Міністерство освіти і науки України
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів обробки природної мови
для створення карти D&D

Виконав:

Випускник 2 курсу, групи ІПЗздм-19-1

Грінько К.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо наукова

Керівник к. т. н., доц. Назаров О. С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри З.В. Дудар
(підпис) (прізвище, ініціали)

2021р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти

Кафедра Програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«___»_____ 20 ___ р.

**ЗАВДАННЯ**

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента *Грінько Костянтина Олексійовича*
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів обробки природної мови для створення карти D&D

затверджена наказом університету від _____ № _____

2. Термін подання роботи до екзаменаційної комісії 13 05 2021р.

3. Вихідні дані до роботи Розробити веб-додаток генерації мап для D&D Використовувати технології: Webstorm, Tenserflow, React, Redux, мови програмування Javascript, середа виконання веб-браузер

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі, постановка задачі, математична модель, проведення експерименту, висновки, перелік джерел посилань, додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Кваліфікаційна робота магістра, Мета роботи, Постановка задачі, Аналоги, Методи Рішення, Flux Architecture, Flux Architecture, Дослідження, Наукова Апробація, Подальший розвиток, Висновки, Дякую за увагу

6. Консультанти розділів роботи (проекту)

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| Спец. розділ | Доц. каф. ПІ Назаров О. С. | | |

7. Дата видачі завдання «_25_»__01__2021 р.

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка * |
|---|---|---|---|
| 1. | Аналіз предметної галузі | 03.02.2021 | Виконано |
| 2. | Огляд існуючих методів | 20.02.2021 | Виконано |
| 3. | Методи оброки природної мови | 05.03.2021 | Виконано |
| 4. | Підготовка пояснювальної записки | 08.03.2021 | Виконано |
| 5. | Підготовка презентації | 25.03.2021 | Виконано |
| 6. | Підготовка доповіді | 04.04.2021 | Виконано |
| 7. | Нормоконтроль, рецензування | 03.06.2021 | Виконано |
| 8. | Занесення диплома в електронний архів | 08.06.2021 | Виконано |
| 9. | Попередній захист | 11.05.2021 | Виконано |
| 10. | Допуск до захисту у зав. кафедри | 11.05.2021 | Виконано |
| * заповнюється вручну після виконання чергового пункту | | | |

Дата видачі завдання 25 01 2021р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н. доц. каф. ПІ Назаров О.С.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційна роботи магістра, 60 стор., 5 рис., 6 табл., 11 джерел.

JAVASCRIPT, TENSERFLOW.JS, NODE, DUNGEON & DRAGONS, WEB APP, WEBSTORM, NATURAL LANGUAGE PROCESSING КВАЛІФІКАЦІЙНА РОБОТА ШТУЧНИЙ ІНТЕЛЕКТ.

Об'єкт розробки – веб додаток для створення бойових карт D&D.

Мета розробки – порівняння різних методів NATURAL LANGUAGE PROCESSING для створення візуальної репрезентації описаної локації.

Метод рішення – бібліотека для машинного навчання Tenserflow.js, Webstorm та Natural, мова JavaScript, React.

Створено веб додаток під різні версії браузерів, що створює мапу локації для гри D&D за допомогою розпізнавання тексту. Застосування може бути використано у будь-якому сучасному браузері.

An explanatory note to the attestation of the masters degree, 60 pg., 5 pic., 6 tab., 11 of sources.

JAVASCRIPT, TENSERFLOW.JS, NODE, DUNGEON & DRAGONS, WEB APP, WEBSTORM, NATURAL LANGUAGE PROCESSING

Object of development – web application.

Development goal – Research of Natural Language Processing Methods for Creation of a D&D Map.

Solution – Tenserflow.js, Webstorm, Natural, JavaScript, React.

Created a web application for different browser versions, that generates a location map for the D&D with the help of natural language processing. Web application can be used in any modern browser.

Я, *Грінько Костянтин Олексійович.*, студент групи *ІПЗздм-19-1*, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів обробки природної мови для створення карти D&D», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

# ЗМІСТ

# ВСТУП

Обробка природної мови (NLP) відноситься до галузі інформатики - а точніше, до галузі штучного інтелекту або штучного інтелекту - що стосується надання комп'ютерам здатності розуміти текст і вимовлені слова майже так само, як це можуть люди.

НЛП поєднує в собі обчислювальну лінгвістику - моделювання людської мови на основі правил - зі статистичними моделями, машинним навчанням та моделями глибокого навчання. Разом ці технології дозволяють комп'ютерам обробляти людську мову у вигляді текстових чи голосових даних та «розуміти» її повне значення разом із задумами та настроями мовця чи письменника. NLP керує комп'ютерними програмами, які перекладають текст з однієї мови на іншу, реагують на вимовлені команди та швидко узагальнюють великі обсяги тексту - навіть у реальному часі.

Є велика ймовірність того, що ви взаємодіяли з NLP у вигляді голосових систем GPS, цифрових помічників, програмного забезпечення для диктування мови в текст, чат-ботів обслуговування клієнтів та інших зручностей для споживачів.

Але NLP також відіграє зростаючу роль у корпоративних рішеннях, які допомагають впорядкувати ділові операції, підвищити продуктивність праці співробітників та спростити критично важливі бізнес-процеси.

Сучасні машини тепер здатні виконувати вузько визначені завдання з великою точністю, але - і це важливе застереження - що точність настільки ж хороша, як і якість, а в деяких випадках і кількість даних, що керують моделлю. Сучасний стан машинного навчання, завдяки введенню ретельно продуманих даних, зробить можливим незліченні вдосконалення існуючих продуктів і, врешті-решт, розробку окремо стоячого ШІ, хоча і не повністю автономних пристроїв ШІ від "роботів, що працюють без розуму "Вид.

Але в міру поглиблення машинного навчання ми починаємо наступний крок до все більш досконалого ШІ: глибоке навчання. Складний аналіз глибокого навчання досягається за допомогою нейронних мереж, так званих, оскільки вони

вільно імітують взаємопов’язану структуру людського мозку, щоб забезпечити багатошарову функціональність.

Також спостерігається прогрес у тому, як другий ключовий будівельний блок ШІ, обробка природних мов (NLP), перетворився на розуміння природної мови (NLU). Якщо НЛП - це здатність перекладати усну або письмову мову у форму, яку алгоритм може зрозуміти, а потім відповісти результатами на усній або письмовій мові, яку люди можуть зрозуміти, то НЛУ є загальновідомішою: здатність робити висновок про значення мови, а потім реагуйте відповідно, як люди роблять інстинктивно. Siri та Alexa - це перші кроки до того, щоб надати ШІ набагато простішу, більш людську простоту використання.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Аналіз предметної галузі

Dungeons & Dragons (зазвичай скорочується як D&D або DnD) - це фантастична настільна рольова гра, спочатку розроблена Гері Гігаксом та Дейвом Арнесоном. Вперше вона була опублікована у 1974 році Tactical Studies Rules, Inc. Вона публікується Wizards of the Coast (нині дочірньою компанією Hasbro) з 1997 року. Гра була виведена з мініатюрних бойових ігор, а варіація гри Chainmail служила початковою системою правил. Публікація D&D загальновизнана початком сучасних рольових ігор та рольової ігрової індустрії[1].

D&D відходить від традиційних бойових ігор, дозволяючи кожному гравцеві створити власного персонажа для гри замість військового формування. Ці персонажі починають уявні пригоди в межах фантастичної обстановки. Dungeon Master (DM) служить суддею гри та казкарем, зберігаючи при цьому обстановку, в якій відбуваються пригоди, і граючи роль жителів ігрового світу. Персонажі створюють вечірку, і вони взаємодіють із мешканцями обстановки та між собою. Разом вони вирішують дилеми, беруть участь у битвах, досліджують та збирають скарби та знання. У процесі персонажі заробляють очки досвіду (XP), щоб піднятися на рівень і стають дедалі потужнішими за низку окремих ігрових сесій.

Ранній успіх D&D призвів до поширення подібних ігрових систем. Незважаючи на конкуренцію, D&D залишається лідером на ринку рольової ігрової галузі.

D&D залишалася найвідомішою та найбільш продаваною рольовою грою в США, де, за оцінками, у цю гру зіграло 20 мільйонів людей, а книга та обладнання перевищили 1 мільярд доларів США. продажі по всьому світу. Гра була доповнена багатьма заздалегідь зробленими пригодами, а також комерційними налаштуваннями кампанії, придатними для використання звичайними ігровими групами.

