**Пояснювальна записка  
до курсової роботи**

на тему: Веб-гра «Сапер» з можливістю авторизації та перегляду результатів ігор

Київ – 2021 року

**ЗМІСТ**

[**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**](#_30j0zll) **9**

[**ВСТУП**](#_1fob9te) **10**

[**1**](#_3znysh7) **АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 11

[1.1](#_2et92p0) Загальні положення 11

[1.2](#_tyjcwt) Змістовний опис і аналіз предметної області 12

[1.3](#_qsh70q) Аналіз успішних IT-проектів 13

[1.3.1 Аналіз відомих технічних рішень 1](#_3as4poj)3

[1.3.2 Аналіз відомих програмних продуктів 1](#_3dy6vkm)7

[1.4](#_1t3h5sf) Аналіз вимог до програмного забезпечення 20

[1.4.1 Розроблення функціональних вимог 2](#_4d34og8)0

[1.4.2 Розроблення](#_2s8eyo1) нефункціональних вимог 26

[1.4.3 Постановка задачі 2](#_17dp8vu)7

[1.5](#_3rdcrjn) Висновки по розділу 28

[**2**](#_26in1rg) **МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 29

[2.1](#_lnxbz9) Моделювання та аналіз програмного забезпечення 29

[2.2](#_35nkun2) Архітектура програмного забезпечення 30

[2.3](#_1ksv4uv) Конструювання програмного забезпечення 30

[2.4](#_44sinio) Висновки по розділу 32

[**3**](#_2jxsxqh) **АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 33

[3.1](#_z337ya) Аналіз якості ПЗ 33

[3.2](#_3j2qqm3) Опис процесів тестування 33

[3.3](#_1y810tw) Опис контрольного прикладу 37

[**4**](#_4i7ojhp) **ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 39

[4.1](#_2xcytpi) Розгортання програмного забезпечення 39

[4.2](#_1ci93xb) Висновок до розділу 41

[**ВИСНОВКИ**](#_3whwml4) **42**

[**ПЕРЕЛІК ПОСИЛАНЬ**](#_2bn6wsx) **44**

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

WEB – Всесвітня павутина (Word Wide Web) — розподілена система, що надає доступ до пов'язаних між собою документів, розташованих на різних комп'ютерах, підключених до мережі Інтернет. Для позначення Всесвітньої павутини також використовують слово WEB

OS/2 – стара операційна система від IBM, у створенні якої брала участь Microsoft

БД – база даних

ОС – операційна система

СКБД – система керування базою даних

ПК – Персональний комп’ютер

ПЗ – Програмне забезпечення

DOM – Об'єктна модель документа (Document Object Model) – це програмний інтерфейс (API) для HTML і XML документів. DOM надає структуроване представлення документа та визначає те, як ця структура може бути доступна з програм, які можуть змінювати вміст, стиль та структуру документа. Подання DOM складається з структурованої групи вузлів та об'єктів, які мають властивості та методи. Фактично, DOM з'єднує веб-сторінку з мовами опису сценаріїв чи мовами програмування.

ORM (англ. Object-relational mapping, Об'єктно-реляційна проекція) – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».

Open source (або open-source software) – програмне забезпечення з відкритим початковим кодом.

Фреймворк (англ. Framework, каркас, платформа, структура, інфраструктура) – інфраструктура програмних рішень, що полегшує розробку складних систем.

**ВСТУП**

Комп'ютерні ігри – це складні програми, які служать людині навчання чи розваги і розраховані різні вікові групи. І не секрет, що пік іграшкового виробництва припав саме на пік технологічного прогресу середини ХХ століття. У цей час разом з винаходом комп'ютерів з'явилися й комп'ютерні ігри. Історія появи комп'ютерних ігор розпочинається з 1940 року. Тривалий час ці ігри були непопулярні з однієї простої причини – люди просто не мали комп'ютерів вдома. Вже з 1970-х років починають з'являтися аркадні автомати, ігрові консолі та домашні персональні комп'ютери.

Гра – неодмінний супутник розвитку людства. На стадії гри виконуються надзвичайно важливі функції. Вони використовувалися для соціалізації підростаючого покоління, підготовки до колективного полювання, для тренування. Але навчально-тренувальні функції були не головними у давній ігровій діяльності. Головне поле гри.

Сапер - це оманлива проста гра для розвитку та логіки, яка стала однією з найпопулярніших ігор. Мета гри: знайти порожні комірки, але не чіпати комірки, що містять міни.

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальні положення

Сапер бере свій початок з найперших мейнфреймових ігор 1960-х і 1970-х років. Які є предками без програмної реалізації. Найпершою ластівкою стала гра Cube від розробника Джерімака Ретліффа, але справжнім прабатьком Сапера саме в тому вигляді, в якому ми знаємо стала гра Mined-Out від розробника Яна Ендрю, випущеною на персональний комп'ютер Sinclair ZX81 у 1983 році та написаною мовою BASIC .

Суперечки про те, чи скопійований Сапер з Mined-Out йшли досить довго. Один з розробників Сапера Курт Джонсон, програміст-самоучка, що перейшов у Microsoft на початку 90-х, стверджує, що це не так і надихався він зовсім іншою грою для іншої платформи (Macintosh) з чорно-білими зображеннями для відображення всіх осередків. . Щоправда, як відомо, десь посередині.

Сам Сапер з'явився на світ, як і багато інших геніальних речей у нашому світі, практично випадково. Джонсон зовсім не був розробником ігор, він кодив відладчик для OS/2, а у вільний час колупав графічні інтерфейси. Це й у пригоді для релізу першої версії Сапера, яка розповзлася по мережі через Фідонет та подібні борди.

Далі більше. Сапер був випущений у наборі Windows Entertainment Pack - збірник ігор, що випускається з 1990 по 1992 рік

А потім успішно емігрував у першу серйозну ОС від Microsoft - Windows 3.11 (серпень 1993), в якій з'явився звичний віконний інтерфейс. У такому вигляді гра проіснувала аж до Windows Vista, коли під тиском протестів було трохи змінено концепцію гри. Але справжні фанати завжди пам'ятатимуть гру за її олдскульний інтерфейс.

## Змістовний опис і аналіз предметної області

WEB додаток, за допомогою якого можна дізнатися правила гри Сапер, зареєструватися та подивитись рейтинг ігор всіх користувачів та прагнути до кращого результату. У адміна є можливість видалити результат гри

Сапер є логічною грою, основною метою якої є знаходження всіх захованих бомб на мінному полі. Ваше завдання відкрити всі осередки поля, які не містять бомб, заблокувавши (позначивши) при цьому осередки, в яких розташовані бомби прапорцем.

