МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ТОКАР СТАНІСЛАВ СЕРГІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри інформаційних технологій,
д-р. техн. наук, професор
_____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

РОЗРОБКА ВЕБЗАСТОСУНКУ У ЖАНРІ СЮЖЕТНОЇ КАРТКОВОЇ ГРИ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Науковий керівник:
Олексій МОСКВІН, доцент кафедри
інформаційних технологій,
канд. тех. наук

_____
(підпис)

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2026

АНОТАЦІЯ

Токар С.С. Розробка вебзастосунку у жанрі сюжетної карткової гри. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2026.

У кваліфікаційній (бакалаврській) роботі досліджено та проаналізовано поняття вебгри, виділені основні їх принципи та створення. За допомогою таких технологій як: Adobe Ilustrator, HTML, SCSS, TypeScript, бібліотеки Vue.ts, був розроблений дизайн та вебгра, головний сюжет якої розгортається навколо епічного твору античної культури.

Ключові слова: вебгра, «Іліада», вебдизайн, TypeScript, Vue.ts.

78 ст., 63 рис., 3 табл., 0 дод., 43 джерел.

ABSTRACT

Tokar S.S. Web Application Development in the Genre of a Story-Based Card Game. Specialisation 122 «Computer Science», educational programme «Computer Science». Vasyl Stus Donetsk National University Vasyl Stus, Vinnytsia 2026.

This undergraduate thesis examines and analyses the concept of web games, identifying their main principles and development. Using technologies such as Adobe Illustrator, HTML, SCSS, TypeScript and the Vue.js library, a design and a web game were developed, the main plot of which revolves around an epic work of ancient culture.

Keywords: web game, «The Iliada», web design, TypeScript, Vue.js.

78 pages, 63 figures, 3 tables, 0 appendices, 43 references.

ЗМІСТ

ВСТУП

Сучасна індустрія комп'ютерних ігор є однією із найрозвиненіших та популярніших сфер цифрових технологій, що поєднує в собі різні галузі: від комп'ютерних технологій, чому буде присвячено чимало часу у нашій роботі, до кінематографічного сторітелінгу. Постійний попит на такий вид продукції вимагає постійного вдосконалення для задоволення потреб користувачів та домінування на ринку, тому резонно, що ця сфера є однією із найперспективніших та найунікальніших у своєму стилі.

Актуальність цифрових ігор почала зростати в міру зростання попиту та технологічного розвитку галузі. Разом із нею поступово змістився акцент із розважального контенту до культурно-освітнього. У статті від Fromtiersin.org із назвою «Editorial: Video games for impact: design projects that can change the way we think» ми знайомимося із терміном serious games (*укр. «серйозні ігри»*), що, особливо, з 2000-х років набув стійкого значення і відтоді його застосовують до різножанрових комп'ютерних ігор, основна мета яких не є розвага [1].

З часом відеоігри стали важливими культурними формами, що мають потенціал для залучення різноманітної аудиторії в різних соціальних, політичних та культурних контекстах. Окрім розваг, вони дедалі більше демонструють потенціал для розвитку емпатії, подолання стереотипів, підвищення обізнаності про соціальні та екологічні проблеми та розширення доступу до змістовної гри. Ці ігри класифікуються як серйозні ігри та форми серйозної гри, що підкреслює, що їхня мета виходить за рамки простого задоволення [1].

Метою бакалаврської роботи є розробка інтерактивної наративної комп'ютерної гри за мотивами «Іліади» Гомера у жанрі вибору та наслідків, яка поєднує розважальну, освітню та культурно-популяризаційну функції, а також дослідження особливостей адаптації класичного літературного твору до сучасного цифрового ігрового середовища.

Нашим завдання буде проаналізувати існуючі ігри у подібному жанрі та стилі, дослідити та підкреслити основні ключові моменти та переваги, порівняти їх із нашими та зробити висновки щодо кращого напрямку розробки. Також ми

проаналізуємо інструменти для розробки ігор у вебсередовищі: які можливості та обмеження цього підходу, порівняємо його із іншими існуючими альтернативами та окреслимо наші основні переваги та недоліки. Чи не найважливішим завданням всієї цієї роботи після теоретичного аналізу та створення концепції буде сама реалізація інді-гри, де ми покроково будемо слідувати згідно окресленого заздалегідь плану, починаючи від зручного інтерфейсу користувача до перших алгоритмів роботи нелінійного сюжету.

Об’єктом дослідження у даній бакалаврській роботі стане процес розробки комп’ютерних наративних ігор, орієнтованих на інтерактивну взаємодію користувача із сюжетом через систему вибору та наслідків. У межах дослідження розглядаються вебпідходи до створення цифрових ігрових продуктів. Особлива увага приділяється використанню відеоігор як засобу адаптації літературної спадщини до сучасного цифрового формату, що дозволяє розглядати гру не лише як розважальний продукт, а і як інструмент популяризації класичної культури, як було зазначено раніше.

Предметом дослідження є методи, принципи та особливості проєктування і програмної реалізації інтерактивної наративної гри у стилі Reigns/Lapse за мотивами «Іліади» Гомера. Зокрема, досліджуються механізми адаптації античного літературного твору до формату сучасної комп’ютерної гри, побудова сюжетних гілок, система вибору та наслідків, баланс ігрових ресурсів, структура користувацького інтерфейсу, а також технічні інструменти, що забезпечують створення функціонального цифрового продукту. Предмет охоплює як творчі аспекти геймдизайну, так і практичні питання програмної реалізації гри.

Результати цієї роботи можуть бути використанні для теоретичного аналізу втілення та адаптації історичних подій у комп’ютерні ігри, освітню роль для різних цільових аудиторій, а також практичне застосування у вигляді реалізації подібних проєктів.

Щодо апробації результатів дослідження та участі у різних конференціях, то автор не брав участі у написанні статей та не брав участь у конференціях.

Структура бакалаврської роботи складається із трьох частин. В першому розділі розглядаються існуючі сюжетні інді-ігри, вебтехнології для їх створення, нелінійний сюжет та особливості «Іліади» як основи для адаптації. В другому розділі ми обираємо відповідні інструменти та обґрунтовуємо доцільність їх використання, розглядаємо концепцію гри, структуру та вигляд сюжету та створюємо модель програми та інтерфейс користувача. У третьому розділі відбувається розробка вебгри із впровадженням переглянутих вище принципів та висновків, а також тестування і аналіз отриманих результатів. Така послідовність є логічною з точки зору розв’язання задач, які ми поставили перед собою у цій роботі.

У роботі було використано 63 ілюстрацій для пояснення та демонстрації результату чи процесів, використано 3 таблиці, що пояснюють роботу програмних алгоритмів, а також у всіх розділах використано 43 джерела, що посилаються на певний параграф чи абзац для пояснення певної концепції, вибору чи ідеї.

РОЗДІЛ 1
ОГЛЯД ТА ПОРІВНЯННЯ ІСНУЮЧИХ СЮЖЕТНИХ ІНДІ-ІГОР

1.1 Нелінійний сюжет у цифрових іграх

Одна важлива річ, що виділяє чергову комп'ютерну гру серед різноманіття інших, надає їй сенсу та цікавості, є сюжет. Це саме та частина, що надає грі глибину, створює зв'язок через емпатію, таким чином залучаючи більшу кількість гравців. Саме тому вдалий сюжет несе чи не вирішальне значення для цифрової гри, навіть попри її досконалу програмну реалізацію та дизайн. Однак спочатку ми розглянемо історію розвитку цифрових ігор і яку роль тут відграє нелінійний сюжет [2].

1.1.1 Історія цифрових ігор

Традиційні ігри, що включають в себе рольові або настільні, беруть свій початок ще від перших спільнот людей, вони тісно вплетені в культуру кожної нації, носячи свій відтінок унікальності, мандруючи крізь століття та тисячоліття. Вони існують досі, здебільшого та переважно для населення юного віку, проте у певний період їх почав поволі витісняти інший тип розваг.

Сама ідея цифрових ігор виникла ще у 1947 році, в епоху розвитку інформаційних та цифрових технологій, тобто створення перших комп'ютерів. Саме в цей момент Томас Т. Голдсміт та Естл Рей Манн подали патент 2 455 992, суть якого полягала у ідеї інтерактивної гри та електронно-променевій трубці [3].

Хоч ця ідея була одна із перших, про неї було забуто. Прийнято вважати, що першою повноцінною грою за сучасними мірками, є «Spacewar!» 1962-го року, яка була розроблена Стівом Расселом, Мартіном Грецем та Вейном Вітанемом у Массачусетському технологічному інституті (MIT).

Рисунок 1.1 — Ден Едвардс (ліворуч) і Пітер Семсон грають у «Spacewar!» на дисплеї PDP-1 Type 30

Ігри тієї епохи не могли перерости фізичні обмеження комп'ютерів — це були надзвичайно громіздкі пристрої, які не міг дозволити собі хто-небудь, крім лабораторій та університетів. Тому логічно припустити, що доступ до процесу створення ігор та їх результату мали лише викладачі та студенти.

Розробка ігор не припинилася і у 1970-х були створенні перші онлайн-ігри, які дозволили об'єднати гравців разом, те що зараз є нормою [4]. Одними із перших таких ігор були «Maze War» та «Spacim» 1974 року.

Також у 1970-х відеоігри вийшли за межі та стали комерційним продуктом, завдяки появі перших аркадних автоматів і домашніх консолей, як от як Pong та Magnavox Odyssey. Саме цей період започаткував масову популярність відеоігор.

Рисунок 1.2 — Перша у світі комерційна домашня ігрова консоль Magnavox Odyssey

Однак перші відеоігри мали надзвичайно прозаїчний сюжет, він був переважно орієнтований на простий геймплей для аркад, змагань, реакцію та набір очок і ще не мав тієї глибини та залученості, хоч цей тип розваг ставав неймовірно популярним через свою новизну та цікавість.

Ігровий ринок стрімко зростав, з'явилися домашні консолі, персональні комп'ютери та нові жанри. Проте перенасичення ринку призвело до великої кризи 1983 року. Відновлення індустрії відбулося завдяки успіху Nintendo Entertainment System, що започаткувала новий етап розвитку.

Рисунок 1.3 — Гральна консоль Nintendo Entertainment System

Технологічний прорив стався у 1990-х, разом із вдосконаленням комп'ютерів, що ознаменувався переходом від 2D до 3D-графіки, розвиток CD-ROM, Інтернету. Відтоді багатокористувацькі ігри значно розширили можливості гейм-дизайну. У цей час з'являються складніші сюжетні, рольові та нелінійні ігри.

А починаючи із початку 2000-х років аж до сьогодення ігрова індустрія стала глобальною та багатоплатформною та розвивається в міру розвитку інформаційно-цифрових технологій. Зараз ми маємо:

- мобільний геймінг;
- онлайн-ігри;
- VR;
- інді-розробка;

- інтерактивний сторітелінг.

Також варто зазначити, що сьогодні відеоігри виконують не лише розважальну, а й освітню, культурну та соціальну функції [5].

1.1.2 Нелінійний сюжет

Що ж до сюжету, що є основним двигуном будь-якої гри, ми зосередимо свою увагу на нелінійному сюжеті. Техніки створення сюжету не рідко кочують із найпопулярнішої сфери їх використання: кіновиробництва. Зазвичай найпростіший у своїй структурі тип сюжету — лінійний. Якщо ми говоримо про ігри, то це означає, що гравець рухається відповідно до певного чіткого плану та прогресу, щоб завершити гру.

Нелінійний сюжет має широкий аспект визначення саме у цифрових іграх. У кінофільмах нелінійний сюжет, що зветься нелінійним наративом, є особливим інструментом, що не підвладний кожному сценаристу. Його майстерне використання змушує певних людей почуватися дурнями, а інших — розумними. Основна ідея такого способу подачі полягає у представленні історії у не хронологічному порядку, а мета лежить в бажанні створення таємничості, глибини та введення глядачів в оману для «Вау-ефекту». Найбільшими майстрами такого жанру є Крістофер Нолан та Квентін Тарантіно [6].

Що ж до нелінійного сюжету в іграх, він може мати представлення у такому самому вигляді, що сценарій у типових фільмах. Це змушує аудиторію не лише напружено шукати рішення та заглиблюватись у сюжет, а робити сам продукт унікальним та відомим.

Ще одним визначення нелінійної гри буде термін гри «відкритого світу». Замість лінійного проходження певних етапів, гравець тепер має змогу самому вирішувати яким чином йому діяти, тобто гра чітко пропонує своїм гравцям досліджувати власний світ.

Також існують ігри із змішаними підходами, тобто у певний момент історія має лінійний характер, але згодом плавно перетікає у нелінійну оповідь або

навпаки. Це дозволяє урізноманітнити сюжет, зробити його більш унікальним та кастомним для аудиторії [7].

1.2 Особливості епосу «Іліада» як основи для адаптації

За основу створення цифрової інді-гри було взято героїчний епос «Іліада» давньогрецького поета Гомера. На вибір цього жанру та твору вплинула ціла низка важливих факторів [8].

По-перше, «Іліада» є важливим літературним та культурним твором, о має важливе значення в світовій літературі. Прикладом цьому є постійне перевидання, нові переклади та нові продажі. Одні із важливих критеріїв такої популярності є смислова, хоч і поверхнева знайомість більшості людей із цим твором, а самі образи епосу є впізнаваними та не застарілими з точки зору актуальності. У «Іліаді» розглядаються та порушуються важливі питання: питання долі, сенсу життя, честь, влада, війна, вибір [9].

По-друге, з точки зору сюжету — це насичена подіями історія, що має багато персонажів із різними сюжетними арками та мотиваціями. Також там включено різні політичні конфлікти, а також можливість розвинення альтернативних сценаріїв. Чимало героїв резонують із нашими поглядами, вони запитують нас який би шлях ми обрали у цьому житті: шлях довгого нудного життя чи коротке, але сповнене величі та слави. Це дає нам важливий наративний матеріал, що буде залучати аудиторію до вибору та взаємодії, що дуже важливо у таким іграх [9].