Гравцям досить часто доводиться створювати або купувати додаткові елементи, що допоможуть їм зануритись у ігровий світ. Будь то картки з описом заклинань та вмінь, фігурки персонажів або карти для візуалізації текстового опису локації. Адже сама гра являє собою лише дві книги правил та декілька листів персонажів, утримувати усю цю інформацію у голові майже неможливо для більшості гравців.

Найчастіше використовуються карти які відповідають текстовому опису локації з компанії. Оскільки вони видаються в лише одно екземплярі та тільки для Данжен Мастера, щоб той міг візуалізувати усю повість для себе та пересказати її іншим гравцям. Для вирішення цієї проблеми були створенні такі онлайн ресурси як incarnate, dungeonfog, pyromancers. Ці веб-сайти дозволяють користувачам створювати мапи для їх компаній онлайн за допомогою лімітованого тайлсету який надається обраним сайтом. Розглянемо конкурентів даного продукту у таблиці 1.1.

Таблиця 1.1 – Порівняння конкурентів з кваліфікаційною роботою

| Назва сервісу | Можливість редагування | Підтримка скошених тайлів | Генерація мапи за допомогою опису |
|---|---|---|---|
| Incarnate | Так | Ні | Ні |
| Dungeon Fog | Так | Так | Ні |
| D&D Map Gen | Ні | Ні | Так |

Dungeon Fog дозволяє користувачу створювати мапи локації вказаного розміру. Сам сайт надає декілька видів тайлів які відрізняються один від одного графічно та естетично. Користувач може обирати будь-які з доступних йому тайлів, перетягувати та встановлювати їх на бажаній позиції. Серед тайлів можна знайти зображення різних предметів інтер’єру таких як – стіл, стілець, килим, діжки та багато інших. Також тайлсет має предмети екстер’єру, стіни, колодязь, двері, фонтан. Та предмети ландшафту, щебень, вода, гірки та кар’єри. Усі ці зображення дозволяють користувачу зібрати сцену локації схожу з описом який йде з набором пригоди D&D.

Incarnate також дозволяє створювати захоплюючі мапи локацій за допомогою тайлсетів проте їх спеціалізація це створення більш масивних мап таких як наприклад мапа острову, або планети та їх інструменти для роботи з більш локалізованими мапа, як наприклад мапа кімнати підземелля не настільки зручні як в Dungeon Fog.

З наведеної вище таблиці можна побачити декілька відмінностей між цим проектом та його конкурентам. Перш за все D&D Map Gen створюється для досліду можливостей NLP та AI у галузі настільних ігр для генерації карт спираючись на опис локації наданий користувачем у текстовому форматі. Через це було прийнято рішення не імплементувати можливість конструктора для користувачів та підтримку скошених тайлів у перших ітераціях продукту.

## 1.2 Веб-браузер як оточення

В наш час веб-технології стрімко розвиваються, це дозволяє веб-розробниками створювати нові, біль стабільні та легко масштабуємі сайти та веб-додатки. Переважно, це робиться за допомогою новіших технологій веб-розробки, таких як Webpack, React, Vue або Angular. Одна загальна риса усіх перелічених технологій це факто того, що вони відходять від стандартного підходу створення статичних сайтів з використанням HTML, CSS та JQuery.

HTML або Hyper Text Markup Language, представляє собою мову розмітки для документів, призначених для відображення у веб-браузері, є основою усіх веб-сайтів та веб-додатків. У той час як CSS (Cascading Style Sheets) – мова таблиці стилів, використовується для опису презентації документа написаного мовою розмітки HTML. Ці дві технології є основою яка використовується й досі для створення більшості сайтів, особливо статичних, таких як наприклад сайтів створених на Wordpress.

Проте, як зазначено вище, створювати сучасні веб-сайти використовуючи лише ці дві технології є декілька архаїчним, оскільки такі веб-сайти не надають кращий User Experience, вони не дуже інтерактивні та не надають користувачі можливості зручно взаємодіяти з ними на усіх платформах. Саме ці перелічені недоліки виправляють такі, відносно, нові технології як React, Vue, Angular та Webpack. Ці технології дозволяють розробникам легко створювати такий тип веб-додатків як Progressive Web App.

Progressive Web App – це технологія, яка дозволяє сучасним мобільним браузерам перетворити веб-сайт у мобільний додаток, при цьому зберігаючи весь функціонал присутній у веб-сайті.

Таблиця 1.2 – Порівняння веб-фреймворків

| **Фреймворк** | **React** | **Vue** | **Angular** |
|---|---|---|---|
| Підтримка PWA | Так | Ні | Так |
| Багаторазові компоненти | Так | Так | Так |
| Висока продуктивність | Так | Так | Ні |
| Підтримує Flux архітектуру | Так | Ні | Так |

Як можна побачити з порівняння у таблиці 1.2 найбільше за все нам підходить веб-фреймворк React, оскільки він дозволяє створювати сучасні Progressive Web App та підтримує Flux архітектуру, за допомогою бібліотеки під назвою Redux. Разом з усіма цими плюсами React також є високо продуктивним фреймворком, що дозволяє загружати веб-сторінки дуже швидко[2].

Сам React – це Javascript бібліотека для створення юзер інтерфейсів. Він дозволяє створювати та розбивати елементи інтерфейсу на віддільні компоненти, які потім можна перевикористати у будь-якій частині веб-сайту, або веб-додатку. Ці компоненти можуть мати свій стан, який контролює відображення компоненту

у будь-який момент часу. Також великим плюсом React на відмінну від Angular є те, що він поширюється більше як бібліотека, ніж фреймворк, що дозволяє розробникам самим обирати необхідно архітектуру та технічний стек, у той чай як Angular будучи повноцінним фреймворком змушує розробника використовувати його структуру з коробки та адаптуватися під набір наданих абстракцій.

Саме тому обравши React нам необхідно обрати архітектуру, яка буде допомагати працювати з різними сторінками у веб-додатку та зберігати інформацію про загальний стан додатку.