Поле гри задано як двомірного масиву. У цьому масиві розташовані осередки. Спочатку всі вони закриті. Осередки можуть бути порожні, з цифрами та бомбами. Для кожного такого осередку ми ставимо клас. При натисканні лівої клавіші миші відкривається комірка, при натисканні правої кнопки виставляється прапорець, при цьому лівою кнопкою миші ви вже не можете натиснути на заблоковану комірку, але правою кнопкою можна зняти позначку прапорця. Осередок із цифрою позначає те, скільки мін знаходиться в окрузі цього осередку. При першому натисканні кнопки миші на комірку не може відкритися комірка з бомбою.

Гра вважається програною, якщо ви відкрили комірку з бомбою.

Гра вважається виграною, якщо на ігровому полі всі осередки з бомбами відзначені та решта осередків – відкриті.

Можливість використати підказку, при натисканні на лампочку відкриється комірка в якій немає міни та яка знаходиться біля вже відкритої комірки, якщо немає відкритої комірки, тоді нічого не відбувається.

Можливість почати гру спочатку, або обрати рівень складності гри, від рівня складності залежить розмір поля, а також кількість мін.

## Аналіз успішних ІТ-проектів

### Аналіз відомих технічних рішень

Для технічного рішення я вибрав мову програмування Javascript. Оскільки будучи найпопулярнішою мовою програмування, JavaScript також є однією з найуніверсальніших технологій розробки ПЗ, дозволяючи створювати якісні програми для багатьох платформ.

Але на цій галузі застосування JS не закінчуються. Останнім часом ця мова часто розглядається як головний претендент для програмування на стороні сервера. І позначило це зрушення в індустрії веб-розробки поява Node.js.

Серверна частина курсової роботи написана на мові програмування Node.js. Насправді Node.js - це не фреймворк або бібліотека, а середовище виконання, яке базується на движку Chrome V8 JavaScript.

За даними Google Trends, інтерес до цього відгалуження JS досяг піку в 2017 році і залишається високим. І на те є вагомі причини.

Використовуючи Node.js як основу для бекенда, ви автоматично отримуєте всі переваги повнофункціональної розробки на JavaScript, серед яких:

* оптимальна швидкість роботи додатків;
* можливість спільного та повторного використання;
* безліч безкоштовних інструментів;
* кросплатформність;

Node.js працює швидко. Це підтверджують результати тестів продуктивності, представлені [[toptal.com](https://www.toptal.com/nodejs/nodejs-guide-integration-tests)]. Дослідники порівняли, як популярні мови, такі як GO, PHP, Java та Node.js, обробляють одночасні запити.

Ще однією важливою перевагою є асинхронна обробка запитів. У контексті серверної частини синхронна обробка передбачає, що код виконується послідовно. Таким чином, кожен новий запит блокує потік інших, а інші команди почнуть виконуватись лише після того, як буде виконано попередню. У свою чергу, асинхронний метод, який використовується в Node.js, максимально використовуючи однопотокову обробку, що скорочує час відгуку в кілька разів.

Приклади говорять самі за себе: багато провідних компаній після переходу на Node.js помітили значні покращення — PayPal, наприклад, повідомив про зниження часу відгуку на 35% після міграції з Java.

Багата екосистема. NPM - менеджер пакетів Node.js за замовчуванням - також є основною платформою для інструментів JavaScript з відкритим вихідним кодом, які відіграють важливу роль у розвитку цієї мови програмування. З урахуванням того, що на даний момент у реєстрі npm доступно близько мільйона бібліотек і щотижня публікується понад 10 000 нових, екосистема Node.js досить багата. За даними статистики, 97% сучасних веб-додатків складаються з модулів npm. І це є доказом його незаперечної популярності серед розробників.

Порівняння Node.js з Ruby on Rails та Django. Існує не так багато технологій, які могли б конкурувати з Node.js за популярністю. Однак, якщо поглянути на існуючі альтернативи у сфері серверного програмування, можна побачити ще як мінімум два «зіркові» кандидати. Це Ruby on Rails та Django.

Ruby on Rails відомий простотою і наявністю власної екосистеми пакетів користувача Rails. Він успішно використовується такими брендами як GitHub, Twitter і Airbnb, що доводить його універсальність.

Хоча Rails не може конкурувати з продуктивністю та масштабованістю Node, ця мова може бути найкращим вибором для швидкої розробки.

Ще один популярний веб-фреймворк, який може використовуватися для програмування бекенд - Django. Позиціонований як інструмент для перфекціоністів, він створений, щоб робити програми максимально швидкими, структурованими, безпечними та легкими для розуміння. У порівнянні з Node.js, однак інших вагомих переваг він не має. Швидше, це залежить від особистих переваг розробника та досвіду роботи з JavaScript чи Python.

В якості основи для створення адаптивного web-додатку був взятий клієнтський фреймворк React.js

React.js вперше представлена ​​у 2013 році, використовується для створення динамічних веб та мобільних додатків. React.js передбачає створення незалежних компонентів, що повторно використовуються, що значно підвищує продуктивність додатків. Ця бібліотека використовує концепцію віртуального DOM – імітаційного способу представлення структурного документа за допомогою об'єктів.

Віртуальний DOM дозволяє імітувати обробку на стороні сервера в браузері користувача без необхідності перезавантаження сторінки для оновлення інформації. Це дозволяє створювати по-справжньому динамічні, інтерактивні веб-застосунки: будь-який елемент інтерфейсу можна оновити, не торкаючись і не перезавантажуючи решту сторінки.

Бібліотека React.js включає компоненти, відповідальні за спілкування в соціальних мережах, інтерфейс користувача, управління станом програми та інші складні завдання. Модульний підхід у React.js спрощує проектування та надає розробникам набори різноманітних готових компонентів.

Компактність. React.js доступна у компактній редакції Minimal React. Цей невеликий пакет не вимагає часу та зусиль для налаштування. Він поставляється із зручною опцією поділу коду, яка допомагає скоротити час завантаження веб-сайту, оскільки ця функція запобігає одночасному рендерингу всіх компонентів.

Гнучкість та розвинена екосистема. React.js поставляється з безліччю компонентів, інтегрується з іншими платформами – все це дозволяє розробникам створювати складні веб-програми для будь-яких цілей. Цим пояснюється величезна популярність бібліотеки React.js.

Зворотня сумісність. Найбільша перевага використання React.js полягає в тому, що нові версії бібліотеки повністю сумісні з попередніми випусками. Старі бібліотеки не стають марними після виходу оновлень.

В якості бази даних була обрана СКБД PostgreSQL

Відкритий вихідний код знаходиться у вільному доступі за ліцензією на відкритий вихідний код. Це дає вам свободу використовувати, змінювати та впроваджувати його відповідно до потреб вашого бізнесу.

Зниження витрат Як продукт з відкритим вихідним кодом, PostgreSQL нічого не вартий - жодних ліцензійних зборів! Це абсолютно безкоштовно! Вам більше не потрібно турбуватися про витрати на ліцензування, проблеми з контрактами та роботу з високобюджетним продуктом. Він доступний вам, коли ви цього хочете і так, як ви цього хочете.