По-третє, сюжет ідеально підходить під реалізацію інді-гри для жанру, який ми обрали у цій бакалаврській роботі і його детальніше буде розглянуто у концепції гри. Наразі немає подібних ігор у подібному жанрі та стилі реалізації із картковим вибором та нелінійним сюжетом, хоча існують схожі за виглядом чи концепцією прототипи, проте гра у цій роботі є унікальною.

Основні виклики полягають у написанні самого сюжету. «Іліада» є великим твором, тому для повної реалізації сюжету у правильному формату, а також додавання альтернативних гілок розвитку вимагає чималих ресурсів та часу.

Тому у цій бакалаврській роботі сюжет у розробці буде свідомо обмежений по довжині через очевидні причини.

Наразі можна з впевненістю сказати, що сукупність вищевказаних причин надають нам міцну сюжетну основу, вона підходить для реалізації нашої інді-гри, хоч як і у кожному проєкті у нас є свої виклики та проблеми.

1.3 Аналіз механіки карткових виборів у інших іграх

Механіка карткових виборів (card-based decision system) є однією з популярних форм реалізації інтерактивного сюжету у сучасних відеоіграх. Її суть полягає в тому, що події подаються у вигляді карток або окремих ситуацій, де гравцю пропонується кілька варіантів дій. Обрані рішення безпосередньо впливають на розвиток сюжету, зміну ресурсів, взаємодію з персонажами та фінальний результат гри. Такий формат особливо поширений серед інді-ігор завдяки простоті реалізації та можливості створювати нелінійний сюжет навіть за обмежених ресурсів і взяв цю ідею із додатку для знайомств Tinder [10].

Одним із найвідоміших прикладів є гра «Reigns», у якій гравець приймає рішення за допомогою свайпу карток ліворуч або праворуч. Кожен вибір впливає на основні показники держави: церкву, армію, народ і економіку. Завдяки цьому навіть проста механіка дозволяє створити глибоку політичну симуляцію з багатьма сюжетними наслідками.

Подібний підхід використано і в «Lapse: A Forgotten Future», де карткова система поєднується з постапокаліптичним сюжетом. У цій грі рішення впливають не лише на ресурси, але й на подальший розвиток історії, що підвищує реіграбельність та створює відчуття відповідальності за майбутнє ігрового світу.

Схожі механіки можна побачити також у «The Cosmic Wheel Sisterhood» та «Cultist Simulator». У цих проєктах картки використовуються не тільки як система вибору, а і як інструмент розвитку сюжету, управління ресурсами або побудови складніших ігрових процесів.

З точки зору гейм-дизайну, карткові механіки належать до систем інтерактивної літератури та нелінійного наративу, де гравець бере активну участь у формуванні історії через власні рішення [11].

Сучасні дослідження також показують, що такі системи добре підходять для мобільних і вебігор, оскільки поєднують простий інтерфейс із великою варіативністю подій, а також зменшують когнітивне навантаження на користувача завдяки обмеженій кількості виборів [12].

Таким чином, аналіз існуючих ігор показує, що механіка карткових виборів є ефективним інструментом реалізації інтерактивного сюжету. Вона дозволяє створювати цікаві нелінійні історії, зберігаючи баланс між простотою розробки та глибиною ігрового процесу, що робить її особливо актуальною для вебгеймдеву та інді-розробки.

1.4 Вебтехнології для створення інтерактивних ігор

Сучасний розвиток цифрових технологій суттєво розширив можливості створення інтерактивних ігор, зокрема у вебсередовищі. Якщо ранні браузерні ігри характеризувалися обмеженим функціоналом, спрощеною графікою та переважно розважальним спрямуванням, то сучасні вебтехнології дозволяють реалізовувати повноцінні ігрові продукти зі складною логікою, нелінійним сюжетом, адаптивним інтерфейсом та широкими можливостями взаємодії користувача.

Використання вебтехнологій у гейм-деві стало особливо актуальним завдяки розвитку HTML5, CSS3, JavaScript та супутніх фреймворків, які забезпечують кросплатформність, доступність і відносно низький поріг входу для розробки. На відміну від традиційних ігрових рушіїв, веброзробка дозволяє створювати ігри, доступні без встановлення додаткового програмного забезпечення, що значно спрощує доступ користувачів до продукту через браузер.

Розглянемо існуючі інструменти.

HTML5 — нова версія мови розмітки гіпертексту HTML. Це основа для перегляду контенту в браузері, мова має спеціальні елементи «теги», якими будується структура та наповнюється контентом вся сторінка. Тобто HTML є будівельним матеріалом для відображення інформації на вебсторінці [13].

Canvas — тег HTML, який діє як контейнер чи полотн для створення растрової графіки, анімацій, ігор та візуальних елементів безпосередньо у браузері за допомогою скриптів [14].

CSS — мова стилів, яка використовується для оформлення зовнішнього вигляду вебсторінок, написаних мовами розмітки (найчастіше HTML). Вона визначає кольори, шрифти, розміри та розташування елементів, дозволяючи відокремити вміст сторінки (HTML) від її дизайну [15].

SCSS — це один із двох синтаксисів популярного препроцесора Sass, що розширює можливості звичайного CSS. Він дозволяє використовувати змінні, вкладені правила, міксини та математичні операції, роблячи написання стилів швидшим та структурованим, тобто робить процес розробки легким, зручнішим та приємнішим. Файли .scss компілюються у стандартний CSS, який розуміють браузери [16].

JavaScript — це високорівнева, динамічна, прототипна об’єктно-орієнтована мова програмування, що інтерпретується. Вона є основною мовою для створення інтерактивних та динамічних вебсторінок (front-end), працюючи безпосередньо в браузері користувача. Завдяки Node.js, JS також використовується для розробки серверних застосунків (back-end), однак у цій роботі ми будемо працювати лише із клієнтською стороною [17].

TypeScript — це мова програмування, розроблена Microsoft, яка є надмножиною (superset) JavaScript, додаючи до неї статичну типізацію та розширені можливості. Вона компілюється в чистий JavaScript, що дозволяє запускати її в будь-якому браузері чи середовищі. Основна мета — полегшити розробку великих застосунків, зменшуючи кількість помилок завдяки виявленню їх на етапі написання коду [18].

Vue / React / Angular — це три найпопулярніші сучасні інструменти (фреймворки та бібліотеки) JavaScript для розробки інтерфейсів користувача (front-end) та односторінкових застосунків (SPA). Вони мають свої унікальні особливості, різні рівні складності, популярності серед розробників та підтримку, але об'єднує їх те, що вони спрощують створення складних, інтерактивних вебсайтів, дозволяючи оновлювати вміст сторінки без її перезавантаження [19].

React — це радше бібліотека, ніж повний фреймворк. Він дуже гнучкий, має величезну екосистему, сильний ринок й підтримку і добре підходить для масштабних застосунків. Але через цю гнучкість розробник часто сам обирає архітектуру, маршрутизацію, менеджмент стану та інші інструменти. Це дає свободу, але може ускладнювати навчання й підтримку.

Angular — повноцінний enterprise-фреймворк із великою кількістю вбудованих рішень: Dependency Injection, RxJS, TypeScript-first підхід, модулі, CLI. Він потужний для великих корпоративних систем, але має високу складність входу, багато шаблонного коду й часто вважається важчим для невеликих або середніх проєктів.

Vue часто вважають «золотою серединою». Він простіший у вивченні, ніж Angular, і структурованіший, ніж React. Vue має зрозумілий синтаксис, зручну реактивність, хорошу документацію та поступове масштабування — від маленьких компонентів до великих SPA. Його шаблони ближчі до звичайного HTML, що особливо зручно для новачків [25].

Phaser — це швидкий, безкоштовний фреймворк з відкритим вихідним кодом, призначений для розробки 2D-ігор на HTML5 для настільних комп'ютерів та мобільних пристроїв. Він використовує JavaScript або TypeScript для створення кросплатформних ігор, які працюють у сучасних браузерах завдяки рендерингу WebGL та Canvas [20].

Firebase — це комплексна хмарна платформа від Google для розробки мобільних та вебдодатків. Вона надає готові інструменти (Backend-as-a-Service, BaaS), що дозволяють розробникам швидко створювати додатки без необхідності

керувати серверами. Основні можливості включають NoSQL бази даних (Firestore, Realtime Database), автентифікацію, хостинг та аналітику. Це класний інструмент для початківців, тестувальників продуктів та тих, хто не хоче розгортати складні сервіси.

LocalStorage — це механізм вебсховища (частина Web Storage API), який дозволяє JavaScript зберігати пари ключ-значення безпосередньо у браузері користувача, що на відміну від локальних чи хмарних баз даних може бути зручним для невеликих проектів або тих, які не потребують класичного збереження інформації із прив'язкою до акаунтів.

JSON (JavaScript Object Notation) — це текстовий формат для зберігання та обміну даними у спосіб, який є зручним для читання людиною та машинним аналізом. Як результат, JSON відносно легко вивчати та усувати неполадки. Хоча JSON бере свій початок у мови програмування JavaScript, він перетворився на дуже потужний формат даних, який спрощує обмін даними на різних платформах та мовах програмування [27].

Цей формат активно використовується веброзробниками для передання даних між сервером і клієнтською частиною вебсайту, наприклад через API або із баз даних, що підтримують цей формат. Його простота та гнучкість роблять його надзвичайно популярним у цій сфері, однак не обмежують лише цим, бо JSON може взаємодіяти із дуже різними популярними мовами програмування.

Структура JSON-файлу інтуїтивно зрозуміла, а сама суть формату кочує від JavaScript об'єктів: «ключ» — «значення». В якості ключа виступає назва, виділена лапками, а значення може мати кілька типів:

- **Рядок**. Один із найпопулярніших варіантів. У подвійних чи одинарних лапках задається текстовий рядок, кожен елемент якого є символом Unicode.
- **Масив**. Тип даних масиву — це впорядкована колекція значень. У JSON значення масиву повинні бути типу рядок, число, об'єкт, масив, логічне значення або null.

- **Логічне значення.** Логічне значення позначаються як «true» або «false». Логічне значення не береться в лапки та розглядається як рядкове значення.
- **Null.** Null представляє значення, яке навмисно залишено порожнім. Коли ключу не присвоєно значення, його можна розглядати як null.
- **Число.** Числа використовуються для зберігання числових значень для різних цілей, таких як обчислення, порівняння або аналіз даних. JSON підтримує як додатні, так і від'ємні числа, а також десяткові коми. Число JSON відповідає формату подвійної точності з плаваючою комою JavaScript.
- **Об'єкти.** Тип даних об'єкта JSON — це набір пар імен або значень, вставлених між {} (фігурними дужками). Ключі мають бути рядками, розділеними комами та унікальними.

Загалом ми виділили основні інструменти, крім яких існує ще дуже багато, однак для розробки невеликої інді-гри цього цілком вистачить. Які із них ми будемо використовувати у власному проєкті буде розглянуто детально нижче.

1.5 Порівняння існуючих інді-ігор

1.5.1 Поняття інді-ігор

Спочатку розглянемо поняття інді-ігор. Насамперед, це відеогра, створена невеликою командою або окремими розробниками без значної фінансової підтримки великих видавців. Такі проєкти зазвичай відрізняються творчою свободою, експериментальними механіками та нестандартними ідеями, оскільки автори не обмежені комерційними вимогами великих студій. Інді-ігри часто фокусуються на оригінальній подачі, унікальному візуальному стилі або інноваційних ігрових механіках, а їх розповсюдження здебільшого відбувається через цифрові платформи [22].

Часто інді-проєкти роблять невеликими та легкими, адже на початку немає великих ресурсів для реалізації великих задумів. Саме тому серед ігор, які мають

потенційну схожість у розробці та дизайні, я б хотів виділити декілька особливих, що надихали саме мене, та розглянути їх детально для порівняння.

1.5.2 «Reigns»

Першою в списку буде стратегічна відеогра 2016 року «Reigns», яка була розроблена лондонською ігровою студією Nerial та видана Devolver Digital. Суть цієї інді-гри зводиться до керування гравцем середньовічним королем, якому вони мають допомогти керувати державою якомога довше [23].

Як було зауважено раніше — це стратегічна відеогра. Задача гравця — приймати одне із двох рішень, які надаються їм, як петиції від радників, народу, армії тощо. Механіка вибору працює як свайп вправо чи ліво. Прийняті рішення впливають на силу та позначку одного або кількох із чотирьох критеріїв: армія, народ, церква та багатство короля. Ці критерії представлені лічильниками, які змінюються в залежності від нашого рішення. До прикладу, рішення збільшити податок збільшить наше багатство, але рівень підтримки народу впаде. Мета: зберегти ці лічильники в балансі, адже коли ми досягаємо максимуму чи мінімуму, настає кінець гри, відповідно до певної причини.

Рисунок 1.4 — Гра Reigns

Стиль гри змінювався в залежності від серії, однак основні принципи залишалися незмінними. Найважливіший критерій — це мінімалістичний дизайн. Позаду основного ігрового поля із яким взаємодіє користувач може бути

фонове зображення або просто одноманітний фон. На самому полі варто виділити основні блоки.

Найвищий блок — блок станів, що є індикаторами та шкалами, що інформують нас про баланс ресурсів, візуально допомагаючи нам робити відповідне рішення. Іконки відображають відповідний стан і, хоч мають мінімалістичний характер, візуально вони добре інформують свою приналежність.

Наступний блок складається із контентної частини. Тут у нас є текст , що розповідає про ситуацію та описує суть проблеми. Нижче розташоване фото персонажа, виконане у мінімалістичному, однак досить привабливому стилі. При здійсненні swipe жесту вправо чи вліво ми можемо побачити текст нашого рішення. Такий спосіб є надзвичайно зручним, економить місце і добре відповідає загальній UI структурі. Спосіб перегортання карт не є унікальним: команда розробників «Reigns» взяла до уваги спосіб відтворення контенту, як у додатку для знайомств Tinder.