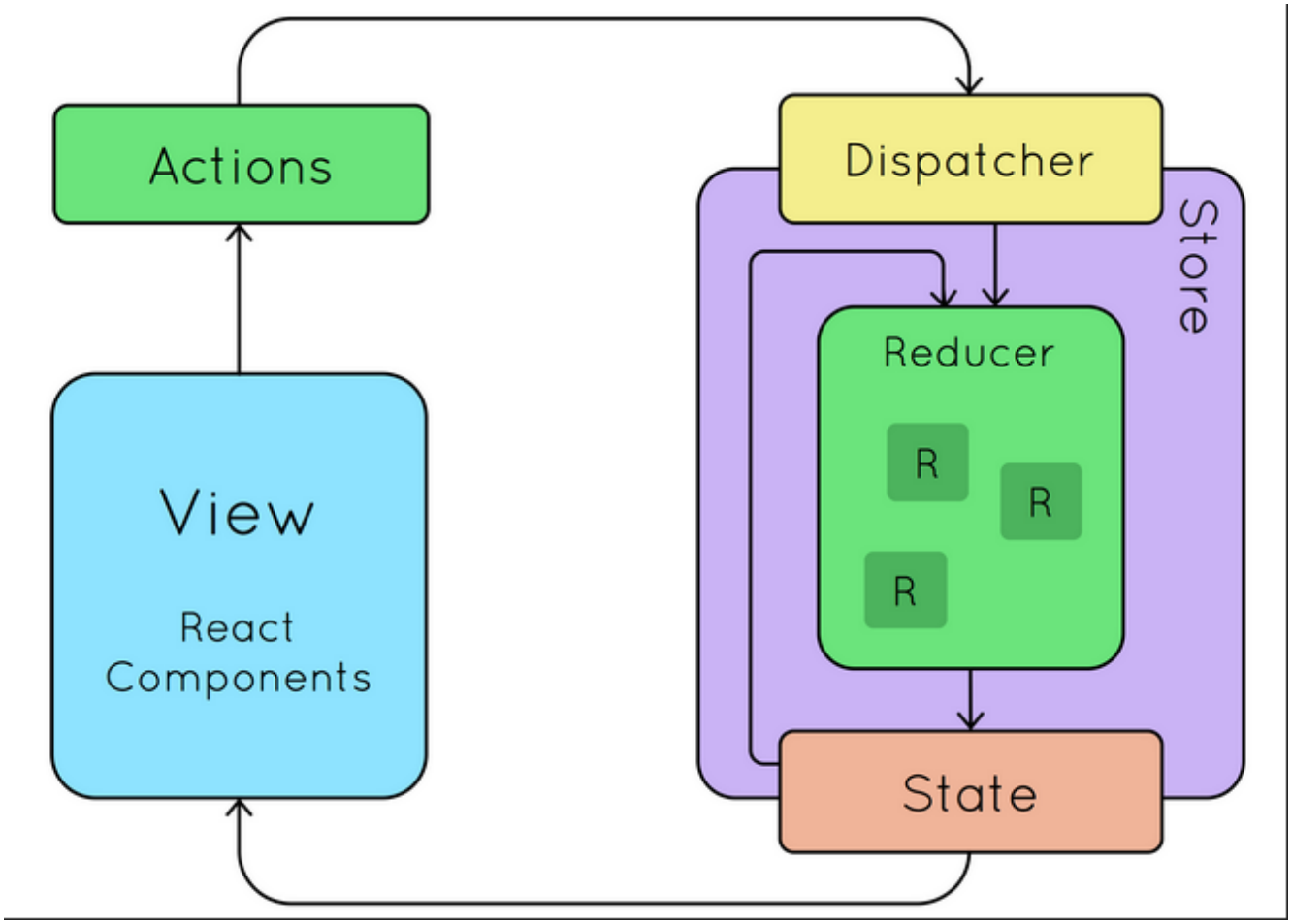


Рисунок 1.1 – Реалізація Flux за допомогою Redux

Для цього було обрано Flux архітектуру, імплементація якої може бути знайдена у найбільш популярній бібліотеці Redux[3]. Як можна побачити з діаграми зображеної на рисунку 1.1 ця архітектура добавляє декілька абстракцій до веб-додатку. Першою з с абстракцій є Action – він являє собою звичайний Javascript

об’єкт, який використовується для відображення події яка відбулася у додатку. Зазвичай Action має визначений тип, який дозволяє його розпізнати, та payload, або іншими слова набір даних який він у собі зберігає.

Далі ми маємо Reducer – це функція яка отримує нинішній стан веб-додатку та об’єкт Action. Ця функція слухає усі події які відбуваються у веб-додатку та вирішує як змінити його стан опираючись на отриманий тип події, якщо це необхідно.

Store – э простою репрезентацією нинішнього стану веб-додатку. Усі компоненти можуть отримати доступ до нього та зрозуміти, що саме вони повинні відображати у даний момент часу. Єдиним способом змінити стан веб-додатку є зміна у функції Reducer, яка зазвичай відбувається через події створенні іншими компонентами.

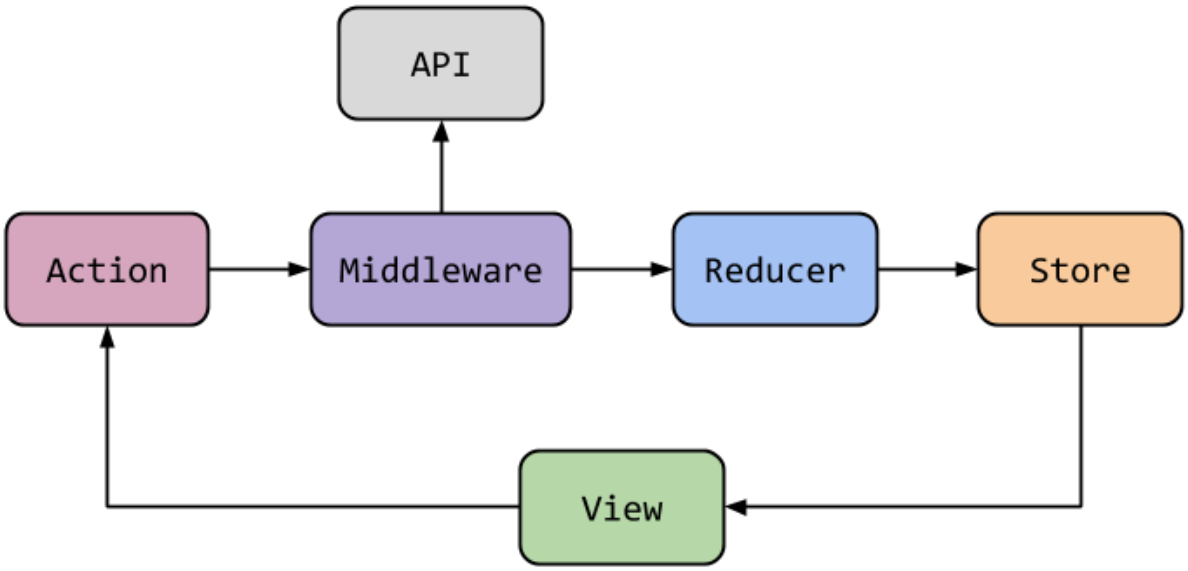


Рисунок 1.2 – Приклад роботи Redux разом з Redux-Saga

Проте не завжди весь життєвий цикл однієї події відбувається на фронтенді, або виконується синхронно. У таких випадках доречно використовувати таку технологію як Redux Saga. Redux Saga э прослойкою проміж Action та Reducer у циклі життя Redux. Вона виступає менеджером асинхронных подій у веб-додатках.

На приклад у даному веб-додатку штучний інтелект вимагає деякий час на обробку отриманих даних, веб-додаток не отримає відповіді від нього миттєво, тож необхідно розуміти, що у разі виконання події генерації мапи, веб-додаток повинен буде зрозуміти, що данні оброблюються або обробка даних закінчена і настав час відобразити їх для користувача. Сам для цього буде використано Redux Saga, вона буде створювати події, які записують у Store, що на даний момент текст ще опрацьовуєтеся.

Приклад Redux Saga можна побачити на рисунку 1.2 який зображає як Redux Saga, вказана як Middleware, робить асинхронний запит до віддільного API.

Тож на виході ми отримуємо веб-додаток який поводяться послідовно, працює у різних середовищах і легко тестуються.

# 2 ПОСТАНОВКА ЗАДАЧІ

## 2.1 Виявлення проблем та актуалізація рішень

Найбільша проблема у створені мап для D&D – це кількість часу яке потрібно витратити щоб прочитати весь модуль з історією, подіями, пастками та секретами які можуть бути заховані на одній мапі. Після цього данжен мастер повинен створити мапу яка зможе відобразити усе це, наповнити її інтер'єром, екстер'єром та, можливо, пастками й секретами. Отже проблемою є витрачання часу, а відповіддю автоматизація. Оскільки користувач завжди має створений ним, або куплений модуль який детально описує кожну локацію ми можемо використати увесь цей текст для того, щоб зрозуміти яку сам мапу потрібно створити. На щастя саме для таких задач було створено бібліотеку Tenserflow.js від Google

Таблиця 2.1 – Порівняння технологій доповненої реальності

| AI технологія | Язык програмування | Потребує backend | Має вже натренеровані моделі | Має інструменти візуалізації |
|---|---|---|---|---|
| Tenserflow | Javascript | Ні | Так | Так |
| Keras | Pyton | Так | Так | Ні |
| Microsoft CNTK | C++ | Так | Ні | Ні |

Tenserflow.js - це новіша розробка Google яка дозволяє створювати та тренувати моделі штучного інтелекту у браузері[4]. Також Tenserflow.js дозволяє конвертувати Python моделі для роботи з ними у браузері чи з Node.js. Tenserflow.js також надає можливість використовувати декілька заздалегідь натренованих моделей штучного інтелекту. Такий підхід дозволяє поширити використання штучного інтелекту та робить його більш доступним як для розробників так й для користувачів.

Штучний інтелект – це галузь яка дуже швидко розвивається у сучасному світі. Він може бути інтегрованим у будь-яку іншу галузь для покращення продуктивності, або автоматизації[5]. Та перш ніж стрибнути до використання Tenserflow.js розглянемо альтернативи у таблиці 2.1.

Альтернативами Tenserflow.js є Keras або Microsoft CNTK, проте ці бібліотеки спираються на язики програмування які потребують віддільного інстансу бекенду для роботи та обробки даних у той час як Tenserflow дозволяє використовувати та тренувати модель у самому браузері[6]. Також на відмінну від інших бібліотек Tenserflow.js має заздалегідь натреновані моделі які можна розширити та інструменти для візуалізації даних[7].