Націленість на співтовариство – надійність Декілька компаній та приватних осіб роблять свій внесок у проект і впроваджують інновації вже понад 25 років. Сильне співтовариство гарантує, що помилки будуть виправлені без зволікання.

PG підтримується широким спектром розширень, а також безліччю моделей даних SQL та NoSQL.

Безпека Існує безліч функцій для підвищення безпеки завдяки легкій розширюваності: однак, якщо ви використовуєте правильні (TDE, Data Masking), ви отримуєте дуже безпечну базу даних, яка піклується про ваш найцінніший актив - ваші дані.

Масштабованість Ваша база даних PostgreSQL може зростати разом з вами. Існує кілька технічних варіантів масштабної роботи з PostgreSQL. Ось чому ваша база даних PostgreSQL може зростати разом з вами і може бути настільки великою, як вам потрібно.

### Аналіз відомих програмних продуктів

На момент написання роботи, були знайдені аналоги

* онлайн Сапер від Google активний конкурент;

Головну сторінку гри Сапер від Google можна побачити на рисунку. 1.1.

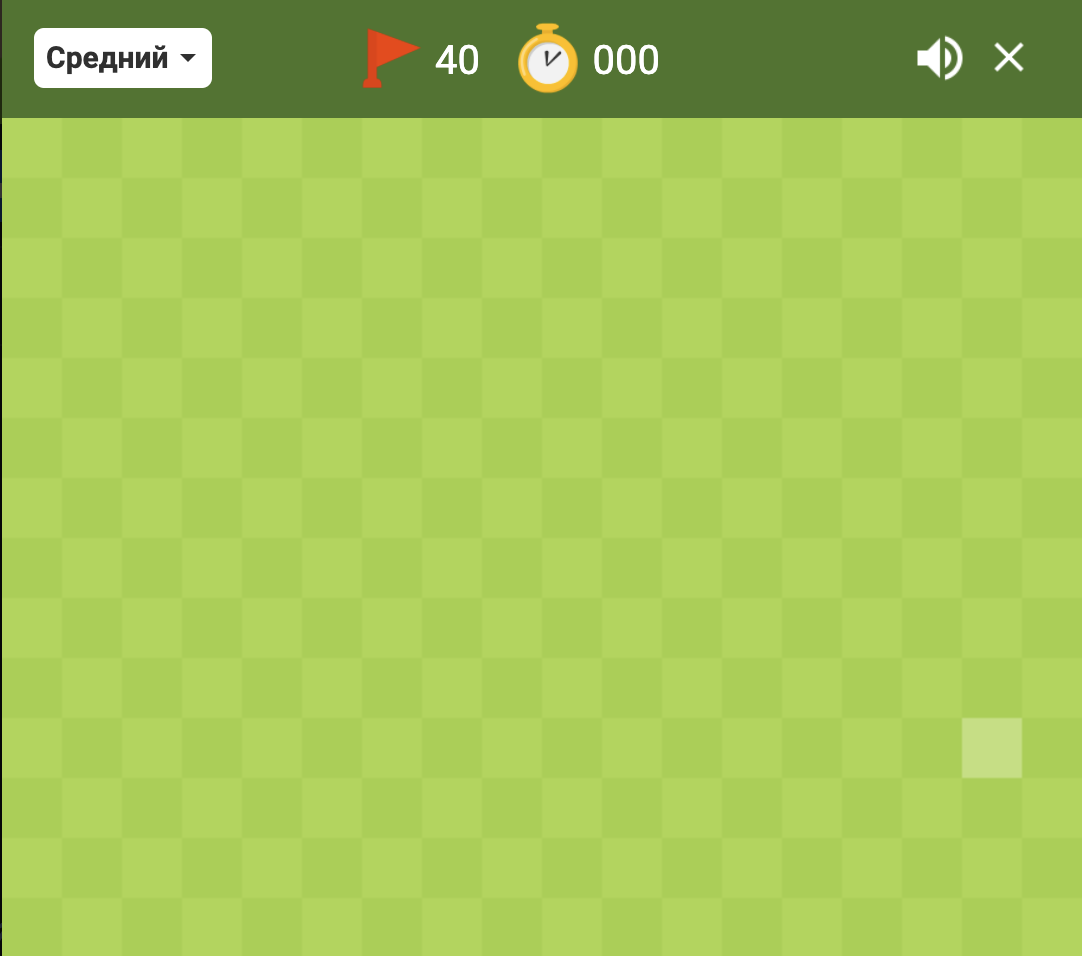


Рисунок 1.1 – Сапер від Google

* Сапер зі пакету ігор за замовчувань Windows;

Головну сторінку гри під Windows можна побачити на рисунку. 1.2.

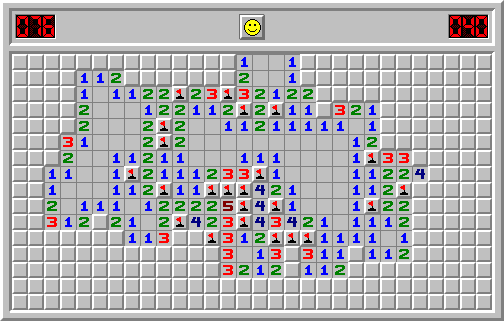


Рисунок 1.2 – Сапер під Windows

* декілька ігор Сапер для мобільних пристроїв;

Я проаналізував конкурентів, та можу зробити висновок, так як гра в моїй реалізації має можливість зберігати результат, я позиціоную її як web додаток, в якому можна проводити змагання, оцінка в рейтингу йде кількість часу проходження гри, та рівень складності, в такому випадку ми намагаємось мінімізувати час проходження гри и це легше робити граючи мишкою, завдяки цієї умови ми не будемо порівнювати мобільні реалізації гри.

Головну сторінку моєї реалізації гри можна побачити на рисунку. 1.3.