Нижче основного блоку розташовано підпис, ім’я чи посада персонажа, що може варіюватися в залежності від вимог чи сюжету.

Найнижчий блок часто несе інформативний та додатковий характер. Там може відображатися прогрес у вигляді кількості пройдених днів чи пунктів, підказки, певні ігрові дані тощо.

Також варто зауважити, що анімації, звук та кольорова гама дібрані досить добре. У нас плавні повзунки, немає надлишку рухів чи лишніх і непотрібних деталей, які б не мали практичної користі. Супровідний звук та музика створює враження, що людина поглиблюється у відповідну епоху, що робить подачу контенту ще кращою. Кольори гарно збалансовані відповідно теорії кольорів, що насамперед дає приємну картинку та створює позитивне враження у користувачів.

Рисунок 1.5 — Стиль гри «Reigns: The Witcher»

Написання сюжету, на який вплинув французький літературний рух Уліпо (фр. Oulipo), керував розробник Франсуа Алліот. Я не буду зупинятися на самому сюжеті, але акцентую увагу на його структурі. Вона побудована за принципом нелінійного сюжету, де вибір гравця суттєво впливає на хронологію та гілку подій, що робить гру неочікуваною та цікавою, що сприяє залученості користувача.

Сама гра отримала «загалом схвальні» відгуки від гравців, де ігровий процес вважали цікавим та винахідливим, а Pocket Games заявив, що, дослівно, це «вдале поєднання стратегії та казуальної мобільної гри, яка змушує вас сміятися, плакати та замислюватися» [24].

Гра була випущена для кількох популярних платформ, не обмежуючись лише певним колом користував. Вона доступна на Android, iOS, Linux, macOS та Windows. До серпня 2019 року було продано два мільйонів копій гри.

Такий успіх дозволив розпочати новий етап із створення франшизи Reigns. За рік, у 2017, вже було випущено сиквел із назвою «Reigns: Her majesty», де збереглася механіка, стиль та ідея, окрім сюжету, що цього разу вже переносить нас у епосу Ренесансу, де ми керуємо монархом, що керує королівством та приймає рішення. У 2018 році вийшла наступна гра — «Reigns: Game of Thrones», де її сюжет був побудований на популярному телесеріалі «Гра престолів». Четверта гра «Reigns: Beyond» була випущена у 2020, а її стиль та сюжет побудовані навколо науково-фантастичного сеттінгу. Остання на цей час гра була

видана у 2022 із назвою «Reigns: The Witcher», що була адаптованою до серії відеоігор «Відьмак».

Як висновок, можна сказати, що попри свій мінімалістичний дизайн, ідею та сюжет, гра має добре продуману структуру UI/UX, унікальний дизайн, механіку вибору та чудовий наратив історії, що змушує гравців не просто розважатись, а думати та приймати важливі рішення. Це в свою чергу доводять позитивні відгуки користувачів та критиків, що потягнуло за собою продовження серії ігор, що означає, що вони мають досить добрий попит серед ігрової аудиторії.

1.5.3 «Lapse»

Наступний продукт, який я хотів розглянути це серія із 2 однокористувацьких ігор.

Перша із них, «Lapse: A Forgotten Future» — це мобільна гра-симулятор політичної дії, заснована на науково-фантастичній історії, написаній та розробленій Корнаго Стефано у 2017. Гравець керує президентом Холлом, лідером неназваної країни, що опинилась посеред екологічної кризи, спричиненої ядерними опадами та тривалою війною. Подібно до гри Reigns, гравець повинен збалансувати вплив своїх рішень на чотири фактори: навколишнє середовище, також відоме як радіація; щастя людей; військова міць; та економіка.

Після кожної смерті гра перезавантажується, але історія все одно розвивається, оскільки гравець застрягає в часовій аномалії, яка робить його безсмертним. Гра закінчується, якщо гравцеві вдається досягти однієї з трьох кінцівок.

Рисунок 1.6 — Гра «Lapse: A Forgotten Future»

«Lapse: Before Zero» — продовження серії, реалізоване у 2018 році. Сюжет обертається навколо фараона, який не пам’ятає свого минулого та має провали із пам’яттю, однак має керувати великою древньою державою, балансуючи ресурсами та рішеннями. Гра має кілька кінцівок, після кожної смерті гравець відновлюється, бо насправді він із майбутнього.

Рисунок 1.7 — Гра «Lapse: Before Zero»

Щодо свого сюжету, дизайну, звуку — ця серія дуже подібна до попередньої, але це не робить її гіршою. Навпаки, ця серія має досить високий рейтинг, хоч не така популярна за свого попередника, проте вона зберігає свій рівень та глибину історії, змушує нас вірити у наратив і заглиблюватись в історію, попри мінімалізм у підході, який легко компенсувався вагою вибору та контролем. Особливістю є голоси персонажів, що немов ідеально відтворюють

минулі втрачені мови. З мого власного досвіду, це зіграло чималу роль у оцінювані цього продукту.

В загальному висновку ми розглянули дві цікаві реалізації ігор, у стилі яких буде створено власну інді-гру. Ми бачимо, що такий підхід, попри свій мінімалізм та простоту, чітко показує, як різні розробники по своєму реалізовують цікаві рішення та створюють неповторні ігри завдяки вдало дібраним компонентним рішенням та глибині історії, що занурюють користувачів у ігрову динаміку попри серйозну реалістичну графіку чи можливості.

РОЗДІЛ 2
ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ВЕБГРИ

2.1 Концепція гри

Концепція гри є основою всього проєкту, оскільки визначає її жанрову спрямованість, сюжетну структуру, ключові механіки та цільову аудиторію. У межах даної бакалаврської роботи розробляється інтерактивна наративна гра за провідна ідея якої повністю побудована на античній літературі, а саме на епічній поемі Гомера, що має назву «Іліада». Гра буде реалізована у стилі таких проєктів, як «Reigns», Lapse: Before Zero» та «Lapse: A Forgotten Future», де центральним елементом виступає система вибору та наслідків.

Основна ідея гри полягає в адаптації класичного літературного твору до сучасного інтерактивного формату, що дозволяє користувачу не лише ознайомитися з подіями Троянської війни, а й безпосередньо впливати на їх перебіг через ухвалення рішень. Гравець бере на себе роль одного з ключових персонажів, Ахілла, який повинен балансувати між військовими ресурсами, підтримкою союзників, моральним авторитетом та власною метою.

Гра будується на принципі послідовного прийняття рішень, де кожна подія подається у формі ситуаційної картки або діалогу з кількома варіантами дій. Кожен вибір змінює параметри головного героя, а також може призвести до нових сюжетних гілок, конфліктів або альтернативних фіналів. Такий підхід забезпечує високу взаємодію та стимулює стратегічне мислення.

Таблиця 2.1. Основні параметри гри.

Назва параметра	**Опис**	**Максимальне значення параметра**	**Мінімальне значення параметра**

Репутація	Визначає підтримку та віру союзників та друзів Ахілла у нього.	Цар Агамемнон, полководець ахейців, почне заздрити і спробує убити або прогнати героя.	Герой може втратити свою репутацію та бути вигнаним своїм народом.
Слава	Визначає чи має головний герой честь у очах друзів, ворогів та богів.	Заздрість у ворогів, друзів та богів, що змовляться для його смерті.	Змусить відвернутися від нього усе живе.
Підтримка богів	Визначає наскільки боги Олімпу задоволенні героєм.	Це посіє чвари на Олімпі та почне війну богів.	Боги відвернуться від героя та переконають мати виказати його вразливе місце.

Саме ці три критерії будуть основними показниками прогресу гри, а мета гравця полягатиме у балансуванні рішеннями так, щоб не наблизити до критичного мінімуму чи максимуму певного критерію.

Візуальна концепція передбачає мінімалістичний інтерфейс із картковою подачею подій, стилізованою античною естетикою, тематичними ілюстраціями та акцентом на текстову взаємодію. Такий формат дозволяє зосередити увагу на сюжеті, рішеннях та атмосфері.

Цільова аудиторія гри включає:

- студентів;
- школярів;
- шанувальників історичних ігор;
- користувачів, зацікавлених у міфології;

- любителів сюжетних стратегій.

Таким чином, концепція гри поєднує розважальний, освітній та культурний аспекти. Концепція спрямована на створення продукту, що не лише розважає, а й сприяє глибшому розумінню античної літератури, історичного контексту та складності людських рішень. Основною перевагою такого підходу є здатність представити класичний твір у формі, яка відповідає сучасним цифровим тенденціям та потребам нової аудиторії.

2.2 Вебінструменти

У цій роботі буде використано основні інструменти веброзробки, такі як HTML, CSS та JS, що є базовими для створення будь-якого контенту в web.

Однак було вирішено не використовувати сторонні бібліотеки для анімації чи створення ігор, а розробити весь код додатку власноруч з використанням мови програмування JavaScript.

Замість CSS пропонується використати препроцесор SaSS із синтаксисом SCSS, адже це спрощує розробку та збільшує можливості при роботі із стилізацією контенту.

В якості frontend-фреймворка пропонується використання Vue, оскільки порівняно з React та Angular, розглянутими в розділі 1.4, Vue має баланс структури та свободу дій, простий та інтуїтивно зрозумілий синтаксис, має багато вбудованих рішень, і відповідно підходить для простих та невеликих проєктів, яким є дана робота.

Також для кращого процесу розробки я використовуватиму синтаксис TypeScript для Vue. Такий підхід допоможе виявляти помилки типізації на ранніх стадіях і краще структурувати програмну реалізацію.

В роботі було прийнято рішення використати localStorage для зберігання даних користувачів без їх збереження в централізовану базу даних. Цей підхід вимагає розгортання серверної частини, що не входить в задачі даної роботи. Додатково для більшої залученості аудиторії та створення атмосфери притаманній «Іліаді», пропонується використати ШІ-інструмент для генерації

аудіо ElevenLabs, щоб створити враження, що розроблені ігрові персонажі живі [26].

У вкладці «Text to speech» даного інструменту після реєстрації ми можемо перетворити звичайний текст у вимову, що віддаватиме відлунням тієї епохи. Інструмент надає декілька безкоштовних кредитів, чого цілком вистачить для створення невеликої колекції вимов; ми маємо можливість обирати, добирати та видозмінювати різні голоси, підбираючи необхідні для нашої сцени.

Щодо транскрипції та досягнення рівня вимови давньогрецької, ми не маємо подібного інструменту мови у ElevenLabs, проте можемо обійти цю проблему через створення відповідної транскрипції [].

Наприклад, речення «**Fos pee-ROS ai-oh-NEE-oo psoo-KHAYN hay-GAY-tai**» в сукупності із правильно вибраною вимовою буде створювати античний ефект.

Рисунок 2.1 — Створення давньогрецької вимови через транскрипцію у ElevenLabs

Отже, в даному підрозділі було визначено основні інструменти для розв’язання поставлених в роботі задач. Способи застосування будуть детально розглянуто у наступному розділі.

2.3 Структура нелінійного сюжету

Як заначено в розділі 1, що чи не найважливішу роль у грі відіграє саме сюжет. Лише його гарно продумана структура, внутрішня ідея та конфлікт дозволяють утримувати користувачів та залучати їх до історії. Сюжет може мати чимало розгалужень, які відбуваються через різні рішення гравців. У такому

випадку незручно записувати і зберігати інформацію про сюжет лінійно, бо це значно ускладнить його програмну обробку та інтерпретацію. Тому в роботі пропонується використання формату JSON для моделювання нелінійності сюжету.

Однією із крутих особливостей JSON є вкладеність. Це типовий спосіб розміщення об’єкту всередині об’єкту, що має свої переваги, як от спрощене читання, логічність та структуризація. Приклад вкладеності можна побачити на рис. 2.2, де атрибут «article» є об’єктом із вкладеними полями, доступ до яких легко отримати за послідовним зверненням до полів, наприклад **obj_name.article.title**, що дозволить нам прочитати значення цього поля.

Рисунок 2.2 — Демонстрація вкладеності JSON

Ще однією перевагою використання JSON у нашому проєкті, буде його сумісність та подібність із JavaScript синтаксисом об’єктів. Для веброзробників дуже важливо розуміти як будуються та використовуються json-файли, тому для нас стане перевагою розуміння цього інструменту. JavaScript повністю сумісний із таким типом файлів та має свої методи та інструменти для роботи із ним, що дозволяє нам зробити очевидний висновок, що саме формат JSON для нашого сюжету буде найкращим рішенням.

Тепер перейдемо безпосередньо до створення самого сюжету. Оглядати JSON ми будемо через онлайн-сервіс jsonhero.io, який дозволяє зручно переглядати файли JSON: їхню внутрішню структуру, контент, вкладеність, типи

даних тощо, що дозволяє нам ефективно будувати та наповнювати структуру сюжету [28].

Рисунок 2.3 — Робочий стіл ресурсу jsonhero.io

Всередині нашого JSON-файлу сюжету буде масив із об’єктів. Кожен об’єкт матиме унікальний ідентифікатор, наприклад «chapter1», що є назвою першого розділу нашої історії.

Візьмемо цей розділ для прикладу, щоб розглянути його детальніше на вкладені структури. У кожному полі із назвою типу «chapter[number]» у нас має бути як мінімум три великі конструкції: два масиви та один об’єкт. Це мінімальна кількість полів, однак в процесі розробки гри може виникнути необхідність додати ще кілька полів, тому початкова кількість не є критичною, не обмежується чим числом та може дозволити собі зростання структури.

