDoc(doc(db, "users", UID), { achievements: arrayUnion(achievementId) }, { merge: true }) зливає нове досягнення у масив achievements у колекції users. Таким чином кожний результат записується саме в документі відповідного користувача, і при наступних досягненнях старі значення не перезаписуються, а доповнюються.

3.7 Хостинг вебсайту

Вебсайт розгорнуто через Firebase Hosting – хмарну послугу Google для швидкого та безпечного розміщення статичних ресурсів (HTML, CSS, JavaScript, зображення тощо) [14]. Конфігурація хостингу міститься у файлі firebase.json: тут вказано публічну директорію "public": "public", куди зібрані всі статичні файли, та правилом "rewrites" перенаправлено кореневий запит ("/") на файл welcome.html. Файл .firebaserc містить ідентифікатор проекту ("diplomna-yevseyeyv"). При публікації Firebase Hosting використовує глобальну CDN з SSL по замовчуванню) [14]. Тобто сторінки сервісу web.app та firebaseapp.com відображаються у браузері з HTTPS.

Процес публікації та оновлення додатку автоматизовано через CI/CD на базі GitHub Actions. Команди Firebase CLI згенерували два YAML-файли у каталозі .github/workflows: один для пушів у гілку main (firebase-hosting-merge.yml) і один для pull request’ів (firebase-hosting-pull-request.yml). При кожному мерджі в main відбувається автоматичне розгортання поточної версії додатку на Firebase Hosting. При відкритті pull request створюється попередній канал (preview), у якому зміни розгортаються для тестування (GitHub Action додає коментар з URL попереднього перегляду) Завдяки цьому кожен коміт автоматично деплоїться у відповідний канал, оновлюючи застосунок без ручних дій [15]. Використовуються секрети (FIREBASE_SERVICE_ACCOUNT_*) для безпечного доступу CI до проєкту Firebase. Таким чином нова версія додатку потрапляє у продакшн одразу після злиття гілок, а pull request’и автоматично перевіряються на працездатність.

3.8 Тестування вебсайту

Для перевірки коректності роботи реалізовано кілька видів тестування. Функціональне тестування полягало в перевірці кожного основного сценарію застосунку: реєстрації/логіна користувача, роботи шифрів, нарахування балів та присвоєння досягнень. Наприклад, було перевірено сценарій: користувач успішно шифрує текст за ключем, після чого очікувано змінюється рахунок (score), а при виконанні всіх завдань блок Цезаря відмічається як пройдений і додається відповідне досягнення.

Валідація введення полягала в контролі коректності введених даних у шифрах: якщо введено недопустимі символи або порожні поля, то додаток формує попередження або ігнорує введення замість аварійного завершення. Наприклад, при спробі ввести цифри у полі для тексту шифрування виводиться повідомлення про помилку або символи просто ігноруються. Тест на стійкість до помилок включав перевірку поведінки при відсутності мережі: у разі відключення інтернет-з’єднання під час збереження прогресу в Firestore викликається обробка помилки в консолі (catch-блок), але інтерфейс лишається стабільним без зависань. Також перевірено, що незареєстрований користувач (без логіна) при натисненні кнопки «Зберегти прогрес» не викликає критичної помилки, а просто нічого не відбувається (через перевірку if (!user) return).

У результаті тестування встановлено, що ключовий функціонал відповідає очікуванням. Наприклад, при тестах шифру Цезаря було підтверджено, що для тексту «КИЇВ» з ключем 1 результатом стає «ЛІЙГ» (зсув на один символ): це збігається з математичною моделлю шифру. Аналогічно, для шифру Віженера з певним ключем розшифровування відтворює вихідний текст.

Результати тестів (успішні та помилкові випадки) зафіксовані у вигляді коротких описів кроків, а також вручну внесені до протоколу (таблиці) перевірки функціоналу та валідності введення. Наявні приклади: успішне розпізнавання неправильно введеного символу в полі вводу, коректне застосування arrayUnion для додавання нових досягнень без дублювання, і реакція інтерфейсу на натискання

кнопки «Вихід» при відсутності логіна (коректне завершення сеансу). Усі ці сценарії показали, що реалізація працює на практиці стабільно.

Крім зазначених вище перевірок, важливе місце в тестуванні зайняло тестування продуктивності клієнтської частини. Для цього було виміряно час виконання алгоритмів шифрування й дешифрування на текстах різного обсягу – від сотень до кількох тисяч символів. Результати показали, що навіть при обробці 5 000 знаків кожен алгоритм виконується за 20–50 мілісекунд у стандартних умовах браузера, що свідчить про належну оптимізацію коду та відсутність “затримок” під час роботи користувача.

Не менш важливим етапом стало регресійне тестування після внесення виправлень та розширення функціоналу. Після кожного оновлення код перевірявся на наборі сценаріїв, збережених у протоколі: від базових випадків шифрування до обробки нестандартних вводів. Всі тести успішно пройшлися, підтверджуючи, що виправлення в одній частині системи не призвели до несподіваних збоїв в інших модулях.

Такий підхід до тестування гарантує стабільність роботи застосунку в різних умовах експлуатації та служить підґрунтям для його подальшого розвитку.

ВИСНОВКИ

1. Проведено аналіз обраної предметної галузі в сфері навчання криптографічним методам. Визначено основні потреби користувачів у зручному та інтерактивному освітньому середовищі.

2. Проведено аналіз програмного забезпечення, що існує для вивчення криптографічних методів. Визначено основні функції, переваги та недоліки наявних рішень, які були враховані при розробці навчального модуля.

3. Досліджено сучасні підходи до гейміфікації освітнього процесу. Визначено ефективні механіки гейміфікації, що були реалізовані у створеному модулі для підвищення мотивації користувачів.

4. Визначено функціональні та нефункціональні вимоги до навчального модуля. Проведено моделювання за допомогою діаграм варіантів використання.

5. Проведено аналіз засобів програмної реалізації. Обрано відповідні інструменти та технології для реалізації вебсайту згідно з поставленими вимогами.

6. Розроблено основні компоненти навчального модулю, включаючи модуль завдань, елементи гейміфікації та інтерфейс вебсайту, відповідно до визначених вимог.

7. Реалізовано функціонал створення акаунту користувача, що дозволяє зберігати прогрес навчання та персоналізувати взаємодію з платформою.

8. Робота пройшла апробацію:

1. Євсєєв Т.Є Проблематика створення навчального модуля з елементами гейміфікації для вивчення методів шифрування // II Всеукраїнська науково-практична конференція «Цифрова гуманістика: Інформаційні технології та інформаційне моделювання на сучасному етапі розвитку суспільства». Збірник тез 22-23 травня 2025 року, ЦДУ імені В. Винниченка, м. Кропивницький (подано до друку).