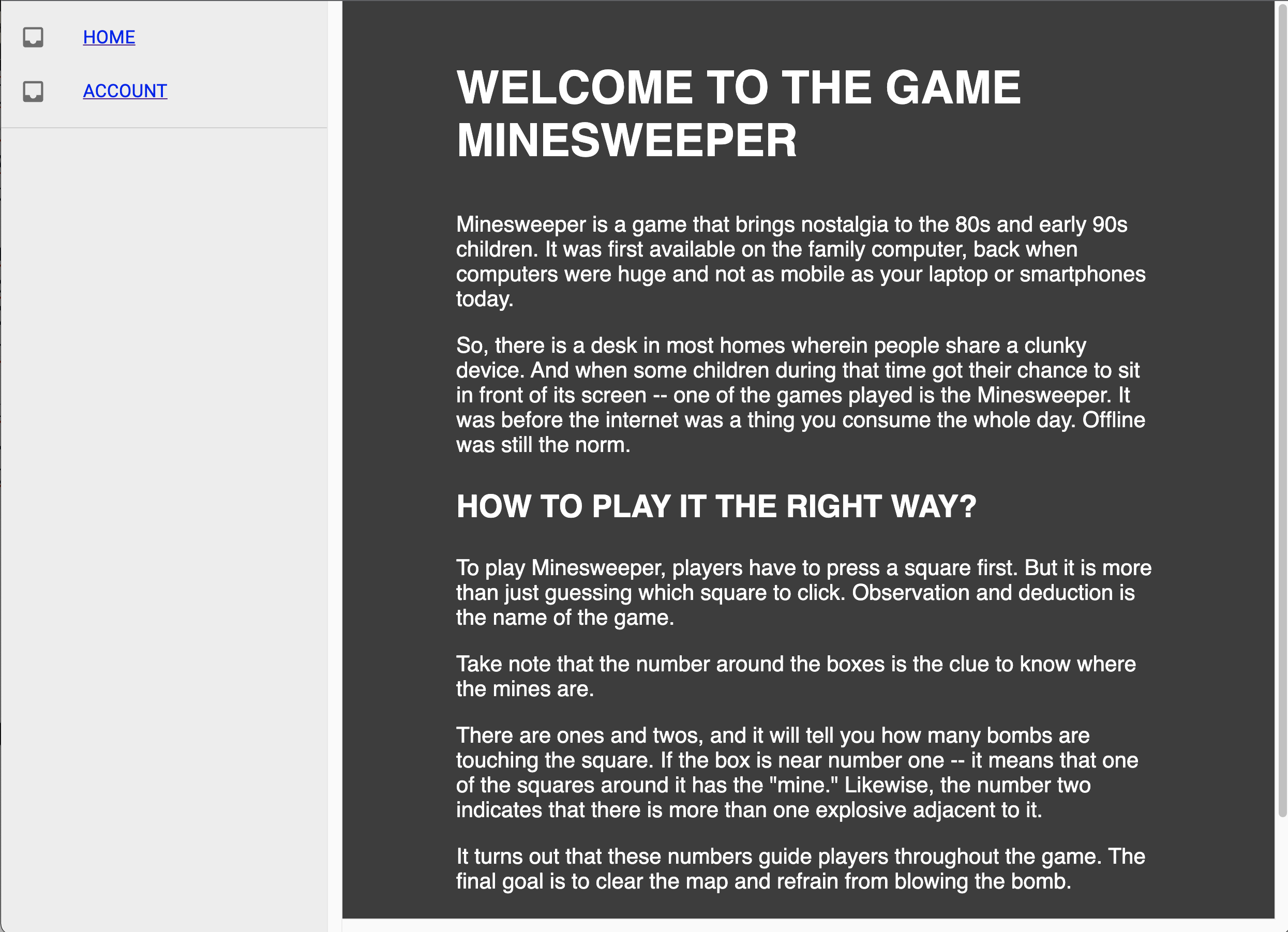


Рисунок 1.3 – Сапер курсової роботи

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.3.

Таблиця 1.1 – Порівняння з аналогом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал | Курсова робота | Аналоги Сапер від інших розробників | | Пояснення |
| Сапер | Windows | Google |  |
| Таймер | + | + | + | Підраховує час проходження гри |
| Рівні | + | + | + | Рівні складності за (розміром поля, кількістю мін) |
| Підказка | + | - | - | Можливість відкрити клітинку без міни |
| Збереження результатів | + | - | - | При завершенні (виграші гри) збереження (часу проходження, складності рівня, …) |
| Перегляд результатів | + | - | - | Можливість перегляду результатів (рейтингу) всіх користувачів |
| Авторизація | + | - | - | Можливість авторизуватись |
| Кросплатформеність | + | - | + | Можливість грати на будь якій операційній системі |
| Інструкція | + | - | - | Можливість подивитися як грати в цю гру |

## Аналіз вимог до програмного забезпечення

### Розроблення функціональних вимог

Головною функцією програмного забезпечення є надання користувачу можливості грати в інтелектуальну гру Сапер, зберігати свої результати та порівнювати їх результатами інших користувачів, а також мати змогу авторизуватися.

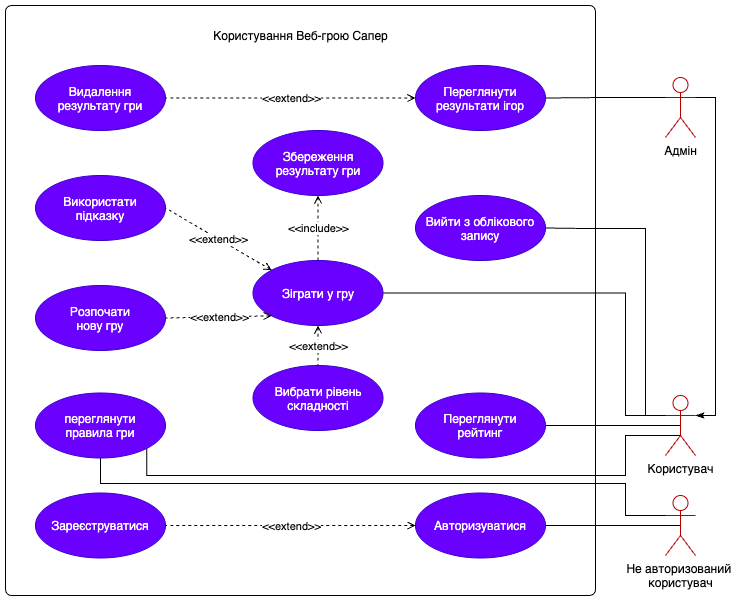


Рисунок 1.4 – Діаграма варіантів використання

В таблицях 1.2 - 1.8 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Авторизація |
| Use case ID | UC-01 |
| Goals | Авторизований користувач в системі |
| Actors | Гість (не авторизований користувач) |
| Trigger | Гість бажає авторизуватися |
| Pre-conditions | не авторизований користувач |
| Flow of Events | Користувач переходить на сторінку account. В поля для авторизації вводяться відповідні дані: ім’я користувача, пароль в системі. Після заповнення даних користувача натискає кнопку авторизації. |
| Extension | В випадку введення не правильних даних висвічується alert з помилкою |
| Post-Condition | Перехід на сторінку користувача |

Таблиця 1.3 - Варіант використання UC-2

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-02 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Гість (не авторизований користувач) |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | не авторизований користувач |

Продовження таблиці 1.3

|  |  |
| --- | --- |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім’я користувача, пароль в системі. Після заповнення даних користувача натискає кнопку реєстрації. |
| Extension | В випадку введення неправильних даних висвічується alert з помилкою |
| Post-Condition | Створення сторінки користувача, перехід на сторінку користувача |

Таблиця 1.4 - Варіант використання UC-3

|  |  |
| --- | --- |
| Use case name | Вийти з облікового запису |
| Use case ID | UC-03 |
| Goals | Користувач вийшов з облікового запису |
| Actors | Користувач |
| Trigger | Користувач бажає вийти з облікового запису |
| Pre-conditions | Користувач авторизованій |
| Flow of Events | Користувач переходить на свою сторінку. Натискає на кнопку вийти. |
| Post-Condition | Перехід на сторінку авторизації |