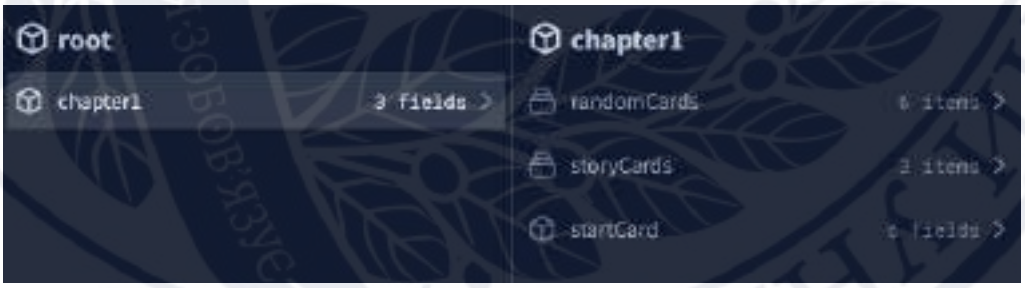

Рисунок 2.4 — Внутрішня структура першого розділу

Розглянемо перший масив із назвою «randomCards». У ньому будуть вкладені впорядковані об’єкти із однаковими полями. Основний із них це «id», що є ідентифікаційним ключем нашого об’єкта. Зазвичай це має бути число,

однак іноді у нас будуть спеціальні унікальні картки, тому вони можуть мати текстовий вигляд, як у прикладі нижче. Ці карти використовуватимуться у специфічних випадках, коли наше рішення можу змінюватися в залежності від сюжету, а нам потрібно буде створити нелінійну структуру. Саме такий спосіб ідентифікації карт допоможе нам досягнути цієї цілі.

Рисунок 2.5 — Спосіб представлення ідентифікаторів

Після ідентифікатора ми маємо два важливі поля для візуального представлення. «ulr» — відповідає за назву фото персонажа та його розширення, яке ми будемо підключати згодом. «voise» — є полем, що відповідає за назву голосової доріжки персонажа, так як ми раніше вирішили, що наші персонажі будуть мати власні короткі доріжки вимови для глибшої історії гри.

Далі у нас ідуть послідовні об’єкти із глибшими вкладеностями «ua», «uk», «ar», що є абревіатурами мов. Це є основним необхідним кроком для створення багатомовної гри. Усі вони матимуть однакову кількість полів, а самих полів, що стосуються мов може бути десяток, однак я вирішив зупинитися на 2-3 основних мовах, щоб не ускладнювати структуру та більше зосередитися на розробці. Проте технологія багатомовності у грі залишається незмінною.

Рисунок 2.6 — Багатомовність у JSON-файлі

Всередині кожного поля різної мови ми маємо однакову структуру. Відповідно це поле «title», що відповідає за текст, який буде представлятись у кожній ігровій ситуації та створюватиме експозицію. Поле «name» відповідає за підпис ім’я персонажа внизу його ілюстрації. Ще нижче ми маємо поле «choices», що має глибшу вкладену структуру і відповідає за два рішення, які має зробити наш гравець. Для кожного рішення при свайпі вліво «leftOption» та вправо — «rightOption». Кожна із них має відповідне текстове поле, що відображає нам варіант вибору, масив «impact» відповідає за числовий вплив, яке матиме наше рішення на три параметри у грі. Поле «next» визначає чи йде після даного слайду спеціальний наступний слайд. Зазвичай це поле порожнє, бо у полі випадкових карт вибиратимуться випадкові карти, окрім тих, як було зазначено раніше, мають певний вплив на історію та потребують послідовності. Саме ця відмітка дозволить нам реалізувати та налаштувати перехід до наступної карти по ідентифікатору, якщо це потрібно.

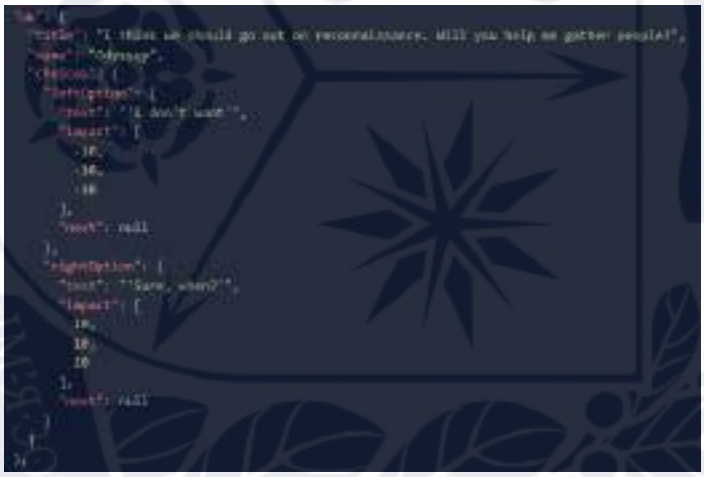

Рисунок 2.7 — Структура всередині поля, що відповідає англомовній розкладці

Відповідно у розкладці української мови зберігаються всі ті самі поля, змінюються лише значення.

Наступний великий блок складає масив «storyCards». Це відповідно є колекцією подібних до попереднього блоку об’єктів з однаковими полями та та структурою, однак єдиною структурною відмінністю буде поле «next», адже цього разу карти ситуацій випадатимуть не у випадковому порядку, а у чіткій

сформованій послідовності, необхідній для нелінійного руху сюжету, що впирається на вибір гравця.

Рисунок 2.8 — Структура всередині поля «storyCards» має заповнене поле «next»

Ще одним важливим полем є «startCard». Мета цього поля полягає у відправній точці нашого розділу, тобто має певну унікальну інформацію та візуалізацію на початку гри. Відповідно вона має власний ідентифікатор, посилання та назву фото, голосовий супровід, якщо це потрібно, а також текст та значення рішень, вплив яких відповідно поки дорівнює 0. Як і у інших карт, ця має поля для різних мов, щоб ігровий процес був зручний для користувачів, які розуміють лише певну мову.

Рисунок 2.9 — Структура та контент «startCard»

Ми не обмежуємо лише цими трьома полями, адже можна додати поле, що відповідає за закінчення гри у певному розділі та багато інших, проте кожен розділ має містити мінімум три основних поля, які ми детально розглянули вище, бо саме вони є основними у розвитку сюжету та динаміці гри.

Отже, ми визначили формат, який ідеально відповідає вимогам побудови нашого нелінійного сюжету, його переваги та можливості для нашого проєкту, потім на наочному прикладі розглянули його структуру та будову сюжету у вигляді, який вже буде у використанні та розробці самої гри.

2.4 Моделювання ігрової механіки карткового вибору

Механіка карткового вибору є основним елементом взаємодії гравця з грою у даному проєкті. Вона використовується для реалізації нелінійного сюжету, де кожне рішення впливає на розвиток подій та ігрові ресурси. У науковому контексті така система належить до інтерактивного наративу (interactive fiction) та branching narrative, де сюжет будується як послідовність пов’язаних подій і виборів [29].

Кожна подія у грі реалізується як окрема картка. Вона містить опис ситуації, варіанти дій, наслідки та перехід до наступної події. Така структура може бути представлена у вигляді дерева рішень або графа станів. Це дозволяє легко розширювати гру та змінювати сюжет без перебудови всієї системи [2].

У проєкті на основі «Іліади» цей підхід є логічним, оскільки оригінальний твір складається з окремих подій, конфліктів і рішень, які можна розділити на сценарні блоки.

Кожна картка містить:

- опис події;
- персонажа або контекст;
- 2 варіанти вибору;
- наслідки для ресурсів;
- перехід до наступної події.

Сюжет реалізується як data-driven система, де дані зберігаються у JSON або базі даних, тобто це підхід, у якому логіка та поведінка програми визначаються не жорстко в коді, а через зовнішні дані (файли або бази даних). Це спрощує реалізацію у вебформаті та дозволяє швидко змінювати контент без зміни коду.

У грі використовуються ресурси (репутація, слава героя, підтримка богів), які змінюються залежно від вибору гравця. Це створює ефект довгострокових наслідків і відповідає концепції meaningful choice — коли рішення мають реальний вплив на гру.

Алгоритм роботи системи:

- показ картки;
- вибір гравця;
- застосування наслідків;
- оновлення ресурсів;
- перехід до наступної події.

Такий цикл добре підходить для вебігор і легко реалізується через реактивні фреймворки.

Карткова система має переваги: простий інтерфейс, низьке навантаження на користувача та зручність для мобільних пристроїв. Через це вона використовується в інді-іграх та вебпроєктах.

Отже, карткова механіка поєднує простий інтерфейс, нелінійний сюжет і систему ресурсів, що робить її зручною для реалізації інтерактивних історій у вебіграх.

2.5 Проєктування архітектури програмного забезпечення

2.5.1 SPA

Через те, що наша гра буде написана на Vue.ts, варто розглянути архітектуру цього способу побудови вебігор. Основний термін, який потрібно добре розуміти — це SPA (Single Page Aplication) [30].

Раніше ми вже визначили, що вебсайти будуються завдяки стеку технологій HTML, CSS та JavaScript, однак такий підхід часто називається MPA (Multi Page Application), а його суть зводиться до того, що для багатосторінкового вебсайту створюється чимало нових сторінок html, тобто кожна взаємодія та перехід вимагає нової сторінки. Це в свою чергу означає те, що користувач мусить чекати повного відтворення та завантаження всіх ресурсів знову і знову, а при слабкому інтернеті ця проблема стає дуже критичною, особливо, якщо вебсайт складний та має багато сторінок, як от сайти із продажів із довгими каталогами.

Сучасні фреймворки, як от React, Angular та Vue, з яким ми будемо працювати, використовують підхід SPA. Це популярний тип вебдодатків, які працюють без перезавантаження сторінки. Основні ресурси (HTML, CSS і JavaScript) лише раз завантажуються під час першого запиту користувача, а внутрішні інтерактивні елементи додаються динамічно. Користувачеві може здаватися, що він переходить новими сторінками, але насправді перебуває на одній, бо змінюється зовсім невелика частина даних. Це швидко, зручно й без неприємних бліків [30].

Принцип роботи SPA полягає в початковому завантаженні основної HTML-структури, стилів та JavaScript-логіки, після чого подальша взаємодія користувача із системою відбувається шляхом локального оновлення даних. При прийнятті рішення гравцем змінюються внутрішні змінні стану, після чого відповідні компоненти автоматично оновлюють лише ті частини інтерфейсу, які потребують змін. Наприклад:

- оновлюються показники ресурсів;
- змінюється текст події;
- відображаються нові картки;
- активуються сценарні прапори;
- перевіряються умови завершення гри.

Таким чином, користувач не стикається з технічними затримками, а сам процес гри сприймається як цілісний і безперервний.

Для проєкту гри за мотивами «Іліади» SPA є особливо доцільним через наявність великої кількості послідовних сюжетних рішень, де кожен вибір має потенційні наслідки для подальшого розвитку історії. Нелінійний сюжет потребує постійного оновлення стану без руйнування загального контексту гри. Якщо при кожному новому рішенні відбувалося б повне перезавантаження сторінки, це негативно впливало б на:

- занурення;
- швидкість гри;
- цілісність наративу;
- зручність інтерфейсу.

SPA, навпаки, забезпечує:

- безперервність сценарію;
- миттєві переходи;
- плавність інтерактивності;
- стабільність користувацького досвіду.

Таким чином, вибір SPA для реалізації вебгри є не лише технічно обґрунтованим, але й стратегічно оптимальним рішенням, що відповідає сучасним вимогам frontend-архітектури.

Наступним важливим питанням, яке потрібно розглянути — це компонентна архітектура. Вона є одним із фундаментальних принципів сучасної frontend-розробки, що передбачає поділ програмного забезпечення на незалежні функціональні модулі, кожен із яких відповідає за конкретну частину інтерфейсу або логіки системи. У контексті створення вебгри з нелінійним сюжетом компонентний підхід є особливо важливим, оскільки дозволяє поєднати складну сценарну систему, інтерактивний інтерфейс та управління станом у межах структурованої та масштабованої архітектури. Дослідження у сфері software architecture підтверджують, що модульна організація значно покращує підтримуваність, повторне використання коду та адаптивність цифрових систем. Bass, Clements та Kazman у праці *Software Architecture in Practice* підкреслюють,

що компонентність є ключовою умовою ефективного проєктування складних програмних продуктів [31].

2.5.2 Компонентна структура

У межах даного проєкту компонентна структура застосунку будується навколо центральної ігрової логіки та системи відображення карткових подій. Основними структурними елементами гри виступають:

- App — головний контейнер застосунку, який координує відображення основних ігрових сцен;
- Card — компонент, що відповідає за показ поточної сюжетної картки;
- Settings — компонент, що відповідає за налаштування гри;
- Menu — головна та початкова сторінка при завантаженні гри;
- Header — система візуалізації ресурсів (підтримка богів, слова та репутація);
- Footer — відображається прогрес по днях у грі, а також буде посиланням на налаштування;
- Game — основний модель для відображення гри та її механіки.

Кожен із цих компонентів виконує чітко визначену функцію, що відповідає принципам separation of concerns. Наприклад, Card відповідає виключно за відображення події та отримання вибору, тоді як Settings обробляє логіку змін налаштувань чи прапорів мови чи станів. Такий розподіл дозволяє мінімізувати залежності між модулями та спрощує процес тестування й розширення системи.

З наукової точки зору компонентний підхід відповідає принципам reusable software modules, що широко використовуються у SPA-архітектурі та сучасних JavaScript-фреймворках. Дослідження компонентно-орієнтованих інтерфейсів вказують, що така структура:

- спрощує масштабування;
- знижує складність підтримки;
- покращує читабельність коду;
- дозволяє швидко інтегрувати нові функції;

- сприяє довгостроковій стабільності проєкту.

Взаємодія між компонентами будується за принципом централізованого обміну даними:

1. Game обирає поточну картку;
2. Card відображає її користувачу;
3. Game отримує рішення та оновлює ресурси;
4. Header та Footer відображає зміни;
5. Game активує наступну подію.

Подібна схема забезпечує передбачуваність роботи системи та відповідає сучасним підходам reactive UI architecture.