2. Євсєєв Т.Є, Аброскін Ю.Ю Розробка навчального модуля з елементами гейміфікації для вивчення методів шифрування // IV Міжнародна науково-практична конференція «Інформаційні технології в освіті та науці». Збірник тез 20 травня 2025 року, МДПУ, м. Запоріжжя (подано до друку);

ПЕРЕЛІК ПОСИЛАНЬ

1. Волошена В. Гейміфікація як інтерактивний засіб навчання математики. Problems of the modern textbook. 2024. № 33. С. 57–67. URL: https://doi.org/10.32405/2411-1309-2024-33-57-67 (дата звернення: 19.05.2025).

2. Communication processes security using visual cryptography / I. Rymchuk et al. Cybersecurity: education, science, technique. 2024. Vol. 2, no. 26. P. 258–267. URL: https://doi.org/10.28925/2663-4023.2024.26.677 (date of access: 19.05.2025).

3. Experiential learning: experience as the source of learning and development. Englewood Cliffs, N.J : Prentice-Hall, 1984. 256 p.

4. Кушнірук С., Ігнатенко І. Переваги гейміфікації інтерактивно-освітнього контенту стосовно підготовки до професійної діяльності викладача в сучасних умовах. Перспективи та інновації науки. 2025. № 2(48). URL: https://doi.org/10.52058/2786-4952-2025-2(48)-603-613 (дата звернення: 19.05.2025).

5. Intrinsic motivation and self-determination in human behavior / ed. by R. R. M. New York : Plenum, 1985. 371 p.

6. From game design elements to gamefulness / S. Deterding et al. The 15th international academic mindtrek conference, Tampere, Finland, 28–30 September 2011. New York, New York, USA, 2011. URL: https://www.researchgate.net/publication/230854710_From_Game_Design_Elements_to_Gamefulness_Defining_Gamification (date of access: 19.05.2025).

7. Digital interactive learning technologies as an integral component of the modern educational process / O. V. Skliarenko et al. Innovate pedagogy. 2024. Vol. 2, no. 68. P. 250–256. URL: https://doi.org/10.32782/2663-6085/2024/68.2.51 (date of access: 19.05.2025).

8. Agile Education and Sustainable Competencies. Computing Curricula 2020: Paradigms for Global Computing Education / CC2020 Task Force. Association for Computing Machinery, New York, NY, United States, 2021. С. 186– 195. URL: https://dl.acm.org/doi/book/10.1145/3467967 (дата звернення: 19.05.2025).

9. Кривонос О. М., Мінгальова Ю. І., Кривонос М. П., Стельмашенко Я. А., Махенько Я. Д. Методика вивчення розділу «парадигми та технології програмування» [Електронний ресурс]. Академічні візії. 2023. Вип. 23. URL: https://eprints.zu.edu.ua/37972/1/document.pdf. (Дата перегляду: 19.05.2025).

10. Zourmpakis A.-I., Kalogiannakis M., Papadakis S. Adaptive gamification in science education: an analysis of the impact of implementation and adapted game elements on students' motivation. Computers. 2023. Vol. 12, no. 7. P. 143. URL: https://doi.org/10.3390/computers12070143 (date of access: 19.05.2025).

11. GeeksforGeeks. Client-Server Model - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/client-server-model/ (date of access: 21.05.2025).

12. Caesar cipher in cryptography - geeksforgeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/ (date of access: 19.05.2025).

13. Vigenère cipher in cryptography. Tutorials on Technical and Non Technical Subjects. URL: https://www.tutorialspoint.com/cryptography/cryptography_vigenere_cipher.htm#:~:text=An%20algorithm%20called%20the%20Vigenere,It%20is (date of access: 22.05.2025).

14. Get started with Firebase Hosting. Firebase. URL: https://firebase.google.com/docs/hosting/quickstart#:~:text=Firebase%20Hosting%20gives%20you%20a,dynamic%20content%20and%20host%20microservices (date of access: 19.05.2025).

15. Deploy to live & preview channels via GitHub pull requests | Firebase Hosting. Firebase. URL: https://firebase.google.com/docs/hosting/github-integration#:~:text=,change%20with%20each%20new%20commit (date of access: 19.05.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка навчального модуля з елементами геймифігації для вивчення методів шифрування

Виконав студент 4 курсу
групи ПД-42
Євсєєв Тимофій Євгенович
Керівник роботи
асистент кафедри Аброскін Юрій Юрійович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – підвищення зацікавленості до вивчення криптографічних методів, шляхом навчання з елементами гейміфікації.
- **Об’єкт дослідження** – процес навчання криптографічним методам у сфері інформаційних технологій.
- **Предмет дослідження** – засоби програмної реалізації гейміфікованого навчального середовища для викладання криптографії.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз обраної предметної галузі в сфері навчання криптографічним методам.
2. Порівняння існуючого програмного забезпечення, для вивчення криптографічних методів.
3. Дослідити засоби гейміфікації.
4. Визначити функціональні та нефункціональні вимоги.
5. Провести аналіз засобів програмної реалізації.
6. Розробити основні компоненти веб сайту відносно до вимог.
7. Реалізувати функціонал створення аккаунту.

3

АНАЛІЗ АНАЛОГІВ

Показник	Coursera	Cryptohack	CryptoZombies	CryptoGame
Підтримка української мови	Окремі уроки	-	-	+
Елементи гейміфікації	-	+	+	+
Відкритий доступ	-	+	+	+
Доступність на платформах	Web, IOS, Android	Web	Web	Web

4

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:
1. Реєстрація та автентифікація користувача.
2. Проходження інтерактивних навчальних блоків.
3. Система балів, досягнень і рівнів (гейміфікація).
4. Виконання практичних завдань на шифрування/дешифрування.
5. Збереження прогресу користувача.

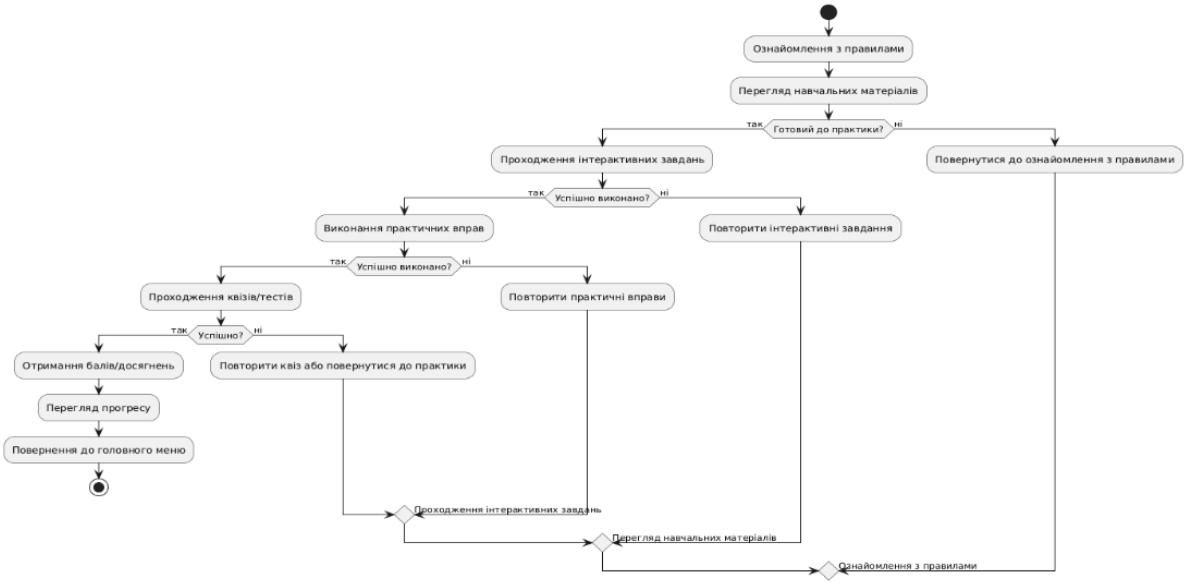
Нефункціональні вимоги:
1. Оформлення інтерфейсу у вигляді комп'ютерних ігор.
2. Адаптивність інтерфейсу до різних браузерів.
3. Продуктивність і швидкість завантаження до 1 секунд.
4. Автоматична підтримка масштабування без власного сервера