Таблиця 1.5 - Варіант використання UC-4

|  |  |
| --- | --- |
| Use case name | Переглянути результати ігор |
| Use case ID | UC-04 |
| Goals | Admin знаходиться на сторінці перегляду результатів |
| Actors | Admin |
| Trigger | Admin бажає видалити результат або переглянути результат |
| Pre-conditions | Admin знаходиться на своїй сторінці |
| Flow of Events | Admin переходить на сторінку результатів. Має змогу видалити результат при натисканні на іконку смітничка |
| Post-Condition | Можливість видалення результату |

Таблиця 1.6 - Варіант використання UC-5

|  |  |
| --- | --- |
| Use case name | Переглянути рейтинг виграних ігор |
| Use case ID | UC-05 |
| Goals | Користувач знаходиться на сторінці рейтингу виграних ігор |
| Actors | Користувач |
| Trigger | Користувач бажає подивитись рейтинг виграних ігор усіх користувачів |
| Pre-conditions | Користувач знаходиться на своїй сторінці |
| Flow of Events | Користувач переходить на сторінку рейтингу виграних ігор усіх користувачів. Має змогу подивитися рейтинг відсортований за часом проходження. Виділені зірочкою лідери рейтингу. |
| Post-Condition | Можливість видалення результату |

Таблиця 1.7 - Варіант використання UC-6

|  |  |
| --- | --- |
| Use case name | Зіграти у гру |
| Use case ID | UC-06 |
| Goals | Користувач розгадує ігрове поле |
| Actors | Користувач |
| Trigger | Користувач бажає покращити свій результат та зіграти у гру |
| Pre-conditions | Користувач знаходиться на сторінці гри, але не почав грати |
| Flow of Events | Користувач натискає на комірку, перший крок зроблений, далі ставить прапорець де розташована міна та розкриває усі комірки, якщо позначити всі комірки з мінами та розкрити всі інші користувач виграє, якщо відкрити міну, тоді навпаки програє. |
| Post-Condition | Після закінчення гри зберігається результат |

Таблиця 1.8 - Варіант використання UC-7

|  |  |
| --- | --- |
| Use case name | Використати підказку |
| Use case ID | UC-07 |
| Goals | Відкривається комірка без міни |
| Actors | Користувач |
| Trigger | Користувач не знає яку комірку відкрити |
| Pre-conditions | Користувач знаходиться на сторінці гри, гра розпочата |
| Flow of Events | Відкривається комірка без міни |
| Post-Condition | Відкривається комірка без міни |

Web-додаток розділений на модулі. Кожен модуль має свій певний набір функцій. В таблицях 1.9 – 1.14 наведений опис функціональних вимог до програмного забезпечення, розділений на модулі відповідно. Матрицю трасування вимог можна побачити на рисунку 1.15.

Таблиця 1.9 – Функціональна вимога FR-1

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Система повинна надавати можливість реєстрації користувачеві шляхом введення  ім’я користувача, паролю. Далі надсилання цих даних до серверу, та збереження до бази даних |

Таблиця 1.10 – Функціональна вимога FR-2

|  |  |
| --- | --- |
| Назва | Авторизація користувача |
| Опис | Система повинна надавати можливість авторизації користувачеві шляхом введення  ім’я користувача, паролю. Слід за цим пошук користувача в базі даних, отримання та порівняння токену на сервері з зашифрованим паролем. Далі перенаправлення на сторінку користувача |

Таблиця 1.11 – Функціональна вимога FR-3

|  |  |
| --- | --- |
| Назва | Збереження авторизація |
| Опис | При перезагрузці сторінки авторизація зберігається завдяки збереженню токена |

Таблиця 1.12 – Функціональна вимога FR-4

|  |  |
| --- | --- |
| Назва | Перегляд рейтингу |
| Опис | Отримання усіх виграшних ігор з бази даних для авторизованого користувача |

Таблиця 1.13 – Функціональна вимога FR-5

|  |  |
| --- | --- |
| Назва | Редагування списку завершених ігор |
| Опис | При натисканні на іконку смітника біля результату гри видаляється результат з бази даних та перерендується компонент |

Таблиця 1.14 – Функціональна вимога FR-6

|  |  |
| --- | --- |
| Назва | Проходження гри |
| Опис | На клієнті вибирається рівень складності, можливість почати гру спочатку, після завершення гри зберігається результат в базу даних |

Таблиця 1.14 – Функціональна вимога FR-7

|  |  |
| --- | --- |
| Назва | Вихід |
| Опис | Видалення токену з локального сховища |

Таблиця 1.15 – Матриця трасування вимог

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC-01 | UC-02 | UC-03 | UC-04 | UC-05 | UC-06 | UC-07 |
| FR-1 |  | + |  |  |  |  |  |
| FR-2 | + |  |  |  |  |  |  |
| FR-3 |  |  |  |  |  |  |  |
| FR-4 |  |  |  | + | + |  |  |
| FR-5 |  |  |  | + | + | + |  |
| FR-6 |  |  |  |  |  | + | + |
| FR-7 |  |  | + |  |  |  |  |

### Розроблення нефункціональних вимог

Програмне забезпечення повинно бути:

* зрозумілим для користувача;
* бути легким у використанні;
* підтримуватись всіма сучасними браузерами:
  1. Firefox 93.0 та новіша
  2. Chrome 95.0.4638.54 та новіша
  3. Opera 80.0.4170.63 та новіша
  4. Edge 95.0.1020.30 та новіша
  5. Safari 13.0 та новіша

### Постановка задачі

Розробка призначена для кінцевих користувачів**,** що мають метою:

* проводити вільний час з користю, граючи в логічну гру;
* зберігати свої ігрові результати;
* спостерігати за своїми досягненнями серед інших гравців;

Розробка ставить перед собою досягнення наступних цілей:

* створення веб-гру зі зручним інтерфейсом;
* реалізація системи авторизації;
* взаємодія з базою даних для зберігання результатів користувачів;
* реалізація гри, з можливістю використання підказок;
* надання прав видаляти результати ігор користувачу з роллю admin;

Також програмне забезпечення для досягнення мети має бути зроблені всі поставлені функціональні та нефункціональні вимоги, які описані в технічному завданні.

## Висновки по розділу

В першому розділі курсової роботи було проаналізовано та обрано стек технологій для написання гри, завдяки перегляду можливих конкурентів та виявленню їх сильних та слабких сторін, виконав аналіз предметної області, виявив конкурентів, розглянув позитивні й негативні сторони конкурентів за допомогою таблиці порівнянь. Була побудована UML діаграма use case в якій розкрито сценарій предметної області зі сторони користувача, з детальним описом кожного use case за допомогою таблиць де вказано актор виконання цієї дії. Зроблений детальний опис функціональних вимог. Створено матрицю трасування в якій видно залежність функціональних вимог від use case. Були вказані не функціональні вимоги. Вказана постановка задачі та призначення мети, а також цілі які ставить перед собою розробка гри

# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується BPMN модель.