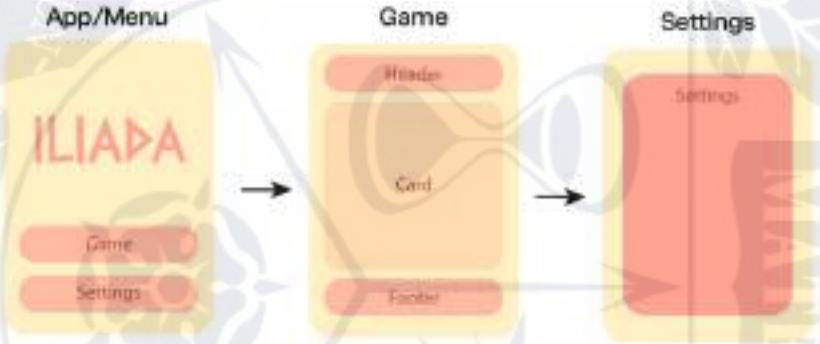

Рисунок 2.10 — Узагальнена схема компонентного підходу гри

2.5.3 Глобальний стан

Тепер розглянемо не менш важливу тему — систему управління станом гри. Це необхідний ресурс та інструмент для того, щоб зручно взаємодіяти між компонентами, передаючи їм важливу інформацію. Для того, щоб це було реалізовано добре і працювало, я мушу створити окремий файл TypeScript, який міститиме реактивну зміну із певними даними, наприклад, обраною користувачем мовою гри, збережений прогрес, налаштування звуку, шрифтів тощо.

Із файлу Story.json ми отримуватимемо дані щодо сюжету лише у компонент Card, однак його надходження коригуватиме головний файл стану Storage.ts, що збереже останню карту і допоможе нам завжди повертатися до

останнього прогресу. Цей файл буде доступний усім компонентам, бо міститиме мову, певні налаштування, а тому його ієрархія та доступність дуже важлива.

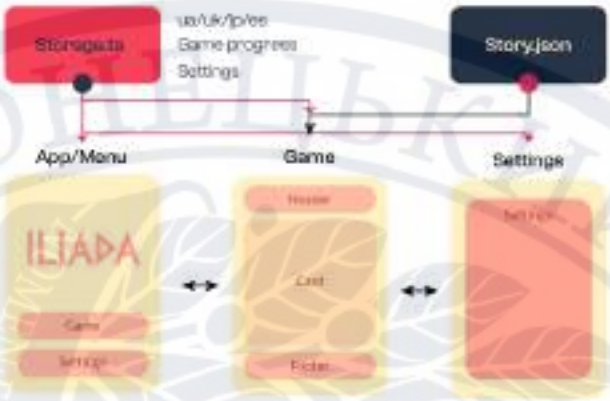

Рисунок 2.11 — Узагальнена структура стану

2.5.4 Збереження даних

Однією з важливих складових архітектури сучасної вебгри є система збереження користувацьких даних, яка забезпечує безперервність ігрового процесу, підтримку прогресу та збереження індивідуальних рішень гравця. У межах даного проєкту для реалізації цієї функції обрано технологію localStorage, яка є частиною Web Storage API та дозволяє зберігати дані безпосередньо у браузері користувача без необхідності використання серверної інфраструктури. Специфікація Web Storage API, стандартизована W3C, визначає localStorage як простий механізм довготривалого клієнтського збереження структурованої інформації [21].

Вибір localStorage обумовлений насамперед специфікою бакалаврського проєкту, орієнтованого на автономну сюжетну вебгру без обов'язкової серверної частини. Такий підхід дозволяє реалізувати:

- збереження поточного прогресу;
- ресурси гравця;
- історію виборів;
- активні сюжетні прапори;
- відкриті фінали;
- користувацькі налаштування.

Завдяки цьому гравець може продовжити проходження навіть після закриття браузера або оновлення сторінки, що суттєво підвищує практичну цінність застосунку.

З точки зору software engineering, використання localStorage є ефективним рішенням для невеликих і середніх SPA-проєктів, де не потрібна складна авторизація, відсутня багатокористувацька синхронізація та важлива швидкість.

Для бакалаврського проєкту недоліки localStorage не є критичними, оскільки система орієнтована переважно на демонстрацію архітектурних та гейм-дизайнерських рішень.

2.6 Проєктування інтерфейсу користувача

Чи не найважливішою річчю, що впливає на гарний досвід гравця і йде після одразу сюжету, є дизайн. Саме його візуальна привабливість та доречність визначають наскільки продукт буде автентичним, впізнаваним та цікавим. Також ми повинні розповідати історію не лише через текст, а візуально через правильно гарно побудований інтерфейс і кольори [32].

Ми почнемо із візуальної концепції інтерфейсу. Наша історія стосується епічної античної події, тому дизайн повинен відповідати стилізації античності. Однак спершу ми окреслимо головні шаблони, які використаємо при створенні дизайну.

Перший визначальний критерій — це стиль. Серед різних стилів я обрав мінімалізм із декоративною деталізацію, суть якого полягає у створенні форм та елементів, які не переобтяжують користувача лишніми образами та деталями, але дають рівно стільки, скільки потрібно для базового розуміння [33].

На рисунку нижче зображено накреслений шаблон макету для стартової сторінки гри. Спочатку я виділю тут фон, який насправді може бути мінімалістичним та задаватися суцільним кольором або картинкою, проте єдине обмеження щодо другого варіанту полягає у відповідності ілюстрації до історії, спрощені кольорові гами та відсутність лишніх форм.

Саме ігрове меню також складатиметься із простих елементів. Найпершим, що буде кидатися в оці користувача — це титри чи епічна ілюстрація гри. У нашому випадку цей елемент буде найбільшим на гральному полотні і може змінюватися відповідно до вподобань, але його роль залишається незмінною.

Нижче розташовуватимуться дві кнопки, що відповідатимуть своїм назвам. Одна дозволить нам розпочати чи продовжити гру, а інша перейти до меню налаштувань.

Рисунок 2.12 — Макет головного екрану

Наступний блок — налаштування. І тут я не хотів би зупиняти великої уваги, адже цей буде реалізовано як pop-up меню, із відповідними налаштуваннями. Тому я одразу перейду до блоку, де користувач проводитиме найбільше часу — ігрового полотна.

Особливість цього полотна полягає у трьох основних блоках. Найвищий, «Header», міститиме три дібрані спеціальні мінімалістичні іконки для відображення стану гри. Нижче йде основний блок, що міститиме текст ситуації, фото героя, що розмовляє із нами та його ім’я. Останній блок є інформативним та буде відображати прогрес грим у днях та буде посиланням для переходу у меню.

Рисунок 2.13 — Шаблон ігрового полотна

Окремо варто виділити механіку руху картки. Моя ідея не нова, але вона полягає у тому, щоб при кліку було доступне перетягування карти вбік, від чого зміщується її обертання та з’являється текст вибору.

Рисунок 2.14 — Ідея механіки руху картки

Отже, ми визначили основний підхід із стилем. Його переваги цілком зрозумілі — користувач не має переобтяжений непотрібними елементами інтерфейс, що дозволяє йому зручно із ним взаємодіяти. Також цей підхід дозволяє не заповнювати картину гри повністю вигадкою автора і дозволяє таким чином гравцю домальовувати незавершені образи у власній голові. Ще це знижує навантаження на розробку, не вимагає чимало ресурсів та інструментів для створення інтерфейсу.

Тепер перейдімо до важливого питання кольорів. У процесі проєктування візуального стилю гри кольорова палітра була сформована з урахуванням історичної тематики, психології сприйняття та принципів сучасного UX/UI-дизайну. Основу інтерфейсу становлять теплі світло-піщані та пергаментні

відтінки, які асоціюються з античними рукописами, стародавніми текстами, мармуром і бронзовою естетикою давньогрецької культури. Використання таких кольорів дозволяє створити атмосферу історичної автентичності, підсилює зв'язок із тематикою «Іліади» та сприяє формуванню цілісного стилістичного середовища.

Контрастні темні кольори були використані для текстових елементів, рамок та ключових інтерфейсних акцентів. Це рішення забезпечує високу читабельність контенту, покращує доступність інтерфейсу та дозволяє чітко виділяти важливі елементи управління. Темно-сині та глибокі темні відтінки також додають інтерфейсу драматичності, що відповідає військово-політичному характеру сюжету, пов'язаного з подіями Троянської війни. Така кольорова композиція поєднує історичну стилізацію з функціональністю сучасного цифрового продукту.

Додаткові акцентні кольори використовуються для підкреслення інтерактивних елементів, ресурсних змін та сюжетно важливих деталей. Це відповідає принципам візуальної ієрархії, згідно з якими користувач швидше орієнтується в інтерфейсі та ефективніше сприймає інформацію. Світлі нейтральні кольори також сприяють зменшенню когнітивного навантаження, створюючи баланс між декоративністю та зручністю [35].

Рисунок 2.15 — Сформована палітра кольорів

Обраний декоративний шрифт «Cormorant Garamond» із виразною стилізацією підсилює міфологічну атмосферу гри, водночас зберігаючи впізнаваність і художню унікальність, додаючи візуальній подачі елементів культурної стилізації.

Таким чином, кольорова та типографічна система гри була сформована на основі поєднання:

- історико-культурної відповідності;
- психологічного впливу кольору;
- принципів usability;
- емоційного занурення;
- покращення навігації та сприйняття.

Щодо іконок та стилізації я вирішив додати елементи, що відповідають античності.

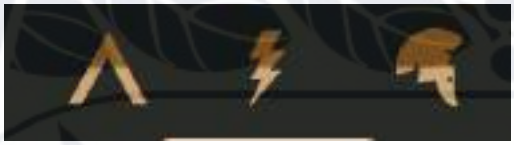

Рисунок 2.16 — Блок «Header» із мінімалістичними іконками критеріїв

Нижче ми бачимо сформовану картку для початку гри, яка несе у собі комбінацію кольорів, стилів та іконок, що відповідають античності та настрою гри.

Рисунок 2.17 — Стартова картка гри з урахуванням античного стилю

Ще одна важлива деталь у створенні гарного досвіду гравця та створення глибшої атмосфери — це саунд-дизайн та музичний супровід. У якості останньої було обрано кількагодинний музичний файл у відкритому доступі в інтернеті, що відповідає античності та не має стрімкої динаміки. Основна роль цього

музичного супроводу є створення фону та занурення у гру. Проте користувач, за бажанням, може вимкнути цю функцію у налаштуваннях.

Щодо саунд-дизайну, то він полягає у двох основних напрямках. Перший відповідає за додавання звуків взаємодії, тобто після натиснення певних кнопок, перелистування і т.д. Другий полягає у створенні голосів для кожного із персонажів. У попередніх пунктах було зазначено, що ми використаємо онлайн ШІ-інструмент для генерації мовлення ElevenLabs. Завдяки створенню транскрипції, що відповідає стандартам вимови давньогрецької, ми можемо не ідентично, але гарно відтворити вимову, що надасть персонажам більшої глибини та античного звучання [36].

Отже, ми визначили основні підходи для розробки дизайну інтерфейсу користувача так, щоб ця важлива складова справляла позитивне враження на гравця. Мінімалістичний дизайн відкидає непотрібні деталі та дає простір для уяви, залишаючи лише необхідне для розвитку сюжету. Кольори, іконки та стилізовані елементи мають важливе значення для створення відповідної картинки, що відповідає епосі у грі, а музичний супровід та голоси персонажів із давньогрецькою вимовою додають всій картині цілісного атмосферного вигляду для гарного досвіду користувача та його залученості до історії.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ГРИ

3.1 Опис середовища розробки

У ролі середовища розробки для нашого проєкту буде обрано IDE Visual Studio Code. Це безкоштовний, легкий, але потужний редактор вихідного коду від Microsoft, що працює на Windows, macOS та Linux випущений у 2015 році. Він підтримує широкий спектр мов програмування, має вбудовану підтримку Git, відлагодження (debugging) та велику бібліотеку розширень, що робить його одним із найпопулярніших інструментів серед розробників.

На відміну від свого повноцінного IDE-аналога, Visual Studio, VS Code розроблений як легкий і добре настроюваний, що робить його ідеальним для розробників, які хочуть швидко та ефективно кодувати без зайвих речей.

Рисунок 3.1 — Середовище розробки Visual Studio Code

Серед основних переваг цієї IDE варто виділити наступні [37]:

1. Крос-платформна сумісність. VS Code доступний для Windows, macOS та Linux, що дозволяє розробникам використовувати один і той самий редактор незалежно від операційної системи.

2. Вбудована інтеграція з Git. Однією з найпотужніших функцій VS Code є його безшовна інтеграція з Git. Ви можете клонувати репозиторії, фіксувати зміни, надсилати оновлення та керувати гілками – і все це в межах редактора.

3. Інтелектуальні пропозиції коду (IntelliSense). VS Code пропонує інтелектуальне завершення коду, підсвічування синтаксису та виявлення помилок, що пришвидшує кодування та зменшує кількість потенційних помилок. IntelliSense підтримує JavaScript, Python, C++, Java та багато інших мов чи фреймворків.

4. Широкий ринок розширень. VS Code дуже добре налаштовується завдяки великій бібліотеці розширень. Розробники можуть встановлювати теми, мовну підтримку, інструменти налагодження та додаткові функції для покращення робочого процесу.

5. Вбудований термінал. Вбудований термінал дозволяє розробникам виконувати команди, не виходячи з редактора, заощаджуючи час і підвищуючи ефективність.

6. Можливості налагодження. VS Code включає потужний інструмент, який підтримує точки зупинки, покрокове налагодження та перевірку змінних, що полегшує пошук та усунення несправностей.

7. Легкість та висока продуктивність. На відміну від традиційних інтегрованих середовищ розробки (IDE), VS Code є легким та оптимізованим для швидкості, що забезпечує безперебійне кодування навіть на машинах з низьким рівнем продуктивності.

8. Підтримка декількох мов програмування. З коробки VS Code підтримує такі мови, як JavaScript, Python, TypeScript, C++, C#, HTML, CSS та PHP. За допомогою розширень ми можемо додати підтримку Go, Rust, Ruby, Swift та багатьох інших.