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

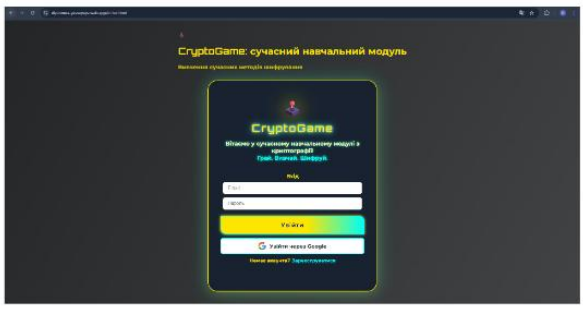
6

ДІАГРАМА ДІЯЛЬНОСТІ

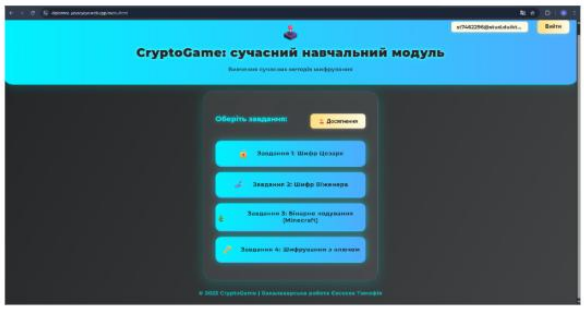


7

ЕКРАННІ ФОРМИ



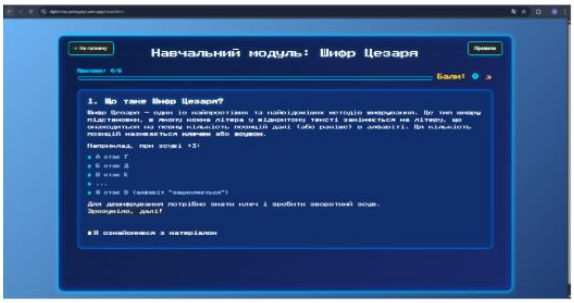
Реєстрація користувача



Головне меню

8

ЕКРАННІ ФОРМИ

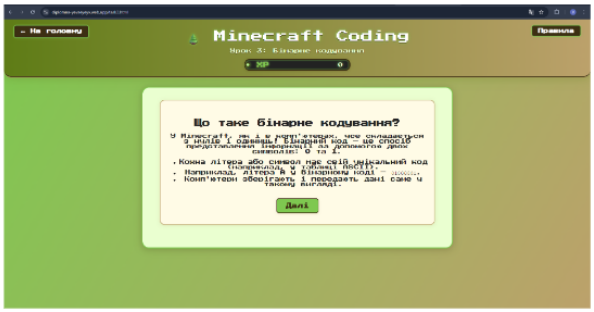


Перший урок

Другий урок

9

ЕКРАННІ ФОРМИ



Третій урок

Четвертий урок

10

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Євсєєв Т.Є, Аброскін Ю.Ю Розробка навчального модуля з елементами гейміфікації для вивчення методів шифрування // IV Міжнародна науково-практична конференція «Інформаційні технології в освіті та науці». Збірник тез 20 травня 2025 року, МДПУ, м.Запоріжжя(подано до друку);
2. Євсєєв Т.Є Проблематика створення навчального модуля з елементами гейміфікації для вивчення методів шифрування // II Всеукраїнська науково-практична конференція «Цифрова гуманістика: Інформаційні технології та інформаційне моделювання на сучасному етапі розвитку суспільства». Збірник тез 22-23 травня 2025 року, ЦДУ імені В. Винниченка, м.Кропивницький(подано до друку).

11

ВИСНОВКИ

1. Проведено аналіз обраної предметної галузі в сфері навчання криптографічним методам. Визначено основні потреби користувачів у зручному та інтерактивному освітньому середовищі.
2. Проведено аналіз програмного забезпечення, що існує для вивчення криптографічних методів. Визначено основні функції, переваги та недоліки наявних рішень, які були враховані при розробці навчального модуля.
3. Досліджено сучасні підходи до гейміфікації освітнього процесу. Визначено ефективні механіки гейміфікації, що були реалізовані у створеному модулі для підвищення мотивації користувачів.
4. Визначено функціональні та нефункціональні вимоги до навчального модуля. Проведено моделювання за допомогою діаграм варіантів використання.
5. Проведено аналіз засобів програмної реалізації. Обрано відповідні інструменти та технології для реалізації вебсайту згідно з поставленими вимогами.
6. Розроблено основні компоненти навчального модулю, включаючи модуль завдань, елементи гейміфікації та інтерфейс вебсайту, відповідно до визначених вимог.
7. Реалізовано функціонал створення акаунту користувача, що дозволяє зберігати прогрес навчання та персоналізувати взаємодію з платформою.

12

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
// File: firebase.js
// Підключення Firebase SDK (додай ці скрипти у <head> HTML):
// <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-app-compat.js"></script>
// <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-auth-compat.js"></script>
// <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-firestore-compat.js"></script>

// 1. Ініціалізація Firebase (замініть на свої дані з Firebase Console)
const firebaseConfig = {
  apiKey: "AIzaSyAtu2-XCqHs-zVvUC9nQTUypesY7_CFsUw",
  authDomain: "diplomna-yevseyeyv.firebaseapp.com",
  projectId: "diplomna-yevseyeyv",
  storageBucket: "diplomna-yevseyeyv.appspot.com",
  messagingSenderId: "291283814406",
  appId: "1:291283814406:web:c89b1f865c806cf4b8db87",
  measurementId: "G-2ENF6226HM"
};
firebase.initializeApp(firebaseConfig);
const auth = firebase.auth();
const db = firebase.firestore();

// 2. Реєстрація
function register(email, password) {
  return auth.createUserWithEmailAndPassword(email, password);
}

// 3. Логін
function login(email, password) {
  return auth.signInWithEmailAndPassword(email, password);
}