## 2.2 Постановка задачі

Спираючись на аналіз предметної галузі та виявлення проблем можна сформулювати задачі та створити вимоги до програмного продукту. Метою даної кваліфікаційної роботи є проектування та розробка веб сайту який використовує штучний інтелект та різні методи NLP для генерації мап D&D відповідно до опису користувача.

Методи NLP дозволять розбити введений користувачем текст у необхідні данні для обробки данні. Для обробки текстового опису локацій з описом інтер’єру, екстер’єру та загальним описом предметів, що знаходяться у ній було обрано метод обробки природної мови під назвою NER, або Named Entity Recognition[8].

Named Entity Recognition – один з найпростіших, та найкорисніших прийомів у обробці природної мови. Він вилучає різні сутності у тексті. У ньому висвітлено основні поняття та посилання, котрі користувач використовував для створення тексту. Розпізнавання іменованих сутностей (NER) ідентифікує такі сутності, як люди, місцезнаходження, організації, дати тощо, з тексту[9]. У нашому випадку розпізнавання іменованих сутностей буде використовуватись для ідентифікації локацій, та предметів.

Основна мета – створити інструмент для автоматичного створення бойових мап для настільної гри D&D, за допомогою штучного інтелекту. Даний інструмент представлятиме з себе веб-сайт який працюватиме в усіх сучасних браузерах, це дозволить будь-якому користувачу швидко створити необхідну для нього мапу.

Задачі які необхідно виконати:

- визначити вимоги для інструменту;
- спроектувати архітектуру проекту;
- створити UML моделі штучного інтелекту;
- розробити веб-сайт;
- протестувати інструмент;

Визначення вимог допоможе сформулювати потреби користувача веб-сайту. Після чого слід буде розробити архітектуру програмного забезпечення та відобразити основні ідеї та рішення у UML діаграмах.

## 2.3 Вимоги до програмного забезпечення

Веб-додаток має декілька вимог, а саме:

- веб-додаток повинен бути відкритим у будь-якому з сучасних браузерів;
- веб-додаток повинен мати юзер френдлі інтерфейс;
- веб-додаток повинен отримувати текст від користувача та віддавати зображення мапи на виході;
- текст, який надає користувач, повинен описувати одну локацію (кімнату, поляну, печеру, тощо);
- натренована модель має не перевищувати ліміти tenserflow.js.
- додаток повинен мати функцію зберігання зображення на девайсі користувача

Важливо зауважити, останні пункти. Текст повинен описувати одну локацію, а саме її вид та що в ній знаходиться. Така обмеженість є результатом лімітів технології Tenserflow.js, адже розмір моделі не повинен перевищувати 25MB. Також не менш важливим є пункт про зберігання мапи на девайсах користувачів,

не дивлячись не те, що він не є має відношення до ставлення експериментів цей пункт надає можливість впровадження додатку, оскільки він вирішує одну з великих проблем, з усіма знайденими конкурентами.

Працювати веб-додаток повинен в усіх сучасних браузерах, що не дивлячись на простоту звучання пункту не легко досягнути, оскільки майже усі сучасні браузери мають різні CSS стандарти та різні движки для виконання Javascript коду. Javascript код узагалі має декілька стандартів, кожен з яких привносить нові функції у язик, та новіші стандарти не завжди підтримуються браузерами. Саме тому в даному проекті було використано поліфіл Babel, та бандлер Webpack.

# 3 МАТЕМАТИЧНА МОДЕЛЬ

## 3.1 Опис методів дослідження

Основною метою дослідження є вплив різних методів обробки природної мови на кінцевий результат виводу веб-додатку, а саме створеної мапи локації. Необхідно порівняти результати обробки двох обраних методів обробки природної мови та визначити який з них найбільше підходить для виконання поставленої задачі – генерації мапи для D&D. Також необхідно виявити залежності від яких кінцевий результати може змінюватись.

Для дослідження цієї задачі необхідно по перш за все обрати методи обробки природної мови які будуть використовуватись для експериментів та провести декілька досліджень звертаючи увагу на швидкість виконання задачі та схожість тексту з отриманим зображенням.

Кінцевим результатом експерименту буде відсоткове відношення вдалих зображення мап від вибірки параграфів з різних мануалів D&D.

Дослідження буде проводитись за такими параметрами як швидкість генерації мапи моделлю та схожість зображення з текстовим описом локації (схоже, або не схоже).

Для цього введемо наступні метрики:

- S – швидкість генерації мапи(хв);
- T – загальна кількість спроб згенерувати мапу(шт);
- R – оцінка мапи на відповідність вимогам вказаним у тексті;
- V – відсоток вдалих мап від загальної кількості спроб(%).

Крім того будуть проведенні додаткові дослідження з корегуванням тих чи інших параметрів у моделях та перевірка їх впливу на кінцевий результат, щоб зрозуміти чи буде впливати зміна оточуючої системи на процес генерації мапи та отриманні данні.

## 3.2 Етапи наукових досліджень

Перший етап наукового дослідження полягає в зборі належного датасету, для тренування моделей Tenserflow. Датасет повинен включати в себе англійське слово, його частину мови, та тип зображення якому воно відповідає. Слова з датасету повинні бути підібрані з часто зустрічаємих у D&D мануалах для проходження компанії.

Другий етап дослідження полягає в тренуванні моделі. Ця модель буде використовувати такий метод обробки натуральної мови як – Named Entity Recognition, також відомий як NER результати виконання якого буде використано для того, щоб обрати відповідні картинки.

Третій етап полягає в тестуванні моделі за допомогою абзаців тексту, у якому описується та чи інша локація то зборі необхідних даних опираючись на результати виконання додатку.

Четвертий, останній, етап полягає у зборі та аналізі отриманих результатів експерименту. Це дозволить нам перевірити, які фактори можуть впливати на результати, та якщо необхідно, виправити їх. Після цього ми зможемо робити висновки, щодо ефективності моделей, та можливо, поліпшити веб-додаток.

## 3.3 Аналіз математичного апарату

Основною метою дослідницької роботи є дослідження здібностей штучного інтелекту у середі веб-браузеру. Для вирішення цієї необхідно розглянути та обрати математичну модель. Вона дозволить розрахувати та проаналізувати данні які є предметом дослідження, та отримати на результати на основі яких можна буде зробити висновки, щодо ефективності, або її відсутності обраного метода дослідження.

Саме дослідження буде використовувати вище зазначенні параметри, а саме – швидкість генерації мапи, загальна кількість спроб згенерувати мапу, кількість отриманих схожих на текстовий опис мап та відсоток вдалих мап від загальної кількості спроб.

S – це параметр, який відображає швидкість за яку було згенеровано мапу, його ми отримаємо від самого додатку після закінчення його виконання.

T – загальна кількість спроб введення тексту у експерименті

R – оцінка мапи на відповідність вимогам вказаним у тексті, вимірюється наступним образом

$$R = (M / K) * X$$

де K - – кількість іменників у наданому тексті (шт)

M – кількість зображень, що відповідає іменникам у тексті (шт)

X – відображає чи правильно був підібраний задній фон мапи (0 або 1)

мапа вважається вдалою при R >= 0.5

V – це відсоток вдалих мап (R > 0.5) від загальної кількості спроб, вимірюється наступним образом:

$$V = (R / T) * 100$$

що дозволить нам підрахувати відсоткове співвідношення вдалих мап за раунд експерименту.

# 4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ

## 4.1 Методологія проведення експерименту

Необхідно визначити загальну структуру експерименту, послідовність виконання дій, необхідних для отримання результату та визначення суті експерименту. Сам експеримент повинен відобразити якісність створеної мапи, та самого загального результату виконання веб додатку.

Для цього необхідно обрати набір абзаців які описують загальний вигляд локації з різної детальністю опису. Ці набори тестових абзаців будуть обрани з вже існуючих модулів пригод D&D, офіційних опублікованих від Wizards of the Coast, або створенні іншими організаціями, гравцями. Такі кастомні пригоди також називаються Homebrew.

Для того щоб перевірити точність виконання задачі веб-додатком, буде проведено декілька раундів в кожному з яких буде використовуватися різні абзаци опису локації з різною детальністю опису середовища.

Кожен раунд буде оцінено якість отриманої мапи та швидкість за яку мапу було згенеровано.

Останнім етапом буде аналіз отриманих даних проведеного експерименту та створення висновків на основі отриманих даних.

## 4.2 Специфікація програмного забезпечення

Розроблене програмне забезпечення – це веб-додаток який дозволяє користувачам ввести текст, що описує локацію (поляну, кімнату) та отримати мапу яка візуально схожа на текстовий опис, та має квадратне розділення для зручного подальшого використання у кампаніях пригод D&D.