Схему опису бізнес процесу реєстрації користувача можна побачити на рисунку 2.1

Схему опису бізнес процесу авторизації користувача можна побачити на рисунку 2.2

Схему опису бізнес процесу проходження гри можна побачити на рисунку 2.3

Схему опису бізнес процесу видалення результатів гри адміном можна побачити на рисунку 2.4

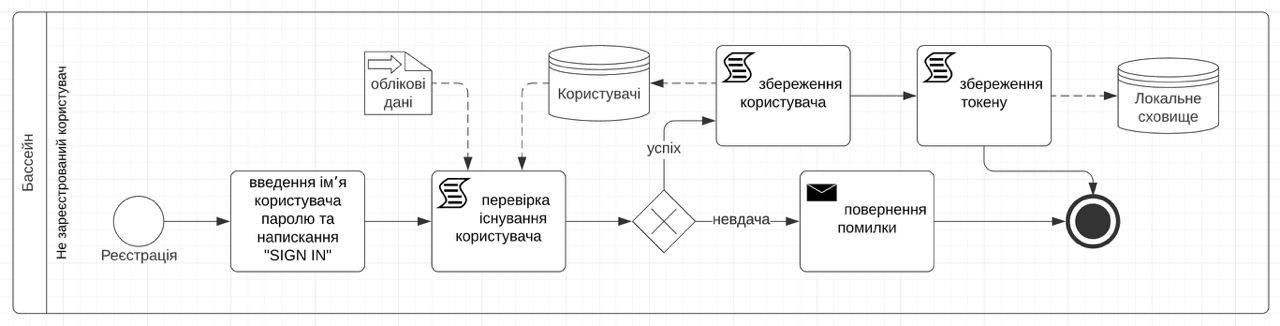


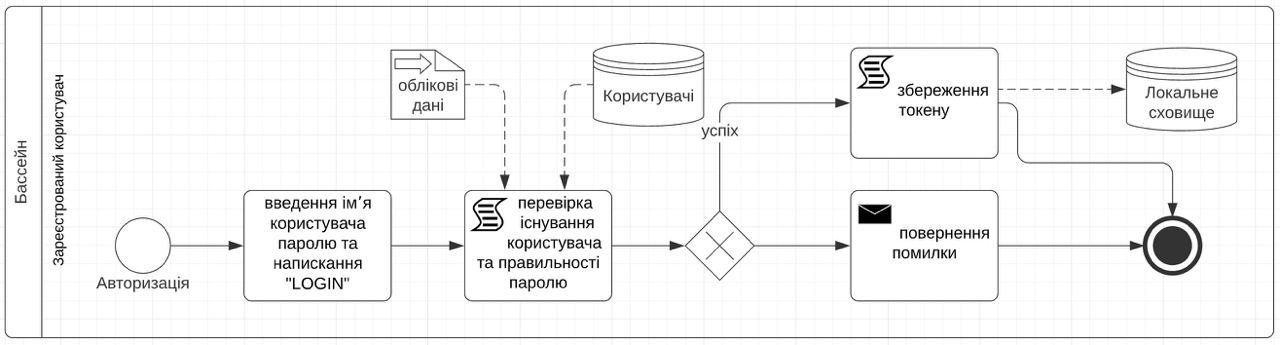
Рисунок 2.1 – Схема бізнес-процесу реєстрації користувача

Рисунок 2.2 – Схема бізнес-процесу авторизації користувача

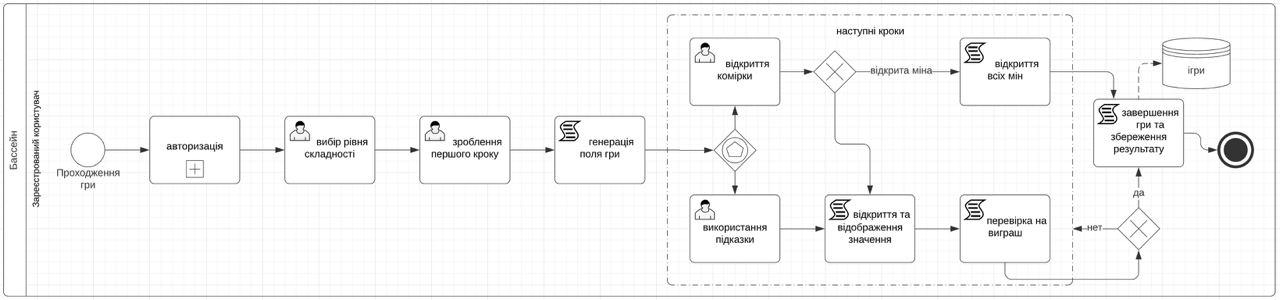
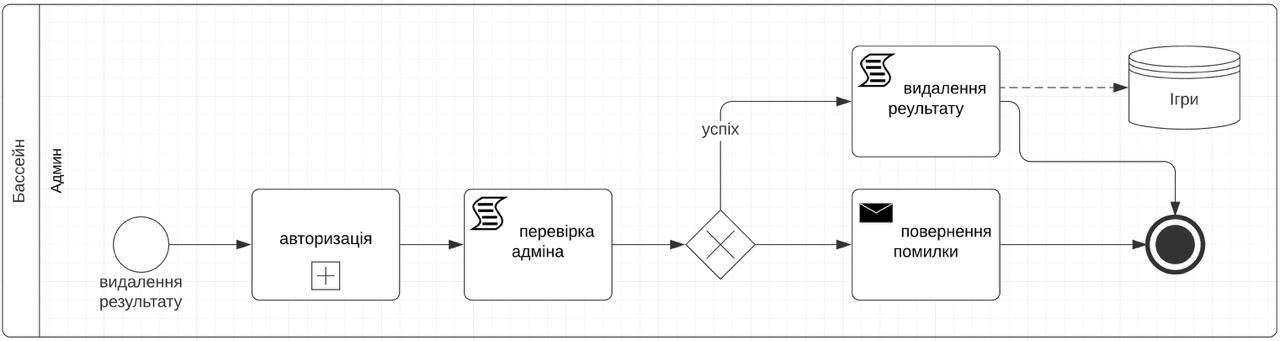
Рисунок 2.3 – Схема бізнес-процесу проходження гри 

Рисунок 2.4 – Схема бізнес-процесу видалення результатів гри адміном

## Архітектура програмного забезпечення

За основу архітектури програмного забезпечення серверної частини був взятий багаторівневий шаблон