// 4. Вихід
function logout() {
  return auth.signOut();
}

// 5. Зберегти прогрес (progressObj — будь-який об’єкт)
function saveProgress(progressObj) {
  const user = auth.currentUser;
  if (!user) return Promise.reject('Not logged in');
  return
function renderAchievementsModal(achievementsArr = []) {
  let html = '<h2 style="font-size:1.4em;margin-bottom:1em;">Ваші досягнення</h2>';
  html += '<div>';
  for (const a of ACHIEVEMENTS) {
    const completed = achievementsArr.includes(a.key);
    html += `<div class="achievement-item${completed ? ' completed' : ''}"><span class="icon">${a.icon}</span> ${a.label} ${completed ? '✅' : ''}</div>`;
  }
  html += '</div>';
  document.getElementById('achievements-modal-content').innerHTML = html;
}
// --- Відкриття/закриття модального вікна ---
document.getElementById('achievements-btn').addEventListener('click', async () => {
  let achievementsArr = [];
  if (auth.currentUser) {
    const userRef = doc(db, "users", auth.currentUser.uid);
    const snap = await getDoc(userRef);
    if (snap.exists()) {
      achievementsArr = snap.data().achievements || [];
    }
  }
  renderAchievementsModal(achievementsArr);
  const overlay = document.getElementById('achievements-modal-overlay');
  overlay.style.display = 'flex';
  setTimeout(() => { overlay.style.opacity = 1; }, 10);
});
document.getElementById('achievements-modal-close').addEventListener('click', () => {
  const overlay = document.getElementById('achievements-modal-overlay');
  overlay.style.opacity = 0;
  setTimeout(() => { overlay.style.display = 'none'; }, 400);
});

// Початковий запуск
showSection(1);
updateProgress();
updateScore(0);

function isValidUAText(text) {
  // Дозволяємо лише українські літери, пробіли, розділові знаки (без цифр, латиниці, незалежно від регістру)
  // Латиниця: a-zA-Z, цифри: 0-9
  // Українські літери: АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ (та малі)
  // Дозволені знаки: пробіл, . , ! ? " ' -
```

```
db.collection('progress').doc(user.uid).set(progressObj);
}

// 6. Завантажити прогрес
function loadProgress() {
  const user = auth.currentUser;
  if (!user) return Promise.reject('Not logged in');
  return
db.collection('progress').doc(user.uid).get().then(doc =>
doc.exists ? doc.data() : null);
}

// 7. Відстеження авторизації (автоматичне
завантаження прогресу)
auth.onAuthStateChanged(user => {
  if (user) {
    // Користувач увійшов
    loadProgress().then(progress => {
      // Тут можна оновити UI, наприклад:
      // updateUIWithProgress(progress);
      console.log('Progress loaded:', progress);
    });
  } else {
    // Користувач вийшов
    // updateUIForLoggedOut();
    console.log('User logged out');
  }
});

// 8. Приклад використання:
// register('test@email.com', '123456').then(...)
// login('test@email.com', '123456').then(...)
// saveProgress({ cesar: { done: true, score: 50 } })
// loadProgress().then(progress => ...)

async function setAchievement(key) {
  console.log('setAchievement викликано', key);
  if (typeof saveAchievementToFirestore === 'function')
await saveAchievementToFirestore(key);
}

// File: script_achievements.js
// --- Досягнення для головного меню ---
const ALL_ACHIEVEMENTS = [
  // Caesar
  { key: 'cesar_all_done', label: 'CAESAR MASTER —
Пройдено урок шифру Цезаря', icon: '🏛️', desc:
'Ретро-комп’ютерна класика. Caesar Cipher (стиль:
Retro Computer)' },
  // Vigenere
  { key: 'vijener_all_done', label: 'VIGENERE
HUNTER — Пройдено урок шифру Віженера', icon:
'🩸', desc: 'Готичний мисливець. Vigenere Cipher
(гра: Devil May Cry)' },
  // Minecraft
  { key: 'mc_all_done', label: 'MINECRAFT
REDSTONE — Пройдено Minecraft-урок', icon: '⛏️',
desc: 'Блоки, XP та редстоун. Minecraft Binary (гра:
Minecraft)' },
  // Cyberpunk
  { key: 'cp_all_done', label: 'CYBERPUNK
  return
/^[АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯа-
яґєіїйюя .,!?"'\-]+$/.test(text);
}

// File: script_hex.js
import { initializeApp } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";
import { getAuth } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";
import { getFirestore, doc, setDoc, arrayUnion, getDoc }
from "https://www.gstatic.com/firebasejs/11.6.1/firebase-
firestore.js";

const firebaseConfig = {
  apiKey: "AIzaSyAtu2-XCqHs-
zVvUC9nQTUypesY7_CFsUw",
  authDomain: "diplomna-yevseyeyv.firebaseapp.com",
  projectId: "diplomna-yevseyeyv",
  storageBucket: "diplomna-yevseyeyv.firebasestorage.app",
  messagingSenderId: "291283814406",
  appId: "1:291283814406:web:c89b1f865c806cf4b8db87",
  measurementId: "G-2ENF6226HM"
};
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);

async function saveAchievementToFirestore(achievementId)
{
  console.log('saveAchievementToFirestore викликано',
achievementId, auth.currentUser);
  const user = auth.currentUser;
  if (!user) return;
  const userRef = doc(db, "users", user.uid);
  try {
    await setDoc(userRef, { achievements:
arrayUnion(achievementId) }, { merge: true });
  } catch (e) {
    console.error("Не вдалося зберегти ачивку:", e);
  }
}

let cpRAM = 0;
const cpRAMMax = 8;
function updateRAMBar() {
  const percent = Math.min(100, Math.round((cpRAM /
cpRAMMax) * 100));
  document.getElementById('cp-ram-bar').style.width =
percent + '%';
  document.getElementById('cp-ram-value').textContent =
cpRAM + '/' + cpRAMMax;
}
function addRAM(amount) {
  cpRAM = Math.min(cpRAM + amount, cpRAMMax);
  updateRAMBar();
}
function nextSection(next) {
  document.querySelectorAll('.cp-section').forEach(s =>
s.classList.remove('visible'));
  document.getElementById('cp-section-' +
next).classList.add('visible');
```

```
COMPLETE — Пройдено кіберпанк-урок', icon:
'💾', desc: 'Неоновий світ RAM та XOR. Cyberpunk
2077 (гра: Cyberpunk 2077)' },
];

import { initializeApp } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-
app.js";
import { getAuth, onAuthStateChanged } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-
auth.js";
import { getFirestore, doc, getDoc, setDoc, arrayUnion
} from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-
firestore.js";

const firebaseConfig = {
  apiKey: "AIzaSyAtu2-XCqHs-
zVvUC9nQTUypesY7_CFsUw",
  authDomain: "diplomna-yevseyeyv.firebaseapp.com",
  projectId: "diplomna-yevseyeyv",
  storageBucket: "diplomna-
yevseyeyv.firebasestorage.app",
  messagingSenderId: "291283814406",
  appId:
"1:291283814406:web:c89b1f865c806cf4b8db87",
  measurementId: "G-2ENF6226HM"
};
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);

async function getUserAchievements() {
  const user = auth.currentUser;
  if (!user) return [];
  const userRef = doc(db, "users", user.uid);
  const snap = await getDoc(userRef);
  if (!snap.exists()) return [];
  return snap.data().achievements || [];
}

function renderAchievementsModal(achievementsArr =
[]) {
  let html = '<h2 style="font-size:1.4em;margin-
bottom:1em;">Ваші досягнення</h2>';
  html += '<div>';
  for (const a of ALL_ACHIEVEMENTS) {
    const completed = achievementsArr.includes(a.key);
    html += `<div class="achievement-item${completed
? ' completed' : ''}"><span
class="icon">${a.icon}</span>
<b>${a.label}</b><br><span
class="desc">${a.desc}</span> ${completed ? '✅' :
''}</div>`;
  }
  html += '</div>';
  document.getElementById('achievements-modal-
content').innerHTML = html;
}