В якості оточення було обрано веб-браузер користувача для того щоб дозволити використання веб-додатку на будь-яких платформах, що підтримують сучасні веб-браузери. Сам веб-додаток було розроблено за допомогою таких веб-технологій як React.js, для роботи з DOM браузеру та Redux для імплементації Flux архітектури разом з бібліотекою для тренування штучного інтелекту від Google.

Наведемо вимоги користувача для даного веб-додатку:

– користувач повинен мати можливість ввести бажаний текст;
– користувач повинен мати можливість зберегти згенеровану мапу;
– користувач повинен мати можливість перегляду мапу після генерації;
– користувач повинен мати можливість взаємодії с юзер інтерфейсом;
– користувач повинен мати можливість побачити швидкість з якою було згенеровано мапу;
– користувач повинен мати можливість побачити оцінку згенерованої мапи.

Інтерфейс дозволить користувачеві взаємодіяти з веб-додатком на будь-якому девайсі. Для першої ітерації проекту було реалізовано лише десктопний варіант інтерфейсу, проте не дивлячись на це, користувачі усе-одно можуть взаємодіяти з веб-додатком за допомогою мобільних девайсів.

Також важливими пунктами для дослідження є відображення оцінки мапи та швидкість її генерації, оскільки ці два пункту дозволять зручно збирати необхідні для дослідження метрики.

Зберігання створеної мапи  э дуже важливим для подальшого використання у кампаніях та пригодах D&D користувача. Подібний експорт зображення у форматі .png, або .jpg актуалізує даний веб-додаток не тільки як предмет дослідження, а й як корисний інструмент для всіх гравців у D&D, особливо данжон мастерів, оскільки цей функціонал дозволить їм не витрачати час на копіювання, або створення мапи власноруч.

## 4.3 Проведення експерименту

Для проведення експерименту було обрано десять різних абзаців з описом локації, різної складності. Сам текст було обрано з офіційних D&D модулів та з модулів створених гравцями.

Таблиця 4.1 – Результати експерименту

| **Очікувана кількість зображень на мапі** | **Отримана кількість зображень** | **Швидкість виконання** | **Оцінка мапи** | **Правильно підібраний задній фон** |
|---|---|---|---|---|
| 6 | 4 | 1.2c | 0.6 | Так |
| 3 | 3 | 0.7c | 1 | Так |
| 7 | 4 | 1.5c | 0.5 | Так |
| 8 | 3 | 1.8c | 0 | Ні |
| 4 | 4 | 1.2c | 1 | Так |
| 3 | 2 | 0.8c | 0.6 | Так |
| 5 | 3 | 1.1c | 0.6 | Так |
| 6 | 2 | 1c | 0.3 | Так |
| 5 | 2 | 1c | 0.4 | Так |
| 3 | 1 | 0.8c | 0.3 | Так |
| **Відсоток вдалих мап** | 60% | | | |

Як можна зазначити з результатів експерименту кількість вдалих мап склала шістдесят відсотків від загальної кількості спроб згенерувати мапу. Такий відсоток помилок зумовлений тим, що текст з більшою кількістю опису має більший шанс містити слова, які натренованій моделі невідомі. Це дуже гарно помітно на експериментах у яких очікуєма кількість зображень на мапі більше ніж чотири.

## 4.4 Аналіз результатів експерименту

Розглянемо кожний з параметрів представлених на таблиці 5.1 індивідуально.

Очікувана кількість зображень на мапі представляє собою кількість іменників, які описують собою предмети ландшафту, інтер'єру, екстер'єру.

Отримана кількість зображень – це кількість зображень які відповідають описаним у тексті ландшафту, інтер'єру, екстер'єру.

Швидкість виконання – це швидкість з котрою мапу було згенеровано з наданим текстом.

Оцінка мапи обчислюється наступним образом:

$$R = (M / K) * X$$

де K - – кількість іменників у наданому тексті (шт)

M – кількість зображень, що відповідає іменникам у тексті (шт)

S – відображає чи правильно був підібраний задній фон мапи (0 або 1)

мапа вважається вдалою при R >= 0.5

Отже для першого елементу, згідно таблиці 5.1, ми можем обчислити оцінку мапи наступним образом:

$$R = (4 / 6) * 1 = 0.6$$

Згідно нашого математичного апарату ця мапа була створено успішно, оскільки її R більше ніж 0.5 використовуючи цю. формулу було обчислено усі інші мапи, що представленні в таблиці 5.1.

Більшість з них отримала оцінку вище ніж 0.5, що є досить непоганим для даного дослідження, оскільки високий відсоток вдало створених мап дозволить користувача якісно генерувати мапи.

V – це відсоток вдалих мап (R >= 0.5) від загальної кількості спроб, вимірюється наступним образом:

$$V = (R / T)* 100$$

де T – загальна кількість спроб введення тексту у експерименті.

У нашому випадку V = (6 / 10) * 100 = 60%, що є досить непоганим результатом.

З отриманих даних можна зробити висновок, що у більшості випадків, веб-додаток здатний створювати мапи, які відповідають текстовому опису локації з параграфів важкість яких є легкою, або середньою (від 1 до 5ти об’єктів які необхідно зобразити на мапі).

Використаний метод розпізнавання натуральної мови, а саме Named Entity Recognition справляється з поставленою задачею на задовільному рівні, що дозволяє автоматизувати процес створення мап, для данжон мастерів та звичайних гравців в D&D.

Даний веб-додаток можна покращити, методом розширення тайлсету, який використовується для накладення необхідних зображень на мапу для D&D, це дозволить відображати більшу кількість розпізнаних предметів на згенерованих мапах та покращить результати наступних експериментів.

# 5 ОПИС РОЗРОБЛЕННОЇ ПРОГРАМНОЇ СИСТЕМИ

## 5.1 Обгрунтування середовища розробки

Для виконання поставленої задачі було обрано розробку Google, Tensorflow. Ця бібліотека для тренування моделей штучного інтелекту має дві різні імплементації – перша створена для машинного навчання на Python, та друга, яка була створена зовсім нещодавно Tensorflow.js. Обидві ці версії використовуються для створення та тренування моделей штучного інтелекту. У нашому випадку було обрано Tenserflow.js оскільки він дозволяє працювати зі штучним інтелектом у оточенні Node.js [10].

Node.js – це міжплатформне середовище виконання язика програмування Javascript. Він дозволяє розробникам використовувати Javascript не лише для роботи у браузері, а й для створення серверної частини. Також Node.js дозволяє розробникам веб-додатків ефективно працювати з асинхронними операціями.

Завдяки оточенню Node.js ми можемо використовувати Tenserflow.js не тільки на серверній частині, а й у самому браузері. Це дозволяє значно спростити вимоги до архітектури веб-додатку, та цілком відмовитися від серверної частини у випадках коли вона потрібна лише для тренування та взаємодії із моделлю штучного інтелекту.

Працювати у веб середовищі використовуя лише статичний HTML та CSS є досить архаїчною практикою, оскільки вона не надає гнучкості в плані архітектури веб-додатку. Саме тому було прийнято рішення використовувати React.

React досить часто називають фреймворком, не дивлячись на те, що самі розробники затверджують, що їх проект є лише бібліотекою яка надає можливість сучасним проектам самостійно обрати необхідну для них архітектуру. React також використовує JavaScript Syntax Extender, який дозволяє під час створення логіки для того чи іншого компоненту також вмістити в собі теги схожі на HTML та які під час рендеру на нього й перетворюються. Це дозволяє створювати «HTML у Javascript коді».

Однією з найкращих переваг React'у є модернізований DOM (Document Object Model) він являє собою представлення даних об'єктів, що складають структуру та зміст документа в Інтернеті. А разом з ним й життєвий цикл кожного з компонентів.