Серед мінусів варто зауважити, що VS Code може бути ресурсоємним при встановленні багатьох розширень, йому бракує деяких розширених функцій повноцінних IDE, таких як Visual Studio або JetBrains IntelliJ і іноді редактор вимагає ручного налаштування для деяких мов програмування.

Однак не зважаючи на ці мінуси, саме цей редактор коду ідеально підходить для новачків та професіоналів, а особливо для веброзробників (чудово підходить для HTML, CSS, JavaScript).

3.2 Підготовка до розробки

3.2.1 Встановлення проєкту

Як ми вже вирішили раніше, розробка відбуватиметься на JavaScript фреймворку Vue із TypeScript синтаксисом, тому ми на власному комп'ютері ми встановлюємо node.js, а після цього ми створюємо сам проєкт через термінал у необхідній директиві чи папці завдяки офіційній команді від Vite **npm create vue@latest.** Далі ми робимо необхідні кроки, створюючи назву проєкту, налаштовуючи його залежності та додаткові можливості [38].

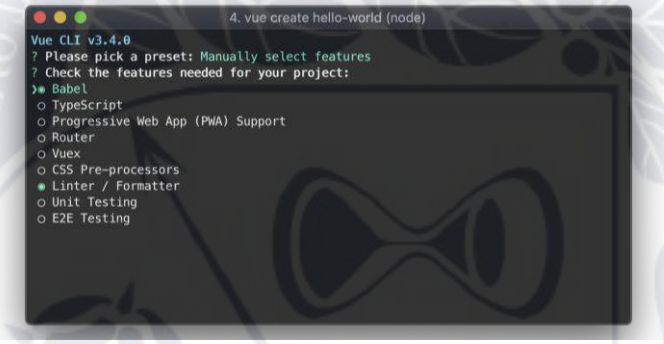

Рисунок 3.2 — Створення та налаштування проєкту Vue

Окремо варто звернути увагу, що проєкт створюється у поєднанні із Vite. Це інструмент для фронтенд-збірки нового покоління, який робить акцент на швидкості та простоті. Його назва, яка французькою означає «швидкий», ідеально описує його функції: Vite неймовірно швидкий як у робочих процесах розробки, так і у виробничому середовищі [39].

Традиційні інструменти, такі як Webpack та Parcel, часто мають недоліки у швидкості та складності конфігурації, тому як Vite вирішує ці питання через свою блискавичну швидкість, спрощену конфігурацію, підтримку з «коробки», що не вимагає додаткового завантаження плагінів, оптимізовану збірку для виробництва, має надійну екосистему плагінів тощо [39].

Після успішного створення проєкту, необхідно налаштувати систему папок та інших даних, поки проєкт ще пустий. Попередньо я встановлюю SaSS у проєкт

через термінал завдяки команді **npm insall sass**, щоб мати змогу працювати із препроцесором.

3.2.2 Робота із файлами

У папці src проєкту я створюю нову із назвою styles і створюю кілька scss-файлів перед початком роботи.

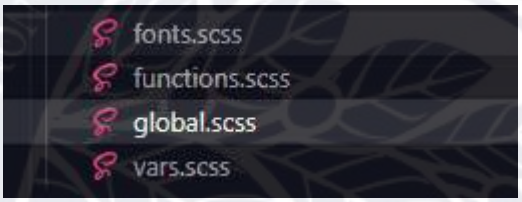

Рисунок 3.3 — SCSS-файли для налаштування

Кожен файл матиме своє унікальне призначення. Наприклад, global.scss міститиме стилі, які скидатимуть браузерні стилі за замовчуванням. Це необхідний крок у кожній веброзробці. Файл vars.scss міститиме змінні scss, які часто використовуватимуться, наприклад, кольори чи шрифти. Файл fonts.scss відповідатиме за роботу із шрифтами, їх підключення, а functions.scss міститиме функції чи міксини scss, наприклад для перетворення пікселів у відносний та адаптивний формат rem/em.

Рисунок 3.4 — Приклад вмісту файлів global.scss та vars.scss

Опісля ми підключаємо ці файли глобально у конфігураційному файлі vite.config.ts:

Рисунок 3.5 — Підключення scss файлів до глобально

Щодо global.scss, що містить стилі для скидання, його буде підключено через імпорт у main.ts для уникнення зациклення.

Наступний крок — робота із директивами та файлами контенту. Ми будемо працювати у папці public, бо це дає нам перевагу роботи із статичними файлами і не потребує обробки. Тут ми створимо дві теки — img для зберігання ілюстративного контенту персонажів та декорацій та sounds для збереження інтерактивних звуків, фонової музики та голосів персонажів.

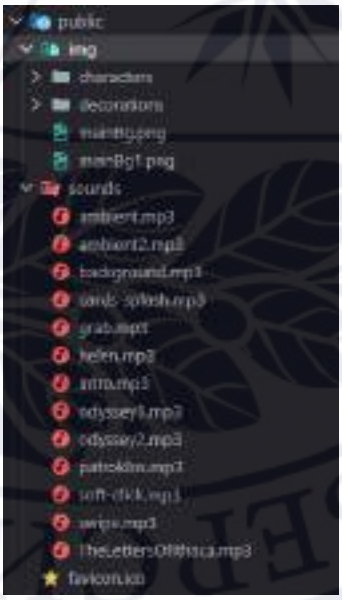

Рисунок 3.6 — Тека public із контентом

Потім нам треба створити у теці scr теку із даними по сюжету, тобто яка міститиме JSON-файл чи файли по сюжету. Ще один важливий крок: створення

файлу GameTypes.ts, що міститиме типи для даних під час розробки, а також дуже важливий файл globalData.ts, що міститиме глобальні налаштування гри, як ми вже розглядали це в минулих розділах. Всередині будуть реактивні дані vue, які будуть використовуватися у багатьох компонентах гри і мусять бути синхронізованими.

Рисунок 3.7 — Приклад вмісту файлу globalData.ts

Наразі, це були всі основні кроки, необхідні для налаштування проєкту, що дозволить тепер переходити нам безпосередньо до самої розробки.

3.3 Реалізація основних компонентів

3.3.1 Компонент App

Почнемо ми із основного компонента App. Його роль полягає у відображенні обгортки, що міститиме три ключові елементи: кнопку для прелоадера та запуску, фонове зображення та основний контейнер гри page, що містить меню, гру та налаштування.

Рисунок 3.8 — Всім компоненту App

Почнемо ми кнопки запуску, яка займатиме весь екран пристрою. Її мета полягає у створенні вступу, поки промальовуються зображення, а також для запуску гри та фонової музики, бо браузер по замовчуванню блокує автоматичне відтворення аудіо, тому єдиний варіант — запуск через кнопку, яку користувач точно натисне. Ми створили реактивну зміну isVisibleStartButton, що приймає булеве значення і при зміну цього стану дозволяє відобразитись іншим елементам, як видно у рисунку вище.

Також варто звернути увагу, що аудіо відтворюється лише за умови, що це дозволено, дозвіл встановлено в реактивній змінні глобального сховища, а його зміна відслідковується при методом watch.

Рисунок 3.9 — Код для запуску музичного супроводу та візуалізації

Ще момент на який потрібно звернути увагу, це функція, що присвоєна кожному компоненту, а її мета змінювати параметр реактивної змінної, що має лише три значення, для того, щоб ми могли по черзі відображати окремі компоненти через v-show атрибут.

```
function changeComponent(name: string) {
    if (name === "settings") openSettings()
    else if (name === "game") componentView.value = "game"
    else if (name === "mainMenu") componentView.value = "mainMenu"
    else return
}
```

Рисунок 3.10 — Функція зміни відображення

Загалом, мета компоненту App полягає у відображенні трьох основних компонентів: головне меню (відкрите за замовчуванням на рис. 3.11), ігрове поле та налаштування, надає способи їх перемикання, а також відображає фонову картинку та музику.

Рисунок 3.11 — Реалізований компонент App/Menu

3.3.2 Компонент Settings

Роль цього компонента важлива для налаштування певних процесів гри та особливостей. Наразі ми не маємо розширених можливостей, однак ми торкаємося дуже важливих критеріїв.

Після переходу до цього компоненту, в нас відкриється спливаюче вікно із кількома параметрами: параметр відображення звукових ефектів, голосів персонажів та фонової музики. Останній елемент дозволяє змінити мову гри, що критично важливо для користувачів, що не володіють певними мовами. Ми обмежилися лише трьома вибірковими мовами.

Рисунок 3.12 — Реалізований компонент Settings

Реалізовували ми кнопки через список ul, створивши кастомні поля типу checkbox. Завдяки атрибуту v-model ми могли напряму через імпорт взаємодіяти із глобальним параметром дозволу.

Рисунок 3.13 — Спосіб реалізації кастомного поля

Також ми одразу змінювали локальну мову для налаштувань та меню через константу tempData, що динамічно читає зміни мови. Щодо питання чи варто було зберігати дані мов у глобальних даних чи окремому JSON файлі є певні розбіжності у поглядах, проте тексту для меню та налаштувань так мало, що доцільність створення окремого файлу полягає радше в способі створення більш організованої структури, однак наразі немає надзвичайно потреби займатися цим питанням.

Рисунок 3.14 — Текст для меню та налаштувань у глобальному файлі

3.3.3 Компонент Header

Цей компонент є складовою одного великого компонента, однак наразі ми маємо розглянути його окремо.

Рисунок 3.15 — Реалізований компонент Header

Основні моменти щодо цього компоненту: він містить константи величин заповнювачів для іконок від 0 до 400. Також ми читаємо всі дані по змінам критеріїв із глобального стану та репрезентуємо їх у правильну форму через те, що вони відображаються за замовчуванням дзеркально для наших іконок.

Рисунок 3.16 — Читання критеріїв та їх перетворення

У HTML ми відтворюємо іконки та через маску та маніпуляцію через координати із шарами дозволяємо заповнюватись іконкам. Атрибут style із відповідним правилом transition задає плавність для такої зміни.

Рисунок 3.17 — Відтворення іконок

3.3.4 Компонент Footer

Мета цього компоненту відображати кількість проведених днів в облозі Трої. Ми отримуємо із глобальних даних інформацію про загальну кількість днів, а потім через прості формули перетворюємо це у кількість повних років та днів. Мета гри полягатиме у тому, щоб пожити 10 років, після чого відбудеться фінальна битва у грі. А тим часом це буде відліком часу, який показуватиме наскільки довго протягне герой.

Також ми відслідковуємо зміни мови у змінюємо мову повідомлення відповідно, а також додаємо умову для закінчень слів в українській.

Рисунок 3.18 — Перетворення днів та робота із мовою

Також компонент отримуватиме функцію через пропс, що дозволить при натисканні на цей елемент повернутися в головне меню, наприклад, щоб змінити мову гри чи інші налаштування.

Рисунок 3.19 — Реалізований компонент Footer

3.3.5 Компонент Card

Цей компонент відповідає за візуальне представлення персонажа у вигляді квадратної рамки із 2D зображенням, намальованим власноруч. Основні властивості цього елемента полягають у його інтерактивності. При затисканні фото, ми можемо рухати його вправо або вліво для свайпу, при цьому ми маємо можливість побачити як фото змінює своє забарвлення та стиль. Також з правого чи лівого боку, з залежності в яку сторону потягнути, ми побачимо як плавно з’являтиметься текст, в міру того, як далеко ми тягнемо об’єкт. Позаду основного фото із персонажем, буде спеціальна фонова ілюстрація для створення автентичного античного стилю. Така анімація та властивості реалізовані завдяки поєднанню правил SCSS та JavaScript рішень. Детальніше про це буде розглянуто у пункті 3.4. Також кожен персонаж матиме власний голос, завдяки використанню спеціального вебінструменту для генерації голосів.

Рисунок 3.20 — Реалізовний компонент Card

3.3.6 Компонент Game

Це найбільший та найважливіший компонент у грі, бо він відповідає за представлення одразу кількох інших компонентів, пов’язує та передає дані між ними, змінює сюжет та є мозком гри.

Щодо особливостей візуалізації та стилю, було реалізовано анімацію випадання пустих карток через динамічне додавання елементів завдяки vue-синтаксису для створення гарного ефекту. Також було додано відповідний звуковий супровід.

Рисунок 3.21 — Динамічне додавання анімаційних карт

А завдяки SCSS можливостям ми додали анімацію та затримку, щоб створити гарний ефект, як випадає колода.

Рисунок 3.22 — Цикл SCSS

Сам компонент містить Header, Footer та Card, а крім них текст ситуації чи події і назву героя. Щодо сюжетного механізму роботи цього компонента, то його буде розглянуто детальніше у пункті 3.6.

Рисунок 3.23 — Реалізовний компонент Game

3.4 Реалізація механіки карткового вибору

Механіка карткового вибору полягає у кількох основних кроках. Перший полягає у тому, що ми реалізовуємо анімацію свайпу. Це рішення стало можливим завдяки створенню змінних та моніторингу кліків по елементу. Для цього ми створили окремі функції: onMove визначає рух елемента після кліку, offTouch перевіряє чи не відпустили ми елемент мишкою, leaveZone перевіряє чи не залишили ми зону, за межі якої вже не можна виходити, а ResetCardAnimated повертає карту у вихідне положення, якщо ми потягнули її, але не дійшли до межі, що свайпнути. Свайп відбувається якщо ми перетнемо відповідну межу у якості пікселів, встановлену у змінній TRESHHOLD_X.

Рисунок 3.24 — Код компоненту

Також ми додаємо анімацію для відображення тексту слайду при його рухові:

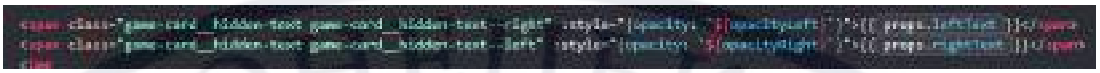

Рисунок 3.25 — Динамічне відображення тексту

Для того, щоб перевірити чи відбувся свайп, ми передаємо у пропси нашого компоненту, окрім основних даних для відображення даних, ще зовнішню функцію із булевим значенням, що відповідає false — свайп вліво, а true — в право. Дані передаються у компонент Game, де ця інформацію опрацьовується і вирішується, яка карта буде наступною.