// --- Відкриття/закриття модального вікна ---
document.getElementById('achievements-
  // Досягнення та RAM
  if (next === '2') { setCpAchievement('cp_intro');
addRAM(1); }
  if (next === '3') { setCpAchievement('cp_encode');
addRAM(2); }
  if (next === '4') { setCpAchievement('cp_decode');
addRAM(2); }
  if (next === '5') { setCpAchievement('cp_all_done');
addRAM(3); }
}
function isValidASCIIBasic(text) {
  // Дозволяємо лише базові ASCII-символи (коди 32-126)
  return /^[\x20-\x7E]+$/.test(text);
}
function encodeToXOR() {
  const text = document.getElementById('cp-xor-text-
input').value.trim();
  const key = document.getElementById('cp-xor-key-
input').value.trim();
  if (!text || !key) {
    document.getElementById('cp-xor-output').textContent =
'Введіть текст і ключ!';
    return;
  }
  if (!isValidASCIIBasic(text) || !isValidASCIIBasic(key)) {
    document.getElementById('cp-xor-output').textContent =
'Використовуйте лише латиницю, цифри та базові
символи!';
    return;
  }
  let hex = Array.from(text).map((ch, i) => {
    const k = key[i % key.length];
    const xor = ch.charCodeAt(0) ^ k.charCodeAt(0);
    return xor.toString(16).toUpperCase().padStart(2, '0');
  }).join(' ');
  document.getElementById('cp-xor-output').textContent =
hex;
  setCpAchievement('cp_encode');
  addRAM(2);
  setTimeout(() => nextSection('3'), 900);
}
function decodeFromXOR() {
  const hex = document.getElementById('cp-xor-hex-
input').value.trim();
  const key = document.getElementById('cp-xor-key2-
input').value.trim();
  if (!hex || !key) {
    document.getElementById('cp-xor-decode-
output').textContent = 'Введіть HEX і ключ!';
    return;
  }
  if (!isValidASCIIBasic(key)) {
    document.getElementById('cp-xor-decode-
output').textContent = 'Використовуйте лише латиницю,
цифри та базові символи у ключі!';
    return;
  }
  let text = '';
  try {
    const hexArr = hex.split(' ');
    text = hexArr.map((h, i) => {
      const k = key[i % key.length];
```

```
btn').addEventListener('click', async () => {
  let achievementsArr = [];
  if (auth.currentUser) {
    achievementsArr = await getUserAchievements();
  }
  renderAchievementsModal(achievementsArr);
  const overlay =
document.getElementById('achievements-modal-
overlay');
  overlay.style.display = 'flex';
  setTimeout(() => { overlay.style.opacity = 1; }, 10);
});
document.getElementById('achievements-modal-
close').addEventListener('click', () => {
  const overlay =
document.getElementById('achievements-modal-
overlay');
  overlay.style.opacity = 0;
  setTimeout(() => { overlay.style.display = 'none'; },
400);
});

async function
saveAchievementToFirestore(achievementId) {
  const user = auth.currentUser;
  if (!user) return;
  const userRef = doc(db, "users", user.uid);
  try {
    await setDoc(userRef, { achievements:
arrayUnion(achievementId) }, { merge: true });
  } catch (e) {
    console.error("Не вдалося зберегти ачивку:", e);
  }
}

// Приклад: викликайте цю функцію при отриманні
ачивки
// saveAchievementToFirestore('назва_ачивки');

// File: script_cesar.js
import { initializeApp } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-
app.js";
import { getAuth } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-
auth.js";
import { getFirestore, doc, setDoc, arrayUnion, getDoc
} from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-
firestore.js";

const firebaseConfig = {
  apiKey: "AIzaSyAtu2-XCqHs-
zVvUC9nQTUypesY7_CFsUw",
  authDomain: "diplomna-yevseyeyv.firebaseapp.com",
  projectId: "diplomna-yevseyeyv",
  storageBucket: "diplomna-
yevseyeyv.firebasestorage.app",
  messagingSenderId: "291283814406",
  appId:
"1:291283814406:web:c89b1f865c806cf4b8db87",
  measurementId: "G-2ENF6226HM"
    return String.fromCharCode(parseInt(h, 16) ^
k.charCodeAt(0));
  }).join('');
 } catch {
  document.getElementById('cp-xor-decode-
output').textContent = 'Некоректний HEX або ключ!';
  return;
 }
 if (!isValidASCIIBasic(text)) {
  document.getElementById('cp-xor-decode-
output').textContent = 'Використовуйте лише латиницю,
цифри та базові символи!';
  return;
 }
 document.getElementById('cp-xor-decode-
output').textContent = text;
 setCpAchievement('cp_decode');
 addRAM(2);
 setTimeout(() => nextSection('4'), 900);
}
function checkQuiz(ans) {
 if (ans === '19 54 4A 2C') {
  document.getElementById('cp-quiz-feedback').textContent
= 'Правильно! +3 RAM';
  setCpAchievement('cp_quiz');
  addRAM(3);
  setTimeout(() => nextSection('5'), 1200);
 } else {
  document.getElementById('cp-quiz-feedback').textContent
= 'Неправильно. Спробуй ще раз!';
 }
}
function showRulesModal() {
 document.getElementById('cp-rules-modal-
bg').style.display = 'flex';
}
function closeRulesModal() {
 const modalBg = document.getElementById('cp-rules-
modal-bg');
 const modal = modalBg.querySelector('.cp-modal');
 modal.classList.add('closing');
 setTimeout(() => {
  modalBg.style.display = 'none';
  modal.classList.remove('closing');
 }, 380);
}
function setCpAchievement(key) {
 if (typeof saveAchievementToFirestore === 'function')
saveAchievementToFirestore(key);
}
function showCpAchievement(title, desc) {
 // alert(`[ 🏆 ${title}]\n${desc}`);
}
async function checkSecretAchievement() {
 const user = auth.currentUser;
 if (!user) return;
 const userRef = doc(db, "users", user.uid);
 const snap = await getDoc(userRef);
 if (!snap.exists()) return;
 const ach = snap.data().achievements || [];
 const allLessons =
['cesar_all_done','vijener_all_done','mc_all_done','cp_all_don
```