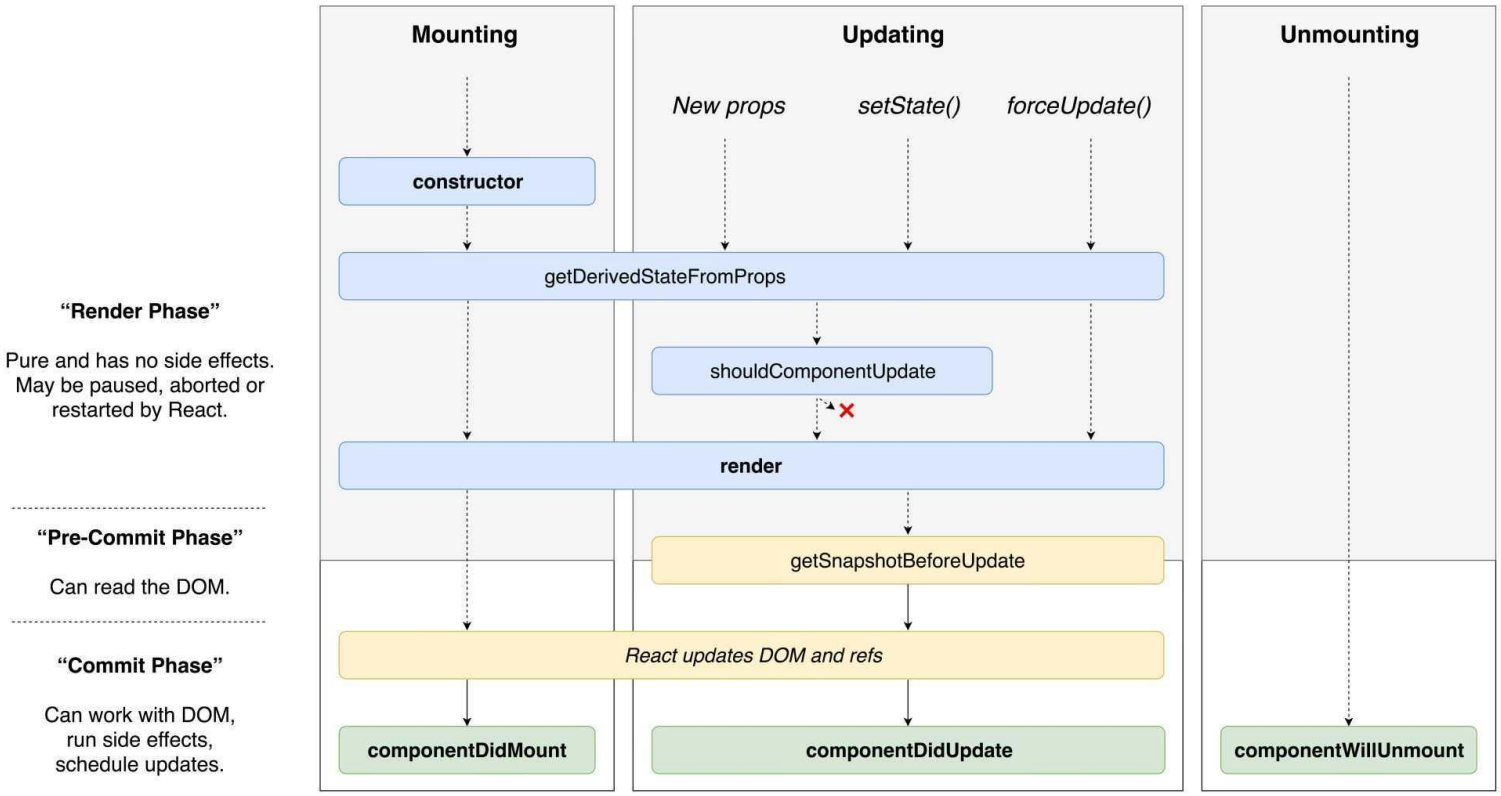


Рисунок 5.1 – Життєвий цикл React компоненту

Як можна побачити на рисунку 5.1 життєвий цикл кожного з React компонентів має три фази, та декілька методів життєвого циклу. Серед них найважливішими є componentDidMount, який викликається одразу після монтажу компоненту, тобто після того як користувач може своїми очами його побачити на веб-сторінці, та componentDidUpdate, який викликається після зміни стану компонента.

Проте описані вище методи життєвого циклу використовуються лише у класовому підході до створення компонентів. Новіші версії React дозволяють використовувати функціональні компоненти для виконання тих же задача, вони

мають набагато коротший синтаксис та дозволяють розробника обирати ООП, або функціональний підхід до створення веб-додатку.

У веб -додатку було використано функціональний підхід, оскільки JavaScript э більш функціональним язиком та тому, що ООП не дуже розвинене у ньому. Потрібно відмітити, що Microsoft створили надбудову над Javascript під назвою TypeScript, яка дозволяє повноцінно використовувати усі парадигми ООП, проте у даному веб-додатку вона не використовується.

## 5.2 Архітектура програмної системи

Даний веб-додаток використовує Flux архітектуру за допомогою імплементації її у бібліотеці Redux.

Flux - це архітектура додатків, яку Facebook використовує для створення веб-додатків на стороні клієнта. Вона доповнює компоненти подання React, використовуючи односпрямований потік даних. Це скоріше шаблон, а не формальний фреймворк, що дозволяє почати використовувати Flux без великої кількості додаткового коду. Саме завдяки цьому було створено бібліотеку Redux, яка за допомогою невеликої кількості коду дозволяє почати працювати із Flux архітектурою.

Як зображено на рисунку 5.2 Flux архітектура має декілька основних абстракцій, а саме:

- Action;
- View;
- Dispatcher;
- Store.

Action відображає події які відбуваються з інтерфейсом користувача в результаті дій самого користувача. Він має встановлений тип події та служить

переносником необхідних даних для Store. Проте сам Action не змінює Store, а лише переносить дані які необхідно змінити у Store, щоб відобразити зміни у View.

View є репрезентацією веб-сторінки, або компоненту який підписаний на зміни у Store.

Зазвичай у React'і розділяють два види компонентів, або View, розумні та дурні компоненти, різниця між ними у тому, що розумні компоненти використовуються лише для того, щоб слідкувати за логікою та її змінами у додатку, у той час як дурні компоненти використовуються лише для того, щоб відобразити ці зміни.

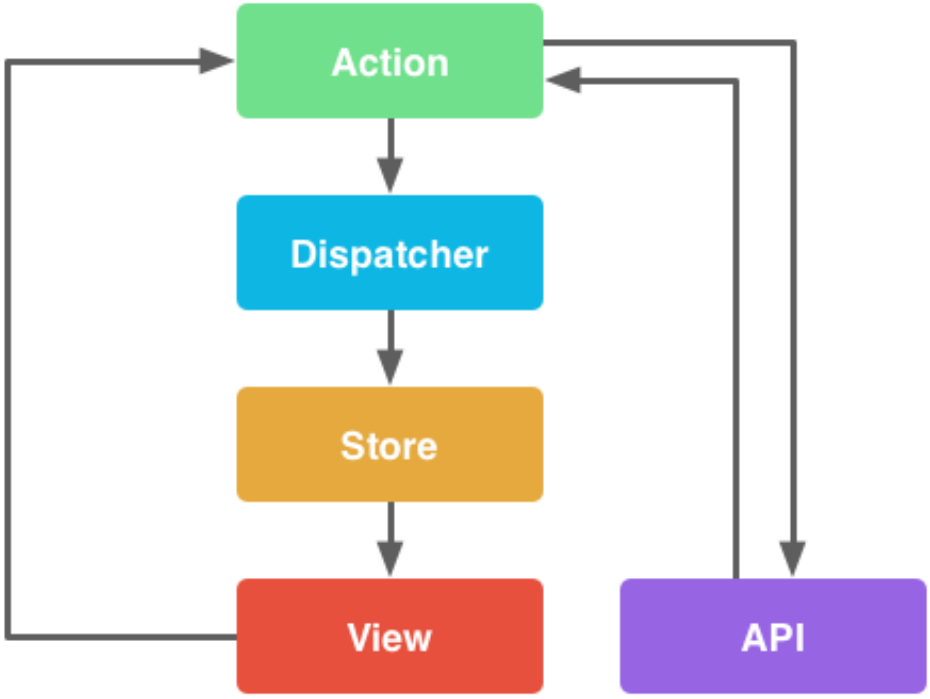


Рисунок 5.2 – Приклад Flux архітектури

Розумні компоненти також називають HOC, або Higher Oreder Component через те, що ці компоненти знаходяться вище у дереві DOM та керують дурними компонентами[11].

Усі дані проходять через диспетчер як центральний вузол. Action надаються Dispatcher методом створення дій і найчастіше походять від взаємодії користувача з представленнями.

Потім Dispatcher викликає зворотні виклики, які Store зареєстрував у ньому, надсилаючи дії у всі Store. У межах зареєстрованих зворотних викликів магазини реагують на те, які дії мають відношення до стану, який вони підтримують.