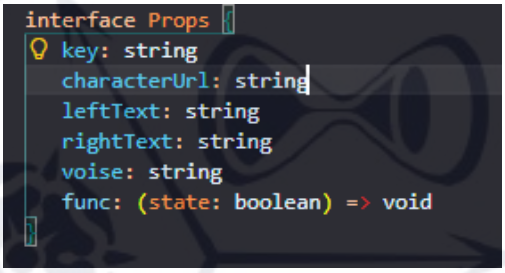

Рисунок 3.26 — Встановлення типу зовнішньої функції через TypeScript

3.5 Реалізація багатомовності

Багатомовність у застосунках дозволяє обхопити та залучити більшу кількість користувачів. У випадку нашої гри я вирішив застосувати принаймні три мови: українську, англійську та іспанську. У попередніх пунктах ми розглядали спосіб зміни мови у налаштуваннях компонента Settings, де через атрибут v-model ми передавали дані у глобальне сховище.

Щоб застосувати чи використати зміну мови у грі, потрібно створити окрему константу, що прив’язуватиметься до глобальної змінної, що містить один із трьох видів даних: «uk», «ua» чи «es». Також ми використовуємо інструмент vue — computed. Це метод для змінної чи константи, що дозвляє динамічно її перезаписати у випадку, якщо властивість з якою вона пов’язана, у нашому випадку мова, змінилася, наприклад після перемикання у налаштуваннях.

```
const lang = computed(() => globalDataForGame.language)
```

Рисунок 3.27 — Створення локальної прив’язки до глобальної мови

На прикладі із роботою щодо сюжету, ми можемо змінювати відповідні дані змінних, використовуючи інструмент watch для моніторингу зміни мови і внесення наступних змін. Саме через відповідну архітектуру сюжету у файлі JSON ми можемо легко підставити відповідний атрибут «uk», «ua» чи «es» у об’єкт, що дасть нам змогу легко отримати відповідні дані. Таким чином ми досягаємо важливої мети — динамічна зміна мови відбувається через одне загальне джерело, що дозволяє нам легко маніпулювати на змінювати атрибути мови по різних компонентах.

Рисунок 3.28 — Використання зміни мови на прикладі

3.6 Реалізація сюжетного механізму

Цей алгоритм розташований у компоненті Game і є один із найважливіших у грі, адже він перетворює JSON-сюжет у ігрову динаміку. Тут розташовано чимало функцій, необхідних для злагодженої роботи, які ми розглянемо по черзі. Однак, спочатку для кращого уявлення про те, як працюють основні функції, варто переглянути узагальнену блок-схему алгоритму.

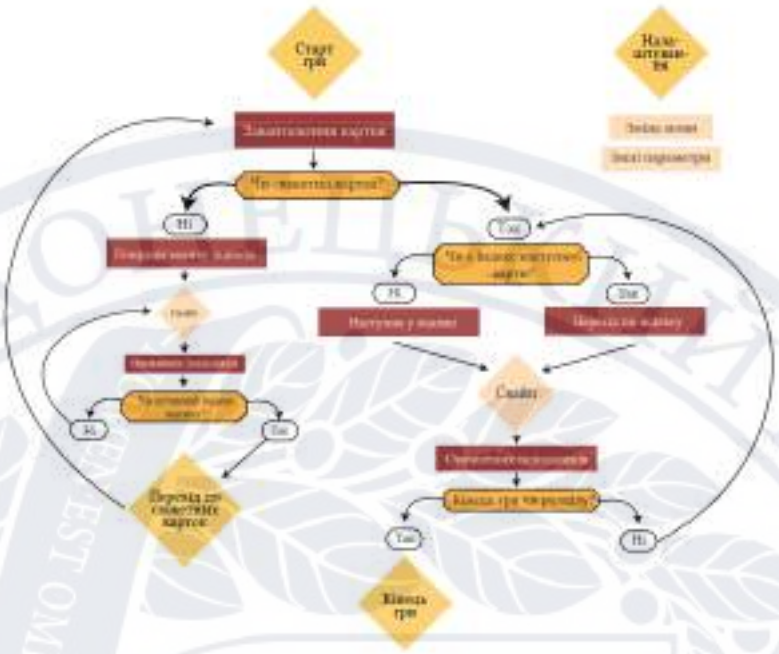

Рисунок 3.29 — Узагальнена блок-схема алгоритму

Блок завантаження даних відповідає за процес із вилучення із глобальної пам’яті двох індексів: число розділу та число картки. Це буде мінімальна реалізація пам’яті.

```
const gameProgressChapter = computed(() => globalDataForGame.progress[0])
const gameProgressCard = computed(() => globalDataForGame.progress[1])
```

Рисунок 3.30 — Завантаження прогресу гри

Одразу після цього ми завантажуємо необхідні дані у константи для сюжетних карт, рутинних та початкових.

Рисунок 3.31 — Завантаження сюжетних карт

Потім ми маємо додати дані із картки на полотно. Для цього спочатку ми створюємо відповідну реактивну константу dataToInput, що міститиме основні поля даних: id, characterUrl, leftText, rightText, title, voise, characterName, isNextLeft, isNextRight — які братимуться із поточної карти за індексом, тобто стартової.

Ці важливі дані вставляються у template компоненту та пропси Card, відображаючи контент.

Рисунок 3.32 — Структура template Game

Потім відбувається свайп, після якого ми перевіряємо тип наступної карти: чи це сюжетна карта чи рутинна. Для останньої у нас на початку формується масив індексів, тому ми рухаємося саме ним. Якщо в процесі ми натрапимо на унікальний індекс або правило, яке змішує нас обрати іншу карту замість індексу, ми це враховуємо та змінюємо алгоритм. Після цього ми збільшуємо лічильник прогресу та перевіряємо чи елемент останній у масиві. Якщо так, то ми переходимо до сюжетних карток, інакше продовжуємо цикл.

У сюжетних картках ситуація інша і ми переходимо не по індексах попередньо сформованого масиву, а по внутрішніх індексах JSON-полів. В кінці ми завжди перевіряємо чи елемент є завершальним, бо якщо це так, то ми переходимо до нового розділу, в іншому випадку продовжуємо.

Тепер розглянемо список основних функцій у коді та за що вони відповідають. Спочатку розглянемо допоміжні функції.

Таблиця 3.1. Допоміжні функції компоненту Game

Функція	Призначення
buildDataToInput(card, language)	Перетворює будь-яку картку у єдиний формат даних для відображення в

	компоненті Card, як конструктор із відповідними полями для dataToInput.
resolveNextDirection(choices)	Визначає, яка сторона картки (left/right) веде до примусової наступної картки.
getNextCardId(choices, direction)	Отримує ID наступної картки залежно від вибору гравця.
resolveUniqueIndex(index, cardPool)	Уникає повторного показу унікальних карток, замінюючи індекс випадковим.
checkScales(scales, max, min)	Перевіряє, чи досягнуто критичних меж репутації, богів або слави.

Тепер розглянемо основні функції, наведені у таблиці 3.2 нижче.

Таблиця 3.2. Основні функції компоненту Game

Функція	Призначення
loadNextRoutineCard(sequenceIndex, swipedRight)	Завантажує наступну рутинну картку з урахуванням спеціальних переходів та унікальних карток.
updatePendingFromCard(index)	Зчитує з картки можливий примусовий наступний перехід і зберігає його.
applyImpact(cardIndex, swipedRight)	Застосовує вплив вибору гравця на глобальні шкали характеристик.
loadEndCard(trigger)	Завантажує кінцеву картку при критичному перевищенні або падінні шкал.
loadStoryCard(storyCards, index, swipedRight)	Завантажує сюжетну картку та оновлює прогрес історії.

Є ще одна із найважливіших функцій — catchCardSwipe(swipedRight). Це основний обробник свайпу, він керує логікою переходів між стартом, рутинними картами та кінцівками. Саме цю функцію ми передаємо в пропс у компонент Card, де повертаємо відповідне значення свайпу.

Загалом цей компонент має чимало важливих деталей, які не було враховано через об’єм, тому у цій роботі було виділено лише основні функції та їх призначення. Як висновок, компонент Game реалізує управління ігровими процесами, пов’язує чимало компонентів із JSON-сюжетом, реалізує систему вибору, переходи між главами та типами карток, реалізує багатомовність та візуальне відображення.

3.7 Тестування

3.7.1 Функціональне тестування

У цьому пункті нам варто впевнитись у тому чи працюють основні функції коректно, а якщо є помилки, то вжити заходів для їх виправлення. Щоб зробити це, ми маємо спочатку запустити нашу гру і почати цілеспрямовано запускати основні функції через вебінтерфейс, паралельно перевіряючи процес у консолі браузера.

Перший тест стосується завантаження картинки та фонової музики. Зображення та компоненти з’являються, як і фонова музика після кліку по основному зображенню. Доказом цьому є відповідна іконка вкладці проєкту у браузері. Таким чином ми обійшли заборону браузера відтворювати автоматично аудіо.

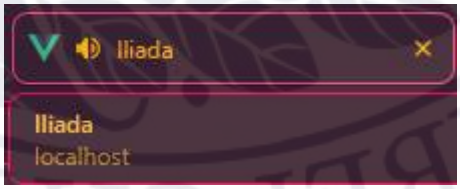

Рисунок 3.33 — Відтворення аудіо у браузері

Відповідно кнопки навігації головного меню працюють правильно, не даючи похибок. Це також стосується Footer, що дозволяє нам повертатись назад при кліку.

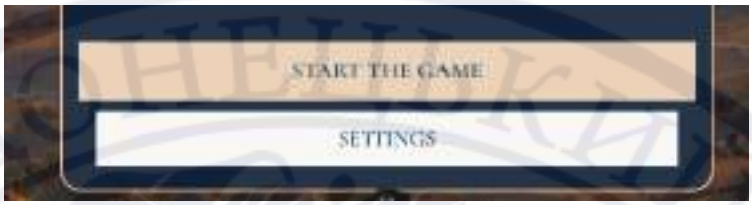

Рисунок 3.34 — Робота із головним меню

Механізм роботи карток також не дає проблем: ми отримуємо правильні дані із пропсів та їх візуалізацію, також можемо взаємодіяти та свайпати зображення. При цьому ми бачимо як плавно з’являється текст рішень, а картка зникає лише у випадку, якщо ми перетнемо встановлену у коді межу, інакше вона просто та плавно повертається на місце.

Рисунок 3.35 — Робота компоненту Card

Показники індикаторів також працюють відповідно добре, змінюючись відповідно до рішень. Анімація дозволяє виглядати цим переходам плавніше. На рисунку нижче видно легкий дисбаланс, адже у нас із шкалою у 400 балів, максимальне коливання від -10 до 10, тому деякі зміни візуально помітити на рис. 3.36 важко, однак вони є, а щоб змінити рівень впливу потрібно змінити це вручну у JSON файлі сюжету.

Рисунок 3.36 — Рівень показників гри

Дані по днях у Footer працюють коректно. Дані про це надходять із Card, що генерує певне число (**Math.floor(Math.random() * 30) + 1**) та додає до глобальних даних, а Footer читає їх та трансформує. Збільшення днів відбувається лише у випадку якщо це випадкові карти, у сюжетних дні зупиняються, а в наступному розділі продовжують роботу.

Рисунок 3.36 — Зміна днів у Footer

Важлива функція налаштувань працює дуже добре: ми можемо вмикати та вимикати фонову музику, голоси персонажів та змінювати мову інтерфейсу. Наразі критичних чи незначних помилок у роботі не було.

Загалом можна зробити висновок, що основні функції працюють злагоджено та добре, хоч є певні незначні нюанси, які потрібно потім пропрацювати.

3.7.2 Сценарне тестування

Ми протестували найважливішу частину гри шляхом проходження її сюжету. В результаті було виявлено кілька помилок із роботою індексів та переходами розділів, проте після невеликого рефакторингу ми змогли уникнути критичних помилок.

Було трохи змінено структуру JSON для сюжетних карт. Завдяки маніпуляціями певними полями ми тепер можемо змінювати хід сюжету через рішення гравця. Додавання endChapter закінчує розділ і переходить у інший, однак водночас інше рішення може продовжити гру.

Рисунок 3.37 — Маніпуляція правилами типу «endChapter»

Загалом алгоритм працює дуже добре, особливо для випадкових карт, де від гравця вимагалося прийняти рішення, що зміщувало сюжет до особливих карт. Хоч у нас був замалий сюжет, щоб протестувати набагато більшу гілку, проте позитивні результати у цьому масштабі свідчать, що збільшення сюжету ймовірно не вплине на хорошу роботу.

3.7.3 UX/UI тестування

Один із найважливіших кроків у тестуванні — дослідити UX/UI, де UI (User Interface) — визначає користувацький інтерфейс та його вигляд: дизайн, кольори, анімація, контент тощо [40].

У нашій роботі ми розробили мінімалістичний дизайн, що був дуже інтуїтивно зрозуміли та простий, не мав надлишку деталей чи непотрібних елементів. Кольорова гамма та звуковий супровід створювали унікальну античну атмосферу для глибшого занурення в гру. Іконки та спосіб управляння також були надзвичайно простими та черпали натхнення у іграх такого жанру. Налаштування дуже прості і змінюють лише основні параметри гри: звуки та мову.

UX (User Experience) — користувацький досвід, тобто наскільки зручно та легко буде людину користуватися інтерфейсом [40].

Для досягнення високого показника ми працювали у кількох напрямках. Мінімалізм та інтуїтивна зрозумілість із UI вже дають чимало переваг: ми не потребуємо забагато інформації, кліків чи рухів, щоб грати, а це важливо, бо

«один клік кращий за два». Наступний підхід — це адаптація під різні розширення екранів. Для досягнення цього ми використовували медіа-запити, інструмент CSS.