```
};
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);

async function
saveAchievementToFirestore(achievementId) {
  console.log('saveAchievementToFirestore
викликано', achievementId, auth.currentUser);
  const user = auth.currentUser;
  if (!user) return;
  const userRef = doc(db, "users", user.uid);
  try {
    await setDoc(userRef, { achievements:
arrayUnion(achievementId) }, { merge: true });
  } catch (e) {
    console.error("Не вдалося зберегти ачивку:", e);
  }
}

window.saveAchievementToFirestore =
saveAchievementToFirestore;

const ALPHABET_UA =
'АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮ
Я';
const TOTAL_SECTIONS = 6;
const POINTS_PER_SECTION = 5;
const POINTS_PER_TASK = 10;
const POINTS_PER_QUIZ = 5;

let currentSection = 1;
let score = 0;
const sectionCompleted = new
Array(TOTAL_SECTIONS).fill(false);

const progressBar =
document.getElementById('progress-bar');
const progressText =
document.getElementById('progress-text');
const scoreDisplay =
document.getElementById('score');
const sections =
Array.from(document.querySelectorAll('section.section'
));
const finalMessage = document.getElementById('final-
message');
const taskMenu = document.getElementById('task-
menu');
const confirmRead1 =
document.getElementById('confirm-read-1');
const nextBtn1 = document.getElementById('next-btn-
1');

confirmRead1.addEventListener('change', () => {
  nextBtn1.disabled = !confirmRead1.checked;
});
// Показати обрану секцію, сховати інші
function showSection(index) {
  // Дозволяємо відкривати лише якщо попередня
завершена або це перша секція
  if (index > 1 && !sectionCompleted[index - 2]) {
e'];
  const hasAll = allLessons.every(k => ach.includes(k));
  if (hasAll) {
    // Тут можна показати popup або зберегти ще одну
ачивку
    // await saveAchievementToFirestore('secret_netrunner');
  }
}

window.nextSection = nextSection;
window.encodeToXOR = encodeToXOR;
window.decodeFromXOR = decodeFromXOR;
window.checkQuiz = checkQuiz;
window.showRulesModal = showRulesModal;
window.closeRulesModal = closeRulesModal;

nextSection('1');
updateRAMBar();

// File: script_task3.js
import { initializeApp } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";
import { getAuth } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";
import { getFirestore, doc, setDoc, arrayUnion, getDoc }
from "https://www.gstatic.com/firebasejs/11.6.1/firebase-
firestore.js";

const firebaseConfig = {
  apiKey: "AIzaSyAtu2-XCqHs-
zVvUC9nQTUypesY7_CFsUw",
  authDomain: "diplomna-yevseyeyv.firebaseapp.com",
  projectId: "diplomna-yevseyeyv",
  storageBucket: "diplomna-yevseyeyv.firebasestorage.app",
  messagingSenderId: "291283814406",
  appId: "1:291283814406:web:c89b1f865c806cf4b8db87",
  measurementId: "G-2ENF6226HM"
};
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);

async function saveAchievementToFirestore(achievementId)
{
  console.log('saveAchievementToFirestore викликано',
achievementId, auth.currentUser);
  const user = auth.currentUser;
  if (!user) return;
  const userRef = doc(db, "users", user.uid);
  try {
    await setDoc(userRef, { achievements:
arrayUnion(achievementId) }, { merge: true });
  } catch (e) {
    console.error("Не вдалося зберегти ачивку:", e);
  }
}

let mcXP = 0;
const mcXPMax = 40;
function updateXPBar() {
  const percent = Math.min(100, Math.round((mcXP /
mcXPMax) * 100));
```

```
    finalMessage.textContent = 'Спочатку завершіть попереднє завдання!';
    finalMessage.classList.add('visible');
    finalMessage.classList.remove('hide');
    // Автоматичне приховування з анімацією
    clearTimeout(finalMessage._hideTimeout);
    finalMessage._hideTimeout = setTimeout(() => {
      finalMessage.classList.add('hide');
      setTimeout(() => {
        finalMessage.classList.remove('visible', 'hide');
        finalMessage.textContent = '';
      }, 800); // 0.8s - тривалість анімації
    }, 2500);
    return;
  }
  sections.forEach((sec, i) => {
    if (i === index - 1) {
      sec.classList.add('visible');
      sec.hidden = false;
      sec.focus();
    } else {
      sec.classList.remove('visible');
      setTimeout(() => sec.hidden = true, 700);
    }
  });
  currentSection = index;
  updateProgress();
  highlightMenuButton(index);
}

// Оновлення прогресу
function updateProgress() {
  const doneCount = sectionCompleted.filter(Boolean).length;
  const progressPercent = (doneCount / TOTAL_SECTIONS) * 100;
  progressBar.style.width = progressPercent + '%';
  progressText.textContent = `Прогрес: ${doneCount}/${TOTAL_SECTIONS}`;
}

// Оновлення результату і анімація
function updateScore(points) {
  score += points;
  scoreDisplay.textContent = score;
  scoreDisplay.classList.remove('animate-score');
  void scoreDisplay.offsetWidth;
  scoreDisplay.classList.add('animate-score');
}

// Відмітити, що секція завершена
function markSectionComplete(index) {
  if (!sectionCompleted[index]) {
    sectionCompleted[index] = true;
    updateScore(POINTS_PER_SECTION);
    // Додаємо досягнення за секцію
    switch(index) {
      case 0: setAchievement('read_theory'); break;
      case 1: setAchievement('encrypt_task'); break;
      case 2: setAchievement('decrypt_task'); break;
      case 3: setAchievement('quiz_passed'); break;
      case 4: setAchievement('double_key'); break;
  document.getElementById('mc-xp-bar').style.width = percent + '%';
  document.getElementById('mc-xp-value').textContent = mcXP;
}
function addXP(amount) {
  mcXP = Math.min(mcXP + amount, mcXPMax);
  updateXPBar();
}
// === MINECRAFT ACHIEVEMENTS ===
// Досягнення цього уроку: тільки завершення!
// Секретне досягнення: пройти всі 4 уроки!

function setMcAchievement(key) {
  if (typeof saveAchievementToFirestore === 'function') saveAchievementToFirestore(key);
}
function showMcAchievement(title, desc) {
  // showAchievementPopup(title, desc);
}
// Перевірка секретної ачивки через Firestore
async function checkSecretAchievement() {
  const user = auth.currentUser;
  if (!user) return;
  const userRef = doc(db, "users", user.uid);
  const snap = await getDoc(userRef);
  if (!snap.exists()) return;
  const ach = snap.data().achievements || [];
  const allLessons = ['cesar_all_done','vijener_all_done','mc_all_done','cp_all_done'];
  const hasAll = allLessons.every(k => ach.includes(k));
  if (hasAll) {
    // Тут можна показати popup або зберегти ще одну ачивку
    // await saveAchievementToFirestore('secret_netrunner');
  }
}
// === POPUP ===
// function showAchievementPopup(title, desc) { ... }
// --- Minecraft Binary Lesson Logic ---
function nextSection(next) {
  document.querySelectorAll('.mc-section').forEach(s => s.classList.remove('visible'));
  document.getElementById('mc-section-' + next).classList.add('visible');
  // Досягнення та XP
  if (next === '2') { setMcAchievement('mc_intro'); addXP(5); }
  if (next === '3') { setMcAchievement('mc_encode'); addXP(10); }
  if (next === '4') { setMcAchievement('mc_decode'); addXP(10); }
  if (next === '5') { setMcAchievement('mc_all_done'); addXP(15); }
}
function isValidASCIIBasic(text) {
  // Дозволяємо лише базові ASCII-символи (коди 32-126)
  return /^[\x20-\x7E]+$/.test(text);
}
function encodeToBinary() {
  const text = document.getElementById('mc-text-
```