## Конструювання програмного забезпечення

Опис БД, або структури зберігання даних

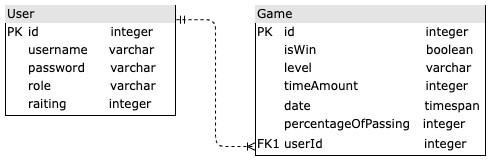


Рисунок 2.5 – Діаграма бази даних

Таблиця 2.1 – Опис таблиці user для бази даних

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва сутності | Назва поля | Тип даних | PK | M | Детальна інформація |
| User | id | integer | X | X | Ідентифікатор користувача |
| username | varchar |  | X | Ім’я користувача |
| password | varchar |  | X | Пароль користувача |
| role | varchar |  |  | Роль |
| raiting | integer |  | X | Рейтинг |

Таблиця 2.2 – Опис таблиці game для бази даних

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва сутності | Назва поля | Тип даних | PK | M | Детальна інформація |
| Game | id | integer | X | X | Ідентифікатор гри |
| isWin | boolean |  | X | Пройшов гру чи ні |
| level | varchar |  | X | Рівень складності |
| timeAmount | integer |  |  | Час проходження гри |
| date | timespan |  | X | Дата проходження гри |
| percentageOfPassing | integer |  | X | Процент проходження гри |

Аналіз системних вимог

Таблиця 2.3 – Аналіз системних вимог

|  |  |
| --- | --- |
| Системні одиниці | Бажане значення |
| Тип процесору | Intel core i3 |
| Об‘єм ОЗП | 4 ГБ |
| Підключення до мережі Інтернет зі швидкістю від | 15 мегабіт |

Опис утиліт які були використані для розробки наведено в таблиці 2.4.

Таблиця 2.4 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| № п/п | Назва утиліти | Опис застосування |
| 1 | WebStorm | Головне середовище розробки програмного забезпечення серверної та клієнтської частини курсової роботи. |
| 2 | Postman | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 3 | Datagrid | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних. |

## Висновки по розділу

В другому розділі курсової роботи було розглянуто та проаналізовано наступне:

* створено опис роботи додатку за допомогою BPMN;
* була обрана модель архітектурного шаблону;
* побудована діаграма бази даних;
* зроблено опис сутностей в діаграмі;
* описано рекомендовані системні вимоги;
* зазначено програмні утиліти які будуть використовуватись в роботі;

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

Сьогодні є безліч фреймворків для тестування, що підтримують практично всі мови. Здавалося б, можна брати і автоматизувати. Але навіть зараз ручні випробування важливі. Одне з пояснень їх необхідності полягає в тому, що при ручному тестуванні функціоналу ми можемо набагато швидше отримати інформацію про стан продукту, який аналізуємо, якість розробки. Крім того, при автоматизації попередньо розроблені кейси часто доводиться міняти та актуалізувати, а на написання автотестів потрібен певний час. При врахуванні якості програмного забезпечення брались часові показники відповіді сервера на клієнтські запити як в звичайному режимі, так і в режимі високого навантаження.

Оцінювалась зручність та зрозумілість інтерфейсу, розташування кнопок та меню web-додатку. Також було враховано лаконічність інтерфейсу, кольорова гамма, на інтерактивність (анімації, динамічні зміни форм та значень). Вигляд екранних форм можна побачити в додатку до курсової роботи.

## Опис процесів тестування

Тестування програмного коду було здійснено за допомогою ручного тестування. Детальний опис та аналіз процесів тестування наведено в «Програма та методика тестування» курсової роботи.

## Опис контрольного прикладу

В таблицях 3.1 - 3.6 описано мануальне тестування web-додатку.

Таблиця 3.1 – Тестування № 1.1 авторизації користувача

|  |  |
| --- | --- |
| Тест | Перевірка авторизації користувача |
| Номер тесту | 1.1 |
| Початковий стан системи: | Відкрита сторінка авторизації, користувач вже зареєстрований в системі |
| Вхідні дані: | Ім’я та пароль користувача |
| Опис проведення тесту: | В відповідні поля вводяться коректні дані і після цього натискається кнопка «login» |
| Очікуваний результат: | Авторизація повинна пройти успішно, а користувача має перенаправити на сторінку свого облікового запису |
| Стан системи після проведення тесту | Авторизація успішна, користувач перенаправлений на сторінку облікового запису |

Таблиця 3.2 – Тестування № 1.2 реєстрації користувача

|  |  |
| --- | --- |
| Тест | Перевірка реєстрації користувача |
| Номер тесту | 1.2 |
| Початковий стан системи: | Відкрита сторінка реєстрації |
| Вхідні дані: | Ім’я та пароль користувача |
| Опис проведення тесту: | В відповідні поля вводяться коректні дані і після цього натискається кнопка «sign in» |
| Очікуваний результат: | Реєстрація повинна пройти успішно, а користувача має перенаправити на сторінку свого облікового запису |
| Стан системи після проведення тесту | Авторизація успішна, користувач перенаправлений на сторінку облікового запису |

Таблиця 3.3 – Тестування № 1.3 Видалення результату гри адміном

|  |  |
| --- | --- |
| Тест | Видалення результату гри адміном |
| Номер тесту | 1.3 |
| Початковий стан системи: | Відкрита сторінка адміна, користувач вже авторизований в системі як адмін |
| Вхідні дані: | - |
| Опис проведення тесту: | Натискання на смітник біля результату гри яку треба видалити |
| Очікуваний результат: | Повинна пройти успішне видалення результату |
| Стан системи після проведення тесту | Успішне видалення результату з списку ігор |

Таблиця 3.4 – Тестування № 1.4 Вибір рівня складності гри

|  |  |
| --- | --- |
| Тест | Вибір рівня складності гри |
| Номер тесту | 1.4 |
| Початковий стан системи: | Відкрита сторінка гри, користувач вже авторизований в системі |
| Вхідні дані: | - |
| Опис проведення тесту: | Натискання на селект та вибір певного рівня складності (SENIOR, MIDDLE, JUNIOR) |
| Очікуваний результат: | Повинна пройти успішний рендер нового рівня складності, заміна кількості мін та зміна розміру поля |
| Стан системи після проведення тесту | Успішне проходження рендеру нового рівня складності, заміна кількості мін та зміна розміру поля |

Таблиця 3.5 – Тестування № 1.5 Першого кроку гри

|  |  |
| --- | --- |
| Тест | Першого кроку гри |
| Номер тесту | 1.5 |
| Початковий стан системи: | Відкрита сторінка гри, користувач вже авторизований в системі, вибраний рівень складності |
| Вхідні дані: | - |
| Опис проведення тесту: | Натискання на будь-яку комірку на полі гри |
| Очікуваний результат: | Повинна пройти успішне відкриття декількох комірок, разом з натиснутою коміркою |
| Стан системи після проведення тесту | Успішне проходження відкриття декількох комірок, разом з натиснутою коміркою |

Таблиця 3.6 – Тестування № 1.6 Проходження гри

|  |  |
| --- | --- |
| Тест | Проходження гри |
| Номер тесту | 1.6 |
| Початковий стан системи: | Відкрита сторінка гри, користувач вже авторизований в системі, вибраний рівень складності, почата гра |
| Вхідні данні: | - |
| Опис проведення тесту: | Натискання на комірку під якою знаходиться міна, або відкриття усіх комірок без мін |
| Очікуваний результат: | Повинна пройти успішне відображення результату гри “You are win” чи “You are loose” |
| Стан системи після проведення тесту | Успішне відображення результату гри “You are win” чи “You are loose” |

3.4 Висновки до розділу

Під час проведення тестування, було протестована функціонал web-додатку. Описані основні тест кейси, та стани системи після проведення тестування.