Рисунок 3.38 — Медіа-запит для класу

3.7.4 Кросбраузерне тестування

Кросбраузерне тестування є важливим етапом розробки вебзастосунків, що передбачає перевірку коректності роботи програмного продукту в різних браузерах, операційних системах та на різних типах пристроїв. Його основною метою є забезпечення однакового відображення інтерфейсу, стабільності функціоналу та збереження позитивного користувацького досвіду незалежно від програмного середовища користувача. Оскільки сучасні браузери, такі як Google Chrome, Mozilla Firefox, Microsoft Edge, Safari чи Opera, використовують різні рушії обробки HTML, CSS і JavaScript, навіть якісно написаний код може поводитися по-різному в кожному з них (наприклад консоль чи Lighthouse Chrome) [41].

Для того, щоб провести це тестування, необхідно відкрити та провести принаймні 3 попередні тести у новому браузері. Наша основна платформа була Mozilla Firefox, а тепер нам потрібно переглянути вебгру у двох інших: Chrome та Microsoft Edge.

Перший етап полягає у ручному тестуванні основного функціоналу. Я відкрив застосунок у кожному браузері та перевірив коректність запуску, адаптивність інтерфейсу, правильність відображення стилів, функціонування

кнопок, механіку свайпів, анімації, звуковий супровід, зміну мов тощо. Особливу увагу приділив можливим візуальним змінам, таким як відступи, масштабування елементів, неправильне відображення шрифтів або збої в JavaScript-логіці.

Другий етап передбачав використання інструментів розробника (DevTools), які дозволяють емулятувати різні розміри екранів, мобільні пристрої, змінювати продуктивність мережі та перевіряти помилки консолі. Це допомагає швидко виявити технічні проблеми без необхідності використання великої кількості фізичних пристроїв.

Рисунок 3.39 — Тест у браузері Microsoft Edge

В загальному підсумку гра працювала коректно у всіх браузерах, стилі залишалися однаковими завдяки scss, який задає пресікси; також адаптація під різні пристрої не ламала інтерфейсу, а в DevTools панелі та консолі ми не знайшли критичних помилок.

3.7.5 Продуктивність

На продуктивність вебгри впливають чимало чинників, деякі із них пов’язанні із зовнішніми факторами, наприклад, домени чи швидкість інтернету користувача. Наша задача — добити максимальної оптимізації гри, щоб навіть при найбільших навантаженнях чи поганому інтернеті вона могла бути працездатною [42].

Перший крок полягав у використанні семантики — добиранні максимально влучних тегів HTML для контенту. Такий підхід дозволяє браузеру ефективніше обробляти структуру сторінки, покращує швидкість рендерингу та спрощує

підтримку коду. Семантична розмітка також позитивно впливає на доступність застосунку, полегшуючи взаємодію з допоміжними технологіями, а також сприяє кращій SEO-оптимізації вебпроєкту.

Другий крок полягав у стисненні контенту, тобто зображень у менші розмір та формати для економії ресурсів.

Наступним важливим етапом стала оптимізація структури ресурсів та способів їх завантаження. Для цього статичні медіа-файли були організовані у відповідних директоріях із прямим доступом через public, великі файли формату JPG були конвертовані у набагато легші WEBP-фото, що дозволило зменшити навантаження на систему збірки та прискорити отримання контенту браузером.

Ще одним важливим кроком є використання кешування браузера. Для цього було використано Service Worker, що дозволяє обрані кешувати дані, в нашому випадку це були зображення [43].

Це дозволяє мінімізувати повторне завантаження ресурсів, зберігати прогрес гри без звернення до сервера та забезпечувати стабільну роботу навіть за умов нестабільного інтернет-з’єднання.

Таким чином, поєднання семантичної структури, оптимізації медіа-контенту, ефективного керування ресурсами та локального збереження значно підвищує продуктивність вебгри, покращує користувацький досвід і забезпечує її доступність для ширшої аудиторії.

3.8 Аналіз результатів роботи

Отже, проаналізуємо результати, що ми отримали. Усі, без винятку, компоненти були реалізовані разом із їх основним функціоналом та працюють злагоджено, як це було перевірено у тестуванні. Однак через брак часу не було розроблено кілька додаткових функцій.

Перша, це LocalStorage. Це корисний інструмент для кешування даних у браузері, що не вимагає великих ресурсів, однак і не може зберігати чимало даних. Однак наша гра не вимагає великих ресурсів для збереження даних. Як обхідний шлях та у майбутньому, ми можемо функціонал для створення профіля

користувача із унікальним ідентифікатором. Головна мета такого підходу — імпортувати дані в різні браузери без централізованого сховища. Для цього потрібно налаштувати систему для створення ключа, пароля чи ідентифікатора користувача, що сформує дуже довгий хеш та ключ, який потім можна розшифрувати на певному пристрої чи браузері. Такий підхід може бути не дуже легким в реалізації, однак єдина його перевага полягає у тому, що нам не потрібно створювати авторизації, JWT чи окремої бази даних. Однак і мінусів не бракує: у нас не буде синхронізації та видалення усіх даних на всіх пристроях водночас приведе нас у безвихідну ситуацію без можливості відновити прогрес.

Ще ми не реалізували вимкнення та увімкнення звуків у налаштуваннях, що відповідають за відтворення коротких звуків після взаємодії із кнопками чи картками. Це не складне в реалізації завдання просто було залишене через потребу у розробці важливіших елементів.

Були також моменти із правилом SCSS overflow для game, де ми хотіли створити ефект виходу карток за межі контейнера, однак через конфлікти SCSS поки зупинилися на обрізанні картинки. Це не впливає на ігровий процес, але можливо трохи негарно з точки зору дизайну, тому це питання обов’язково потрібно розглянути у майбутньому.

Щодо самого коду, хоч він і працює добре, однак потребує глибокого рефакторингу та оптимізації, адже при розробці гри ігнорувались певні невеликі відхилення від основних шаблонів програмування [42].

TypeScript допомагав виявляти основні помилки синтаксису, однак не перевіряв грамотності написаного коду, що вимагає у майбутньому глибокого аналізу та перетворення коду у зрозумілі та читабельні алгоритми не лише для нас самих, а і для інших програмістів.

ВИСНОВКИ

У процесі виконання бакалаврської роботи було систематизовано основні підходи до створення сюжетних інді-ігор, проаналізовано принципи побудови нелінійного сюжету, механіки карткового вибору, а також досліджено сучасні вебтехнології, що використовуються для розробки інтерактивних ігрових продуктів.

Було досліджено особливості епосу Гомера «Іліада» як сюжетної основи для створення цифрової адаптації, визначено специфіку трансформації літературного твору у формат інтерактивної гри та виявлено можливості поєднання культурно-історичного змісту з сучасними ігровими механіками.

Обґрунтовано вибір програмного забезпечення та інструментів розробки, зокрема фреймворку Vue для побудови клієнтської частини застосунку, TypeScript для підвищення надійності коду, JSON для організації сюжетної структури, а також додаткових графічних інструментів для створення дизайну інтерфейсу та візуального оформлення гри.

Було спроєктовано архітектуру програмного забезпечення, структуру сюжетних розгалужень, механіку карткового вибору, систему багатомовності та інтерфейс користувача, що забезпечило логічну організацію проєкту, масштабованість та зручність подальшого розвитку гри.

У ході практичної реалізації розроблено основні компоненти вебгри, створено систему інтерактивної взаємодії користувача із сюжетними подіями, реалізовано динамічну зміну ігрових сцен, механізм обробки виборів гравця, систему прогресу, багатомовну підтримку та адаптивний дизайн для коректного використання на різних пристроях.

Було підготовлено візуальне оформлення гри, яке включає дизайн карток, інтерфейсні елементи, стилізацію під тематику твору та загальну художню концепцію, що покращує сприйняття користувачем ігрового процесу.

Проведено тестування функціональних можливостей програмного продукту, під час якого встановлено стабільність роботи основних систем,

коректність реалізації сюжетної логіки, механік вибору та багатомовності, а також відповідність гри визначеним вимогам.

Отримані результати підтверджують ефективність використання сучасних вебтехнологій для створення сюжетно-орієнтованих інтерактивних ігор, а розроблений програмний продукт може слугувати основою для подальшого розвитку подібних освітньо-розважальних проєктів, присвячених адаптації літературної спадщини у цифровий формат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Frontiers. Editorial: Video games for impact: design projects that can change the way we think. URL: https://surl.li/gxrkcy (дата звернення 11.03.2026)

2. StudioBinder. What is a Non-Linear Plot — How to Write Stories Out of Order. URL: https://surl.li/hdhsgt (дата звернення 11.03.2026)

3. Google Patents. Cathode-ray tube amusement device. URL: https://surl.li/eojnxy (дата звернення 14.03.2026)

4. EBSCO. Computer Networks. URL: https://surl.li/unnoku (дата звернення 18.03.2026)

5. EBSCO. History of Video Games. URL: https://surl.li/bqsfoe (дата звернення 18.03.2026)

6. Medium. Tarantino vs. Nolan: Mastering the Non-Linear Narrative. URL: https://surl.li/irtrut (дата звернення 21.03.2026)

7. Medium. Linear vs Non-Linear: The true Critical Framework of Gaming. URL: https://surl.lu/ddmhlp (дата звернення 28.03.2026)

8. Homer. The Iliad. Translated by Robert Fagles. Penguin Classics, 1998.

9. Retrospect Journal. Why does Homer's Iliad still resonate? URL: https://surl.lu/kubjlz (дата звернення 04.04.2026)

10. Medium. Tinder card swipe — Figma Prototyping. URL: https://medium.com/@FullStackDesigner/tinder-card-swipe-figma-prototyping-9c9e78fff869 (дата звернення: 06.04.2026)

11. Aarseth E. Cybertext: Perspectives on Ergodic Literature. Baltimore: Johns Hopkins University Press, 1997.

12. Springer Nature Link. Alternate realities in interactive digital narratives – understanding and improving design and prosocial effects through empirical methods. URL: https://surl.li/mbwcqk (дата звернення 07.04.2026)

13. MDN Web Docs. HTML5 Reference. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 10.04.2026)

14. MDN Web Docs. Canvas API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API (дата звернення: 10.04.2026)

15. MDN Web Docs. CSS Reference. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 10.04.2026)

16. Sass Official Documentation. URL: https://sass-lang.com/documentation/ (дата звернення: 10.04.2026)

17. MDN Web Docs. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 10.04.2026)

18. Microsoft. TypeScript Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 10.04.2026)

19. Medium. My Take on the Big Three: React, Angular, and Vue. URL: https://medium.com/@daveberning/my-take-on-the-big-three-react-angular-and-vue-c82f57fef2ea (дата звернення: 11.04.2026)

20. Phaser Official Documentation. URL: https://docs.phaser.io/ (дата звернення: 17.05.2026)

21. MDN Web Docs. Web Storage API (localStorage). URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API (дата звернення: 17.04.2026)

22. Meshy. Що таке інді-гра? Огляд 2025 року. URL: https://www.meshy.ai/uk/blog/what-is-an-indie-game (дата звернення: 23.04.2026)

23. Nerial. Officail Web site. URL: https://www.nerial.co.uk/ (дата звернення: 22.04.2026)

24. PocketGamer. Reigns review - A strategy game mixed with Tinder. URL: https://surl.li/mkenpx (дата звернення 28.04.2026)

25. Vue.js Official Documentation. URL: https://vuejs.org/guide/introduction.html (дата звернення: 01.05.2026)

26. ElevenLabs. Офіційний вебресурс. URL: https://elevenlabs.io/ (дата звернення: 03.05.2026)

27. ORACLE. What Is JSON?. URL: https://www.oracle.com/database/what-is-json/ (дата звернення 07.05.2026)

28. JsonHero. Офіційний вебресурс JsonHero. URL: https://jsonhero.io/ (дата звернення 07.05.2026)

29. Meegle. Branching Storylines. https://www.meegle.com/en_us/topics/game-design/branching-storylines (дата звернення 10.05.2026)

30. Asabix. Що таке Single Page Application? URL: https://asabix.com.ua/what-is-a-single-page-application/ (дата звернення 13.05.2026)

31. Bass L., Clements P., Kazman R. Software Architecture in Practice. Boston: Addison-Wesley, 2012.

32. Norman D. The Design of Everyday Things. Basic Books, 2013.

34. Nielsen J. Usability Heuristics for User Interface Design. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 10.05.2026)

35. Ixdf. Color Theory. URL: https://ixdf.org/literature/topics/color-theory (дата звернення: 11.05.2026)

36. Digital Residency.Game Sound Design: The Art of Video Game Sound. URL: https://digitalresidency.com/game-sound-design-the-art-of-video-game-sound/ (дата звернення: 13.05.2026)

37. Top-Keis. Що таке Visual Studio Code. URL: https://top-keis.com.ua/shho-take-visual-studio-code/ (дата звернення 14.05.2026)

38. Vite Official Guide. URL: https://vitejs.dev/guide/ (дата звернення: 17.05.2026)

39. Medium. Vite: The Build Tool for Modern Web Development. URL: https://medium.com/@emre.deniz/vite-the-build-tool-for-modern-web-development-0e99906f2f0c (дата звернення 15.05.2026)

40. Prjct. Що таке UI/UX дизайн: ваш гід професією. URL: https://surl.li/qeqmrj (дата звернення 13.05.2026)

41. Google Developers. Lighthouse Performance Audits. URL: https://developer.chrome.com/docs/lighthouse/ (дата звернення: 16.05.2026)

42. Google Web.dev. Performance Optimization. URL: https://web.dev/performance/ (дата звернення: 17.05.2026)

42. Scalastic. Principles of Software Development: SOLID, DRY, KISS, and more. URL: https://scalastic.io/en/solid-dry-kiss/ (дата звернення 17.05.2026)

43. MDN Web Docs. Service Worker API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API (дата звернення 18.05.2026)