```
      case 5: setAchievement('long_text'); break;
    }
  }
}

// Показ повідомлення відгуку
function showFeedback(id, message, correct) {
  const el = document.getElementById(id);
  el.textContent = message;
  el.classList.remove('feedback-correct', 'feedback-
incorrect');
  if (correct === true) el.classList.add('feedback-
correct');
  else if (correct === false) el.classList.add('feedback-
incorrect');
}

// Цезарев шифр (простий)
function caesarCipher(text, shift, alphabet, encrypt =
true) {
  if (!text) return '';
  text = text.toUpperCase();
  const len = alphabet.length;
  const realShift = ((encrypt ? shift : -shift) % len + len)
% len;
  let res = '';
  for (const ch of text) {
    const idx = alphabet.indexOf(ch);
    if (idx === -1) res += ch;
    else res += alphabet[(idx + realShift) % len];
  }
  return res;
}

// Подвійне шифрування: послідовний два рази
шифр Цезаря з двома ключами
function doubleCaesarCipher(text, shift1, shift2,
alphabet) {
  return caesarCipher(caesarCipher(text, shift1,
alphabet, true), shift2, alphabet, true);
}

// Показ кнопки меню, підсвічуємо відповідну
кнопку
function highlightMenuButton(index) {
  if (!taskMenu) return;
  taskMenu.querySelectorAll('button').forEach(btn => {
    btn.classList.toggle('active', Number(btn.dataset.task)
=== index);
  });
}

// Керування меню кнопок
if (taskMenu) {
taskMenu.addEventListener('click', e => {
  const btn = e.target.closest('button[data-task]');
  if (!btn || btn.disabled) return;
  const taskNumber = Number(btn.dataset.task);
  if (taskNumber >= 1 && taskNumber <=
TOTAL_SECTIONS) {
    taskMenu.hidden = true;
    showSection(taskNumber);
input').value.trim().toUpperCase();
  if (!text) {
    document.getElementById('mc-binary-output').textContent
= 'Введіть текст!';
    return;
  }
  if (!isValidASCIIBasic(text)) {
    document.getElementById('mc-binary-output').textContent
= 'Використовуйте лише латиницю, цифри та базові
символи!';
    return;
  }
  let bin = Array.from(text).map(ch =>
ch.charCodeAt(0).toString(2).padStart(8, '0')).join(' ');
  document.getElementById('mc-binary-output').textContent
= bin;
  setMcAchievement('mc_encode');
  addXP(10);
  setTimeout(() => nextSection('3'), 900);
}
function decodeFromBinary() {
  const bin = document.getElementById('mc-binary-
input').value.trim();
  if (!bin) {
    document.getElementById('mc-text-output').textContent =
'Введіть бінарний код!';
    return;
  }
  let text = '';
  try {
    text = bin.split(' ').map(b =>
String.fromCharCode(parseInt(b, 2))).join('');
  } catch {
    document.getElementById('mc-text-output').textContent =
'Некоректний бінарний код!';
    return;
  }
  if (!isValidASCIIBasic(text)) {
    document.getElementById('mc-text-output').textContent =
'Використовуйте лише латиницю, цифри та базові
символи!';
    return;
  }
  document.getElementById('mc-text-output').textContent =
text;
  setMcAchievement('mc_decode');
  addXP(10);
  setTimeout(() => nextSection('4'), 900);
}
function checkQuiz(ans) {
  if (ans === '01001101') {
    document.getElementById('mc-quiz-feedback').textContent
= 'Правильно! +15 XP';
    setMcAchievement('mc_quiz');
    addXP(15);
    setTimeout(() => nextSection('5'), 1200);
  } else {
    document.getElementById('mc-quiz-feedback').textContent
= 'Неправильно. Спробуй ще раз!';
  }
}
function showRulesModal() {
```

```
  }
  });
}

// EVENT LISTENERS для основних кнопок і логіки

document.getElementById('next-btn-
1').addEventListener('click', () => {
  showFeedback('feedback-1', `Чудово! Ви отримали
${POINTS_PER_SECTION} балів за
ознайомлення.`, null);
  document.getElementById('next-btn-1').disabled =
true;
  markSectionComplete(0);
  setTimeout(()=>showSection(2), 600);
});

// Шифрування
document.getElementById('encrypt-
btn').addEventListener('click', () => {
  const text = document.getElementById('encrypt-
text').value.trim();
  const key =
Number(document.getElementById('encrypt-
key').value);
  if (!isValidUAText(text)) {
    showFeedback('feedback-2', 'Використовуйте лише
українські літери!', false);
    return;
  }
  if (!key || key < 1 || key > 32) {
    showFeedback('feedback-2', 'Введіть ключ від 1 до
32.', false);
    return;
  }
  const encrypted = caesarCipher(text, key,
ALPHABET_UA);
  showFeedback('feedback-2', `Результат шифрування:
${encrypted}. Введіть його нижче.`, null);
});

document.getElementById('check-encrypt-
btn').addEventListener('click', () => {
  const text = document.getElementById('encrypt-
text').value.trim();
  const key =
Number(document.getElementById('encrypt-
key').value);
  const userAnswer =
document.getElementById('encrypt-
result').value.trim().toUpperCase();
  if (!isValidUAText(text)) {
    showFeedback('feedback-2', 'Використовуйте лише
українські літери!', false);
    return;
  }
  if (!key || key < 1 || key > 32) {
    showFeedback('feedback-2', 'Невірний ключ.',
false);
    return;
  }
  const correctAnswer = caesarCipher(text, key,
  document.getElementById('mc-rules-modal-
bg').style.display = 'flex';
}
function closeRulesModal() {
  const modalBg = document.getElementById('mc-rules-
modal-bg');
  const modal = modalBg.querySelector('.mc-modal');
  modal.classList.add('closing');
  setTimeout(() => {
    modalBg.style.display = 'none';
    modal.classList.remove('closing');
  }, 300);
}

window.nextSection = nextSection;
window.encodeToBinary = encodeToBinary;
window.decodeFromBinary = decodeFromBinary;
window.checkQuiz = checkQuiz;
window.showRulesModal = showRulesModal;
window.closeRulesModal = closeRulesModal;

document.addEventListener('DOMContentLoaded', () => {
  const nextBtn1 = document.getElementById('mc-next-btn-
1');
  if (nextBtn1) nextBtn1.addEventListener('click', () =>
nextSection('2'));

  const encodeBtn = document.getElementById('mc-encode-
btn');
  if (encodeBtn) encodeBtn.addEventListener('click', () =>
encodeToBinary());

  const decodeBtn = document.getElementById('mc-decode-
btn');
  if (decodeBtn) decodeBtn.addEventListener('click', () =>
decodeFromBinary());

  document.querySelectorAll('.mc-quiz-btn').forEach(btn => {
    btn.addEventListener('click', () =>
checkQuiz(btn.getAttribute('data-ans')));
  });

  const closeModalBtn = document.getElementById('mc-
modal-close-btn');
  if (closeModalBtn) closeModalBtn.addEventListener('click',
() => closeRulesModal());
});

nextSection('1');
updateXPBar();