Store видають подію зміни, щоб попереджати подання контролера про те, що відбулася зміна рівня даних.

Контроллеры прослуховують ці події та отримують дані з Store у обробнику подій. View контролери викликають власний метод setState(), викликаючи оновлення себе та всіх своїх нащадків у дереві компонентів.

Отже дані отримані з Action проходять скрізь центральний Dispatcher, єдиний елемент у Flux архітектурі, який має право змінювати Store, у той час як сам Store лише зберігає у собі інформацію про нинішній стан додатку.

Кожна з цих абстракцій також імплементована й у Redux, проте у той час як Flux надає можливість направляти потік даних одно Higher Order Component, Redux йде далі та пропонує налаштувати такий самий потік даних в усьому веб-додатку.

Також додаток використовує один з найновіших стандартів Javascript під назвою ES6. Цей новий стандарт привносить багато цікавого та корисного функціоналу у Javascript, який дозволяє покращити та прискорити розробку веб додатків, проте не весь новий функціонал ще підтримується браузерами, тож необхідно обрати поліфіл, який дозволить їх використовувати.

Поліфіл – це код, який реалізує функцію у веб-браузерах, які не підтримують цю функцію. Найчастіше це стосується бібліотеки JavaScript, яка реалізує веб-стандарт HTML5 або CSS, або встановлений стандарт (підтримується деякими браузерами) у старих браузерах, або запропонований стандарт (не підтримується жодним браузером) у існуючих браузерах.

У нашому випадку для цього буде використано Babel - набір інструментів, який в основному використовується для перетворення коду ECMAScript 2015+ у сумісну з попередніми версію JavaScript у поточних та старих браузерах чи середовищах[12]. Окрім цього Babel також здатний конвертувати JSX та типові анотації від Typescript.

Окрім цього необхідно обрати бандлер який буде збирати весь вихідний код та слідкувати за його залежностями, для цього було обрано найпопулярніший бандлер під назвою Webpack.

## 5.3 Інтерфейс користувача

Інтерфейс користувача – це засіб, за допомогою якого людина керує програмним додатком або апаратним пристроєм. Хороший користувальницький інтерфейс забезпечує "зручний" досвід, що дозволяє користувачеві взаємодіяти з програмним чи апаратним забезпеченням природним та інтуїтивно зрозумілим способом.

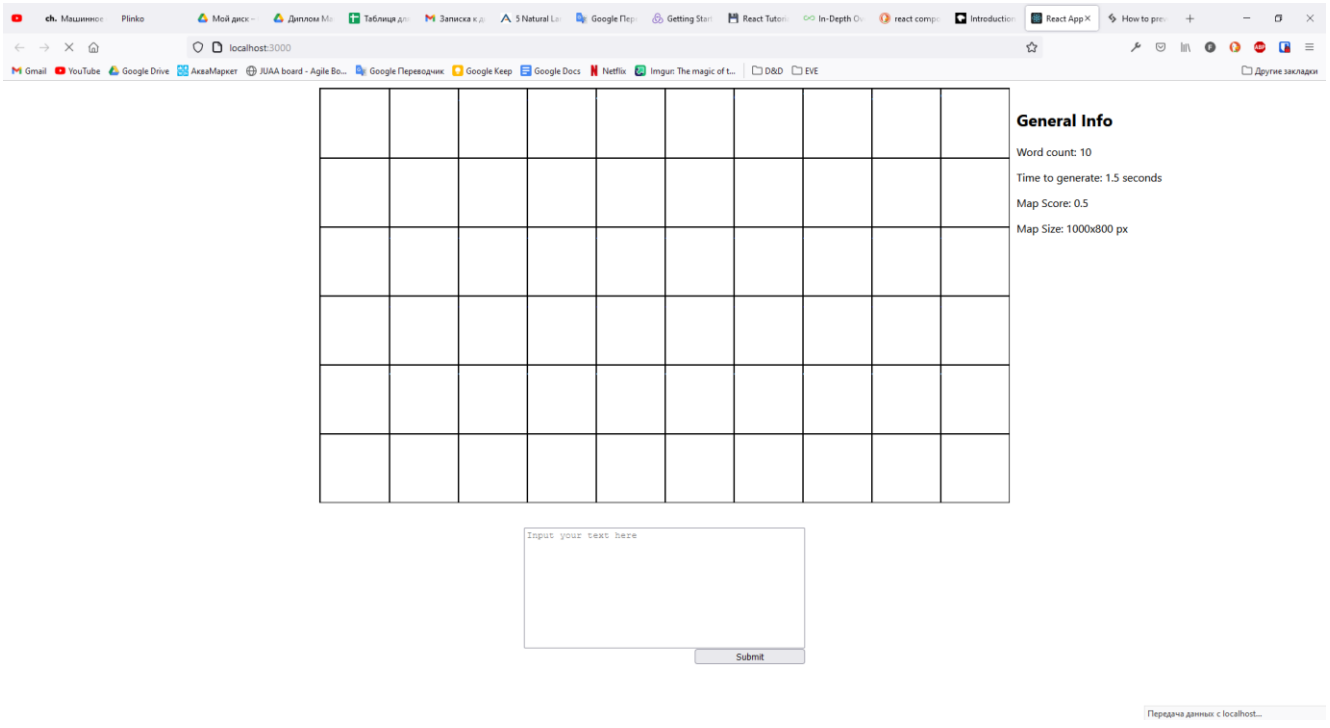


Рисунок 5.3 – Макет інтерфейсу користувача

Для веб-розробників інтерфейс користувача настільки ж важливий, як й логіка, що схована за ним.

Використання React в даній роботі дозволяє створювати компоненти, або елементи інтерфейс один раз та перевикористовувати їх коли це необхідно.

Також React змінює загальний метод створення веб-сторінок за допомогою свого кастомного DOM. Це, насамперед, дозволяє створювати весь веб-додаток на одній сторінці. Дуже важливо пам’ятати, що не важно на якій частині сайту знаходиться користувач він, на сам перед, на тій же сторінці де він почав свою подорож.

Таблиця 5.1 – порівняння CSS з SCSS

| Технологія | CSS | SCSS |
|---|---|---|
| Глобальні змінні | Ні | Так |
| Підтримка усіх css версій | Так | Так |
| Необхідна кількість коду | Більше | Менше |
| Синтаксис «гнездування» | Ні | Так |

Такий підхід «перемальовування» сторінки дозволяє веб-додаткам створеним на React завантажуватися вдвічі швидше ніж звичайним веб-сайтам.

Оскільки даний додаток знаходиться у веб просторі, то найефективнішим способом створення інтерфейсу є використання HTML та CSS. Проте використовувати звичайний CSS, хоч і можливо, не є оптимальним методом створення веб-додатків у сучасності. Саме тому для створення юзер інтерфейсу було використано Syntactically Awesome Style Sheet, або SCSS.

SCSS є надстройкою над звичайним CSS, що розширяє його звичайні функції, SCSS був розроблений Хемптоном Кетліном і розроблений Крісом Еппштейном та Наталі Вайценбаум.

Як можна побачити з таблиці 5.1 SCSS має декілька значних переваг перед класичним CSS. По перш за все SCSS підтримує абсолютно усі версії класичного CSS, а отже він може бути використаний у будь-якому проекті.

По-друге SCSS дозволяє створювати глобальні змінні, та розміщати їх віддільно від файлу у якому вони використовуються, це значно зменшує кількість

коди який необхідно написати, оскільки розробник не повинен копіювати усі кольори та типи шрифтів з одно файлу до іншого.

По-третє загальна кількість CSS коду значно змішується завдяки підтримці синтаксису «гніздування», тобто не потрібно описувати класи під кожен елемент на сторінці, якщо розробнику необхідно змінити стилі одного з дочірніх елементів.

Та обрати лише компілятор CSS зазвичай недостатньо, оскільки у більшості проектів організація коду вирішує швидкість я якою буде розроблюватись проект, та наскільки легко його буде підтримувати у майбутньому

Методологія CSS - це набір керівних принципів для написання модульного, багаторазового та масштабованого коду. Незважаючи на те, що CSS - це проста мова для написання, без узгодженої конвенції код стає безладним майже так само швидко, як і пишеться. Оскільки кожна декларація CSS визначається в окремому рядку, файли швидко отримують величезні обсяги, що робить їх важкими для обслуговування.

. На даний момент є багато різних методологій використання, деякі з них, разом з перевагами, які вони додають до проекту перелічені у таблиці 5.2.