Завдяки ручному тестуванню знизили кількість помилок, виявили та усунули «вузькі місця», забезпечили стабільну роботу додатку, оцінили зручність експлуатації, і в результаті отримали більш якісний продукт, що задовольняє очікуванням.

Виявили сильні сторони ручного тестування

Проект із ручного тестування можна швидко запустити

* ручне тестування дозволяє вносити коригування (міняти та актуалізувати тести) у разі потреби;
* легко адаптується до динамічних змін системи, що тестується;
* ручне тестування повністю імітує фактичне використання системи кінцевим користувачем;
* дозволяє отримати фідбек, особливо на разових і нетривалих проектах;

Основні етапи ручного тестування програмного забезпечення.

* підготовчий
  1. Аналіз вихідних документів про систему (наприклад, технічне завдання, паспорт проекту, бізнес-вимоги, функціональні вимоги та інші)
  2. Розробка та узгодження плану ручного тестування, тест-кейсів, термінів, кількості ітерацій та інше
  3. Оцінка можливих ризиків, визначення меж проекту
* основний
  1. Виконання ручного тестування на основі специфікацій вимог та заздалегідь підготовленим тестовим сценаріям
  2. Фіксація виявлених дефектів у системі відстеження помилок
* заключний
  1. Розробка та погодження звітів про проведене тестування
  2. Створення рекомендацій щодо впровадження ПЗ, вдосконалення та виправлення помилок в додатку

# ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

Для роботи програмного забезпечення ви повинні виконати деякі пункти:

* перейти по посиланню <https://github.com/ilkrbk?tab=repositories>
* натисніть кнопку «CODE»
* натисніть на пункт «Download ZIP»
* перейдіть до папки Завантаження
* розархівуйте завантажений файл додатком на вибір (WinRAR, 7-ZIP, або скористайтеся додатком за замовчувань)
* завантажте Node.js та за посиланням <https://nodejs.org/en/>
* відкрийте термінал та виконайте команду npm install --global yarn
* завантажте базу даних PostgreSQL та postgres за посиланням <https://www.postgresql.org/download/> <https://postgresapp.com/>
* запустіть postgres та створіть базу даних
* потрібно перейти в роз архівовану папку та в папку server
* заповніть файл .env

PORT=8000

DB\_NAME=«ім’я бази даних»

DB\_USER=«ім’я користувача»

DB\_PASSWORD=«пароль»

DB\_HOST=localhost

DB\_PORT=5432

SECRET\_KEY=minesweeper\_react\_postgresql\_node\_express\_\_ilkrbk

* відкрийте два вікна термінала
* в першому введіть команди

cd шлях до завантаженої папки/server

yarn install

yarn run dev

* в другому введіть команди

cd шлях до завантаженої папки/front

yarn install

yarn run start

* перейдіть за посиланням <http://localhost:3000/>
* розважайтесь з задоволенням

## Висновок до розділу

Під час написання цього розділу, було описана інструкція як запустити курсовий проект на власному ПК. В якій наданні посилання для завантаження проекту та додаткового програмного забезпечення. Також встановлення і налаштування цих додатків

**ВИСНОВКИ**

Основна мета роботи була виконана. В рамках даної курсової роботи було створено WEB-додаток для гри в Сапер. Також були досягнені усі поставлені перед розробкою цілі:

* створення веб-гру зі зручним інтерфейсом
* реалізація системи авторизації
* взаємодія з базою даних для зберігання результатів користувачів
* реалізація гри, з можливістю використання підказок
* надання прав видаляти результати ігор користувачу з роллю admin

Проаналізовано та обрано стек технологій для написання гри, завдяки перегляду можливих конкурентів та виявленню їх сильних та слабких сторін, виконав аналіз предметної області, виявив конкурентів, розглянув позитивні й негативні сторони конкурентів за допомогою таблиці порівнянь. Була побудована UML діаграма use case в якій розкрито сценарій предметної області зі сторони користувача, з детальним описом кожного use case за допомогою таблиць де вказано актор виконання цієї дії. Зроблений детальний опис функціональних вимог. Створено матрицю трасування в якій видно залежність функціональних вимог від use case. Були вказані не функціональні вимоги. Вказана постановка задачі та призначення мети, а також цілі які ставить перед собою розробка гри

Створено опис роботи додатку за допомогою BPMN, обрана модель архітектурного шаблону, побудована діаграма бази даних, зроблено опис сутностей в діаграмі, зазначено програмні утиліти які будуть використовуватись в роботі та описано рекомендовані системні вимоги

Було проведено тестування ПЗ завдяки ручному тестуванню. За допомогою якого знизили кількість помилок, виявили та усунули «вузькі місця», забезпечили стабільну роботу додатку, оцінили зручність експлуатації, і в результаті отримали більш якісний продукт, що задовольняє очікуванням.

Було описана інструкція як запустити курсовий проект на власному ПК. В якій наданні посилання для завантаження проекту та додаткового програмного забезпечення. Також встановлення і налаштування цих додатків

**ПЕРЕЛІК ПОСИЛАНЬ**

1. Тестування [Електронний ресурс]. <https://www.appline.ru/services/testing/functionalnoe-testirovanie/ruchnoe-testirovanie>
2. PostgreSQL [Електронний ресурс]. <https://www.postgresql.org/>
3. Node.js [Електронний ресурс]. <https://nodejs.org/uk/docs/>
4. Плюси та мінуси розробки веб-додатків на Node.js [Електронний ресурс]. <https://codernet.ru/articles/web/plyusyi_i_minusyi_razrabotki_veb-prilozhenij_na_nodejs/>
5. Ручне тестування [Електронний ресурс]. <https://habr.com/ru/company/skillbox/blog/418889/>
6. Посібник Node.js з фактичного виконання інтеграційних тестів [Електронний ресурс]. <https://www.toptal.com/nodejs/nodejs-guide-integration-tests>
7. Історія комп’ютерної гри [Електронний ресурс]. <https://ru.wikipedia.org/wiki/Компьютерная_игра>
8. Історія створення гри «Сапера» [Електронний ресурс]. <https://storedigital.ru/2020/10/26/istoriya-sozdaniya-velichajshej-igry-kak-poyavilsya-na-svet-sapyor/>