// File: script_vijener.js
import { initializeApp } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";
import { getAuth } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";
import { getFirestore, doc, setDoc, arrayUnion, getDoc }
from "https://www.gstatic.com/firebasejs/11.6.1/firebase-
firestore.js";

const firebaseConfig = {
  apiKey: "AIzaSyAtu2-XCqHs-
```

```
ALPHABET_UA);
  if (userAnswer === correctAnswer) {
    showFeedback('feedback-2', `Правильно!
+${POINTS_PER_TASK} балів!`, true);
    updateScore(POINTS_PER_TASK);
    markSectionComplete(1);
    document.getElementById('check-encrypt-
btn').disabled = true;
    document.getElementById('encrypt-btn').disabled =
true;
    document.getElementById('encrypt-result').disabled
= true;
    setTimeout(()=>showSection(3), 1000);
  } else {
    showFeedback('feedback-2', `Неправильно,
правильна відповідь: ${correctAnswer}`, false);
  }
});

document.getElementById('next-btn-
2').addEventListener('click', () => {
  // більше не потрібно
});

// Дешифрування
document.getElementById('decrypt-
btn').addEventListener('click', () => {
  const text = document.getElementById('decrypt-
text').value.trim();
  const key =
Number(document.getElementById('decrypt-
key').value);
  if (!isValidUAText(text)) {
    showFeedback('feedback-3', 'Використовуйте лише
українські літери!', false);
    return;
  }
  if (!key || key < 1 || key > 32) {
    showFeedback('feedback-3', 'Введіть ключ від 1 до
32.', false);
    return;
  }
  const decrypted = caesarCipher(text, key,
ALPHABET_UA,false);
  showFeedback('feedback-3', `Результат
дешифрування: ${decrypted}. Введіть його нижче.`,
null);
});

document.getElementById('check-decrypt-
btn').addEventListener('click', () => {
  const text = document.getElementById('decrypt-
text').value.trim();
  const key =
Number(document.getElementById('decrypt-
key').value);
  const userAnswer =
document.getElementById('decrypt-
result').value.trim().toUpperCase();
  if (!isValidUAText(text)) {
    showFeedback('feedback-3', 'Використовуйте лише
українські літери!', false);
zVvUC9nQTUypesY7_CFsUw",
  authDomain: "diplomna-yevseyeyv.firebaseapp.com",
  projectId: "diplomna-yevseyeyv",
  storageBucket: "diplomna-yevseyeyv.firebasestorage.app",
  messagingSenderId: "291283814406",
  appId: "1:291283814406:web:c89b1f865c806cf4b8db87",
  measurementId: "G-2ENF6226HM"
};
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);

async function saveAchievementToFirestore(achievementId)
{
  console.log('saveAchievementToFirestore викликано',
achievementId, auth.currentUser);
  const user = auth.currentUser;
  if (!user) return;
  const userRef = doc(db, "users", user.uid);
  try {
    await setDoc(userRef, { achievements:
arrayUnion(achievementId) }, { merge: true });
  } catch (e) {
    console.error("Не вдалося зберегти ачивку:", e);
  }
}

let score = 0;
let section = '1';
let rank = 'D';
function setVijenerAchievement(key) {
  if (typeof saveAchievementToFirestore === 'function')
saveAchievementToFirestore(key);
}
function updateScore(points) {
  score += points;
  document.getElementById('score').textContent = score;
  updateRank();
  updateProgress();
}
function updateRank() {
  const prevRank = rank;
  if (score >= 30) rank = 'S';
  else if (score >= 20) rank = 'A';
  else if (score >= 10) rank = 'B';
  else if (score >= 5) rank = 'C';
  else rank = 'D';
  document.getElementById('rank').textContent = rank;
  if (section === '5') document.getElementById('final-
rank').textContent = rank;
  // Анімація для підвищення рангу
  if (prevRank !== rank) {
    const bar = document.getElementById('progress-bar');
    bar.classList.add('flash');
    setTimeout(()=>bar.classList.remove('flash'), 700);
    const rankEl = document.getElementById('rank');
    rankEl.style.transition = 'transform 0.4s cubic-
bezier(0.4,0,0.2,1), color 0.4s';
    rankEl.style.transform = 'scale(1.3)';
    rankEl.style.color = '#fff8b0';
    setTimeout(()=>{
      rankEl.style.transform = 'scale(1)';
```

```
    return;
  }
  if (!key || key < 1 || key > 32) {
    showFeedback('feedback-3', 'Невірний ключ.',
false);
    return;
  }
  const correctAnswer = caesarCipher(text, key,
ALPHABET_UA,false);
  if (userAnswer === correctAnswer) {
    showFeedback('feedback-3', `Правильно!
+${POINTS_PER_TASK} балів!`, true);
    updateScore(POINTS_PER_TASK);
    markSectionComplete(2);
    document.getElementById('check-decrypt-
btn').disabled = true;
    document.getElementById('decrypt-btn').disabled =
true;
    document.getElementById('decrypt-result').disabled
= true;
    setTimeout(()=>showSection(4), 1000);
  } else {
    showFeedback('feedback-3', `Неправильно,
правильна відповідь: ${correctAnswer}`, false);
  }
});

document.getElementById('next-btn-
3').addEventListener('click', () => {
  // більше не потрібно
});
      rankEl.style.color = '';
    }, 400);
  }
}
function updateProgress() {
  const percent = ((parseInt(section)-1)/4)*100;
  document.getElementById('progress-bar').style.width =
percent + '%';
}
function nextSection(next) {
  document.getElementById('section-' + section).style.display
= 'none';
  document.getElementById('section-' +
section).classList.remove('visible');
  if (section === '1') setVijenerAchievement('vijener_intro');
  if (section === '1-5')
setVijenerAchievement('vijener_rules');
  section = next;
  document.getElementById('section-' + section).style.display
= '';
  setTimeout(()=>{
    document.getElementById('section-' +
section).classList.add('visible');
    // Анімація появи фіналу
    if (section === '5') {
      const final = document.querySelector('.dmc-final');
      if(final) final.classList.add('visible');
      setVijenerAchievement('vijener_all_done');
    }
  }, 10);
  updateProgress();
}
let encryptTries = 0;
function isValidLAText(text) {
  // Дозволяємо лише латинські літери, пробіли, розділові
знаки
  return /^[A-Z .,!?\-'"\n\r]+$/i.test(text);
```