Таблиця 5.1 – методологій CSS

| Методологія | Розбиття на об’єкти | Розбиття на дрібні елементи | Класифікація елементів | Розбиття на модифікатори |
|---|---|---|---|---|
| BEM | Ні | Так | Так | Так |
| OOCSS | Так | Так | Так | Ні |
| SMACSS | Ні | Так | Ні | Ні |
| SUITCSS | Ні | Так | Так | Ні |
| Atomic | Ні | Так | Ні | Ні |

Для створення та підтримки довгострокових та швидко зростаючих проектів найбільш за все підходить методологія BEM. Blocks, Elements and Modifiers (BEM) є назвою методології[13].

Вона пропонує розбивати усі CSS класи на Блоки, Елементи та Модифікатори. Блоками є, наприклад логін форма, вона матиме такі елементи як інпути та кнопки. Модифікаторами у такому випадку будуть стилі які відповідають за різний візуальний стан в якому находяться елементи. Наприклад у разі помилки верифікації інпут матиме стан помилки та до нього буде присвоєно модифікатор, який змінить колір його рамки на червоний, що дасть користувачеві візуальний фідбек.

Кожен юзер інтерфейс повинен слідкувати декільком правилам, які допоможуть створити не тільки гарний на вигляд інтерфейс, а й зручний User Experience, а саме:

- Інтерфейс користувача повинен бути якомога простіше.
- Інтерфейс користувача повинен бути узгодженим з рештою елементів
- Інтерфейс користувача повинен передавати нинішній стан додатку користувачеві
- Інтерфейс користувач повинен створювати ієрархію і розуміння за допомогою типографії

Усі ці пункти дозволять створити інтерфейс користувача, який інтуїтивно зрозумілий, легкий в навігації та який направляє погляд користувача у необхідну область. Користуючись цими пунктами було створено макет інтерфейсу користувача, який можна побачити на рисунку 5.3.

Цей макет інтерфейсу має усі необхідні для початку дослідження елементи. Більша частина вільного місця по середині зайняти контейнером, який розчерчено сіткою. Ця сітка відображає клітинки у мапах D&D та використовується для рахування кількості ходів, яку може зробити гравець, її присутність дуже важлива для гравців та данжон мастерів, під цю розкреслену сітку буде накладено зображення текстури землі, та зображення які відповідають предметам знайденим у наданому тексті.

Для вводу тексту було додано HTML елемент під назвою Textarea, він дозволяє вводити необмежену кількість символів, саме цей елемент буде

використовуватись для отримання тексту від користувача, у той час як результат обробки тексту буде відображено у розділу для мапи над цим елементом.

Цей Textarea було розміщено у елементі Form, який дозволяє отримувати введенні користувачем данні після натискання кнопки Submit. Ця форма також буде використана у майбутньому для налаштування різних інших елементів, які впливають на генерацію мапи.

Також додатково було розташовано декілька полів, що відображають інформацію, яка може знадобитись для проведення експерименту, а саме:

– кількість слів у тексті;
– час, що знадобився для генерування мапи;
– оцінка мапи;
– розмір мапи.

Розмір мапи на даний момент є статичним, усі отримані мапи будуть мати розмір в 1000 пікселів у ширину та 800 пікселів у висоту. Розмір сітки також є статичним, кожне креслення на сітці має розмір в 100 пікселів на 100 пікселів. У майбутніх ітераціях проекту планується перетворити цю статику на змінні, що дозволить провидити більш обширні експерименти.

Також на макеті зображено кнопку яка відповідає за скачування створеної мапи, натиснув її користувач отримає копію згенерованої мапи на свій девайс.

## 5.4 Майбутні ітерації

У майбутніх ітераціях продукту планується додати декілька елементів інтерфейсу разом з новим фунціоналом. Одними з найважливіших нових елементів інтерфейсу є повзунок з інпут полем які дозволять користувачеві обрати такі параметри як розмір сітки та її позицію.

Також вільне місце з лівої сторони макету буде використано, як сайдбар меню. У цьому меню буде зображено декілька елементів, які відображають тип

мапи, що буде згенеровано. Це надасть користувачеві обрати один з існуючих пресетів для генерації мапи. Або якщо користувач вже згенерував мапу та не задоволений обраним штучним інтелектом шаблоном тайлів та заднього фону, цей же сайдбар дозволить йому обрати інший шаблон.

Окрім цього найбільшим покращенням нинішньої системи буде надання користувачеві можливості виправити помилки штучного інтелекту за допомогою Drag & Drop функціоналу на згенерованій мапі.

Останнім етапом розвитку проекту є надання користувачеві повного контролю над заднім фоном та шаблоном тайлів, тобто користувач зможе власноруч розташовувати зображення на мапі до чи після того як вона буде згенерована.

# ВИСНОВКИ

Під час створення кваліфікаційною роботи було спроектовано та створено веб-додаток генерування мап для настільної гри D&D. Цей веб-додаток працює у будь-якому сучасному браузері. Розроблений веб-додаток є інструментом який отримує опис кімнати або локації та основі отриманого тексту створює картинку у форматі .png або .jpg для зберігання користувачем.

Для імплементації даного проекту було використано Tenserflow.js - новіша розробка Google яка дозволяє створювати та тренувати моделі штучного інтелекту у браузері та React для взаємодії за браузером.

Було обрано та імплементовано Flux архітектуру за допомогою Redux. Веб-додаток використовує нові можливості EC6, які добавляються у браузер за допомогою поліфіла під назвою Babel.

Також Tenserflow.js дозволяє конвертувати Python моделі для роботи з ними у браузері чи з Node.js. Tenserflow.js, а також надає можливість використовувати декілька заздалегідь натренованих моделей штучного інтелекту. Такий підхід дозволяє поширити використання штучного інтелекту та робить його більш доступним як для розробників так й для користувачів. Сам штучний інтелект використовує методи обробки природної мови для обробки тексту.

Було спроектовано та створено юзер інтерфейс який надає максимально комфортний юзер експірієнс та дозволяє ефективно збирати необхідні данні про поставленні експерименти.

Отримавши результати та провівши аналіз можна прийти до висновку, що системи обробки придонної мови дозволяють досить ефективно генерувати мапи для настільної гри Dungeons & Dragons.

# ПЕРЕЛІК ПОСИЛАННЯ

1. Dungeon & Dragons/ Материал из Википедии – свободной энциклопедии. URL: https://en.wikipedia.org/wiki/Dungeons_&_Dragons (дата звернення: 15.02.2021)

2. React - A JavaScript library for building user interfaces URL: React - A JavaScript library for building user interfaces (дата звернення: 13.04.2021)

3. Redux - A predictable state container for JavaScript apps URL: https://redux.js.org/introduction/getting-started (дата звернення: 01.04.2021)

4. Штучний інтелект / Материал из Википедии – свободной энциклопедии. URL: https://ru.wikipedia.org/wiki/Штучний_інтелект (дата звернення: 11.02.2021)

5. Обработка естественного языка/ Материал из Википедии – свободной энциклопедии. URL: https://ru.wikipedia.org/wiki/Обработка_естественного_языка (дата звернення: 11.03.2021)

6. Tenserflow documentation. URL: https://docs.unity3d.com/Manual/index.html (дата звернення: 18.01.2021)

7. Keras vs Tenserflow – Which Should You Choose? URL: https://www.guru99.com/tensorflow-vs-keras.html#5/ (дата звернення 21.02.2021)

8.Best Frameworks and Libraries for AI. URL: https://dzone.com/articles/progressive-tools10-best-frameworks-and-libraries (дата звернення: 13.02.2021)

9. Smelyakov K., Chupryna A., Karachevtsev D., Kulemza D., Samoilenko Y., Patlan O. Effectiveness of Preprocessing Algorithms for Natural Language Processing Applications // 2020 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 6-9 Oct. 2020, Kharkiv, Ukraine. – P. 1-5.

10. Назаров О. С., Грінько К.О. Обучение Модели Искусственного интеллекта в Веб Окружении. European Scientific Discussions, Rome, Italy. – 25-27 April 2021.

11. Start using Flux architecture now. URL: https://facebook.github.io/flux/docs/in-depth-overview (дата звернення: 28.03.2021)

12. Babel - The compiler for next generation JavaScript. URL: https://babeljs.io/docs/en/ (дата звернення: 05.04.2021)

13. Introduction to using BEM. URL: http://getbem.com/introduction/ (дата звернення: 22.03.2021)